Applied and Computational Harmonic Analysis 77 (2025) 101763

Contents lists available at ScienceDirect

Applied and Computational Harmonic Analysis

journal homepage: www.elsevier.com/locate/acha

Full Length Article

Mathematical algorithm design for deep learning under societal and judicial constraints: The algorithmic transparency requirement

Holger Boche [a,b,c,d,*], Adalbert Fono [e], Gitta Kutyniok [e,f,g,h]

[a] Institute of Theoretical Information Technology, TUM School of Computation, Information and Technology, Technical University of Munich, Germany
[b] Munich Center for Quantum Science and Technology (MCQST), Munich, Germany
[c] Munich Quantum Valley (MQV), Munich, Germany
[d] BMBF Research Hub 6G-life, Munich, Germany
[e] Department of Mathematics, Ludwig-Maximilians-Universität München, Germany
[f] Department of Physics and Technology, University of Tromsø, Norway
[g] Munich Center for Machine Learning (MCML), Munich, Germany
[h] DRL-German Aerospace Center, Institute of Robotics and Mechatronics, Germany

ARTICLE INFO

Communicated by Joan Bruna

Keywords:
Deep learning
Trustworthiness
Algorithmic transparency
Turing machines
Blum-Shub-Smale machines

ABSTRACT

Deep learning still has drawbacks regarding trustworthiness, which describes a comprehensible, fair, safe, and reliable method. To mitigate the potential risk of AI, clear obligations associated with trustworthiness have been proposed via regulatory guidelines, e.g., in the European AI Act. Therefore, a central question is to what extent trustworthy deep learning can be realized. Establishing the described properties constituting trustworthiness requires that the factors influencing an algorithmic computation can be retraced, i.e., the algorithmic implementation is transparent. Motivated by the observation that the current evolution of deep learning models necessitates a change in computing technology, we derive a mathematical framework that enables us to analyze whether a transparent implementation in a computing model is feasible. The core idea is to formalize and subsequently relate the properties of a transparent algorithmic implementation to the mathematical model of the computing platform, thereby establishing verifiable criteria.

We exemplarily apply our trustworthiness framework to analyze deep learning approaches for inverse problems in digital and analog computing models represented by Turing and Blum-Shub-Smale machines, respectively. Based on previous results, we find that Blum-Shub-Smale machines have the potential to establish trustworthy solvers for inverse problems under fairly general conditions, whereas Turing machines cannot guarantee trustworthiness to the same degree.

1. Introduction

The core idea of machine learning, that is, to enable an algorithm to extract relevant information from an available data set to solve a given problem, coupled with an evolution of digital computing technology and power, led to a revolution in a wide range

* Corresponding author.
E-mail address: boche@tum.de (H. Boche).

https://doi.org/10.1016/j.acha.2025.101763
Received 25 January 2024; Received in revised form 19 October 2024; Accepted 18 March 2025

Available online 24 March 2025

of applications [1–3]. Even more, by further augmenting the machine learning models and the data sets, great advances have been made via the deep learning framework in fields such as natural language processing, which were expected to be amenable to this approach to a lesser degree due to their inherent complexity [4,5]. Deep learning [6–8] refers to a specific class of machine learning models, so-called (deep) artificial neural networks [9] that are adjusted and optimized on given training data via fairly simple loss functions and basic iterative methods such as stochastic gradient descent along with backpropagation [10].

Therefore, it is widely acknowledged that the success of deep learning can be attributed mainly to three pillars. First, the availability of vast amounts of data enabled the breakthrough of the deep learning approach by outperforming previous methods by a large margin [11]. Second, the advancements in digital computing hardware and the accompanying increase in computational power allowed for the effective processing of large data sets in the training phase resulting in larger and deeper networks that tend to be more capable. Hereby, the initial breakthrough relied on incorporating GPUs in the training of neural networks. By construction, GPUs are better equipped than purely CPU-powered computers to carry out the applied algorithms heavily depending on matrix multiplication operations [11,12]. The ongoing process of increasing training data sets and computing power cumulated at this stage in digital high-performance computing approaches optimized for implementing deep learning [13–15]. Third, the progress in neural network architecture from fully-connected feedforward over convolutional [11] and residual networks [16] to transformers [17] as well as in training methodology, e.g., incorporating techniques such as self-supervision [18] and reinforcement learning (with human feedback) [19], transferred the potential benefits of larger training sets and more computing power into practical improvements.

1.1. Energy and scaling limitations of deep learning

However, it is unclear how far the current approaches can be further scaled and improved under the deep learning framework. Indications suggest that this development may slow down or even halt [20,21]. The data sets employed to train state-of-the-art large language models already include a noteworthy fraction of (English) text on the internet [4]. Hence, even larger and more suitable databases need to be generated to train future models by combining different data types such as text, audio, and video. Besides, the ongoing digitization of the physical world via sensors, which observe and perceive aspects of their environment – think of autonomous vehicles for instance –, leads to an accumulation of additional data but also greater demands in storing and throughput capacity [22].

Therefore, to process the collected data and to apply algorithmic methods such as deep learning more computing capacity is required, i.e., the necessary number of computational steps increases. Since at present there exists a direct connection between the number of computational operations and the total energy consumption of a (digital) computing device [23,24], it seems unlikely that the already immense energy consumption of the current deep learning models does not substantially increase in the future unless the applied techniques are structurally adjusted. Hence, dramatically more data and energy-efficient methods have to be incorporated or the underlying computing and processing paradigm needs to change so that more efficient but equally powerful computations can be carried out. One promising alternative to the present, purely digital computing approach is incorporating analog devices in the computing pipeline since analog computing offers potential benefits if its fault tolerance can be increased [22,25].

1.2. The need for trustworthy deep learning

Due to its ongoing advancements, the scope of tasks that deep learning models can successfully tackle is ever-increasing. On the one hand, existing models are tweaked to adapt to similar tasks of the same complexity. On the other hand, new, more capable model types are introduced – the last one being foundational models [26] such as large language models – that can solve previously unattainable problems. Mainly, current models still impact their environment indirectly by influencing human decisions but not by direct control of the physical environment. Large language models are a prime example of these interactions [27]. However, the increasing capabilities of deep learning models and the actual goal of artificial general intelligence [28,29] indicate that the type of interactions may change soon, e.g., autonomous vehicles with their sensing and decision-making powered by deep learning will act as physical agents and thereby cross the barrier from indirect to direct interactions with the physical environment. This is reflected by discussions on machine ethics in the context of autonomous driving, i.e., the questions of how algorithms should decide in certain situations [30–32]. While formulating and agreeing on a decision-making framework for (autonomous) physical agents is undeniably important, the question of how to implement and guarantee abidance by the chosen framework is equally relevant.

The latter goal is complicated by the black box, unreliable, non-accountable, and non-robust behavior of current deep learning models [33–37]. A well-known failure in this regard is the non-robustness of artificial neural networks towards minimal input perturbations, which entails non-reliability on the network's output [34,38–42]. These drawbacks can be summarized as a lack of trustworthiness [43–47] – an umbrella term for privacy, security, resilience, reliability, and accountability [48]. Thereby, the notion of trustworthiness includes amongst others aspects such as

- robustness, i.e., resilience against a variety of challenges: changing environments or situations, noisy or incomplete data, and adversarial attacks;
- transparency and interpretability, i.e., clear justification and explanation of the decision-making process;
- fairness, ethical compliance, and privacy, i.e., avoidance of biases, equitable treatment of diverse user groups, and secure management of sensitive information;
- safety and security, i.e., protection against potential threats and preventing unintended harmful consequences.

Failing to establish trustworthiness in deep learning systems entails that no performance guarantees can be provided or, at least, circumstances may arise in which deep learning systems exhibit unexpected and potentially harmful behavior. This fact is well-acknowledged, even beyond the science community, since the present and future deep learning applications are expected to impact all of society. Therefore, policymakers already proposed guidelines and regulations that deep learning models need to satisfy. Among the most influential ones are certainly the European AI Act [49] and the G7 Hiroshima Leaders Communiqué [50], which describe various degrees of requirements and demands concerning the trustworthiness of deep learning systems. In particular, the European AI Act formulates a clear legal framework that might act as a blueprint for further regulation proposals. This raises the question of the extent to which trustworthiness can be achieved with the deep learning approach. The lack of trustworthiness in deep learning has persisted throughout its evolution since the fundamental approach remained unchanged. This is in contrast to other AI methods such as expert systems [51], which by design offer trustworthiness benefits, but fail to reach the capabilities provided by deep learning in other areas. Thus, an open problem is whether adapting key components of deep learning may change the trajectory of trustworthiness.

1.3. Our contributions

Since remedying energy concerns may require introducing and integrating novel computing technologies, we assess theoretical computational requirements to establish trustworthy deep learning models. For this purpose, different angles need to be considered.

First, trustworthiness lacks a universally acknowledged formal definition. Therefore, in Subsection 2.1 abstract principles and potential legal structures based on the introduced aspects of trustworthiness are discussed. Thereby, clear requirements on trustworthiness with a focus on the transparency condition are posed.

Second, the capabilities and limits of algorithms with deep learning models being a specific type can only be evaluated with respect to the hardware/computing paradigm. Consequently, we need to take into account the underlying computing model. To that end, in Subsection 2.2 we present two different computing models – digital and analog – and their respective promises for energy-efficient performance. By considering idealized mathematical abstractions illustrating the core idea of the computing approaches, we describe conditions that guarantee trustworthy deep learning implementations, i.e., we convert the introduced non-formal trustworthiness conditions into a mathematical framework.

Third, trustworthiness in deep learning is a broad topic. Hence, we restrict to a particular use case – inverse problems – which is specific enough to allow for a formal treatment but potentially enables us to draw more general conclusions. We define the inverse problem setting and the associated deep learning solvers in Section 3.

Subsequently, Section 4 applies the derived framework of trustworthy computations in the inverse problem use case. The basis of our analysis are [52] and [53], which treat the algorithmic solvability of inverse problems on idealized digital and analog hardware, respectively. We present and compare their respective findings and subsequently embed them in our framework. The results imply that digital and analog computing have diverging capabilities to enable trustworthy algorithms (and thereby deep learning systems): Digital computing modeled by Turing machines [54] has certain limitations that potentially can be avoided by analog computing modeled by Blum-Shub-Smale (BSS) machines [55].

1.4. Limitations

Can we transfer the observations in our framework about trustworthy algorithms from the field of inverse problems to a broader class of deep learning applications? Although each application requires in-depth consideration of its own, we try to motivate some general conclusions. The existence of a trustworthy algorithm solving a task may depend on the underlying computing model and different outcomes may indeed arise. In theory, analog computing may enhance the capacity for achieving trustworthiness and overcome some limitations arising on digital hardware. This pattern is especially relevant for tasks involving real-world physical processes or, more generally, tasks modeled and represented on continuous domains. The integrity between the intrinsically discrete digital computing model and the continuous problem description may be lost in these cases. In this sense, inverse problems are only a – well-studied and well-behaved – representative example of a relevant class of problems tackled by deep learning. A decisive question is, whether analog computing that translates theoretical into practical benefits can be realized. We analyzed analog computing under the BSS model, but there is no universally accepted mathematical model precisely formalizing analog computations (as with the Turing model for digital computations). Hence, further research is required to establish appropriate theoretical models that include, for instance, error correction and approximate analog computing, i.e., computing models that can trade energy and computing time with accuracy (presumably how biological brains operate) [56–59].

The potential of a trustworthy algorithmic computation is evaluated via the notion of algorithmic solvability, which describes a correct, reliable, and accountable method to solve a given problem; see Section 4. Thus, algorithmic solvability may also provide a basis for verifying the abidance of legal requirements, in particular in the field of deep learning. We demonstrate that algorithmic solvability of inverse problems strongly depends on the specific problem formulation so that it may guide us toward descriptions that in principle allow for trustworthy solutions. However, inverse problems are a particular and restricted setting, e.g., the forward operator, which essentially contains the relevant information about the task and defines the ground truth, is known. Hence, inverse problems may neither represent a complex (and to some degree chaotic) real-world setting nor a typical deep learning task, where the solution approach crucially depends on a learning framework. Therefore, the conclusions derived in the inverse problem use case may not be transferable to other settings- or at least need to be further substantiated. Moreover, intricate and diverse real-world tasks that require advanced and rather general solution techniques may not be amenable to the notion of algorithmic solvability.

Either the generality of the tasks prevents algorithmic solvability directly or situations may arise in which decisions have to be made under incomplete or uncertain information so that no clear assignment of right and wrong behavior is feasible. The latter cases would be difficult to transfer to the introduced formalism and consequently, different tools may be necessary to derive statements about trustworthy algorithmic solutions. Nevertheless, the observation that even in a well-behaved setting like inverse problems limitations towards trustworthy algorithmic solutions on digital hardware exist suggests a generally occurring feature also affecting the previously described scenarios.

Algorithmic solvability is certainly not the only approach to gain a better understanding, but other methods to determine the trustworthiness of algorithms have been proposed; we'll revisit this topic and discuss further approaches Subsection 1.5. The utilized notion of algorithmic solvability and the underlying computability concepts foremost guarantee the (technological) integrity of a system [48], which refers to a state in which the system in question resides within its specified margin of operation. Thus, the utilized computing machine does not (inadvertently) interfere with and influence the expected outcome of the performed computation. Therefore, adherence to a framework provided by legal regulations may be ensured via algorithmic solvability. Yet, in principle, using AI systems is feasible without adhering to algorithmic solvability if the arising drawbacks are acknowledged. In addition, if human intelligence is understood as an algorithm, it presumably may not fully guarantee algorithmic solvability and associated trustworthiness properties since it is subject to instabilities such as cognitive biases. As a result, a scenario is conceivable in which AI systems prevail without ensuring the aforementioned principles, for example by leading to better results on average than purely human intelligence. A concrete example is autonomous driving, which could prevail when the expectation is reached that autonomous vehicles could improve accident statistics. Nevertheless, the outlined scenario exhibits a severe lack of trustworthiness, which may cause liability issues in case of accidents and ultimately may prevent its occurrence.

1.5. Related work

The need to understand if an algorithmic method behaves in an intended manner is not new but has gained traction in recent years due to the rise of deep learning. The key innovation of deep learning that raises the complexity of this process is the learning part, which does not prescribe a fixed approach to tackle a problem but aims to extract a solution based on collected data about the problem. This contrasts with classical software, which ideally implements a provably correct method to solve a given task so inconsistencies only arise at the implementation level, i.e., development and design. The difficulty in avoiding inconsistencies and errors lies in software systems' complexity, dependencies, and interconnectedness. Hence, proving the correctness of a software system is an intricate but in principle viable task in the sense of deductive verification/theorem proving. However, an automated verification process is desirable to manage resources effectively yet not fully realizable in practice (on digital hardware) as Turing's initial work on the halting problem and the generalizations by Rice showed [54,60]. Therefore, various methods have been proposed, representing trade-offs on a spectrum from efficient to expressive. The most efficient approach is simple testing, i.e., trying to falsify the system on specific instances, whereas model checking at the cost of higher computational resources aims to verify the model of a system against its formal specification [61]. In general, even the most expressive (automated) methods do not guarantee correctness if errors are not found but provide an elevated degree of confidence in the system.

The heightened demands in deep learning were initially not accompanied by newly developed validation methods – justified by the successes benchmarking by simple testing achieved. Nevertheless, problematic properties such as adversarial examples [34] indicating a mismatch between intended and actual behavior were discovered. These were either met with slightly adjusted learning procedures, e.g., adversarial training [62,63], to avoid the occurrence of the undesired properties or with explainability techniques aiming to make the (decision) process of deep learning models comprehensible for the user [64–68]. The proposed methods succeeded only partially and the (to a certain degree) negligible impact of the described mismatch in narrow use cases, now poses a significant challenge where deep-learning-based models tackle multi-task problems and are expected to power autonomous agents [26]. Hence, there is a need for appropriate benchmarking that goes beyond straightforward testing and guarantees 'good' behavior in complex real-world environments. Instead of establishing suitable measures that support the crucial step from benchmarks to real-world environments, the idea of deep learning models interpretable by design has been proposed [69]. The motivation is that it may not be feasible to assess certain requested properties after the development of the model but they can be taken into account in the design phase to enable their later occurrence [69]. Thus, the main innovation here is to develop deep learning models that are explicitly constructed with certain interpretability goals in mind (and possibly iteratively improved by incorporating the insights from specifically constructed tests and the mentioned explainability methods) [70,71].

Another blindspot of standard benchmarking based on testing are ethical and moral considerations. Deep learning models have displayed biased and unfair decision-making and also privacy concerns remain an issue [72–74]. To mitigate these drawbacks technical adjustments have been proposed [75,76] but how to cope with them from an ethical or legal point of view is still unresolved [77–81]. Overall, the degree to which trustworthiness is required, i.e., the trust that a deep learning model acts in an intended manner (and avoids the mentioned technical and ethical pitfalls), depends on the circumstances. Safety-critical scenarios (possibly including autonomous agents) certainly require elevated trustworthiness standards ideally accompanied by some certification [82–84]. The main idea of this work is to establish a 'trustworthy by design' framework that precedes both classical and specific deep learning validation methods for safety-critical applications. This is achieved by providing an exact mathematical formulation of algorithmic transparency, which subsequently allows us to analyze questions of the form: Does the combined system of a given computing platform with a suitable algorithm represent a transparent algorithmic solution of a problem (and does such a solution exist under certain conditions)?

1.6. Potential impact and extensions

In this paper, we propose a mathematical framework for algorithmic trustworthiness considerations based on computability theory, i.e., an approach to assess the possibility of a transparent algorithmic implementation of a problem given a computing model. Although it is not a universally applicable framework, it provides a step forward in the trustworthiness analysis of deep learning in the following sense: On the one hand, it allows us to assess whether transparency may in principle be attainable in a given scenario. Hence, further analysis can evaluate a potential trustworthy implementation or the seriousness of the lack of trustworthiness in a specific scenario needs to be considered. On the other hand, a trustworthiness analysis may steer the implementation of a problem by adapting associated parameters such as computing platform, expected accuracy, and generality so that a potentially trustworthy implementation becomes feasible. This approach can be used in a detail-oriented setting for a specific task as well as on a larger scale that asks for promising directions for establishing trustworthiness in a broader scope. In this sense, the present work may act as a starting point for future work studying a broader range of topics related to deep learning going beyond the inverse problem use case.

Concerning inverse problems, our analysis indicates that a trustworthy solver either presupposes a sufficiently narrow problem description or if a more general solver is envisioned, then computing capacities beyond digital computing may be necessary or certain limits need to be accepted, i.e., a provable trustworthiness certificate in the introduced framework is not feasible. Thus, our analysis enforces the observation that a shift from purely digital information processing (and especially computing) to novel approaches also comprising analog techniques seems inevitable – not only due to demands on energy efficiency and data throughput [22] but also from the trustworthiness perspective. An emerging example of the changing paradigm is given by neuromorphic computing [85] with promising energy and processing gains, however, a trustworthiness analysis based on adequate computing models is still pending.

2. Trustworthiness framework

2.1. Societal and judicial requirements

Trustworthiness is a multifaceted property, where the individual features are partially overlapping and entail one another. Yet, it provides a comprehensive description of qualities potential guidelines may require. Hence, trustworthiness (and the implied abidance by an approved decision-making framework) can be seen as a prerequisite to implementing and operating deep leaning systems in certain scenarios, including safety-critical and high-leverage settings with direct influence on the physical environment such as autonomous driving. Due to a lack of technical assurances, abstract principles, codes of conduct, and legal regulations have been proposed such as Algorithmic Transparency, Algorithmic Accountability, and Right to Explanation for technology assessment in the context of trustworthiness [86,87]:

- Algorithmic Transparency (AgT) refers to the requirement of the factors determining the result of an algorithm-based decision being visible to legislators, operators, users, and other affected individuals.
- Algorithmic Accountability (AgA) refers to the question of which party, individual, or possibly system is to be held accountable for harm or losses resulting from algorithm-based decisions, particularly those that are deemed to be faulty.
- Right to Explanation (RtE) refers to the right of an individual who is affected by an algorithm-based decision to know the entirety of relevant factors and their specific expression that lead to the decision.

These notions may not capture all relevant nuances discussed in social, judicial, and political science. Still, they do guarantee or at least approach aspects of trustworthiness covered by robustness, interpretability, fairness, safety, etc. However, no widely accepted technological characterization in the form of standards and specifications exists at present. The need is accentuated by the fact that existing deep learning techniques do not result in models that satisfy AgT, AgA, and RtE and perform sufficiently well simultaneously. Therefore, the question is whether future methods can abide by these (potential) regulations, and if a positive answer is found to what degree. That is, can we expect future methods to solve trustworthiness issues in a broader context? In particular, are certain aspects that result in non-trustworthy behavior of deep learning models structurally inherent to the approach, or can suitable adaptations avoid them? We can make first strides in answering these questions by introducing formal (technical) requirements describing AgT, AgA, and RtE. Hereby, we focus on AgT since RtE can be considered as a direct application of AgT. Moreover, AgA is impossible without a clear understanding of the algorithm-based decision-making provided by AgT. Therefore, AgT is the backbone of the introduced trustworthiness notions.

An often neglected fact when discussing technological standards is the interplay between hardware platforms for computing, such as digital, neuromorphic, and quantum hardware, and the implemented algorithms representing the software side. Deep learning – in essence, just a specific algorithm class – needs to be expressed by a set of instructions in a (programming) language associated with the utilized hardware. Consequently, the capabilities of a given implementation also hinge on the power of the programming language, respectively the employed hardware. Thus, we analyze the potential impact of the hardware platform on trustworthy outcomes characterized by AgT based on two abstract computing models, namely the Turing model [54] for digital computing and the Blum-Shub-Smale (BSS) model [88] for (idealized) analog computing.

2.2. Transparency condition

An algorithmic computation of a problem ideally provides an explicit and reliable approach that is guaranteed to (or clearly describes the degree and the circumstances under which it) succeed(s). In mathematical terms, an algorithm is a set of instructions

that operate under the premises of some formal language characterizing the tackled problem. Thus, the specific definition of an algorithm depends on the considered formal language and the underlying computing model, e.g., digital and analog computations, which will be formally introduced in Subsection 2.2.1 and Subsection 2.2.2, respectively. A real-world physical problem can be translated into a mathematical model that describes its domain with feasible inputs, outputs, and operations of a potential algorithm. Hence, independent of the individual algorithmic steps, we can describe the abstract input-output relation characterized by a function on the identified domain that the algorithm needs to realize. Note that, in general, the domain of the algorithm may differ from the domain of the mathematical model of the physical problem; this behavior occurs, for example, in digital computing. By interpreting the mathematical model as a function describing the input-output relation of the problem, we can rely on the following notion of the realization of an algorithm tackling problems on continuous quantities described by real numbers.

Definition 2.1. Given a problem described by the input-output relation of a function $f : \mathbb{R}^m \to \mathbb{R}^n$, an *algorithm* $\mathcal{A}$ computing f *realizes a mapping* $\mathcal{A}_f : \mathbb{R}^m \to \mathbb{R}^n$ with $\mathcal{A}_f = f$.

Remark 2.2. The realization of an algorithm is particularly important for digital computing, where the computations are performed on representations of real numbers that constitute a specific subset of real numbers; see Subsection 2.2.1 for more details.

Subsequently, we will discuss the relevance of the realization of algorithms for AgT. The mathematical model (respectively the derived function) may act as the ground truth and serve as the foundation for the trustworthy requirements introduced in Subsection 2.1. In particular, the mathematical model provides a basis to assess AgT by identifying the factors influencing the underlying problem. Factors outside the mathematical model may not impact the algorithmic computation and outcome, since the transparency of the algorithmic computation can then no longer be guaranteed. Although these considerations may seem trivial, implementing the algorithm on a given computing platform, which is subject to mathematical modeling itself, adds another layer of complexity. To execute the algorithm on a suitable hardware platform, the abstract problem (respectively the corresponding mathematical model) needs to be translated into a machine-readable language. However, the mathematical model of the problem and the mathematical model of the computing platform do not necessarily agree. Thus, the input and output expressions must be unambiguously translatable between these two systems to guarantee proper implementation of the algorithm abiding by AgT.

To illustrate this issue consider the machine-readable description of a real number such as π in the digital computing model. Due to its infinite binary expansion, π can only be represented by finite algorithms that approximate it to any desired precision on digital computers. Despite the difference in description as a mathematical entity and a machine-readable object, we can identify both representations as equally valid and translatable, yet not unambiguous, since a real number may have multiple – equally valid – machine-readable descriptions. Besides, it is well-known that not all irrational numbers possess an unequivocal digital representation [54]. Consequently, the properties of the applied computing device need to be considered when the feasibility of an algorithmic computation as well as its trustworthiness is evaluated.

Definition 2.3. An *algorithmic implementation* is *transparent* in a given computing model if the realization $\mathcal{A}_f$ of some function $f : \mathbb{R}^m \to \mathbb{R}^n$ by an algorithm $\mathcal{A}$ is not altered by its implementation in the computing model. We then say that f allows for a *transparent algorithmic implementation* in the given computing model.

Remark 2.4. In general, if a real-world problem P can be expressed as a function f, then the input domain of f necessarily represents every factor determining the outcome of P. Hence, a transparent algorithmic implementation of a closed-form expression of f guarantees that P can be solved by an algorithm abiding AgT in the considered computing model since the realization of the algorithm solely relies on the factors constituting the problem. Consequently, the algorithmic decision-making can be retraced in principle to make providing AgT feasible. However, in deep learning a potential solver for P is sought based on a (imperfect) data set describing P so that f may not possess a closed-form expression or may be unknown, e.g., not all relevant factors determining P are identified. Thus, the contribution of the individual factors affecting the solver may not be apparent due to the black box behavior of deep learning, i.e., assessing AgT requires a successive analysis, which presupposes a transparent algorithmic implementation. Therefore, a transparent algorithmic implementation can be seen as a minimum requirement to obtain AgT; the actual conditions depend on the chosen computing model and its formalization of an algorithm. For instance, in the digital case transparency requires the algorithm to be independent of the specific representation of a real number.

Just constructing a transparent algorithm does not suffice, since besides being comprehensible we also expect the algorithm to deliver a solution to the considered problem. Therefore, it is important to maintain the integrity between the mathematical model of the problem and the applied algorithm, i.e., the algorithmic solution needs to unequivocally reflect the 'true' solution of the mathematical model.

Observation 2.5. To attain AgT and provide the correct output, an algorithm must reside within its specified margin of operation, i.e., the algorithm preserves the input-output relation of the underlying problem specified by a mathematical model (respectively the derived function).

In the subsequent analysis, our focus does not reside on the practical limitations of real-world hardware related to computation time, memory, and energy consumption. Via mathematical models of the considered computing paradigm – the most prominent one certainly being digital computations –, we study the possibility of theoretical guarantees for trustworthiness.

2.2.1. Digital computations

The concept of digital machines is encapsulated by the mathematical model of Turing machines [54]. The widely accepted Church-Turing Thesis [89] implies that Turing machines are a definitive model of digital computers, describing their (theoretical) capabilities perfectly. Thus, Turing machines provide a framework for analyzing digital computations by taking into account their inevitable approximate behavior concerning irrational numbers. More exactly, Turing machines introduce a notion of effective computations in a finite number of steps on real numbers. Thereby effectiveness refers to the condition that a Turing machine not only computes an approximate solution but also guarantees that the solution is within some previously prescribed error bound, which can be arbitrarily small. Hence, the reliability and correctness of an obtained algorithmic solution are guaranteed by design.

Next, we shortly formalize effective computations via *recursive functions* [90], which constitute a special subset of the set $\bigcup_{n=0}^{\infty}\{f : \mathbb{N}^n \hookrightarrow \mathbb{N}\}$, where '$\hookrightarrow$' denotes a partial mapping. Recursive functions coincide with the functions $f : \mathbb{N}^n \hookrightarrow \mathbb{N}$ that are computable by Turing machines, i.e., there exists a Turing machine that accepts input $x \in \mathbb{N}^n$ only if $f(x)$ is defined, and, upon acceptance, computes $f(x)$ [91].

Lemma 2.6. *A function $f : \mathbb{N}^n \hookrightarrow \mathbb{N}$ is a recursive function if and only if it is computable by a Turing machine.*

We can identify the set of effectively computable real numbers via recursive functions. By introducing a machine-readable description of real numbers, which was exemplarily demonstrated for π in Subsection 2.2 and can be formally expressed via recursive functions [90], one can construct sequences of rational numbers converging (with error control) to the considered real numbers. In this way, the computations performed by Turing machines are reduced to the rational domain, where exact computations are feasible and simultaneously the accumulated error due to the non-exact representation of real numbers is controlled.

Definition 2.7. A sequence $(r_k)_{k\in\mathbb{N}} \subset \mathbb{Q}$ of *rational numbers* is *computable*, if there exist three recursive functions $a, b, s : \mathbb{N} \to \mathbb{N}$ such that $b(k) \neq 0$ and

$$r_k = (-1)^{s(k)} \frac{a(k)}{b(k)} \qquad \text{for all } k \in \mathbb{N}.$$

A *real number* $x \in \mathbb{R}$ is *computable*, if there exists a computable sequence $(r_k)_{k\in\mathbb{N}}$ of rationals such that

$$\left|r_k - x\right| \leq 2^{-k} \qquad \text{for all } k \in \mathbb{N}.$$

We refer to the sequence $(r_k)_{k\in\mathbb{N}}$ as a *representation* for x.

Remark 2.8. The definition can be straightforwardly extended to vectors and complex numbers by considering each component and part individually, respectively.

Having established a formal notion of an algorithm via Turing machines, we can specify the characterization of a transparent algorithm from Definition 2.3. The key observation is that any Turing machine strictly operates on rational representations although the tackled problem may reside in the real domain, i.e., Turing machines map input representations to (computable) sequences. Hence, we can identify a Turing machine TM with an associated mapping $\Psi_{\mathrm{TM}} : R \to S$, where R and S denote the representation space and the space of (computable) sequences, respectively, defined as

$$R := \{(r_k)_{k\in\mathbb{N}} : (r_k)_{k\in\mathbb{N}} \text{ is a representation of some } x \in \mathbb{R}\}$$

and

$$S := \{(s_k)_{k\in\mathbb{N}} : (s_k)_{k\in\mathbb{N}} \subset \mathbb{Q} \text{ is a sequence computed by a Turing machine on input of a representation of some } x \in \mathbb{R}\}.$$

However, the transfer between the different domains – the representation space and the real numbers – shall not impact the algorithmic computation to preserve transparency.

Definition 2.9. An *algorithmic implementation* of a problem in the real domain is *transparent in the Turing model* if the associated Turing machine TM operates consistently on every input in the following sense: For any two representations $(r_k^1)_{k\in\mathbb{N}}, (r_k^2)_{k\in\mathbb{N}}$ of an input instance $x \in \mathbb{R}^m$, the output sequences $\Psi_{\mathrm{TM}}((r_k^1)_{k\in\mathbb{N}})$ and $\Psi_{\mathrm{TM}}((r_k^2)_{k\in\mathbb{N}})$ encode the same outcome in the underlying (real) problem domain, i.e., the outcome exclusively depends on x but not on its specific representation and other factors.

Remark 2.10. Consider a toy example in which an algorithm $\mathcal{A}$ realizes the real-valued function $f(x) = ax + b$ for some constants $a, b \in \mathbb{R}$. Then transparency requires that the implementation of $\mathcal{A}$ in the Turing model is independent of the representation of inputs $x \in \mathbb{R}$ as well as of the representation of the constants a, b.

Finally, observe that a Turing machine TM with an associated mapping Ψ_{TM} does not necessarily constitute a well-defined algorithm $\mathcal{A}$ with realization $\mathcal{A}_f$ on the real domain, i.e., $\Psi_{\mathrm{TM}}((r_k)_{k\in\mathbb{N}})$ may not represent a real number for an admissible input $(r_k)_{k\in\mathbb{N}}$. The notion of computable functions, which essentially describes the effective computation of functions by Turing machines on real numbers, circumvents this issue.

Definition 2.11. A function $f : \mathbb{R}^m \to \mathbb{R}^n$ is *Borel-Turing computable*, if there exists a Turing machine that transforms each representation of a vector $x \in \mathbb{R}^m$ into a representation for $f(x)$.

Remark 2.12. There exist different notions of computable functions on real numbers. We refer to [92–94] for an in-depth treatment of the topic and we highlight only the key properties of computable functions that help in providing an intuitive understanding. The need for approximate computations is closely related to the real-valued domain of f. For functions operating on natural or rational numbers, exact computations can in principle be expected. Moreover, Borel-Turing computability can also be applied to complex-valued functions by identifying real and imaginary parts as real numbers.

Remark 2.13. Borel-Turing computability implies the existence of an algorithm $\mathcal{A}$ realizing a mapping $\mathcal{A}_f : \mathbb{R}^m \to \mathbb{R}^n$, i.e., $\mathcal{A}_f = f$, via the associated Turing machine. From a practical point of view, Borel-Turing computability can be seen as a requirement for an algorithmic computation of the input-output relation of a problem (described by a function) on perfect digital hardware in the following sense. In particular, the associated Turing machine, i.e., the algorithm, takes input representations and determines a sufficient input precision, i.e., suitable elements of the representations, so that the performed computation will terminate once an output within a prescribed worst-case error bound $\varepsilon > 0$ is obtained. Here, the input representation is itself a Turing machine, which can be queried with a precision parameter and provides an approximation of the "exact" input. In theory, by iteratively calling the algorithm with a declining sequence of error bounds $\varepsilon_k = 2^{-k}$ for a fixed representation of an input $x \in \mathbb{R}^m$, one would obtain a (computable) representation encoding $f(x)$.

Remark 2.14. In general, the representation of a computable vector x is not unique. Hence, the representation of a Borel-Turing computable function f at $f(x)$ may depend on the representation of x given as input to the Turing machine. A small wrinkle is added by the fact that not every real number has a description based on recursive functions, i.e., only the computable real numbers (indeed a proper subset of the real numbers) are considered admissible inputs in the Borel-Turing setting. In contrast, any real number can be approximated by a convergent sequence of rational numbers so that the concept of Borel-Turing computability can be extended to the whole real number domain under certain conditions. For our needs, these subtle differences and their implications can be neglected and we apply the notion of Borel-Turing computability introduced in the definition. We point to [95] for the treatment of the inverse problem use case under the adapted notion.

Aside from Turing machines, further abstractions of digital computations have been established, for instance, the Blum-Shub-Smale (BSS) machines represent a common heuristic formalization (in contrast to the precise model of Turing machines according to the Church-Turing thesis) [88]. Therefore, BSS machines do not provide a suitable starting point for our intended trustworthiness considerations on digital hardware by design. At the same time, it turns out that the BSS framework suits a different context via its relation to analog hardware. Although a widely accepted formalization equivalent to the Turing model for other types than digital hardware does not exist, the BSS framework is a candidate to (abstractly) model several forms of analog computing [96].

2.2.2. Analog computations

The study of analog hardware gained considerable traction in the last years due to the rapidly increasing demand for energy and storage of digital information processing and computing [21,20]. Even more, convincing arguments indicate that the scaling of computation and information processing at the current level is not sustainable if the same (digital) technologies are applied, i.e., a technological disruption based on the introduction of new (analog) approaches is necessary [22,25]. For instance, innovative memory and storage technology as well as novel approaches in world-machine interfaces that can sense, perceive, and reason based on low operational power and latency are required. This may only be realizable by incorporating analog (electronic) components in the (currently mostly digital) computing and information processing pipeline. A prime example is provided by neuromorphic computing and signal processing systems [97–102].

Neuromorphic systems are inspired by the structure and information processing of biological neural networks and can be realized in analog, digital, and mixed analog-digital fashion [85]. Digital computers typically follow the von Neumann architecture [103] that leads to inherently large time and energy overhead in data transport [104,105]. In contrast, neuromorphic computers incorporate emerging concepts such as 'in-memory computing', which avoid these bottlenecks by design [106–109]. At present, the main advantage of neuromorphic systems is the expected savings in energy consumption, in particular, by deploying artificial intelligence applications such as deep learning on neuromorphic hardware [97–101,110]. Further promising (but not yet realizable) analog computing paradigms comprise biocomputing [96,111–113] and (analog) quantum simulation [114,115].

Moreover, analog computing may also provide benefits for the computability of certain problems. Important tasks in information theory, signal processing, and simulation are not Borel-Turing computable [116–122], whereas, computability on BSS machines has been established for certain applications [123–125]. Computability in BSS sense conveys the same concept as computability on digital devices but under a more general framework. BSS machines carry over complexity theory in the Turing machine model to a larger variety of structures by operating on arbitrary rings or fields, even infinite fields such as $\mathbb{R}$ are feasible. In addition, BSS machines on $\mathbb{Z}_2 = \{\{0,1\},+,\cdot\}$ recover the theory of Turing machines. Thus, BSS machines are a structure-wise similar, generalized abstraction of Turing machines. BSS machines operate as Turing machines on an infinite strip of tape according to a program illustrated by a finite directed graph with different types of nodes associated with operations such as input processing, computing, branching, and output processing. For a detailed introduction and comparison, we refer to [88,126] and the references therein. We only wish to highlight that a BSS machine, which operates on $\mathbb{R}$, processes real numbers (as entities) and performs field operations ('+','·') via compute nodes and comparisons ('<','>','=') via branch nodes exactly. Thus, BSS machines offer a mathematical framework to investigate analog real number processing and computation. Hereby, BSS computable functions are simply input-output maps of BSS machines, i.e., the set of BSS computable functions precisely characterizes functions that can be computed (in finite time) by algorithmic means in the BSS model.

Definition 2.15. A function $f : \mathbb{R}^m \to \mathbb{R}^n$ is *BSS computable* if there exists a BSS machine with an input-output relation described by f.

Remark 2.16. The output $\Psi_{\mathcal{B}}(x)$ of a BSS machine $\mathcal{B}$ is defined if $\mathcal{B}$ according to its program terminates its calculations on input $x \in \mathbb{R}^m$ after a finite number of steps. The hereby introduced map $\Psi_{\mathcal{B}} : \mathbb{R}^m \to \mathbb{R}^n$ is the input-output function of $\mathcal{B}$, which directly relates to the realization of an algorithm (i.e., BSS machine) introduced in Definition 2.1.

Remark 2.17. The definition can be straightforwardly extended to complex functions, however, the representation of the complex field impacts the capabilities of corresponding BSS machines and thereby also the set of computable functions. BSS machines that treat complex numbers as entities can not perform comparisons of arbitrary complex numbers but only check the equality to zero at their branch nodes since $\mathbb{C}$ is not an ordered field. As a consequence, elementary complex functions such as $z \mapsto \mathfrak{R}(z)$, $z \mapsto \mathfrak{I}(z)$, $z \mapsto \bar{z}$, and $z \mapsto |z|$ are not BSS computable in this setting [88]. Identifying $\mathbb{C}$ with $\mathbb{R}^2$ instead and employing BSS machines that take complex inputs z in the form of $(\mathfrak{R}(z), \mathfrak{I}(z))$ entails that $z \mapsto \mathfrak{R}(z)$ and $z \mapsto \mathfrak{I}(z)$ are computable since the corresponding BSS machine only needs to process the respective part of the representation of z.

The crucial property of the BSS framework is the handling of the elements of the associated ring or field as entities, which implies the exact storing and processing of real numbers in this computation model. Thus, the real BSS model can not be implemented on digital hardware. It is even unclear if and to what degree a computing device realizing the real BSS model can be constructed by (future) hardware technology due to physical constraints [127]. For instance, random noise in physical processes, which execute the mathematical operations in a hypothetical computing device, complicates or even prevents exact processing and computation. Moreover, the BSS model strongly focuses on algebraic properties, leading to the non-computability of trigonometric, logarithmic, and root functions in the BSS model on real numbers. These functions are typically considered elementary functions expected to be computable in a practical and useful computing model (as indeed is the case in the Turing model). Therefore, real BSS machines are a strongly idealized model and they may not capture the true capabilities of forthcoming analog computing devices so theoretical benefits may not turn into practical ones. Nonetheless, the study of BSS computable functions provides to a certain degree an outline of the limits of analog computations.

2.2.3. Computability and trustworthiness

Next, we will analyze under which conditions a trustworthy algorithmic solution of a problem described by an input-output relation, i.e., an associated function, can be expected. The crucial step is establishing AgT because it is the basis for various trustworthiness considerations; see Subsection 2.1. In Observation 2.5, we derived via Definition 2.3 a necessary prerequisite to obtain AgT. Applying this framework to the Turing model, we can derive a necessary condition for a transparent algorithm on digital hardware.

Lemma 2.18. *Given a problem with an input-output relation described by a function $f : \mathbb{R}^m \to \mathbb{R}^n$, let $\mathcal{A}$ be an algorithm implemented on a Turing machine. If the algorithmic implementation is not transparent, then $\mathcal{A}$ does not realize f.*

Proof. Assume the algorithmic implementation of $\mathcal{A}$ on a Turing machine TM is not transparent. Thus, by definition, two representations of some input instance $x \in \mathbb{R}^m$ exist such that the computed output sequences by TM do not encode the same real number. Therefore, at least one of the (computable) output sequences does not represent $f(x)$. Hence, $\mathcal{A}$ does not realize f. □

We can immediately infer from Lemma 2.18 that Borel-Turing non-computability of a function prevents the existence of an algorithm complying with transparency in the digital computing model so that the following statement holds.

Theorem 2.19. *There exists an algorithm $\mathcal{A}$ with transparent implementation in the Turing model realizing $\mathcal{A}_f$ if and only if $f : \mathbb{R}^m \to \mathbb{R}^n$ is Borel-Turing computable.*

Remark 2.20. An algorithm adhering to AgT is necessarily transparent, i.e., Borel-Turing computability of the tackled problem is a prerequisite for AgT. In contrast, Borel-Turing non-computability of a problem implies that any algorithmic approach implemented on digital hardware will have unavoidable flaws or at least certain limits: For any algorithm, there exists a representation of some input $x \in \mathbb{R}^m$ such that the computed output sequence does not converge at all or does not converge to $f(x)$. Crucially, the integrity between the mathematical model of the problem and the mathematical model of the computing platform is lost. Hence, AgT via Observation 2.5 can not be guaranteed.

Similar reasoning can be adopted for BSS machines and BSS computable problems. In the BSS model algorithms directly operate on real numbers so that each real number is uniquely represented by itself. Thus, any problem that is BSS computable by definition fulfills the transparency condition. In contrast, the transparency of an algorithm successfully solving a problem immediately implies BSS computability of the problem.

Theorem 2.21. *There exists an algorithm $\mathcal{A}$ with transparent implementation in the BSS model realizing $\mathcal{A}_f$ if and only if $f : \mathbb{R}^m \to \mathbb{R}^n$ is BSS computable.*

Hence, by studying the computability of a problem or, more accurately, the computability of a function describing the problem in mathematical terms, we can formally assess the existence of trustworthy algorithms (based on AgT). Thus, we obtained a precise tool to decide whether trustworthiness in an algorithmic computation can be attained. We will apply the introduced framework to a specific use case of deep learning – finite dimensional inverse problems –, aiming to derive broadly applicable observations.

3. Use case for trustworthiness analysis

3.1. Inverse problems

Inverse problems in imaging sciences, i.e., image reconstruction from measurements, is a recurrent task in industrial, scientific, and medical applications such as magnetic resonance imaging (MRI) and X-ray computed tomography (CT), where the measurements are acquired by the Fourier and Radon transform, respectively.

Definition 3.1. An *inverse problem* in the finite-dimensional, underdetermined, and linear setting can be formulated as:

$$\text{Given noisy measurements } y = Ax + e \in \mathbb{C}^m \text{ of } x \in \mathbb{C}^N, \text{ recover } x, \tag{3.1}$$

where $A \in \mathbb{C}^{m \times N}, m < N$, is the *sampling operator*, $e \in \mathbb{C}^m$ is a noise vector, $y \in \mathbb{C}^m$ is the *vector of measurements*, and $x \in \mathbb{C}^N$ is the object to recover.

Remark 3.2. In the context of the definition, a typical object to recover is a vectorized discrete image. Furthermore, the underdetermined setting $m < N$ with a limited number of measurements is standard in practice due to time, cost, power, or other constraints.

Due to the ill-posedness of (3.1) a typical solution strategy is to consider a mathematically more tractable description via an optimization problem. The simplest form is given by the least-squares problem

$$\arg\min_{x \in C^N} \|Ax - y\|_{\ell_2}.$$

Although the solution map is considerably simpler than the one of the original problem (3.1), the solution is generally not unique. By adding regularization terms to the optimization problem, one tries to steer the optimization process towards favorable solutions. For instance, sparse solutions tend to possess desirable properties but explicitly enforcing them via the ℓ_0 norm is typically intractable. However, incorporating regularization terms that promote sparsity in the recovery resulted in various solution techniques [128–138] for common approaches such as (quadratically constrained) *basis pursuit* [139,140]

$$\arg\min_{x \in \mathbb{C}^N} \|x\|_{\ell_1} \text{ such that } \|Ax - y\|_{\ell_2} \leq \varepsilon \tag{BP}$$

and unconstrained square root *lasso* [141,142]

$$\arg\min_{x \in \mathbb{C}^N} \lambda \|x\|_{\ell_1} + \|Ax - y\|_{\ell_2}, \tag{Lasso}$$

where the magnitude of $\varepsilon > 0$ and $\lambda > 0$ controls the relaxation, respectively. In recent years, deep learning techniques led to a paradigm shift and were established as the predominant method to tackle inverse problems [143–150]. The core idea behind the deep learning approach is to learn the underlying relations of the reconstruction process based on data samples.

3.2. Deep learning

In deep learning a structure called *(artificial) neural network*, loosely inspired by biological brains, is employed to approximate an unknown function via a set of given input-output value pairs. A neural network is essentially a parameterized mapping with properties and capabilities depending on its specific design [11,16,17]. The simplest form is feedforward neural networks, which we will focus on in the remainder.

Definition 3.3. A *(feedforward) neural network* $\Phi : \mathbb{R}^d \to \mathbb{R}^k$ is given by

$$\Phi(x) = T_L \rho(T_{L-1} \rho(\dots \rho(T_1 x))), \quad x \in \mathbb{R}^d, \tag{3.2}$$

where $T_\ell : \mathbb{R}^{n_{\ell-1}} \to \mathbb{R}^{n_\ell}$, $\ell = 1, \dots, L$, are affine-linear maps

$$T_\ell x = W_\ell x + b_\ell, \quad W_\ell \in \mathbb{R}^{n_\ell \times n_{\ell-1}}, b_\ell \in \mathbb{R}^{n_\ell} \text{ with } n_0 = d, n_L = k,$$

and $\rho : \mathbb{R} \to \mathbb{R}$ is a non-linear function acting component-wise on a vector. The matrices W_ℓ are called *weights*, the vectors b_ℓ *biases*, and the function ρ *activation function*, with a common one being the basic *ReLU activation* $\rho(x) = \max\{0, x\}$.

Remark 3.4. In addition, a neural network can easily be adapted to work with complex-valued inputs by representing them as real vectors consisting of the real and imaginary parts.

By adjusting the network's parameters, i.e., its weights and biases, according to an optimization process on available data samples, the network ideally learns to approximate the sought function. This process – the standard technique is to apply stochastic gradient descent coupled with backpropagation [10] – is usually referred to as the training of a neural network. We refer to [6–8] for an in-depth overview of deep learning.

3.3. Deep learning for inverse problems

Turning to inverse problems, deep learning techniques can be incorporated into the solution approach in various ways [151]. The most fundamental and generally applicable approach is to directly learn a mapping from measurements y to reconstructions x without making problem-specific assumptions. Hence, the goal is to obtain a neural network that for some fixed sampling operator $A \in \mathbb{C}^{m \times N}$ and optimization parameter $\mu > 0$ approximates the reconstruction map

$$\Xi_{P,A,\mu} : \mathbb{C}^m \rightrightarrows \mathbb{C}^N, \quad y \mapsto P(A, y, \mu), \tag{3.3}$$

where $P(A, y, \mu)$ represents the set of minimizers of an optimization problem P given a measurement $y \in \mathbb{C}^m$. For instance, P is described in (BP) for basis pursuit with $\mu := \varepsilon$. Note that the reconstruction map is typically set-valued, denoted by '$\rightrightarrows$', since the corresponding optimization problem does not possess a unique solution. Therefore, it is not entirely correct to state that the goal is to compute a neural network that approximates the mapping $\Xi_{P,A,\mu}$. We do not expect a neural network to reproduce all minimizers for a single input but it suffices if the network approximates one specific minimizer. In Section 4, we will return to and clarify this issue.

Ideally, the training process results in a neural network capable of solving instances of a particular inverse problem defined by the sampling operator A, the optimization problem P, and the optimization parameter μ. More powerful would be a neural network that can solve generic inverse problems under specific conditions, e.g., a network that approximates the reconstruction map

$$\Xi_{P,m,N} : \mathbb{C}^{m \times N} \times \mathbb{C}^m \times \mathbb{R}_{>0} \rightrightarrows \mathbb{C}^N, \quad (A, y, \mu) \mapsto P(A, y, \mu) \tag{3.4}$$

of any inverse problem of dimension $m \times N$ corresponding to an optimization problem P. One can generalize the objective further by allowing the dimension and/or the optimization problem as additional inputs. However, it is a priori not even clear if networks that approximate $\Xi_{P,A,\mu}$ and $\Xi_{P,m,N}$ exist and can be found via the described deep learning framework. Whereas the former problem can be approached by an expressivity analysis of neural networks, which is already supported by a large body of literature [152–159], the latter problem is more intricate. Despite the existence of the sought networks, obtaining them based solely on data samples may not be feasible or the computation process may result in networks with unfavorable properties such as a lack of trustworthiness. Hence, we assess the possibility of computing neural networks that solve inverse problems under the introduced computability framework.

4. Algorithmic solvability of inverse problems

It is intuitively clear that the training and inference of neural networks are distinct problems with varying difficulty. For inverse problems, the training process targets a neural network, which approximates the mapping from measurements to the original data. In other words, one is interested in finding a neural network, i.e., suitable weights and biases, that realizes the mapping in (3.3) or (3.4). We focus on the more general case (3.4), where the associated optimization problem is given by (BP) or (Lasso). Before turning to the question if the desired network can be computed algorithmically, we wish to remark that once obtained the execution of said network on a given input can be performed reliably.

Theorem 4.1 *([52,53]). A neural network Φ as defined in (3.2) is a Turing/BSS computable function given that the activation function $\rho : \mathbb{R} \to \mathbb{R}$ is Turing/BSS computable.*

Proof Sketch. This follows from the fact that under the given conditions Φ is a composition of computable functions in both computing models. □

Remark 4.2. Note that we can extend the observation in the theorem to more advanced architectures such as convolutional networks [160]. In particular, the statement holds for any network architecture composed of basic computable building blocks, which indeed is true for many common variants. Furthermore, standard activations applied in practice such as ReLU are indeed computable.

Does a similar statement hold for the training phase? First, note that the mapping (3.4) targeted by the training process is generally multi-valued since the solution of the associated optimization problem does not need to be unique. Thus, it neither fits in the introduced deep learning nor computability framework. However, we can circumvent this issue by establishing suitable single-valued functions that suit both frameworks. The underlying idea is that typically one is not interested in the whole solution set, but one particular element of the solution set or even an element reasonably close to the solution set suffices. Hence, of interest is not to compute the entire set described by the map $\Xi_{P,m,N}$ for given input (A, y, μ) in (3.4) but to compute one element of $\Xi_{P,m,N}(A, y, \mu)$, i.e., exactly one minimizer of the optimization problem P. In particular, it is not relevant which of the (possibly infinitely many) minimizers is obtained, since any of those is an appropriate solution.

Formally, this concept can be captured by single-valued restrictions of a multi-valued function $f : \mathcal{V} \rightrightarrows \mathcal{Z}$: For each input $v \in \mathrm{dom}(f)$ there exists at least one element $z_v \in f(v) \subset \mathcal{P}(\mathcal{Z})$ so that the map

$$f^s : \mathcal{V} \to \mathcal{Z}, \quad v \mapsto z_v$$

is well-defined. We denote by $\mathcal{M}_f$ the set of all the single-valued functions associated with the multi-valued function f, i.e., all single-valued functions f^s that are formed by restricting the output of a multi-valued map f to a single value for each input.

Definition 4.3. A problem with an input-output relation described by a multi-valued function $f : \mathcal{X} \rightrightarrows \mathcal{Y}$ is *algorithmically solvable on a BSS or Turing machine* if there exists a function $f^s \in \mathcal{M}_f$ that is computable on a BSS or Turing machine, respectively.

By applying the notion of algorithmic solvability, we reduced our task to evaluate the computability of well-defined single-valued functions. A task is deemed algorithmically solvable if at least one computable function in $\mathcal{M}_f$ exists.

Remark 4.4. The presented approach is not the only viable option to assess the computability of a multi-valued mapping f. The (non-)existence of algorithms can also be established via the distance to the solution set measured by an appropriate metric. Therefore, a hypothetical algorithm does not approximate a fixed element of the solution as the admissible distance to the solution set varies, which is the case in our approach via single-valued restrictions. Thus, a problem characterized by f may not be algorithmically solvable in the introduced sense, but an algorithm obeying this distance description may still exist. Note that this case only arises if f is indeed a multi-valued and not a single-valued function. Moreover, the notion of algorithmic transparency needs to be adjusted to cover 'the distance to the solution set' approach. We refer to [95] for more details, where algorithmic solvability via this notion is pursued in the inverse problem setting.

Finally, we can apply the framework of algorithmic solvability to inverse problems described via basis pursuit and square root lasso. Indeed, we find differences in algorithmic solvability in the Turing and BSS model as we now detail.

4.1. Algorithmic non-solvability of inverse problems in Turing model

In the Turing setting, for a certain range of optimization parameters we not only establish algorithmic non-solvability but even non-approximability.

Theorem 4.5 *([52]). Consider the optimization problems (BP), (Lasso) and the associated mappings $\Xi_{BP,m,N}(\cdot,\cdot,\varepsilon)$ and $\Xi_{Lasso,m,N}(\cdot,\cdot,\lambda)$, where $N \geq 2$ and $m < N$, for fixed parameters $\varepsilon \in (0, 1/4)$ and $\lambda \in (0, 5/4) \cap \mathbb{Q}$, respectively. The problems described by $\Xi_{BP,m,N}(\cdot,\cdot,\varepsilon)$ and $\Xi_{Lasso,m,N}(\cdot,\cdot,\lambda)$ are not algorithmically solvable on Turing machines.*

Proof Sketch. The main step is to characterize algorithmic non-solvability conditions for functions $\Xi_{P,m,N}(\cdot,\cdot,\mu)$, introduced in (3.4) based on the solution set of the optimization problem P with optimization parameter μ. The idea is to 'encode' a recursively enumerable but non-recursive set $B \subset \mathbb{N}$ in the domain of $\Xi_{P,m,N}(\cdot,\cdot,\mu)$, i.e., a set such that there exists a Turing machine that takes numbers $n \in \mathbb{N}$ as input and confirms (after a finite amount of time) that n is an element of B if $n \in B$ does indeed hold, but fails to decide whether $n \in B$ or $n \in B^c$ in general. To that end, a computable sequence $(\xi_n)_{n\in\mathbb{N}} \subset \Xi_{P,m,N}(\cdot,\cdot,\mu)$ is constructed so that $\{\Xi_{P,m,N}(\xi_n,\mu) : n \in B\}$ can be distinguished from $\{\Xi_{P,m,N}(\xi_n,\mu) : n \in B^c\}$ by algorithmic means provided that $\Xi_{P,m,N}(\cdot,\cdot,\mu)$ is

Borel-Turing computable. However, one can construct a Turing machine that decides $n \in B$ or $n \in B^c$ for arbitrary $n \in \mathbb{N}$, contradicting the non-recursiveness of B. Subsequently, the construction can be verified for the considered functions $\Xi_{\mathrm{BP},m,N}(\cdot,\cdot,\varepsilon)$ and $\Xi_{\mathrm{Lasso},m,N}(\cdot,\cdot,\lambda)$. □

Remark 4.6. The statement in the theorem can be strengthened by providing a lower bound on the achievable algorithmic approximability of the problem, i.e., how precise single-valued restrictions of the sought reconstruction map can be approximated by Borel-Turing computable functions. Note that the limitations do not arise due to the unboundedness of the input domain, but hold on a compact input set. Furthermore, algorithmic non-solvability is not connected to poor conditioning of the inverse problem instances; one can construct input domains consisting only of well-conditioned instances with the same limitations. For details, we refer to [52].

By invoking Theorem 2.19, we infer that no transparent algorithms to solve inverse problems exist on Turing machines.

Corollary 4.7. *In the setting of Theorem 4.5, there does not exist a transparent algorithm solving inverse problems described by* $\Xi_{BP,m,N}(\cdot,\cdot,\varepsilon)$ *and* $\Xi_{Lasso,m,N}(\cdot,\cdot,\lambda)$.

4.2. Algorithmic solvability of inverse problems in BSS model

In the BSS setting, a general algorithmic non-solvability statement does not hold. We indeed can establish algorithmic solvability under specific circumstances. To that end, we distinguish between a real and complex domain since BSS machines show distinct behavior depending on the underlying structure.

4.2.1. Real case

First, we consider the real case. Given a multi-valued mapping $f : \mathcal{V} \rightrightarrows \mathcal{Z}$, we denote by $f^{\mathbb{R}}$ its restriction to real inputs and outputs. Although only the complex domain was studied explicitly in Theorem 4.5, the algorithmic non-solvability in the Turing model remains valid for basis pursuit (BP) described by the mapping $\mathcal{M}_{\Xi^{\mathbb{R}}_{\mathrm{BP},m,N}(\cdot,\cdot,\varepsilon)}$ since the proof idea translates to the strictly real case. In contrast, the same problem is algorithmically solvable in the BSS model.

Theorem 4.8 *([53]). Consider the optimization problem* (BP) *restricted to the real domain and the associated mapping* $\Xi^{\mathbb{R}}_{BP,m,N}$. *The problem described by* $\Xi^{\mathbb{R}}_{BP,m,N}$ *is algorithmically solvable on BSS machines.*

Proof Sketch. The approach is to rewrite the problem such that an established algorithm for finding a minimizer of a polynomial on a semialgebraic set can be applied. Due to the restriction to the real domain, the involved terms can indeed be transferred to the required setting. □

Remark 4.9. Note that the optimization parameter acts as an additional input to the mapping $\Xi^{\mathbb{R}}_{\mathrm{BP},m,N}$ (whereas in the Turing setting in Theorem 4.5 the optimization parameter was fixed beforehand). Therefore, in the BSS setting, we proved the existence of an even stronger algorithm (with an additional input parameter) than the one assessed in Theorem 4.5. We refer to [53] for more details and the proof.

Remark 4.10. Although an algorithm solving the problem $\Xi^{\mathbb{R}}_{\mathrm{BP},m,N}$ exists, its computational complexity may remain inappropriately high. Hence, the theorem only provides a theoretical existence result neglecting the question of practical implementation.

In the case of (square root) lasso optimization (Lasso), algorithmic solvability can not be established on BSS machines, not even on the restricted real domain. The underlying issue is the BSS non-computability of the square root function on the real numbers, which arises due to the algebraic structure of the BSS model [88]. Hence, the ℓ_2 norm is not BSS computable, which renders (square root) lasso optimization infeasible on BSS machines. Note that the explicit computation of the ℓ_2 norm can be avoided for basis pursuit, because it arises only in the description of the constraint, whereas for (square root) lasso it is directly incorporated in the objective. One can circumvent this problem by assuming that BSS machines possess an additional module that can be called to compute the square root. This is motivated by the fact that the square root is an elementary function that should be computable in a practical model as is the case for Turing machines [93]. Without this additional assumption, we either need to modify or approximate the objective of the corresponding optimization problem to obtain algorithmic solvability on BSS machines. We will consider the former approach and return to the latter approach in the complex setting. Instead of square root lasso, we can consider lasso optimization [142,141,161] given by

$$\arg\min_{x\in\mathbb{C}^N} \lambda \|x\|_{\ell_1} + \|Ax - y\|^2_{\ell_2}. \tag{Lasso^2}$$

Here, the objective does not require the computation of the square root function (due to the squaring of the ℓ_2 norm) and, indeed, this change is sufficient to establish algorithmic solvability via the same approach as in Theorem 4.8.

Theorem 4.11 *([53]). Consider the optimization problem* (Lasso^2) *restricted to the real domain and the associated mapping* $\Xi^{\mathbb{R}}_{\text{Lasso}^2,m,N}$. *The problem described by* $\Xi^{\mathbb{R}}_{\text{Lasso}^2,m,N}$ *is algorithmically solvable on BSS machines.*

Remark 4.12. The analogous proof technique as for the square root lasso minimization problem in Theorem 4.5 can be applied to derive algorithmic non-solvability of lasso minimization (Lasso^2) on Turing machines [53].

Applying Theorem 2.21, we conclude that a transparent algorithm for solving real inverse problems exists on BSS machines.

Corollary 4.13. *In the setting of Theorem 4.8 and Theorem 4.11, there does exist a transparent algorithm solving inverse problems described by* $\Xi^{\mathbb{R}}_{BP,m,N}$ *and* $\Xi^{\mathbb{R}}_{\text{Lasso}^2,m,N}$, *respectively.*

4.2.2. Complex case

We have to choose a suitable representation for BSS machines operating on complex numbers. As described in Remark 2.17, considering complex numbers as entities results in the non-computability of elementary complex functions. In contrast, identifying $\mathbb{C}$ with $\mathbb{R}^2$ and representing complex inputs z in the form of $(\Re(z), \Im(z))$ circumvents this problem to a certain degree. However, even in the $\mathbb{R}^2$-representation ℓ_p norms are generally not computable functions, since they require the computation of a square root (which is not a real BSS computable function). Similarly to the real case, we can introduce adjusted optimization problems that promote solutions with properties similar to the original solutions but allow for algorithmic solvability in the BSS model. For details, we refer to [53] and wish to mention that algorithmic solvability in the Turing model is still not achievable for the adjusted problems. Although the adaptation of the objectives tries to maintain the original structural properties, they are not derived by rigorous reasoning.

Instead of replacing the optimization problem, we can also approximate its objective. This approach is demonstrated for basis pursuit by establishing an adequate (BSS computable) approximation of the ℓ_1 norm.

Theorem 4.14 *([53]). Let* $\beta, \gamma > 0$. *For* $A \in \mathbb{C}^{m\times N}$, $y \in \mathbb{C}^m$ *and* $\varepsilon > 0$ *consider the optimization problem*

$$\arg\min_{x\in I_\beta} p_{\beta,\gamma}(x) \text{ such that } \|Ax - y\|_{\ell_2} \leq \varepsilon, \tag{BP-A}$$

where $p_{\beta,\gamma}$ *is a polynomial satisfying*

$$\sup_{x\in I_\beta} \left| \|x\|_{\ell_1} - p_{\beta,\gamma}(x) \right| \leq \gamma$$

and

$$I_\beta := \{x \in \mathbb{C}^N : \|x\|_{\ell_2} < \sqrt{N}\beta\}$$

Then, the problem described by $\Xi_{\text{BP-A},m,N}$ *is algorithmically solvable in the BSS model.*

Proof Sketch. Applying the Weierstrass approximation theorem, in particular, its constructive proof via Bernstein polynomials, we can derive a (BSS computable) polynomial $p_{\beta,\gamma}$ approximating the ℓ_1 norm up to an error of γ on I_β. Finally, the algorithmic solvability of $\Xi_{\text{BP-A},m,N}$ follows along the same lines as in the proof of Theorem 4.8. □

Remark 4.15. The objective $p_{\beta,\gamma}$ in (BP-A) approximates up to an error of γ the ℓ_1 norm, i.e., the objective of basis pursuit optimization, on the set I_β. In this sense, (BP-A) represents an approximation of basis pursuit if its minimizers are contained in I_β. Additionally, one can construct a BSS computable function that decides for input (A, y, ε) if basis pursuit (BP) has at least one solution and if the solution(s) are contained in I_β. Therefore, there does exist a BSS machine $\mathcal{B}_{\beta,\gamma}$ that checks if the solutions of basis pursuit for (A, y, ε) are contained in I_β. If the answer is positive, a solution of (BP-A) is computed consecutively. Otherwise, the computation is aborted since the approximation accuracy γ and acceptance domain depending on β can not be adjusted autonomously. In other words, for each pair of parameters (β, γ) a distinct BSS machine $\mathcal{B}_{\beta,\gamma}$ needs to be constructed. Moreover, note that the obtained minimizers of (BP-A) and basis pursuit need not agree and we do not obtain worst-case bounds on their distance, for details we refer to [53].

Remark 4.16. In the Turing model, the outlined approach to approximate basis pursuit is not feasible [53]. Even more, due to lower bounds on the algorithmic approximability in the Turing model (see Remark 4.6), different approximation schemes necessarily have certain limits in this setting.

Finally, via Theorem 2.21 we can state a similar result for trustworthiness as in the real case.

Corollary 4.17. *In the setting of Theorem 4.14, there does exist a transparent algorithm approximating inverse problems described by* $\Xi_{\text{BP-A},m,N}$.

4.3. Comparison of results in Turing and BSS model

The presented findings indicate that the degree of algorithmic solvability of inverse problems depends on both the considered problem description and the computing model. In the Turing model, a rather general algorithmic non-solvability statement holds, also supported by the results in [95], whereas the landscape is more diverse in the BSS setting. Here, the potential of algorithmic solvability is connected to the specific properties of the underlying optimization problem, which can to a certain extent be positively influenced by modifying or approximating the objective. Although the adjustments may typically not be applied in practice, they maintain the properties of the original formulation to some degree and show that a wide range of inverse problem descriptions is in principle algorithmically solvable in the BSS model.

In contrast, related approaches appear to be infeasible in the Turing model. On the one hand, (reasonable) modifications of the objectives of the optimization problems do not influence algorithmic solvability, since algorithmic non-solvability is related to properties of the underlying solution set of the given task, which pertain to a broad class of inverse problem descriptions. On the other hand, algorithmic non-approximability in the Turing model also renders approximate approaches impractical. Characterizing classes of inputs that allow for algorithmic solvability and thereby identifying problematic inputs, that violate performance guarantees, could potentially alleviate the non-computability issue. However, it was found that implementing an exit-flag functionality, i.e., aborting the computation and notifying the user once a 'problematic' input is recognized, on Turing machines is in general not feasible for inverse problems [162].

Consequently, we can observe a gap in algorithmic solvability between the Turing and the BSS model. In particular, BSS machines provide a greater capacity to solve inverse problems algorithmically. However, the power of the BSS model is heavily dependent on the (exact) representation and processing of real numbers as entities: Essentially the same limitations as in the Turing model arise if approximating sequences are employed to represent real numbers [95]. It should also be noted that algorithmic solvability of inverse problems was assessed in a very general framework, i.e., we did not consider a specific but broad class of inverse problems. Hence, restricting to a more narrow framework consisting of a limited number of classes may change the degree of algorithmic solvability. Thereby, specific properties of the considered problems could be exploited, which at the same time can not be incorporated into a more universal approach. Thus, a trade-off between generality and trustworthiness expressed through AgT may not be avoidable in our framework, but the degree may vary with the underlying computing paradigm.

Acknowledgments

This work of H. Boche was supported in part by the German Federal Ministry of Education and Research (BMBF) in the programme of Souverän. Digital. Vernetzt, research HUB 6G-life, project identification number: 16KISK002, and by the BMBF Quantum Projects QUIET, Grant 16KISQ093, QD-CamNetz, Grant 16KISQ077, and QuaPhySI, Grant 16KIS1598K. H. Boche was also partially supported by the project "Next Generation AI Computing (gAIn)", funded by the Bavarian Ministry of Science and the Arts and the Saxon Ministry for Science, Culture, and Tourism.

This work of Gitta Kutyniok was supported in part by the Konrad Zuse School of Excellence in Reliable AI (DAAD), the Munich Center for Machine Learning (BMBF) as well as the German Research Foundation under Grants DFG-SPP-2298, KU 1446/31-1 and KU 1446/32-1. G. Kutyniok acknowledges support from LMUexcellent, funded by the Federal Ministry of Education and Research (BMBF) and the Free State of Bavaria under the Excellence Strategy of the Federal Government and the Länder as well as by the Hightech Agenda Bavaria. Furthermore, G. Kutyniok was also partially supported by the project "Next Generation AI Computing (gAIn)", funded by the Bavarian Ministry of Science and the Arts and the Saxon Ministry for Science, Culture, and Tourism.

Data availability

No data was used for the research described in the article.

References

[1] A.W. Senior, et al., Improved protein structure prediction using potentials from deep learning, Nature 577 (2020) 706–710.
[2] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: surpassing human-level performance on ImageNet classification, in: ICCV 2015, IEEE, 2015, pp. 1026–1034.
[3] D. Silver, et al., Mastering the game of Go with deep neural networks and tree search, Nature 529 (2016) 484–503.
[4] T. Brown, et al., Language models are few-shot learners, in: H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, H. Lin (Eds.), NeurIPS 2020, in: Curran Associates, vol. 33, Inc., 2020, pp. 1877–1901.
[5] R. Lam, et al., Learning skillful medium-range global weather forecasting, Science 382 (6677) (2023) 1416–1421.
[6] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (2015) 436–444.
[7] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016, http://www.deeplearningbook.org.
[8] J. Berner, P. Grohs, G. Kutyniok, P. Petersen, The modern mathematics of deep learning, in: Mathematical Aspects of Deep Learning, Cambridge University Press, 2022.
[9] W.S. McCulloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity, Bull. Math. Biophys. 5 (4) (1943) 115–133.
[10] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors, Nature 323 (1986) 533–536.
[11] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, Commun. ACM 60 (6) (2017) 84–90.
[12] M. Pandey, et al., The transformational role of GPU computing and deep learning in drug discovery, Nat. Mach. Intell. 4 (3) (2022) 211–221.
[13] C. Silvano, et al., A survey on deep learning hardware accelerators for heterogeneous HPC platforms, arXiv:2306.15552, 2023.

[14] N. Jouppi, et al., TPU v4: an optically reconfigurable supercomputer for machine learning with hardware support for embeddings, in: ISCA 2023, Association for Computing Machinery, New York, NY, USA, 2023.
[15] A.C. Elster, T.A. Haugdahl, Nvidia Hopper GPU and grace CPU highlights, Comput. Sci. Eng. 24 (2) (2022) 95–100.
[16] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: CVPR 2016, IEEE, 2016, pp. 770–778.
[17] A. Vaswani, et al., Attention is all you need, in: NIPS 2017, vol. 30, Curran Associates, Inc., 2017.
[18] R. Balestriero, et al., A cookbook of self-supervised learning, arXiv:2304.12210, 2023.
[19] P.F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, D. Amodei, Deep reinforcement learning from human preferences, in: NIPS 2017, vol. 30, Curran Associates, Inc., 2017.
[20] N.C. Thompson, K. Greenewald, K. Lee, G.F. Manso, Deep learning's diminishing returns: the cost of improvement is becoming unsustainable, IEEE Spectr. 58 (10) (2021) 50–55.
[21] N.C. Thompson, K. Greenewald, K. Lee, G.F. Manso, The computational limits of deep learning, arXiv:2007.05558, 2020.
[22] Semiconductor Research Corporation, The decadal plan for semiconductors, https://www.src.org/about/decadal-plan/, 2021. (Accessed 19 March 2025).
[23] R. Landauer, Irreversibility and heat generation in the computing process, IBM J. Res. Dev. 5 (3) (1961) 183–191.
[24] A. Bérut, A. Arakelyan Artak Petrosyan, S. Ciliberto, R. Dillenschneider, E. Lutz, Experimental verification of Landauer's principle linking information and thermodynamics, Nature 483 (2012) 187–189.
[25] National Science & Technology Council, Pioneering the future advanced computing ecosystem: a strategic plan, https://www.nitrd.gov/pubs/Future-Advanced-Computing-Ecosystem-Strategic-Plan-Nov-2020.pdf, 2020.
[26] R. Bommasani, et al., On the opportunities and risks of foundation models, arXiv:2108.07258, 2022.
[27] Y. Chang, et al., A survey on evaluation of large language models, ACM Trans. Intell. Syst. Technol. 15 (3) (2024).
[28] B. Goertzel, Artificial general intelligence: concept, state of the art, and future prospects, J. Artif. Gen. Intell. 5 (1) (2014) 1–48.
[29] M. Roser, AI timelines: what do experts in artificial intelligence expect for the future?, in: Our World in Data, 2023, https://ourworldindata.org/ai-timelines.
[30] S.M. Thornton, S. Pan, S.M. Erlien, J.C. Gerdes, Incorporating ethical considerations into automated vehicle control, IEEE Trans. Intell. Transp. Syst. 18 (6) (2017) 1429–1439.
[31] S. Karnouskos, Self-driving car acceptance and the role of ethics, IEEE Trans. Eng. Manag. 67 (2) (2020) 252–265.
[32] M. Geisslinger, F. Poszler, J. Betz, C. Lütge, M. Lienkamp, Autonomous driving ethics: from trolley problem to ethics of risk, Philos. Technol. 34 (4) (2021) 1033–1055.
[33] G. Ras, N. Xie, M. van Gerven, D. Doran, Explainable deep learning: a field guide for the uninitiated, J. Artif. Intell. Res. 73 (2022).
[34] C. Szegedy, et al., Intriguing properties of neural networks, in: Y. Bengio, Y. LeCun (Eds.), ICLR 2014, 2014.
[35] Y. Zhang, et al., Siren's song in the AI ocean: a survey on hallucination in large language models, arXiv:2309.01219, 2023.
[36] A. Bastounis, A.C. Hansen, V. Vlačić, The mathematics of adversarial attacks in AI – why deep learning is unstable despite the existence of stable neural networks, arXiv:2109.06098, 2021.
[37] B. Adcock, N. Dexter, The gap between theory and practice in function approximation with deep neural networks, SIAM J. Math. Data Sci. 3 (2) (2021) 624–655.
[38] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, A. Madry, Adversarial examples are not bugs, they are features, in: NeurIPS 2019, Curran Associates Inc., Red Hook, NY, USA, 2019.
[39] N. Carlini, D. Wagner, Audio adversarial examples: targeted attacks on speech-to-text, in: SPW 2018, IEEE, 2018, pp. 1–7.
[40] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, A. Madry, Robustness may be at odds with accuracy, in: ICLR 2019, 2019.
[41] V. Antun, F. Renna, C. Poon, B. Adcock, A.C. Hansen, On instabilities of deep learning in image reconstruction and the potential costs of AI, Proc. Natl. Acad. Sci. 117 (2020) 088.
[42] S.-M. Moosavi-Dezfooli, A. Fawzi, P. Frossard, Deepfool: a simple and accurate method to fool deep neural networks, in: CVPR 2016, IEEE, 2016, pp. 2574–2582.
[43] A. Boulemtafes, A. Derhab, Y. Challal, A review of privacy-preserving techniques for deep learning, Neurocomputing 384 (2020) 21–45.
[44] Y. He, G. Meng, K. Chen, X. Hu, J. He, Towards security threats of deep learning systems: a survey, IEEE Trans. Softw. Eng. 48 (5) (2022) 1743–1770.
[45] X. Liu, et al., Privacy and security issues in deep learning: a survey, IEEE Access 9 (2021) 4566–4593.
[46] F. Mireshghallah, M. Taram, P. Vepakomma, A. Singh, R. Raskar, H. Esmaeilzadeh, Privacy in deep learning: a survey, arXiv:2004.12254, 2020.
[47] O. Willers, S. Sudholt, S. Raafatnia, S. Abrecht, Safety concerns and mitigation approaches regarding the use of deep learning in safety-critical perception tasks, in: A. Casimiro, F. Ortmeier, E. Schoitsch, F. Bitsch, P. Ferreira (Eds.), SAFECOMP 2020 Workshops, Springer International Publishing, Cham, 2020, pp. 336–350.
[48] G. Fettweis, H. Boche, On 6G and trustworthiness, Commun. ACM 65 (4) (2022) 48–49.
[49] EU Artificial Intelligence Act, High-level summary of the AI Act, https://artificialintelligenceact.eu/high-level-summary/. (Accessed 19 March 2025).
[50] G7 Hiroshima Summit 2023, G7 hiroshima leaders' communiqué, https://www.mofa.go.jp/files/100506878.pdf, 2023.
[51] H. Tan, A brief history and technical review of the expert system research, IOP Conf. Ser., Mater. Sci. Eng. 242 (1) (2017).
[52] H. Boche, A. Fono, G. Kutyniok, Limitations of deep learning for inverse problems on digital hardware, IEEE Trans. Inf. Theory 69 (12) (2023) 7887–7908.
[53] H. Boche, A. Fono, G. Kutyniok, Inverse problems are solvable on real number signal processing hardware, Appl. Comput. Harmon. Anal. 74 (2025).
[54] A.M. Turing, On computable numbers, with an application to the Entscheidungs-problem, Proc. Lond. Math. Soc. s2–42 (1) (1936) 230–265.
[55] L. Blum, M. Shub, S. Smale, On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines, Bull., New Ser., Am. Math. Soc. 21 (1) (1989) 1–46.
[56] B. Ulmann, Analog Computing, De Gruyter Oldenbourg, Berlin, Boston, 2022.
[57] W. Haensch, T. Gokmen, R. Puri, The next generation of deep learning hardware: analog computing, Proc. IEEE 107 (1) (2019) 108–122.
[58] R. Hamerly, S. Bandyopadhyay, D. Englund, Asymptotically fault-tolerant programmable photonics, Nat. Commun. 13 (1) (2022).
[59] M. Miscuglio, et al., Approximate analog computing with metatronic circuits, Commun. Phys. 4 (1) (2021) 196.
[60] H.G. Rice, Classes of recursively enumerable sets and their decision problems, Trans. Am. Math. Soc. 74 (2) (1953) 358–366.
[61] E. Clarke, O. Grumberg, D. Peled, D. Peled, Model Checking (the Cyber-Physical Systems Series), MIT Press, 1999.
[62] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, A. Vladu, Towards deep learning models resistant to adversarial attacks, in: ICLR, 2018.
[63] N. Carlini, D. Wagner, Towards evaluating the robustness of neural networks, in: IEEE Symposium on Security and Privacy, IEEE, 2017, pp. 39–57.
[64] Z.C. Lipton, The mythos of model interpretability, Commun. ACM 61 (10) (2018) 36–43.
[65] L.H. Gilpin, D. Bau, B.Z. Yuan, A. Bajwa, M. Specter, L. Kagal, Explaining explanations: an overview of interpretability of machine learning, in: 2018 IEEE 5th International Conference on Data Science and Advanced Analytics, 2018, pp. 80–89.
[66] W. Samek, T. Wiegand, K.-R. Müller, Explainable artificial intelligence: understanding, visualizing and interpreting deep learning models, arXiv:1708.08296, 2017.
[67] M.T. Ribeiro, S. Singh, C. Guestrin, "Why should I trust you?": explaining the predictions of any classifier, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Association for Computing Machinery, New York, NY, USA, 2016, pp. 1135–1144.
[68] F. Doshi-Velez, B. Kim, Towards a rigorous science of interpretable machine learning, arXiv:1702.08608, 2017.
[69] J.A. Kroll, et al., Accountable algorithms, Univ. Pa. Law Rev. 165 (2017) 633–706.
[70] C. Olah, Mechanistic interpretability, variables, and the importance of interpretable bases, https://www.transformer-circuits.pub/2022/mech-interp-essay, 2022. (Accessed 19 March 2025).

[71] L. Kästner, B. Crook, Explaining AI through mechanistic interpretability, [Online]. Available: http://philsci-archive.pitt.edu/22747/, 2023.
[72] J. Angwin, J. Larson, S. Mattu, L. Kirchner, Machine bias, https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing, 2016. (Accessed 19 March 2025).
[73] M. Veale, R. Binns, Fairer machine learning in the real world: mitigating discrimination without collecting sensitive data, Big Data Soc. 4 (2) (2017).
[74] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, A. Galstyan, A survey on bias and fairness in machine learning, ACM Comput. Surv. 54 (6) (2021).
[75] S. Vadhan, The complexity of differential privacy, in: Y. Lindell (Ed.), Tutorials on the Foundations of Cryptography: Dedicated to Oded Goldreich, Springer International Publishing, Cham, 2017, pp. 347–450.
[76] S. Barocas, M. Hardt, A. Narayanan, Fairness in Machine Learning: Limitations and Opportunities, MIT Press, 2023, [Online]. Available: https://fairmlbook.org/.
[77] N. Diakopoulos, Accountability in algorithmic decision making, Commun. ACM 59 (2) (2016) 56–62.
[78] S. Wachter, B. Mittelstadt, L. Floridi, Why a right to explanation of automated decision-making does not exist in the general data protection regulation, Int. Data Priv. Law 7 (2) (2017) 76–99.
[79] D.K. Citron, F. Pasquale, The scored society: due process for automated predictions, Wash. Law Rev. 89 (2014) 1.
[80] S. Fazelpour, Z.C. Lipton, Algorithmic fairness from a non-ideal perspective, in: Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society, 2020, pp. 57–63.
[81] L. Floridi, et al., AI4People - an ethical framework for a good AI society: opportunities, risks, principles, and recommendations, Minds Mach. 28 (4) (2018) 689–707.
[82] A. Biondi, F. Nesti, G. Cicero, D. Casini, G. Buttazzo, A safe, secure, and predictable software architecture for deep learning in safety-critical systems, IEEE Embed. Syst. Lett. 12 (3) (2020) 78–82.
[83] H. Zhang, et al., Towards stable and efficient training of verifiably robust neural networks, in: ICLR 2020, 2020.
[84] M. Mirman, A. Hägele, P. Bielik, T. Gehr, M. Vechev, Robustness certification with generative models, in: SIGPLAN PLDI 2021, Association for Computing Machinery, New York, NY, USA, 2021, pp. 1141–1154.
[85] D.V. Christensen, et al., 2022 roadmap on neuromorphic computing and engineering, Neuromorph. Comput. Eng. 2 (2) (2022) 022501.
[86] European Commission, AI Act – shaping Europe's digital future, https://digital-strategy.ec.europa.eu/policies/regulatory-framework-ai. (Accessed 19 March 2025).
[87] European Commission, European centre for algorithmic transparency, https://algorithmic-transparency.ec.europa.eu/about_en. (Accessed 19 March 2025).
[88] L. Blum, F. Cucker, M. Shub, S. Smale, Complexity and Real Computation, Springer Verlag, New York, 1998.
[89] B.J. Copeland, The church-Turing thesis, in: E.N. Zalta (Ed.), Summer 2020, the Stanford Encyclopedia of Philosophy, Metaphysics Research Lab, Stanford University, 2020.
[90] S. Kleene, General recursive functions of natural numbers, Math. Ann. 112 (1936) 727–742.
[91] A.M. Turing, Computability and lambda-definability, J. Symb. Log. 2 (4) (1937) 153–163.
[92] J. Avigad, V. Brattka, Computability and analysis: the legacy of Alan Turing, in: R. Downey (Ed.), Turing's Legacy: Developments from Turing's Ideas in Logic, in: Lecture Notes in Logic, Cambridge University Press, 2014, pp. 1–47.
[93] M.B. Pour-El, J.I. Richards, Computability in Analysis and Physics (Perspectives in Logic), Cambridge University Press, 2017.
[94] K. Weihrauch, Computable Analysis: An Introduction, Springer-Verlag, Berlin, Heidelberg, 2000.
[95] M.J. Colbrook, V. Antun, A.C. Hansen, The difficulty of computing stable and accurate neural networks: on the barriers of deep learning and Smale's 18th problem, Proc. Natl. Acad. Sci. 119 (12) (2022).
[96] L. Grozinger, et al., Pathways to cellular supremacy in biocomputing, Nat. Commun. 10 (2019).
[97] S.K. Esser, R. Appuswamy, P. Merolla, J.V. Arthur, D.S. Modha, Backpropagation for energy-efficient neuromorphic computing, in: C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, R. Garnett (Eds.), NIPS 2015, vol. 28, Curran Associates, Inc., 2015.
[98] J.D. Smith, et al., Neuromorphic scaling advantages for energy-efficient random walk computations, Nat. Electron. 5 (2) (2022) 102–112.
[99] A. Rao, P. Plank, A. Wild, W. Maass, A long short-term memory for AI applications in spike-based neuromorphic hardware, Nat. Mach. Intell. 4 (5) (2022) 467–479.
[100] D. Marković, A. Mizrahi, D. Querlioz, J. Grollier, Physics for neuromorphic computing, Nat. Rev. Phys. 2 (9) (2020) 499–510.
[101] P. Blouw, C. Eliasmith, Event-driven signal processing with neuromorphic computing systems, in: ICASSP 2020, IEEE, 2020, pp. 8534–8538.
[102] C. Schuman, S. Kulkarni, M. Parsa, J. Mitchell, P. Date, B. Kay, Opportunities for neuromorphic computing algorithms and applications, Nat. Comput. Sci. 2 (2022) 10–19.
[103] W. Aspray, John Von Neumann and the Origins of Modern Computing, MIT Press, Cambridge, MA, USA, 1990.
[104] J. Backus, Can programming be liberated from the von Neumann style? A functional style and its algebra of programs, Commun. ACM 21 (8) (1978) 613–641.
[105] D. Efnusheva, A. Cholakoska, A. Tentov, A survey of different approaches for overcoming the processor-memory bottleneck, Int. J. Comput. Sci. Inf. Technol. 9 (2) (2017) 151–163.
[106] I. Boybat, et al., Temperature sensitivity of analog in-memory computing using phase-change memory, in: IEDM 2021, IEEE, 2021.
[107] G. Karunaratne, M. Le Gallo, G. Cherubini, L. Benini, A. Rahimi, A. Sebastian, In-memory hyperdimensional computing, Nat. Electron. 3 (6) (2020) 327–337.
[108] A. Sebastian, M. Le Gallo, R. Khaddam-Aljameh, E. Eleftheriou, Memory devices and applications for in-memory computing, Nat. Nanotechnol. 15 (7) (2020) 529–544.
[109] M. Payvand, M.V. Nair, L.K. Müller, G. Indiveri, A neuromorphic systems approach to in-memory computing with non-ideal memristive devices: from mitigation to exploitation, Faraday Discuss. 213 (2019) 487–510.
[110] Á. Papp, W. Porod, G. Csaba, Nanoscale neural network using non-linear spin-wave interference, Nat. Commun. 12 (2021).
[111] K.F. Wagenbauer, C. Sigl, H. Dietz, Gigadalton-scale shape-programmable DNA assemblies, Nature 552 (2017) 78–83.
[112] P. Poirazi, A. Papoutsi, Illuminating dendritic function with computational models, Nat. Rev. Neurosci. 21 (2020) 303–321.
[113] L.G. Wright, et al., Deep physical neural networks trained with backpropagation, Nature 601 (2022) 549–555.
[114] A.J. Daley, et al., Practical quantum advantage in quantum simulation, Nature 607 (2022) 667–676.
[115] S. Flannigan, et al., Propagation of errors and quantitative quantum simulation with quantum advantage, Quantum Sci. Technol. 7 (4) (2022).
[116] D. Elkouss, D. Pérez-García, Memory effects can make the transmission capability of a communication channel uncomputable, Nat. Commun. 9 (1) (2018).
[117] R.F. Schaefer, H. Boche, H.V. Poor, Turing meets Shannon: on the algorithmic computability of the capacities of secure communication systems (invited paper), in: SPAWC 2019, IEEE, 2019, pp. 1–5.
[118] H. Boche, V. Pohl, On the algorithmic solvability of spectral factorization and applications, IEEE Trans. Inf. Theory 66 (7) (2020) 4574–4592.
[119] H. Boche, U.J. Mönich, Turing computability of Fourier transforms of bandlimited and discrete signals, IEEE Trans. Signal Process. 68 (2020) 532–547.
[120] H. Boche, U.J. Mönich, On the solvability of the peak value problem for bandlimited signals with applications, IEEE Trans. Signal Process. 69 (2021) 103–118.
[121] M.B. Pour-El, N. Zhong, The wave equation with computable initial data whose unique solution is nowhere computable, Math. Log. Q. 43 (4) (1997) 499–509.
[122] H. Boche, V. Pohl, Turing meets circuit theory: not every continuous-time LTI system can be simulated on a digital computer, IEEE Trans. Circuits Syst. I, Regul. Pap. 67 (12) (2020) 5051–5064.
[123] H. Boche, R.F. Schaefer, H. Vincent Poor, Real number signal processing can detect denial-of-service attacks, in: ICASSP 2021, IEEE, 2021, pp. 4765–4769.
[124] H. Boche, M. Cai, H.V. Poor, R.F. Schaefer, Detectability of denial-of-service attacks on arbitrarily varying classical-quantum channels, in: ISIT 2021, IEEE, 2021, pp. 912–917.

[125] H. Boche, Y. Böck, C. Deppe, Deciding the problem of remote state estimation via noisy communication channels on real number signal processing hardware, in: ICC 2022, IEEE, 2022, pp. 4510–4515.
[126] L. Blum, Computing over the reals: where Turing meets Newton, Not. Am. Math. Soc. 51 (9) (2004) 1024–1034.
[127] J.D. Bekenstein, Universal upper bound on the entropy-to-energy ratio for bounded systems, Phys. Rev. D 23 (1981) 287–298.
[128] I. Daubechies, M. Defrise, C. De Mol, An iterative thresholding algorithm for linear inverse problems with a sparsity constraint, Commun. Pure Appl. Math. 57 (11) (2004) 1413–1457.
[129] S.J. Wright, R.D. Nowak, M.A.T. Figueiredo, Sparse reconstruction by separable approximation, IEEE Trans. Signal Process. 57 (7) (2009) 2479–2493.
[130] S. Cotter, B. Rao, K. Engan, K. Kreutz-Delgado, Sparse solutions to linear inverse problems with multiple measurement vectors, IEEE Trans. Signal Process. 53 (7) (2005) 2477–2488.
[131] I. Selesnick, Sparse regularization via convex analysis, IEEE Trans. Signal Process. 65 (17) (2017) 4481–4494.
[132] E. Candes, T. Tao, Decoding by linear programming, IEEE Trans. Inf. Theory 51 (12) (2005) 4203–4215.
[133] E. Candes, J. Romberg, T. Tao, Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information, IEEE Trans. Inf. Theory 52 (2) (2006) 489–509.
[134] E.J. Candes, T. Tao, Near-optimal signal recovery from random projections: universal encoding strategies?, IEEE Trans. Inf. Theory 52 (12) (2006) 5406–5425.
[135] D. Donoho, Compressed sensing, IEEE Trans. Inf. Theory 52 (4) (2006) 1289–1306.
[136] S. Ji, Y. Xue, L. Carin, Bayesian compressive sensing, IEEE Trans. Signal Process. 56 (6) (2008) 2346–2356.
[137] M.F. Duarte, Y.C. Eldar, Structured compressed sensing: from theory to applications, IEEE Trans. Signal Process. 59 (9) (2011) 4053–4085.
[138] M. Elad, Optimized projections for compressed sensing, IEEE Trans. Signal Process. 55 (12) (2007) 5695–5702.
[139] E.J. Candès, J.K. Romberg, T. Tao, Stable signal recovery from incomplete and inaccurate measurements, Commun. Pure Appl. Math. 59 (8) (2006) 1207–1223.
[140] S.S. Chen, D.L. Donoho, M.A. Saunders, Atomic decomposition by basis pursuit, SIAM J. Sci. Comput. 20 (1) (1998) 33–61.
[141] J. Tropp, Just relax: convex programming methods for identifying sparse signals in noise, IEEE Trans. Inf. Theory 52 (3) (2006) 1030–1051.
[142] A. Belloni, V. Chernozhukov, L. Wang, Square-root lasso: pivotal recovery of sparse signals via conic programming, Biometrika 98 (4) (2011) 791–806.
[143] B. Zhu, J.Z. Liu, S.F. Cauley, B.R. Rosen, M.S. Rosen, Image reconstruction by domain-transform manifold learning, Nature 555 (2018) 487–492.
[144] S.R. Arridge, P. Maass, O. Öktem, C.-B. Schönlieb, Solving inverse problems using data-driven models, Acta Numer. 28 (2019) 1–174.
[145] T.A. Bubba, et al., Learning the invisible: a hybrid deep learning-shearlet framework for limited angle computed tomography, Inverse Probl. 35 (6) (2019).
[146] Y. Yang, J. Sun, H. Li, Z. Xu, Deep ADMM-net for compressive sensing MRI, in: D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, R. Garnett (Eds.), NIPS 2016, vol. 29, Curran Associates, Inc., 2016.
[147] K. Hammernik, et al., Learning a variational network for reconstruction of accelerated MRI data, Magn. Reson. Med. 79 (6) (2018) 3055–3071.
[148] C. Chen, Q. Chen, J. Xu, V. Koltun, Learning to see in the dark, in: CVPR 2018, IEEE, 2018.
[149] Y. Rivenson, Z. Göröcs, H. Günaydin, Y. Zhang, H. Wang, A. Ozcan, Deep learning microscopy, Optica 4 (11) (2017) 1437–1443.
[150] M. Araya-Polo, J. Jennings, A. Adler, T. Dahlke, Deep-learning tomography, Lead. Edge 37 (1) (2018) 58–66.
[151] G. Ongie, A. Jalal, C.A. Metzler, R.G. Baraniuk, A.G. Dimakis, R. Willett, Deep learning techniques for inverse problems in imaging, IEEE J. Sel. Areas Inf. Theory 1 (1) (2020) 39–56.
[152] G. Cybenko, Approximation by superpositions of a sigmoidal function, Math. Control Signals Syst. 2 (1989) 03.
[153] K. Hornik, Approximation capabilities of multilayer feedforward networks, Neural Netw. 4 (1991) 251–257.
[154] H. Bölcskei, P. Grohs, G. Kutyniok, P. Petersen, Optimal approximation with sparsely connected deep neural networks, SIAM J. Math. Data Sci. 1 (2019) 8–45.
[155] R. Eldan, O. Shamir, The power of depth for feedforward neural networks, in: COLT 2016 Proceedings, in: Proceedings of Machine Learning Research, vol. 49, PMLR, 2016, pp. 907–940.
[156] T. Poggio, H. Mhaskar, L. Rosasco, B. Miranda, Q. Liao, Why and when can deep - but not shallow - networks avoid the curse of dimensionality: a review, Int. J. Autom. Comput. 14 (2017) 503–519.
[157] I. Daubechies, R. DeVore, S. Foucart, B. Hanin, G. Petrova, Nonlinear approximation and (deep) ReLU networks, Constr. Approx. 55 (2022) 127–172.
[158] R. Gribonval, G. Kutyniok, M. Nielsen, F. Voigtlaender, Approximation spaces of deep neural networks, Constr. Approx. 55 (2022) 259–367.
[159] D. Yarotsky, Error bounds for approximations with deep ReLU networks, Neural Netw. 94 (2017) 103–114.
[160] Y. LeCun, Y. Bengio, Convolutional networks for images, speech, and time series, in: The Handbook of Brain Theory and Neural Networks, MIT Press, Cambridge, MA, USA, 1998, pp. 255–258.
[161] X. Lv, G. Bi, C. Wan, The group lasso for stable recovery of block-sparse signal representations, IEEE Trans. Signal Process. 59 (4) (2011) 1371–1382.
[162] A. Bastounis, A.C. Hansen, V. Vlačić, The extended Smale's 9th problem – on computational barriers and paradoxes in estimation, regularisation, computer-assisted proofs and learning, arXiv:2110.15734, 2021.