GENERALISED HARDNESS OF APPROXIMATION AND THE SCI HIERARCHY – ON DETERMINING THE BOUNDARIES OF TRAINING ALGORITHMS IN AI

LUCA EVA GAZDAG AND ANDERS C. HANSEN

ABSTRACT. Hardness of approximation (HA) – the phenomenon that, assuming P $\neq$ NP, one can easily compute an ϵ-approximation to the solution of a discrete computational problem for $\epsilon > \epsilon_0 > 0$, but for $\epsilon < \epsilon_0$ (the approximation threshold) it suddenly becomes intractable – is a core phenomenon in the foundations of computations that has transformed computer science. In this paper we study the newly discovered phenomenon in the foundations of computational mathematics: generalised hardness of approximation (GHA) – which in spirit is close to classical HA in computer science. However, GHA is typically independent of the P vs. NP question in many cases. Thus, it requires a new mathematical framework that we initiate in this paper. We demonstrate the hitherto undiscovered phenomenon that GHA happens when using AI techniques in order to train optimal neural networks (NNs). In particular, for any non-zero underdetermined linear inverse problem the following phase transition can occur: One can prove the existence of optimal NNs for solving the problem but they can only be computed to a certain accuracy $\epsilon_0 > 0$. Below the approximation threshold ϵ_0 – not only does it become intractable to compute the NN – it becomes impossible regardless of computing power, and no randomised algorithm can solve the problem with probability better than 1/2. Moreover, despite the existence of a stable optimal NN, any attempts of computing it below the approximation threshold ϵ_0 will yield an unstable NN. Our results use and extend the current mathematical framework of the Solvability Complexity Index (SCI) hierarchy and facilitate a program for detecting the GHA phenomenon throughout computational mathematics and AI.

CONTENTS

1. INTRODUCTION

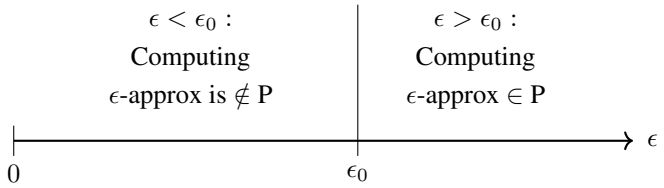
Hardness of approximation [7, 8, 16, 50, 58, 59, 88] describes the following phenomenon: Computing an ϵ-approximation to a discrete computational problem is in P (solvable in polynomial time) for $\epsilon > \epsilon_0$ but the problem often becomes NP-hard (or NP-complete) when $\epsilon < \epsilon_0$, where ϵ_0 is the approximation threshold.

2020 *Mathematics Subject Classification.* 65Yxx, 03D55 (primary) and 90C26, 15A29, 68Q87 (secondary).

Key words and phrases. Generalised hardness of approximation, phase transitions, Solvability Complexity Index hierarchy, boundaries of AI, foundations of computational mathematics.

In particular, assuming P≠NP we have the following phase transition:

Classical phase transition at ϵ_0 in hardness of approximation



If P=NP the phenomenon may not occur, as there may not be any phase transition between the problem being in P for $\epsilon > \epsilon_0$ and not being in P for $\epsilon < \epsilon_0$. However, if P≠NP then there is a phase transition as the problem ceases to be in P when $\epsilon < \epsilon_0$. The phenomenon is a well established part of the foundations of computations that has been the subject of several Gödel and Nevanlinna Prizes. The hardness of approximation phenomenon is almost exclusively associated with combinatorial computational problems, however – as we discuss in this paper and as was discovered in [13] and subsequently in [40] – a generalised form of this phenomenon can happen in other areas of the computational sciences regardless of the P vs. NP question. In this paper we lay down the foundations for a theory on generalised hardness of approximation and study the phenomenon in connection with AI methods for underdetermined inverse problems. In particular, we demonstrate a new discovery in generalised hardness of approximation.

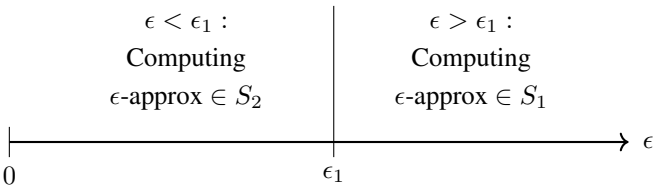
Consider a computational problem, for example, computing the minimiser of a convex optimisation problem or computing the optimal neural network for an inverse problem. We now informally define the general concept of approximate computational problem in computational mathematics.

Approximate computational problem: Given an $\epsilon > 0$, the ϵ-approximate computational problem is the problem of computing an approximation that is no more than ϵ away from the true solution – in some appropriate predefined metric. Suppose that we have a computational problem and two classes of approximate computational problems S_1 and S_2 with $S_1 \cap S_2 = \emptyset$ and $\epsilon_1 \geq \epsilon_2 > 0$. We say that the computational problem has an (S_1, S_2)-phase transition at (ϵ_1, ϵ_2) if we have the following:

$$\begin{aligned} &\text{The approximate computational problem} \in S_1, \text{ for } \epsilon > \epsilon_1, \\ &\text{The approximate computational problem} \in S_2, \text{ for } \epsilon < \epsilon_2. \end{aligned} \tag{1.1}$$

If $\epsilon_1 = \epsilon_2$ in (1.1) we say that the phase transition is sharp and call ϵ_1 the approximation threshold. Schematically, the concept of generalised hardness of approximation with a sharp phase transition can be visualised as follows:

Sharp phase transition at ϵ_1 in generalised hardness of approximation



(1.2)

This definition can of course be generalised to any family of collections $S_1, \ldots, S_k$, $k > 1$ of computational problems with $S_j \cap S_i = \emptyset$ for $j \neq i$ and $\epsilon_1, \ldots, \epsilon_{2(k-1)} > 0$ with $\epsilon_1 \geq \epsilon_2 > \epsilon_3 \geq \epsilon_4 > \ldots > \epsilon_{2k-3} \geq \epsilon_{2(k-1)}$ as follows. We say that we have an $(S_1, \ldots, S_k)$-phase transition at $(\epsilon_1, \ldots, \epsilon_{2(k-1)})$ if we have the following:

$$\begin{aligned} &\text{The approximate computational problem} \in S_1, \text{ for } \epsilon > \epsilon_1, \\ &\text{The approximate computational problem} \in S_2, \text{ for } \epsilon_3 < \epsilon < \epsilon_2, \\ &\qquad\qquad \vdots \\ &\text{The approximate computational problem} \in S_{k-1}, \text{ for } \epsilon_{2k-3} < \epsilon < \epsilon_{2k-4}, \\ &\text{The approximate computational problem} \in S_k, \text{ for } \epsilon < \epsilon_{2(k-1)}, \end{aligned} \tag{1.3}$$

where we say that the phase transition in (1.3) is sharp at ϵ_{2j-1} if $\epsilon_{2j-1} = \epsilon_{2j}$ for some integer j and call ϵ_{2j-1} an approximation threshold.

As the generalised hardness of approximation phenomenon can be independent of the P vs. NP question, one needs a mathematical theory outside of computer science. The mathematics behind the Solvability Complexity Index (SCI) hierarchy is particularly well suited for this. This is a framework in the foundations of computational mathematics that establishes the boundaries of what can be achieved in computational mathematics and has been used in a wide collection of areas in computational mathematics. The SCI hierarchy was introduced in [56] and further developed in [17, 18, 38, 39]. Further results and in particular new techniques for the SCI hierarchy were also developed in [13] in connection with the extended Smale's 9th problem, where the first discovery of generalised hardness of approximation was encountered in optimisation problems such as linear programming [14] – linked to robust optimisation [21, 22] – basis pursuit [72] and lasso [90]. The GHA phenomenon described in this paper, however, is different to the phenomenon described in the work on the extended Smale's 9th problem [13], as the condition analysis [30, 78–80] is different (see also Remark 2.2). Further developments on the SCI hierarchy followed in [40], where generalised harness of approximation has been identified in deep learning, where the phase transitions depend on both accuracy and the amount of data needed, see §3 for details. The theory of the SCI hierarchy is now comprehensive [13, 17–20, 38–40, 45, 56, 57, 69, 70, 87, 96, 97], where more details can be found in §3. Generalised hardness of approximation fits naturally into the SCI framework and allows for analysis in any computational model such as the Blum-Shub-Smale (BSS) [26] or the Turing model [91]. Phase transitions can potentially behave differently in these two models, however, the results in this paper are universal for any model of computation.

1.1. **Optimality of neural networks for underdetermined linear systems.** We study the following underdetermined systems of equations. Let $A : \mathbb{R}^N \to \mathbb{R}^m$ be a linear mapping with non-trivial kernel and let $\mathcal{M}_1 \subset \mathbb{R}^N$ be some subset that we call initial domain. We now consider the following inverse problem:

$$\text{Given measurements } y = Ax + e \text{ of } x \in \mathcal{M}_1, \text{ recover } x \in \mathcal{M}_1, \tag{1.4}$$

where $e \in \mathbb{R}^m$ is a potential noise vector. These types of problems have been extensively studied in sparse recovery and compressed sensing when $\mathcal{M}_1$ is, for example, a collection of sparse vectors or vectors with some structured sparsity [3–5, 11, 12, 23, 28, 31, 32, 44, 46, 64, 65]. However, recent developments have led to a plethora of AI techniques [6, 10, 54, 55, 63, 68, 74, 99] for solving these types of problems. More specifically, deep learning approaches have become popular over the last years, where one trains a neural network $\mathbf{N} : \mathbb{R}^m \to \mathbb{R}^N$ from some training set $\mathcal{T} \subset \mathbb{R}^N \times \mathbb{R}^m$ of the form $\mathcal{T} = \{(x_j, Ax_j)\}_{j=1}^r$, for some $\{x_j\}_{j=1}^r \subset \mathcal{M}_1$. However, it has been established that there is a fundamental stability-accuracy trade-off for such methods [6, 40, 52, 54]. Indeed, great accuracy on certain inputs may cause unstable behaviour and hallucinations in the reconstruction in form of false information in other reconstructed objects. Thus, it becomes important to establish the optimal choice of neural network in order to optimise performance. Such optimal maps have already been a focal point in approximation theory.

Indeed, approximation theory has a rich tradition in the theory of optimal approximations and optimal reconstruction maps. A particular example is the seminal work of A. Cohen, W. Dahmen and R. DeVore [37], where they define the concept of optimal reconstruction maps for underdetermined inverse problems.

Definition 1.1 (Optimality in the sense of Cohen, Dahmen & DeVore [37])**.** 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. 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 $\varphi_\epsilon : \mathcal{M}_2 \to \mathbb{R}^N$ is a family of approximate optimal maps of $(A, \mathcal{M}_1)$ if for all $\epsilon \in (0, 1]$,

$$\sup_{x \in \mathcal{M}_1} d_1^H(\varphi_\epsilon(Ax), x) \leq c_{\text{opt}}(A, \mathcal{M}_1) + \epsilon, \tag{1.5}$$

and that φ_0 is an optimal map for $(A, \mathcal{M}_1)$ if φ_0 satisfies (1.5) with $\epsilon = 0$.

The first key question is whether there exist neural networks that are optimal maps for different underdetermined inverse problems. Moreover, one can ask if such an optimal neural network can be trained from training data. Let $\mathcal{F}$ be collection of underdetermined inverse problems $(A, \mathcal{M}_1)$ and let $\mathcal{NN}_{m,N}$ be the set of NNs mapping $\mathbb{R}^m \to \mathbb{R}^N$ that are bounded on $\bigcup_{(A,\mathcal{M}_1)\in\mathcal{F}} A(\mathcal{M}_1)$, where we will throughout the paper use the more compact notation

$$\bigcup_{\mathcal{F}} \mathcal{M}_2 := \bigcup_{(A,\mathcal{M}_1)\in\mathcal{F}} A(\mathcal{M}_1).$$

We also require that the NNs in $\mathcal{NN}_{m,N}$ have a finite fixed set of nonlinearities that are computable (see Definition 4.11). We then have the following question:

> *Let $A : \mathbb{R}^N \to \mathbb{R}^m$ be fixed, and consider underdetermined linear problems $(A, \mathcal{M}_1)$ as in* (1.4)*. Do there exist neural networks that are optimal mappings for $(A, \mathcal{M}_1)$? Moreover, suppose that the answer to the first question is 'yes', and that there is a corresponding collection Ω of training sets $\mathcal{T} = \{(x_j, Ax_j)\}$ with $\{x_j\} \subset \mathcal{M}_1$ and a mapping $\Xi : \Omega \rightrightarrows \mathcal{NN}_{m,N}$ such that $\Xi(\mathcal{T})$ contains all the neural networks that are optimal for $(A, \mathcal{M}_1)$. Does there then exist an algorithm $\Gamma : \Omega \to \mathcal{NN}_{m,N}$ that approximates any neural network in $\Xi(\mathcal{T})$ for all inputs $\mathcal{T} \in \Omega$ to ϵ-accuracy for a given $\epsilon > 0$?*

The main result of this paper is that the answer to the first question above is often 'yes', but typically there is a generalised hardness of approximation phenomenon such that the optimal neural network can only be computed to a certain accuracy. We need to make the notion of 'computed to ϵ-accuracy' precise in this case.

1.2. **The model of computation.** Fix an $A : \mathbb{R}^N \to \mathbb{R}^m$, and consider a collection $\mathcal{F}$ of inverse problems $(A, \mathcal{M}_1)$ with a corresponding collection Ω of training sets $\mathcal{T}$ and a mapping

$$\Xi : \Omega \rightrightarrows \mathcal{NN}_{m,N}, \text{ such that } \Xi(\mathcal{T}) = \left\{ \mathbf{N}_{opt}^{\mathcal{M}_1} \ : \ \mathbf{N}_{opt}^{\mathcal{M}_1} \text{ is optimal for } (A, \mathcal{M}_1) \right\}. \tag{1.6}$$

Note that Ξ can be multivalued, meaning that the optimal neural network may not be unique, hence the double arrow notation (see Remark 1.3 regarding the measure of error in that case). We will in some cases be interested in bounding the derivative of the optimal mappings to ensure stability. Thus, assuming smoothness of the optimal neural networks, we define $\Xi_D : \Omega \rightrightarrows \mathcal{NN}_{m,N}$ for a positive number D to be

$$\Xi_D(\mathcal{T}) = \left\{ \mathbf{N}_{opt}^{\mathcal{M}_1} \in \Xi(\mathcal{T}) \ : \ \sup_{y \in A(\mathcal{M}_1)} \|\mathrm{D}\mathbf{N}_{opt}^{\mathcal{M}_1}(y)\|_{op} \leq D \right\}, \tag{1.7}$$

where $\mathrm{D}\mathbf{N}_{opt}^{\mathcal{M}_1}(y)$ denotes the Jacobian of $\mathbf{N}_{opt}^{\mathcal{M}_1}$ evaluated at y. Note that $\Xi_D(\mathcal{T})$ could potentially be empty for small values of D. We want to compute $\mathbf{N}_{opt}^{\mathcal{M}_1}$ – or more precisely, one of the optimal neural networks – and in practice we will compute an approximation.

Remark 1.2 (Non-smooth optimal neural networks)**.** In the case of non-smooth neural networks, the bound on the Jacobian in (1.7) can be replaced by a bound on the Lipschitz constant. In our theorems the optimal neural networks will be smooth.

We take the word 'compute' literally, and thus the following quote explains the situation succinctly:

"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 [86])

Indeed, often a training set $\mathcal{T} = \{(x_j, y_j)\}_{j=1}^r$ will not be represented exactly on a computer. This is because A could have rows from the discrete Fourier transform, for example – as in accelerated (subsampled) Magnetic Resonance Imaging (MRI) – and thus A contains irrational numbers. Another issue is that an overwhelming amount of modern software used is based on floating-point arithmetic, and hence even if the input is rational, there will be inexactness due to the floating-point representation. For example, $1/3$ can only be approximated in finite base-2 arithmetic, giving rise to round-off approximation.

Thus, every element $(x_j, y_j) \in \mathcal{T}$ – as input to an algorithm – is represented by a sequence of approximations $\{(x_j^n, y_j^n)\}_{n\in\mathbb{N}}$ such that

$$\|x_j^n - x_j\|_{\ell^2} \leq 2^{-n}, \qquad \|y_j^n - y_j\|_{\ell^2} \leq 2^{-n}, \quad \forall n \in \mathbb{N}, \tag{1.8}$$

and we require that a successful algorithm should work on any such approximating sequence. Note that this extended computational model of having inexact input is standard and can be found in many areas of the mathematical literature, and we mention only a small subset here including the work in [24, 29, 42, 48, 49, 66, 67].

Remark 1.3 (Oracles)**.** The above model means that for each training set $\mathcal{T} = \{(x_j, y_j)\}_{j=1}^r \in \Omega$, we have infinitely many collections of approximate sequences

$$\tilde{\mathcal{T}} = \{\{(x_j^n, y_j^n)\}_{n\in\mathbb{N}} \,|\, (x_j^n, y_j^n) \text{ satisfy (1.8)}, j = 1, \ldots, r\}. \tag{1.9}$$

A sequence of approximations is provided to the algorithm through an 'oracle'. For example, in the case of a Turing machine [91], this would be through an oracle input tape (see [66] for the standard setup), or in the case of a Blum-Shub-Smale (BSS) machine [25], this would be through an oracle node. The algorithm can thus ask for an approximation to any given accuracy as in (1.8), and use as many queries as desired.

Definition 1.4 (Computing the neural network $\mathbf{N}_{opt}^{\mathcal{M}_1}$ to ϵ-accuracy)**.** We say that the mapping Ξ in (1.6) can be computed to ϵ-accuracy if there exists an algorithm Γ such that for any $\mathcal{T} \in \Omega$,

$$\inf_{\mathbf{N}_{opt}^{\mathcal{M}_1} \in \Xi(\mathcal{T})} \sup_{y \in \bigcup_{\mathcal{F}} \mathcal{M}_2} \|\Gamma(\tilde{\mathcal{T}}, \epsilon)(y) - \mathbf{N}_{opt}^{\mathcal{M}_1}(y)\|_{\ell_2} \leq \epsilon, \quad \forall \tilde{\mathcal{T}} \text{ as in (1.9)}. \tag{1.10}$$

Similarly, for Ξ_D in (1.7) and any $\mathcal{T} \in \Omega$,

$$\begin{aligned} \inf_{\mathbf{N}_{opt}^{\mathcal{M}_1} \in \Xi(\mathcal{T})} \sup_{y \in \bigcup_{\mathcal{F}} \mathcal{M}_2} & \|\Gamma(\tilde{\mathcal{T}}, \epsilon)(y) - \mathbf{N}_{opt}^{\mathcal{M}_1}(y)\|_{\ell_2} \\ & \vee \|\mathrm{D}\Gamma(\tilde{\mathcal{T}}, \epsilon)(y) - \mathrm{D}\mathbf{N}_{opt}^{\mathcal{M}_1}(y)\|_{op} \leq \epsilon, \quad \forall \tilde{\mathcal{T}} \text{ as in (1.9)}, \end{aligned} \tag{1.11}$$

where $\mathrm{D}\Gamma(\tilde{\mathcal{T}}, \epsilon)(y)$ and $\mathrm{D}\mathbf{N}_{opt}^{\mathcal{M}_1}$ denote the derivatives of $\Gamma(\tilde{\mathcal{T}}, \epsilon)(y)$ and $\mathbf{N}_{opt}^{\mathcal{M}_1}$ respectively, and where $\|\cdot\|_{op}$ denotes the standard operator norm as defined in eq. (5.2).

Remark 1.5 (Markov model – Computability of the 'oracle')**.** We want to emphasise that our results hold regardless of the computational model for the 'oracle'. In particular, all our results hold in the Markov model based on the Markov algorithm [81] – i.e. when the inexact input is required to be computable, in particular when $\{(x_j^n, y_j^n)\}_{n\in\mathbb{N}}$ is a computable sequence for $j = 1, \ldots, r$. See also Remark 4.7 and Remark 4.15.

Remark 1.6 (Slight abuse of notation for $\Gamma(\mathcal{T}, \epsilon)$)**.** As the algorithm must work on any such representation $\tilde{\mathcal{T}}$ of $\mathcal{T}$ we will frequently abuse notation and write $\Gamma(\mathcal{T}, \epsilon)$ instead of $\Gamma(\tilde{\mathcal{T}}, \epsilon)$. Also, if a result holds for only one specific $\epsilon \in \mathbb{R}_+$, we will frequently drop writing ϵ as an input to Γ, writing $\Gamma(\mathcal{T})$ in place of $\Gamma(\mathcal{T}, \epsilon)$.

2. Main results – Generalized hardness of approximation (GHA)

Our main results are gathered in two main theorems, both presented formally in §5. The first main theorem asserts that there exists phase transitions for a large class of inverse problems for the computational problem described above. The second main theorem asserts fundamental computational barriers when attempting to train neural networks to solve (1.4) in the standard computational model with inexact input. In addition, the second theorem illustrates how small changes in the training set can lead to the collapse of the accuracy of a working algorithm. In particular, the theorem demonstrates how phase transitions in (1.1) and (1.2) can suddenly change with the training data.

For all impossibility results we use a generalized model for computation – that is also used in [13, 15, 38, 40, 56] – in order to obtain universal lower bounds regardless of the computational model. More precisely, when we refer to an algorithm we mean a so called *general algorithm* (the formalism of this is defined in section 4.1). However, for all positive results – of the form "there exists an algorithm" – we use the Turing model to achieve the strongest upper bounds possible. Throughout the paper $B_1(0) = \{(x, y) \in \mathbb{R}^{N\times m} : \|x\|_{\ell_2}, \|y\|_{\ell_2} \leq 1\}$.

Theorem 2.1 (Generalized hardness of approximation – Phase transitions for computing optimal NNs)**.** *For any integers $N > m$ and any $\beta_{max} \geq \beta_{min} > 0$, consider any fixed non-zero linear map $A : \mathbb{R}^N \to \mathbb{R}^m$ such that the spectrum $\mathrm{Sp}(AA^*) \subset [\beta_{min}^2, \beta_{max}^2]$. Then, for any rational $\epsilon_1 \in (0, 3/8]$ and any integer $\ell \geq 2$, there exists a collection of initial domains $\mathcal{M}_1 \subset \mathbb{R}^N$, a domain Ω (as described in §1.1) of training sets $\mathcal{T}$ with $|\mathcal{T}| = \ell$, $\mathcal{T} \subseteq B_1(0)$, a $D \in \mathbb{N}$ and a mapping $\Xi_D : \Omega \rightrightarrows \mathcal{NN}_{m,N}$ as in* (1.7)*. In particular, all optimal neural networks $\mathbf{N}_{opt}^{\mathcal{M}_1} \in \Xi_D(\mathcal{T}) \neq \emptyset$ have uniformly bounded (by D) Jacobians. However, the following happens simultaneously:*

- *(i) No algorithm, not even randomised, can approximate an optimal neural network $\mathbf{N}_{opt}^{\mathcal{M}_1} \in \Xi(\mathcal{T})$ for all inputs $\mathcal{T} \in \Omega$ 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 does exists an algorithm Γ such that $\mathbf{N}_{\mathcal{T},\epsilon} = \Gamma(\mathcal{T}, \epsilon)$ is a NN that approximates an optimal neural network $\mathbf{N}_{opt}^{\mathcal{M}_1} \in \Xi(\mathcal{T})$ to accuracy $\epsilon > 2\epsilon_1$ for all $\mathcal{T} \in \Omega$ with the property that there exists a $K \in \mathbb{N}$ such that the Jacobian satisfies $\|\mathrm{D}\mathbf{N}_{\mathcal{T},\epsilon}(c)\|_{op} \leq K$ for all $c \in \mathbb{R}^m$ and for all $\mathcal{T} \in \Omega$.*
- *(iii) However, for any $K \in \mathbb{N}$, $\delta \in (0, \epsilon_1)$ and any algorithm Γ such that $\mathbf{N}_{\mathcal{T},\epsilon} = \Gamma(\mathcal{T}, \epsilon)$ is a NN that approximates an optimal neural network $\mathbf{N}_{opt}^{\mathcal{M}_1} \in \Xi(\mathcal{T})$ to accuracy $\epsilon \in (\epsilon_1, 2\epsilon_1 - \delta]$ for all $\mathcal{T} \in \Omega$ we have the following: There exists infinitely many $\mathcal{T} \in \Omega$ such that the Jacobian satisfies $\|\mathrm{D}\mathbf{N}_{\mathcal{T},\epsilon}(c)\|_{op} \geq K$ for some $c \in \mathbb{R}^m$. In particular, the Lipchitz constant of the NNs will blow up.*

Remark 2.2 (No $p = 2/3$ randomised algorithm with non-zero probability of not halting)**.** Note that the phase transition described above are different to the phase transitions described in [13]. Indeed, the phase transitions described in [13] would allow for a randomised algorithm with a non-zero probability of not halting to succeed with probability $p = 2/3$. This is not the case for the generalised hardness of approximation phenomenon described in Theorem 2.1. Hence, the phenomenon described in Theorem 2.1 is 'harder' and different to the generalised hardness of approximation phenomenon first discovered in [13].

Remark 2.3 (Consequences of Theorem 2.1)**.** Note that Theorem 2.1 yields the following.

(i) (**GHA in inverse problems independent of P vs NP**). Generalised hardness of approximation happens in underdetermined inverse problems independent of P vs NP. Indeed, since a general algorithm (Definition 4.3) – which is used in all lower bounds – is so powerful that it can simulate all types of algorithms in any reasonable computational model. In particular, any NP-hard problem in the Turing model can be solved in one operation by a general algorithm. Moreover, generalised hardness of approximation can happen for essentially any forward operator linear map

$A : \mathbb{R}^N \to \mathbb{R}^m$, and the approximation threshold $\epsilon_1 \in (0, 3/8]$. The new results suggest a classification program for determining the approximation threshold depending on conditions on A and $\mathcal{M}_1$.

(ii) (**GHA implies an accuracy-stability trade-off**). The accuracy-stability trade-off in AI methods for inverse problems is well documented empirically [6,40,52,54] and to some extent theoretically [40, 54]. Theorem 2.1 provides new theoretical understanding of this phenomenon, and demonstrates that any attempt of computing too accurate NNs will immediately yield arbitrarily unstable NNs. However, as long as the accuracy of the computed NN is above a certain threshold, stability can be achieved. Thus, there is an accuracy-stability trade-off and overperformance immediately yields instabilities.

Theorem 2.4 (Phase transitions – Rapid changes and the halting problem)**.** *Given any $N, m \in \mathbb{N}$ and a fixed non-zero linear map $A : \mathbb{R}^N \to \mathbb{R}^m$ with non-trivial kernel and an integer $\ell \geq 2$, there exists a collection of initial domains $\mathcal{M}_1 \subset \mathbb{R}^N$, a domain Ω (as described in §1.1) of training sets $\mathcal{T}$ with $|\mathcal{T}| = \ell$, $\mathcal{T} \subseteq B_1(0)$, and a mapping $\Xi : \Omega \rightrightarrows \mathcal{NN}_{m,N}$ as in* (1.6)*. In particular, there exists optimal neural networks $\mathbf{N}_{opt}^{\mathcal{M}_1} \in \Xi(\mathcal{T}) \neq \emptyset$. However, the following happens simultaneously:*

(i) The mapping Ξ cannot be computed. In particular, no algorithm that takes training sets $\mathcal{T} \in \Omega$ as inputs, can produce a neural network that approximates any $\mathrm{N}_{opt}^{\mathcal{M}_1} \in \Xi(\mathcal{T})$ to even one digit accuracy for all $\mathcal{T} \in \Omega$.

(ii) However, there exists an infinite collection of training sets $\{\mathcal{T}_i\}_{i\in\mathbb{N}} \subset \Omega$ and an algorithm Γ that takes inputs in Ω, that for any $\epsilon > 0$ and $\mathcal{T}_i$ produces an ϵ-approximation to an optimal neural network in $\Xi(\mathcal{T}_i)$. Yet, for any infinite sequence $\{\mathcal{T}_k\}_{k\in\mathbb{N}} \subset \Omega$ of different elements in Ω and any algorithm Γ that produces an ϵ-approximation to an optimal neural network in $\Xi(\mathcal{T}_k)$ for all $k \in \mathbb{N}$ we have the following. There exists an infinite subsequence $\{\mathcal{T}_{k_j}\}_{j\in\mathbb{N}} \subseteq \{\mathcal{T}_k\}_{k\in\mathbb{N}}$ such that for each $j \in \mathbb{N}$, there is an element $(x, y) \in \mathcal{T}_{k_j}$ and an element $(x', y') \in \mathbb{R}^N \times \mathbb{R}^m$ with $\|x'\|_{\ell_2}, \|y'\|_{\ell_2} \leq 1$ and $\|(x, y) - (x', y')\|_{\ell_2} \leq \sqrt{2}/4^j$ such that if we replace (x, y) with (x', y') then we obtain a new training set $\mathcal{T}'_{k_j} = [\mathcal{T}_{k_j} \setminus (x, y)] \cup (x', y') \in \Omega$ such that

$$\sup_{y\in\bigcup_{\mathcal{F}}\mathcal{M}_2} \|\Gamma(\mathcal{T}'_{k_j}, \epsilon)(y) - \mathbf{N}_{opt}^{\mathcal{M}'_1}(y)\|_{\ell_2} > 10^{-1}, \quad \forall\, \epsilon > 0,$$

for any $\mathbf{N}_{opt}^{\mathcal{M}'_1} \in \Xi(\mathcal{T}'_{k_j})$ optimal neural network for the inverse problem $(A, \mathcal{M}'_1)$.

(iii) Moreover, the statement in (ii) is true if we add one particular element, as opposed to changing one. In particular, for any $j \in \mathbb{N}$ we can add one element in $\mathcal{T}_{k_j}$ and thereby obtain a new training set $\mathcal{T}''_{k_j}$ with $|\mathcal{T}''_{k_j}| = \ell + 1$ such that $\Xi(\mathcal{T}''_{k_j})$ is well defined yielding optimal neural networks for an $\mathcal{M}''_{1,k_j}$ with $\mathcal{M}_{1,k_j} \subset \mathcal{M}''_{1,k_j}$, however Γ makes an error in the first digit on the input $\mathcal{T}''_{k_j}$.

(iv) Computing approximations to $\Xi : \Omega \rightrightarrows \mathcal{NN}_{m,N}$ to one correct digit is at least as hard as deciding the Halting problem.

Remark 2.5 (Consequences of Theorem 2.4)**.** Note that Theorem 2.4 yields the following.

(i) (**Phase transitions can change rapidly with training data**). Both replacing and adding elements in the training set can change the phase transition and the approximation threshold rapidly. Moreover, the training data that causes the change in the phase transition can be arbitrarily close to elements that are already in the original training set. Hence, the phase transitions can be highly unstable and thus the training process itself can be unstable.

(ii) (**Training NNs beyond the approximation threshold is at least as hard as the halting problem**). Theorem 2.4 demonstrates that training optimal NNs beyond the approximation threshold is least as hard as the halting problem, however, where the problem lies in the SCI hierarchy is an open question.

Remark 2.6. In both Theorem 2.1 and Theorem 2.4 the existence of inverse problems and families of training sets are stronger than mere existence results, in fact in section 5.2 and section 5.3 we give explicit descriptions of how these examples can be constructed. Also, in comparison with the results in [13] and [40], we wish to remark that theorem 2.1 can be used to construct examples of problems where an optimal neural network can be constructed to an accuracy of $K-1$ digits, however no algorithm can construct it to an accuracy of K digits and this can be done for any $K \in \mathbb{N}$.

2.1. **The main results in the framework of generalised hardness of approximation.** In the language of generalised hardness of approximation, Theorem theorem 2.1 can be stated as follows. Consider the following computational problem, ϵ-approximate computational problem and classes of ϵ-approximate computational problems:

(Ia) *Computational problem a:* Given the training set $\mathcal{T} \in \Omega$, compute an optimal neural network $\mathrm{N}_{opt}^{\mathcal{M}_1} \in \mathcal{NN}_{m,N}$ for the inverse problem $(A, \mathcal{M}_1)$ in the sense of (1.6).

(Ib) *Computational problem b:* Given the training set $\mathcal{T} \in \Omega$ and a positive number D, compute a stable optimal neural network $\mathrm{N}_{opt}^{\mathcal{M}_1} \in \mathcal{NN}_{m,N}$ for the inverse problem $(A, \mathcal{M}_1)$ in the sense of (1.7).

(IIa) *ϵ-approximate computational problem a:* Given the training set $\mathcal{T} \in \Omega$ compute an ϵ-approximation – in the sense of (1.10) – to an optimal neural network $\mathrm{N}_{opt}^{\mathcal{M}_1} \in \mathcal{NN}_{m,N}$.

(IIb) *ϵ-approximate computational problem b:* Given the training set $\mathcal{T} \in \Omega$ and a positive number D, compute an ϵ-approximation – in the sense of (1.11) – to a stable optimal neural network $\mathrm{N}_{opt}^{\mathcal{M}_1} \in \mathcal{NN}_{m,N}$.

(III) *Classes of ϵ-approximate computational problems:*

$$
\begin{aligned}
S_1^a(\epsilon) = {} & \text{The set of } \epsilon\text{-approximate computational problems for which} \\
& \exists \text{ an algorithm } \Gamma \text{ computing } \epsilon\text{-approximations in the sense of (1.10)}, \\
S_2^a(\epsilon) = {} & \text{The set of } \epsilon\text{-approximate computational problems for which} \\
& \nexists \text{ a randomised algorithm } \Gamma \text{ computing } \epsilon\text{-approximations in the sense of (1.10)} \\
& \text{with probability } p > 1/2. \\
S_1^b(\epsilon) = {} & \text{The set of } \epsilon\text{-approximate computational problems for which} \\
& \exists \text{ an algorithm } \Gamma \text{ computing } \epsilon\text{-approximations in the sense of (1.11)}, \\
S_2^b(\epsilon) = {} & \text{The set of } \epsilon\text{-approximate computational problems for which} \\
& \nexists \text{ an algorithm } \Gamma \text{ computing } \epsilon\text{-approximations in the sense of (1.11)}.
\end{aligned}
$$

We will – with a slight abuse of notation – omit the explicit mention of the ϵ dependency in the S_j^as and S_j^bs. Theorem 2.1 therefore implies the following statement.

Corollary 2.7 (Theorem 2.1 in a generalised hardness of approximation language)**.** *For any integers $N > m$ and any $\beta_{max} \geq \beta_{min} > 0$, consider any non-zero linear map $A : \mathbb{R}^N \to \mathbb{R}^m$ such that the spectrum $\mathrm{Sp}(AA^*) \subset [\beta_{min}^2, \beta_{max}^2]$. Then, for any rational $\epsilon_1 \in (0, 3/8]$ and any integer $\ell \geq 2$, there exists a collection Ω (as described in §1.1) of training sets and a mapping $\Xi : \Omega \to \mathcal{NN}_{m,N}$, such that we have the following. There is a (S_1^a, S_2^a)-phase transition at $(\epsilon_1, 2\epsilon_1)$ and a $D \in \mathbb{N}$ such that there is a sharp (S_1^b, S_2^b)-phase transition at $2\epsilon_1$ (note that* (1.11) *depends on D).*

3. Historical background, the mathematics of the SCI hieararcy and related work

The results in this paper can be viewed as a continuation of Smale's program [25, 83, 84] on the foundations of computational mathematics. Smale asked several fundamental questions on the foundations of computations, among them his 18th problem: what are the limits of artificial intelligence? – which bares similarities with the famous Turing paper from 1950 [92]. Our work can be viewed as a step towards answering this question. There are several results that are very much related to this paper.

Generalised hardness of approximation: The first discovery of the generalised hardness of approximation phenomenon was done by A. Bastounis et al. in [13], where the phenomenon was documented in a large collection of convex optimisation problems. Following this framework, the results [40] by M. Colbrook, V. Antun et al. demonstrated how generalised hardness of approximation happens in deep learning when NNs can be proven to exist and solve optimisation problems, yet there are phase transition depending on the accuracy and also the amount of data available, see also [36].

The mathematics behind the SCI hierarchy: Generalised hardness of approximation is part of the greater program on the mathematics behind the SCI hierarchy, and this foundations program provides the framework for our results and proofs. The SCI framework was introduced in [56] and continued in the work by J. Ben-Artzi et A First-Order Primal-Dual Algorithm for Convex Problems with Applications to Imagingal. [19,20], in the work by M. Colbrook et al. [38,39] as well as in the work by O. Nevanlinna [17,18,57] and co-authors. See also the work by S. Olver and M. Webb [96] and [41]. The SCI hierarchy is directly related to S. Smale's [83, 85] program on the foundations of computational mathematics and scientific computing that initiated the early work by C. McMullen [69, 70, 87] and P. Doyle & C. McMullen [45] on polynomial root-finding. These are pioneering classification results in the SCI hierarchy. See also classification results in the SCI hierarchy by S. Weinberger [97]. Note that the mathematics behind the SCI hierarchy can be traced back to K. Gödel [53] and A. Turing [91], however, the new techniques developed allow for any model of computation.

Instability in AI: Our results are intimately linked to the instability phenomenon in AI methods – which is widespread [35, 51, 61, 62, 71, 89] – and our results add theoretical understandings to this vast research program. There are particular links to the work by B. Adcock et al. [2] and V. Antun et al. [6]. See also recent developments by D. Higham, I. Tyukin et al. [93, 94].

Existence vs computability of NNs: There is a substantial literature on existence results of NNs [27, 75, 98], see for example the review papers by A. Pinkus [76] and the work by R. DeVore, B. Hanin, and G. Petrova [43] and the references therein . However, as established in [40] by M. Colbrook, V. Antun et al., only a small subset of the NNs than can be proven to exist can be computed by algorithms. However, following the framework of A. Chambolle and T. Pock [33, 34], the results in [40] demonstrate how – under specific assumptions – stable and accurate NNs can be computed. See also the work by P. Niyogi, S. Smale and S. Weinberger [73] on existence results of algorithms for learning.

The PCP theorem: The 2001 Gödel Prize was awarded to S. Arora, U. Feige, S. Goldwasser, C. Lund, L. Lovász, R. Motwani, S. Safra, M. Sudan, and M. Szegedy for their work on the much celebrated PCP theorem [7–9, 50] and its connection to hardness of approximation. The PCP theorem implies that – subject to P$\neq$NP – there are large collections of combinatorial optimisation problems for which there is a sharp phase transition at some $\epsilon_0 > 0$ (where ϵ_0 depends on the problem). Our theorems serve a similar role – however for completely different mathematical reasons – demonstrating how there will be large classes of inverse problems for which one can prove the existence of an optimal NN, yet there will be phase transitions at an approximation threshold $\epsilon_1 > 0$. Moreover, as our theorems state: ϵ_1 can take any value, i.e. for any $\epsilon_1 > 0$ there exists a training problem which has ϵ_1 as an approximation threshold.

Acknowledgements. LG acknowledges support from the Niels Henrik Abel and C. M. Guldbergs memorial fund. ACH acknowledges support from the Simons Foundation Award No. 663281 granted to the Institute of Mathematics of the Polish Academy of Sciences for the years 2021-2023, from a Royal Society University Research Fellowship, and from the Leverhulme Prize 2017.

4. Tools for the proofs – Mathematical preliminaries from the SCI hierarchy

The SCI hierarchy and the mathematical framework that comes with it has been very useful in order to establish the boundaries of computational mathematics ranging from spectral problems, inverse problems, optimisation, AI etc. The SCI hierarchy is based on the concept of a computational problem that we formally

define below. This is described by a function

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

that we want to compute, where Ω is some domain, and $(\mathcal{M}, d)$ is a metric space. The mainstay of the hierarchy are the Δ_k^α classes. The α is related to the model of computation as explained below. In particular, given a collection $\mathcal{C}$ of computational problems, then

(i) Δ_0^α is the set of problems that can be computed in finite time, the SCI $= 0$.
(ii) Δ_1^α is the set of problems that can be computed using one limit (the SCI $= 1$) with control of the error, i.e. $\exists$ a sequence of algorithms $\{\Gamma_n\}$ such that $d(\Gamma_n(\iota), \Xi(\iota)) \leq 2^{-n}$, $\forall \iota \in \Omega$.
(iii) Δ_2^α is the set of problems that can be computed using one limit (the SCI $= 1$) without error control, i.e. $\exists$ a sequence of algorithms $\{\Gamma_n\}$ such that $\lim_{n\to\infty} \Gamma_n(\iota) = \Xi(\iota)$, $\forall \iota \in \Omega$.
(iv) Δ_{m+1}^α, for $m \in \mathbb{N}$, is the set of problems that can be computed by using m limits, (the SCI $\leq m$), i.e. $\exists$ a family of algorithms $\{\Gamma_{n_m,\ldots,n_1}\}$ such that

$$\lim_{n_m\to\infty} \ldots \lim_{n_1\to\infty} \Gamma_{n_m,\ldots,n_1}(\iota) = \Xi(\iota), \, \forall \iota \in \Omega. \tag{4.1}$$

In general, this hierarchy cannot be refined unless there is some extra structure on the metric space $\mathcal{M}$. The hierarchy typically does not collapse, and we have:

$$\Delta_0^\alpha \subsetneq \Delta_1^\alpha \subsetneq \Delta_2^\alpha \subsetneq \ldots \subsetneq \Delta_m^\alpha \subsetneq \ldots. \tag{4.2}$$

However, depending on the collection $\mathcal{C}$ of computational problems, the hierarchy (4.2) may terminate for a finite m, or it may continue for arbitrary large m. We will focus on the lower parts of the hierarchy in (4.2) in this paper, however, for the interested reader we point out that for certain metric spaces $\mathcal{M}$ one can extend (4.2) to the full hierarchy and define the Π_j^α and Σ_j^α classes for $j \in \mathbb{N}$. We then get the following hierarchy:

$$\begin{array}{ccccccccccccc}
\Pi_0^\alpha & & & & \Pi_1^\alpha & & & & \Pi_2^\alpha & & & & \\
\| & & & \nwarrow\!\!\!\!\!\!\subsetneq & & \nwarrow\!\!\!\!\!\!\subsetneq & & \nwarrow\!\!\!\!\!\!\subsetneq & & \nwarrow\!\!\!\!\!\!\subsetneq & & \nwarrow\!\!\!\!\!\!\subsetneq & \\
\Delta_0^\alpha & \subsetneq & \Delta_1^\alpha & \subsetneq & \Sigma_1^\alpha \cup \Pi_1^\alpha & \subsetneq & \Delta_2^\alpha & \subsetneq & \Sigma_2^\alpha \cup \Pi_2^\alpha & \subsetneq & \Delta_3^\alpha & \subsetneq & \cdots \\
\| & & & \swarrow\!\!\!\!\!\!\subsetneq & & \swarrow\!\!\!\!\!\!\subsetneq & & \swarrow\!\!\!\!\!\!\subsetneq & & \swarrow\!\!\!\!\!\!\subsetneq & & \swarrow\!\!\!\!\!\!\subsetneq & \\
\Sigma_0^\alpha & & & & \Sigma_1^\alpha & & & & \Sigma_2^\alpha & & & &
\end{array} \tag{4.3}$$

For details about (4.3) see [17, 18, 56]. In order to study the underdetermined system presented in section 1 formally we present the framework for computational problems established in [13].

Definition 4.1. Let Ω be a set, which we call the domain. Let Λ be a set a complex valued functions $f : \Omega \to \mathbb{C}$ such that for $\iota_1, \iota_2 \in \Omega$, $\iota_1 = \iota_2$ if and only if $f(\iota_1) = f(\iota_2)$ for all $f \in \Lambda$, called an evaluation set. Let $(\mathcal{M}, d)$ be a metric space, and finally let $\Xi : \Omega \to \mathcal{M}$ be a function which we call the problem function. We call the collection $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$ a computational problem.

Remark 4.2 (Multivalued functions)**.** When dealing with multivalued problems one needs a framework that can handle multiple solutions. As the setup above does not allow Ξ to be multi-valued we need some slight changes. We allow Ξ to be multivalued, even though towers of algorithms are not. Hence, the only difference to the standard SCI hierarchy is that the last limit in (4.1) is replaced by

$$\mathrm{dist}_{\mathcal{M}}(\Xi(\iota), \Gamma_{n_m}(\iota)) \longrightarrow 0, \qquad n_m \to \infty,$$

where $\mathrm{dist}_{\mathcal{M}}(\Xi(\iota), \Gamma_{n_m}(\iota)) := \inf_{x\in\Xi(\iota)} d_{\mathcal{M}}(x, \Gamma_{n_m}(\iota))$.

4.1. **Algorithms.** Whenever we aim to use neural networks to solve a computational problem, there is a process of constructing an, in some sense, optimal neural network. This is generally called the training process. We formalize this process in terms of an algorithm.

Definition 4.3. Given a computational problem $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$, a general algorithm is a mapping $\Gamma : \Omega \to \mathcal{M}$ such that for each $\iota \in \Omega$

(i) There exists a finite subset of evaluations $\Lambda_\Gamma(\iota) \subseteq \Lambda$.
(ii) The action of Γ on ι only depends on $\{f(\iota)\}_{f\in\Lambda_\Gamma(\iota)}$.
(iii) For every $\iota' \in \Omega$ such that $f(\iota') = f(\iota)$ for every $f \in \Lambda_\Gamma(\iota)$, it holds that $\Lambda_\Gamma(\iota') = \Lambda_\Gamma(\iota)$.

The statements in our theorems are well-defined up to the definition of an algorithm/randomised algorithm. There are a myriad of different types of machines that can be used to model an algorithm: the Turing machine [91] (and its cousins including the Markov model [81]), the BSS machine [26], the von Neumann architecture [95], the real RAM [77], etc. as well as their randomised versions. Indeed, since randomised methods, such as for example randomised gradient decent, are often used when training neural networks, we need to consider randomised algorithms in order to achieve full generality. However, these models are not equivalent when it comes to computability. Thus, to create universal impossibility results we use general algorithms from Definition (4.3) and a randomised general algorithms from Definition 4.4 that encompass any reasonable definition of a computational model in the way that they are more powerful than any standard machine, therefore making the impossibility results stronger. Formally we define a randomised algorithm as follows:

Definition 4.4 (Randomised General Algorithm [13])**.** Given a computational problem $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$, where $\Lambda = \{f_k \,|\, k \in \mathbb{N},\, k \leq |\Lambda|\}$, a *randomised general algorithm* (RGA) is a collection X of general algorithms $\Gamma : \Omega \to \mathcal{M} \cup \{\text{NH}\}$, a sigma-algebra $\mathcal{F}$ on X, and a family of probability measures $\{\mathbb{P}_\iota\}_{\iota\in\Omega}$ on $\mathcal{F}$ such that the following conditions hold:

(i) For each $\iota \in \Omega$, the mapping $\Gamma^{\text{ran}}_\iota : (X, \mathcal{F}) \to (\mathcal{M} \cup \{\text{NH}\}, \mathcal{B})$ defined by $\Gamma^{\text{ran}}_\iota(\Gamma) = \Gamma(\iota)$ is a random variable, where $\mathcal{B}$ is the Borel sigma-algebra on $\mathcal{M} \cup \{\text{NH}\}$.
(ii) For each $n \in \mathbb{N}$ and $\iota \in \Omega$, we have $\{\Gamma \in X \,|\, T_\Gamma(\iota) \leq n\} \in \mathcal{F}$.
(iii) For all $\iota_1, \iota_2 \in \Omega$ and $E \in \mathcal{F}$ so that, for every $\Gamma \in E$ and every $f \in \Lambda_\Gamma(\iota_1)$, we have $f(\iota_1) = f(\iota_2)$, it holds that $\mathbb{P}_{\iota_1}(E) = \mathbb{P}_{\iota_2}(E)$.

It is not immediately clear whether condition (ii) for a given RGA $(X, \mathcal{F}, \{\mathbb{P}_\iota\}_{\iota\in\Omega})$ holds independently of the choice of the enumeration of Λ. This is indeed the case and was established in [13].

We should justify this model before we move on: (i) and (ii) are measure theoretic notions that ensure that natural sets that one might construct for randomized algorithms (such as the minimum runtime) are measurable sets, these assumptions are also satisfied by any classical probabilistic model such as randomized Turing machines or randomized BSS machines. The third point ensures the consistency of the model, in the sense that if the model reads the same information for two inputs then the probability distribution of the outputs should be the same.

4.2. **Inexact input and breakdown epsilons.** We now introduce the notion of inexact input for general computational problems.

Definition 4.5 (Δ_1-information)**.** Suppose we are given a computational problem $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$ and that $\Lambda = \{f_i\}_{i\in I}$ where I is some index that can be finite or infinite. As previously mentioned, in many cases we are forced to deal with inexact inputs. In this case we can not access $f_i(\iota)$, but rather an approximation $f_{i,n}(\iota)$ where $f_{i,n}(\iota) \to f_i(\iota)$ as $n \to \infty$. Throughout this paper we will assume that this can be done with error control. More precisely, we assume that we have access to $f_{i,n} : \Omega \to \mathbb{D}_n + i\mathbb{D}_n$, where

$\mathbb{D}_n = \{k2^{-n} \mid k \in \mathbb{Z}\}$, such that

$$\|\{f_{i,n}(\iota)\}_{i\in I} - \{f_i(\iota)\}_{i\in I}\|_\infty \leq 2^{-n} \quad \forall\, \iota \in \Omega. \tag{4.4}$$

If we for each $n \in \mathbb{N}$ have that there exists an $f_{i,n} : \Omega \to \mathbb{D}_n + i\mathbb{D}_n$ for $i \in I$ such that eq. (4.4) holds for $\{f_{i,n}\}_{i\in I}$, we say that the set $\hat{\Lambda} = \{f_{i,n} \mid i \in I,\ n \in \mathbb{N}\}$ provides Δ_1-*information* for $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$. Moreover, we denote the family of all such $\hat{\Lambda}$ by $\mathcal{L}^1(\Lambda)$.

We want our algorithms to be able to deal with inexact input, in other words we want to have algorithms that can handle the computational problems $\{\Xi, \Omega, \mathcal{M}, \hat{\Lambda}\}$ for all possible choices of $\hat{\Lambda} \in \mathcal{L}^1(\Lambda)$. In order to formalize this we introduce computational problems with Δ_1-information:

Definition 4.6. Given $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$ with $\Lambda = \{f_i\}_{i\in I}$ the corresponding computational problem with Δ_1-information is defined as

$$\{\Xi, \Omega, \mathcal{M}, \Lambda\}^{\Delta_1} = \{\tilde{\Xi}, \tilde{\Omega}, \mathcal{M}, \tilde{\Lambda}\},$$

where

$$\tilde{\Omega} = \{\tilde{\iota} = \{f(\iota)_{i,n}\}_{i\in I, n\in\mathbb{N}} \mid \iota \in \Omega,\ \{f_{i,n}\}_{i\in I} \text{ satisfying eq. (4.4) for all } n \in \mathbb{N}\}, \tag{4.5}$$

$\tilde{\Xi}(\tilde{\iota}) = \Xi(\iota)$ and $\tilde{\Lambda} = \{\tilde{f}_{i,n}\}_{i\in I, n\in\mathbb{N}}$ where $\tilde{f}_{i,n}(\tilde{\iota}) = f_{i,n}(\iota)$. Given an $\tilde{\iota} \in \tilde{\Omega}$, there is a unique $\iota \in \Omega$ for which $\tilde{\iota} = \{f_{i,n}(\iota)\}_{i\in I, n\in\mathbb{N}}$. We say that this $\iota \in \Omega$ corresponds to $\tilde{\iota} \in \tilde{\Omega}$.

We interpret the computational problem $\{\Xi, \Omega, \mathcal{M}, \Lambda\}^{\Delta_1}$ as follows: The domain $\tilde{\Omega}$ is the collection of all the sequences approximating the inputs in Ω, and we say that an algorithm Γ works on inexact input if it works for all $\tilde{\iota} \in \tilde{\Omega}$, that is, for any sequence approximating ι. For later use, we need to specify how the input $\tilde{\iota} \in \tilde{\Omega}$ is passed to a Turing machine T as an input. For this we need to assume that the index set I for Λ is countable. With this assumption in place $\tilde{\iota}$ is represented by an oracle tape that T can access, where on input $(i, n) \in I \times \mathbb{N}$ the oracle returns the unique finite binary string representing $\tilde{f}_{i,n}(\tilde{\iota})$.

Remark 4.7 (Turing vs Markov)**.** It is possible to consider a restriction of Δ_1 information wherein for each j, $\{f_{j,n}\}_{n\in\mathbb{N}}$ forms a computable sequence. This restriction (known as the Markov model [81]) strengthens the negative results and weakens the positive results. See Remark 4.15 for further details.

We wish to investigate the constructibility of optimal neural networks in the theory of underdetermined systems. That is, investigate the existence of training algorithms that obtain a neural network that approximates the inverse of the linear mapping A well. To do this we need the notion of breakdown epsilons.

Definition 4.8 (Strong and probabilistic strong breakdown epsilons)**.** Given a computational problem $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$, the strong breakdown-epsilon $\epsilon^{\mathrm{s}}_{\mathrm{B}}$ is given by

$$\epsilon^{\mathrm{s}}_{\mathrm{B}} = \sup\{\epsilon > 0 \mid \forall \text{ algorithms } \Gamma, \exists\, \iota \in \Omega \text{ such that } d_{\mathcal{M}}(\Gamma(\iota), \Xi(\iota)) > \epsilon\}$$

and the strong probabilistic breakdown-epsilon $\epsilon^{\mathrm{s}}_{\mathbb{P}\mathrm{B}} : [0, 1) \to \mathbb{R}$ is given by

$$\epsilon^{\mathrm{s}}_{\mathbb{P}\mathrm{B}}(p) = \sup\{\epsilon > 0 \mid \forall\, \Gamma^{ran} \in \mathrm{RGA}\ \exists\, \iota \in \Omega \text{ such that } \mathbb{P}_\iota(d_{\mathcal{M}}(\Gamma^{ran}_\iota, \Xi(\iota)) > \epsilon) > p\}.$$

As established in [13], impossibility results for randomised algorithms can differ if one considers only those algorithms that halt on every input, leading to the following two definitions.

Definition 4.9 (Halting randomised general algorithms)**.** A randomised general algorithm Γ^{ran} for a computational problem $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$ is called a *halting randomised general algorithm* (hRGA) if $\mathbb{P}_\iota(\Gamma^{\mathrm{ran}}_\iota = \mathrm{NH}) = 0$, for all $\iota \in \Omega$. We denote the class of all halting randomised general algorithms by hRGA.

Definition 4.10 (Probabilistic strong halting breakdown epsilon)**.** Given the computational problem $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$, where $\Lambda = \{f_k \,|\, k \in \mathbb{N},\, k \leq |\Lambda|\}$, we define the *halting probabilistic strong Breakdown-epsilon* $\epsilon^{\mathrm{s}}_{\mathbb{P}hB} : [0,1) \to \mathbb{R}$ according to

$$\epsilon^{\mathrm{s}}_{\mathbb{P}hB}(\mathrm{p}) = \sup\{\epsilon \geq 0,\, | \,\forall\, \Gamma^{\mathrm{ran}} \in \mathrm{hRGA}\ \exists\, \iota \in \Omega \text{ such that } \mathbb{P}_\iota(\mathrm{dist}_{\mathcal{M}}(\Gamma^{\mathrm{ran}}_\iota, \Xi(\iota)) > \epsilon) > \mathrm{p}\},$$

where $\Gamma^{\mathrm{ran}}_\iota$ is defined in (i) in Definition 4.4.

Throughout this paper we wish to prove the strongest possible lower bounds of computability. Thus we allow our general algorithms to preform arbitrary general operations and we prove all our lower bounds for such algorithms. On the other hand, we also wish to have the strongest possible positive results, thus when we construct algorithms, these algorithms will always be recursive (That is, possible to simulate with a Turing machine). We denote the Δ_0 and Δ_1 class of computational problems that are computable by recursive algorithms by Δ^A_0 and Δ^A_1, where A stands for arithmetic, since we only allow arithmetic operations.

4.3. **Formal description of problem.** In order to prove the negative claims asserted in the previous section we formulate our problem as a computational problem.

We start by formally defining what we mean by a neural network. Deep learning is a rapidly developing field, where new constructions and architectures of neural networks are constantly proposed. Our aim is to capture as many of these constructions as possible with our results, thus we propose the following general formal definition of a neural network:

Definition 4.11. A neural network $\mathbf{N} : \mathbb{C}^m \to \mathbb{C}^N$ is a composition of maps on the form

$$\phi(y) = V_L(\rho_{L-1}(\dots \rho_1(V_1(y)) \dots), y),$$

where each $V_j : \mathbb{C}^{N_{j-1}} \to \mathbb{C}^{N_j}$ is an affine map on the form $V_j(x) = W_j x + b_j$ for $j = 1, \dots, L-1$ and $V_L : \mathbb{C}^{N_{L-1}} \to \mathbb{C}^N$ is an affine map on the form $V_L(x, y) = W_L x + b_L(y)$, where $b_L(y)$ is an affine function on the input y given by $b_L(y) = Ry + c_L \in \mathbb{C}^N$. Furthermore, there exists an index set $I_j \subseteq \{1, \dots N_j\}$ for each $j = 1, \dots, L$ such that the activation function $\rho_j : \mathbb{C}^{N_{j-1}} \to \mathbb{C}^{N_j}$ is given by

$$\rho_j(x)_k = \begin{cases} f_j(x_k), & \text{if } k \in I_j \\ x_k, & \text{otherwise,} \end{cases}$$

where $f_j : \mathbb{C} \to \mathbb{C}$ is a possibly non-linear function.

Remark 4.12. The affine dependence of the last bias term $b_L(y)$ allows skip connections from the input to the last level. By composition this allows us to obtain standard architectures such as residual networks [52, 60, 82] with an arbitrary number of layers between the skip connections.

The domain Ω is the set of objects that gives rise to our computational problem. In our setting these objects are the training sets. Since the error of an algorithm is mostly interesting relative to the size and the bounds of the training set, we will consider bounded training sets of fixed size throughout this paper. More formally let

$$\Omega \subseteq T^1_\ell = \{\mathcal{T} \subseteq \mathbb{R}^N \times \mathbb{R}^m \,:\, \|x\|_{\ell_2}, \|y\|_{\ell_2} \leq 1 \text{ for all } (x,y) \in \mathcal{T}, \text{ and } |\mathcal{T}| = \ell\,\} \tag{4.6}$$

be the domain. The set of measurements Λ is the collection of functions that provide us with the information we are allowed to read as an input to an algorithm. We define the measurements as follows: Given a training set $\mathcal{T} \in \Omega$ we order $\mathcal{T}$ according to the lexicographic ordering and define $f^k_{x,i}$ to be given by $f^k_{x,i}(\mathcal{T}) = \pi_1(\mathcal{T})_{k,i}$, where $\pi_1(\mathcal{T}) := \{x \mid (x,y) \in \mathbb{R}^N \times \mathbb{R}^m,\, (x,y) \in \mathcal{T}\}$ and the indexes k and i denote the i'th coordinate of the of the k'th element according to the lexicographic ordering. We define $f^k_{y,j}$ in the

same way to measure the y-coordinates, more precisely $f^k_{y,i}(\mathcal{T}) = \pi_2(\mathcal{T})_{k,i}$, where $\pi_2(\mathcal{T}) := \{y \mid (x,y) \in \mathbb{R}^N \times \mathbb{R}^m,\ (x,y) \in \mathcal{T}\}$. In summary, we define Λ to be the collection

$$\Lambda = \{f^k_{y,j}, f^k_{x,i} : \Omega \to \mathbb{R} \mid i = 1,\dots,N,\ j = 1,\dots,m,\ k = 1,\dots,\ell\}. \tag{4.7}$$

Next, we define a precise notation for an inexact representation of the elements in the domain Ω. Let $\hat{\Lambda} = \{f_n \mid f \in \Lambda,\ n \in \mathbb{N}\}$ be a set that provides Δ_1-information for Ω as defined in Definition 4.5. Then, for an arbitrary $\mathcal{T} \in \Omega$, let $(x,y) \in \mathcal{T}$ be the k'th element in $\mathcal{T}$ according to the lexicographic ordering. We then define the corresponding inexact representations $\tilde{x}$ of x and $\tilde{y}$ of y to be the sequences

$$\tilde{x} = \{\{f^k_{x,i,n}(\mathcal{T})\}_{i=1}^{i=N}\}_{n\in\mathbb{N}} \quad \text{and} \quad \tilde{y} = \{\{f^k_{y,j,n}(\mathcal{T})\}_{j=1}^{j=m}\}_{n\in\mathbb{N}}. \tag{4.8}$$

At last, given a matrix $A \in \mathbb{R}^{N\times m}$ and a collection $\{(\mathcal{T}, \mathcal{M}_1) \mid \mathcal{T} \in \Omega\}$ we define the problem function Ξ in the following way

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

where $\mathcal{NN}_{m,N}$ is set of neural networks of real input dimension m and real output dimension N that are bounded on $\bigcup_{\mathcal{F}} \mathcal{M}_2 = \bigcup_{(A,\mathcal{M}_1)\in\mathcal{F}} A(\mathcal{M}_1)$. We consider $\mathcal{NN}_{m,N}$ as a metric space equipped with the following metric, for $\mathbf{N_1}$,$\mathbf{N_2} \in \mathcal{NN}_{m,N}$,

$$d(\mathbf{N_1}, \mathbf{N_2}) = \sup_{z\in\bigcup_{\mathcal{F}}\mathcal{M}_2} \|\mathbf{N_1}(z) - \mathbf{N_2}(z)\|_{\ell_2}. \tag{4.9}$$

4.4. **An important preliminary result.** We use the following important proposition to prove the non-existence of algorithms that construct optimal neural networks.

Proposition 4.13 (Proposition 10.5 in [13])**.** *Let $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$ be a computational problem with $\Lambda = \{f_k \mid k \in \mathbb{N}, k \le |\Lambda|\}$ countable, and let $\{\iota^1_n\}_{n=1}^\infty$, $\{\iota^2_n\}_{n=1}^\infty$ be sequences in Ω. Consider the following conditions:*

(a) There are sets $S^1, S^2 \subset \mathcal{M}$ and $\kappa > 0$ such that $\inf_{x_1\in S^1, x_2 \in S^2} d_{\mathcal{M}}(x_1,x_2) \ge \kappa$ and $\Xi(\iota^j_n) \subset S^j$ for $j = 1,2$.
(b) For every $k \le |\Lambda|$ there is a $c_k \in \mathbb{C}$ such that $|f_k(\iota^j_n) - c_k| \le 1/4^n$, for all $j = 1,2$ and $n \in \mathbb{N}$.
(c) There is an $\iota^0 \in \Omega$ such that for every $k \le |\Lambda|$ we have that (b) is satisfied with $c_k = f_k(\iota^0)$.
(d) There is an $\iota^0 \in \Omega$ for which condition (c) holds and additionally $\iota^2_n = \iota^0$, for all $n \in \mathbb{N}$.

Depending on which of the conditions (a) – (d) are fulfilled, there exists a $\hat{\Lambda} \in \mathcal{L}^1(\Lambda)$ such that some of the following claims about the computational problem $\{\Xi, \Omega, \mathcal{M}, \hat{\Lambda}\}$ hold:

(i) $\epsilon^{\mathrm{w}}_{\mathrm{B}} \ge \epsilon^{\mathrm{w}}_{\mathbb{P}\mathrm{B}}(\mathrm{p}) \ge \kappa/2$ for $\mathrm{p} \in [0, 1/2)$,
(ii) $\epsilon^{\mathrm{s}}_{\mathrm{B}} \ge \epsilon^{\mathrm{s}}_{\mathbb{P}h\mathrm{B}}(\mathrm{p}) \ge \kappa/2$ for $\mathrm{p} \in [0, 1/2)$ and $\epsilon^{\mathrm{s}}_{\mathbb{P}\mathrm{B}}(\mathrm{p}) \ge \kappa/2$ for $\mathrm{p} \in [0, 1/3)$,
(iii) $\epsilon^{\mathrm{s}}_{\mathbb{P}\mathrm{B}}(\mathrm{p}) \ge \kappa/2$ for $\mathrm{p} \in [0, 1/2)$.

Concretely, if (a) and (b) are fulfilled, then (i) holds, if (a) – (c) are fulfilled, then (i) and (ii) hold, and finally, if (a) – (d) are fulfilled, then (i) – (iii) hold.

Remark 4.14. Proposition 4.13 is a result about how inexact input can ruin the performance of any algorithm given that the domain has some unfortunate properties. In a way the above statement is very intuitive and should be read as follows: If two things that are arbitrarily close in the domain gets mapped far apart by the problem function Ξ, then any algorithm that works with inexact input (that is, works with the computational problem $\{\Xi, \Omega, \mathcal{M}, \Lambda\}^{\Delta_1}$) will break down. In fact, the result even states something slightly stronger: There exists one specific $\hat{\Lambda}$ that gives Δ_1-information for $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$ such that all algorithms Γ break down for the corresponding computational problem $\{\hat{\Xi}, \hat{\Omega}, \mathcal{M}, \hat{\Lambda}\}$, where

$$\hat{\Omega} = \{\hat{\iota} = \{f(\iota)_{i,n}\}_{i\in I, n\in\mathbb{N}} \mid \iota \in \Omega \text{ and } f_{i,n} \in \hat{\Lambda}\},$$

and $\hat{\Xi}(\hat{\iota}) = \Xi(\iota)$. As before, we notice that there is a canonical bijection between Ω and $\hat{\Omega}$ and by identifying $\Omega \simeq \hat{\Omega}$ the mapping $\hat{\Xi}$ becomes the same as Ξ. Because of this identification we will often write $\{\Xi, \Omega, \mathcal{M}, \hat{\Lambda}\}$ in place of $\{\hat{\Xi}, \hat{\Omega}, \mathcal{M}, \hat{\Lambda}\}$.

Remark 4.15 (Computability of Δ_1 information)**.** Note that if $(c_k)_{k\leq|\Lambda|}$ are computable numbers in the sense of Turing-computability, then $\hat{\Lambda}$ can be chosen so that $\hat{\Lambda} = \{f_{k,n} \,|\, k \leq |\Lambda| \text{ and } n \in \mathbb{N}\}$ and so that, for any $\iota \in \Omega$ and each $f_k \in \Lambda$ there exists a Turing machine, such that, given input $n \in \mathbb{N}$, outputs $f_{k,n}(\iota)$, see [13] for details. Thus, the non-computability result presented in Proposition 4.13 can easily be seen to apply to the Markov model.

5. Formal statements and proofs of the main results

Before we embark on the proofs of the theorems we will introduce some basic notation and discuss some vocabulary that will be used in the proofs. First, we introduce some standard projections. Indeed, for $\mathcal{T} \in \mathbb{R}^N \times \mathbb{R}^m$, $\pi_1(\mathcal{T})$ and $\pi_2(\mathcal{T})$ is defined as follows:

$$\begin{aligned} \pi_1(\mathcal{T}) &:= \{x \,|\, (x,y) \in \mathbb{R}^N \times \mathbb{R}^m,\ (x,y) \in \mathcal{T}\}, \\ \pi_2(\mathcal{T}) &:= \{y \,|\, (x,y) \in \mathbb{R}^N \times \mathbb{R}^m,\ (x,y) \in \mathcal{T}\}. \end{aligned} \tag{5.1}$$

Second, recall Definition 4.5 and Definition 4.6 where we discuss the concepts of Δ_1-information and computational problems with Δ_1-information. In particular, we have the original domain Ω and the corresponding domain $\tilde{\Omega}$ that contains all sequences of approximations to the elements in Ω. We will typically use the notation $\mathcal{T} \in \Omega$ and $\tilde{\mathcal{T}} \in \tilde{\Omega}$. At last, we use $\|\cdot\|_{op}$ to denote the standard operator norm, more precisely for an $M : \mathbb{R}^N \to \mathbb{R}^m$ we have that

$$\|M\|_{op} = \sup\{\|Mx\|_{\ell_2} \,:\, \|x\|_{\ell_2} \leq 1\}. \tag{5.2}$$

5.1. Useful propositions and lemmas.

We start by presenting a driving proposition, from which parts of the first four of our main results will follow as corollaries.

Proposition 5.1. *Let $A : \mathbb{R}^N \to \mathbb{R}^m$ be a non-zero matrix with non-trivial kernel. Then for any $\kappa \leq 3/8$ and $\ell \geq 2$, there exists uncountably many domains $\Omega \subseteq T^1_\ell$ of training sets, which give rise to uncountably many computational problems $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$ as described in §4.3, such that for each domain Ω there is a $D \in \mathbb{N}$ such that Ξ_D in (1.7) satisfies $\Xi_D(\iota) \neq \emptyset$ for all $\iota \in \Omega$, and there exist sequences $\{\iota^1_n\}_{n\in\mathbb{N}}, \{\iota^2_n\}_{n\in\mathbb{N}} \subset \Omega$ such that the conditions $a) - c)$ in Proposition 4.13 are satisfied for $\{\iota^1_n\}_{n\in\mathbb{N}}$ and $\{\iota^2_n\}_{n\in\mathbb{N}}$.*

Proof. Since A has a non-trivial kernel, we can pick an $v \in \ker(A)$ such that $\|v\|_{\ell_2} = 2\kappa$. Since A is non-zero, we can pick a unit vector $e \in \ker(A)^\perp$. Let $\mathcal{T}_b \in T^1_{\ell-2}$ be a finite training set with distinct non-zero x-coordinates such that $\mathcal{T}_b \subset \ker(A)^\perp \times A(\ker(A)^\perp)$ and $(B^+_{1/4}(v), A(B^+_{1/4}(v))) \notin \mathcal{T}_b$ where

$$B^{\ell,+}_{1/4}(v) = \{v' \in \mathbb{R}^N \,|\, v \leq_\ell v' \leq_\ell v + \frac{1}{4}e\},$$

is the potentially empty set where $\leq_\ell$ denotes the lexicographic ordering. We now define $\{\iota^1_n\}_{n\in\mathbb{N}}$ and $\{\iota^2_n\}_{n\in\mathbb{N}}$ as follows. Let

$$\iota^1_n = \mathcal{T}_b \cup \{(0,0), (v+\frac{\theta}{4^n}e, A(v+\frac{\theta}{4^n}e))\} \quad \text{and} \quad \iota^2_n = \mathcal{T}_b \cup \{(0,0),(v,0)\}, \quad n \in \mathbb{N}, \tag{5.3}$$

where $\theta = \|A\|^{-1}_{op}$ if $\|A\|_{op} > 1$ and $\theta = 1$ otherwise. Then it is not hard to see that $\iota^j_n \subseteq T^1_\ell$ for all $n \in \mathbb{N}$ and $j = 1, 2$. We define the domain $\Omega \subseteq T^1_\ell$ to be $\Omega = \{\iota^1_n\}_{n\in N} \cup \{\iota^2_n\}_{n\in\mathbb{N}}$, and we define the corresponding inverse problems $(A, \mathcal{M}^j_{1,n})$ as follows: $\mathcal{M}^j_{1,n} = \pi_1(\iota^j_n)$ with $j = 1, 2$, where $\pi_1(\iota^j_n)$ is defined in (5.1). We show that $\Xi(\iota^j_n) \neq \emptyset$ for all $n \in \mathbb{N}$ and $j = 1, 2$ below. By the arbitrary choice of the elements in $\mathcal{T}_b$ it is clear that there exist uncountably many choices for the domain Ω, and by our choice of $\mathcal{T}_b \in \ker(A)^\perp \times A(\ker(A)^\perp)$ and the fact that $e \in \ker(A)^\perp$ and A is injective on $\ker(A)^\perp$, it is clear that for all $n \in \mathbb{N}$ the elements in ι^1_n are distinct in both the x and y-coordinates. Thus, we can

achieve exact interpolation of every point by a smooth neural network according to [76, Theorem 5.1]. In particular, for each $n \in \mathbb{N}$, there is a neural network $\mathbf{N}_n : \mathbb{R}^m \to \mathbb{R}^N$ such that for each pair $(\xi_n, \eta_n) \in \iota_n^1$ we have $\mathbf{N}_n(\eta_n) = \xi_n$, and it is clear from Definition 1.1 that this is an optimal map for the inverse problem $(A, \mathcal{M}_{1,n}^1)$. By the same argument we can find a neural network $\mathbf{N} : \mathbb{R}^m \to \mathbb{R}^N$ such that for any $(\xi, \eta) \in \mathcal{T}_b$ we have $\mathbf{N}(\eta) = \xi$ and $\mathbf{N}(0) = 1/2v$. We claim that $\mathbf{N}$ is optimal for $(A, \mathcal{M}_{1,n}^2)$. Indeed, note that

$$\inf_{\varphi \colon \mathcal{M}_2 \rightrightarrows \mathbb{R}^N} \sup_{x \in \mathcal{M}_{1,n}^2} d_1^H(\varphi(Ax), x) \geq \inf_{\varphi \colon \mathcal{M}_2 \rightrightarrows \mathbb{R}^N} \max_{x=0, x=v} d_1^H(\varphi(Ax), x) = 1/2\|v\|. \tag{5.4}$$

However, by the definition of $\mathbf{N}$ we have that

$$\sup_{x \in \mathcal{M}_{1,n}^2} d_1^H(\mathbf{N}(Ax), x) = \max_{x=0, x=v} d_1^H(\mathbf{N}(Ax), x) = 1/2\|v\|, \tag{5.5}$$

proving our claim. This means that $\Xi(\iota_n^j) \neq \emptyset$ for $j = 1, 2$. Moreover, It is clear from the choice of Ω that there is a $D \in \mathbb{N}$ such that Ξ_D in (1.7) satisfies $\Xi_D(\iota) \neq \emptyset$ for all $\iota \in \Omega$. Thus the computational problem $\{\Xi_D, \Omega, \mathcal{M}, \Lambda\}$ is now well defined. Next we show that the sequences$\{\iota_n^1\}_{n \in \mathbb{N}}$ and $\{\iota_n^2\}_{n \in \mathbb{N}}$ satisfy points $a)$ and $b)$ in proposition 4.13, as well as condition $c)$:

$a)$. We define

$$S_1 = \bigcup_{n \in \mathbb{N}} \Xi(\iota_n^1) \quad \text{and} \quad S_2 = \bigcup_{n \in \mathbb{N}} \Xi(\iota_n^2).$$

Using the metric defined in (4.9) we get that

$$\begin{aligned}
\inf_{\mathbf{N_1} \in S_1, \mathbf{N_2} \in S_2} d(\mathbf{N_1}, \mathbf{N_2}) &= \inf_{\mathbf{N_1} \in S_1, \mathbf{N_2} \in S_2} \sup_{z \in \bigcup_{\mathcal{F}} \mathcal{M}_2} \|\mathbf{N_1}(z) - \mathbf{N_2}(z)\|_{\ell_2} \\
&\geq \inf_{\mathbf{N_1} \in S_1, \mathbf{N_2} \in S_2} \|\mathbf{N_1}(0) - \mathbf{N_2}(0)\|_{\ell_2} \text{ (optimality of } \mathbf{N_j} \text{ and (5.4), (5.5) give)} \\
&= \|0 - \frac{1}{2}v\|_{\ell_2} = \frac{1}{2}\|v\|_{\ell_2} = \frac{1}{2}2\kappa = \kappa.
\end{aligned}$$

Thus part $a)$ in proposition 4.13 is satisfied.

$b)$. For each $n \in \mathbb{N}$ the elements in $\mathcal{T}_b$ and $(0, 0)$ coincide in ι_n^1 and ι_n^2. Moreover we have put constraints on $\mathcal{T}_b$ such that the elements $(v + \frac{\theta}{4^n}e, A(v + \frac{\theta}{4^n}e)) \in \iota_n^1$ and $(v, 0) \in \iota_n^2$ land on the same index k_v when we list the elements according to the lexicographic ordering for each $n \in \mathbb{N}$. Thus we only need to show that the criteria in part $b)$ holds for each $f_{x,i}^{k_v}(\iota_n^1)$ and $f_{y,j}^{k_v}(\iota_n^1)$ for $i = 1, \ldots, N$, $j = 1, \ldots, m$ and $n \in \mathbb{N}$. We start by $f_{y,j}^{k_v}$. For each j we choose $c_{y,j}^{k_v} = 0$, then we get $|f_{y,j}^{k_v}(\iota_n^1) - c_{y,j}^{k_v}| = |A(\frac{\theta}{4^n}e)_j - 0| = \frac{\theta}{4^n}|A(e)_j| \leq \frac{1}{4^n}$, and $|f_{y,j}^{k_v}(\iota_n^2) - c_{y,j}^{k_v}| = |0 - 0| = 0$ for each $n \in \mathbb{N}$. Thus the criteria in part $b)$ holds for the $f_{y,j}^{k_v}$'s. Next, for each $f_{x,i}^{k_v}$ we choose $c_{x,i}^{k_v} = v_i$, then we get $|f_{x,i}^{k_v}(\iota_n^1) - c_{x,i}^{k_v}| = |v_i + \frac{\theta}{4^n}e_i - v_i| \leq \frac{\theta}{4^n}\|e\|_{\ell_2} \leq \frac{\theta}{4^n} \leq \frac{1}{4^n}$, and $|f_{x,i}^{k_v}(\iota_n^2) - c_{x,i}^{k_v}| = |v_i - v_i| = 0$ for each $n \in \mathbb{N}$. Thus we can conclude that part $b)$ in proposition 4.13 holds for $\{\iota_n^1\}_{n \in \mathbb{N}}$ and $\{\iota_n^2\}_{n \in \mathbb{N}}$.

$c)$. Finally, we observe that if we pick $\iota_0 = \iota_n^2 = \mathcal{T}_b \cup \{(0, 0), (0, v)\}$ then part $b)$ is satisfied with $c_k = f_k(\iota_0)$ for each $k \leq |\Lambda|$. Thus, point $c)$ is also satisfied, this concludes the proof. □

In order to prove the existence of algorithms in part (ii) and (iii) of Theorem 2.4 we need to obtain a recursively constructable neural network. For this we will use the following technical lemma.

Lemma 5.2. *Let $\ell \in \mathbb{N}$ and let $x_1, \ldots, x_\ell \in \mathbb{R}^N$ and $y_1, \ldots, y_\ell \in \mathbb{R}^m$ be such that $x_i \neq x_j$ and $y_i \neq y_j$ for $j \neq i$. Further, let $\phi : \mathbb{R} \to \mathbb{R}$ be given by $\phi(x) = \frac{1}{x^2+1}$, and let $\Phi : \mathbb{R}^N \to \mathbb{R}^\ell$ be the function given by*

$$\Phi(y) = [\phi(\|y - y_1\|_{\ell_2}), \ldots, \phi(\|y - y_\ell\|_{\ell_2})]^T. \tag{5.6}$$

At last, let the matrices $X \in \mathbb{R}^{N \times \ell}$ and $R \in \mathbb{R}^{\ell \times \ell}$ be given by

$$X = (x_1, \ldots, x_\ell) \tag{5.7}$$

and

$$R = \begin{pmatrix} \phi(\|y_1 - y_1\|_{\ell_2}) & \dots & \phi(\|y_\ell - y_1\|_{\ell_2}) \\ \vdots & & \vdots \\ \phi(\|y_1 - y_\ell\|_{\ell_2}) & \dots & \phi(\|y_\ell - y_\ell\|_{\ell_2}) \end{pmatrix} = \begin{pmatrix} 1 & \dots & \phi(\|y_\ell - y_1\|_{\ell_2}) \\ \vdots & & \vdots \\ \phi(\|y_1 - y_\ell\|_{\ell_2}) & \dots & 1 \end{pmatrix}. \quad (5.8)$$

Then R is invertible, and the function $s : \mathbb{R}^m \to \mathbb{R}^m$ given by $s(y) = X \cdot R^{-1} \cdot \Phi(y)$ satisfies $s(y_i) = X \cdot R^{-1} \cdot \Phi(y_i) = x_i$ for all $i = 1, \dots, \ell$. Moreover, $s \in \mathcal{NN}_{m,N}$, that is, s can be represented as a neural network.

Proof. by [47, Section 4.5] R is a symmetric positive definite nonsingular matrix when the y_i's are unique, which they are by assumption. The fact that s interpolates all the pairs (x_i, y_i) for $i = 1, \dots, \ell$ now follows by observing that the solution for the system of inequalities

$$x_i = \begin{pmatrix} x_i(1) \\ \vdots \\ x_i(N) \end{pmatrix} = \begin{pmatrix} \sum_{j=1}^{\ell} \lambda_j^1 \phi(\|y_i - y_j\|_{\ell_2}^2) \\ \vdots \\ \sum_{j=1}^{\ell} \lambda_j^N \phi(\|y_i - y_j\|_{\ell_2}^2) \end{pmatrix} = \begin{pmatrix} \lambda_1^1 & \dots & \lambda_\ell^1 \\ \vdots & & \vdots \\ \lambda_1^N & \dots & \lambda_\ell^N \end{pmatrix} \begin{pmatrix} \phi(\|y_i - y_1\|_{\ell_2}^2) \\ \vdots \\ \phi(\|y_i - y_\ell\|_{\ell_2}^2) \end{pmatrix} =: \lambda \cdot \Phi(y_i).$$

is given by $\lambda = XR^{-1}$ (this can be found by expanding the above inequality to $X = \lambda R$ and using the fact that R is invertible). Thus, $x_i = \lambda \cdot \Phi(y_i) = X \cdot R^{-1} \cdot \Phi(y_i) = s(y_i)$, which proves the interpolation claim. At last, it remains to argue that s can be written as a neural network. Indeed, we observe that

$$s(y) = \mathbf{N}_{\mathcal{T}}(y) := V_3 \rho_2 V_2 \rho_1 V_1(y), \quad (5.9)$$

with the affine maps and non-linear functions defined as follows:

$$\begin{aligned} V_1 &: \ W_1 = [1, \dots, 1]_\ell^T \otimes I_m, \quad b_1 = -\sum_{j=1}^{\ell} e_j^\ell \otimes y_j, \quad \rho_1(t) = t^2 \\ V_2 &: \ W_2 = I_\ell \otimes [1, \dots, 1]_m, \quad b_2 = 0, \quad \rho_2(t) = 1/(t+1), \quad t \in \mathbb{R} \\ V_3 &: \ W_3 = XR^{-1}, \quad b_3 = 0, \end{aligned} \quad (5.10)$$

where $[1, \dots, 1]_m$ is the row vector of length m with ones, and e_j^ℓ is the j-th coordinate column vector of length ℓ. The execution of the different layers in (5.10) for an input $y \in \mathbb{R}^m$ can be calculated as follows:

$$y \mapsto z = V_2 \rho_1 V_1(y) = (\|y - y_1\|_{\ell_2}^2, \dots, \|y - y_\ell\|_{\ell_2}^2)^T \mapsto \rho_2(z) = \Phi(y) \mapsto V_3 \Phi(y) = XR^{-1}\Phi(y).$$

□

5.2. **Formal statement and proof of theorem 2.4.** In this section we present the longest proof of the paper. The impossibility results of the theorem are direct consequences of the driving proposition, while several of the other results require a substantial amount of work. As previously mentioned, we prove our second main result by proving the following more specific, but technical statement.

Theorem 5.3. *Given any $N, m \in \mathbb{N}$ and a fixed non-zero linear map $A : \mathbb{R}^N \to \mathbb{R}^m$ with non-trivial kernel and an integer $\ell \geq 2$, there exists a collection of initial domains $\mathcal{M}_1 \subset \mathbb{R}^N$, a domain Ω (as described in §1.1) of training sets $\mathcal{T}$ with $|\mathcal{T}| = \ell$, $\mathcal{T} \subseteq B_1(0)$, and a mapping $\Xi : \Omega \rightrightarrows \mathcal{NN}_{m,N}$ as in* (1.6)*. In particular, there exists optimal neural networks $\mathbf{N}_{opt}^{\mathcal{M}_1} \in \Xi(\mathcal{T}) \neq \emptyset$. However, the following happens simultaneously:*

(i) For any algorithm $\Gamma : \Omega \to \mathcal{NN}_{m,N}$ that works with inexact input on the computational problem $\{\Xi, \Omega, \mathcal{M}, \Lambda\}^{\Delta_1}$ there exists an inexact representation $\tilde{\mathcal{T}}_1 \in \tilde{\Omega}$ of a training set $\mathcal{T}_1 \in \Omega$ such that

$$\sup_{y \in \bigcup_{\mathcal{F}} \mathcal{M}_2} \|\Gamma(\tilde{\mathcal{T}}_1)(y) - \mathbf{N}_{opt}^{\mathcal{M}_1}(y)\|_{\ell_2} \geq 3/16 > 10^{-1},$$

and such that for any randomised algorithm Γ^{ran} there exists an inexact representation $\tilde{\mathcal{T}}_2 \in \tilde{\Omega}$ of a training set $\mathcal{T}_2 \in \Omega$ such that

$$\mathbf{P}\left(\sup_{y\in\bigcup_{\mathcal{F}}\mathcal{M}_2}\|\Gamma^{\mathrm{ran}}_{\tilde{\mathcal{T}}_2}(y)-\mathbf{N}^{\mathcal{M}_1}_{opt}(y)\|_{\ell_2}\geq 3/16>10^{-1}\right)\geq\frac{1}{2}.$$

(ii) However, there exists an infinite collection of training sets $\{\mathcal{T}_i\}_{i\in\mathbb{N}}\subset\Omega$ and an algorithm Γ that takes inputs in Ω, that for any $\epsilon>0$ and $\mathcal{T}_i$ produces an ϵ-approximation to an optimal neural network in $\Xi(\mathcal{T}_i)$. Yet, for any infinite sequence $\{\mathcal{T}_k\}_{k\in\mathbb{N}}\subset\Omega$ of different elements in Ω and any algorithm Γ that produces an ϵ-approximation to an optimal neural network in $\Xi(\mathcal{T}_k)$ for all $k\in\mathbb{N}$ we have the following. There exists an infinite subsequence $\{\mathcal{T}_{k_j}\}_{j\in\mathbb{N}}\subseteq\{\mathcal{T}_k\}_{k\in\mathbb{N}}$ such that for each $j\in\mathbb{N}$, there exists an element $(x,y)\in\mathcal{T}_{k_j}$ and an element $(x',y')\in\mathbb{R}^N\times\mathbb{R}^m$ with $\|x'\|_{\ell_2},\|y'\|_{\ell_2}\leq 1$ and $\|(x,y)-(x',y')\|_{\ell_2}\leq\sqrt{2}/4^j$ such that if we replace (x,y) with (x',y') then we obtain a new training set $\mathcal{T}'_{k_j}=[\mathcal{T}_{k_j}\setminus(x,y)]\cup(x',y')\in\Omega$ such that

$$\sup_{y\in\bigcup_{\mathcal{F}}\mathcal{M}_2}\|\Gamma(\mathcal{T}'_{k_j},\epsilon)(y)-\mathbf{N}^{\mathcal{M}'_1}_{opt}(y)\|_{\ell_2}>10^{-1},\quad\forall\,\epsilon>0,$$

with $\mathbf{N}^{\mathcal{M}'_1}_{opt}$ being an optimal neural network for the inverse problem $(A,\mathcal{M}'_1)$ which corresponds to the training set $\mathcal{T}'_{k_j}\in\Omega$.

(iii) Moreover, the statement in (ii) is true if we add one particular element, as opposed to changing one. In particular, for any $j\in\mathbb{N}$ we can add one element in $\mathcal{T}_{k_j}$ and thereby obtain a new training set $\mathcal{T}''_{k_j}$ with $|\mathcal{T}''_{k_j}|=\ell+1$ such that $\Xi(\mathcal{T}''_{k_j})$ is well defined yielding an optimal neural network for an $\mathcal{M}''_{1,k_j}$ with $\mathcal{M}_{1,k_j}\subset\mathcal{M}''_{1,k_j}$. However, for any such $\mathcal{T}''_{k_j}$ we get that

$$\sup_{y\in\bigcup_{\mathcal{F}}\mathcal{M}_2}\|\Gamma(\mathcal{T}''_{k_j},\epsilon)(y)-\mathbf{N}^{\mathcal{M}''_1}_{opt}(y)\|_{\ell_2}>10^{-1},\quad\forall\,\epsilon>0.$$

(iv) Approximating Ξ with error control when reading inexact input is at least as hard as deciding the halting problem, because assuming that $\{\Xi,\mathcal{M},\Lambda,\Omega\}^{\Delta_1}\in\Delta_1^A$ implies that we can decide the halting problem.

Proof of theorem 5.3. Let $A:\mathbb{R}^N\to\mathbb{R}^m$ be an arbitrary linear mapping with non-trivial kernel and let $\kappa=3/8$. Let $v,e\in\mathbb{R}^N$ be fixed vectors such that $v\in ker(A)$ with $\|v\|_{\ell_2}=2\kappa$ and $e\in ker(A)^\perp$ with $\|e\|_{\ell_2}=1$, and let $\mathcal{T}_b\in T_{\ell-2}$ is a finite training set with distinct non-zero elements in $\pi_1(\mathcal{T}_b)$ such that

$$\mathcal{T}_b\subset\ker(A)^\perp\times A(\ker(A)^\perp),\quad(B^+_{1/4}(v),A(B^+_{1/4}(v)))\notin\mathcal{T}_b,\tag{5.11}$$

where $B^{\ell,+}_{1/4}(v)=\{v'\in\mathbb{R}^N\mid v\leq_\ell v'\leq_\ell v+\frac{1}{4}e\}$, and where $\leq_\ell$ denotes the lexicographic ordering. We then define $\Omega=\{\iota^1_n\}_{n\in\mathbb{N}}\cup\{\iota^2_n\}_{n\in\mathbb{N}}\cup\{\alpha_n\}_{n\in\mathbb{N}}$ where ι^1_n and ι^2_n are the same as in the proof of Proposition 5.1, given by $\iota^1_n=\mathcal{T}_b\cup\{(0,0),(v+\frac{\theta}{4^n}e,A(v+\frac{\theta}{4^n}e))\}$ and $\iota^2_n=\mathcal{T}_b\cup\{(0,0),(v,0)\}$, where $\theta=\|A\|^{-1}_{op}$ if $\|A\|_{op}>1$ and $\theta=1$ otherwise. The corresponding inverse problems are $(A,\mathcal{M}^j_{1,n})$ where $\mathcal{M}^j_{1,n}=\pi_1(\iota^j_n)$ for $n\in\mathbb{N}$ and $j=1,2$. Further, we define $\alpha_n=\mathcal{T}'_b\cup(0,0)$ for all $n\in\mathbb{N}$, where $\mathcal{T}'_b\subseteq T^1_{\ell-1}$ satisfies the assumptions as stated for $\mathcal{T}_b$ in eq. (5.11), however, $\mathcal{T}'_b$ has cardinality $\ell-1$ while $\mathcal{T}_b$ had cardinality $\ell-2$. Similarly as for ι^j_n, the corresponding inverse problem to α_n is $(A,\mathcal{M}_{1,n})$ with $\mathcal{M}_{1,n}=\pi_1(\alpha_n)$. It is clear that there exists uncountably many linear maps $A:\mathbb{R}^N\to\mathbb{R}^m$ that satisfy the assumptions in Proposition 5.1, and for each such linear map there exists uncountably many domains $\Omega\subseteq T^1_\ell$ on the form described above, each of which gives rise to a computational problem on the form described in section 4.3. Let now $\{\Xi,\Omega,\mathcal{M},\Lambda\}$ be an arbitrary one of these problems, we then proceed by proving points (i), (ii) and (iii), each in turn for $\{\Xi,\Omega,\mathcal{M},\Lambda\}$.

Proof of (i). From the proof of proposition 5.1 it is clear that that the conditions $a)-c)$ in proposition 4.13 are satisfied for $\Omega'=\{\iota^1_n\}_{n\in\mathbb{N}}\cup\{\iota^2_n\}_{n\in\mathbb{N}}\subset\Omega$ for $\kappa=3/8$. Therefore by part (ii) in Proposition 4.13 we can conclude that $\epsilon^s_B\geq\epsilon^s_{\mathbb{P}B}(p)\geq\frac{1}{2}\kappa$, for all $p\in[0,\frac{1}{2})$ for the computational problem $\{\Xi,\Omega',\mathcal{M},\Lambda\}$ and

by the inclusion $\Omega' \subset \Omega$ also for the computational problem $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$. Hence, by the definition of the breakdown epsilons it follows that for any generalized algorithm Γ there exists a training set $\tilde{\mathcal{T}}_{fail} \in \tilde{\Omega}' \subset \tilde{\Omega}$ (recall the definition of $\tilde{\Omega}'$ and $\tilde{\Omega}$ from (4.5) in Definition 4.6) such that

$$\sup_{y \in \bigcup_{\mathcal{F}} \mathcal{M}_2} \|\Gamma(\tilde{\mathcal{T}_{fail}})(y) - \mathbf{N}_{opt}^{\mathcal{M}_1}(y)\|_{\ell_2} \geq \kappa/2 \geq 3/16 > 10^{-1}. \tag{5.12}$$

and for any randomised algorithm Γ^{ran} there exists a training set $\mathcal{T}_2 \in \tilde{\Omega}' \subset \tilde{\Omega}$ such that

$$\mathbf{P}\left(\sup_{y \in \bigcup_{\mathcal{F}} \mathcal{M}_2} \|\Gamma^{\mathrm{ran}}(\tilde{\mathcal{T}_2})(y) - \mathbf{N}_{opt}^{\mathcal{M}_1}(y)\|_{\ell_2} \geq \kappa/2 \geq 3/16 > 10^{-1} \right) \geq \frac{1}{2}.$$

Proof of (ii). Consider the collection $\{\mathcal{T}_i\}_{i \in \mathbb{N}} \subseteq \Omega$ defined by $\mathcal{T}_i = \iota_i^1 = \mathcal{T}_b \cup \{(0,0), (v + \frac{\theta}{4^i}e, A(v + \frac{\theta}{4^i}e))\}$ for all $i \in \mathbb{N}$. We now proceed by proving part (ii) of the theorem in several steps.

Step I. *Proving the computational breakdown in part (ii).* Let $\{\mathcal{T}_k\}_{k \in \mathbb{N}}$ be an arbitrary fixed infinite sequence in Ω and let $\Gamma : \tilde{\Omega} \times (0,1] \to \mathcal{NN}_{m,N}$ be any algorithm that – given any input $\epsilon > 0$ and any of the training sets $\mathcal{T}_k$ – produces an ϵ-approximation, as in (1.10), to an optimal neural network in $\Xi(\mathcal{T}_k)$. We then claim that there exists an infinite subsequence $\{\mathcal{T}_{k_j}\}_{j \in \mathbb{N}}$ and an element $(x', y') \in \mathbb{R}^N \times \mathbb{R}^m$ with $\|x'\|_{\ell_2}, \|y'\|_{\ell_2} \leq 1$ such that for any $j \in \mathbb{N}$ there exists an element $(x, y) \in \mathcal{T}_{k_j}$ such that $\|(x,y) - (x',y')\|_{\ell_2} \leq \sqrt{2}/4^j$, and such that if we replace (x, y) with (x', y') then we obtain a new training set $\mathcal{T}'_{k_j} = [\mathcal{T}_{k_j} \setminus (y, x)] \cup (x', y') \in \Omega$ such that there is a

$$\sup_{y \in \bigcup_{\mathcal{F}} \mathcal{M}_2} \|\Gamma(\mathcal{T}'_{k_j}, \epsilon)(y) - \mathbf{N}_{opt}^{\mathcal{M}'_1}(y)\|_{\ell_2} > 10^{-1}, \quad \forall\, \epsilon > 0,$$

with $\mathbf{N}_{opt}^{\mathcal{M}'_1}$ being an optimal neural network for the inverse problem $(A, \mathcal{M}'_1)$. Indeed, we may deduce from part (i) that $\sup_{y \in \bigcup_{\mathcal{F}} \mathcal{M}_2} \|\Gamma(\tilde{\mathcal{T}_{fail}}, \epsilon)(y) - \mathbf{N}_{opt}^{\mathcal{M}_1}(y)\|_{\ell_2} > 10^{-1}$, for all $\epsilon > 0$, for some $\mathcal{T}_{fail} \in \Omega'$. However, since we have $\sup_{y \in \bigcup_{\mathcal{F}} \mathcal{M}_2} \|\Gamma(\tilde{\mathcal{T}}_k, \epsilon)(y) - \mathbf{N}_{opt}^{\mathcal{M}_1}(y)\|_{\ell_2} \leq \epsilon$ for all $\epsilon > 0$ for infinitely many different $\mathcal{T}_k \in \Omega$ – and after observing that $\Omega = \{\iota_n^1\}_{n \in \mathbb{N}} \cup \{\mathcal{T}_b \cup \{(0,0),(v,0)\}\} \cup \{\mathcal{T}'_b \cup \{(0,0)\}\}$ – it follows that Γ must work for infinitely many of the ι_n^1's.

Thus, by observing that the conditions $a) - c)$ in proposition 4.13 are satisfied for $\Omega' = \{\iota_n^1\}_{n \in \mathbb{N}} \cup \{\iota_n^2\}_{n \in \mathbb{N}} = \{\iota_n^1\}_{n \in \mathbb{N}} \cup \{\mathcal{T}_b \cup \{(0,0),(v,0)\}\}$, it follows that any algorithm that works on infinitely many ι_n^1's must fail on the set $\{\mathcal{T}_b \cup \{(0,0),(v,0)\}\}$. Hence, $\tilde{\mathcal{T}}_{fail}$ must be an inexact representation of the training set $\mathcal{T}_b \cup \{(0,0),(v,0)\}$. Let now $\{\mathcal{T}_{k_j}\}_{j \in \mathbb{N}}$ be the infinite subsequence of $\{\mathcal{T}_k\}_{k \in \mathbb{N}}$ consisting of the elements $\mathcal{T}_k$ that are contained in $\{\iota_n^1\}$. It is then clear that $\mathcal{T}_{fail}$ differs by only one element from each of the training sets $\mathcal{T}_{k_j}$ for $j \in \mathbb{N}$. Hence, by setting $(x', y') = (v, 0)$ and $(x, y) = (v + \frac{\theta}{4^{k_j}}e, A(v + \frac{\theta}{4^{k_j}}e))$ for all $j \in \mathbb{N}$, it's not hard to see that

$$\|(x',y') - (x,y)\|_{\ell_2} = \|(\frac{\theta}{4^{k_j}}e, A(\frac{\theta}{4^{k_j}}e))\|_{\ell_2} = \sqrt{\frac{1}{4^{2k_j}} + \frac{1}{4^{2k_j}}} = \frac{\sqrt{2}}{4^{k_j}} \leq \frac{\sqrt{2}}{4^j},$$

and that by replacing (x, y) with (x', y') in $\mathcal{T}_{k_j}$ we have proved the desired claim.

Step II. *Constructing* Γ. We notice that since $\mathcal{M}_{1,n}^j = \iota_n^j|_{\mathbb{R}^N}$ for all $n \in \mathbb{N}$ and $j = 1, 2$ we get that any neural network $\mathbf{N} \in \mathcal{NN}_{m,N}$ such that $\mathbf{N}(y) = x$ for all $(x, y) \in \iota_n^1$ will be an optimal neural network for $\mathcal{M}_{1,n}^1$ for all $n \in \mathbb{N}$. Thus, we next demonstrate how we can recursively approximate a neural network that interpolates all the points in ι_n^1 for each $n \in \mathbb{N}$. To achieve this we use the radial basis function approach and recursively approximate the interpolating neural network $s : \mathbb{R}^N \to \mathbb{R}^m$ obtained in Lemma 5.2.

We will construct an algorithm $\Gamma : \tilde{\Omega} \times (0,1] \to \mathcal{NN}_{m,N}$, such that for each rational $\epsilon > 0$ and $n \in \mathbb{N}$, we have that $\sup_{y \in \bigcup_{\mathcal{F}} \mathcal{M}_2} \|\Gamma(\tilde{\iota}_n^1, \epsilon)(y) - \mathbf{N}_{\iota_n^1}(y)\| \leq \epsilon$, for all $n \in \mathbb{N}$, where $\mathbf{N}_{\iota_n^1}(y)$ is the interpolating neural network defined in (5.9). To do this we recall (5.9) and observe by (5.10) that one can construct an approximation to $\mathbf{N}_{\iota_n^1}$ from $\tilde{\iota}_n^1 \in \tilde{\Omega}$. Indeed, let $\hat{\Lambda}$ be an arbitrary fixed set that provides Δ_1-information for Ω as defined in definition 4.5, and recall that for each $(x, y) \in \iota_n^1$ we have the corresponding $(\tilde{x}, \tilde{y}) \in \tilde{\iota}_n^1$, as

defined in eq. (4.8). We use the notation $\tilde{x}^j, \tilde{y}^j$ where

$$\|\tilde{x}^j - x\|_{\ell_2} \leq 2^{-j} \quad \text{and} \quad \|\tilde{y}^j - y\|_{\ell_2} \leq 2^{-j}. \tag{5.13}$$

Note that the matrices $X = X(x_1, \ldots, x_\ell)$ and $R = R(y_1, \ldots, y_\ell)$, from (5.7) and (5.8), depend on $(x_1, \ldots, x_\ell)$ and $(y_1, \ldots, y_\ell)$ respectively, where $\{(x_i, y_i)\}_{i=1}^{\ell} = \iota_n^1$. Similarly, we have that $\Phi(\cdot) = \Phi(\cdot, y_1, \ldots, y_\ell)$ from (5.6) depends on $(y_1, \ldots, y_\ell)$. To approximate X, R and Φ we define, for $\{(\tilde{x}_i, \tilde{y}_i)\}_{i=1}^{\ell} = \tilde{\iota}_n^1$, the approximations

$$X_j = X(\tilde{x}_1^j, \ldots, x_\ell^j), \quad R_j = R(\tilde{y}_1^j, \ldots, \tilde{y}_\ell^j), \quad \Phi_j(\cdot) = \Phi(\cdot, \tilde{y}_1^j, \ldots, \tilde{y}_\ell^j). \tag{5.14}$$

Define $k : \mathbb{N} \to \mathbb{N}$ by

$$\begin{aligned} k(j) := \min\{\mu \in \mathbb{N} \,|\, \forall r \geq \mu,\ & R_r \text{ is invertible and} \\ & \sup_{y \in \bigcup_{\mathcal{F}} \mathcal{M}_2} \|X_r R_r^{-1} \Phi_r(y) - X R^{-1} \Phi(y)\|_{\ell_2} \leq 2^{-j}\}. \end{aligned} \tag{5.15}$$

We note that k is well defined: Firstly, since the all the point's in $\pi_2(\iota_n^1)$ are distinct, there must exist a finite $\mu \in \mathbb{N}$ such that $\tilde{y}_i^\mu \neq \tilde{y}_j^\mu$ for all $y_i, y_j \in \pi_2(\iota_n^1)$ with $i \neq j$, making R_r invertible for all $r \geq \mu$ by lemma 5.2. Secondly, from the definitions in (5.7), (5.8) and (5.6), $X_j \to X$, $R_j \to R$, $\Phi_j \to \Phi$ as $j \to \infty$, making k well defined.

We will now estimate an upper bound for k, defined in (5.15), that can be recursively computed from $\{(\tilde{x}_i, \tilde{y}_i)\}_{i=1}^{\ell}$. In order to do that we need to start by recursively finding a lower bound for the smallest singular value β_R of R and an upper bound for the largest singular value β_X of X. The upper bound for X can be derived theoretically in a quite simple way. Indeed, by using the fact that $\|X\|_{op} = \max_{\beta \in \mathrm{Sp}(X^*X)} \sqrt{\beta}$ we can observe that

$$\sqrt{\beta} \leq \|X\|_{op} \leq \sqrt{\sum_{i=1}^{\ell} \|x_i\|_{\ell_2}^2} \leq \sqrt{\sum_{i=1}^{\ell} 1} = \sqrt{\ell} < \ell \quad \text{for all } \beta \in \mathrm{Sp}(X^*X), \tag{5.16}$$

where $\mathrm{Sp}(X^*X)$ denotes the spectrum of the operator X^*X. For the matrix R the task is slightly more complicated, and we need to use sufficient approximations of R in order to achieve the task of finding a lower bound for the smallest singular value. More specifically, we need to find a lower bound for $2^{-h(j)}$ where $h : \mathbb{N} \to \mathbb{N}$ is given by

$$h(j) := \min\{\mu \in \mathbb{N} \,|\, \forall r \geq \mu\,, \sup_{y \in \bigcup_{\mathcal{F}} \mathcal{M}_2} \|R_r(y) - R(y)\|_{\ell_2} \leq 2^{-j}\}. \tag{5.17}$$

Claim: We claim that for any arbitrary fixed $j \in \mathbb{N}$

$$2^{-r} \leq \frac{1}{2} \frac{1}{2m} \frac{1}{\ell} 2^{-j} \;\Rightarrow\; h(j) \leq r, \tag{5.18}$$

where h is defined in (5.17), when $r \in \mathbb{N}$ chosen as in (5.18). Indeed, we verify this by the following series of calculations. We first observe that

$$\begin{aligned} \|R_r - R\|_{op} &= \|[\phi(\|\tilde{y}_j^r - y_i^r\|_{\ell_2}^2) - \phi(\|y_j - y_i\|_{\ell_2}^2)]_{i,j=1}^{i,j=\ell}\|_{\ell_2} \\ &= \|[\hat{\phi}(\tilde{y}_j^r, \tilde{y}_i^r) - \hat{\phi}(y_j, y_i)]_{i,j=1}^{i,j=\ell}\|_{\ell_2}, \end{aligned}$$

where $\hat{\phi}(y_j, y_i) = 1/(1 + \|y_j - y_i\|_{\ell_2}^2)$, and where $\tilde{y}_j^r$ and $\tilde{y}_i^r$ are the approximations of y_j and y_i to accuracy 2^{-r} as defined in eq. (5.13). The mean value theorem for scalar fields then gives us that there exists a $c \in (0, 1)$ such that

$$\|R_r - R\|_{\ell_\infty} = \max_{i,j} |[R_r - R]_{i,j}| \leq \max_{i,j} \|\nabla \hat{\phi}((1-c) w_1^{i,j,r} + c w_2^{i,j}\|_{\ell_2} \|w_2^{i,j} - w_1^{i,j,r}\|_{\ell_2},$$

where $w_1^{i,j,r} = (\tilde{y}_{j1}^r, \ldots, \tilde{y}_{jm}^r, \tilde{y}_{i1}^r, \ldots, \tilde{y}_{im}^r)$ and $w_2^{i,j} = (y_{j1}, \ldots, y_{jm}, y_{i1}, \ldots, y_{im})$. Next, we notice that

$$\sup_{q_1, q_2 \in \mathbb{R}^m} \|\nabla \hat{\phi}(q_1, q_2)\|_{\ell_\infty} = \sup_{q_1, q_2 \in \mathbb{R}^m} (-2\|q_1 - q_2\|_{\ell_\infty})/(1 + \|q_1 - q_2\|_{\ell_2}^2)^2.$$

By substituting $q = q_1 - q_2$, we can find an upper bound for the supremum above, by finding an upper bound for $\max_i \sup_{q\in\mathbb{R}^m} 2|q_i|/(1+\|q\|_{\ell_2}^2)^2$. We notice that we either have that $|q_i| \leq \|q\|_{\ell_2}^2$ or that $|q_i| \leq 1$, and in either way $|q_i| \leq 1 + \|q\|_{\ell_2}^2$, hence we can conclude that $\max_i \sup_{q\in\mathbb{R}^m} 2|q_i|/(1+\|q\|_{\ell_2}^2)^2 \leq 2$, and thus $\|\nabla\hat{\phi}((1-c)w_1^{i,j,r} + cw_2^{i,j})\|_{\ell_2} \leq \sqrt{2^2 m} < 2m$, since $m > 1$. Furthermore we have that $\|w_1^{i,j,r} - w_2^{i,j}\|_{\ell_2} \leq \|\tilde{y}_j^r - y_j\|_{\ell_2} + \|\tilde{y}_i^r - y_i\|_{\ell_2}$, and thus we get that

$$\|R_r - R\|_{op} \leq \|R_r - R\|_{\ell_2} = \sqrt{\sum_{i,j}[R_r - R]_{i,j}^2} \leq \sqrt{\sum_{i,j}(2m)^2(\|\tilde{y}_j^r - y_j\|_{\ell_2} + \|\tilde{y}_i^r - y_i\|_{\ell_2})^2}.$$

At last, since $2^{-r} \leq \frac{1}{2}\frac{1}{2m}\frac{1}{\ell}2^{-j}$, we can conclude that

$$\begin{aligned}\|R_r - R\|_{op} \leq \|R_r - R\|_{\ell_2} &\leq \sqrt{\sum_{i,j}(2m)^2(\|\tilde{y}_j^r - y_j\|_{\ell_2} + \|\tilde{y}_i^r - y_i\|_{\ell_2})^2} \\ &\leq \sqrt{(2m)^2\ell^2(\frac{1}{2}\frac{1}{2m}\frac{1}{\ell}2^{-j} + \frac{1}{2}\frac{1}{2m}\frac{1}{\ell}2^{-j})^2} = \sqrt{(2m)^2\ell^2\frac{1}{(2m)^2}\frac{1}{\ell^2}2^{-2j}} = 2^{-j},\end{aligned}$$

which proves our claim.

Using this, we can now present a recursive algorithm that finds a lower bound for the smallest singular value of R – note that R self-adjoint and positive definite, thus the smallest singular value coincides with the smallest element in the spectrum of R:

Inputs: Oracles for all $\tilde{y} \in \pi_2(\tilde{\iota}_n^1)$.

Outputs: A lower bound for the singular values of R. More precisely, the integer k_R such that $2^{-k_R} \leq \beta_R$ for all $\beta_R \in Sp(R)$.

1. Put $j = \hat{n} = 1$.
2. Choose r such that $2^{-r} \leq \frac{1}{2}\frac{1}{2m}\frac{1}{\ell}2^{-(j+1)}$.
3. Use the input oracles to read all $\tilde{y} \in \pi_2(\tilde{\iota}_n^1)$ to precision r and construct the matrix R_r. As we have seen, the criterion in step 2 implies that $\|R_r - R\|_{op} \leq \|R_r - R\|_{\ell_2} \leq 2^{-(j+1)}$.
4. Check – by trying to compute the Cholesky decomposition – whether the matrix $R_r - 2^{-j}I$ is positive definite.
   a. If it is, then we have found a lower bound for the smallest singular value of R since

$$\begin{aligned}Sp(R_r - 2^j I) &\subseteq Sp(R + \epsilon_{j+1} - 2^{-j}I) \\ &= \{\beta + \beta_\epsilon - 2^{-j} \mid \beta \in Sp(R) \text{ and } |\beta_\epsilon| \leq 2^{-(j+1)}\} \subseteq (0,\infty),\end{aligned}$$

   which implies that $\beta \geq 2^{-j} - \beta_\epsilon \geq 2^{-j} - 2^{-(j+1)} = 2^{-(j+1)}$ for all $\beta \in Sp(R)$, where $Sp(R)$ denotes the spectrum of R, which equals the set of singular values of R. Set $k_R = j+1$ and return 2^{-k_R}.
   b. If it is not, then put $j = \hat{n} + 1$ and repeat steps 2,3 and 4.

We notice that the above process will always terminate after a finite number of steps, since R is finite dimensional and positive definite by lemma 5.2. We are now ready to present a lower bound for $2^{-k(j)}$, where k is as defined in eq. (5.15).

Claim: We claim that for any fixed $j \in \mathbb{N}$, if $r \in \mathbb{N}$ is such that

$$2^{-r} \leq \min\left\{\frac{1}{7}\frac{1}{4}\frac{1}{2m}\frac{1}{\ell^2}2^{-2k_R}2^{-j}, \frac{1}{7}\frac{1}{\ell}2^{-k_R}2^{-j}\right\},$$

and such that $\tilde{y}_i^r \neq \tilde{y}_j^r$ for all $y_i, y_j \in \pi_2(\iota_n^1)$ with $i \neq j$, then 2^{-r} is a lower bound for $2^{-k(j)}$. We now proceed with verifying this claim. With the approximations $\tilde{y}^r$ and $\tilde{x}^r$ as defined in eq. (5.13) we get that:

$$\begin{aligned}
&\sup_{y\in\bigcup_{\mathcal{F}}\mathcal{M}_2} \|X_r R_r^{-1}\Phi_r(y) - XR^{-1}\Phi(y)\|_{\ell_2} \\
&\quad= \sup_{y\in\bigcup_{\mathcal{F}}\mathcal{M}_2} \|X_r R_r^{-1}\Phi_r(y) - XR^{-1}\Phi_r(y) + XR^{-1}\Phi_r(y) - XR^{-1}\Phi(y)\|_{\ell_2} \\
&\quad\leq \sup_{y\in\bigcup_{\mathcal{F}}\mathcal{M}_2} \|X_r R_r^{-1} - XR^{-1}\|_{op}\|\Phi_r(y)\|_{\ell_2} + \|R^{-1}\|_{op}\|X\|_{op}\|\Phi_r(y) - \Phi(y)\|_{\ell_2}.
\end{aligned}$$

With a similar deduction as for $\|R_r - R\|_{op}$ we get that $\sup_y \|\Phi_r(y) - \Phi(y)\|_{\ell_2} \leq 2^{-j}$ if r is such that $\|\tilde{y}^r - y\|_{\ell_2} \leq 2^{-r} \leq \frac{1}{2}\frac{1}{2m}\frac{1}{\sqrt{\ell}}2^{-j}$ for all $y \in \pi_2(\iota_n^1)$, and we can easily see that this bound holds true by our choice of r in the section above. Thus,

$$\sup_{y\in\bigcup_{\mathcal{F}}\mathcal{M}_2} \|R^{-1}\|_{op}\|X\|_{op}\|\Phi_r(y) - \Phi(y)\|_{\ell_2} \leq 2^{k_R}\ell \sup_y \|\Phi_r(y) - \Phi(y)\|_{\ell_2} \leq 2^{k_R}\ell\frac{1}{\ell}2^{-k_R}\frac{1}{7}2^{-j} \leq \frac{1}{7}2^{-j},$$

by our choice of r. To ease notation in further calculations, we fix y to be an arbitrary element in $\bigcup_{\mathcal{F}}\mathcal{M}_2$, and we can continue our calculations as follows:

$$\begin{aligned}
\|X_r R_r^{-1}\Phi_r(y) - XR^{-1}\Phi(y)\|_{\ell_2} &\leq \|X_r R_r^{-1} - XR^{-1}\|_{op}\|\Phi_r(y)\|_{\ell_2} + \frac{1}{7}2^{-j} \\
&= \|X_r R_r^{-1} - X_r R^{-1} + X_r R^{-1} - XR^{-1}\|_{op}\|\Phi_r(y)\|_{\ell_2} + \frac{1}{7}2^{-j} \\
&\leq (\|R_r^{-1} - R^{-1}\|_{op}\|X_r\|_{op} + \|R^{-1}\|_{op}\|X_r - X\|_{op})\|\Phi_r(y)\|_{\ell_2} + \frac{1}{7}2^{-j} \\
&= \|R_r^{-1} - R^{-1}\|_{op}\|X_r\|_{op}\|\Phi_r(y)\|_{\ell_2} + \|R^{-1}\|_{op}\|X_r - X\|_{op}\|\Phi_r(y)\|_{\ell_2} + \frac{1}{7}2^{-j}
\end{aligned} \tag{5.19}$$

The formula above consists of three terms, where we need to calculate the bounds on the first two. For the sake of structure and overview we consider each of the terms separately. We start with the term $\|R^{-1}\|_{op}\|X_r - X\|_{op}\|\Phi_r(y)\|_{\ell_2}$ and we observe that:

$$\begin{aligned}
\|R^{-1}\|_{op}\|X_r - X\|_{op}\|\Phi_r(y)\|_{\ell_2} &= \|R^{-1}\|_{op}\|X_r - X\|_{op}\|\Phi_r(y) - \Phi(y) + \Phi(y)\|_{\ell_2} \\
&\leq \|R^{-1}\|_{op}\|X_r - X\|_{op}(\|\Phi_r(y) - \Phi(y)\|_{\ell_2} + \|\Phi(y)\|_{\ell_2})
\end{aligned}$$

Next we notice that $\|\Phi(y)\|_{\ell_2}^2 = \sum_{1\leq i\leq \ell} 1/(1 + \|y - y_i\|_{\ell_2}^2)^2 \leq \sum_{1\leq i\leq \ell} 1 = \ell$, thus $\|\Phi(y)\|_{\ell_2} \leq \sqrt{\ell}$. Moreover, we have chosen r such that $\|\Phi_r(y) - \Phi(y)\|_{\ell_2} \leq 2^{-j} \leq 1$, thus we can continue our calculations as follows:

$$\|R^{-1}\|_{op}\|X_r - X\|_{op}\|\Phi_r(y)\|_{\ell_2} \leq \|R^{-1}\|_{op}\|X - X_r\|_{op}(1 + \sqrt{\ell}) \leq \frac{1}{7}2^{-j} + \frac{1}{7}2^{-j}, \tag{5.20}$$

where the last inequality follows by our choice of r, more precisely that $\|\tilde{x}^r - x\|_{\ell_2} \leq 2^{-r} \leq \frac{1}{7}\frac{1}{\ell}2^{-k_R}2^{-j}$ for all $x \in \pi_1(\iota_n^1)$ and that $\|X_r - X\|_{op} \leq \sqrt{\ell}\max_{x\in\pi_1(\iota_n^1)}\|\tilde{x}^r - x\|_{\ell_2} \leq \sqrt{\ell}\cdot 2^{-r}$.
We now move on to the second term $\|R_r^{-1} - R^{-1}\|_{op}\|X_r\|_{op}\|\Phi_r(y)\|_{\ell_2}$ of eq. (5.19). By the second resolvent identity we get that

$$\|R_r^{-1} - R^{-1}\|_{op}\|X_r\|_{op}\|\Phi_r(y)\|_{\ell_2} \leq \|R_r^{-1}\|_{op}\|R^{-1}\|_{op}\|R_r - R\|_{op}\|X_r\|_{op}\|\Phi_r(y)\|_{\ell_2}.$$

In order to continue our calculations we need to obtain an upper bound for $\|R_r^{-1}\|_{op}$ and $\|X_r\|_{op}$. For $\|X_r\|_{op}$ this is fairly straight forward and we see that $\|X_r\|_{op} = \|X_r - X + X\|_{op} \leq \|X_r - X\|_{op} + \|X\|_{op} \leq 2^{-j} + \sqrt{\ell} \leq 1 + \sqrt{\ell}$. Deriving the upper bound for $\|R_r^{-1}\|_{op}$ is slightly more complicated.

We start by a similar calculation as for $\|X_r\|_{op}$ and get that $\|R_r^{-1}\|_{op} = \|R_r^{-1} - R^{-1} + R^{-1}\|_{op} \leq \|R_r^{-1} - R^{-1}\|_{op} + \|R^{-1}\|_{op}$. We continue our calculations by using the second resolvent identity: Since $\|R_r^{-1}\|_{op} \leq \|R^{-1}\|_{op}\|R_r^{-1}\|_{op}\|R_r - R\|_{op} + \|R^{-1}\|_{op}$ we get that $\|R_r^{-1}\|_{op}(1 - \|R^{-1}\|_{op}\|R_r - R\|_{op}) \leq \|R^{-1}\|_{op}$. Now, one can deduce that 2^{-r} is chosen small enough to guarantee that $\|R^{-1}\|_{op}\|R_r - R\|_{op} \leq 2^{-j}$. Thus, we get that $\|R_r^{-1}\|_{op}(1 - 2^{-j}) \leq \|R_r^{-1}\|_{op}(1 - \|R^{-1}\|_{op}\|R_r - R\|_{op}) \leq \|R^{-1}\|_{op}$, which gives

us that $\|R_r^{-1}\|_{op} \leq \|R^{-1}\|_{op}/(1-2^{-j}) \leq 2^{k_R}/(1-2^{-j})$. At last, since $j \geq 1$ we get that $\|R_r^{-1}\|_{op} \leq 2^{k_R}/(1-2^{-j}) \leq 2 \cdot 2^{k_R}$.

Using the calculations above and by our choice of r we can finish our calculations as follows:

$$\begin{aligned}&\|R_r^{-1}\|_{op}\|R^{-1}\|_{op}\|R_r - R\|_{op}\|X_r\|_{op}\|\Phi_r(y)\|_{\ell_2} \\ &\leq 2 \cdot 2^{k_R} \cdot 2^{k_R}\|R_r - R\|_{op}(1+\sqrt{\ell})^2 = 2 \cdot 2^{2k_R}(\ell + \sqrt{\ell} + \sqrt{\ell} + 1)\|R_r - R\|_{op} \leq \frac{4}{7}2^{-j}. \quad (5.21)\end{aligned}$$

Finally, by combining the bounds of each of the terms obtained in eq. (5.20) and eq. (5.21) with the equation in eq. (5.19), we can conclude that

$$\|R_r^{-1}X_r\Phi_r(y) - R^{-1}X\Phi(y)\|_{\ell_2} \leq 7 \cdot \frac{1}{7}2^{-j} = 2^{-j}$$

for all arbitrary $y \in \bigcup_{\mathcal{F}} \mathcal{M}_2$, where r satisfies the bound specified in eq. (5.18). Since $y \in \bigcup_{\mathcal{F}} \mathcal{M}_2$ was chosen arbitrarily we can conclude that the inequality above also holds true when we take the supremum over all $y \in \bigcup_{\mathcal{F}} \mathcal{M}_2$. At last, since $r \in \mathbb{N}$ is such that $\tilde{y}_i^r \neq \tilde{y}_j^r$ for all $y_i, y_j \in \pi_2(\iota_n^1)$ with $i \neq j$, we get by lemma 5.2 that R_r is invertible and $R_r^{-1} \circ X_r \circ \Phi_r(y) \in \mathcal{NN}_{m,N}$. Thus we can finally define

$$\Gamma(\tilde{\iota}_n^1, \epsilon) = R_r^{-1} \circ X_r \circ \Phi_r,$$

where j is chosen such that $2^{-j} \leq \epsilon$ and r is chosen such that $2^{-r} \leq 2^{-k(j)}$, with $2^{-k(j)}$ as defined in eq. (5.18). It is then clear that our algorithm $\Gamma : \tilde{\Omega} \times (0,1] \to \mathcal{NN}_{N,m}$ maps into the space of neural networks that are bounded on $\bigcup_{\mathcal{F}} \mathcal{M}_2$ and approximates an optimal neural network for the set $\mathcal{M}_{1,n}^1$ for any of the given training sets ι_n^1 with $n \in \mathbb{N}$. We can conclude that part (ii) of the result holds.

Proof of (iii). In order to prove this part of the theorem we construct a new domain $\hat{\Omega} \subseteq T_{\ell+1}$. Let $\hat{\iota}_n^1$ and $\hat{\iota}_n^2$ be given by $\hat{\iota}_n^1 = \mathcal{T}_b' \cup \{(0,0), (v + \frac{\theta}{4^n}e, A(v + \frac{\theta}{4^n}e))\}$ and $\iota_n^2 = \mathcal{T}_b' \cup \{(0,0), (v,0)\}$, where everything is as in the first section of the proof, except the fact that $\mathcal{T}_b'$ has cardinality $\ell - 1$ while $\mathcal{T}_b$ in the first section of the proof had cardinality $\ell - 2$. Similarly as before, we define the corresponding sets $\mathcal{M}_{1,i}^1$ and $\mathcal{M}_{1,i}^2$ to be $\mathcal{M}_{1,i}^1 = \pi_1(\hat{\iota}_n^1)$ and $\mathcal{M}_{1,i}^2 = \pi_1(\hat{\iota}_n^2)$. Next, just as in part (ii), we can use the radial basis function approach to interpolate the points in $\hat{\iota}_n^1$ and we can prove, by exactly the same argumentation as in part (i), that Γ will fail on $\hat{\iota}_n^2$, with an error of $3/16$. Let now $\{\mathcal{T}_i\} = \alpha_i$ for all $i \in \mathbb{N}$. Then $\alpha_i \subseteq \hat{\iota}_i^1$ and for the set $\mathcal{M}_{1,i}$ corresponding to α_i we have that $\mathcal{M}_{1,i} \subseteq \mathcal{M}_{1,i}^1$, and thus we can conclude that Γ interpolates all the points in α_i and thereby constructs an optimal neural network for $\mathcal{M}_{1,i}$. At last, since $\alpha_i \cup (v,0) = \hat{\iota}_i^2$ we can conclude that by adding the element $(v,0)$ to α_i the algorithm Γ fails with error $3/16$. The third part follows from this.

Proof of (iv). Our aim is to show that the assumption that $\{\Xi, \mathcal{M}, \Lambda, \Omega\}^{\Delta_1} \in \Delta_1^A$ implies the decidability of the halting problem. Let

$$\hat{\Omega} = \{(T,x) \mid T \text{ is a Turing machine, } x \text{ is an input to } T\}$$

and let $\hat{\Xi}(T,x) = 1$ if T halts on x and $\hat{\Xi}(T,x) = 0$ if T does not halt on x. We want to show that there exists a recursive mapping $\hat{\Gamma}$ such that $\hat{\Gamma}(T,x) = \hat{\Xi}(T,x)$. We start by defining the mapping $g_\iota : \hat{\Omega} \to \tilde{\Omega}$ recursively as follows. $g_\iota(T,x) = \{\iota_n\}_{n\in\mathbb{N}}$, where for each $n \in \mathbb{N}$, ι_n is defined as follows:

$$\iota_n = \begin{cases} \iota_n^1 & \text{if } T \text{ has not halted on } x \text{ after } n \text{ steps} \\ \iota_{n'}^1 & \text{if } T \text{ has halted on input } x \text{ after } n' \text{ steps where } 1 \leq n' < n. \end{cases}$$

We need to verify that g_ι indeed maps into $\tilde{\Omega}$, more specifically, that for each pair $(T,x) \in \hat{\Omega}$ there exists an element $\iota_{(T,x)} \in \Omega = \{\iota_n^1\}_{n\in\mathbb{N}} \cup \{\iota_n^2\}_{n\in\mathbb{N}}$ such that for each $k \in \mathbb{N}$ and $f \in \Lambda$ we have that

$$|f(\iota_k) - f(\iota_{(T,x)})| \leq 2^{-k}, \quad (5.22)$$

where $g_\iota(T,x) = \{\iota_n\}_{n\in\mathbb{N}}$. We first consider the pair (T,x) where T halts on input x after n' steps for some $n' \in \mathbb{N}$. It is then not hard to see that the inequality in eq. (5.22) holds with $\iota_{(T,x)} = \iota_{n'}^1$, since $\iota_n = \iota_{n'}$ for

all $n \geq n'$. Next we consider a pair (T, x) where T does not halt on input x. Then $g_\iota(T, x) = \{\iota_n^1\}_{n\in\mathbb{N}}$, and as we have seen in the proof of proposition 5.1 we have that $|f(\iota_k^1) - f(\iota_0)| \leq 2^{-k}$ for all $k \in \mathbb{N}$ and for all $f \in \Lambda$, where $\iota_0 = \mathcal{T}_b \cup \{(0,0),(0,v)\}$. Thus, we can conclude that the inequality in eq. (5.22) is satisfied with $\iota_{(T,x)} = \iota_0$. With this we can conclude that $g_\iota(T, x) \in \tilde{\Omega}$ for all $(T, x) \in \hat{\Omega}$, and that g_ι indeed maps into $\tilde{\Omega}$. At last, we notice that $g_\iota : \hat{\Omega} \to \tilde{\Omega}$ is a recursive function, since for each input $(T, x) \in \hat{\Omega}$ the output can be generated recursively.

Since by assumption $\{\Xi, \mathcal{M}, \Lambda, \Omega\}^{\Delta_1} \in \Delta_1^A$ we can choose an arithmetic tower $\{\Gamma_k\}_{k\in\mathbb{N}}$ where each algorithm $\Gamma_k : \tilde{\Omega} \to \mathcal{NN}_{m,N}$ is such that $\Gamma_k(\iota(T, x)) \in \mathcal{N}_{2^{-k}}(\Xi(\iota(T, x))$, where for a given set $S \in \mathcal{NN}_{m,N}$

$$\mathcal{N}_{2^{-k}}(S) = \{\mathbf{N} : \mathbb{R}^m \to \mathbb{R}^N \mid \inf_{\mathbf{N}'\in S} \sup_x \|\mathbf{N}(x) - \mathbf{N}'(x)\|_{\ell_2} \leq 2^{-k}\}$$

is the 2^{-k} neighbourhood of the set S. In order to construct $\hat{\Gamma} : \hat{\Omega} \to \{0, 1\}$ we proceed as follows. We consider the set $S^2 \subseteq \mathcal{NN}_{m,N}$ given by

$$S^2 = \{\mathbf{N} : \mathbb{R}^m \to \mathbb{R}^N \mid \mathbf{N}(y) = x \ \forall\ (x, y) \in \mathcal{T}_b \text{ and } \mathbf{N}(0) = (1/2)v\},$$

and we choose $k \in \mathbb{N}$ such that $1/(k-1) \leq 3/8$, where we recall that $3/8$ is the breakdown threshold from part (i). We then define

$$\hat{\Gamma}(T, x) = \begin{cases} 0 & \text{if } \Gamma_k(g_\iota(T, x)) \in \mathcal{N}_{k^{-1}}(S^2) \\ 1 & \text{otherwise.} \end{cases}$$

It remains to show that $\hat{\Gamma}(T, x) = \hat{\Xi}(T, x)$ for all $(T, x) \in \hat{\Omega}$ and that $\hat{\Gamma}$ is recursive. We start by showing the recursiveness: $\Gamma_k(g_\iota(T, x))$ is recursive, since $g_\iota : \hat{\Omega} \to \tilde{\Omega}$ is recursive by construction and $\Gamma_k : \tilde{\Omega} \to \mathcal{NN}_{m,N}$ is recursive by assumption. Thus, in order to conclude that $\hat{\Gamma}$ is recursive, it only remains to argue that one can recursively check whether $\Gamma_k(g_\iota(T, x)) \in \mathcal{N}_{k^{-1}}(S^2)$.

We notice first that it suffices to check whether $\|\Gamma_k(g_\iota(T, x)(y') - x'\|_{\ell_2} \leq k^{-1}$ for all $(x', y') \in \mathcal{T}_b \cup \{(0, (1/2)v)\}$. Thus, since the number of points in $\mathcal{T}_b$ is finite and all points in $\mathcal{T}_b \cup \{(0, (1/2)v)\}$ are rational we conclude that the task of checking whether $\Gamma_k(g_\iota(T, x)) \in \mathcal{N}_{k^{-1}}(S^2)$ can be achieved recursively in a finite number of steps.

At last we show that $\hat{\Gamma}(T, x) = \hat{\Xi}(T, x)$ for all $(T, x) \in \hat{\Omega}$, we notice that it follows from the following two arguments:

$\underline{\hat{\Xi}(T, x) = 1 \implies \hat{\Gamma}(T, x) = 1}$: Since $\hat{\Xi}(T, x) = 1$, T halts on input x. Thus, there exists an $n' \in \mathbb{N}$ such that T halts on x after n' steps. Hence, the sequence $g_\iota(T, x) = \{\iota_n\}_{n\in\mathbb{N}}$ is equal to ι_n^1 for all $N \geq n'$ and $\Gamma_k(g_\iota(T, x)) \in \mathcal{N}_{2^{-k}}(\Xi(\iota_n^1)) \subseteq \mathcal{N}_{2^{-k}}(S^1)$, since $\Xi(\iota_n^1) \subseteq S^1$, where

$$S^1 = \{\mathbf{N} : \mathbb{R}^m \to \mathbb{R}^N \mid \mathbf{N}(y) = x \ \forall\ (x, y) \in \mathcal{T}_b,\ \mathbf{N}(v + \frac{\theta}{4^n}e) = A(v + \frac{\theta}{4^n}e), \text{ and } \mathbf{N}(0) = 0\}.$$

Then $\Gamma_k(g_\iota(T, x)) \notin \mathcal{N}_{1^{-k}}(S^2)$ since if it was it would follow that

$$\begin{aligned}\inf_{\mathbf{N_1}\in S^1, \mathbf{N_2}\in S^2} d(\mathbf{N_1}, \mathbf{N_2}) &\leq \inf_{\mathbf{N_1}\in S^1, \mathbf{N_2}\in S^2} d(\mathbf{N_1}, \Gamma_k(g_\iota(T, x)) + d(\mathbf{N_2}, \Gamma_k(g_\iota(T, x)) \\ &\leq 2^{-k} + k^{-1} \leq 1^{-(k-1)} \leq 3/8.\end{aligned}$$

This contradicts the fact that $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$ satisfies assumption $a)$ in proposition 4.13 with $\kappa = 3/8$, which was established in part (i) using proposition 5.1. Thus we can conclude that $\hat{\Gamma}(T, x) = 1$.

$\underline{\hat{\Xi}(T, x) = 0 \implies \hat{\Gamma}(T, x) = 0}$: Since $\hat{\Xi}(T, x) = 0$, T does not halt on input x. Thus $g_\iota(T, x)$ is equal to the sequence $\{\iota_n^1\}_{n\in\mathbb{N}}$ and therefore, by assumption, $\Gamma_k(g_\iota(T, x)) \in \mathcal{N}_{2^{-k}}(\Xi(\iota^0)) \subseteq \mathcal{N}_{k^{-1}}(S^2)$, since $\{\iota_n^1\}_{n\in\mathbb{N}}$ is an inexact representation of ι_0 in the sense of eq. (4.5). Thus, we can conclude that $\hat{\Gamma}(T, x) = 0$. This finishes the proof. □

5.3. **Formal statement and proof of Theorem 2.1.** In the language of the SCI hierarchy Theorem 2.1 has the following formal form.

Theorem 5.4. *For any integers $N > m$ and any $\beta_{max} \geq \beta_{min} > 0$, consider any fixed non-zero linear map $A : \mathbb{R}^N \to \mathbb{R}^m$ such that the spectrum $\mathrm{Sp}(AA^*) \subset [\beta_{min}^2, \beta_{max}^2]$. Then, for any rational $\epsilon_1 \in (0, 3/8]$ and any integer $\ell \geq 2$, there exists a collection of initial domains $\mathcal{M}_1 \subset \mathbb{R}^N$, a domain Ω (as described in §1.1) of training sets $\mathcal{T}$ with $|\mathcal{T}| = \ell$, $\mathcal{T} \subseteq B_1(0)$, a $D \in \mathbb{N}$ and a mapping $\Xi_D : \Omega \rightrightarrows \mathcal{NN}_{m,N}$ as in (1.7). In particular, all optimal neural networks $\mathbf{N}_{opt}^{\mathcal{M}_1} \in \Xi_D(\mathcal{T}) \neq \emptyset$ have uniformly bounded (by D) Jacobians. However, the following happens simultaneously:*

- *(i) For the computational problem $\{\Xi, \Omega, \mathcal{M}, \Lambda\}^{\Delta_1}$, with Ξ as in (1.6), we have $\epsilon_{\mathbb{PB}}^{\mathrm{s}}(\mathrm{p}) \geq \epsilon_1$ for $\mathrm{p} \in [0, 1/2)$. In particular, no algorithm, not even randomized, can approximate the optimal neural network $\mathbf{N}_{opt}^{\mathcal{M}_1}$ for all inputs $\mathcal{T} \in \Omega$ to accuracy ϵ_1 (with probability greater than $p > 1/2$ in the randomized case).*
- *(ii) However, for any $K \in \mathbb{N}$, $\delta \in (0, \epsilon_1)$ and any algorithm Γ such that $\mathbf{N}_{\mathcal{T},\epsilon} = \Gamma(\mathcal{T}, \epsilon)$ is a NN that approximates an optimal neural network $\mathbf{N}_{opt}^{\mathcal{M}_1} \in \Xi_D(\mathcal{T})$ to accuracy $\epsilon \in (\epsilon_1, 2\epsilon_1 - \delta]$ for all $\mathcal{T} \in \Omega$ we have the following: There exists infinitely many $\mathcal{T} \in \Omega$ such that the Jacobian satisfies $\|\mathrm{D}\mathbf{N}_{\mathcal{T},\epsilon}(c)\|_{op} \geq K$ for some $c \in \mathbb{R}^m$. In particular, the Lipchitz constant of the NN will blow up.*
- *(iii) There does exists an algorithm Γ such that $\mathbf{N}_{\mathcal{T},\epsilon} = \Gamma(\mathcal{T}, \epsilon)$ is a NN that approximates the optimal neural network $\mathbf{N}_{opt}^{\mathcal{M}_1} = \Xi(\mathcal{T})$ to accuracy $\epsilon > 2\epsilon_1$ for all $\mathcal{T} \in \Omega$ with the property that there exists a $K \in \mathbb{N}$ such that the Jacobian satisfies $\|\mathrm{D}\mathbf{N}_{\mathcal{T},\epsilon}(c)\|_{op} \leq K$ for all $c \in \mathbb{R}^m$ and for all $\mathcal{T} \in \Omega$.*

Proof of Theorem 5.4. **Step I:** Constructing the computational problems. Our aim is to construct a domain $\Omega \subseteq T_\ell^1$, where T_ℓ^1 is as defined in eq. (4.6), of the form $\Omega = \{\iota_n^1\}_{n\in\mathbb{N}} \cup \{\iota_n^2\}_{n\in\mathbb{N}}$, where $\{\iota_n^1\}_{n\in\mathbb{N}}$ and $\{\iota_n^2\}_{n\in\mathbb{N}}$ are sequences of training sets that satisfy assumptions $a) - c)$ in proposition 4.13. Indeed, we use a very similar construction as in the proof of Proposition 5.1, but with slight modifications. Let $\mathcal{T}_b \subseteq T_{\ell-2}^1$ be such that $\mathcal{T}_b \subseteq \ker(A)^\perp \times A(\ker(A)^\perp)$, and such that the points in $\pi_1(\mathcal{T}_b)$ are non-zero and distinct, where $\pi_1(\mathcal{T}_b) = \{x \,:\, (x, y) \in \mathbb{R}^N \times \mathbb{R}^m,\ (x, y) \in \mathcal{T}_b\}$. Further, let $v \in \ker(A)$ be a fixed element such that $\|v\|_{\ell_2} = 4\epsilon_1$, and let $e \in \ker(A)^\perp$ be some fixed unit vector. At last, we require that

$$B_{1/4}^+(v) \nsubseteq \mathcal{T}_b, \quad \text{where } B_{1/4}^+(v) = \{z \in \mathbb{R}^N \mid v \leq_\ell z \leq_\ell v + 1/4v\}$$

is the potentially empty set where $\leq_\ell$ denotes the lexicographic ordering, as defined in the proof of proposition 5.1. We observe that, even with these restrictions, there exists infinitely many different choices for $\mathcal{T}_b$. We define the sequences $\{\iota_n^1\}_{n\in\mathbb{N}}$ and $\{\iota_n^2\}_{n\in\mathbb{N}}$ as follows: for each $n \in \mathbb{N}$ let

$$\iota_n^1 = \mathcal{T}_b \bigcup \{(0,0), (\frac{\theta}{4^n}e + v, A(\frac{\theta}{4^n}e + v))\} \bigcup_{i=1}^N \{(\epsilon_1 e_i, A(\epsilon_1 e_i))\}$$

and

$$\iota_n^2 = \mathcal{T}_b \bigcup \{(0,0), (v,0)\} \bigcup_{i=1}^N \{(\epsilon_1 e_i, A(\epsilon_1 e_i))\},$$

where $\theta = \|A\|_{op}^{-1}$ if $\|A\|_{op} > 1$ and $\theta = 1$ otherwise, and where e_i are the standard unit vectors in $\mathbb{R}^N$ for $i = 1, \ldots, N$. It is not hard to check that $\iota_n^1, \iota_n^2 \subseteq T_\ell^1$ for all $n \in \mathbb{N}$. We define the corresponding sets $\mathcal{M}_{1,n}^1$ and $\mathcal{M}_{1,n}^2$ as follows: Let

$$\mathcal{M}_{1,n}^1 = \pi_1(\mathcal{T}_b) \cup \{0, \frac{\theta}{4^n}e + v\}, \quad \mathcal{M}_{1,n}^2 = \pi_1(\mathcal{T}_b) \cup \{0, v\}, \quad n \in \mathbb{N}.$$

Due to the fact that the underlying metric spaces $\mathcal{M}_{1,n}^1$ and $\mathcal{M}_{1,n}^2$ are the exactly same as in the proof of Proposition 5.1, we can prove that there exists an optimal neural network for each inverse problem $(A, \mathcal{M}_{1,n}^1)$ and $(A, \mathcal{M}_{1,n}^2)$ for each $n \in \mathbb{N}$ – with uniformly bounded Jacobians – in exactly the same way as in the proof of Proposition 5.1. We omit the details.

Step II: Proving (i). By exactly the same argumentation as in points $a)-c)$ in the proof of proposition 5.1, we can prove that $\{\iota_n^1\}_{n\in\mathbb{N}}$ and $\{\iota_n^2\}_{n\in\mathbb{N}}$ satisfy assumptions $a)-c)$ in proposition 4.13 with $\kappa = 2\epsilon_1$. Part (i) follows from this.

Step III: Proving (ii). Choose $K \in \mathbb{N}$ and $\delta \in (0, \epsilon_1)$ and suppose there is an algorithm Γ such that $\mathbf{N}_{\mathcal{T},\epsilon} = \Gamma(\mathcal{T}, \epsilon)$ is a NN that approximates any optimal neural network $\mathbf{N}_{opt}^{\mathcal{M}_1} \in \Xi(\mathcal{T})$ to accuracy $\epsilon \in (\epsilon_1, 2\epsilon_1 - \delta]$ for all $\mathcal{T} \in \Omega$. We pick an $\hat{n} \in \mathbb{N}$ such that $K \leq 2\delta 4^{\hat{n}}$. Then for any $n \geq \hat{n}$ choose $\mathcal{T} = \{(0,0), (\frac{\theta}{4^n} + v, A(\frac{\theta}{4^n} + v)\} \cup \mathcal{T}_b$ – note that it is clear that we have infinitely many such choices of $\mathcal{T}$. Now consider

$$\begin{aligned}
&\|\mathbf{N}_{\mathcal{T},\epsilon}(0) - \mathbf{N}_{\mathcal{T},\epsilon}(A(\frac{\theta}{4^n}e))\|_{\ell_2} \\
&= \|\mathbf{N}_{\mathcal{T},\epsilon}(0) - \mathbf{N}_{opt}^{\mathcal{M}_1}(0) + \mathbf{N}_{opt}^{\mathcal{M}_1}(0) - \mathbf{N}_{opt}^{\mathcal{M}_1}(A(\frac{\theta}{4^n}e)) + \mathbf{N}_{opt}^{\mathcal{M}_1}(A(\frac{\theta}{4^n}e)) - \mathbf{N}_{\mathcal{T},\epsilon}(A(\frac{\theta}{4^n}e))\|_{\ell_2} \\
&\geq \|\mathbf{N}_{opt}^{\mathcal{M}_1}(0) - \mathbf{N}_{opt}^{\mathcal{M}_1}(A(\frac{\theta}{4^n}e))\|_{\ell_2} - \|\mathbf{N}_{\mathcal{T},\epsilon}(0) - \mathbf{N}_{opt}^{\mathcal{M}_1}(0)\|_{\ell_2} \\
&\quad - \|\mathbf{N}_{\mathcal{T},\epsilon}(A(\frac{\theta}{4^n}e)) - \mathbf{N}_{opt}^{\mathcal{M}_1}(A(\frac{\theta}{4^n}e))\|_{\ell_2} \geq \|\mathbf{N}_{opt}^{\mathcal{M}_1}(0) - \mathbf{N}_{opt}^{\mathcal{M}_1}(A(\frac{\theta}{4^n}e))\|_{\ell_2} - 4\epsilon_1 + 2\delta,
\end{aligned} \tag{5.23}$$

where the last inequality follows from the assumption that $\mathbf{N}_{\mathcal{T},\epsilon} = \Gamma(\mathcal{T}, \epsilon)$ is a NN that approximates the optimal neural network $\mathbf{N}_{opt}^{\mathcal{M}_1} \in \Xi(\mathcal{T})$ to accuracy $\epsilon \in (\epsilon_1, 2\epsilon_1 - \delta]$. By the mean value inequality [1, Proposition 2.4.8] it follows that there exists a $c \in \mathbb{R}^m$ such that

$$\|\mathrm{D}\mathbf{N}_{\mathcal{T},\epsilon}(c)\|_{op}\|A(\frac{\theta}{4^n}e)\|_{\ell_2} \geq \|\mathbf{N}_{\mathcal{T},\epsilon}(0) - \mathbf{N}_{\mathcal{T},\epsilon}(A(\frac{\theta}{4^n}e))\|_{\ell_2}. \tag{5.24}$$

Moreover, from the proof of Proposition 5.1, we know that $\mathbf{N}_{opt}^{\mathcal{M}_1}(0) = 0$ and that $\mathbf{N}_{opt}^{\mathcal{M}_1}(A(\frac{\theta}{4^n}e)) = \frac{\theta}{4^n}e + v$, which gives us that $\|\mathbf{N}_{opt}^{\mathcal{M}_1}(0) - \mathbf{N}_{opt}^{\mathcal{M}_1}(A(\frac{\theta}{4^n}e))\|_{\ell_2}^2 = \|\frac{\theta}{4^n}e + v\|_{\ell_2}^2 = \frac{\theta}{4^n}\|e\|_{\ell_2}^2 + \|v\|_{\ell_2}^2 = \frac{\theta}{4^n} + \|v\|_{\ell_2}^2 > \|v\|_{\ell_2}^2$. Combing this with our previous calculations in (5.23) and (5.24) we get that

$$\begin{aligned}
\|\mathrm{D}\mathbf{N}_{\mathcal{T},\epsilon}(c)\|_{op}\|A(\frac{\theta}{4^n}e)\|_{\ell_2} &\geq \|\mathbf{N}_{\mathcal{T},\epsilon}(0) - \mathbf{N}_{\mathcal{T},\epsilon}(A(\frac{\theta}{4^n}e))\|_{\ell_2} \\
&\geq \|\mathbf{N}_{opt}^{\mathcal{M}_1}(0) - \mathbf{N}_{opt}^{\mathcal{M}_1}(A(\frac{\theta}{4^n}e))\|_{\ell_2} - 4\epsilon_1 + 2\delta \\
&> \|v\|_{\ell_2} - 4\epsilon_1 + 2\delta = 2\delta,
\end{aligned}$$

and by our choice of $\hat{n}$ and the fact that $n \geq \hat{n}$ we can conclude that $\|\mathrm{D}\mathbf{N}_{\mathcal{T},\epsilon}(c)\|_{op} > 2\delta 4^n \geq 2\delta 4^{\hat{n}} \geq K$.

Step IV: In order to prove part (iii) it remains to show that there exists an algorithm $\Gamma : \tilde{\Omega} \times \mathbb{R}_+ \to \mathcal{NN}_{m,N}$ that works on inexact input, such that for each $\tilde{\mathcal{T}} \in \tilde{\Omega}$, Γ constructs a neural network $\Gamma(\tilde{\mathcal{T}}, \epsilon)$ such that

$$\sup_{y \in \bigcup_{\mathcal{F}} \mathcal{M}_2} \|\Gamma(\tilde{\mathcal{T}}, \epsilon)(y) - x\|_{\ell_2} \leq \epsilon, \tag{5.25}$$

for any $\epsilon > 2\epsilon_1$. Our strategy is to construct an algorithm that approximates the pseudo inverse $A^\dagger$ of A. In order to achieve this we first need to obtain an approximation of A, which we can do by running the following algorithm for any $\tilde{\mathcal{T}} \in \tilde{\Omega}$:

Inputs: Oracles for all $(\tilde{x}, \tilde{y}) \in \tilde{\mathcal{T}}$, the dimension $N \in \mathbb{N}$, and $k \in \mathbb{N}$.
Outputs: An approximation A_k of A such that $\|A_k - A\|_{op} \leq 2^{-k}$.

1. Identifying the elements $(\tilde{x}, \tilde{y}) \in \tilde{\mathcal{T}}$ that are inexact representations of $(\epsilon_1 e_i, A(\epsilon_1 e_i)) \in \mathcal{B}$ for $i = 1, \ldots, N$.
   Choose k' such that $2^{-k'} \leq \frac{\epsilon_1}{N} \cdot 2^{-k}$ and run the following pseudo code:
   for $i = 1, \ldots, N$:
   for $\tilde{x} \in \pi_1(\tilde{\mathcal{T}})$:
   if $\|\epsilon_1 e_i - x_2\|_{\ell_2} \leq 2^{-2} = 1/4$:

(Then we have found the pair $(\tilde{x}, \tilde{y})$ that approximates $(\epsilon_1 e_i, A(\epsilon_1 e_i))$).
Set $A_k^i = \frac{1}{\epsilon_1} \cdot \tilde{y}^{k'}$ (where $\tilde{y}^{k'}$ is as defined in eq. (5.13))
break.

2. Use the elements in $\tilde{\mathcal{B}}$ that we identified in step 1 to construct the desired approximation of A.
Set $A_k = (A_k^1, \dots, A_k^N)$ and return A_k.

We notice that A_k yields the requested approximation since

$$\begin{aligned}\|A_k - A\|_{op} &\leq \sum_{i=1}^{N} \|A_k^i - A_i\|_{\ell_2} = \sum_{i=1}^{N} \|\frac{1}{\epsilon_1}\tilde{y}_i^{k'} - A(e_i)\|_{\ell_2} \\ &\leq \sum_{i=1}^{N} \|\frac{1}{\epsilon_1}\tilde{y}_i^{k'} - \frac{1}{\epsilon_1}A(\epsilon_1 e_i)\|_{\ell_2} \leq \sum_{i=1}^{N} \frac{1}{\epsilon_1} \cdot 2^{-k'} \leq N \cdot \frac{1}{\epsilon_1} \cdot \frac{\epsilon_1}{N} \cdot 2^{-k} = 2^{-k}\end{aligned}$$

since $\tilde{y}_i^{k'}$ is such that $\|\tilde{y}_i^{k'} - y_i\|_{\ell_2} = \|\tilde{y}_i^{k'} - A(\epsilon_1 e_i)\|_{\ell_2} \leq 2^{-k'} \leq \frac{\epsilon_1}{N} \cdot 2^{-k}$ for all $i = 1, \dots, N$.

Next, we define $h : \mathbb{N} \to \mathbb{N}$ by

$$h(j) = \min\{\mu \in \mathbb{N} \,|\, \forall r \geq \mu\,, \sup_{y \in \bigcup_{\mathcal{F}} \mathcal{M}_2} \|A_r^\dagger(y) - A^\dagger(y)\|_{\ell_2} \leq 2^{-j}\}. \tag{5.26}$$

We will now estimate an upper bound for h by finding a lower bound for 2^{-h} that can be recursively computed. By the assumption that A is full rank we have that $A^\dagger$ can be expressed as $A^\dagger = A^T(AA^T)^{-1}$.
Claim: We claim that if $r \in \mathbb{N}$ is such that $2^{-r} \leq \frac{1}{2}\frac{1}{4}(1 + \beta_{max})^{-2}\alpha^4 2^{-j}$, with $\alpha = \min\{1, \beta_{min}\}$, then 2^{-r} is a lower bound for $2^{-h(j)}$.
We now proceed with verifying this claim. Indeed, let $r \in \mathbb{N}$ be as described in the claim above, and such that A_r is of full rank. We notice that it is always possible to find such an r, since the smallest singular value of A is bounded strictly away from 0. Moreover, $A_r^\dagger = A_r^T(A_r A_r^T)^{-1}$ can be recursively computed since both matrix multiplication and calculating the inverse of a matrix can be done recursively. We now have that

$$\begin{aligned}\|A_r^\dagger - A^\dagger\|_{op} &= \|A_r^T(A_r A_r^T)^{-1} - A^T(AA^T)^{-1}\|_{op} \\ &= \|A_r^T(A_r A_r^T)^{-1} + A_r^T(AA^T)^{-1} - A_r^T(AA^T)^{-1} - A^T(AA^T)^{-1}\|_{op} \\ &\leq \|A_r^T\|_{op}\|(A_r A_r^T)^{-1} - (AA^T)^{-1}\|_{op} + \|A_r^T - A^T\|_{op}\|(AA^T)^{-1}\|_{op} \\ &\leq (\beta_{max} + 2^{-r})\|(A_r A_r^T)^{-1} - (AA^T)^{-1}\|_{op} + \|A_r - A\|_{op}\frac{1}{\beta_{min}^2} \\ &\leq (\beta_{max} + 1)\|(A_r A_r^T)^{-1} - (AA^T)^{-1}\|_{op} + \frac{1}{4}2^{-j}.\end{aligned} \tag{5.27}$$

We continue our calculations by considering the term $\|(A_r A_r^T)^{-1} - (AA^T)^{-1}\|_{op}$. By the second resolvent identity we get that

$$\|(A_r A_r^T)^{-1} - (AA^T)^{-1}\|_{op} \leq \|(A_r A_r^T)^{-1}\|_{op}\|A_r A_r^T - AA^T\|_{op}\|(AA^T)^{-1}\|_{op}, \tag{5.28}$$

and by considering the term $\|(A_r A_r^T)^{-1}\|_{op}$ we observe that

$$\begin{aligned}\|(A_r A_r^T)^{-1}\|_{op} &\leq \|(A_r A_r^T)^{-1} - (AA^T)^{-1}\|_{op} + \|(AA^T)^{-1}\|_{op} \\ &= \|(A_r A_r^T)^{-1}\|_{op}\|A_r A_r^T - AA^T\|_{op}\|(AA^T)^{-1}\|_{op} + \|(AA^T)^{-1}\|_{op},\end{aligned}$$

where the second equality again follows by the second resolvent identity. This implies that

$$\|(A_r A_r^T)^{-1}\|_{op}(1 - \|(AA^T)^{-1}\|_{op}\|A_r A_r^T - AA^T\|_{op}) \leq \|(AA^T)^{-1}\|_{op}. \tag{5.29}$$

Next, we claim that $\|(AA^T)^{-1}\|_{op}\|A_rA_r^T - AA^T\|_{op} < 2^{-j}$. Indeed, with $\mathcal{E}_r = A_r - A$, we see that

$$\begin{aligned}\|(AA^T)^{-1}\|_{op}\|A_rA_r^T - AA^T\|_{op} &= \frac{1}{\beta_{min}^2}\|(A+\mathcal{E}_r)(A+\mathcal{E}_r)^T - AA^T\|_{op} \\ &\leq \frac{1}{\alpha^2}\|AA^T + A\mathcal{E}_r^T + \mathcal{E}_rA^T + \mathcal{E}_r\mathcal{E}_r^T - AA^T\|_{op} \\ &\leq \frac{1}{\alpha^2}(\|A\mathcal{E}_r^T\|_{op} + \|\mathcal{E}_rA^T\|_{op} + \|\mathcal{E}_r\|_{op}^2) \\ &\leq \frac{1}{\alpha^2}(\beta_{max}2^{-r} + 2^{-r}\beta_{max} + 2\cdot 2^{-r}) \\ &= \frac{1}{\alpha^2}\cdot 2\cdot 2^{-r}(\beta_{max}+1) \leq \frac{1}{4}\alpha^2(\beta_{max}+1)^{-1}2^{-j} < 2^{-j},\end{aligned} \tag{5.30}$$

by our choice of 2^{-r}. Combining this with eq. (5.29) and that $j \geq 1$ we see that

$$\|(A_rA_r)^{-1}\|_{op} \leq \|(AA^T)^{-1}\|_{op}/(1-2^{-j}) \leq 2\|(AA^T)^{-1}\|_{op}. \tag{5.31}$$

Hence, it follows from eq. (5.28), eq. (5.30) and eq. (5.31) that

$$\begin{aligned}\|(A_rA_r^T)^{-1} - (AA^T)^{-1}\|_{op} &\leq 2\|(AA^T)^{-1}\|_{op}^2\|A_rA_r^T - AA^T\|_{op} \\ &\leq 2\frac{1}{\beta_{min}^4}\|A_rA_r^T - AA^T\|_{op} \leq 2\frac{1}{\beta_{min}^4}2\cdot 2^{-r}(\beta_{max}+1).\end{aligned}$$

finally, by inserting this back into eq. (5.27), we conclude that

$$\begin{aligned}\|A_r^\dagger - A^\dagger\|_{op} &\leq 2(\beta_{max}+1)^2\frac{1}{\beta_{min}^4}\cdot 2\cdot 2^{-r} + \frac{1}{\beta_{min}^2}\|A_r - A\|_{op} \\ &\leq (\beta_{max}+1)^2\frac{1}{\beta_{min}^4}\cdot 4\cdot 2^{-r} + \frac{1}{\beta_{min}^2}2^{-r} \leq \frac{1}{2}2^{-j} + \frac{1}{2}2^{-j} = 2^{-j},\end{aligned}$$

by our choice of 2^{-r}. At last, since $\|y\|_{\ell_2} \leq 1$ for all $y \in \bigcup_{\mathcal{F}}\mathcal{M}_2$ we get that

$$\sup_{y\in\bigcup_{\mathcal{F}}\mathcal{M}_2}\|A_r^\dagger(y) - A^\dagger(y)\|_{\ell_2} \leq \|A_r^\dagger - A^\dagger\|_{op} \leq 2^{-j},$$

which was what we wished to prove.

We are finally ready to present our algorithm $\Gamma : \tilde{\Omega}\times\mathbb{R}_+ \to \mathcal{NN}_{m,N}$. Let $\epsilon_2 > 0$ be rational and such that $2\epsilon_1 + \epsilon_2 \leq \epsilon$, and note that such an ϵ_2 always exists because ϵ is strictly bigger then $2\epsilon_1$. Then, for any $\tilde{\mathcal{T}} \in \tilde{\Omega}$, $\Gamma(\tilde{\mathcal{T}},\epsilon)(y)$ does the following: It applies $A_r^\dagger$ to y and adds the bias term $\frac{1}{2}v$, where r is chosen such that $\|A_r^\dagger - A^\dagger\|_{op} \leq \epsilon_2$. We notice that we can compute such an r recursively, by computing a $j \in \mathbb{N}$ such that $2^{-j} \leq \epsilon_2$ and then computing an r such that $2^{-r} \leq \frac{1}{2}\frac{1}{4}(1+\beta_{max})^{-2}\alpha^4 2^{-j}$ with $\alpha = \min\{1,\beta_{min}\}$ (then r is an upper bound for $h(j)$ in eq. (5.26)). More precisely, let

$$\Gamma(\tilde{\mathcal{T}},\epsilon)(y) = A_r^\dagger(y) + \frac{1}{2}v,$$

where r satisfies $2^{-r} \leq \frac{1}{2}\frac{1}{4}(1+\beta_{max})^{-2}\alpha^4 2^{-j}$, with $\alpha = \min\{1,\beta_{min}\}$ and $2^{-j} \leq \epsilon_2$, and where $v \in \ker(A)$ with $\|v\|_{\ell_2} = 4\epsilon_1$. It only remains to show that

$$\sup_{y\in\bigcup_{\mathcal{F}}\mathcal{M}_2}\|\Gamma(\tilde{\mathcal{T}},\epsilon)(y) - \mathbf{N}_{opt}^{\mathcal{M}_1}(y)\|_{\ell_2} \leq \epsilon$$

for all $\tilde{\mathcal{T}} \in \tilde{\Omega}$, and that $\|\mathrm{D}\Gamma(\tilde{\mathcal{T}},\epsilon)(c)\|_{op} \leq K$ for all $c \in \bigcup_{\mathcal{F}}\mathcal{M}_2$ and some $K \in \mathbb{N}$. Indeed, we start by noticing that

$$\begin{aligned}\sup_{y\in\bigcup_{\mathcal{F}}\mathcal{M}_2}\|\Gamma(\tilde{\mathcal{T}},\epsilon)(y) - \mathbf{N}_{opt}^{\mathcal{M}_1}(y)\|_{\ell_2} &= \sup_y\|A_r^\dagger(y) + \frac{1}{2}v - \mathbf{N}_{opt}^{\mathcal{M}_1}(y)\|_{\ell_2} \\ &\leq \sup_y\|A^\dagger(y) + \frac{1}{2}v - \mathbf{N}_{opt}^{\mathcal{M}_1}(y)\| + \|A_r^\dagger - A^\dagger\|_{op},\end{aligned}$$

where the last inequality follows since $A_r(y) = A(y) + A_r(y) - A(y)$ and $\|y\|_{\ell_2} \leq 1$. Now, we consider the term $\|A^\dagger(y) + \frac{1}{2}v - \mathbf{N}_{opt}(y)\|_{\ell_2}$ and show that it must always be less than $2\epsilon_1$. Indeed, we notice

that if $y \in A(\mathcal{M}_n^1)$ for some $n \in \mathbb{N}$, then either $\mathbf{N}_{opt}(y) = x$ for some $x \in \ker(A)^\perp$ (in the case where $y \in \pi_2(\mathcal{T}_b) \cup \{0\}$), or $\mathbf{N}_{opt}(y) = \frac{\theta}{4^n}e + v$ (in the case where $y = A(\frac{\theta}{4^n}e + v)$). In the first case we get that

$$\|A^\dagger(y) + \frac{1}{2}v - \mathbf{N}_{opt}^{\mathcal{M}_1}(y)\|_{\ell_2} = \|x + \frac{1}{2}v - x\|_{\ell_2} = \frac{1}{2}\|v\|_{\ell_2} = \frac{1}{2}4\epsilon_1 = 2\epsilon_1,$$

and in the second case we get that

$$\|A^\dagger(y) + \frac{1}{2}v - \mathbf{N}_{opt}^{\mathcal{M}_1}(y)\|_{\ell_2} = \|x + \frac{1}{2}v - x - v\|_{\ell_2} = \frac{1}{2}\|v\|_{\ell_2} = \frac{1}{2}4\epsilon_1 = 2\epsilon_1.$$

On the other hand, if $y \in A(\mathcal{M}^2)$ we either have that $\mathbf{N}_{opt}^{\mathcal{M}_1}(y) = x$ for some $x \in \ker(A)^\perp$ (in the case where $y \in \pi_2(\mathcal{T}_b)$), or $\mathbf{N}_{opt}^{\mathcal{M}_1}(y) = \frac{1}{2}v$ (in the case where $y = 0$ or in the case where $y = v$). In the first case we get the exact same bound as above, while in the second case we get that

$$\|A^\dagger(y) + \frac{1}{2}v - \mathbf{N}_{opt}^{\mathcal{M}_1}(y)\|_{\ell_2} = \|\frac{1}{2}v - \frac{1}{2}v\|_{\ell_2} = 0.$$

Using this, and the fact that $\|A_r^\dagger - A^\dagger\|_{op} \leq 2^{-j} \leq \epsilon_2$, this we arrive at the conclusion that

$$\sup_{y \in \bigcup_{\mathcal{F}} \mathcal{M}_2} \|\Gamma(\tilde{\mathcal{T}}, \epsilon)(y) - \mathbf{N}_{opt}^{\mathcal{M}_1}(y)\|_{\ell_2} \leq \sup_{(x,y)} \|A^\dagger(y) + \frac{1}{2}v - \mathbf{N}_{opt}^{\mathcal{M}_1}(y)\| + \|A_r^\dagger - A^\dagger\|_{op} \leq 2\epsilon_1 + \epsilon_2 \leq \epsilon,$$

Moreover, since $\Gamma(\tilde{\mathcal{T}}, \epsilon)$ is a neural network, we can conclude that Γ approximates $\mathbf{N}_{opt}^{\mathcal{M}_1}$ to accuracy ϵ for all $\tilde{\mathcal{T}} \in \tilde{\Omega}$. At last, we have that for $\mathbf{N}_{\mathcal{T},\epsilon} = \Gamma(\tilde{\mathcal{T}}, \epsilon)$, it immediately follows that

$$\|\mathrm{D}\mathbf{N}_{\mathcal{T},\epsilon}(c)\|_{op} = \|A_r^\dagger\|_{op} \leq \|A^\dagger\|_{op} + 2^{-j} \leq \frac{1}{\beta_{min}} + \epsilon_2, \quad \forall\, c \in \bigcup_{\mathcal{F}} \mathcal{M}_2.$$

This establishes part (iii). □

REFERENCES

[1] R. Abraham, J. Marsden, and T. Ratiu. *Manifolds, Tensor Analysis, and Applications*. Applied Mathematical Sciences. Springer New York, 2012.

[2] 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.

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

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

[5] 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.

[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. *Proc. Natl. Acad. Sci. USA*, 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. 45(1), 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] A. Bastounis, B. Adcock, and A. C. Hansen. From global to local: Getting more from compressed sensing. *SIAM News*, 50(8):1–4, 2017.

[12] 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.

[13] A. Bastounis, A. C. Hansen, and V. Vlačić. The extended Smale's 9th problem – On computational barriers and paradoxes in estimation, regularisation, computer-assisted proofs and learning. *arXiv:2110.15734*, 2021.

[14] D. Bayer and J. C. Lagarias. The nonlinear geometry of linear programming. ii. legendre transform coordinates and central trajectories. *Transactions of the American Mathematical Society*, 314:527–581, 1989.

[15] S. Becker and A. Hansen. Computing solutions of schroedinger equations on unbounded domains- on the brink of numerical algorithms, 2020.

[16] M. Bellare, O. Goldreich, and M. Sudan. Free bits, pcps, and nonapproximability – towards tight results. *SIAM J. Comput.*, 27(3):804–915, 1998.

[17] 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:1508.03280*, 2020.

[18] 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.

[19] J. Ben-Artzi, M. Marletta, and F. Rösler. Computing the sound of the sea in a seashell. *Foundations of Computational Mathematics*, pages 1–35, 2021.

[20] J. Ben-Artzi, M. Marletta, and F. Rösler. Computing scattering resonances. *Journal of the European Mathematical Society*, (to appear).

[21] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski. *Robust Optimization*. Princeton Series in Applied Mathematics. Princeton University Press, October 2009.

[22] A. Ben-Tal and A. Nemirovski. Robust solutions of linear programming problems contaminated with uncertain data. *Mathematical Programming*, 88(3):411–424, 2000.

[23] 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.

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

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

[26] 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.

[27] H. Bölcskei, P. Grohs, G. Kutyniok, and P. Petersen. Optimal approximation with sparsely connected deep neural networks. *SIAM Journal on Mathematics of Data Science*, 1(1):8–45, 2019.

[28] 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.

[29] M. Braverman and S. Cook. Computing over the reals: Foundations for scientific computing. *Notices of the American Mathematical Society*, 53(3):318–329, 2006.

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

[31] 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.

[32] E. J. Candes, T. Strohmer, and V. Voroninski. Phaselift: Exact and stable signal recovery from magnitude measurements via convex programming. *Communications on Pure and Applied Mathematics*, 66(8):1241–1274, 2013.

[33] A. Chambolle. An algorithm for total variation minimization and applications. *Journal of Mathematical Imaging and Vision*, 20(1):89–97, 2004.

[34] A. Chambolle and T. Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145, May 2011.

[35] C. Choi. 7 revealing ways AIs fail. *IEEE Spectrum*, September, 2021.

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

[37] 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.

[38] M. Colbrook and A. C. Hansen. The foundations of spectral computations via the solvability complexity index hierarchy. *Journal of the European Mathematical Society*, (to appear).

[39] M. J. Colbrook. Computing spectral measures and spectral types. *Communications in Mathematical Physics*, 384(1):433–501, 2021.

[40] 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.

[41] M. J. Colbrook, A. Horning, and A. Townsend. Computing spectral measures of self-adjoint operators. *SIAM Review*, 63(3):489–524, 2021.

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

[43] R. DeVore, B. Hanin, and G. Petrova. Neural network approximation. *Acta Numerica*, 30:327–444, 2021.

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

[45] P. Doyle and C. McMullen. Solving the quintic by iteration. *Acta Mathematica*, 163(3-4):151–180, 1989.

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

[47] G. F. Fasshauer. *Meshfree Approximation Methods with MATLAB*. World Scientific Publishing Co., Inc., USA, 2007.

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

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

[50] 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.

[51] 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.

[52] M. Genzel, J. Macdonald, and M. Marz. Solving inverse problems with deep neural networks - robustness included. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2022.

[53] K. Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Monatshefte für mathematik und physik*, 38(1):173–198, 1931.

[54] N. M. Gottschling, V. Antun, B. Adcock, and A. C. Hansen. The troublesome kernel: why deep learning for inverse problems is typically unstable. *arXiv:2001.01258*, 2020.

[55] 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.

[56] 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.

[57] A. C. Hansen and O. Nevanlinna. Complexity issues in computing spectra, pseudospectra and resolvents. *Banach Center Publications*, 112:171–194, 2016.

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

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

[60] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

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

[62] D. P. Hoffman, I. Slavitt, and C. A. Fitzpatrick. The promise and peril of deep learning in microscopy. *Nature Methods*, 18(2):131–132, 2021.

[63] 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.

[64] 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.

[65] 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.

[66] K. Ko. *Complexity Theory of Real Functions*. 1991.

[67] 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.

[68] 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.

[69] C. McMullen. Families of rational maps and iterative root-finding algorithms. *Annals of Mathematics*, 125(3):467–493, 1987.

[70] C. McMullen. Braiding of the attractor and the failure of iterative algorithms. *Inventiones Mathematicae*, 91(2):259–272, 1988.

[71] 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.

[72] Y. E. Nesterov and A. Nemirovski. On first-order algorithms for l1/nuclear norm minimization. *Acta Numerica*, 22:509–575, 2013.

[73] P. Niyogi, S. Smale, and S. Weinberger. A topological view of unsupervised learning from noisy data. *SIAM Journal on Computing*, 40(3):646–663, 2011.

[74] 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.

[75] P. Petersen and F. Voigtlaender. Optimal approximation of piecewise smooth functions using deep relu neural networks. *Neural Networks*, 108:296–330, 2018.

[76] A. Pinkus. Approximation theory of the mlp model in neural networks. *Acta Numerica*, 8, 1999.

[77] F. Preparata and M. Shamos. *Computational Geometry: An Introduction*. Monographs in Computer Science. Springer New York, 2012.

[78] J. Renegar. A polynomial-time algorithm, based on newton's method, for linear programming. *Mathematical Programming*, 40(1-3):59–93, 1988.

[79] J. Renegar. Incorporating condition measures into the complexity theory of linear programming. *SIAM Journal on Optimization*, 5(3):506–524, 1995.

[80] J. Renegar. Linear programming, complexity theory and elementary functional analysis. *Mathematical Programming*, 70(1):279–351, 1995.

[81] A. Salomaa, C. U. Press, G. Rota, B. Doran, T. Lam, P. Flajolet, M. Ismail, and E. Lutwak. *Computation and Automata*. EBL-Schweitzer. Cambridge University Press, 1985.

[82] A. Sinha, J. Lee, S. Li, and G. Barbastathis. Solving inverse problems using residual neural networks. page W1A.3, 01 2016.

[83] S. Smale. The fundamental theorem of algebra and complexity theory. *Bulletin of the American Mathematical Society*, 4(1):1–36, 1981.

[84] S. Smale. On the efficiency of algorithms of analysis. *Bulletin of the American Mathematical Society*, 13(2):87–121, 1985.

[85] S. Smale. Complexity theory and numerical analysis. In *Acta numerica, 1997*, volume 6 of *Acta Numer.*, pages 523–551. Cambridge Univ. Press, Cambridge, 1997.

[86] S. Smale. Mathematical problems for the next century. *Mathematical Intelligencer*, 20:7–15, 1998.

[87] S. Smale. The work of Curtis T McMullen. In *Proceedings of the International Congress of Mathematicians I, Berlin*, Doc. Math. J. DMV, pages 127–132. 1998.

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

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

[90] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1996.

[91] A. M. Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, S2-42(1):230, 1936.

[92] A. M. Turing. I.-Computing machinery and intelligence. *Mind*, LIX(236):433–460, 10 1950.

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

[94] I. Tyukin, D. Higham, A. Gorban, and E. Woldegeorgis. The feasibility and inevitability of stealth attacks. *arXiv2106.13997*, 2021.

[95] J. von Neumann. First draft of a report on the edvac. *IEEE Annals of the History of Computing*, 15(4):27–75, 1993.

[96] M. Webb and S. Olver. Spectra of Jacobi operators via connection coefficient matrices. *Communications in Mathematical Physics*, 382(2):657–707, 2021.

[97] S. Weinberger. *Computers, Rigidity, and Moduli: The Large-Scale Fractal Geometry of Riemannian Moduli Space*. Princeton University Press, USA, 2004.

[98] D. Yarotsky. Optimal approximation of continuous functions by very deep relu networks. In *Conference on learning theory*, pages 639–649. PMLR, 2018.

[99] 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):487–492, 2018.

DEPARTMENT OF MATHEMATICS, UNIVERSITY OF OSLO
Email address: lucaeg@student.matnat.uio.no

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