INSTABILITY IN DEEP LEARNING – WHEN ALGORITHMS CANNOT COMPUTE UNCERTAINTY QUANTIFICATIONS FOR NEURAL NETWORKS

LUCA EVA GAZDAG, VEGARD ANTUN, AND ANDERS C. HANSEN

ABSTRACT. In deep learning, interval neural networks are used to quantify the uncertainty of a pre-trained neural network. Suppose we are given a computational problem P and a pre-trained neural network Φ_P that aims to solve P. An interval neural network is then a pair of neural networks $(\underline{\phi}, \overline{\phi})$, with the property that $\underline{\phi}(y) \leq \Phi_P(y) \leq \overline{\phi}(y)$ for all inputs y, that are specifically trained to quantify the uncertainty of Φ_P, in the sense that the size of the interval $[\underline{\phi}(y), \overline{\phi}(y)]$ quantifies the uncertainty of the prediction $\Phi_P(y)$. In this paper we investigate the phenomenon when algorithms cannot compute interval neural networks in the setting of inverse problems. We show that in the typical setting of a linear inverse problem, the problem of constructing an optimal pair of interval neural networks is non-computable, even with the assumption that the pre-trained neural network Φ_P is an optimal solution. In other words, there exist classes of training sets Ω, such that there is no algorithm, even randomized (with probability $p > 1/2$), that computes an optimal pair of interval neural networks for each training set $\mathcal{T} \in \Omega$. This phenomenon happens even when we are given a pre-trained neural network $\Phi_{\mathcal{T}}$ that is optimal for $\mathcal{T}$. This phenomenon is intimately linked to the instability in deep learning.

CONTENTS

1. INTRODUCTION

Computing confidence intervals for neural networks (NNs) have great potential for significantly enhancing the trust of AI systems. In recent years, such intervals NNs as described in the abstract, have been used for a wide range of task, including uncertainty estimation [67, 77, 81], verifying predictions [9, 71] and enhancing our understanding of their approximation properties [95]. However, recent studies have also raised issues surrounding the limitations of such intervals [9,71,95]. It has, for example, been demonstrated how interval analysis of NNs cannot necessarily verify robust predictions [71], or how for certain problems it is NP-hard to compute confidence intervals for NNs [95]. More broadly, this follows the larger trend in machine learning the last few years of studying the limitations of AI-systems [87], whether it is for large language models [100], predictive models [45, 50] or limitations of training algorithms [39]. Thus, a pertinent question is the following:

> *Suppose there exist interval NNs that provide uncertainty quantification of a neural network, can they be computed by an algorithm?*

In this work, we continue the trend mentioned above and address the above question by investigating whether one can design classes of training datasets for which any choice of training algorithm fails to compute so called interval NNs for uncertainty quantification. We consider the procedure proposed by Oala et al. in [77] for computing interval NNs for linear discrete inverse problems and we show that for certain classes of training data there does not exist any algorithm that can compute approximations to these interval NNs beyond a certain accuracy.

The mechanism causing this phenomenon utilises an inherent instability of the training procedure with respect to the training datasets. A consequence of this instability and the subsequent lack of reliable algorithms have several consequences (see §3.3 for a more detailed description):

One may prove existence of interval NNs providing uncertainty quantification for DL methods, over some class of input training sets. However, any algorithm trying to compute the interval NNs over this class will yield incorrect uncertainty on some input training set. The incorrectness can manifest in two ways:

(1) The incorrectness can lead to the conclusion that there are no – or small – uncertainties, when in fact the uncertainties may be severe.
(2) Or the incorrectness can lead to the conclusion that there are severe uncertainties – when in fact – we have sharpness for the optimal interval neural networks.

Another issue that arises from the instability, is that training interval NNs will be susceptible to *data poisoning*. That is, the phenomenon where an attacker is allowed to alter some portion of the training data, often in an imperceivable way, causing the trained system to behave in an undesirable way for certain inputs. In this work, we demonstrate that there exist inputs such that even an arbitrarily small perturbation of the training dataset leads to significantly different interval neural networks when using standard training procedures.

1.1. **Notation.** We start by introducing some useful notation. The main essence of this paper is the study of neural networks, we therefore start by giving a precise definition of this concept. Let K be a natural number and let $\mathbf{N} := (N_0, N_1, \ldots, N_{K-1}, N_K) \in \mathbb{N}^{K+1}$. A neural network with dimension $(\mathbf{N}, K)$ is a map $\Phi : \mathbb{R}^{N_0} \to \mathbb{R}^{N_K}$ such that

$$\Phi(x) \coloneqq V^{(K)} \circ \sigma \circ V^{(K-1)} \circ \sigma \circ V^{(K-2)} \circ \cdots \circ \sigma \circ V^{(1)} x,$$

where, for $k = 1, \ldots, K$, the map $V^{(k)}$ is an affine map from $\mathbb{R}^{N_{k-1}} \to \mathbb{R}^{N_k}$, that is $V^{(k)}x^{(k)} = W^{(k)}x^{(k)} + b^{(k)}$, where $b^{(k)} \in \mathbb{R}^{N_k}$ and $W^{(k)} \in \mathbb{R}^{N_{k-1} \times N_k}$. The map $\sigma : \mathbb{R} \to \mathbb{R}$ is interpreted as a coordinate wise map and is called the *non-linearity* or *activation function*: typically σ is chosen to be continuous and non-polynomial, in [77] σ is chosen to be the ReLu-function.

Two other useful concepts for this paper are the element wise min and max functions, as these play an essential role in the construction of interval neural networks and in our proofs. For any vectors $x^1, x^2 \in \mathbb{R}^n$, we define $\max\{x^1, x^2\} = z_{max}$ and $\min\{x^1, x^2\} = z_{min}$, with

$$z^i_{max} = \max\{x^1_i, x^2_i\} = \begin{cases} x^1_i, & \text{if } x^1_i \geq x^2_i \\ x^2_i, & \text{otherwise} \end{cases} \quad \text{and} \quad z^i_{min} = \min\{x^1_i, x^2_i\} = \begin{cases} x^1_i, & \text{if } x^1_i \leq x^2_i \\ x^2_i, & \text{otherwise.} \end{cases}$$

for $i = 1, \ldots, n$.

2. Inverse problems and interval neural networks

2.1. **Linear discrete inverse problems.** Throughout this paper we study the construction of interval neural networks in the context of solving linear discrete inverse problems. A linear discrete inverse problem can be expressed as follows:

$$\text{Given measurements } y = Ax + e \in \mathbb{R}^m \text{ of } x \in \mathbb{R}^N, \text{ recover } x. \tag{2.1}$$

Here A is an $m \times N$ real-valued matrix modelling the measurement process of x, and e models measurement noise. While seemingly simple, this equation is sufficiently expressive to model many real-world

applications. Examples include most types of medical and industrial imaging (MRI, CT, microscopy) [3], image deblurring, statistical estimation [92], etc. These types of problems have been extensively studied in sparse recovery and compressed sensing when x is a sparse vector or a vector with some structured sparsity [2–4, 10, 20, 25, 27, 28, 43, 46, 60, 61]. However, recent developments have led to a plethora of AI techniques [6, 8, 53, 54, 59, 68, 78] for solving these types of problems.

In many of the applications listed above the $\operatorname{rank}(A) < N$, either because $m < N$ or because the matrix A is rank-deficient. In both cases, the kernel of the matrix is non-trivial, which means that even in the noiseless setting, the linear system in (2.1) does not have a unique solution. To make the problem tractable whenever $\operatorname{rank}(A) < N$, it is therefore customary to introduce a set $\mathcal{U}_1 \subset \mathbb{R}^N$ called the *model class* [3, 27, 43, 60, 61], and assume that x belongs to this set. We may, therefore, rewrite the noiseless problem as follows:

$$\text{Given measurements } y = Ax \in \mathbb{R}^m \text{ of } x \in \mathcal{U}_1 \subset \mathbb{R}^N, \text{ recover } x. \tag{2.2}$$

For the remainder of this paper we consider the noiseless version of (2.1) presented above, because providing results about computational barriers in the noiseless setting gives stronger results. If constructing optimal interval neural networks is non-computable in the noiseless setting, there should be no hope of constructing optimal interval neural networks from noisy measurements either.

Traditional methods for solving (2.2) typically make explicit assumptions about the set $\mathcal{U}_1$ and design the reconstruction mapping based on this choice. Examples of sets $\mathcal{U}_1$ found in the literature are sparse vectors, union of subspaces, manifolds, etc. [19, 27, 66]. Nowadays, learning based methods are popular alternatives to the traditional methods whenever one has access to a dataset,

$$\mathcal{T} = \{(x^{(1)}, y^{(1)}), \dots, (x^{(\ell)}, y^{(\ell)})\} \subset \mathbb{R}^N \times \mathbb{R}^m, \tag{2.3}$$

of training examples. For these methods, one seeks to find a NN $\Phi\colon \mathbb{R}^m \to \mathbb{R}^N$ for which $\Phi(y) \approx x$, for each $(x, y) \in \mathcal{T}$ and preferably also for all (x, y) in some holdout test set. Thus, for learning based methods, one does not describe $\mathcal{U}_1$ mathematically as for the traditional methods, but one implicitly aims to learn $\mathcal{U}_1$ from the data $\mathcal{T}$, by training a neural network which maps $Ax \mapsto x$ for each $x \in \mathcal{U}_1$.

2.2. **Computing interval neural networks for inverse problems.** In [77] the authors proposed to compute a pair of interval NNs $\overline{\phi}, \underline{\phi}\colon \mathbb{R}^m \to \mathbb{R}^N$ for the problem in (2.1), whose objective is to provide confidence intervals for a NN $\Phi\colon \mathbb{R}^m \to \mathbb{R}^N$ trained to solve the inverse problem. The confidence intervals would be computed component wise as $\overline{\phi}(y') - \underline{\phi}(y')$ for inputs y', by ensuring that the trained interval NNs satisfy

$$\underline{\phi}(y) \leq \Phi(y) \leq \overline{\phi}(y) \text{ componentwise for all inputs } y.$$

The idea proposed in [77], is then to utilize the component wise confidence intervals provided by these NNs to detect potential failure modes for Φ.

The method proposed in [77] works as follows. Given a dataset $\mathcal{T}$ as in (2.3) one first trains a feed-forward ReLU NN $\Phi\colon \mathbb{R}^m \to \mathbb{R}^N$ using a squared-error loss function. Then, given the weights $W^{(k)}$ and the biases $b^{(k)}$ of Φ, one trains two interval NNs $\underline{\phi}, \overline{\phi}\colon \mathbb{R}^m \to \mathbb{R}^N$ with the same number of layers as Φ, by minimizing the objective function

$$F_{\mathcal{T},\beta}(\underline{\phi}, \overline{\phi}) := \sum_{(x,y)\in\mathcal{T}} \|\max\{x - \overline{\phi}(y), 0\}\|_2^2 + \|\max\{\underline{\phi}(y) - x, 0\}\|_2^2 + \beta\|\overline{\phi}(y) - \underline{\phi}(y)\|_1, \quad \beta > 0 \tag{2.4}$$

subject to the constraints $\underline{W}^{(k)} \leq W^{(k)} \leq \overline{W}^{(k)}$ and $\underline{b}^{(k)} \leq b^{(k)} \leq \overline{b}^{(k)}$, where all inequalities are meant entry wise, and $\underline{W}^{(k)}, \overline{W}^{(k)}, \underline{b}^{(k)}, \overline{b}^{(k)}$ are the weights and the biases of $\underline{\phi}, \overline{\phi}$ respectively.

The observation made in [77] is that one can design a clever architecture for $\underline{\phi}, \overline{\phi}$, that can mimic interval arithmetic for the weights and biases by using the interval property of these parameters. This architecture

heavily exploits the fact that the ReLU activation function ensures that the input to each layer is non-negative. We refer to [77], for further details on this, but we note that this ensures that $\underline{\phi}(y) \leq \Phi(y) \leq \overline{\phi}(y)$ for all $y \in \mathbb{R}^m$.

Our goal in this work is to investigate whether there exists classes of training sets, for which no algorithm can reliably compute interval NNs. Following [77], these interval NNs must necessarily depend on the pre-trained NN Φ, the optimization parameter $\beta \in \mathbb{R}$, and the considered training set $\mathcal{T}$. In this work, we assume throughout that $\beta > 0$ is a given fixed dyadic number (a number which is exactly representable on a computer). In addition, we make several well-posedness assumptions about $\mathcal{T}$ and Φ, to ensure that our results are not a consequence of unreasonable training data or poor training of the neural network Φ.

We start with assumptions about $\mathcal{T}$. Let $B_1^N(0)$ denote the open unit Euclidean ball in $\mathbb{R}^N$, and assume that for a given integer $\ell > 1$ and a given model class $\mathcal{U}_1 \subseteq B_1^N(0)$, each x-coordinate of the training set $\mathcal{T} = \{(x^{(i)}, Ax^{(i)})\}_{i=1}^{\ell}$ belongs to the model class $\mathcal{U}_1$. That is, the set $\mathcal{T}$ is a member of

$$T_{\ell,\mathcal{U}_1} \coloneqq \{\{(x^{(1)}, Ax^{(1)}), \dots, (x^{(\ell)}, Ax^{(\ell)})\} \subset \mathcal{U}_1 \times A(\mathcal{U}_1) : x^{(1)}, \dots, x^{(\ell)} \in \mathcal{U}_1\}. \tag{2.5}$$

We require that $\mathcal{U}_1 \subset B_1^N(0)$ because we are interested in measuring the error of an algorithm relative to the size of the training sets, and $B_1^N(0)$ is an arbitrary, yet practical, restriction on the size of the training sets.

Throughout the paper, to mitigate the effects due to poor training of Φ, we assume that for a given training set $\mathcal{T}$, the corresponding pre-trained neural network Φ, satisfies $\Phi(y) \in \mathcal{V}_{\mathcal{T}}(y)$ for all $y \in \mathbb{R}^m$, where $\mathcal{V}_{\mathcal{T}} : \mathbb{R}^m \rightrightarrows \mathbb{R}^n$ is, the potentially multivalued map, given by

$$\mathcal{V}_{\mathcal{T}}(y) \coloneqq \operatorname*{argmin}_{z \in R^N} \operatorname{dist}(z, \pi_1(\mathcal{T})) + \|Az - y\|_2, \tag{2.6}$$

with $\operatorname{dist}(z, \pi_1(\mathcal{T})) \coloneqq \inf_{u \in \pi_1(\mathcal{T})} \|z - u\|_2$, and where $\pi_1(\mathcal{T}) \coloneqq \{x \in \mathbb{R}^N : (x, y) \in \mathcal{T}\}$. The double arrow notation $\rightrightarrows$ indicates a multi-valued map. This mapping is inspired by the notion of *instance optimality*, introduced in [34], and later developed in works such as [24, 42, 98]. Since $x \in \pi_1(\mathcal{T})$ implies that $x \in \mathcal{V}_{\mathcal{T}}(Ax)$, we get that the neural network Φ has an excellent performance on all elements in $\pi_2(\mathcal{T})$, where $\pi_2(\mathcal{T}) \coloneqq \{y \in \mathbb{R}^m : (x, y) \in \mathcal{T}\}$, which means that Φ maps $y = Ax$ to a unique x, or selects one out of the (many) solutions when $\pi_1(\mathcal{T})$ has redundant elements.

Further, we need to make sure that the classes of neural networks that we are working with are rich enough to guarantee the existence of well-behaved interval neural networks. We do not wish for our computational barriers to arise due to the lack of existence of interval neural networks nor the lack of expressiveness of these networks. The following assumption is a technical assumption that ensures this.

Assumption 2.1. *For a given fixed integer $\ell \geq 1$, we assume that $\mathcal{NN}_\ell$ is a class of neural networks such that for any collection $\mathcal{T} = \{(x^{(1)}, y^{(1)}), \dots, (x^{(\ell)}, y^{(\ell)})\} \subset \mathbb{R}^N \times \mathbb{R}^m$, where each y-coordinate is distinct, the following holds:*

- *(i) (ℓ-interpolatory): There exists a neural network $\Psi \in \mathcal{NN}_\ell$, such that $\Psi(y) = x$ for each $(x, y) \in \mathcal{T}$.*
- *(ii) For any choice of $x' \in \mathbb{R}^N$, any $k \in \{1, \dots, \ell\}$ and any $\Psi \in \mathcal{NN}_\ell$ satisfying (i), there exist neural networks $\underline{\phi}, \overline{\phi} \in \mathcal{NN}_\ell$, such that*
  - *(a) $\underline{\phi}(y) \leq \Psi(y) \leq \overline{\phi}(y)$ for all $y \in \mathbb{R}^m$, and*
  - *(b) such that $\underline{\phi}(y) = \overline{\phi}(y) = x$ for all $(x, y) \in \mathcal{T} \setminus \{x^{(k)}, y^{(k)}\}$, and*
  - *(c) $\underline{\phi}(y^{(k)}) = \min\{x', x^{(k)}\}$ and $\overline{\phi}(y^{(k)}) = \max\{x', x^{(k)}\}$.*

Here the first condition states that the network Ψ is well-trained, i.e., that it interpolates all the training data in the set $\mathcal{T}$. The second condition is centered on the existence of certain interval neural networks. Condition (a) states that there exists interval neural networks $\underline{\phi}$ and $\overline{\phi}$ that provide elementwise upper and lower bounds for the chosen network Ψ. Condition (b), states that these interval neural networks provide

sharp interval bounds for all elements in $\mathcal{T}$, except for on one (arbitrarily chosen) input pair $(x^{(k)}, y^{(k)})$, and condition (c) gives precise values on the output of these networks for the chosen input $y^{(k)}$.

Our main motivation for introducing Assumption 2.1 is to allow the most general results possible, while securing existence of well-behaved interval neural networks. We do not restrict our attention to classes of ReLU networks, but rather extend our results to any class of neural networks that satisfies Assumption 2.1. We wish to highlight that Assumption 2.1 is satisfied by any class of ReLU-networks of fixed depth greater than or equal to 2. We illustrate this in Appendix A. Thus, proving results for classes of NNs that satisfy Assumption 2.1 is strictly stronger than addressing the classes of neural networks that are considered by the authors in [77].

We assume throughout the paper that all the neural networks that we are working with are contained in a class $\mathcal{NN}_\ell$ that satisfies Assumption 2.1. In particular we assume that, for a given training set $\mathcal{T}$ and a given class of neural networks $\mathcal{NN}_\ell$, that satisfies Assumption 2.1, the pre-trained neural network Φ belongs to the following set of $\mathcal{W}(\mathcal{T}, \mathcal{NN}_\ell)$ defined by

$$\mathcal{W}(\mathcal{T}, \mathcal{NN}_\ell) := \{\Psi \in \mathcal{NN}_\ell : \Psi(y) \in \mathcal{V}_\mathcal{T}(y) \text{ for all } y \in \mathbb{R}^m\}. \tag{2.7}$$

We also require that the interval neural networks $\underline{\phi}$ and $\overline{\phi}$ belong to $\mathcal{NN}_\ell$, and that they respect the interval property $\underline{\phi}(y) \leq \Phi(y) \leq \overline{\phi}(y)$ for all $y \in \mathbb{R}^m$. In order to ensure this we introduce the following optimization classes:

$$\mathcal{NN}^u_\Phi = \{\underline{\phi} : \mathbb{R}^m \to \mathbb{R}^N \mid \underline{\phi} \in \mathcal{NN}_\ell \text{ and } \underline{\phi}(y) \leq \Phi(y) \text{ for all } y \in \mathbb{R}^m\}, \text{ and} \tag{2.8}$$

$$\mathcal{NN}^o_\Phi = \{\overline{\phi} : \mathbb{R}^m \to \mathbb{R}^N \mid \overline{\phi} \in \mathcal{NN}_\ell \text{ and } \Phi(y) \leq \overline{\phi}(y) \text{ for all } y \in \mathbb{R}^m\}. \tag{2.9}$$

Remark 2.2. The optimization classes above state properties of the neural networks $\Phi, \underline{\phi}$ and $\overline{\phi}$, rather than properties of the parameters of these networks. This gives us much greater flexibility in designing these NNs, because the condition $\underline{\phi}(y) \leq \Phi(y) \leq \overline{\phi}(y)$ is strictly weaker than the conditions of having both $\underline{W}^{(k)} \leq W^{(k)} \leq \overline{W}^{(k)}$ and $\underline{b}^{(k)} \leq b^{(k)} \leq \overline{b}^{(k)}$.

Remark 2.3. In [77], the neural networks Φ and $\underline{\phi}, \overline{\phi}$ have slightly different architectures, due to the fact that $\underline{\phi}$ and $\overline{\phi}$ model interval arithmetic for Φ. It is straightforward to state Assumption 2.1, with two different neural network classes, adopting this detail into our result. However, in the interest of keeping things short and clean, we do not make this distinction.

3. Main Theorem

3.1. **Preliminaries.** We introduce some notation on how to encode the parameters of a neural network Φ. Given the number of layers $K \in \mathbb{N}$, and some pre-fixed dimensions $N_0, N_1, \ldots, N_K \in \mathbb{N}$, we define θ_Φ to be the collection

$$\theta_\Phi = \bigcup_{k=0}^{K} \{b_j^{(k)}\}_{j=1}^{j=N_k} \cup \{W_{i,j}^{(k)}\}_{i,j=1}^{i=N_{k-1}, j=N_k}. \tag{3.1}$$

In other words, θ_Φ stores the information about the entries of the weights and the biases of the neural network Φ with pre-fixed dimensions. With this in place, we give the following more precise statement about our objective. For a given class $\mathcal{NN}_\ell$ that satisfies Assumption 2.1, a given model class $\mathcal{U}_1$, a parameter $\beta > 0$, a collection

$$\Omega \subseteq \{(\mathcal{T}, \theta_\Phi) \mid \mathcal{T} \in T_{\ell, \mathcal{U}_1},\ \theta_\Phi \text{ parameters of } \Phi, \text{ and } \Phi \in \mathcal{W}(\mathcal{T}, \mathcal{NN}_\ell)\}, \tag{3.2}$$

and a mapping $\Xi_\beta \colon \Omega \rightrightarrows \mathcal{NN}^u_\Phi \times \mathcal{NN}^o_\Phi$, given by

$$\Xi_\beta(\mathcal{T}, \theta_\Phi) = \underset{\underline{\psi} \in \mathcal{NN}^u_\Phi, \overline{\psi} \in \mathcal{NN}^o_\Phi}{\operatorname{argmin}} F_{\mathcal{T},\beta}(\underline{\psi}, \overline{\psi}), \tag{3.3}$$

our goal is to investigate whether we can compute an element $(\underline{\phi}, \overline{\phi}) \in \Xi_\beta(\mathcal{T}, \theta_\Phi)$ for each pair $(\mathcal{T}, \theta_\Phi) \in \Omega$, when reading inexact inputs. Throughout the paper we will assume that $\Xi_\beta(\mathcal{T}, \theta_\Phi) \neq \emptyset$. This is obvious for standard choices of the nonlinear σ.

We say that an algorithm $\Gamma : \Omega \to \mathcal{NN}^u_\Phi \times \mathcal{NN}^o_\Phi$ approximates an optimal pair of interval neural networks $(\underline{\phi}, \overline{\phi}) \in \mathcal{NN}^u_\Phi \times \mathcal{NN}^o_\Phi$ to accuracy δ, for a training set $\mathcal{T}$, if

$$\mathrm{dist}_\mathcal{T}(\Gamma(\mathcal{T}, \theta_\Phi), \Xi_\beta(\mathcal{T}, \theta_\Phi)) := \inf_{(\underline{\phi}, \overline{\phi}) \in \Xi_\beta(\mathcal{T}, \theta_\Phi)} d_\mathcal{T}(\Gamma(\mathcal{T}, \theta_\Phi), (\underline{\phi}, \overline{\phi})) \leq \delta,$$

with

$$d_\mathcal{T}((\underline{\phi}, \overline{\phi}), (\underline{\psi}, \overline{\psi})) = \max \left\{ \sup_{y \in \pi_2(\mathcal{T})} \|\overline{\phi}(y) - \overline{\psi}(y)\|_2^2, \sup_{y \in \pi_2(\mathcal{T})} \|\underline{\phi}(y) - \underline{\psi}(y)\|_2^2 \right\}, \tag{3.4}$$

where $\pi_2(\mathcal{T}) := \{y \in \mathbb{R}^m \, : \, (x, y) \in \mathcal{T}\}$.

3.2. **Inexactness and the main theorem.** For completeness, we give an intuitive explanation of why we need to consider inexact input and what we mean by inexact input. Due to the discrete nature of modern computers, a training set $\mathcal{T} = \{(x^{(j)}, y^{(j)})\}_{j=1}^r$ can often not be represented exactly on a computer, because the only numbers that are exactly representable are a subset of the *dyadic rationals* given by $\mathbb{D} = \{a2^{-j} \, : \, a \in \mathbb{Z}, \, j \in \mathbb{N}\}$. The matrix A could, for example, have irrational entries, which happens in many real world applications. The entries of A can only be approximated in finite base-2 arithmetic, giving rise to round-off approximations, and an inexact representation of the elements in the training set $\mathcal{T}$.

Thus, for each choice of training set $\mathcal{T}$, we assume that the algorithm has access to a sequence of dyadic training sets $\{\mathcal{T}_n\}_{n \in \mathbb{N}}$, where for each $(x, y) \in \mathcal{T}$ and $n \in \mathbb{N}$, there is a pair $(x^n, y^n) \in \mathcal{T}_n \subset \mathbb{D}^N \times \mathbb{D}^m$, such that for all $i = 1, \ldots, N$ and $j = 1, \ldots, m$, we have that $|x_i^n - x_i| \leq 2^{-n}$ and that $|y_j^n - y_j| \leq 2^{-n}$. With $n(N) = n + \lceil \log_2(\sqrt{N}) \rceil$ and $n(m) = n + \lceil \log_2(\sqrt{m}) \rceil$ we then get that

$$\|x^{n(N)} - x\|_2 \leq 2^{-n} \quad \text{and} \quad \|y^{n(m)} - y\|_2 \leq 2^{-n}.$$

Similarly, for each parameter θ_Φ^h for $h \in H$, where H is some finite index set for the parameters of Φ, there is a dyadic sequence of numbers $\{\theta_\Phi^{h,n}\}_{n \in \mathbb{N}}$, such that for each $n \in \mathbb{N}$ we have that $|\theta_\Phi^{h,n} - \theta_\Phi^h| \leq 2^{-n}$. At last, we require that the approximations satisfy the same well posedness assumptions as $\mathcal{T}$ and Φ. Specifically, that $\mathcal{T}_n \in T_{\ell, \mathcal{U}_1}$ for each $n \in \mathbb{N}$, and that the neural network Φ_n, represented by the parameters $\{\theta_\Phi^{h,n}\}_{h \in H}$, satisfies $\Phi_n \in \mathcal{W}(\mathcal{T}_n, \mathcal{NN}_\ell)$ for each $n \in \mathbb{N}$.

We require that a successful algorithm should work on any approximating sequence of the form $(\{\mathcal{T}_n\}_{n \in \mathbb{N}}, \{\theta_{\Phi_n}\}_{n \in \mathbb{N}})$. Note that this extended computational model accepting 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 [12, 21, 35–37, 39, 40, 48, 49, 51, 55, 56, 63, 65, 89].

Remark 3.1. The above model means that for each training set $\mathcal{T} = \{(x^{(j)}, y^{(j)})\}_{j=1}^r$ and for each pre-trained neural network $\Phi \in \mathcal{W}(\mathcal{T}, \mathcal{NN}_\ell)$, we have infinitely many collections of approximate sequences

$$(\tilde{\mathcal{T}}, \tilde{\theta}_\Phi) = (\{\{(x^{(j),n}, y^{(j),n})\}_{n \in \mathbb{N}}\}_{j=1}^r, \{\{\theta_\Phi^{h,n}\}_{n \in \mathbb{N}}\}_{h \in H}) \tag{3.5}$$

as described above. A sequence of approximations is provided to the algorithm through an 'oracle'. For example, in the case of a Turing machine [89], this would be through an oracle input tape (see [63] for the standard setup), or in the case of a Blum-Shub-Smale (BSS) machine [22], this would be through an oracle node. The algorithm can thus ask for an approximation to any given accuracy as described above, and use as many queries as desired. 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 [82] – i.e. when the inexact input is required to be computable, in particular when $\{(x^{(j),n}, y^{(j),n})\}_{n \in \mathbb{N}}$ and $\{\theta_\Phi^{h,n}\}_{n \in \mathbb{N}}$ are computable sequences for each $j = 1, \ldots, r$ and $h \in H$.

We end this section by informally stating the main theorem of this paper. See Theorem 5.8 for the detailed statement.

Theorem 3.2 (Non-existence of algorithms)**.** *Let $m, N \in \mathbb{N}$ with $N \geq 2$, and let $A \in \mathbb{R}^{m\times N}$ be such that $1 \leq \operatorname{rank}(A) < N$. Let $\ell \geq 2$, $\kappa \in (0, 1/3)$ and let $\mathcal{NN}_\ell$ be a class of neural networks that satisfy Assumption* 2.1*. Then, for any $\beta \in (0, \sqrt{\kappa}/(4\sqrt{2N}))$, there exist infinitely many model classes $\mathcal{U}_1 \subseteq B_1^N(0)$, and for each model class, infinitely many collections*

$$\Omega = \{(\mathcal{T}, \theta_\Phi) \,:\, \mathcal{T} \in T_{\ell,\mathcal{U}_1} \text{ and } \Phi \in \mathcal{W}(\mathcal{T}, \mathcal{NN}_\ell)\}$$

of training sets and corresponding pre-trained neural networks, such that no algorithm, even randomised (with probability $p > 1/2$), can approximate an optimal pair of interval neural networks $(\underline{\phi}, \overline{\phi}) \in \Xi_\beta(\mathcal{T}, \theta_\Phi)$ for all elements $(\mathcal{T}, \theta_\Phi) \in \Omega$ to accuracy $\kappa/2$, when reading inexact input.

3.3. **Consequences of the main theorem.** Theorem 3.2 tells us that there exists classes of training sets Ω, such there exists no algorithm, even randomized, that can approximate an optimal pair of interval neural networks, for each training set $\mathcal{T} \in \Omega$. Moreover, that this happens even when we are given a pre-trained neural network $\Phi_\mathcal{T}$, that is optimal for $\mathcal{T}$, for each $\mathcal{T} \in \Omega$. The problem of constructing an optimal $\Phi_\mathcal{T}$ for each $\mathcal{T} \in \Omega$ is non-computable in itself [51], but Theorem 3.2 shows that, even if we had an oracle providing such a network, there are cases where we would not be able to give a correct classification of it's uncertainty. Recall that the projections onto the coordinates of a training set $\mathcal{T}$ are denoted by

$$\pi_1(\mathcal{T}) = \{x \in \mathbb{R}^N \,:\, (x, y) \in \mathcal{T}\} \quad \text{and} \quad \pi_2(\mathcal{T}) = \{y \in \mathbb{R}^m \,:\, (x, y) \in \mathcal{T}\}. \tag{3.6}$$

The goal of minimizing (2.4) is to provide the sharpest possible bounds for the set $X_y :- \{x \in \pi_1(\mathcal{T}) \,:\, y = Ax\}$ for each $y \in \pi_2(\mathcal{T})$, in the sense that the interval $[\underline{\phi}(y), \overline{\phi}(y)]$ should be the smallest possible with the property that $\underline{\phi}(y) \leq x \leq \overline{\phi}(y)$ for all $x \in X_y$, where the inequalities are meant component wise. The failure of constructing an optimal pair of interval neural networks can lead to two different types of errors:

(1) *(Wrong certainty prediction.)* In the case where $X_y = \{x^1, x^2\}$ with $x^1 \neq x^2$ for a $y \in \pi_2(\mathcal{T})$, the reading of inexact input could lead the computer to interpret the pairs $(x^1, y), (x^2, y) \in \mathcal{T}$ as two pairs (x^1, y^1) and (x^2, y^2) with $y^1 \neq y^2$. Minimizing (2.4) with this interpretation would yield that $\underline{\phi}(y^1) = \overline{\phi}(y^1) = x^1$ and that $\underline{\phi}(y^2) = \overline{\phi}(y^2) = x^2$, yielding an uncertainty score of zero for both y^1 and y^2, whereas an optimal solution should give non-zero uncertainty score to the element y instead. In other words, inexact representations might lead us to think that there is no uncertainty in our predictions, even when there is uncertainty present.

(2) *(Failure of sharpness).* On the other hand, the inexact reading of inputs could result in the computer interpreting two distinct points $(x^1, y^1), (x^2, y^2) \in \mathcal{T}$ with $y^1 \neq y^2$ as pairs $(x^1, y), (x^2, y)$, in this case, the algorithm over-quantifies the uncertainty, and we will think that our prediction is less certain than it is.

(3) *(Data poisoning).* Data poisoning is a phenomenon where an attacker subtly alters a portion of the training data, often in an imperceptible manner, leading to the trained system behaving unreliably for specific inputs. The above instabilities demonstrate that certain inputs exist such that even minimal perturbations to the training dataset can result in significantly different interval neural networks when using standard training procedures.

To make matters even worse, we will have no way of detecting whether we are in failure mode (1) or (2) above, or in other words whether we in general should expect that there is more, or less, uncertainty than what our uncertainty estimation predicts.

4. Connection to previous work

The results in this paper are part of the program on the limitations of AI, which is the topic of the 18th problem on Smale's list of mathematical problems for the 21st century [86]. This paper specifically connects the limitations of AI to the instability of neural networks and the challenges of reliably quantifying uncertainty computationally. It achieves this by establishing lower bounds on the 'breakdown epsilon' for the problem of computing interval neural networks. The concept of 'breakdown epsilon' stems from *generalized hardness of approximation* [12, 39, 47], which is part of the mathematics behind the Solvability Complexity Index (SCI) hierarchy.

Instability in AI. Our results are closely linked to the instability phenomenon in AI methods. Neural networks become universally unstable (non-robust) when trained to solve such problems in virtually any application [3, 5, 6, 29, 33, 50, 57, 58, 72, 88]. For early work on the instability of neural networks, see S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard et al. [72, 73]. Our work particularly relates to the research of A. Bastounis et al. [11], B. Adcock et al. [1], and V. Antun et al. [6]. For recent developments, see D. Higham and I. Tyukin et al. [90, 91], as well as the results by L. Bungert and G. Trillos et al. [26], and S. Wang, N. Si, and J. Blanchet [94].

Generalised hardness of approximation and the mathematics behind the SCI hierarchy. Note that the tools used and further developed in this paper have roots in generalised hardness of approximation (GHA) and the mathematics behind the Solvability Complexity Index (SCI) hierarchy. For the initial results on GHA see A. Bastounis et al. [12], M. Colbrook et al. [39]. See also [51] and Problem 5 (J. Lagarias) in [47]. The SCI framework, introduced in [55], has been further developed through significant contributions by J. Ben-Artzi, M. Marletta and F. Rösler [15, 16], M. Colbrook et al. [36, 38], and O. Nevanlinna with co-authors [13, 14, 56]. Notable work also includes contributions by S. Olver and M. Webb [37, 96]. The SCI hierarchy is intrinsically connected to S. Smale's [83, 84] foundational program on computational mathematics and scientific computing. This initiative spurred early pioneering work by C. McMullen [69, 70, 85], as well as P. Doyle and C. McMullen [44], particularly in the area of polynomial root-finding, leading to crucial classification results which can be interpreted within the SCI hierarchy. Additional classification efforts in the SCI hierarchy have been made by S. Weinberger [97]. The mathematical roots of the SCI hierarchy trace back to the work of K. Gödel [52] and A. Turing [89], though contemporary techniques have expanded to accommodate any model of computation.

Robust optimization. Our results and the concept of GHA are directly linked to robust optimization and the seminal work by Ben-Tal, El Ghaoui, and Nemirovski [17, 18, 74, 75]. Specifically, GHA is essential for any robust optimisation theory aimed at computing minimisers of both convex and non-convex optimisation problems. Our work illustrates the delicate issues occurring when AI, instability and robust optimisation meet uncertainty quantification.

Existence vs computability of NNs. There is extensive literature on the existence results of neural networks (NNs) [23, 79, 99]. Noteworthy contributions include those by F. Voigtlaender et al. [93], review papers by A. Pinkus [80], and the work of R. DeVore, B. Hanin, and G. Petrova [41]. We also mention approximation theorems developed by P. Kidger and T. Lyons [62]. However, as demonstrated by M. Colbrook, V. Antun et al. in [39], only a small subset of the NNs that are theoretically proven to exist can be computed by algorithms. Moreover, within the framework proposed by A. Chambolle and T. Pock [31, 32], the results in [39] show that, under specific assumptions, stable and accurate NNs can indeed be computed. Additionally, the work by P. Niyogi, S. Smale, and S. Weinberger [76] provides important insights into the existence of algorithms for learning.

5. Mathematical preliminaries from the SCI hierarchy

5.1. **Computational problems and Δ_1-information.** We formulate our results in the language of the Solvability Complexity Index (SCI) hierarchy. The SCI hierarchy is a mathematical framework designed to classify the intrinsic difficulty of different computational problems found in scientific computing. As its theory is now extensive, we only cite a limited number of results [12–16, 35–37, 39, 55, 56]. For completeness, we include the definitions from the SCI hierarchy that are needed for our statements and proofs, and we follow the lines of [12] in our presentation.

The most basic building block in the SCI hierarchy is the notion of a computational problem. We start by abstractly defining the concept of a computational problem, and thereafter, we describe how computing interval neural networks can be embedded into this language.

Definition 5.1 (Computational problem [12])**.** Let Ω be some set, which we call the *domain* or *input set*. Let Λ be a set of complex valued functions on Ω such that for $\iota_1, \iota_2 \in \Omega$, then $\iota_1 = \iota_2$ 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.

In the above definition, the function $\Xi\colon \Omega \to \mathcal{M}$ is the problem function which gives rise to the computational problem. It is this function we seek to approximate/compute. The set Ω is the domain of Ξ and the metric space $(\mathcal{M}, d)$ is the range of this function. The set Λ describes which type of information we can acquire about the input to Ξ. It is this information which can be accessed by an algorithm. For convenience, we throughout the manuscript restrict our attention to sets Λ that are finite.

Recall that for a given training set size $\ell \geq 2$, model class $\mathcal{U}_1 \subset B_1^N(0)$, and matrix $A \in \mathbb{R}^{m\times N}$, we have that

$$T_{\ell,\mathcal{U}_1} = \{\{(x^{(1)}, Ax^1), \dots, (x^{(\ell)}, Ax^{(\ell)})\} \subset \mathcal{U}_1 \times A(\mathcal{U}_1) : x^{(1)}, \dots, x^{(\ell)} \in \mathcal{U}_1\} \tag{5.1}$$

denotes the class of all training sets of size ℓ with respect to the model class $\mathcal{U}_1$. Then, for a given class $\mathcal{NN}_\ell$ of neural networks that satisfy Assumption 2.1, the domain Ω of our computational problem is a collection of pairs of such training sets and the corresponding parameters of a pre-trained neural network $\Phi \in \mathcal{W}(\mathcal{T}, \mathcal{NN}_\ell)$, where $\mathcal{W}(\mathcal{T}, \mathcal{NN}_\ell)$ is as defined in (2.7). More precisely,

$$\Omega \subseteq \{(\mathcal{T}, \theta_\Phi) \mid \mathcal{T} \in T_{\ell,\mathcal{U}_1},\ \theta_\Phi \text{ parameters of } \Phi, \text{ and } \Phi \in \mathcal{W}(\mathcal{T}, \mathcal{NN}_\ell)\}. \tag{5.2}$$

For any fixed $\beta > 0$, we then seek to compute a minimization of (2.4) for neural networks $\underline{\phi} \in \mathcal{NN}_\Phi^u$ and $\overline{\phi} \in \mathcal{NN}_\Phi^o$. That is, for a given domain Ω of training sets and pre-trained neural networks, our problem function $\Xi_\beta\colon \Omega \rightrightarrows \mathcal{M}$, with $\mathcal{M} := \mathcal{NN}_\mathcal{T}^u \times \mathcal{NN}_\mathcal{T}^o$ is given by

$$\Xi_\beta(\mathcal{T}, \theta_\Phi) = \operatorname*{argmin}_{\underline{\psi}\in\mathcal{NN}_\Phi^u, \overline{\psi}\in\mathcal{NN}_\Phi^o} F_{\mathcal{T},\beta}(\underline{\psi}, \overline{\psi}), \tag{5.3}$$

In summary, the problem function Ξ_β maps a training set $\mathcal{T}$, and the weights θ_Φ of a corresponding pre-trained neural network Φ, to the set of interval neural networks $\{(\underline{\psi}, \overline{\psi})\} \subseteq \mathcal{NN}_\Phi^u \times \mathcal{NN}_\Phi^o$, which attain the minimum value of $F_{\mathcal{T},\beta}$.

Now, in order to measure the approximation error of different algorithms, whose objective is to solve the computational problem defined above, we need to equip $\mathcal{M}$ with a suitable distance function. We therefore view $\mathcal{M}$ as a metric space equipped with the following metric:

$$d_\mathcal{M}((\underline{\phi}, \overline{\phi}), (\underline{\psi}, \overline{\psi})) := \max\left\{ \sup_{y\in A(\mathcal{U}_1)} \|\overline{\phi}(y) - \overline{\psi}(y)\|_2^2, \sup_{y\in A(\mathcal{U}_1)} \|\underline{\phi}(y) - \underline{\psi}(y)\|_2^2 \right\}, \tag{5.4}$$

where we identify the neural networks that are equivalent on $A(\mathcal{U}_1)$. The observant reader might notice that this metric is slightly different from the distance function introduced in (3.4). The reason for this is that

we wish to have a unified metric for the whole domain Ω, and the distance function in (3.4) is not suitable for this, because it depends on each individual training set. However, we will prove that the computational breakdown for any training set $\mathcal{T}$ happens at a point contained in $\pi_2(\mathcal{T})$.

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 for the training sets $\mathcal{T}$ and the parameters θ_Φ as follows: Given a training set $\mathcal{T}$, let $f^k_{x,i} : \Omega \to \mathbb{R}$ be given by $f^k_{x,i}((\mathcal{T}, \theta_\Phi)) = x^{(k)}_i$, where the index i denotes the i'th coordinate of the vector $x^{(k)}$. We define $f^k_{y,j} : \Omega \to \mathbb{R}$ in the same way to measure the y-coordinates of $\mathcal{T}$, more precisely $f^k_{y,j}((\mathcal{T}, \theta_\Phi)) = y^{(k)}_j$. Next, we introduce the measurements of the parameters θ_Φ. As we have seen in (3.1), the parameters θ_Φ of a neural network Φ are represented as a finite collection of real numbers $\theta_\Phi = \{\theta^h_\Phi\}_{h\in H}$ with $|H| < \infty$. We then introduce the following measurements: Let $f^h_\theta : \Omega \to \mathbb{R}$ be given by $f^h_\theta((\mathcal{T}, \theta_\Phi)) = \theta^h_\Phi$. In summary, our set of measurements is given by

$$\Lambda = \{f^k_{y,j}, f^k_{x,i}, f^h_\theta : \Omega \to \mathbb{R} \,|\, i = 1, \dots, N \ , \ j = 1, \dots, m, \ , \ k = 1, \dots, \ell, \text{ and } h \in H\}. \tag{5.5}$$

As we have mentioned, in these cases we cannot necessarily access or store the values $f_j(\iota)$ on a computer, but we can compute approximations $f_{j,n}(\iota) \in \mathbb{D} + i\mathbb{D}$ such that $f_{j,n}(\iota) \to f_j(\iota)$ as $n \to \infty$. The concept of Δ_1-information formalizes this.

Definition 5.2 (Δ_1-information [12])**.** Let $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$ be a computational problem. We say that Λ has Δ_1-information if for each $f_j \in \Lambda$ there are mappings $f_{j,n} : \Omega \to \mathbb{Q} + i\mathbb{Q}$ such that $|f_{j,n}(\iota) - f_j(\iota)| \leq 2^{-n}$ for all $\iota \in \Omega$. Furthermore, if $\widehat{\Lambda}$ is a collection of such functions, such that Λ has Δ_1-information, we say that $\widehat{\Lambda}$ provides Δ_1-information for Λ and we denote the family of all such $\widehat{\Lambda}$ by $\mathcal{L}^1(\Lambda)$.

In the above definition the set $\widehat{\Lambda}$ corresponds to one particular choice of Δ_1-information. However, we want to have algorithms that can handle any choice of such information. To formalize this, we define computational problems with Δ_1-information.

Definition 5.3 (Computational problem with Δ_1-information [12])**.** Let J be an index set for Λ. Then a computational problem where Λ has Δ_1-information is denoted by $\{\Xi, \Omega, \mathcal{M}, \Lambda\}^{\Delta_1} := \{\widetilde{\Xi}, \widetilde{\Omega}, \mathcal{M}, \widetilde{\Lambda}\}$, where

$$\widetilde{\Omega} = \left\{\widetilde{\iota} = \{f_{j,n}(\iota)\}_{j,n\in J\times\mathbb{N}} \,:\, \iota \in \Omega, \{f_j\}_{j\in J} = \Lambda, |f_{j,n}(\iota) - f_j(\iota)| \leq 2^{-n}\right\},$$

Moreover, if $\widetilde{\iota} = \{f_{j,n}(\iota)\}_{j,n\in J\times\mathbb{N}} \in \widetilde{\Omega}$ then we define $\widetilde{\Xi}(\widetilde{\iota}) = \Xi(\iota)$ and $\widetilde{f}_{j,n}(\widetilde{\iota}) = f_{j,n}(\iota)$. We also set $\widetilde{\Lambda} = \{\widetilde{f}_{j,n}\}_{j,n\in J\times\mathbb{N}}$. Note that $\widetilde{\Xi}$ is well-defined by Definition 5.1 of a computational problem and that the definition of $\widetilde{\Omega}$ includes all possible instances of Δ_1-information $\widehat{\Lambda} \in \mathcal{L}^1(\Lambda)$.

Lastly, we describe the precise notation for inexact representations of the elements in our specific case. Let $\hat{\Lambda} = \{f_n \,|\, f \in \Lambda, n \in \mathbb{N}\}$ be a set that provides Δ_1-information for Ω as defined in Definition 5.2. Then, for an arbitrary $(\mathcal{T}, \theta_\Phi) \in \Omega$ and $k \in \{1, \dots, \ell\}$, the corresponding inexact representation of $(x^{(k)}, y^{(k)}) \in \mathcal{T}$ is the pair of sequences $(\tilde{x}^{(k)}, \tilde{y}^{(k)})$, where

$$\tilde{x}^{(k)} = \{\{f^k_{x,i,n}(\mathcal{T}, \theta_\Phi)\}^{i=N}_{i=1}\}_{n\in\mathbb{N}} \quad \text{and} \quad \tilde{y}^{(k)} = \{\{f^k_{y,j,n}(\mathcal{T}, \theta_\Phi)\}^{j=m}_{j=1}\}_{n\in\mathbb{N}}. \tag{5.6}$$

Similarly, for the parameters $\theta_\Phi = \{\theta^h_\Phi\}_{h\in H}$: The inexact representation $\tilde{\theta}^h_\Phi$ of θ^h_Φ is the sequence

$$\tilde{\theta}^h_\Phi = \{f^h_{\theta,n}(\mathcal{T}, \theta_\Phi)\}_{n\in\mathbb{N}}, \quad \text{for each } h \in H.$$

We also recall that the approximations satisfy the same well posednes assumptions as $\mathcal{T}$ and Φ. More precisely, that $\mathcal{T}_n \in T_{\ell,\mathcal{U}_1}$ for each $n \in \mathbb{N}$, and that the neural network Φ_n, represented by the parameters $\{\theta^{h,n}_\Phi\}_{h\in H}$, satisfies $\Phi_n \in \mathcal{W}(\mathcal{T}_n, \mathcal{NN}_\ell)$ for each $n \in \mathbb{N}$. We do this to show that computational barriers arise, even when the approximations are well-posed.

5.2. **Algorithmic preliminaries: a user-friendly guide.** The main goal in this section is to introduce the concepts of a general algorithm, and of breakdown epsilons, which are needed for the formal statement and proof of Theorem 3.2. We follow the lines of [12] in our presentation of these concepts also.

5.2.1. *General algorithms.* The most common theoretical definition of an algorithm is a Turing machine, however, in this paper we use a more general definition of an algorithm, as this makes our impossibility statements stronger. In particular, by using a definition which encompasses any model of computation (such as Turing machines or real BSS machines), we ensure that proved lower bounds for computations are universally true for all reasonable models of computation. In the SCI hierarchy, this definition is given in terms of a general algorithm (defined below), whose goal is to capture the notion of a deterministic algorithm for a given computational problem.

Definition 5.4 (General Algorithm [12])**.** Given a computational problem $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$, a general algorithm is a mapping $\Gamma : \Omega \to \mathcal{M} \cup \{\mathrm{NH}\}$ such that for each $\iota \in \Omega$ the following hold

(i) There exists a non-empty subset of evaluations $\Lambda_\Gamma(\iota) \subset \Lambda$, and whenever $\Gamma(\iota) \neq \mathrm{NH}$, then $\Lambda_\Gamma(\iota)$ is finite.
(ii) The action of Γ on ι is determined uniquely by $\{f(\iota)\}_{f\in\Lambda_\Gamma(\iota)}$,
(iii) For every $\iota' \in \Omega$ such that $f(\iota') = f(\iota)$ for every $f \in \Lambda_\Gamma(\iota)$, it holds that $\Lambda_\Gamma(\iota') = \Lambda_\Gamma(\iota)$.

This definition requires some comments. The object NH stands for "non-halting" and is meant to capture the event that an algorithm runs forever. The introduction of this object requires that we extend the metric $d_{\mathcal{M}}$ on $\mathcal{M}$ to also handle NH. We do this as follows

$$d_{\mathcal{M}}(x,y) = \begin{cases} d_{\mathcal{M}}(x,y) & \text{if } x, y \in \mathcal{M}, \\ 0 & \text{if } x = y = \mathrm{NH}, \\ \infty & \text{otherwise.} \end{cases} \tag{5.7}$$

The three other properties listed in the above definition are requirements that any reasonable definition of an algorithm must satisfy. The first says that if the algorithm does halt, then it can only have read a finite amount of information. The second condition says that the algorithm can only depend on the information it has read. In particular, it cannot depend on information about ι that it has not read. The third condition ensures consistency. It says that if the algorithm reads the same information about two different inputs, then it cannot behave differently for these two inputs.

Remark 5.5 (The purpose of a general algorithm: universal impossibility results)**.** The purpose of a general algorithm is to have a definition that encompasses any model of computation, and that allows impossibility results to become universal. Given that there are several non-equivalent models of computation, impossibility results are shown with this general definition of an algorithm.

Randomness is an indispensable tool in computational mathematics which is used in many different areas, including optimization, algebraic computation, network routing, learning and data science. This motivates the need for formalizing what we mean by a randomised algorithm. According to [12], we define a randomised general algorithm as follows.

Definition 5.6 (Randomised General Algorithm [12])**.** 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 $\Gamma : \Omega \to \mathcal{M} \cup \{\mathrm{NH}\}$, 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 $\Gamma_\iota^{\mathrm{ran}} : (X, \mathcal{F}) \to (\mathcal{M} \cup \{\mathrm{NH}\}, \mathcal{B})$, defined by $\Gamma_\iota^{\mathrm{ran}}(\Gamma) = \Gamma(\iota)$, is a random variable, where $\mathcal{B}$ is the Borel sigma-algebra on $\mathcal{M} \cup \{\mathrm{NH}\}$.

(ii) For each $n \in \mathbb{N}$ and $\iota \in \Omega$, we have $\{\Gamma \in X \,|\, T_\Gamma(\iota) \leq n\} \in \mathcal{F}$, where

$$T_\Gamma(\iota) := \sup\{m \in \mathbb{N} \,|\, f_m \in \Lambda_\Gamma(\iota)\}$$

is the minimum amount of input information.

(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 $\mathbb{P}_{\iota_1}(E) = \mathbb{P}_{\iota_2}(E)$.

Here, the first two conditions are measure theoretic and ensure that one can use standard tools from probability theory to prove results about randomised general algorithms. The final condition ensures that the algorithm acts consistently on the inputs. In particular, for identical evaluations the probabilistic laws do not change. It was established in [12] that 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.

Remark 5.7. Following [12], we abuse notation slightly, and also write RGA for the family of all randomised general algorithms $(X, \mathcal{F}, \{\mathbb{P}_\iota\}_{\iota\in\Omega})$, and refer to the algorithms in RGA by Γ^{ran}.

Finally, we end this section with a formalized version of our main theorem.

Theorem 5.8. *Let $m, N \in \mathbb{N}$ with $N \geq 2$, and let $A \in \mathbb{R}^{m\times N}$ be such that $1 \leq \mathrm{rank}(A) < N$. Let $\ell \geq 2$, $\kappa \in (0, 1/3)$ and let $\mathcal{NN}_\ell$ be a class of neural networks that satisfies Assumption* 2.1. *Then for any $\beta \in (0, \sqrt{\kappa}/(4\sqrt{2N}))$ there exist infinitely many model classes $\mathcal{U}_1 \subseteq B_1^N(0)$, and for each model class, infinitely many computational problems $\{\Xi_\beta, \Omega, \mathcal{M}, \Lambda\}$, where the objects are as described in Equations* (5.2)–(5.5)*, such that there exists a set $\hat{\Lambda} \in \mathcal{L}^1(\Lambda)$, giving Δ_1-information, such that for the corresponding computational problem $\{\Xi_\beta, \Omega, \mathcal{M}, \hat{\Lambda}\}$, we have that there exists no algorithm, not even randomised (with probability $p > 1/2$), that approximates an optimal pair of interval neural networks $(\underline{\phi}, \overline{\phi}) \in \Xi_\beta(\mathcal{T}, \theta_\Phi)$, in the sense of* (3.4)*, for all elements $(\mathcal{T}, \theta_\Phi) \in \Omega$ to accuracy $\kappa/2$.*

6. Proof of the main result

6.1. A few preliminary results for the proof of Theorem 5.8.

Lemma 6.1. *Let $A \in \mathbb{R}^{m\times N}$ with $1 \leq \mathrm{rank}(A) < N$. Furthermore, let $\kappa, \beta > 0$, $\ell \geq 2$, $\mathcal{U}_1 \subset \mathbb{R}^N$, and let $\mathcal{NN}_\ell$ be a class of neural networks that satisfies Assumption* 2.1. *Assume that there exists two elements $x^{(1)}, x^{(2)} \in \mathcal{U}_1$, such that $Ax^{(1)} = Ax^{(2)}$ and such that $\|x^{(1)} - x^{(2)}\|_2^2 = \kappa$. Moreover, let $\mathcal{T}_0 = \{(x^{(1)}, Ax^{(1)}), (x^{(2)}, Ax^{(2)})\}$ and let $\mathcal{T}_1 \in T_{\ell-2,\mathcal{U}_1}$ be any training set where each of the second coordinates are distinct and not equal to $Ax^{(1)}$. With $\mathcal{T} = \mathcal{T}_0 \cup \mathcal{T}_1$, let $\Phi \in \mathcal{W}(\mathcal{T}, \mathcal{NN}_\ell)$, where $\mathcal{W}(\mathcal{T}, \mathcal{NN}_\ell)$ is as defined in* (2.7)*, and let $\mathcal{NN}_\Phi^u$ and $\mathcal{NN}_\Phi^o$ be the the optimization classes defined in* (2.8)*. Then*

$$\min_{\underline{\phi}\in\mathcal{NN}_\Phi^u, \overline{\phi}\in\mathcal{NN}_\Phi^o} F_{\mathcal{T}_0\cup\mathcal{T}_1,\beta}(\underline{\phi}, \overline{\phi}) = \min_{\underline{\phi}\in\mathcal{NN}_\Phi^u, \overline{\phi}\in\mathcal{NN}_\Phi^o} F_{\mathcal{T}_0,\beta}(\underline{\phi}, \overline{\phi}) \leq 2\beta\sqrt{N\kappa}. \tag{6.1}$$

Proof. We start by arguing for the first equality in (6.1). Since all the second coordinates in $\mathcal{T}_1$ are distinct, we must have that the pre-trained neural network $\Phi \in \mathcal{W}(\mathcal{T}, \mathcal{NN}_\ell)$ interpolates all the points in $\mathcal{T}_1$, this is because $\Phi \in \mathcal{W}(\mathcal{T}, \mathcal{NN}_\ell)$ implies that the neural network Φ has an excellent performance on all elements in $\pi_2(\mathcal{T})$, which means that Φ maps $y = Ax$ to the unique x in the case where the elements in $\pi_2(\mathcal{T}_1)$ are distinct. Thus, by part (b) in Assumption 2.1, we get that there exists $\underline{\phi} \in \mathcal{NN}_\Phi^u$ and $\overline{\phi} \in \mathcal{NN}_\Phi^o$ that for each $(x, y) \in \mathcal{T}_1$, maps $y \mapsto x$, this yields that the objective function $F_{\mathcal{T}_1,\beta}$ is zero on the set $\mathcal{T}_1$. Clearly, we can choose neural networks with this property on the extended training set $\mathcal{T}_0 \cup \mathcal{T}_1$ without affecting the optimal value of $F_{\mathcal{T}_0\cup\mathcal{T}_1,\beta}$, since we are minimizing over a class of networks which is ℓ-interpolatory. This means that the optimal neural networks interpolate all the data in $\mathcal{T}_1$ resulting in the first equality.

To prove the inequality in (6.1), let $y = Ax^{(1)} = Ax^{(2)}$ and let $(\underline{\psi}, \overline{\psi}) \in \mathcal{NN}_\Phi^u \times \mathcal{NN}_\Phi^o$ be such that $\overline{\psi}(y) = \max\{x^{(1)}, x^{(2)}\}$ and $\underline{\psi}(y) = \min\{x^{(1)}, x^{(2)}\}$. We then have that

$$\min_{\underline{\phi}\in\mathcal{NN}_\Phi^u, \overline{\phi}\in\mathcal{NN}_\Phi^o} F_{\mathcal{T}_0,\beta}(\underline{\phi}, \overline{\phi}) \leq F_{\mathcal{T}_0,\beta}(\underline{\psi}, \overline{\psi}) = 2\beta \| \max\{x^{(1)}, x^{(2)}\} - \min\{x^{(1)}, x^{(2)}\}\|_1$$

$$= 2\beta \sum_{i=1}^N |x_i^{(1)} - x_i^{(2)}| = 2\beta \|x^{(1)} - x^{(2)}\|_1 \leq 2\beta\sqrt{N}\|x^{(1)} - x^{(2)}\|_2 = 2\beta\sqrt{N\kappa},$$

where we in the last equality used the assumption that $\|x^{(1)} - x^{(2)}\|_2 = \sqrt{\kappa}$. □

Lemma 6.2. *Let $\kappa > 0$, $c \in (0, 1]$, and let $a \in \mathbb{R}^n$ be vector with non-negative entries, satisfying $\|a\|_2^2 \geq c\kappa/2$. Then $\tilde{b} = -\sqrt{\frac{c\kappa}{2}}\frac{1}{\|a\|_2}a$ is the unique minimizer of the optimization problem,*

$$\min_{b\in\mathbb{R}^n} \|a + b\|_2^2 \quad \textit{subject to} \quad \|b\|_2^2 \leq c\kappa/2. \tag{6.2}$$

Proof. Let $t \in [0, 1]$, we then start by solving the following optimization problem

$$\min_{b\in\mathbb{R}^n} \|a + b\|_2^2 \quad \text{subject to} \quad \|b\|_2^2 = t\frac{c\kappa}{2}. \tag{6.3}$$

The Lagrange function for (6.3) is given by $L(b, \lambda) = \|a + b\|_2^2 + \lambda(\|b\|_2^2 - tc\kappa/2)$. Now, all the extremal points (b, λ) must satisfy the following system of equations, given by the method of Lagrange multipliers,

$$2(a + b) + 2\lambda b = 0 \quad \text{and} \quad \|b\|_2^2 - t\tfrac{c\kappa}{2} = 0.$$

From the first equality we have that $a = -(1+\lambda)b$. Using the second equality gives $\|a\|_2^2 = (1+\lambda)^2 tc\kappa/2$. This implies that $\lambda = -1 \pm (\sqrt{tc\kappa/2})^{-1}\|a\|_2$. Thus, we get the extremal points $b^1 = -\tilde{b}\sqrt{t}$ and $b^2 = \tilde{b}\sqrt{t}$, with $\tilde{b} = -\sqrt{\frac{c\kappa}{2}}\frac{1}{\|a\|_2}a$. Since there are no other solutions to this system, and a has non-negative entries and satisfies $\|a\|_2^2 \geq c\kappa/2$ we get that b^1 is a global maximum and that b^2 is a global minimum. We conclude that b^2 is the unique minimizer, and obtain the general result by observing that

$$\min_{b\in\mathbb{R}^n}\{\|a + b\|_2^2 : \|b\|_2^2 \leq \frac{c\kappa}{2}\} = \inf_{t\in[0,1]} \min_{b\in\mathbb{R}^n}\{\|a + b\|_2^2 : \|b\|_2^2 = t\frac{c\kappa}{2}\} = \inf_{t\in[0,1]} \|a + \tilde{b}\sqrt{t}\|_2^2 = \|a + \tilde{b}\|_2^2.$$

□

Lemma 6.3. *Let $\kappa > 0$ and $c \in (0, 1]$, and let $a, b^1, b^2 \in \mathbb{R}^n$ be such that*

(i) $\|b^i\|_2^2 \leq c\kappa/2$ *for* $i = 1, 2$,

(ii) $\|a\|_2^2 = 2\kappa$.

Then

$$\| \max\{a + b^1, 0\}\|_2^2 + \| \max\{-a + b^2, 0\}\|_2^2 \geq \left(2 + c - \sqrt{8c}\right)\kappa. \tag{6.4}$$

Proof. We start by splitting the vector a into its' positive and negative parts a^+ and a^-, where $a^+ := \max\{a, 0\}$, and $a^- := \max\{-a, 0\}$. We note that $a = a^+ - a^-$ and that the entries of a^- are non-negative. Furthermore, since the two vectors a^+ and a^- have disjoint support we have that $\|a^+\|_2^2 + \|a^-\|_2^2 = \|a\|_2^2 = 2\kappa$ from condition (ii). To ease the notation in what follows, we let $S^+ = \text{supp}(a^+) \subset \{1, \ldots, N\}$ denote the denote support set of a^+, and we let $S_c^+ = \{1, \ldots, N\} \setminus S^+$ denote its' complement. Furthermore, for an index set $S \subset \{1, \ldots, N\}$ and a vector $x \in \mathbb{R}^N$, we let x_S be the vector with entries $(x_S)_i = x_i$ if $i \in S$ and zero otherwise.

Consider the following two optimization problems,

$$\min_{b\in\mathbb{R}^n} \| \max\{a + b, 0\}\|_2^2 \quad \text{subject to} \quad \|b\|_2^2 \leq \frac{c\kappa}{2}, \tag{Q_1}$$

$$\min_{b\in\mathbb{R}^n} \| \max\{-a + b, 0\}\|_2^2 \quad \text{subject to} \quad \|b\|_2^2 \leq \frac{c\kappa}{2}. \tag{Q_2}$$

In what follows we find a lower bound for the optimal value of each of these optimization problems. Once we have these bounds, we can use these to compute the lower bound in (6.4).

We start by considering (Q_1), and consider the two cases $\|a^+\|_2^2 \leq c\kappa/2$ and $\|a^+\|_2^2 > c\kappa/2$ separately. If $\|a^+\|_2^2 \leq c\kappa/2$, it is easy to see that $b = -a^+$ is a feasible point for (Q_1), and that $b = -a^+$ attains the minimum value 0. On the other hand, if $\|a^+\|_2^2 > c\kappa/2$ and $b \in \mathbb{R}^N$ satisfies $\|b\|_2^2 \leq c\kappa/2$, we have that

$$\begin{aligned}\|\max\{a+b,0\}\|_2^2 &= \|\max\{a^+ + b_{S^+} + b_{S_c^+} - a^-, 0\}\|_2^2 \geq \|\max\{a^+ + b_{S^+}, 0\}\|_2^2 \\ &\geq \left\|a^+ - \sqrt{\tfrac{c\kappa}{2}}\tfrac{1}{\|a^+\|_2}a^+\right\|_2^2 = \left(1 - \sqrt{\tfrac{c\kappa}{2}}\tfrac{1}{\|a^+\|_2}\right)^2 \|a^+\|_2^2 \\ &= \frac{c\kappa}{2} + \|a^+\|_2\left(\|a^+\|_2 - \sqrt{2c\kappa}\right) > 0.\end{aligned}$$

Here the first inequality holds since the support of $a^+ + b_{S^+}$ and $b_{S_c^+} - a^-$ are disjoint. The second inequality holds because $-\sqrt{\frac{c\kappa}{2}}\frac{1}{\|a^+\|_2}a^+$ is supported on S^+, it is a feasible point, and from Lemma 6.2 we know that it attains the minimum value over all choices of b_{S^+} satisfying the constraint $\|b_{S^+}\|_2^2 \leq c\kappa/2$. Finally, the last inequality holds since $\|a^+\|_2 > \sqrt{c\kappa/2}$.

From symmetry it is clear that if $\|a^-\|_2^2 \leq c\kappa/2$, then the objective function of (Q_2) is zero, and if $\|a^-\|_2^2 > c\kappa/2$, we have that

$$\min_{b\in\mathbb{R}^N}\left\{\|\max\{-a+b,0\}\|_2^2 : \|b\|_2^2 \leq c\kappa/2\right\} = \frac{c\kappa}{2} + \|a^-\|_2\left(\|a^-\|_2 - \sqrt{2c\kappa}\right) > 0$$

Next, observe that due to assumption (i), we have that

$$\begin{aligned}&\|\max\{a+b^1\}\|_2^2 + \|\max\{-a+b^2,0\}\|_2^2 \\ &\geq \min_{b\in\mathbb{R}^N}\left\{\|\max\{a+b,0\}\|_2^2 : \|b\|_2^2 \leq c\kappa/2\right\} + \min_{b\in\mathbb{R}^N}\left\{\|\max\{-a+b,0\}\|_2^2 : \|b\|_2^2 \leq c\kappa/2\right\}\end{aligned} \tag{6.5}$$

From assumption (ii) we have that $\|a\|_2^2 = \|a^+\|_2^2 + \|a^-\|_2^2 = 2\kappa$, which implies that we cannot have that $\|a^+\|_2^2 \leq c\kappa/2$ and $\|a^-\|_2^2 \leq c\kappa/2$ at the same time. Thus, to bound the expression on the right hand side in (6.5), we need to consider the three cases,

(A) $\|a^+\|_2^2 > c\kappa/2$ and $\|a^-\|_2^2 > c\kappa/2$,
(B) $\|a^-\|_2^2 \leq c\kappa/2$,
(C) $\|a^+\|_2^2 \leq c\kappa/2$.

We start by considering (A), so suppose that this holds. Observe that $\|a^-\|_2 = \sqrt{2\kappa - \|a^+\|_2^2}$ since $\|a\|_2^2 = 2\kappa$ and that $c\kappa/2 < \|a^+\|_2^2 < 2\kappa - c\kappa/2$ due to assumption (A). From above, we know that the minimum value of both (Q_1) and (Q_2) are positive in this case. Combining this fact with (6.5), we see that

$$\begin{aligned}&\|\max\{a+b^1\}\|_2^2 + \|\max\{-a+b^2,0\}\|_2^2 \\ &\geq c\kappa + \|a^+\|_2\left(\|a^+\|_2 - \sqrt{2c\kappa}\right) + \|a^-\|_2\left(\|a^-\|_2 - \sqrt{2c\kappa}\right) \\ &= c\kappa + \|a^+\|_2\left(\|a^+\|_2 - \sqrt{2c\kappa}\right) + \sqrt{2\kappa - \|a^+\|_2^2}\left(\sqrt{2\kappa - \|a^+\|_2^2} - \sqrt{2c\kappa}\right) \\ &= (2+c)\kappa - \sqrt{2c\kappa}\left(\|a^+\|_2 + \sqrt{2\kappa - \|a^+\|_2^2}\right) \\ &\geq (2+c)\kappa - \sqrt{8c}\kappa = (2+c-\sqrt{8c})\kappa.\end{aligned}$$

Where the last inequality follows from the fact that function $f(x) = x + \sqrt{2\kappa - x^2}$ on the domain $\sqrt{c\kappa/2} < x < \sqrt{2\kappa - c\kappa/2}$ attains it maximum in $x = \sqrt{\kappa}$. This shows the lower bound in (6.4), whenever (A) holds.

Now, suppose that (B) holds. Since $\|a^-\|_2^2 \leq c\kappa/2$, we have that $\|a^+\|_2 \geq \sqrt{2\kappa - c\kappa/2} > \sqrt{c\kappa/2}$. Furthermore, since $\|a^-\|_2^2 \leq c\kappa/2$ we know that the minimum value of (Q_2) is zero, and the lower bound for (6.5) becomes,

$$\begin{aligned}\|\max\{a+b^1\}\|_2^2 + \|\max\{-a+b^2,0\}\|_2^2 \geq \frac{c\kappa}{2} + \|a^+\|_2\left(\|a^+\|_2 - \sqrt{2c\kappa}\right) &\geq \left(2 - \sqrt{4c - c^2}\right)\kappa \\ &\geq (2 + c - \sqrt{8c})\kappa\end{aligned}$$

where we used that $\|a^+\|_2 \geq \sqrt{2\kappa - c\kappa/2}$ for the second to last inequality. By symmetry we get the same lower bound if we assume that (C) holds. We, therefore, conclude that the lower bound in (6.4) holds. □

6.2. **Breakdown epsilons – The key to proving the impossibility statements.** The key to proving the impossibility statements in Theorem 3.2 is via the breakdown epsilons introduced in [12]. In an intuitive sense we have the following definition: The *strong breakdown epsilon* is the largest ϵ, for which any algorithm fails to achieve ϵ accuracy on a given computational problem for at least one input. We now present the precise versions of the breakdown epsilons, both in the deterministic and randomised setting.

Definition 6.4 (Strong breakdown epsilons [12])**.** Given a computational problem $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$, we define;

(i) the *strong breakdown epsilon* by

$$\epsilon_{\mathrm{B}}^{\mathrm{s}} \coloneqq \sup\{\epsilon \geq 0 : \forall \text{ general algorithms } \Gamma, \exists \iota \in \Omega \text{ such that } \mathrm{dist}_{\mathcal{M}}(\Gamma(\iota), \Xi(\iota)) > \epsilon\}, \text{ and}$$

(ii) the *probabilistic strong breakdown epsilon* by $\epsilon_{\mathbb{P}\mathrm{B}}^{\mathrm{s}} \colon [0,1) \to \mathbb{R}$ by

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

where $\mathrm{dist}_{\mathcal{M}}(\Gamma(\iota), \Xi(\iota)) = \inf_{\xi \in \Xi(\iota)} d_{\mathcal{M}}(\Gamma(\iota), \xi)$.

We need the following important proposition from [12] to prove the computational breakdown in Theorem 5.8.

Proposition 6.5 (Proposition 9.5 in [12])**.** *Let $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$ be a computational problem with $\Lambda = \{f_k | k \in \mathbb{N},\ k \leq |\Lambda|\ \}$ countable. Suppose that there are two sequences $\{\iota_n^1\}_{n\in\mathbb{N}}, \{\iota_n^2\}_{n\in\mathbb{N}} \subseteq \Omega$ satisfying the following conditions:*

a) There are sets $S_1, S_2 \subseteq \mathcal{M}$ and $\kappa > 0$ such that $\inf_{x_1\in S_1, x_2 \in S_2} d_{\mathcal{M}}(x_1, x_2) \geq \kappa$ and $\Xi(\iota_n^j) \subseteq S_j$ for $j = 1, 2$.

b) For every $k \leq |\Lambda|$ there is a $c_k \in \mathbb{C}$ such that $|f_k(\iota_n^j) - c_k| \leq 1/4^n$ for all $n \in \mathbb{N}$ and $j = 1, 2$.

c) There is $\iota_0 \in \Omega$ such that for every $k \leq |\Lambda|$ we have that b) is satisfied with $c_k = f_k(\iota_0)$.

Then there exists $\tilde{\Lambda} \in \mathcal{L}(\Lambda)$ such that $\epsilon_B^s \geq \epsilon_{\mathbb{P}hB}^s(p) \geq \kappa/2$ for $p \in [0, 1/2)$ and $\epsilon_{\mathbb{P}B}^s \geq \kappa/2$ for $p \in (0, 1/3)$ for the computational problem $\{\Xi, \Omega, \mathcal{M}, \tilde{\Lambda}\}$.

Remark 6.6. Note that the above proposition holds also in the Markov model. Since this is the main mechanism for the impossibility result in our main results, Theorem 3.2 holds also in the Markov model. The mathematical theory for converting between the two models (Markov model and computable Δ_1-information) was developed in [30, 64].

We remind the reader that we are interested in proving that the computational breakdown for any training set $\mathcal{T}$ happens at a point contained in $\pi_2(\mathcal{T})$, where $\pi_2(\mathcal{T}) = \{y \in \mathbb{R}^m \,:\, (x, y) \in \mathcal{T}\}$. More precisely, we wish to prove that for any $\kappa \in (0, 1/3)$, there exists domains Ω such that, for any algorithm Γ, there exists at least one pair $(\mathcal{T}, \theta_\Phi) \in \Omega$ such that $\mathrm{dist}_{\mathcal{T}}(\Gamma(\mathcal{T}, \theta_\Phi), \Xi_\beta(\mathcal{T}, \theta_\Phi)) \coloneqq \inf_{(\underline{\phi}, \overline{\phi}) \in \Xi_\beta(\mathcal{T}, \theta_\Phi)} d_{\mathcal{T}}(\Gamma(\mathcal{T}, \theta_\Phi), (\underline{\phi}, \overline{\phi}))$ $\geq \kappa/2$, with

$$d_{\mathcal{T}}((\underline{\phi}, \overline{\phi}), (\underline{\psi}, \overline{\psi})) = \max\left\{ \sup_{y\in\pi_2(\mathcal{T})} \|\overline{\phi}(y) - \overline{\psi}(y)\|_2^2, \sup_{y\in\pi_2(\mathcal{T})} \|\underline{\phi}(y) - \underline{\psi}(y)\|_2^2 \right\}. \tag{6.6}$$

We therefore proceed by proving the computational breakdown for the alternative inverse problem $\{\Xi_\beta, \Omega, \mathcal{M}', \hat{\Lambda}\}$ where $\mathcal{M}' = \mathcal{NN}_\Phi^u \times \mathcal{NN}_\Phi^o$, but with the metric $d_{\mathcal{M}}((\underline{\phi}, \overline{\phi}), (\underline{\psi}, \overline{\psi}))$, introduced in (5.4), replaced by the metric $d_{\mathcal{M}'}((\underline{\phi}, \overline{\phi}), (\underline{\psi}, \overline{\psi}))$ given by

$$d_{\mathcal{M}'}((\underline{\phi}, \overline{\phi}), (\underline{\psi}, \overline{\psi})) \coloneqq \max\left\{ \sup_{y\in\mathcal{F}_\Omega} \|\overline{\phi}(y) - \overline{\psi}(y)\|_2^2, \sup_{y\in\mathcal{F}_\Omega} \|\underline{\phi}(y) - \underline{\psi}(y)\|_2^2 \right\}, \tag{6.7}$$

with $\mathcal{F}_\Omega = \bigcap_{\mathcal{T}\in\Omega} \pi_2(\mathcal{T})$, where we identify the neural networks that are equal on $\mathcal{F}_\Omega$. We observe that proving that $\epsilon^s_{\mathrm{B}} > \kappa/2$ for $\{\Xi_\beta, \Omega, \mathcal{M}', \hat{\Lambda}\}$ immediately implies that there exists at least one pair $(\mathcal{T}, \theta_\Phi) \in \Omega$ such that $\mathrm{dist}_{\mathcal{T}}(\Gamma(\mathcal{T}, \theta_\Phi), \Xi_\beta(\mathcal{T}, \theta_\Phi)) \geq \kappa/2$, when reading inexact input.

Proposition 6.7. *Let $m, N \in \mathbb{N}$ with $N \geq 2$, and let $A \in \mathbb{R}^{m\times N}$ be such that $1 \leq \mathrm{rank}(A) < N$. Let $\ell \geq 2$, $\kappa \in (0, 1/3)$ and let $\mathcal{NN}_\ell$ be a class of neural networks that satisfies Assumption* 2.1. *Then for any $\beta \in (0, \sqrt{\kappa}/(4\sqrt{2N}))$ there exists infinitely many model classes $\mathcal{U}_1 \subseteq B^N_1(0)$, and for each model class, infinitely many choices for Ω, giving rise to infinitely many computational problems $\{\Xi_\beta, \Omega, \mathcal{M}, \Lambda\}$, where the objects are as described in Equations* (5.2)–(5.5)*, such that there exists a set $\hat{\Lambda} \in \mathcal{L}^1(\Lambda)$, giving Δ_1-information, such that $\epsilon^s_{\mathrm{B}} \geq \epsilon^s_{\mathbb{P}\mathrm{B}}(p) \geq \kappa/2$ for $p \in [0, 1/2)$ for the computational problem $\{\Xi_\beta, \Omega, \mathcal{M}', \hat{\Lambda}\}$, where $\mathcal{M}'$ is the metric defined in* (6.7)*.*

Proof. We prove the above result by constructing a set $\mathcal{U}_1 \subset \mathbb{R}^N$ and a domain Ω, as described in (5.2), with sequences $\{(\iota^1_n, \Phi^1_n)\}_{n\in\mathbb{N}}, \{(\iota^2_n, \Phi^2_n)\}_{n\in\mathbb{N}} \subseteq \Omega$ that satisfy the following conditions:

(a) There are sets $S^1, S^2 \subset \mathcal{M}'$ such that $\inf_{(\underline{\phi},\overline{\phi})\in S^1,(\underline{\psi},\overline{\psi})\in S^2} d_{\mathcal{M}'}((\underline{\phi},\overline{\phi}),(\underline{\psi},\overline{\psi})) \geq \kappa$ and $\Xi_\beta(\iota^j_n, \Phi^j_n) \subset S^j$ for $j = 1, 2$.

(b) For every $k \in |\Lambda|$ there is a $c_k \in \mathbb{C}$ such that $|f_k(\iota^j_n, \Phi^j_n) - c_k| \leq 1/4^n$, for $j = 1, 2$ and all $n \in \mathbb{N}$.

(c) There is an $(\iota^0, \Phi^0) \in T_{\ell,\mathcal{U}_1} \times \mathcal{W}(\iota_0, \mathcal{NN}_\ell)$ such that for every $k \leq |\Lambda|$ we have that (b) is satisfies with $c_k = f_k(\iota^0, \Phi^0)$.

(d) There is an $(\iota^0, \Phi^0) \in T_{\ell,\mathcal{U}_1} \times \mathcal{W}(\iota^0, \mathcal{NN}_\ell)$ for which condition (c) holds and additionally $(\iota^2_n, \Phi^2_n) = (\iota^0, \Phi^0)$, for all $n \in \mathbb{N}$.

Then it follows from Proposition 6.5, that there exists a $\hat{\Lambda} \in \mathcal{L}^1(\Lambda)$, such that $\epsilon^s_{\mathrm{B}} \geq \epsilon^s_{\mathbb{P}\mathrm{B}}(p) \geq \epsilon^s_{\mathbb{P}\mathrm{hB}}(p) \geq \kappa/2$ for $p \in [0, 1/2)$ for the computational problem $\{\Xi_\beta, \Omega, \mathcal{M}', \hat{\Lambda}\}$.

We proceed by constructing $\mathcal{U}_1$ and start by making a few observations. The kernel of A must be non-trivial since $\mathrm{rank}(A) < N$. We can, therefore, pick a non-zero vector $v \in \ker(A)$ with squared norm $\|v\|^2_2 = 8\kappa$. Furthermore, since $\mathrm{rank}(A) \geq 1$, we can pick a non-zero vector $w \in \ker(A)^\perp$ with $\|w\|_2 = (3 - 2\sqrt{2})\kappa$.

Let $\mathcal{S} = \{tw : t \in [-1, -1/2]\}$ and let the model set $\mathcal{U}_1 \coloneqq \{v + tw : t \in [0, 1]\} \cup \{0\} \cup \mathcal{S}$. For $\ell > 2$ let $\mathcal{T}_b \subset A(\mathcal{S}) \times \mathcal{S}$ be a set with cardinality $\ell - 2$, where all of the first components are distinct. We note that this implies that all the second components are distinct as well, since $\mathcal{S} \subset \ker(A)^\perp$. If $\ell = 2$, we let $\mathcal{T}_b = \emptyset$. Given the above setup, we can now define the two sequences $\{\iota^1_n\}_{n\in\mathbb{N}}$ and $\{\iota^2_n\}_{n\in\mathbb{N}}$ as follows. Let

$$\iota^1_n = (\mathcal{T}_b, (0, 0), (v + \frac{1}{4^n}w, \frac{1}{4^n}Aw)) \quad \text{and} \quad \iota^2_n = (\mathcal{T}_b, (0, 0), (v, 0)), \quad n \in \mathbb{N}. \tag{6.8}$$

We observe that the elements in ι^1_n and ι^2_n are listed such that all the equal elements land on the same index and such that $(x^{(\ell)}, A(x^{(\ell)})) = (v + \frac{1}{4^n}w, \frac{1}{4^n}Aw) \in \iota^1_n$ and $(x^{(\ell)}, A(x^{(\ell)})) = (v, 0) \in \iota^2_n$. It is clear that $\iota^j_n \subseteq T_{\ell,\mathcal{U}_1}$ for all $n \in \mathbb{N}$ and $j = 1, 2$, and that $\mathcal{U}_1 \subseteq B^N_1(0)$.

Next, since, by Assumption 2.1, $\mathcal{NN}_\ell$ is an ℓ-interpolatory class of neural networks and ι^1_n consists of ℓ distinct points, there exists a neural network $\Phi^1_n \in \mathcal{NN}_\ell$ that interpolates all the points in ι^1_n for each $n \in \mathbb{N}$. It then follows from the definition of $\mathcal{W}(\iota^1_n, \mathcal{NN}_\ell)$ in (2.7) that $\Phi^1_n \in \mathcal{W}(\iota^1_n, \mathcal{NN}_\ell)$ for all $n \in \mathbb{N}$. In the case of ι^2_n the map $\mathcal{V}_{\iota^2_n}(y)$ in (2.6) is multivalued at the point $y = 0$. However, we observe that any map $\Psi_1 \in \mathcal{NN}_\ell$ that interpolates all the points in $\mathcal{T}_b$ and that satisfies $\Psi_1(0) = 0$ has the property that $\Psi_1 \in \mathcal{W}(\iota^2_n, \mathcal{NN}_\ell)$. In particular, this gives us that $\Phi^1_n \in \mathcal{W}(\iota^2_n, \mathcal{NN}_\ell)$, therefore, by simply setting $\Phi^2_n = \Phi^1_n$ for each $n \in \mathbb{N}$, we may conclude that $\Phi^2_n \in \mathcal{W}(\iota^2_n, \mathcal{NN}_\ell)$ for each $n \in \mathbb{N}$.

With this in place, we let $\Omega = \{(\iota^1_n, \Phi^1_n)\}_{n\in N} \cup \{(\iota^2_n, \Phi^2_n)\}_{n\in\mathbb{N}}$ in what follows. By the arbitrary choice of the elements in $\mathcal{T}_b$ and $w \in \ker(A)^\perp$ it is clear that there exists uncountably many choices for the model class $\mathcal{U}_1$, and for each choice of $\mathcal{U}_1$, there exists uncountably many choices for Ω.

With the above sequences well defined, we proceed by showing that these sequences satisfy the claims in (a)-(d) above. We start by considering (a), with $S^j = \bigcup_{n=1}^\infty \Xi_\beta(\iota^j_n, \Phi^j_n)$, for $j = 1, 2$. We observe that

any pair of ℓ-interpolatory NNs, which interpolates all the training data in ι_n^1, attains zero training loss on $F_{\iota_n^1,\beta}$. In particular, this implies that any pair $(\underline{\phi},\overline{\phi}) \in \Xi_\beta(\iota_n^1,\Phi_n^1)$ interpolates the data in ι_n^1.

Next, we consider the constant sequence ι_n^2 where both $(0,0) \in \iota_n^2$ and $(v,0) \in \iota_n^2$ have identical y-coordinates, and all other other y-coordinates are distinct. Furthermore, since $\|v-0\|_2^2 = 8\kappa$ we know from Lemma 6.1, with $\kappa' = 8\kappa$, that for any pair $(\underline{\psi},\overline{\psi}) \in \Xi_\beta(\iota_n^2,\Phi_n^2)$, we have $F_{\iota_n^2,\beta}(\underline{\psi},\overline{\psi}) \leq 4\beta\sqrt{2N\kappa} < \kappa$, where the last inequality follows from the fact that $\beta < \sqrt{\kappa}/(4\sqrt{2N})$.

It remains to show that

$$d_{\mathcal{M}'}((\underline{\phi},\overline{\phi}),(\underline{\psi},\overline{\psi})) = \max\left\{\sup_{y\in\mathcal{F}_\Omega} \|\overline{\phi}(y)-\overline{\psi}(y)\|_2^2,\ \sup_{y\in\mathcal{F}_\Omega} \|\underline{\phi}(y)-\underline{\psi}(y)\|_2^2\right\} \geq \kappa \tag{6.9}$$

for all $(\underline{\phi},\overline{\phi}) \in S^1$ and $(\underline{\psi},\overline{\psi}) \in S^2$. Assume for a contradiction that this is not the case. By the above argumentation we know that $\underline{\phi}(0) = \overline{\phi}(0) = 0$ and by the assumption that $d_{\mathcal{M}'}((\underline{\phi},\overline{\phi}),(\underline{\psi},\overline{\psi})) < \kappa$ we get that $\|\overline{\psi}(0)-0\|_2^2 = \|\overline{\psi}(0)-\overline{\phi}(0)\|_2^2 < \kappa$, and similarly we get that $\|\underline{\psi}(0)-0\|_2^2 = \|\underline{\psi}(0)-\underline{\phi}(0)\|_2^2 < \kappa$. Next, we observe that

$$\begin{aligned} F_{\iota_n^2,\beta}(\underline{\psi},\overline{\psi}) &\geq \|\max\{v-\overline{\psi}(0),0\}\|_2^2 + \|\max\{\underline{\psi}(0)-v,0\}\|_2^2 \\ &= \|\max\{v-0+\overline{\phi}(0)-\overline{\psi}(0),0\}\|_2^2 + \|\max\{\underline{\psi}(0)-\underline{\phi}(0)+0-v,0\}\|_2^2 \geq 2\kappa \end{aligned}$$

where the last inequality follows from Lemma 6.3, with $\kappa' = 4\kappa$, $c = 1/2$, $b^1 = \overline{\phi}(0)-\overline{\psi}(0)$, $b^2 = \underline{\phi}(0)-\underline{\psi}(0)$ and $a = v-0$. However, this is a contradiction, as we know from the above discussion that $F_{\iota_n^2,\beta}(\underline{\psi},\overline{\psi}) < \kappa$. This establishes (6.9), and thereby also (a).

To establish items (b)-(d), we see that it is sufficient to establish (b) with $c_k = f(\iota_n^2,\Phi_n^2)$ for any $n \in \mathbb{N}$, since ι_n^2 is constant for all $n \in \mathbb{N}$. We notice that, for each $n \in \mathbb{N}$, we have that $\Phi_n^1 = \Phi_n^2$, and the elements in $\mathcal{T}_b$ and $(0,0)$ coincide in ι_n^1 and ι_n^2 and are listed on the same index. Moreover we have that $(x^{(\ell)},A(x^{(\ell)})) = (v+\frac{1}{4^n}e, A(v+\frac{1}{4^n}e)) \in \iota_n^1$ and $(x^{(\ell)},A(x^{(\ell)})) = (v,0) \in \iota_n^2$, thus we only need to show that the criteria in part (b), for $f_{x,i}^\ell$ and $f_{y,j}^\ell$ where $i = 1,\dots,N$, $j = 1,\dots,m$. We start by $f_{y,j}^\ell$. For each j have that $|f_{y,j}^\ell(\iota_n^1) - f_{y,j}^\ell(\iota_n^2)| = |A(\frac{1}{4^n}e)_j - 0| = \frac{1}{4^n}|A(e)_j| \leq \frac{1}{4^n}$. A similar calculation show the result for $f_{x,i}^\ell$, but we omit the details. This establishes (b)-(d) above, and concludes the proof. □

6.3. **Proof of Theorem 5.8.** *Proof of Theorem* 5.8. By Proposition 6.7, and the definition of the breakdown epsilons ϵ_{B}^s and $\epsilon_{\mathbb{P}\mathrm{B}}^s(p)$, it follows that there exists a set $\hat{\Lambda} \in \mathcal{L}^1(\Lambda)$, giving Δ_1-information for $\{\Xi_\beta,\Omega,\mathcal{M}',\Lambda\}$, such that, for any algorithm $\Gamma : \hat{\Omega} \to \mathcal{M}'$, there exists a pair $(\mathcal{T}_1,\theta_\Phi^1) \in \Omega$ such that $\mathrm{dist}_{\mathcal{M}'}(\Xi_\beta(\mathcal{T}_1,\theta_\Phi^1),\Gamma(\mathcal{T}_1,\theta_\Phi^1)) \geq \kappa/2$, and that, for any randomised algorithm $\Gamma^{ran} : \hat{\Omega} \to \mathcal{M}$, there exists a pair $(\mathcal{T}_2,\theta_\Phi^2) \in \Omega$ such that $\mathbb{P}(\mathrm{dist}_{\mathcal{M}'}(\Xi_\beta(\mathcal{T}_2,\theta_\Phi^2),\Gamma(\mathcal{T}_2,\theta_\Phi^2)) \geq \kappa/2) \geq 1/2$. The statement of Theorem 5.8 follows from this. □

APPENDIX A. RELU-NETWORKS SATISFY ASSUMPTION 2.1

In this appendix we present a proposition that illustrates that Assumption 2.1 is satisfied by any class of ReLU-networks, with input dimension m and output dimension N, of fixed depth greater than or equal to 2. We start by showing a lemma the that asserts that Assumption 2.1 is satisfied for ReLu-networks of depth equal to 2 with output dimension $N = 1$. We the assert then general result by a corollary.

We start by introducing the following notation: For a vector $z \subseteq \mathbb{R}^m$, $\pi_1(z)$ is the projection onto the first coordinate defined as:

$$\pi_1(z) := \{z_1 \mid z = (z_1,\dots,z_m) \in \mathbb{R}^m\}, \tag{A.1}$$

We repeat the definition of a neural network and Assumption 2.1 for completeness.

Definition A.1. Let K be a natural number and let $\mathbf{N} := (N_0, N_1, \ldots, N_{K-1}, N_K) \in \mathbb{N}^{K+1}$. A neural network with dimension $(\mathbf{N}, K)$ is a map $\Phi : \mathbb{R}^{N_0} \to \mathbb{R}^{N_K}$ such that

$$\Phi(x) \coloneqq V^{(K)} \circ \sigma \circ V^{(K-1)} \circ \sigma \circ V^{(K-2)} \circ \cdots \circ \sigma \circ V^{(1)} x,$$

where, for $k = 1, \ldots, K$, the map $V^{(k)}$ is an affine map from $\mathbb{R}^{N_{k-1}} \to \mathbb{R}^{N_k}$, that is $V^{(k)}x^{(k)} = W^{(k)}x^{(k)} + b^{(k)}$, where $b^{(k)} \in \mathbb{R}^{N_k}$ and $W^{(k)} \in \mathbb{R}^{N_{k-1} \times N_k}$. The map $\sigma : \mathbb{R} \to \mathbb{R}$ is interpreted as a coordinate wise map and is called the *non-linearity* or *activation function*.

Assumption 2.1. For a given integer $\ell \geq 1$, we assume that $\mathcal{NN}_\ell$ is a class of neural networks such that for any collection $\mathcal{T} = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(\ell)}, y^{(\ell)})\} \subset \mathbb{R}^N \times \mathbb{R}^m$, where each y-coordinate is distinct, the following holds:

(i) (ℓ-interpolatory): There exists a neural network $\Psi \in \mathcal{NN}_\ell$, such that $\Psi(y) = x$ for each $(x, y) \in \mathcal{T}$.

(ii) For any choice of $x' \in \mathbb{R}^N$, any $k \in \{1, \ldots, \ell\}$ and any $\Psi \in \mathcal{NN}_\ell$ satisfying (i), there exist neural networks $\underline{\phi}, \overline{\phi} \in \mathcal{NN}_\ell$, such that
   (a) $\underline{\phi}(y) \leq \Psi(y) \leq \overline{\phi}(y)$ for all $y \in \mathbb{R}^m$, and
   (b) such that $\underline{\phi}(y) = \overline{\phi}(y) = x$ for all $(x, y) \in \mathcal{T} \setminus \{x^{(k)}, y^{(k)}\}$, and
   (c) $\underline{\phi}(y^{(k)}) = \min\{x', x^{(k)}\}$ and $\overline{\phi}(y^{(k)}) = \max\{x', x^{(k)}\}$.

Lemma A.2. *Let $m \in \mathbb{N}$ and let $N = 1$. Let $\mathcal{NN}^2_{ReLu}$ denote the class of neural networks, according to Definition* A.1*, where $K = 2$, $N_0 = m$, $N_1 \in \mathbb{N}$ and $N_2 = N = 1$, and where $\sigma : \mathbb{R} \to \mathbb{R}$ is given by*

$$\sigma(y) = ReLu(y) = \begin{cases} y & \text{if } x > 0 \\ 0 & \text{otherwise.} \end{cases}$$

Then $\mathcal{NN}^2_{ReLu}$ satisfies Assumption 2.1.

Proof. Let $\ell \in \mathbb{N}$ and let $\mathcal{T} \subset \mathbb{R} \times \mathbb{R}^m$ be as described in Assumption 2.1. More precisely, $\mathcal{T} = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(\ell)}, y^{(\ell)})\} \subset \mathbb{R} \times \mathbb{R}^m$, where each y-coordinate is distinct. We proceed by showing that there exists three neural networks $\Phi, \underline{\phi}, \overline{\phi} \in \mathcal{NN}^2_{\text{ReLu}}$ that satisfy points (i) and (ii) in Assumption 2.1. Indeed, let $k \in \{1, \ldots, \ell\}$ and let $x' \in \mathbb{R}$. Set $x^{(k),1} = \min\{x', x^{(k)}\}$ and $x^{(k),2} = \max\{x', x^{(k)}\}$. We then construct neural networks $\Phi, \underline{\phi}, \overline{\phi} \in \mathcal{NN}^2_{\text{ReLu}}$ such that

- Φ interpolates the points $\{(y^{(i)}, x^{(i)})\}_{i=1}^{\ell}$,
- $\underline{\phi}$ interpolates the points $\{(y^{(k)}, x^{(k),1})\} \cup [\{(y^{(i)}, x^{(i)})\}_{i=1}^{\ell-1} \setminus \{(y^{(k)}, x^{(k)})\}]$,
- $\overline{\phi}$ interpolates the points $\{(y^{(k)}, x^{(k),2})\} \cup [\{(y^{(i)}, x^{(i)})\}_{i=1}^{\ell-1} \setminus \{(y^{(k)}, x^{(k)})\}]$,

and such that $\underline{\phi}(y) \leq \Phi(y) \leq \overline{\phi}(y)$ for all $y \in \mathbb{R}^m$.

Let $\pi_1(y^{(i)}) = y_1^{(i)}$ be the projection onto the first coordinate of the vector $y^{(i)} \in \mathbb{R}^m$. We may assume without loss of generality that the points in $\mathcal{T}$ are ordered such that $y_1^{(1)} \leq y_1^{(2)} \leq \ldots, \leq y_1^{(\ell)}$. Moreover, if $y_1^{(i)} = y_1^{(i+1)}$ for some $i \in \{1, \ldots, \ell\}$ we may remove $y_1^{(i+1)}$ from the list, thereby, we may assume without loss of generality that $y_1^{(1)} < y_1^{(2)} < \ldots, < y_1^{(\ell)}$. We then construct piecewise linear functions $f : \mathbb{R} \to \mathbb{R}$, $\underline{f} : \mathbb{R} \to \mathbb{R}$, and $\overline{f} : \mathbb{R} \to \mathbb{R}$ such that

- f interpolates the points $\{(y_1^{(i)}, x^{(i)})\}_{i=1}^{\ell}$,
- $\underline{f}$ interpolates the points $\{(y_1^{(k)}, x^{(k),1})\} \cup [\{(y_1^{(i)}, x^{(i)})\}_{i=1}^{\ell-1} \setminus \{(y_1^{(k)}, x^{(k)})\}]$,
- $\overline{f}$ interpolates the points $\{(y_1^{(k)}, x^{(k),2})\} \cup [\{(y_1^{(i)}, x^{(i)})\}_{i=1}^{\ell-1} \setminus \{(y_1^{(k)}, x^{(k)})\}]$.

Indeed, let $\underline{f} : \mathbb{R} \to \mathbb{R}$, and $\overline{f} : \mathbb{R} \to \mathbb{R}$ be given by

$$\underline{f}(y) = \begin{cases} \frac{x^{(i+1)}-x^{(i)}}{y_1^{(i+1)}-y_1^{(i)}}(y - y_1^{(i)}) + x^{(i)} & \text{when } y \in [y_1^{(i)}, y_1^{(i+1)}) \text{ and } i \neq k, i \neq k+1 \\ \frac{x^{(k),1}-x^{(k-1)}}{y_1^{(k)}-y_1^{(k-1)}}(y - y_1^{(k-1)}) + x^{(k-1)} & \text{when } y \in [y_1^{(k-1)}, y_1^{(k)}) \\ \frac{x^{(k+1)}-x^{(k),1}}{y_1^{(k+1)}-y_1^{(k)}}(y - y_1^{(k)}) + x^{(k),1} & \text{when } y \in [y_1^{(k)}, y_1^{(k+1)}), \end{cases} \quad \text{and}$$

$$\overline{f}(y) = \begin{cases} \frac{x^{(i+1)}-x^{(i)}}{y_1^{(i+1)}-y_1^{(i)}}(y - y_1^{(i)}) + x^{(i)} & \text{when } y \in [y_1^{(i)}, y_1^{(i+1)}) \text{ and } i \neq k, i \neq k+1 \\ \frac{x^{(k),2}-x^{(k-1)}}{y_1^{(k)}-y_1^{(k-1)}}(y - y_1^{(k-1)}) + x^{(k-1)} & \text{when } y \in [y_1^{(k-1)}, y_1^{(k)}) \\ \frac{x^{(k+1)}-x^{(k),2}}{y_1^{(k+1)}-y_1^{(k)}}(y - y_1^{(k)}) + x^{(k),2} & \text{when } y \in [y_1^{(k)}, y_1^{(k+1)}), \end{cases}$$

and let $f : \mathbb{R} \to \mathbb{R}$ be given by $f(y) = \frac{x^(i+1)-x^{(i)}}{y_1^{(i+1)}-y_1^{(i)}}(y-y_1^{(i)})+x^{(i)}$ when $y \in [y_1^{(i)}, y_1^{(i+1)}]$. It is then straight forward to check that f, $\underline{f}$ and $\overline{f}$ are piecewise linear functions such that $\underline{f}(y) \leq f(y) \leq \overline{f}(y)$ for all $y \in \mathbb{R}$ and such that they interpolate the desired points. By [7, Theorem 2.2] every such function can be represented by a 2-layer ReLu-network. More precisely we have that $f = V^{(2)} \circ \sigma \circ V^{(1)}$, $\underline{f} = \underline{V}^{(2)} \circ \sigma \circ \underline{V}^{(1)}$, and $\overline{f} = \overline{V}^{(2)} \circ \sigma \circ \overline{V}^{(1)}$. Finally, the projection $\pi_1 : \mathbb{R}^m \to \mathbb{R}$ onto the first coordinate is a linear map. Hence we may set $\Phi(y) = V^{(2)} \circ \sigma \circ V^{(1)} \circ \pi_1(y)$, $\underline{\phi}(y) = \underline{V}^{(2)} \circ \sigma \circ \underline{V}^{(1)} \circ \pi_1(y)$, and $\overline{\phi}(y) = \overline{V}^{(2)} \circ \sigma \circ \overline{V}^{(1)} \circ \pi_1(y)$ to obtain the desired result. □

Corollary A.3. *Let $m, N, d \in \mathbb{N}$ with $d \geq 2$. Let $\mathcal{NN}^d_{ReLu}$ denote the class of neural networks, according to Definition* A.1*, where $K = d$, $N_0 = m$, $N_1, \ldots, N_{d-1} \in \mathbb{N}$ and $N_d = N$, and where $\sigma : \mathbb{R} \to \mathbb{R}$ is given by*

$$\sigma(y) = ReLu(y) = \begin{cases} y & \text{if } y > 0 \\ 0 & \text{otherwise.} \end{cases}$$

Then $\mathcal{NN}^d_{ReLu}$ satisfies Assumption 2.1.

Proof. To assert the statement in the corollary from Lemma A.2 it suffices to prove the following two statements:

(1) Given a d-layer neural network $\Phi : \mathbb{R}^m \to \mathbb{R}^N$, it is possible to add a ReLU-layer, without changing the output, to obtain a $d+1$-layer neural network.
(2) Given $x \in \mathbb{R}^N$, $y \in \mathbb{R}^m$ and d-layer neural networks $\Phi_1, \ldots, \Phi_N$ such that $\Phi_i(y) = x_i$ for $i = 1, \ldots, N$, we can construct a d-layer neural network $\Phi : \mathbb{R}^m \to \mathbb{R}^N$ such that $\Phi(y) = x$.

We start by proving (i): We observe that, with $\sigma = \text{ReLu}$, we have that $x = \sigma(x) - \sigma(-x)$. Hence, by setting $V^{(1)} = [I_N, -I_N]^T$, where I_N denotes the N dimensional identity matrix, and $V^{(2)} = [I_N, -I_N]$, we get that $V^{(2)} \circ \sigma \circ V^{(1)}(x) = \sigma(x) - \sigma(-x) = x$ for any $x \in \mathbb{R}^N$. Thus, let $\Phi : \mathbb{R}^m \to \mathbb{R}^N$ be an arbitrary d-layer neural network, then the composition $V^{(2)} \circ \sigma \circ V^{(1)} \circ \Phi(x) = \Phi(x)$ is a $d+1$-layer neural network with the same output as Φ. This asserts part (i).

We now prove part (ii): Let $x = (x_1, \ldots, x_N) \in \mathbb{R}^N$ be an arbitrary vector and assume that for a given $y \in \mathbb{R}^m$ there exists d-layer neural networks $\Phi_1, \ldots, \Phi_N : \mathbb{R}^N \to \mathbb{R}$ such that $\Phi_i(y) = x_i$ for each $i = 1, \ldots, N$. We can then "stack" the neural networks $\Phi_1, \ldots, \Phi_N$ to construct a neural network $\Phi : \mathbb{R}^N \to \mathbb{R}^m$ such that $\Phi(y) = x$ in the following way. Let $V^{(j)} = [V_1^{(j)}, \ldots, V_N^{(j)}]^T$ for $j = 1, \ldots, d$, where $V_i^{(j)}$ denotes the j'th affine map in the neural network Φ_i for $i = 1, \ldots, N$. We then define $\Phi : \mathbb{R}^m \to \mathbb{R}^N$ to be the neural network $\Phi = V^{(d)} \circ \sigma \circ \cdots \circ \sigma \circ V^{(1)}$, then it's clear that Φ is a d-layer neural network. Moreover, $\Phi(y) = [\Phi_1(y), \ldots, \Phi_N(y)]^T = (x_1, \ldots, x_N) = x$, this asserts part (ii). □

Competing Interests. The authors declare none.

REFERENCES

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

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

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

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

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

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

[7] R. Arora, A. Basu, P. Mianjy, and A. Mukherjee. Understanding deep neural networks with rectified linear units. *Electron. Colloquium Comput. Complex.*, TR17, 2016.

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

[9] M. Baader, M. Mirman, and M. Vechev. Universal approximation with certified networks. In *International Conference on Learning Representations*, 2020.

[10] 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.

[11] A. Bastounis, A. C. Hansen, and V. Vlacic. The mathematics of adversarial attacks in AI – Why deep learning is unstable despite the existence of stable neural networks. *arXiv:2109.06098*, 2021.

[12] 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.

[13] 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.

[14] 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.

[15] J. Ben-Artzi, M. Marletta, and F. Rösler. Computing the sound of the sea in a seashell. *Found. Comput. Math.*, 22(3):697–731, 2022.

[16] J. Ben-Artzi, M. Marletta, and F. Rösler. Computing scattering resonances. *J. Eur. Math. Soc.*, (to appear).

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

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

[19] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

[20] 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.

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

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

[23] 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.

[24] A. Bourrier, M. E. Davies, T. Peleg, P. Pérez, and R. Gribonval. Fundamental performance limits for ideal decoders in high-dimensional linear inverse problems. *IEEE Transactions on Information Theory*, 60(12):7928–7946, 2014.

[25] 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.

[26] L. Bungert, N. García Trillos, and R. Murray. The geometry of adversarial training in binary classification. *Information and Inference: A Journal of the IMA*, 12(2):921–968, 01 2023.

[27] 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.

[28] E. J. Candès, 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.

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

[30] G. S. Ceĭtin. Algorithmic operators in constructive metric spaces. *Trudy Mat. Inst. Steklov.*, 67:295–361, 1962.

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

[32] 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.

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

[34] 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.

[35] M. Colbrook. On the computation of geometric features of spectra of linear operators on hilbert spaces. *Foundations of Computational Mathematics*, (to appear).

[36] M. Colbrook and A. C. Hansen. The foundations of spectral computations via the solvability complexity index hierarchy. *J. Eur. Math. Soc.*, (to appear).

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

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

[39] 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. *Proc. Natl. Acad. Sci. USA*, 119(12):e2107151119, 2022.

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

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

[42] R. DeVore, G. Petrova, and P. Wojtaszczyk. Instance-optimality in probability with an l_1-minimization decoder. *Appl. Comput. Harmon. Anal.*, 27, 2009.

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

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

[45] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1625–1634, 2018.

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

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

[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] 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.

[51] L. E. Gazdag and A. C. Hansen. Generalised hardness of approximation and the SCI hierarchy – on determining the boundaries of training algorithms in AI, 2023.

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

[53] 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.

[54] 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.

[55] 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.

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

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

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

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

[60] 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.

[61] 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.

[62] P. Kidger and T. Lyons. Universal Approximation with Deep Narrow Networks. In J. Abernethy and S. Agarwal, editors, *Proceedings of Thirty Third Conference on Learning Theory*, volume 125 of *Proceedings of Machine Learning Research*, pages 2306–2327. PMLR, 09–12 Jul 2020.

[63] K. Ko. *Complexity Theory of Real Functions*. Birkhauser, 1991.

[64] G. Kreisel, D. Lacombe, and J. R. Shoenfield. Partial recursive functionals and effective operations. In *Constructivity in mathematics: Proceedings of the colloquium held at Amsterdam, 1957 (edited by A. Heyting)*, Stud. Logic Found. Math., pages 290–297. North-Holland, Amsterdam, 1959.

[65] 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.

[66] Y. Ma and Y. Fu. *Manifold Learning Theory and Applications*. Taylor & Francis, 2011.

[67] J. Macdonald, M. März, L. Oala, and W. Samek. Interval neural networks as instability detectors for image reconstructions. In *Bildverarbeitung für die Medizin 2021*, pages 324–329, Wiesbaden, 2021. Springer Fachmedien Wiesbaden.

[68] M. T. McCann, K. H. Jin, and M. Unser. Convolutional neural networks for inverse problems in imaging: A review. *IEEE Signal Process Mag.*, 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] M. B. Mirman, M. Baader, and M. Vechev. The fundamental limits of neural networks for interval certified robustness. *Transactions on Machine Learning Research*, 2022.

[72] 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.

[73] S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: a simple and accurate method to fool deep neural networks. *CoRR*, abs/1511.04599, 2015.

[74] A. Nemirovski. Lectures on Robust Convex Optimization. Available online at `https://www2.isye.gatech.edu/~nemirovs/`, 2009.

[75] A. Nemirovskii. Several np-hard problems arising in robust stability analysis. *Mathematics of Control, Signals and Systems*, 6(2):99–105, 1993.

[76] 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.

[77] L. Oala, C. Heiz, J. Macdonald, M. Marz, W. Samek, and G. Kutyniok. Interval neural networks: Uncertainty scores, 2020.

[78] 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.

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

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

[81] J. Sadeghi, M. de Angelis, and E. Patelli. Efficient training of interval neural networks for imprecise training data. *Neural Networks*, 118:338–351, 2019.

[82] 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.

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

[85] 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.

[86] S. Smale. Mathematical problems for the next century. In V. Arnold, M. Atiyah, P. Lax, and B. Mazur, editors, *Mathematics: Frontiers and Perspectives*. American Mathematical Society, 2000.

[87] E. Strickland. 2021's top stories about AI spoiler: A lot of them talked about what's wrong with machine learning today. *IEEE Spectrum*, Dec 2021.

[88] 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.

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

[90] 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.

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

[92] C. R. Vogel. *Computational Methods for Inverse Problems*. Society for Industrial and Applied Mathematics, 2002.

[93] F. Voigtlaender. The universal approximation theorem for complex-valued neural networks. *Applied and Computational Harmonic Analysis*, 64:33–61, 2023.

[94] S. Wang, N. Si, J. Blanchet, and Z. Zhou. On the foundation of distributionally robust reinforcement learning. *arXiv:2311.09018*, 2023.

[95] Z. Wang, A. Albarghouthi, G. Prakriya, and S. Jha. Interval universal approximation for neural networks. *Proc. ACM Program. Lang.*, 6(POPL), jan 2022.

[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] P. Wojtaszczyk. Stability and instance optimality for Gaussian measurements in compressed sensing. *Found. Comput. Math.*, 10, 2010.

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

[100] M. Zhang, O. Press, W. Merrill, A. Liu, and N. A. Smith. How language model hallucinations can snowball. *arXiv:2305.13534*, 2023.

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

DEPARTMENT OF MATHEMATICS, UNIVERSITY OF OSLO
Email address: vegarant@math.uio.no

DEPARTMENT OF APPLIED MATHEMATICS AND THEORETICAL PHYSICS, UNIVERSITY OF CAMBRIDGE
Email address: ach70@cam.ac.uk