ON GENERALISED HARDNESS OF APPROXIMATION, HALLUCINATIONS, INSTABILITY AND TRUSTWORTHINESS IN AI FOR INVERSE PROBLEMS

ALEXANDER BASTOUNIS AND ANDERS C. HANSEN

ABSTRACT. AI techniques are transforming inverse problems, much as they are reshaping the sciences more broadly. This transformation brings substantial opportunities alongside serious challenges, foremost among them the trustworthiness of AI-based methods. Advancing trustworthiness requires a precise understanding of the phenomena that undermine it. In this paper, we analyse three such phenomena – *hallucinations*, *generalised hardness of approximation (GHA)*, and *instability*. Although these issues are widespread in modern AI, we examine their specific manifestations in inverse problems. We document that, owing to these phenomena, there are fundamental limits, stability-accuracy trade-offs, and 'no free lunch' results for inverse problem reconstruction methods. Only by recognising and characterising these mechanisms can we make progress towards trustworthy AI for inverse problems; accordingly, we distil principles and recommendations to guide the development of trustworthy AI methods.

CONTENTS

1. Introduction 2
2. Machine learning in inverse problems and imaging 4
2.1. Optimisation in machine learning and inverse problems 5
2.2. Randomised algorithms in machine learning, optimisation and inverse problems 7
3. Generalised hardness of approximation (GHA) and phase transitions 7
3.1. GHA in computations – Including randomised algorithms 9
4. GHA in optimisation and machine learning – Smale's 9th problem 9
4.1. Inexact input – a realistic model of computation 10
4.2. Consequences of GHA in ML – Proving existence of neural networks is not enough 12
5. GHA in deep learning for inverse problems 13
5.1. Optimality of neural networks for underdetermined linear systems 13
5.2. Consequences of GHA in inverse problems – Accuracy-stability trade-off 17
6. Hallucinations in AI for inverse problems – Detail transfer 17
6.1. Coping with hallucinations coming from detail transfer – No free lunch 19
7. The instability phenomenon in modern AI 19
7.1. Instability in classification problems & adversarial attacks 19
7.2. Instabilities in inverse problems and imaging 22
7.3. Coping with instabilities in AI methods – The accuracy-stability trade-off 23
References 24

2020 *Mathematics Subject Classification.* 65J22, 15A29, 68T07 (primary) and 68U10, 03Dxx, 68W20 (secondary).

Key words and phrases. Inverse problems, Artificial Intelligence, hallucinations, trustworthiness, generalised hardness of approximation, phase transitions, foundations of computational mathematics.

1. INTRODUCTION

AI is reshaping the sciences – spanning scientific computing and applied/computational mathematics – and, more broadly, society. Inverse problems arise throughout these areas and are inherently interdisciplinary; as AI transforms the surrounding disciplines, it necessarily transforms inverse problems as well. The broader challenges of AI in science – above all, the need for trustworthy methods – therefore apply with equal force here. Creating trustworthy AI techniques is highly challenging and one of the final frontiers in AI research, and this paper is thus organised around the following fundamental question:

Q1: How can we make AI techniques trustworthy for inverse problems?

In order to address the above question, we must identify the different phenomena that make this problem challenging. There are several of these phenomena, and we will focus on the following three issues:

(i) *Hallucinations* – The phenomenon where plausible but wrong and deceptive outputs are produced by an AI method.
(ii) *Generalised hardness of approximation (GHA)* – The phenomenon where algorithms can easily compute a solution to a problem with accuracy $\epsilon > \epsilon_1$ (the approximation threshold), but it becomes hard or impossible to compute a solution with accuracy $\epsilon < \epsilon_1$.
(iii) *Instability* – The phenomenon where small perturbations of inputs to an algorithm produce huge differences in the output.

We do not attempt to fully resolve question Q1 above on achieving trustworthy AI for inverse problems. As the theorems below indicate, fundamental barriers preclude simple remedies – a 'no free lunch' phenomenon. Consequently, trade-offs are inevitable, and progress likely requires methods that can acknowledge uncertainty and simply say 'I don't know'. We elaborate on three central issues below.

Hallucinations – the phenomenon where an AI produces incorrect yet highly plausible outputs or provides false reasoning – are one of the most significant challenges facing modern AI systems. They are particularly delicate because they are difficult to detect, yet frequently occur when AI is used in scientific and industrial sectors, as well as in society more broadly. Their presence is a major obstacle to the development of fully trustworthy AI, which raises concerns in safety-critical applications:

> "*The most serious issue when applying deep learning for discovery is that of hallucination. [...] These hallucinations are deceptive artifacts that appear highly plausible in the absence of contradictory information and can be challenging, if not impossible, to detect.*" — Nature Methods (2019) [21].

Moreover, it has become apparent that hallucinations occur in most, if not all, applications of AI-inspired techniques. This includes medical imaging [33, 80] (see Figure 1), microscopy [21], as well as inverse problems more generally [56], chatbots and large language models (LLM) [92], computer vision [79], automated medical diagnosis, and image classification [52] (see Figure 3). Reflecting its widespread impact on current scientific and public discourse, the Cambridge Dictionary selected 'hallucinate' as its 2023 Word of the Year [42].

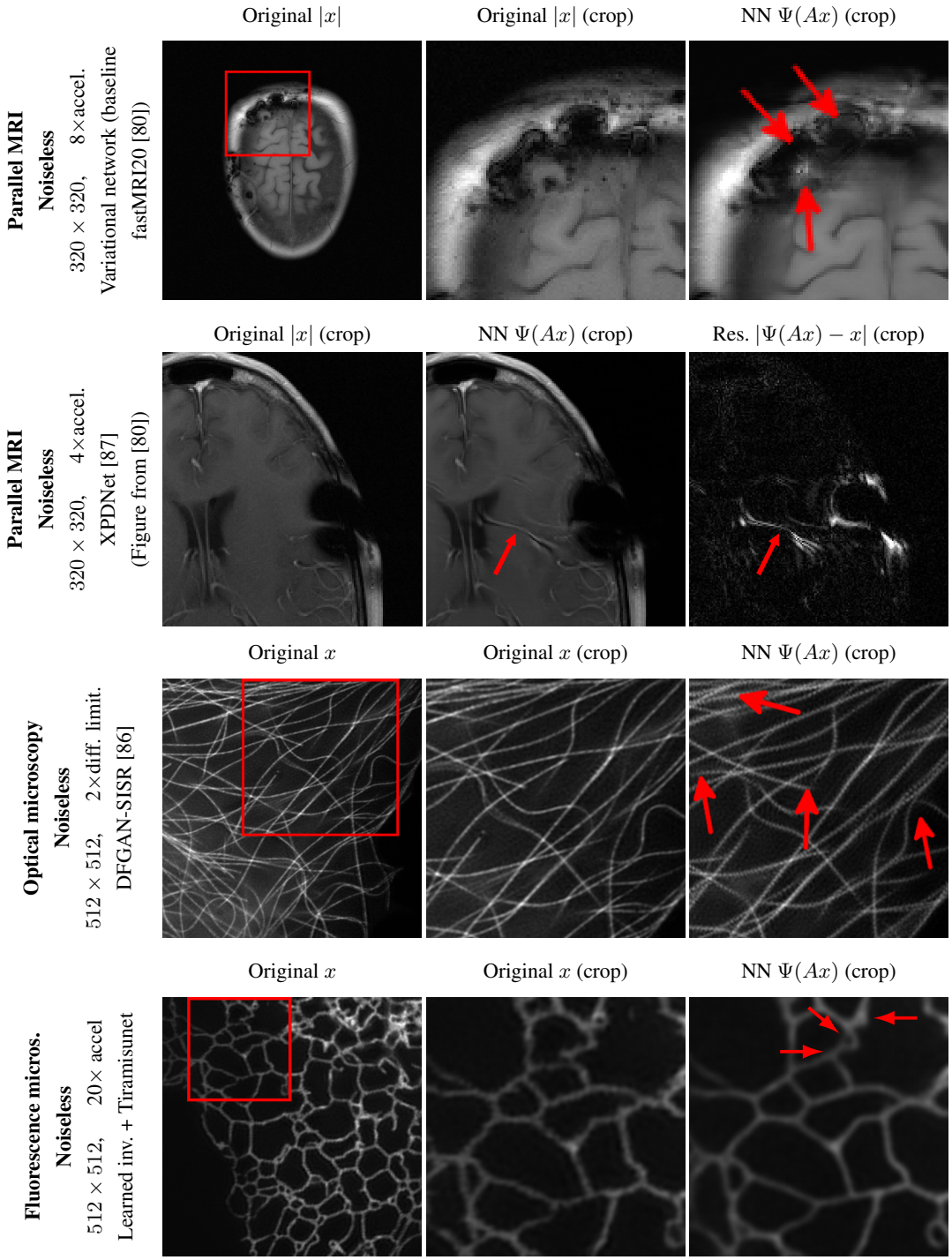


FIGURE 1. (**AI-generated hallucinations in different imaging modalities**). Trained NNs $\Psi\colon \mathbb{C}^m \to \mathbb{R}^N$ for different imaging modalities generate hallucinations, i.e., realistic-looking artifacts, when evaluated on test data. Note that m, N and A (from (2.1)) vary between the experiments (different imaging modalities). In the first three rows we consider trained NNs from the cited publications. In row four, we trained a NN using data from [86]. For more information on the training procedure, see [56].

Generalised hardness of approximation (GHA) [8, 14, 15, 17, 20, 41, 48, 53] is the phenomenon whereby one can efficiently compute an ϵ-approximation to a solution for any $\epsilon > \epsilon_1 > 0$, but once ϵ drops below the threshold ϵ_1, the task abruptly becomes hard or impossible – e.g., not solvable in polynomial time or non-computable. This phenomenon is widespread in optimisation used in machine learning and also training of AI methods. Moreover, this phase transition phenomenon arises when training neural networks (NNs) to solve inverse problems. Specifically, for any underdetermined linear inverse problem with nontrivial nullspace, there exists a family of training sets Ω such that for each $\mathcal{T} \in \Omega$ an optimal NN exists, yet it is computable only up to accuracy $\epsilon_1 > 0$. For $\epsilon < \epsilon_1$, computing such NNs is not merely intractable but impossible: no randomized algorithm succeeds with probability exceeding $1/2$. Moreover, despite the existence of a stable optimal NN, any attempt to compute it to accuracy below twice the threshold, $2\epsilon_1$, yields an unstable network (see §5).

Instability is the phenomenon when a small perturbation to the input of an AI method makes huge changes to the output. Neural networks become universally unstable (non-robust) when trained to solve such problems in virtually any application [3,5,6,19,36–38,52,64,65,95], making the non-robustness issue one of the fundamental issues in AI. The vast literature on this issue – often referring to the instability phenomenon as a vulnerability to adversarial attacks – has not been able to solve the problem. Inverse problems are not an exception, and instabilities have been reported throughout various applications such as Magnetic Resonance Imaging (MRI), Computerised Tomography (CT), etc [6, 56].

Notation. We denote by $\mathcal{B}_\delta^p(x)$ the open ball of radius δ centred at x in ℓ^p and set $\overline{\mathcal{B}}_\delta^p(x)$ as its closure. Let $L, d, m, N \in \mathbb{N}$ and $\mathbf{N} := (N_L, N_{L-1}, \dots, N_1, N_0)$ be a vector in $\mathbb{N}^{L+1}$ with $N_0 = m, N_L = N$. A (feedforward) *Neural Network (NN) with dimensions* $(\mathbf{N}, L)$ is a map $\phi : \mathbb{R}^m \to \mathbb{R}^N$ such that $\phi = W^L \sigma W^{L-1} \sigma W^{L-2} \dots \sigma W^1$ where, for $l = 1, 2, \dots, L$, the map $W^l : \mathbb{R}^{N_{l-1}} \to \mathbb{R}^{N_l}$ is affine i.e. $W^l x^l = A^l x^l + b^l$ with $b^l \in \mathbb{R}^{N_l}$ and $A^l \in \mathbb{R}^{N_l \times N_{l-1}}$. The map $\sigma : \mathbb{R} \to \mathbb{R}$ is interpreted as a coordinate-wise map and is called the *activation function*, usually chosen to be continuous and non-polynomial [3]. For $\epsilon > 0$ and $y \in \mathbb{C}^m$, we define the *local ϵ-Lipschitz constant* of a mapping $\Phi \colon \mathbb{C}^m \to \mathbb{C}^N$ as

$$\mathcal{L}^\epsilon(\Phi, y) = \sup_{\substack{z \in \mathbb{C}^m \\ 0 < \|z-y\|_p \leq \epsilon}} \frac{\|\Phi(z) - \Phi(y)\|_p}{\|z - y\|_p}, \text{ where } \|\cdot\|_p \text{ is the } p \text{ norm.}$$

We also define the *global Lipschitz constant* as $\mathcal{L}(\Phi) = \sup_{y,z \in \mathbb{C}^m,\, y \neq z} \frac{\|\Phi(z)-\Phi(y)\|_p}{\|z-y\|_p}$. We denote the set of neural networks with fixed dimensions $\mathbf{N}$ and L by $\mathcal{NN}(\mathbf{N}, L)$. For $m, N \in \mathbb{N}$, we also denote the set of all neural networks from $\mathbb{R}^m \to \mathbb{R}^N$ by $\mathcal{NN}_{m,N}$.

2. Machine learning in inverse problems and imaging

Over the past decade, AI techniques have markedly advanced inverse problems and imaging [3, 21, 59, 67, 78, 84, 86, 99]. As with any powerful tool, careful deployment is essential: AI methods can hallucinate or produce other unexpected errors [6, 21, 56, 73, 80]. A recent *SIAM News* article, 'Learning in Image Reconstruction: A Cautionary Tale,' [33] offers a careful discussion of these points. Such considerations are particularly important in safety-critical settings such as medical imaging, where reliability is paramount and errors may have significant consequences (see Figure 1). At the core of image reconstruction lies a canonical form of inverse

problem, typically modelled as:

$$\text{Given measurements } y = Ax + e, \text{ recover } x. \tag{2.1}$$

Here, $A \in \mathbb{C}^{m\times N}$ is the *sampling operator* (also known as the *measurement matrix*), $y \in \mathbb{C}^m$ represents the observed *measurements*, $e \in \mathbb{C}^m$ denotes *measurement noise*, and $x \in \mathbb{C}^N$ is the unknown signal or image to be recovered – often a vectorized form of a 2D or 3D image.

Despite its apparent simplicity, equation (2.1) effectively models a wide range of imaging problems. However, these problems are frequently *ill-posed*, particularly when the number of measurements is limited or the measurement process is ill-conditioned. For instance, in applications like undersampled MRI, we often have $\mathrm{rank}(A) < N$, which implies that the null space $\mathcal{N}(A)$ is non-trivial. In fact, whenever $m < N$, there are infinitely many possible solutions consistent with the data. In other scenarios, such as image debluring, A may have full rank but be *ill-conditioned*, making the solution highly sensitive to noise due to the numerical instability.

To solve the inverse problem (2.1), the aim is to construct a *reconstruction map*:

$$\Psi : \mathbb{C}^m \to \mathbb{C}^N, \text{ such that } \Psi(Ax) \approx x, \quad x \in \mathcal{M}_1. \tag{2.2}$$

One normally assumes that the desired images x belong to some set $\mathcal{M}_1 \subset \mathbb{C}^N$, which is sometimes referred to as a *model class* or *image manifold*. AI-based methods are *data-driven*, meaning that a map Ψ is learned from a *training set* $\mathcal{T} = \{(x_i, y_i)\}_{i=1}^{\ell} \subset \mathbb{C}^N \times \mathbb{C}^m$, consisting of (noisy) measurements $y_i = Ax_i + e_i$ and the corresponding ground truth images x_i, which (implicitly) defines the model class $\mathcal{M}_1$. However, as discussed above, in practice the reconstruction method Ψ in (2.2) may produce unwanted results, leading to the fundamental and pressing question:

Q2: Why does the hallucination phenomenon happen in AI-based methods for solving inverse problems such as (2.1), and why can AI-based methods become unstable?

Understanding issues (i) and (iii) above are clearly central to answering question Q2. However, as we will see, issue (ii) on the phenomenon of GHA is also crucial, and this can be traced from its origins in optimisation. Because modern machine learning and AI are fundamentally optimisation-driven, answering question Q1 and Q2 requires an examination of GHA in optimisation.

2.1. **Optimisation in machine learning and inverse problems.** Convex optimisation provides a mainstay for machine learning including optimisation problems such as Linear Programming (LP), Semidefinite Programming (SDP), Basis Pursuit (BP) and LASSO (constrained and unconstrained). Classic examples include LP formulations of Markov decision processes [75, 85] and ℓ_1-regularized support vector machines that utilise LPs [103]. Similarly SDPs are used to learn matrices (e.g. to make recommender systems) via matrix completion [70, 76]. Moreover, LASSO is used in high-dimensional regression, classification and feature selection [63] Finally, all four of these convex optimisation problems are used in inverse problems via compressed sensing and low-rank matrix recovery [3].

The aforementioned optimisation problems are defined as follows:

(i) Linear Programming (LP)

$$\Xi(y, A) := \underset{x}{\mathrm{argmin}} \langle x, c\rangle \text{ subject to } Ax = y, \quad x \geq 0, \tag{2.3}$$

(ii) Basis Pursuit (BP)

$$\Xi(y,A) := \operatorname*{argmin}_x \|x\|_1 \text{ subject to } \|Ax-y\|_2 \leq \delta, \qquad \delta \in [0,1], \tag{2.4}$$

(iii) Unconstrained Lasso (UL)

$$\Xi(y,A) := \operatorname*{argmin}_x \|Ax-y\|_2^2 + \lambda \|x\|_1, \qquad \lambda \in (0,1], \tag{2.5}$$

(iv) Constrained Lasso (CL)

$$\Xi(y,A) := \operatorname*{argmin}_x \|Ax-y\|_2 \text{ subject to } \|x\|_1 \leq \tau, \qquad \tau > 0, \tag{2.6}$$

(v) Semidefinite Programming (SDP)

$$\begin{aligned}\Xi(b_1,b_2,\ldots,b_m,A_1,A_2,\ldots,A_m) := \operatorname*{argmin}_{X\in\mathbb{S}^n} \langle C,X\rangle_{\mathbb{S}^n} \text{ subject to } \langle A_k,X\rangle_{\mathbb{S}^n} = b_k,\\ X \succeq 0,\, k=1,\ldots,m.\end{aligned} \tag{2.7}$$

In the above notation we have $A \in \mathbb{R}^{m\times N}, y \in \mathbb{R}^m, c \in \mathbb{R}^N$. For SDP, the notation is

$$C, A_k \in \mathbb{S}^n \text{ (real } n\times n \text{ symmetric matrices)}, \quad b_k \in \mathbb{R}, \quad \langle C,X\rangle_{\mathbb{S}^n} = \operatorname{trace}(C^T X).$$

Important note! In machine learning and inverse problems these optimisation problems are often about computing a minimiser or an approximate minimiser. They are not about computing the objective function.

We have deliberately stated the above optimisation problems as 'argmin-problems'. That is, the function Ξ to be computed is the set-valued function of all the minimisers. However, in the case of several minimisers, we are only interested in approximating any minimiser and not every minimisers. The reason why this is important is that there is a crucial difference between computing approximations to minimisers compared to computing an approximation to the objective function in optimisation.

Remark 2.1 (Objective function vs minimisers). There is a very rich literature [25, 30, 34, 72, 81–83, 100, 101] on how to compute the objective function, and, in particular, the minimum value $f(x^*) = \min\{f(x) \,|\, x \in \mathcal{X}\}$, for some convex function $f : \mathbb{R}^d \to \mathbb{R}$, convex set $\mathcal{X} \subset \mathbb{R}^d$, and minimiser $x^* \in \mathcal{X}$. The traditional problem of interest is as follows. Given $\epsilon > 0$, compute an $x_\epsilon \in \mathbb{R}^d$ such that $f(x_\epsilon) - f(x^*) \leq \epsilon$. Note that

$$f(x_\epsilon) - f(x^*) \leq \epsilon \implies \|x_\epsilon - x^*\| \leq \epsilon.$$

Thus, computing x_ϵ such that

$$\|x_\epsilon - x^*\| \leq \epsilon,$$

which is the main goal in many machine learning and inverse problems, is a different and harder problem than computing an approximation to the objective function.

The reason why Remark 2.1 is central is because the GHA phenomenon happens almost universally when computing minimisers to the above problems. Moreover, GHA happens also when considering randomised algorithms, which represent a mainstay in modern machine learning.

2.2. **Randomised algorithms in machine learning, optimisation and inverse problems.** When addressing optimisation in machine learning, one must include randomised algorithms. Indeed, randomised algorithms in optimisation are woven through modern machine learning because injecting randomness may help explore large, complicated search spaces more efficiently; may potentially avoid worst-case behaviour of deterministic rules; and may cut computation time by working with randomly chosen samples instead of full data sets. In deep learning they are crucial: stochastic gradient descent (and its variants) uses mini-batches to scale to huge datasets, and may potentially help escaping saddle points and local minima which are present due to the use of non-convex loss functions.

Randomisation is also natural in AI methods for inverse problems. Note that in *learning-based* methods, one makes little or no explicit assumptions about $\mathcal{M}_1$ in (2.2), as for example done in more classical methods such as compressed sensing [3]. Instead, one is given a *training set* $\mathcal{T} = \{(x_i, y_i)\}_{i=1}^{\ell} \subset \mathbb{C}^N \times \mathbb{C}^m$, where $y_i = Ax_i + e_i$ are noisy measurements of x_i. One then uses $\mathcal{T}$ to learn a reconstruction map $\Psi \colon \mathbb{C}^m \to \mathbb{C}^N$. For instance, given a class $\mathcal{NN}$ of NNs $\varphi \colon \mathbb{C}^m \to \mathbb{C}^N$, a regularization term $J \colon \mathcal{NN} \to \mathbb{R}_{\geq 0}$ and a regularization parameter $\lambda \geq 0$, a standard choice involves (approximately) solving the regularized training problem

$$\min_{\varphi \in \mathcal{NN}} \frac{1}{|\mathcal{T}|} \sum_{(x,y)\in\mathcal{T}} \frac{1}{2}\|\varphi(y) - x\|^2 + \lambda J(\varphi). \tag{2.8}$$

Typically, the training optimisation problem (2.8) is attempted solved with randomised algorithms (e.g. stochastic gradient descent) – for the reasons discussed above. This discussion on optimisation in machine learning and inverse problems leads to the following key question:

Q3: How can one guarantee accurate computations of approximate minimisers to convex and non-convex optimisation problems – potentially using randomised algorithms?

It is impossible to answer to above question without understanding the highly delicate and intricate phenomenon of generalised hardness of approximation.

3. Generalised hardness of approximation (GHA) and phase transitions

GHA captures how the computational difficulty of achieving a prescribed accuracy depends on the accuracy parameter itself: as the tolerance ϵ varies, the difficulty of obtaining an ϵ-accurate solution can shift abruptly from easy-to-compute to intractable and potentially non-computable. More precisely, let

$$\Xi : \Omega \to \mathcal{M},$$

which is the desired function to compute approximately, where Ω is some set and $(\mathcal{M}, d)$ is a metric space. The associated ϵ-approximate computational problem is: given an instance $\iota \in \Omega$ and tolerance $\epsilon > 0$, compute $\xi \in \mathcal{M}$ such that $\operatorname{dist}(\xi, \Xi(\iota)) = \inf_{a\in\Xi(\iota)} d(\xi, a) \leq \epsilon$. GHA concerns how the difficulty of this task changes with ϵ.

Suppose we have an ϵ-approximate computational problem and two disjoint classes of computational problems, S_1 and S_2, corresponding to two different hardness/easiness regimes. Let $\epsilon_1 \geq \epsilon_2 > 0$ be thresholds and suppose that $S_1 \cap S_2 = \emptyset$. For example:

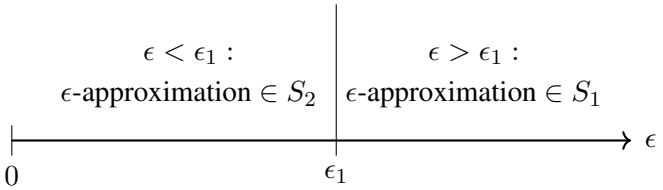
$$S_1 = P \quad \text{(solvable in polynomial time)}, \qquad S_2 = P^c \quad \text{(the complement of } P\text{)}.$$

We say that the problem exhibits an (S_1, S_2)-*phase transition* at (ϵ_1, ϵ_2) if:

$$\begin{aligned} &\text{The approximate computational problem} \in S_1, \quad \text{for } \epsilon > \epsilon_1, \\ &\text{The approximate computational problem} \in S_2, \quad \text{for } \epsilon < \epsilon_2. \end{aligned} \tag{3.1}$$

If $\epsilon_1 = \epsilon_2$, the transition is said to be *sharp*, and the common value ϵ_1 is called the *approximation threshold*. This sharp transition is illustrated schematically below:

Sharp phase transition at ϵ_1 in generalised hardness of approximation



(3.2)

A well established case of this phenomenon is hardness of approximation in computer science, in particular, in discrete optimisation.

Example 3.1 (Hardness of approximation in computer science)**.** GHA extends the classical notion of hardness of approximation (HA) [7,8,20,51,61,94] from discrete complexity theory to general settings in computational mathematics. The much celebrated PCP theorem [7–9, 51] implies that there are large collections of discrete combinatorial optimisation problems for which there is a sharp (S_1, S_2)-phase transition at $\epsilon_1 \geq 0$ (where ϵ_1 depends on the problem), where

$$S_1 = P, \quad S_2 = P^c.$$

GHA abstracts and generalises this threshold behaviour beyond combinatorics – e.g., to continuous optimisation, numerical linear algebra, and spectral problems. The strictness of the boundary, in the discrete combinatorial setting, typically hinges on the P vs. NP question: $\epsilon_1 > 0$ under the prevailing assumption P $\neq$ NP, whereas $\epsilon_1 = 0$ would follow if P $=$ NP. Note however that, in general, the GHA phenomenon and the ϵ_1 threshold is independent of the P vs. NP question. In particular, the P vs. NP question only surfaces in the discrete combinatorial settings with $S_1 = P$.

This framework above naturally extends to more than two classes. Let $S_1, \ldots, S_k$ be mutually disjoint sets of approximate problems (i.e., $S_i \cap S_j = \emptyset$ for $i \neq j$), and let the sequence of thresholds $\epsilon_1, \ldots, \epsilon_{2(k-1)} > 0$ satisfy:

$$\epsilon_1 \geq \epsilon_2 > \epsilon_3 \geq \epsilon_4 > \cdots > \epsilon_{2k-3} \geq \epsilon_{2(k-1)}.$$

Then, the problem exhibits an $(S_1, \ldots, S_k)$-*phase transition* at $(\epsilon_1, \ldots, \epsilon_{2(k-1)})$ if:

$$\begin{aligned} &\text{The } \epsilon\text{-approximate problem lies in } S_1, \quad \text{for } \epsilon > \epsilon_1, \\ &\text{The } \epsilon\text{-approximate problem lies in } S_2, \quad \text{for } \epsilon_3 < \epsilon < \epsilon_2, \\ &\quad \vdots \\ &\text{The } \epsilon\text{-approximate problem lies in } S_{k-1}, \quad \text{for } \epsilon_{2k-3} < \epsilon < \epsilon_{2k-4}, \\ &\text{The } \epsilon\text{-approximate problem lies in } S_k, \quad \text{for } \epsilon < \epsilon_{2(k-1)}. \end{aligned} \tag{3.3}$$

A transition at ϵ_{2j-1} is called *sharp* if $\epsilon_{2j-1} = \epsilon_{2j}$ for some j, in which case ϵ_{2j-1} is an *approximation threshold*.

3.1. **GHA in computations – Including randomised algorithms.** The GHA phenomenon can be found throughout the computational sciences and include many core areas as mentioned in the list below:

(i) *Computer science and discrete optimisation (hardness of approximation).* Famous examples in computer science include the MAX-3SAT problem and the MAX-CUT problem [62]. However, there is a plethora of problems in discrete approximation where the GHA phenomenon occurs.

(ii) *Computing optimal neural networks for inverse problems*. As we discuss in this paper (see §5), computing optimal neural networks for inverse problems is a situation where GHA occurs and links to the phenomenon of instability.

(iii) *Computing neural networks in deep learning*. GHA occurs in general when computing neural networks in deep learning. This includes both image classifications and inverse problems [18, 41].

(iv) *Computing minimisers of convex optimisation problems (continuous optimisation).* The GHA phenomenon is particularly rich in continuous optimisation including basic problems such as Linear Programming, LASSO, Basis Pursuit, etc. (see §4).

In the following we shall describe the widespread nature of the GHA phenomenon across AI for inverse problems. The very barriers that GHA describes therefore have a wide impact on the difficulty of establishing trustworthy algorithms. We mention in passing that GHA is part of the program on the Solvability Complexity Index (SCI) hierarchy – which is a program that allows for characterising the foundations of computational mathematics. We refer the reader to [22–24, 40, 41, 60] for more details.

Remark 3.2 (GHA and randomised algorithms)**.** We note an important remark that in the above cases (ii)-(iv) the GHA phenomenon is valid also when considering randomised algorithms. This is crucial as randomness is ubiquitous in algorithms used in modern AI techniques. We emphasise that all our results regarding GHA and randomised algorithms hold for deterministic algorithms as well. Thus, from a GHA point of view, it does not matter if one uses, for example, stochastic gradient descent or a deterministic gradient descent algorithm, they will both be subject to GHA.

4. GHA IN OPTIMISATION AND MACHINE LEARNING – SMALE'S 9TH PROBLEM

The first example of GHA, that we discuss in detail, concerns the optimisation problems defined specifically in §2.1. These optimisation problems are at the heart of modern computational mathematics. They have been extremely well studied, but there are still important questions to be answered. For instance, *Smale's 9th problem* contained in the list of mathematical problems for the 21st century [91] focuses on linear programs and reads

> *Is there a polynomial time algorithm over the real numbers which decides the feasibility of the linear system of inequalities $Ax \geq y$, and if so, outputs such an x?*

Here, there is an underlying assumption that the inputs (y, A) to any such algorithm are represented exactly. In practice, this assumption breaks down due to the underlying nature of modern computers, which employ floating point arithmetic to represent numbers. Problems involving irrational numbers cannot be represented exactly on a computer (this is an immediate consequence of having uncountably many irrational numbers). Furthermore, base-2 floating point arithmetic

does not even allow fractions like $1/3$ to be stored exactly. This fact has not escaped mathematicians interested in computation. For example, in the list of mathematical problems for the 21st century, S. Smale writes

> *"But real number computations and algorithms which work only in exact arithmetic can offer only limited understanding. Models which process approximate inputs and which permit round-off computations are called for."*
>
> — S. Smale (from the list of mathematical problems for the 21st century [91])

Merging both Smale's 9th problem and limitations on representing data exactly yield an extension of Smale's 9th problem; that is, on the possibility of designing strongly polynomial time algorithms that work on inexact data and yield a (potentially) inexact answer. As we shall soon see, it is a key example of GHA in practice.

4.1. **Inexact input – a realistic model of computation.** We can formalise inexactness as follows. We fix the parameters c, δ, λ and τ in the problems from §2.1. Instead of being able to access elements of (y, A) we assume that an algorithm accesses approximations. For example, if we try to solve an LP problem which consists of a matrix $A \in \mathbb{R}^{m\times N}$ and a vector $y \in \mathbb{R}^m$, our algorithm cannot see A or y but instead has access to an oracle $\mathscr{O}_{\text{vec}} : \mathbb{N} \to \mathbb{R}^m$ and an oracle $\mathscr{O}_{\text{mat}} : \mathbb{N} \to \mathbb{R}^{m\times N}$ such that

$$\|\mathscr{O}_{\text{mat}}(k) - A\|_\infty, \|\mathscr{O}_{\text{vec}}(k) - y\|_\infty \leq 2^{-k}, \qquad \forall\, k \in \mathbb{N}, \tag{4.1}$$

where the infinity norm of a matrix should be interpreted as its maximum (in absolute value) entry. More generally, we can embed some subset $\Omega_{m,N}$ of relevant pairs (y, A) into a space $\mathbb{R}^n$, and then, for each $\iota \in \Omega_{m,N}$, consider any oracle $\mathscr{O}(k)$ satisfying

$$\|\mathscr{O}(k) - \iota\|_\infty \leq 2^{-k} \qquad \forall\, k \in \mathbb{N}. \tag{4.2}$$

In the Turing model [93, 97] the machine queries $\mathscr{O}$ via an oracle tape, while in the BSS model [28,29] it uses an oracle node. This "inexact input" extension of the computational model appears widely in the mathematical literature; see e.g. [27, 43, 49, 50, 71, 74]. We can even analyse complexity in this setting: we assume that the cost of accessing $\mathscr{O}(k)$ is bounded by $\text{poly}(k)$ times the dimensions of n. We extend this concept to a set Ω of varying dimensions in the obvious way.

We can now consider algorithms to be maps from the space of possible oracles to a candidate solution $\tilde{x}$ that can be implemented on a Turing machine or BSS machine. For an integer K, we say that an algorithm Γ *produces K correct digits of output accuracy* on input $\iota \in \Omega$, if, for every $\mathscr{O}$ as in (4.2), we have that there exists $\tilde{x} \in \Xi(\iota)$ so that

$$\|\Gamma(\mathscr{O}) - \tilde{x}\| \leq 10^{-K} \tag{4.3}$$

(the norm $\|\cdot\|$ can be taken to be any ℓ^p norm in the remainder of this section). Informally, for each choice of Ξ in §2.1, we write $\|\Gamma(\iota) - \Xi(\iota)\| \leq 10^{-K}$ if for every choice of $\mathscr{O}$ representing ι, there is at least one $\tilde{x}$ so that (4.3) holds. This models the idea that an algorithm is able to take these oracles for inexact input and produce some approximate minimiser for each of the optimisation problems listed above.

One crucial consequence of this definition is that we only consider the algorithm to have produced K correct digits if it does so *for every oracle* $\mathscr{O}$ that satisfies (4.2); in this sense, the algorithm must be *oracle-agnostic*. However, the exact $\tilde{x}$ that the algorithm should approximate may depend on the choice of $\mathscr{O}$. Of course, if there are some choices of oracle for which we

are unable to obtain a suitable approximation then we cannot consider an algorithm implemented with inexact arithmetic (as is the current standard) to be trustworthy.

If $\Omega_{m,N} \subset \mathbb{R}^{m\times(m\times N)}$ is a non-empty set of relevant inputs for optimisation problems (as in §2.1) with dimensions m, N (so $(y, A) \in \Omega_{m,N} \implies y \in \mathbb{R}^m, A \in \mathbb{R}^{m\times N}$), we can generalise the above and consider a set of inputs Ω which encompasses many dimensions, so that Ω is of the form

$$\Omega = \bigcup_{N>m\geq 4}^{\infty} \Omega_{m,N}, \tag{4.4}$$

With the above notation, we obtain the following theorem.

Theorem 4.1 (The extended Smale's 9th problem - computing solutions [17])**.** *For each of the problems listed in* (2.3)–(2.7) *from §2.1 and for suitable choices of parameters c, δ, λ and τ, we have the following. Let $K > 2$. There exists a set $\Omega = \Omega(K)$ as in* (4.4) *such that each of the following hold:*

i) No algorithm can produce K correct digits on each input in Ω as in (4.3)*.*

ii) There does exist an algorithm (a Turing or a BSS machine) that produces $K-1$ correct digits for all inputs in Ω. However, any such algorithm will need an arbitrarily long time to achieve this. In particular, for any fixed dimensions m, N, any $T > 0$, and any algorithm Γ, there exists an input $\iota \in \Omega$ with $\iota \in \mathbb{R}^m \times \mathbb{R}^{m\times N}$ such that either Γ (on input ι) does not produce $K-1$ correct digits for $\Xi(\iota)$ or the runtime of Γ on ι exceeds T.

iii) There exists a polynomial $\mathrm{pol} : \mathbb{R} \to \mathbb{R}$, as well as a Turing machine and a BSS machine that both produce $K-2$ correct digits for all inputs in Ω, so that the number of arithmetic operations for both machines is bounded by $\mathrm{pol}(n)$, where $n = m+mN$ is the number of variables, and the largest k required from the oracle (4.2) *is bounded by $\mathrm{pol}(\log(n))$. Moreover, the space complexity of the Turing machine is bounded by $\mathrm{pol}(n)$.*

This is a very natural example of GHA in practical machine learning. The proof of the theorem constructs a class of inputs for which computing K digits is impossible, asking for less accuracy by computing $K-1$ digits can take arbitrarily long, whereas asking for even less accuracy by trying to compute $K-2$ digits is achievable in a natural analogue of strongly polynomial time.

Remark 4.2. Of course, as stated in the theorem, the collection of inputs $\Omega(K)$ used to demonstrate Theorem 4.1 must depend on K.

Remark 4.3 (Condition does not affect Theorem 4.1)**.** The statements (i) - (iii) above are true even when we require the input in each Ω to be well-conditioned and bounded from above. In particular, for any input $\iota = (y, A) \in \Omega(K)$ we have $\|y\|_\infty \leq 2$, and $\|A\|_{\max} = 1$ and the standard condition number of AA^*, given by $\|(AA^*)^{-1}\|_2\|AA^*\|_2$, is at most 4. In addition, condition numbers which are perhaps better suited to optimisation like *the feasibility condition number* which is large if a problem is close to infeasible, and the *condition of a map* which measures the relative change in the map Ξ (with the choice of Ξ taken from §2.1), can be bounded by 4 and 200 respectively. For more details see [17].

Remark 4.4 (Choice of parameters)**.** It suffices to choose any parameters in Theorem 4.1 so that c is a vector with every coordinate set to 1 and so that $\delta \in [0,1]$, $\lambda \in (0,1/3]$, and $\tau \in [1/2,2]$,

In light of the discussion in §2.2, the reader may wonder what happens if we allow randomness. Could it be the case that using a randomised algorithm allows us to solve these problems and avoid the GHA phenomenon? An extension of the above theorem shows that this is impossible.

Theorem 4.5 (The extended Smale's 9th problem - randomised algorithms [17])**.** *With the same choice of $\Omega = \Omega(K)$ as above (in Theorem 4.1), we have the following.*

(i) For any $\mathrm{p} > \frac{1}{2}$, no randomised algorithm can produce K correct digits with probability greater than or equal to p on each input in Ω.

(ii) If we allow randomised algorithms with a non-zero probability of not halting (not producing an output), then, for any $\mathrm{p} > \frac{2}{3}$, no such algorithm can produce K correct digits with probability greater than or equal to p on each input in Ω. However, there does exist such an algorithm that can produce K correct digits on each input in Ω with probability $2/3$.

(iii) For any randomised algorithm Γ^{ran} and $\mathrm{p} < 1/2$, as well as any $T > 0$ and fixed dimensions m, N there exists an input $\iota \in \Omega$, with $\iota \in \mathbb{R}^m \times \mathbb{R}^N$ such that

$$\mathbb{P}\big(\Gamma^{\mathrm{ran}}(\iota) \text{ does not produce } K-1 \text{ correct digits for } \Xi(\iota) \text{ or the runtime of } \Gamma \text{ on } \iota \text{ exceeds } T\big) > \mathrm{p}.$$

In other words, the GHA phase-transition phenomenon observed in Theorem 4.1 is preserved even when we use randomness, provided we expect our algorithms to avoid failing with probability greater than $1/3$ (which can be strengthened to $1/2$ if we also insist that our algorithm always halts). The precise analysis and proofs of the above results are established in [17]. This paper considers scenarios that occur when fusing Smale's 9th problem and considerations on inexact input. Theorem 4.1 and Theorem 4.5 show just how delicate this question is if we work in an inexact model as is done in practice.

4.2. **Consequences of GHA in ML – Proving existence of neural networks is not enough.** Theoretical results in the mathematics of deep learning often assert that *there exists* a neural network with specified properties – frequently as a (possibly approximate) minimiser of an optimisation problem. In view of the above results, this raises a fundamental question:

Q4. When a theorem guarantees the existence of such a neural network (e.g., as a minimiser), how do we know that it can be computed to a specific accuracy?

The preceding question cannot be addressed without the GHA framework. As shown above, even classical convex optimisation problems – including linear programming, LASSO, and Basis Pursuit – exhibit GHA. The same phenomenon arises when computing optimal neural networks for inverse problems. In § 5.1 we prove the existence of an optimal network but show that it can

be computed only up to a fixed accuracy, even with randomised algorithms. Consequently, Q4 reduces to the central question in GHA:

Q5: Given a computational problem, what is the approximation threshold $\epsilon_1 \geq 0$ – or the approximation thresholds $\epsilon_1, \ldots, \epsilon_n \geq 0$ (see (3.3))?

In particular, beyond a mathematical theory of existence of neural networks and their approximation properties, we need a framework that characterises the approximation thresholds of the computational problems involved in computing the desired networks (see also [1, 57]). This becomes a classification theory in GHA.

5. GHA IN DEEP LEARNING FOR INVERSE PROBLEMS

We recall the inverse problem from (2.1), and the reconstruction map Ψ from (2.2). In particular, let $A : \mathbb{R}^N \to \mathbb{R}^m$ be a linear mapping with non-trivial kernel and $\mathcal{M}_1 \subset \mathbb{R}^N$ be the model class. Given measurements $y = Ax + e$, with $x \in \mathcal{M}_1$, we want to reconstruct an approximation to $x \in \mathcal{M}_1$, where $e \in \mathbb{R}^m$ is a noise vector. That is, we want to find $\Psi : \mathbb{C}^m \to \mathbb{C}^N$ such that

$$\Psi(Ax) \approx x, \quad x \in \mathcal{M}_1. \tag{5.1}$$

Problems of this form have been studied extensively in sparse recovery and compressed sensing, particularly when $\mathcal{M}_1$ denotes a class of sparse or structured-sparse vectors [2–4, 13, 16, 26, 31, 35, 44, 45, 68, 69]. In this setting the reconstruction map is defined by the basis pursuit denoising problem

$$\Psi(y) \coloneqq \arg\min_x \|x\|_1 \quad \text{subject to} \quad \|Ax - y\|_2 \leq \delta, \tag{5.2}$$

where the tolerance $\delta \geq 0$ is chosen to reflect the noise level. Moreover, under suitable assumptions, Ψ in (5.2) can be shown to be optimal [39]. We have discussed the computational properties of this map in §4. More recently, a growing literature has introduced AI-driven methods for tackling similar problems [10, 54, 58, 67, 77, 84]. However, in this case, $\mathcal{M}_1$ may not be the collection of sparse vectors with certain sparsity. Thus, we are presented with a new fundamental problem on the optimality of the reconstruction map. Clearly, we want to construct the best possible Ψ, which leads to the following question.

Q6: Given $A \in \mathbb{R}^{m \times N}$ and the model class $\mathcal{M}_1 \subset \mathbb{R}^N$ in (5.1), what is the optimal choice of Ψ in (5.1), and can an optimal choice of Ψ be computed if we can prove the existence of an optimal Ψ?

5.1. **Optimality of neural networks for underdetermined linear systems.**

In order to address question Q6 we first need to define what we mean by optimality. Fortunately, such optimal maps have already been a focal point in approximation theory, and we follow the framework in [39] when defining optimality.

Definition 5.1 (Optimality of reconstruction maps [39])**.** Let $A : \mathbb{R}^N \to \mathbb{R}^m$ be linear, $\mathcal{M}_1 \subset \mathbb{R}^N$ and $\mathcal{M}_2 := A(\mathcal{M}_1)$. Define the optimality constant for the pair $(A, \mathcal{M}_1)$ as

$$c_{\text{opt}}(A, \mathcal{M}_1) = \inf_{\varphi : \mathcal{M}_2 \rightrightarrows \mathbb{R}^N} \sup_{x \in \mathcal{M}_1} d_1^H(\varphi(Ax), x),$$

where d_1^H denotes the Hausdorff metric (in particular, for a non-empty set $M \subset \mathbb{R}^N$ we have $d_1^H(M,x) = \sup_{\tilde{x}\in M} \|\tilde{x}-x\|_2$ and if φ is single-valued then $d_1^H(\varphi(Ax),x) = d_1(\varphi(Ax),x)$). Here, the double arrow notation $\rightrightarrows$ denotes that the mapping can be multivalued. We define a family of approximately optimal maps of $(A,\mathcal{M}_1)$ as follows. We say that $\varphi_\epsilon : \mathcal{M}_2 \rightrightarrows \mathbb{R}^N$ is a family of approximate optimal maps of $(A,\mathcal{M}_1)$ if for all $\epsilon \in (0,1]$,

$$\sup_{x\in\mathcal{M}_1} d_1^H(\varphi_\epsilon(Ax),x) \leq c_{\text{opt}}(A,\mathcal{M}_1) + \epsilon, \tag{5.3}$$

that φ_ϵ is ϵ-optimal, and that φ_0 is an optimal map for $(A,\mathcal{M}_1)$ if φ_0 satisfies (5.3) with $\epsilon = 0$.

To understand the success of deep learning approaches to inverse problems it thus becomes crucial to ascertain whether or not there exist neural networks that are optimal maps – or ϵ-optimal – for different underdetermined inverse problems. However, the existence of an optimal neural network is not enough. We also need to find a procedure that will allow us to find something close to an optimal neural network, if one exists.

More precisely, given A as above and $\ell \in \mathbb{N}$, we define the class $T_\ell(A)$ of all training sets with ℓ elements is defined as

$$\begin{aligned} T_\ell(A) :=& \{\mathcal{T} \in (\mathbb{R}^N \times \mathbb{R}^m)^\ell \,|\, \mathcal{T} = ((x^k,y^k))_{k=1}^\ell \text{ with } \|x^k\|_2, \|y^k\|_2 \leq 1 \\ & \text{and } \ y^k = Ax^k, (x^k,y^k) \neq (x^j,y^k) \text{ for every } j,k \in \{1,2,\ldots,\ell\} \text{ with } j \neq k\}. \end{aligned} \tag{5.4}$$

In short, to evaluate whether or not a deep learning based approach can address the inverse problem from (2.1), we can further analyse Q6 by studying the following questions:

Let $A \in \mathbb{R}^{m\times N}$ be given, let $\Omega \subset T_\ell(A)$ be a collection of training sets, and assume that there is some prescribed way of associating to each training sample $\mathcal{T} \in \Omega$ an initial domain $\mathcal{M}_1(\mathcal{T}) \subset \mathbb{R}^N$.

Q6i: Does there exist, for each $\mathcal{T} \in \Omega$ and each $\epsilon \geq 0$, a neural network that is ϵ-optimal for the inverse problem $(A,\mathcal{M}_1(\mathcal{T}))$?

Q6ii: Does there exist an algorithm Γ that, for each $\mathcal{T} \in \Omega$, can produce a neural network that is ϵ-close to an optimal neural network for the inverse problem $(A,\mathcal{M}_1(\mathcal{T}))$?

We can formalise Q6ii in the following way: fix an $A : \mathbb{R}^N \to \mathbb{R}^m$, and consider a collection $\Omega \subset T_\ell(A)$ of training sets $\mathcal{T}$. For reasons that will become clear, we consider

$$\mathcal{M}_1(\mathcal{T}) := \{x \,|\, (x,y) \in \mathcal{T}\}. \tag{5.5}$$

That is, we study the problem of computing neural networks that have optimal performance on the training set $\mathcal{T}$ for each $\mathcal{T} \in \Omega$. The motivation for this is to get the strongest possible negative results. In particular, if the answer to question Q6ii is negative for the choice of $\mathcal{M}_1(\mathcal{T})$ as in (5.5), any hope of computing an optimal NN on more general choices of $\mathcal{M}_1(\mathcal{T})$ will be gone (under the weak assumption that each $(x,y) \in \mathcal{T}$ has $x \in \mathcal{M}_1(\mathcal{T})$ – in some sense, that the training set is relevant to the inverse problem).

Next, we can precisely state the goal of Q6ii. Our aim will be to compute a mapping

$$\Xi : \Omega \rightrightarrows \mathcal{NN}_{m,N}, \text{ where } \Xi(\mathcal{T}) := \left\{ \mathbf{N}_{opt}^{\mathcal{M}_1(\mathcal{T})} \,|\, \mathbf{N}_{opt}^{\mathcal{M}_1(\mathcal{T})} \text{ is optimal for } (A,\mathcal{M}_1(\mathcal{T})) \right\}, \tag{5.6}$$

where we recall that $\mathcal{NN}_{m,N}$ is the set of neural networks taking $\mathbb{R}^m \to \mathbb{R}^N$. Note the necessary double arrow notation $\rightrightarrows$, indicating that the map can be multivalued, because the optimal neural network may not be unique. Thus, the map Ξ can be multivalued. If $\Xi(\mathcal{T})$ is non-empty for each $\mathcal{T} \in \Omega$ then the answer to Q6i is positive. Our task is to compute $\mathbf{N}_{opt}^{\mathcal{M}_1(\mathcal{T})}$ – or more precisely, one of the optimal neural networks.

As in §4, a consequence of the use of floating point arithmetic is that we must analyse computations in the presence of inexact arithmetic. This is especially crucial given the applications of inverse problems include classical examples like the subsampled MRI or CT problem wherein A contains irrational numbers taken from entries of the discrete Fourier transform. Following §4, we assume that we cannot see elements of a training set $\mathcal{T} = ((x^1, y^1), (x^2, y^2), \dots, (x^\ell, y^\ell))$ with exact arithmetic. Instead, we consider oracles $\mathscr{O}_x^1, \mathscr{O}_x^2, \dots, \mathscr{O}_x^\ell$ each mapping $\mathbb{N}$ to $\mathbb{R}^N$ and $\mathscr{O}_y^1, \mathscr{O}_y^2, \dots, \mathscr{O}_y^\ell$ each mapping $\mathbb{N}$ to $\mathbb{R}^m$ such that

$$\|\mathscr{O}_x^i(k) - x^i\|_\infty, \|\mathscr{O}_y^i(k) - y^i\|_\infty \leq 2^{-k} \qquad (\forall\, i \in \{1, 2, \dots, \ell\},\ \forall\, k \in \mathbb{N}). \tag{5.7}$$

As in (4.2), we can condense (5.7) so that after embedding Ω into $\mathbb{R}^n$, we consider $\mathscr{O}$ such that, for some $\mathcal{T} \in \Omega \subset \mathbb{R}^n$ and all $k \in \mathbb{N}$ we have

$$\|\mathscr{O}(k) - \mathcal{T}\|_\infty \leq 2^{-k} \qquad (\forall\, \mathcal{T} \in \Omega,\ \forall\, k \in \mathbb{N}). \tag{5.8}$$

We say that such an $\mathscr{O}$ *corresponds* to $\mathcal{T}$. We once again require that a successful algorithm should work correctly on any $\mathcal{T} \in \Omega$ and any choice of oracles corresponding to $\mathcal{T}$. Thus we have the following definition:

Definition 5.2 (Computing Ξ to ϵ-accuracy)**.** We say that the mapping Ξ in (5.6) can be computed to ϵ-accuracy if there exists an algorithm Γ such that for any $\mathcal{T} \in \Omega$

$$\inf_{\mathbf{N}_{opt}^{\mathcal{M}_1(\mathcal{T})} \in \Xi(\mathcal{T})} \sup_{y \in \bigcup_{\mathcal{T} \in \Omega} A(\mathcal{M}_1(\mathcal{T}))} \|\Gamma(\mathscr{O}, \epsilon)(y) - \mathbf{N}_{opt}^{\mathcal{M}_1(\mathcal{T})}(y)\|_2 \leq \epsilon,\ \forall\, \mathscr{O} \text{ corresponding to } \mathcal{T}. \tag{5.9}$$

Similarly, such a Γ is said to compute Ξ to ϵ-accuracy.

The following theorem applies to any $A \in \mathbb{R}^{m \times N}$ with $N > m$, provided that the spectrum of AA^* is bounded from below.

Theorem 5.3 (GHA – Phase transitions for computing optimal NNs [53])**.** *For any rational $\epsilon_1 \in (0, 3/16]$ there exists a domain $\Omega \subset T_\ell(A)$ of training sets and a set of corresponding initial domains $\{\mathcal{M}_1(\mathcal{T}) : \mathcal{T} \in \Omega\}$ such that $\Xi(\mathcal{T}) \neq \emptyset$ for each $\mathcal{T} \in \Omega$ and each of the following occur simultaneously:*

- *(i) No (randomised) algorithm can approximate any optimal neural network $\mathbf{N}_{opt}^{\mathcal{M}_1(\mathcal{T})} \in \Xi(\mathcal{T})$ for all choices of oracle and $\mathcal{T} \in \Omega$ to accuracy ϵ_1 (with probability greater than $p > 1/2$ in the randomised case – this is even the case if the algorithm has a non-zero probability of not halting).*
- *(ii) There is an algorithm Γ such that Γ computes Ξ to ϵ accuracy for all dyadic $\epsilon > 2\epsilon_1$ (in the sense of* (5.9)*).*

Remark 5.4 (No $p = 2/3$ randomised algorithm with non-zero probability of not halting)**.** In the setting of §4, the phase transitions discussed in Theorem 4.5 yield randomised algorithms with a

non-zero probability of not halting which succeed with probability $p = 2/3$. This is not the case for the generalised hardness of approximation phenomenon described in Theorem 5.3; the largest guaranteed probability of success here is $1/2$. This distinction can be interpreted as a 'harder' phase transition than in Theorem 4.5.

Theorem 5.3 establishes the existence of algorithms if the required accuracy is worse than $2\epsilon_1$, and the non-existence of algorithms if we ask for better accuracy than ϵ_1. It is therefore natural to ask about the behaviour of algorithms in-between, that is, if we ask for an accuracy between ϵ_1 and $2\epsilon_1$. This is the situation discussed in Theorem 5.5. Perhaps surprisingly, in this scenario we observe a phase transition in the available stability of the computed neural networks.

To make this formal recall that for a function $\mathbf{N} : \mathbb{R}^m \to \mathbb{R}^N$ we define the global Lipschitz constant $\mathcal{L}$ by

$$\mathcal{L}(N) = \sup_{y,z\in\mathbb{R}^m,\, y\neq z} \frac{\|\mathbf{N}(z) - \mathbf{N}(y)\|_p}{\|z - y\|_p}$$

We can analyse stability by examining the Lipschitz constant. More precisely, for $D > 0$,we can define a restriction $\Xi_D : \Omega \rightrightarrows \mathcal{NN}_{m,N}$ of Ξ so that Ξ_D only yields neural networks which are D-Lipschitz. Succinctly,

$$\Xi_D(\mathcal{T}) \coloneqq \left\{ \mathbf{N}_{opt}^{\mathcal{M}_1(\mathcal{T})} \in \Xi(\mathcal{T}) \mid \mathcal{L}(\mathbf{N}_{opt}^{\mathcal{M}_1(\mathcal{T})}) \leq D \right\}. \tag{5.10}$$

Note that $\Xi_D(\mathcal{T})$ could potentially be empty for small values of D.

If we want a stable network, we thus insist that Ξ_D is both non-empty and can be computed. Theorem 5.5 analyses this situation:

Theorem 5.5 (Instability of NNs in inverse problems as a result of GHA [53])**.** *Consider Theorem 5.3. For the same $\epsilon_1 \in (0, 3/16]$, domain $\Omega \subset T_\ell(A)$ of training sets and a set of corresponding initial domains $\{\mathcal{M}_1(\mathcal{T}) : \mathcal{T} \in \Omega\}$ as in Theorem 5.3, there exists a T_1 in Ω and a $D > 0$ so that*

(i) $\Xi_D(\mathcal{T}_1) \neq \emptyset$ that is, there exists an optimal neural network $N_{opt}^{\mathcal{M}_1(\mathcal{T}_1)}$ for $\mathcal{T}_1$ such that the Lipschitz constant $\mathcal{L}(N_{opt}^{\mathcal{M}_1(\mathcal{T}_1)}) \leq D$. Yet, there is an oracle $\mathscr{O}$ corresponding to $\mathcal{T}_1$, such that for any $K > 0$, $\delta \in (0, \epsilon_1)$ and any algorithm $\hat{\Gamma}$ which computes Ξ to ϵ accuracy (with $\epsilon \in (\epsilon_1, 2\epsilon_1 - \delta]$) we have the following: for every $\eta > 0$ there exists a $y \neq 0$ with $\|y\|_2 \leq \eta$ and

$$\frac{\|\mathbf{N}_{\mathscr{O},\epsilon}(0) - \mathbf{N}_{\mathscr{O},\epsilon}(y)\|_2}{\|y\|_2} > K. \tag{5.11}$$

where $\mathbf{N}_{\mathscr{O},\epsilon}$ is the neural network produced by $\hat{\Gamma}$ on input $\mathscr{O}$ and ϵ.

(ii) By contrast, if every element of the spectrum of AA^* is at least $\beta > 0$ then the algorithm Γ from Theorem 5.3 part (ii) can be constructed so that, on any input $\mathscr{O}$ and $\epsilon > 2\epsilon_1$, Γ always produce a neural network $\mathbf{N}_{\mathscr{O},\epsilon}$ with global lipschitz constant satisfying $\mathcal{L}(\mathbf{N}_{\mathscr{O},\epsilon}) \leq 2/\beta$.

In short, (5.11) demonstrates the instability issue that follows from demanding that $\hat{\Gamma}$ produces neural networks with accuracy between ϵ_1 and $2\epsilon_1$. In fact, this instability is arbitrarily bad and

is local around 0. It should be noted that the construction of $\mathcal{T}_1$ ensures that $(x, 0) \in \mathcal{T}_1$ for some $x \in \mathbb{R}^N$, so it is not unnatural to analyse the stability of $\mathbf{N}_{\mathscr{O},\epsilon}$ around 0.

The contrasting result shown in Theorem 5.5 part (ii) is a strong bound on the stability of $N^{\mathscr{O}}$; unlike the local bound in part (i), the bound in part (ii) is a bound on the *global* lipschitz constant.

5.2. **Consequences of GHA in inverse problems – Accuracy-stability trade-off.** The accuracy–stability trade-off in AI methods for inverse problems is widely documented empirically [6, 41, 54, 56] and, to a lesser extent, theoretically [41, 56]. Theorems 5.3 and 5.5 sharpen this picture. In brief, they show that for any $A \in \mathbb{R}^{m\times N}$ with $N > m$, any $\epsilon_1 \in (0, 3/16]$, and assuming the spectrum of AA^* is bounded below (i.e., bounded away from 0), there exist training sets for which:

(I) An optimal neural network for the inverse problem exists, yet it cannot be computed to accuracy better than ϵ_1.

(II) One can compute an approximation to the optimal neural network to accuracy $> 2\epsilon_1$ – such that the approximate neural network is also stable.

(III) Any attempt to compute an optimal neural network to accuracy $< 2\epsilon_1$ produces a neural network that is arbitrarily unstable, even though a stable optimal network does exist.

Together, these results provide a rigorous explanation of the observed accuracy-stability tension. Paradoxically, attempts to compute a neural network with sufficiently high accuracy yield arbitrarily unstable reconstruction methods. By contrast, if the computed neural network's accuracy remains below a critical threshold (not too accurate), stability is attainable. Thus there is an accuracy-stability trade-off: exceeding the threshold in accuracy precipitates instability.

Remark 5.6 (GHA and compressed sensing)**.** In the field of compressed sensing the phenomenon of GHA is present, however, in a very subtle way. Indeed, because of the standard assumption in compressed sensing that the measurement matrix A in (2.1) satisfies the robust null space property, the approximation threshold $\epsilon_1 > 0$ is on the level of the δ in the basis pursuit problem (2.4) – see [3,17] for details. Hence, the GHA phenomenon can be controlled when there are appropriate assumptions on the measurement matrix A linking it to $\mathcal{M}_1$. This example from compressed sensing illustrates the challenge of controlling GHA in machine learning, as extra assumptions need to be incorporated in the training process. This is highly non-trivial, as Theorems 5.3 and 5.5 demonstrate that such assumptions must create limitations on the training data.

6. HALLUCINATIONS IN AI FOR INVERSE PROBLEMS – DETAIL TRANSFER

Hallucinations in AI methods for inverse problems [41,54,56] are caused by different phenomena. In this section we cover *detail transfer*, that can cause AI methods to hallucinate, regardless of whether they are stable or not. Hallucinations due to instability are covered in §7.2. We recall from above that to solve the inverse problem (2.1), the aim is to construct a reconstruction map:

$$\Psi : \mathbb{C}^m \to \mathbb{C}^N, \quad \Psi(Ax) \approx x, \quad x \in \mathcal{M}_1, \tag{6.1}$$

where we assumes that the desired images x belong to a model class $\mathcal{M}_1 \subset \mathbb{C}^N$. However, the following theorem below makes no assumption on how Ψ was constructed, nor about the model class.

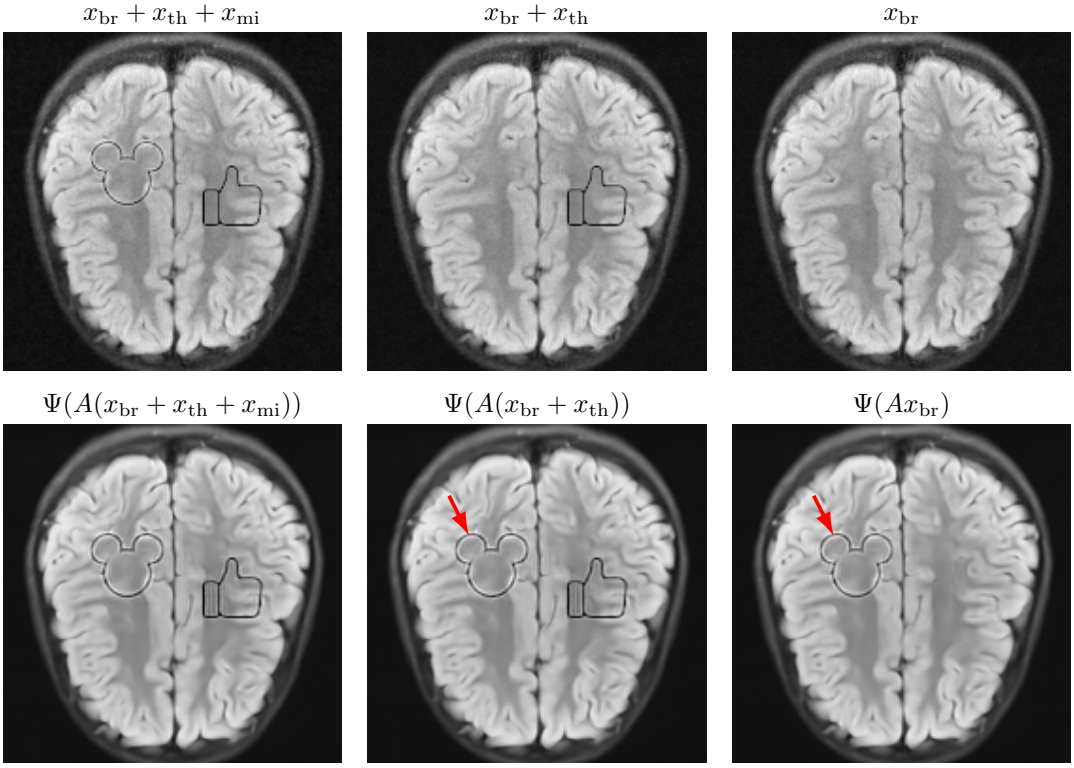


FIGURE 2. (**Hallucinations due to detail transfer**). A trained NN $\Psi\colon \mathbb{C}^m \to \mathbb{C}^N$ accurately recovers the image in the first column. But it hallucinates by incorrectly transferring the 'Mickey Mouse' detail x_{mi} in the first column when recovering the images in the second and third columns. The measurement matrix is a subsampled Fourier transform with $m/N = 20\%$, which models an MRI acquisition with 5-fold acceleration. See [56] for further details. Note that the NN does not transfer the 'Thumb' detail x_{th}. Theorem 6.1 explains why this is the case.

Theorem 6.1 (Hallucinations from detail transfer in inverse problems [56])**.** *Let $A \in \mathbb{C}^{m\times N}$, $\delta, c > 0$ and $x_{\mathrm{Det}} \in \mathbb{C}^N$ with $\|Ax_{\mathrm{Det}}\|_p \leq \delta$.*

(i) (Ψ hallucinates by transferring details). *Let $\Psi\colon \mathbb{C}^m \to \mathbb{C}^N$ have $\mathcal{L}(\Phi) \leq c$ and suppose that*

$$\|\Psi\big(A(x + x_{\mathrm{Det}})\big) - (x + x_{\mathrm{Det}})\|_p \leq \delta. \tag{6.2}$$

for some $x \in \mathbb{C}^N$. Then for every $e \in \mathcal{B}_\delta^p(0)$, there is a $z \in \mathbb{C}^N$ with $\|z\|_p \leq (1+2c)\delta$ such that

$$\Psi(Ax + e) = x + x_{\mathrm{Det}} + z. \tag{6.3}$$

(ii) (There always exists a NN that hallucinates). *For any dimensions $(\mathbf{N}, L)$ with $\mathbf{N}_0 = m$ and $\mathbf{N}_L = N$, there is a $\widetilde{\Psi} \in \mathcal{NN}(\mathbf{N}, L)$ satisfying* (6.2) *and $\mathcal{L}(\widetilde{\Psi}) \leq c$.*

Suppose that $\delta > 0$ is small. Theorem 6.1 establishes that a detail x_{Det} embedded in the image $x' := x + x_{\mathrm{Det}}$ will be transferred onto the *detail-free* image x when Ψ is given either noisy or

noiseless measurements of the form $Ax + e$, where $e \in \mathcal{B}_0^p(\delta)$. Crucially, Theorem 6.1 becomes relevant when $\|x_{\mathrm{Det}}\|_p \gg \delta \geq \|Ax_{\mathrm{Det}}\|_p$, indicating that the detail is significant (i.e. large in norm) but yields small measurements. This situation can easily occur when A has a non-trivial kernel $\mathcal{N}(A)$ (i.e., $\mathrm{rank}(A) < N$), which is typical in many applications (for example, when $m < N$). It may also arise if $\mathcal{N}(A) = \{0\}$ (i.e., $\mathrm{rank}(A) = N$) but A is ill-conditioned. Note that Theorem 6.1 does not require the map Ψ to be unstable. Hallucinations arising as a result of instability are discussed later in this article. Figure 2 illustrates a representative example of this result. The 'Mickey Mouse' detail $x_{\mathrm{mi}} \in \mathcal{N}(A)$, while the 'Thumb' detail x_{th} has comparatively large measurements, namely $\|Ax_{\mathrm{th}}\|_p \gg 0$. The NN is trained to reconstruct the image $x_{\mathrm{br}} + x_{\mathrm{th}} + x_{\mathrm{mi}}$. As a consequence, it mistakenly transfers the detail x_{mi}, whereas the detail x_{th} is handled correctly (i.e., it is not transferred).

6.1. **Coping with hallucinations coming from detail transfer – No free lunch.** Theorem 6.1 shows that there is an *accuracy-hallucination trade-off*. If the map Ψ performs too well on a certain image x_1 with detail lying close to $\mathcal{N}(A)$, then it will hallucinate, by incorrectly transferring this detail to another image x_2. Thus, the only way to protect against detail transfer in theorem 6.1 is to link the null space $\mathcal{N}(A)$, the (implicit) model class $\mathcal{M}_1$ and the reconstruction map Ψ. In the previous example, for instance, this means that Ψ must not be allowed to recover both $x_{\mathrm{br}} + x_{\mathrm{th}} + x_{\mathrm{mi}}$ and $x_{\mathrm{br}} + x_{\mathrm{th}}$ accurately. But this can only be achieved by imposing conditions on A, $\mathcal{M}_1$ and Ψ.

In terms of what conditions may be required, recall classical *model-based* methods such as sparse regularization. Here, $\mathcal{M}_1$ is the set of approximately sparse images in some transform (e.g., wavelets), and the theory of compressed sensing links $\mathcal{N}(A)$, $\mathcal{M}_1$ and the reconstruction map Ψ via assumptions such as the *robust Null Space Property* [3]. These assumptions mean that such methods typically avoid the conclusions of Theorem 6.1. A key aim of AI-based methods is to outperform model-based methods by dispensing with such careful, *handcrafted* design. However, if such conditions are absent, then, as Theorem 6.1 shows, hallucinations are inevitable. This observation presents a revision of the prospects for AI in inverse problems. When first introduced, the initial hope [102] was that AI-based methods could circumvent careful mathematical modelling, and simply learn a good reconstruction map Ψ and (implicit) model class $\mathcal{M}_1$ from training data, regardless (and even without knowledge) of A [102]. Theorem 6.1 shows this is generally impossible – there is no free lunch.

7. The instability phenomenon in modern AI

The instability phenomenon in modern AI methods based on neural networks seems universal, and it is crucial to understand this global phenomenon. Hence, to provide a more complete picture, we consider the issue of *instability in inverse problems* in addition to *instability in classification*. The latter phenomenon was discovered several years before the former. Interestingly, while these phenomena are very similar, the mathematical explanations are rather different.

7.1. **Instability in classification problems & adversarial attacks.** Instability in classification problems – which renders them susceptible to *adversarial attacks* – is arguably one of the most well-known mechanism for hallucinations and errors in modern AI. It is of particular importance

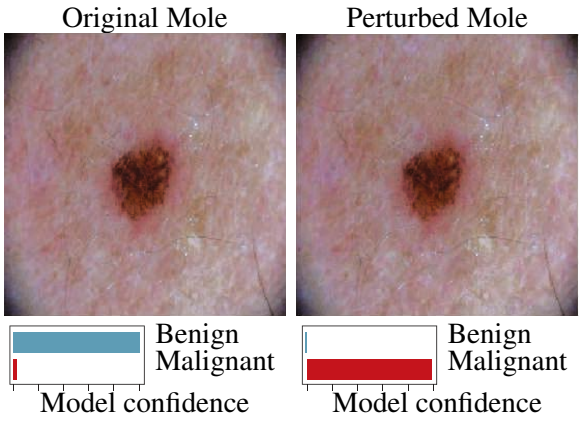


FIGURE 3. **AI-generated misdiagnosis and unstable confidence estimation**. Left: Dermatoscopic image of a benign melanocytic nevus along with the diagnostic probability computed by a deep NN. Right: Combined image with a slight perturbation and the diagnostic probabilities from the same deep NN (Source: *Science* [52]). The NN provides an incorrect diagnosis due to a small perturbation.

in computer vision, wherein small changes can impact the ability of a classifier to identify an object or its properties (see Figure 3). While this phenomenon was initially observed through empirically [95], there have since been a number of theoretical approaches to understanding adversarial attacks. Broadly speaking, instability can be shown to result from: *random networks* having a high probability of being unstable, as shown through *concentration of measure* arguments; *complexity or data issues* (e.g., insufficient data or computational time to recover a suitable network); and from *training picking up unstable features*. We discuss each of these in turn.

7.1.1. *Random networks.* [12] shows that if the weights of a NN are i.i.d. normal random variables and the smallest layer is not significantly smaller than the widest layer, then the network will be vulnerable to adversarial attacks. This result is intriguing because it suggests that, in some sense, networks susceptible to adversarial attacks are 'typical' in the space of networks. However, it does not explain why a particular network trained using a process like stochastic gradient descent would be prone to such attacks.

7.1.2. *Concentration of measure arguments.* If the number of neurons is large or the dimensionality is high, concentration of measure arguments show that adversarial attacks become inevitable. This is the case in [46] and [55] which '*hypothesize[s] that [adversarial attacks are] due to the high-dimensional geometry of data manifolds*'. There is some debate over whether improving a classifier's performance on data before considering stability helps or harms robustness: [55] provides an example where it is necessary to do so, whereas works like [96] argue the opposite, asserting that robustness and accuracy cannot be simultaneously achieved. Moreover, some of these arguments have been criticized for being overly broad, particularly when applied to classifiers assumed to be robust on the same dataset. As noted in [89], '*their bounds also apply to the human visual system. However, an important aspect of adversarial examples is that they often fool current classifiers, yet are still easy to recognize for humans.*'

Concentration of measure arguments have further implications. Consider the problem of building a classifier on either the unit sphere or unit cube, with an associated probability distribution for samples from these spaces. If the probability distribution has a sufficiently small supremum or the dimension grows, [90] shows that adversarial attacks become inevitable or the classifier performs poorly on the original dataset. This uses the isoperimetric inequality and applies to general classifiers, which makes this argument susceptible to the criticism above. Meanwhile, [98] gives general conditions that imply a high dimensional classifier will be susceptible to adversarial attacks without the isoperimetric inequality.

7.1.3. *Complexity/computability or data issues.* In some cases, there provably exists a stable NN, but it is difficult or impossible to compute it. This can happen because not enough samples of the data were retrieved to get the right network (e.g. [89]) or because there is an inherent computational complexity or computability issue in getting a stable network [32] – as discussed in the previous section.

7.1.4. *Training picks up unstable correlating features.* A different scenario occurs when stable features exist, but the AI (during training) learns correlated features that are unstable. This argument is made in [66], while [47] provides an example of this phenomenon by considering two classes: the first contains images with vertical lines, whereas the second consists of horizontal lines with a tiny perturbation added to the background. A classifier attempting to distinguish between these classes instead learns the background, which is unstable. The phenomenon of correlating features is widespread, as highlighted by Y. LeCun:

> "*It's difficult to collect large amounts of labeled data that are not biased in some way. I'm not necessarily talking about societal bias, but about correlations in the data that the system should not be using.*" – Y. LeCun (2022) IEEE Spectrum.

Theorem 7.1 (with a summary in Figure 4) shows a marriage of the previous two categories. In this situation, *a stable classifier will exist, but is difficult to find* and so instead *unstable correlating structures will be acquired* and hence any training procedure will lead to an unstable classifier.

Theorem 7.1 (Instability of trained NNs despite existence of a stable NN [18])**.** *Fix $d \geq 2$. There is an uncountable collection $\mathcal{C}_1$ of classification functions $f : [0,1]^d \to \{0,1\}$ and a constant $C > 0$ such that the following holds. For every $f \in \mathcal{C}_1$, any norm $\|\cdot\|$ and every $\epsilon > 0$, there is an uncountable family $\mathcal{C}_2$ of probability distributions on $[0,1]^d$ so that for any $\mathcal{D} \in \mathcal{C}_2$, any $\mathbf{N}$ and L with $L > 2$ and $\mathbf{N}_L = 1$, and any $\mathrm{p} \in (0,1)$, if we draw training data $\mathcal{T} = \{x^1, \ldots, x^r\}$ and test data $\mathcal{V} = \{y^1, \ldots, y^s\}$, where the x^j and y^j are drawn independently at random from $\mathcal{D}$, the following occur with probability exceeding $1 - \mathrm{p}$ provided that r is sufficiently large:*

- *(i)* **(Great generalisability)** *For every $\mathcal{R}$ with $\mathcal{R}(v,w) = 0$ iff $v = w$ (e.g. Cross-Entropy loss), there exists an NN ϕ that minimises $\mathcal{R}$ over possible $\mathcal{NN}(\mathbf{N}, L)$ and $\phi(x) = f(x)$ for all $x \in \mathcal{T} \cup \mathcal{V}$.*
- *(ii)* **(Any successful NN with these fixed dimensions is universally unstable)** *Yet, for any neural network $\hat{\phi} \in \mathcal{NN}(\mathbf{N}, L)$ (and thus, in particular, for $\hat{\phi} = \phi$) and any monotonic $g : \mathbb{R} \to \mathbb{R}$, there is a subset $\tilde{\mathcal{T}} \subset \mathcal{T} \cup \mathcal{V}$ such that there exist uncountably many universal adversarial perturbations $\eta \in \mathbb{R}^d$ so that for each $x \in \tilde{\mathcal{T}}$ we have*

$$|g \circ \hat{\phi}(x + \eta) - f(x + \eta)| \geq 1/2, \quad \|\eta\| < \epsilon, \quad |\mathrm{Supp}(\eta)| \leq 2. \tag{7.1}$$

  The set $\tilde{\mathcal{T}}$ can be arbitrarily large by taking r arbitrarily large.
- *(iii)* **(Other stable and accurate NNs exist)** *However, there exists a stable and accurate NN $\psi \in \mathcal{NN}_{d,1}$ that satisfies $\psi(x) = f(x)$ for all $x \in \mathcal{B}^{\infty}_{\hat{\epsilon}}(\mathcal{T} \cup \mathcal{V})$, when $\hat{\epsilon} \leq (C\mathrm{p}/(r \vee s))^4$.*

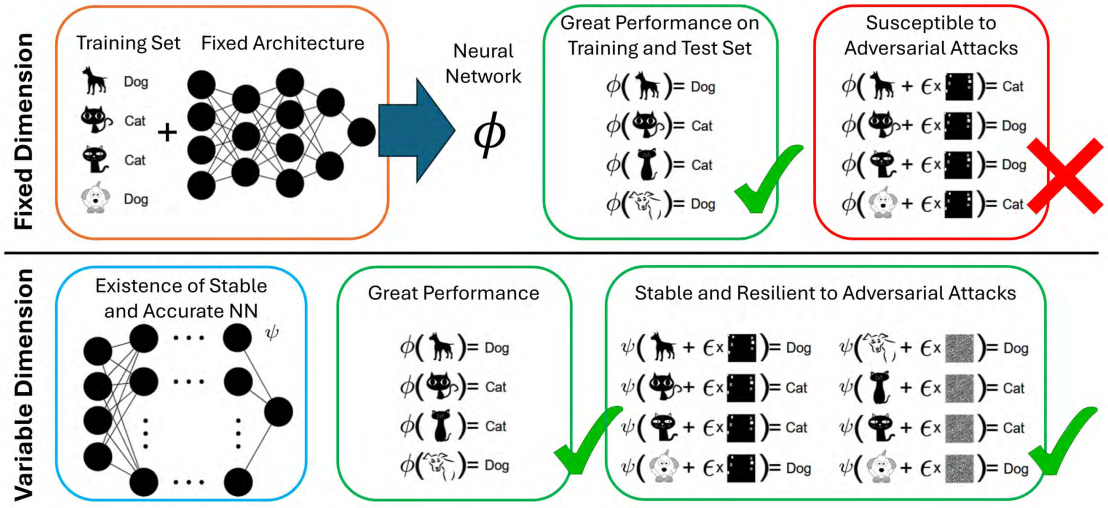


FIGURE 4. **Training with fixed architecture yields instability: variable dimensions for NNs is necessary for stability.** A visual interpretation of theorem 7.1. A fixed dimension training procedure can lead to excellent performance and yet be highly susceptible to adversarial attacks, even if there exists a NN which has both great performance and excellent stability properties. However, such a stable and accurate ReLu network must have variable dimensions depending on the input.

The three parts of Theorem 7.1 demonstrate how adversarial attacks can occur. Specifically, training a NN with a simple cross-entropy cost function can give us the impression that the NN works well, on both the training and a test set. However, the NN we recover is unstable (and the same perturbation can be applied to change the class of a large set, known as *universal instability*). This instability applies in the presence of any choice of function g used to transition between the output of the NN and a confidence value for the classifier. The perturbations involved are small, both in norm and in the sense that they only change two components. A consequence of (7.1) is that even a regularised cost function will lead to either instability or poor generalisability. Despite this, there is a stable NN out there, but a standard training procedure cannot recover it, so that the *stable classifier exists but is difficult to find*.

LeCun's comments above on the difficulty of avoiding unwanted correlations directly apply here. Namely, the proof of Theorem 7.1, as presented in [18], shows that the reason that the network performs well in the part (i) is that it finds an *unstable correlation*. This correlation is in some sense easy to find, but unstable, which results in the poor robustness properties in part (ii). To summarise, these unstable correlations are the cause of both the great generalisation of the NN and the unwanted instability.

7.2. **Instabilities in inverse problems and imaging.** Much like image classification, AI-based approaches to inverse problems – recall (2.1) and (2.2) – can exhibit severe instabilities, as shown in [6]. However, these instabilities do not arise from unstable correlating features. Instead, they may stem from *over- or inconsistent performance* of the reconstruction map – a concept explained below. This phenomenon is highlighted in Theorem 7.2.

Theorem 7.2 (Instabilities in inverse problems [56])**.** *Let $\Psi\colon \mathbb{C}^m \to \mathbb{C}^N$ be continuous and $p \in [1,\infty]$ satisfy*

$$\|\Psi(Ax) - x\|_p < \eta, \quad \|\Psi(Ax') - x''\|_p < \eta, \quad \|A(x-x')\|_p \leq \eta, \tag{7.2}$$

for some $A \in \mathbb{C}^{m\times N}$, $x, x', x'' \in \mathbb{C}^N$ and $\eta > 0$. Then the following hold.

(i) **(Ψ is unstable)** *There is a $\delta_1 > 0$ such that, for all $\epsilon \geq \eta$, if $\tilde{y} \in \overline{\mathcal{B}}^p_{\delta_1}(y)$ and $y = Ax$ then*

$$L^\epsilon(\Psi, \tilde{y}) \geq \frac{1}{\eta}\left(\|x - x''\|_p - 2\eta\right). \tag{7.3}$$

(ii) **(Ψ hallucinates)** *There exist $z \in \mathbb{C}^N$ with $\|z\|_p \geq \|x - x''\|_p$ (e.g., $z = x'' - x$), $e \in \mathbb{C}^m$ with $\|e\|_p \leq \eta$, and $\delta_2 > 0$ so that $\|\Psi(A\tilde{x} + \tilde{e}) - (\tilde{x} + \tilde{z})\|_p \leq \eta, \quad \forall\, \tilde{x} \in \overline{\mathcal{B}}^p_{\delta_2}(x)$, $\tilde{e} \in \overline{\mathcal{B}}^p_{\delta_2}(e)$, $\tilde{z} \in \overline{\mathcal{B}}^p_{\delta_2}(z)$.*

Following the discussion in [56], to explain the concepts of over- and inconsistent performance we consider the following examples.

Example 7.3. If $x'' = x'$ and $\|x - x'\|_p > 2\eta$, then (7.2) asserts that Ψ *overperforms*: it recovers two images x and x' well, whose measurements are Ax and Ax' are very close. The result is that Ψ is unstable and it hallucinates, with both effects becoming worse as η decreases. Moreover, because both effects occur in balls, they are *not rare events*. On the other hand, suppose that x' and x are close, but x'' is far from x, i.e., $\|x - x''\|_p > 2\eta$. Then (7.2) asserts that Ψ performs *inconsistently*: it recovers x well, x' poorly, despite x and x' having similar measurements. Then Ψ necessarily that Ψ is unstable and hallucinates. In Figure 5 we show several examples of these effects, where hallucinations and errors are caused by small random perturbations.

7.3. **Coping with instabilities in AI methods – The accuracy-stability trade-off.** Theorem 7.1 demonstrates a clear challenge in our existing approach to machine learning. Namely, it will be impossible to overcome instability in classification without changing the training procedure to make use of a varying structure for the trained NN. Training with a fixed number of nodes will (for some classification problems) create a network that either performs poorly or picks up unstable correlations. Hence, current techniques are subject to an accuracy-stability trade-off. However, as Theorem 7.1 shows, there is scope for a new class of algorithm design that may resolve the instability issues. Yet, as we have seen, computing stable NNs (even though their existence may be guaranteed) could be impossible and an understanding of the links to the foundations of computational mathematics will be necessary.

Theorem 7.2 also implies an accuracy-stability trade-off. If the reconstruction map Ψ overperforms (on as few as two images) then it is necessarily unstable, with the instability becoming arbitrarily large as the performance increases. Thus, an immediate coping mechanism to reduce instability is to reduce overall performance. There are several ways that this can be achieved. For example, one can add extra random noise to the training data in the training process. Another approach is to put restrictions on the Lipchitz constant of the reconstruction map in the training process [11]. See [41, 54, 56] for practical examples demonstrating such mitigations confirming the theoretical predictions of the accuracy-stability trade-off.

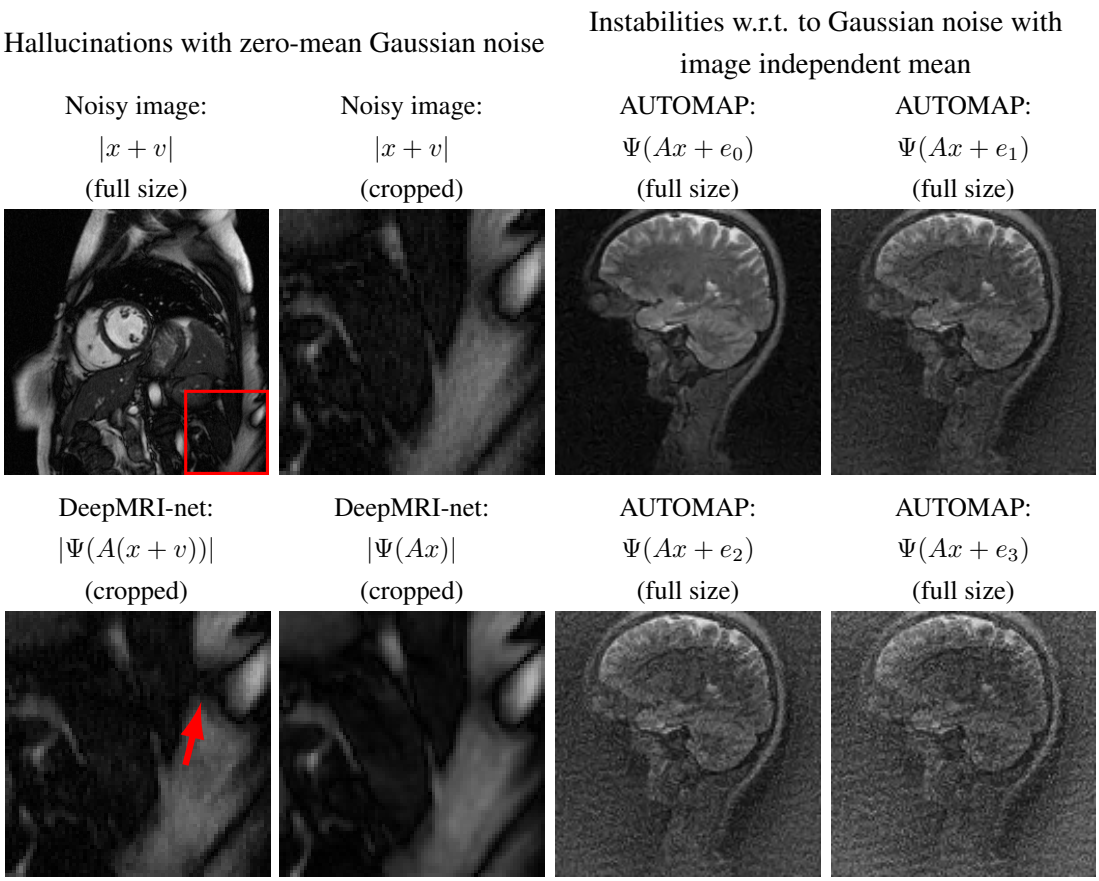


FIGURE 5. (**Hallucinations and instabilities due to random noise**) Two DL methods exhibit hallucinations and instabilities due to random noise. On the left, the DeepMRI-net [88] reconstruction map is unstable to mean-zero Gaussian noise v. In this case, the NN hallucinates by removing a key image feature (see the red arrow). On the right, the AUTOMAP [102] reconstruction map is unstable to Gaussian noise. The noise vector e_0 is drawn from a mean-zero Gaussian distribution, whereas the means of the distributions used to generate e_1, e_2 and e_3 are based on three worst-case noise vectors computed for AUTOMAP with respect to a different image. This makes the noise image independent. The underlying instability of the reconstruction map produces noticeable artefacts in all cases. The measurement matrix in these experiments is a subsampled Fourier transform with 33% (left) and 60% (right) subsampling, respectively. See [56] for further information.

REFERENCES

[1] B. Adcock and N. Dexter. The gap between theory and practice in function approximation with deep neural networks. *SIAM Journal on Mathematics of Data Science*, 3(2):624–655, 2021.

[2] B. Adcock and A. C. Hansen. Generalized sampling and infinite-dimensional compressed sensing. *Foundations of Computational Mathematics*, 16(5):1263–1323, 2016.

[3] B. Adcock and A. C. Hansen. *Compressive Imaging: Structure, Sampling, Learning*. Cambridge University Press, 2021.

[4] B. Adcock, A. C. Hansen, C. Poon, and B. Roman. Breaking the coherence barrier: A new theory for compressed sensing. *Forum of Mathematics, Sigma*, 5:1–84, 001 2017.

[5] N. Akhtar and A. Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6:14410–14430, 2018.

[6] V. Antun, F. Renna, C. Poon, B. Adcock, and A. C. Hansen. On instabilities of deep learning in image reconstruction and the potential costs of AI. *Proceedings of the National Academy of Sciences*, 117(48):30088–30095, 2020.

[7] S. Arora and B. Barak. *Computational Complexity - A Modern Approach*. Princeton University Press, 2009.

[8] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, May 1998.

[9] S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998.

[10] S. Arridge, P. Maass, O. Öktem, and C.-B. Schönlieb. Solving inverse problems using data-driven models. *Acta Numerica*, 28:1–174, 2019.

[11] S. Aziznejad, H. Gupta, J. Campos, and M. Unser. Deep neural networks with trainable activations and controlled lipschitz constant. *IEEE Transactions on Signal Processing*, 68:4688–4699, 2020.

[12] P. Bartlett, S. Bubeck, and Y. Cherapanamjeri. Adversarial examples in multi-layer random reLU networks. In *Advances in Neural Information Processing Systems*, 2021.

[13] A. Bastounis, B. Adcock, and A. C. Hansen. From global to local: Getting more from compressed sensing. *SIAM News*, 50(8):1–4, 2017.

[14] A. Bastounis, F. Cucker, and A. C. Hansen. When can you trust feature selection? – i: A condition-based analysis of lasso and generalised hardness of approximation. preprint, 2023.

[15] A. Bastounis, F. Cucker, and A. C. Hansen. When can you trust feature selection? – II: On the effects of random data on condition in statistics and optimisation. *arXiv:2312.11429*, 2023.

[16] A. Bastounis and A. C. Hansen. On the absence of uniform recovery in many real-world applications of compressed sensing and the restricted isometry property and nullspace property in levels. *SIAM Journal on Imaging Sciences*, 10(1):335–371, 2017.

[17] A. Bastounis, A. C. Hansen, and V. Vlačić. The extended Smale's 9th problem - on computational barriers and paradoxes in estimation, regularisation, learning and computer-assisted proofs. *Preprint*, 2022.

[18] A. Bastounis, A. C. Hansen, and V. Vlačić. The mathematics of adversarial attacks in AI – Why deep learning is unstable despite the existence of stable neural networks. *Eur. Jour. Appl. Math.*, 2024 (to appear).

[19] L. Beerens and D. J. Higham. Adversarial ink: componentwise backward error attacks on deep learning. *IMA Journal of Applied Mathematics*, 89(1):175–196, 2023.

[20] M. Bellare, O. Goldreich, and M. Sudan. Free bits, PCPs, and nonapproximability – towards tight results. *SIAM Journal on Computing*, 27(3):804–915, 1998.

[21] C. Belthangady and L. A. Royer. Applications, promises, and pitfalls of deep learning for fluorescence image reconstruction. *Nature methods*, 16(12):1215–1225, 2019.

[22] J. Ben-Artzi, M. J. Colbrook, A. C. Hansen, O. Nevanlinna, and M. Seidel. Computing spectra – On the solvability complexity index hierarchy and towers of algorithms. *arXiv:1508.03280v5*, 2020.

[23] J. Ben-Artzi, A. C. Hansen, O. Nevanlinna, and M. Seidel. New barriers in complexity theory: On the solvability complexity index and the towers of algorithms. *Comptes Rendus Mathematique*, 353(10):931 – 936, 2015.

[24] J. Ben-Artzi, M. Marletta, and F. Rösler. Computing the sound of the sea in a seashell. *Foundations of Computational Mathematics*, 22(3):1–35, 2021.

[25] A. Ben-Tal and A. S. Nemirovski. *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001.

[26] J. Bigot, C. Boyer, and P. Weiss. An analysis of block sampling strategies in compressed sensing. *IEEE Transactions on Information Theory*, 62(4):2125–2139, 2016.

[27] E. Bishop. *Foundations of Constructive Analysis*. McGraw-Hill Series in higher mathematics. McGraw-Hill, 1967.

[28] L. Blum, F. Cucker, M. Shub, and S. Smale. *Complexity and Real Computation*. Springer-Verlag, Berlin, Heidelberg, 1997.

[29] L. Blum, M. Shub, and S. Smale. On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. *Bulletin of the American Mathematical Society*, 21(1):1–46, 1989.

[30] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[31] C. Boyer, J. Bigot, and P. Weiss. Compressed sensing with structured sparsity and structured acquisition. *Applied and Computational Harmonic Analysis*, 46(2):312 – 350, 2019.

[32] S. Bubeck, Y. T. Lee, E. Price, and I. Razenshteyn. Adversarial examples from computational constraints. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 831–840. PMLR, 09–15 Jun 2019.

[33] M. Burger and T. Roith. Learning in image reconstruction: A cautionary tale. *SIAM News*, 57(08), Oct 2024.

[34] P. Bürgisser and F. Cucker. *Condition : the geometry of numerical algorithms*. Grundlehren der mathematischen Wissenschaften. Springer, Berlin, Heidelberg, New York, 2013.

[35] E. J. Candès, J. Romberg, and T. Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52(2):489–509, 2006.

[36] N. Carlini and D. Wagner. Audio adversarial examples: Targeted attacks on speech-to-text. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 1–7. IEEE, 2018.

[37] C. Choi. 7 revealing ways AIs fail: Neural networks can be disastrously brittle, forgetful, and surprisingly bad at math. *IEEE Spectrum*, 58(10):42–47, 2021.

[38] C. Choi. Some AI systems may be impossible to compute. *IEEE Spectrum*, March, 2022.

[39] A. Cohen, W. Dahmen, and R. DeVore. Compressed sensing and best k-term approximation. *Journal of the American Mathematical Society*, 22(1):211–231, 2009.

[40] M. Colbrook. On the computation of geometric features of spectra of linear operators on Hilbert spaces. *Foundations of Computational Mathematics*, 24, 12 2022.

[41] M. J. Colbrook, V. Antun, and A. C. Hansen. The difficulty of computing stable and accurate neural networks: On the barriers of deep learning and Smale's 18th problem. *Proceedings of the National Academy of Sciences*, 119(12):e2107151119, 2022.

[42] E. Creamer. 'Hallucinate' chosen as Cambridge dictionary's word of the year. *The Guardian*, available at `https://www.theguardian.com/books/2023/nov/15/hallucinate-cambridge-dictionary-word-of-the-year` (15. Nov, 2023).

[43] F. Cucker and S. Smale. Complexity estimates depending on condition and round-off error. *Journal of the ACM*, 46(1):113–184, 1999.

[44] D. L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006.

[45] A. Fannjiang and T. Strohmer. The numerics of phase retrieval. *Acta Numerica*, 29:125–228, 2020.

[46] A. Fawzi, H. Fawzi, and O. Fawzi. Adversarial vulnerability for any classifier. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[47] A. Fawzi, H. Fawzi, and O. Fawzi. Adversarial vulnerability for any classifier. In *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*, pages 1186–1195, 2018.

[48] C. Fefferman, A. C. Hansen, and S. Jitomirskaya, editors. *Computational mathematics in computer assisted proofs*, American Institute of Mathematics Workshops. American Institute of Mathematics, 2022. Available online at `https://aimath.org/pastworkshops/compproofsvrep.pdf`.

[49] C. Fefferman and B. Klartag. Fitting a C^m-Smooth Function to Data II. *Revista Matematica Iberoamericana*, 25(1):49 – 273, 2009.

[50] C. L. Fefferman and B. Klartag. Fitting a C^m-smooth function to data. I. *Annals of Mathematics*, 169(1):315–346, 2009.

[51] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Interactive proofs and the hardness of approximating cliques. *Journal of the ACM*, 43(2):268–292, 1996.

[52] S. G. Finlayson, J. D. Bowers, J. Ito, J. L. Zittrain, A. L. Beam, and I. S. Kohane. Adversarial attacks on medical machine learning. *Science*, 363(6433):1287–1289, 2019.

[53] L. E. Gazdag, A. Bastounis, and A. C. Hansen. Generalised hardness of approximation and the SCI hierarchy – On determining the boundaries of training algorithms in AI. *Foundations of Computational Mathematics*, (to appear).

[54] M. Genzel, J. Macdonald, and M. März. Solving inverse problems with deep neural networks – Robustness included? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):1119–1134, 2022.

[55] J. Gilmer, L. Metz, F. Faghri, S. S. Schoenholz, M. Raghu, M. Wattenberg, and I. J. Goodfellow. Adversarial spheres. In *6th International Conference on Learning Representations*, 2018.

[56] N. M. Gottschling, V. Antun, A. C. Hansen, and B. Adcock. The troublesome kernel: On hallucinations, no free lunches, and the accuracy-stability tradeoff in inverse problems. *SIAM Review*, 67(1):73–104, 2025.

[57] P. Grohs and F. Voigtlaender. Proof of the theory-to-practice gap in deep learning via sampling complexity bounds for neural network approximation spaces. *Foundations of Computational Mathematics*, 24(4):1085–1143, 2024.

[58] K. Hammernik, T. Klatzer, E. Kobler, M. P. Recht, D. K. Sodickson, T. Pock, and F. Knoll. Learning a variational network for reconstruction of accelerated MRI data. *Magnetic Resonance in Medicine*, 79(6):3055–3071, 2018.

[59] Y. Han and J. C. Ye. Framing U-Net via deep convolutional framelets: Application to sparse-view CT. *IEEE Transactions on Medical Imaging*, 37(6):1418–1429, 2018.

[60] A. C. Hansen. On the solvability complexity index, the n-pseudospectrum and approximations of spectra of operators. *Journal of the American Mathematical Society*, 24(1):81–124, 2011.

[61] J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, 182(1):105–142, 1999.

[62] J. Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, 2001.

[63] T. Hastie, R. Tibshirani, and M. Wainwright. *Statistical Learning with Sparsity: The Lasso and Generalizations*. Chapman and Hall/CRC, Boca Raton, FL, 2015.

[64] D. Heaven. Why deep-learning AIs are so easy to fool. *Nature*, 574(7777):163–166, Oct. 2019.

[65] Y. Huang et al. Some investigations on robustness of deep learning in limited angle tomography. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 145–153. Springer, 2018.

[66] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry. Adversarial examples are not bugs, they are features. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[67] K. H. Jin, M. T. McCann, E. Froustey, and M. Unser. Deep convolutional neural network for inverse problems in imaging. *IEEE Transactions on Image Processing*, 26(9):4509–4522, 2017.

[68] A. Juditsky, F. Kilinç-Karzan, A. Nemirovski, and B. Polyak. Accuracy guaranties for ℓ_1 recovery of block-sparse signals. *The Annals of Statistics*, 40(6):3077 – 3107, 2012.

[69] A. B. Juditsky, F. Kilinç-Karzan, and A. Nemirovski. Verifiable conditions of ℓ_1-recovery for sparse signals with sign restrictions. *Mathematical Programming*, 127(1):89–122, 2011.

[70] R. H. Keshavan, A. Montanari, and S. Oh. Matrix completion from a few entries. *IEEE Transactions on Information Theory*, 56(6):2980–2998, June 2010.

[71] K. Ko. *Complexity Theory of Real Functions*. Birkhauser, Boston, MA, 1991.

[72] J. C. Lagarias, B. Poonen, and M. H. Wright. Convergence of the restricted nelder–mead algorithm in two dimensions. *SIAM Journal on Optimization*, 22(2):501–532, 2012.

[73] R. F. Laine, I. Arganda-Carreras, R. Henriques, and G. Jacquemet. Avoiding a replication crisis in deep-learning-based bioimage analysis. *Nature Methods*, 18(10):1136–1144, 2021.

[74] L. Lovasz. *An Algorithmic Theory of Numbers, Graphs and Convexity*. CBMS-NSF Regional Conference Series in Applied Mathematics. Society for Industrial and Applied Mathematics, 1987.

[75] A. Malek, Y. Abbasi-Yadkori, and P. Bartlett. Linear programming for large-scale markov decision problems. In E. P. Xing and T. Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 496–504, Bejing, China, 22–24 Jun 2014. PMLR.

[76] R. Mazumder, T. Hastie, and R. Tibshirani. Spectral regularization algorithms for learning large incomplete matrices. *Journal of Machine Learning Research*, 11:2287–2322, 2010.

[77] M. T. McCann, K. H. Jin, and M. Unser. Convolutional neural networks for inverse problems in imaging: A review. *IEEE Signal Process Magazine*, 34(6):85–95, 2017.

[78] V. Monga, Y. Li, and Y. C. Eldar. Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing. *IEEE Signal Processing Magazine*, 38(2):18–44, 2021.

[79] S. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard. Universal adversarial perturbations. In *IEEE Conference on computer vision and pattern recognition*, pages 86–94, July 2017.

[80] M. J. Muckley, B. Riemenschneider, A. Radmanesh, S. Kim, G. Jeong, J. Ko, Y. Jun, H. Shin, D. Hwang, M. Mostapha, S. Arberet, D. Nickel, Z. Ramzi, P. Ciuciu, J.-L. Starck, J. Teuwen, D. Karkalousos, C. Zhang, A. Sriram, Z. Huang, N. Yakubova, Y. W. Lui, and F. Knoll. Results of the 2020 fastMRI challenge for machine learning MR image reconstruction. *IEEE Transactions on Medical Imaging*, 40(9):2306–2317, 2021.

[81] Y. Nesterov. *Introductory lectures on convex optimization : a basic course*. Applied optimization. Kluwer Academic Publ., Boston, Dordrecht, London, 2004.

[82] Y. Nesterov and A. Nemirovskii. *Interior-point polynomial algorithms in convex programming*. SIAM studies in applied mathematics. Society for Industrial and Applied Mathematics, Philadelphia, 1994.

[83] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, New York, NY, USA, second edition, 2006.

[84] G. Ongie, A. Jalal, C. A. Metzler, R. G. Baraniuk, A. G. Dimakis, and R. Willett. Deep learning techniques for inverse problems in imaging. *IEEE Journal on Selected Areas in Information Theory*, 1(1):39–56, 2020.

[85] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons, New York, 1994. Reprinted by Wiley-Interscience, 2005.

[86] C. Qiao, D. Li, Y. Guo, C. Liu, T. Jiang, Q. Dai, and D. Li. Evaluation and development of deep neural networks for image super-resolution in optical microscopy. *Nature Methods*, 18(2):194–202, 2021.

[87] Z. Ramzi, J. Starck, and P. Ciuciu. XPDNet for MRI reconstruction: An application to the 2020 fastMRI challenge. In *2021 ISMRM annual meeting, no. Abstract*, volume 275, 2021.

[88] J. Schlemper, J. Caballero, J. V. Hajnal, A. Price, and D. Rueckert. A deep cascade of convolutional neural networks for MR image reconstruction. In *Int. Conf. Inf. Proc. Med. Imaging*, pages 647–658. Springer, 2017.

[89] L. Schmidt, S. Santurkar, D. Tsipras, K. Talwar, and A. Madry. Adversarially robust generalization requires more data. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[90] A. Shafahi, W. R. Huang, C. Studer, S. Feizi, and T. Goldstein. Are adversarial examples inevitable? In *International Conference on Learning Representations*, 2019.

[91] S. Smale. Mathematical problems for the next century. In V. I. Arnold, M. Atiyah, P. Lax, and B. Mazur, editors, *Mathematics: Frontiers and Perspectives*, pages 271–294. American Mathematical Society, Providence, RI, 2000.

[92] C. Smith. Hallucinations could blunt chatGPT's success. *IEEE Spectrum*, March, 2023.

[93] R. I. Soare. *Turing Computability: Theory and Applications*. Springer Publishing Company, Incorporated, 1st edition, 2016.

[94] M. Sudan. Probabilistically checkable proofs. *Communications of the ACM*, 52(3):76–84, 2009.

[95] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *Proceedings of the International Conference on Learning Representations*, 2014.

[96] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry. Robustness may be at odds with accuracy. In *International Conference on Learning Representations*, 2019.

[97] A. M. Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proc. London Math. Soc. (2)*, 42(3):230–265, 1936.

[98] I. Y. Tyukin, D. J. Higham, and A. N. Gorban. On adversarial examples and stealth attacks in artificial intelligence systems. *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6, 2020.

[99] G. Wang, J. C. Ye, K. Mueller, and J. A. Fessler. Image reconstruction is a new frontier of machine learning. *IEEE Transactions on Medical Imaging*, 37(6):1289–1296, 2018.

[100] M. H. Wright. The interior-point revolution in optimization: history, recent developments, and lasting consequences. *Bulletin of the American Mathematical Society*, 42(1):39–56, 2005.

[101] S. J. Wright. *Primal-Dual Interior-Points Methods*. SIAM, Philadelphia, PA, USA, 1997.

[102] B. Zhu, J. Z. Liu, S. F. Cauley, B. R. Rosen, and M. S. Rosen. Image reconstruction by domain-transform manifold learning. *Nature*, 555(7697):487, 03 2018.

[103] J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani. 1-norm support vector machines. In S. Thrun, L. K. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16 (NIPS 2003)*, pages 49–56. MIT Press, 2003.

DEPARTMENT OF MATHEMATICS, KING'S COLLEGE LONDON
Email address: alexander.bastounis@kcl.ac.uk

DEPARTMENT OF APPLIED MATHEMATICS AND THEORETICAL PHYSICS, UNIVERSITY OF CAMBRIDGE
Email address: a.hansen@damtp.cam.ac.uk