CAN STABLE AND ACCURATE NEURAL NETWORKS BE COMPUTED? – ON THE BARRIERS OF DEEP LEARNING AND SMALE'S 18TH PROBLEM

VEGARD ANTUN*, MATTHEW J. COLBROOK†, AND ANDERS C. HANSEN†

ABSTRACT. Deep learning (DL) has had unprecedented success and is now entering scientific computing with full force. However, DL suffers from a universal phenomenon: instability, despite universal approximating properties that often guarantee the existence of stable neural networks (NNs). We show the following paradox. There are basic well-conditioned problems in scientific computing where one can prove the existence of NNs with great approximation qualities, however, there does not exist any algorithm, even randomised, that can train (or compute) such a NN. Indeed, for any positive integers $K > 2$ and L, there are cases where simultaneously: (a) no randomised algorithm can compute a NN correct to K digits with probability greater than $1/2$, (b) there exists a deterministic algorithm that computes a NN with $K - 1$ correct digits, but any such (even randomised) algorithm needs arbitrarily many training data, (c) there exists a deterministic algorithm that computes a NN with $K - 2$ correct digits using no more than L training samples. These results provide basic foundations for Smale's 18th problem and imply a potentially vast, and crucial, classification theory describing conditions under which (stable) NNs with a given accuracy can be computed by an algorithm. We begin this theory by initiating a unified theory for compressed sensing and DL, leading to sufficient conditions for the existence of algorithms that compute stable NNs in inverse problems. We introduce Fast Iterative REstarted NETworks (FIRENETs), which we prove and numerically verify are stable. Moreover, we prove that only $\mathcal{O}(|\log(\epsilon)|)$ layers are needed for an ϵ accurate solution to the inverse problem (exponential convergence), and that the inner dimensions in the layers do not exceed the dimension of the inverse problem. Thus, FIRENETs are computationally very efficient.

CONTENTS

1. INTRODUCTION

Deep learning (DL) has demonstrated unparalleled accomplishments in fields ranging from image classification and computer vision [56, 67, 76], voice recognition and automated diagnosis in medicine [43, 69, 82], to inverse problems and image reconstruction [12, 63, 71, 85] (this list of areas is by no means exhaustive, see [78], for example). However, at the same time, there is now overwhelming empirical evidence that DL leads to unstable methods, a phenomenon that seems universal and present in all of the applications listed

*DEPARTMENT OF MATHEMATICS, UNIVERSITY OF OSLO.
†DEPARTMENT OF APPLIED MATHEMATICS AND THEORETICAL PHYSICS, UNIVERSITY OF CAMBRIDGE.
E-mail addresses: vegarant@math.uio.no, m.colbrook@damtp.cam.ac.uk, a.hansen@damtp.cam.ac.uk.
Key words and phrases. Foundations of computational mathematics, computational neural networks, stability and accuracy, SCI Hierarchy, inverse problems, deep learning, Smale's 18th problem.

above [7, 9, 36, 52, 70, 90, 105] and in most of the new artificial intelligence (AI) technologies. These instabilities are often detected by what has become commonly known in the literature as "adversarial attacks". Moreover, the instabilities can be present even in random cases and not just worst-case scenarios [59] (see also "stealth attacks" introduced in [108] which concern perturbations to the AI system itself as opposed to inputs). There is a growing awareness of this problem in high-stake applications and society as a whole [13, 52, 64], and instability seems to be the Achilles' heel of modern AI and DL – see Figure 1 for an example (this is to be contrasted with Figure 2). However, classical approximation theorems show that a continuous function can be approximated arbitrarily well by a neural network (NN). Thus, stable problems described by stable functions can always be solved stably with a NN. This leads to the basic question:

> *Why does DL lead to universally unstable methods even when one can prove that stable and accurate neural networks exist?*

It is important to note that proof of the existence of suitable NNs does not immediately imply that they can be constructed by an algorithm. In particular, we are faced with the following fundamental problem:

(**Question I: Can neural networks that provably exist be trained/computed?**) *Let $\Omega_{\mathcal{T}}$ be a collection of training data $\iota = \{x_j\}$, and suppose that for each $\iota \in \Omega_{\mathcal{T}}$ one can prove that there exists a NN $\Phi_\iota \in \mathcal{N}$ (with certain properties). In particular, there is a mapping $\mathcal{K}\colon \Omega_{\mathcal{T}} \to \mathcal{N}$. Does there exist an algorithm, taking $\iota \in \Omega_{\mathcal{T}}$ as input, that computes an approximation to $\mathcal{K}(\iota) = \Phi_\iota$ for all $\iota \in \Omega_{\mathcal{T}}$, and to what accuracy?*

Question I is the key problem that we address (see §1.2 for a detailed summary), and our barriers, as well as our positive results, help shed light on why the desired NNs that exist can or cannot be computed.

Remark 1.1 (**Stability/accuracy trade-off and existence of algorithms).** It is, of course, not difficult to compute stable NNs: the zero network is stable, however, not particularly useful. Hence, the big problem is to compute accurate and stable NNs. Scientific computing is based on two pillars: stability and accuracy, however, there is often a trade-off between the two [93]. There may be barriers preventing the existence of accurate and stable algorithms, and some accuracy may have to be sacrificed in order to secure stability. ⊠

1.1. **Hilbert's program, Smale's 18th problem and the foundations of AI.** The strong optimism regarding the abilities of AI and DL is summarised in The New Yorker's (April 2017 issue) quote of G. Hinton: "*They should stop training radiologists now.*" One can argue that this optimism is comparable to the optimism about mathematics in the early 20th century, led by D. Hilbert. Hilbert believed that mathematics could prove or disprove any statement and, moreover, that there were no restrictions on which problems could be solved by algorithms. The latter being emphasised in Hilbert's 10th problem [84]: "*Find an algorithm* to determine whether a given polynomial Diophantine equation with integer coefficients has an integer solution." Hilbert did not consider the case that such an algorithm may not exist (even though, indeed, no such algorithm exists), suggesting a substantial optimism on what mathematics and algorithms can solve. However, Hilbert was also well aware that such foundations were not established in mathematics and initiated a vast program leading to the beginning of modern logic and subsequently modern computer science.

Gödel [57] and Turing [106] turned Hilbert's optimism upside down by their foundational contributions establishing impossibility results on what mathematics and digital computers can achieve [94]. We argue that a program on the foundations of AI, similar to Hilbert's program, is needed, where impossibility results are provided in order to establish the boundaries of DL and AI. Such a program is already suggested in Smale's 18th problem, from the list of mathematical problems for the 21st century [103], which echoes Turing's paper from 1950 [107] on the question: what is AI? Turing asks if a computer can think, and suggests the imitation game as a test for his question about AI. Smale takes the question even further and asks in his 18th problem:

> *"What are the limits of intelligence, both artificial and human?"*
>
> — Smale's 18th problem (from the list of mathematical problems for the 21st century [103])

The question is followed by a discussion on the problem that ends as follows: *"Learning is a part of human intelligent activity. The corresponding mathematics is suggested by the theory of repeated games, neural*

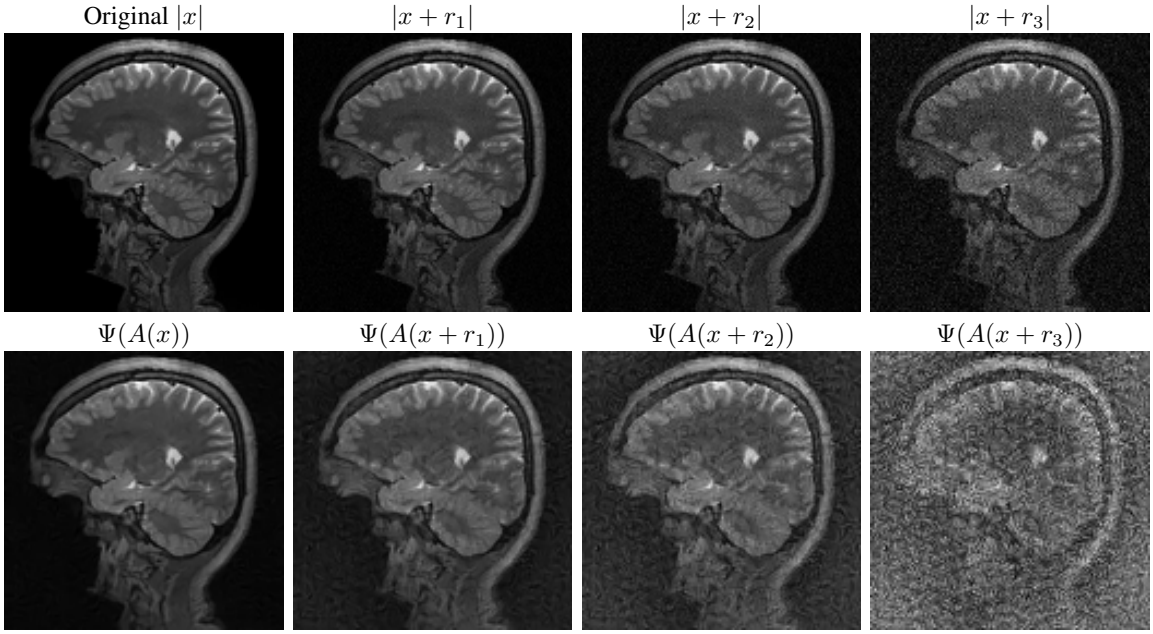

FIGURE 1. (**Unstable neural network in image reconstruction**). The neural network AUTOMAP (*Nature* (2018) [113]) represents the tip of the iceberg of DL in inverse problems. The paper promises that one can "... observe superior immunity to noise...". Moreover, the follow-up announcement (*Nature Methods* "*AI transforms image reconstruction*," [104]) proclaims: "A deep-learning-based approach improves speed, accuracy and robustness of biomedical image reconstruction". However, the figure shows $|x+r_j|$, where x is the original image and the r_js are perturbations meant to simulate worst-case effect, as well as the that AUTOMAP reconstruction $\Psi(A(x+r_j))$ from the subsampled Fourier MRI data $A(x+r_j)$ (here $A \in \mathbb{C}^{m\times N}$ is a subsampled Fourier transform, see §4 for details) concluding that this network is completely unstable. Note that the condition number $\mathrm{cond}(AA^*) = 1$, so the instabilities are not caused by poor condition. As demonstrated in [9], this is a universal phenomenon in DL for inverse problems. Experimental details are given in §4.

nets and genetic algorithms." Given the recent unprecedented developments in DL and NNs [78], and the impact these developments may have on AI, it is timely to consider Smale's 18th problem. We interpret the words "artificial intelligence" as the current state-of-the-art AI for which DL is essential. Our results provide foundations for Smale's 18th problem since they imply a potentially vast classification theory for determining the limits of what DL can achieve. Importantly, this classification theory cannot be determined by the extensive collection of non-constructive existence theorems (à la universal approximation theorems) for NNs.

1.2. **Summary of the main results.** Our main results demonstrate that there are fundamental barriers preventing NNs, despite their existence, from being computed by algorithms. This helps shed light on the intricate question on why current algorithms in DL produce unstable networks, despite the fact that stable NNs often exist in the particular application. Indeed, our results demonstrate that there is a rich and unknown classification theory on which types of stable NNs can be computed by algorithms. Our proof techniques stem from the Solvability Complexity Index (SCI) hierarchy that has recently been used to settle longstanding questions in scientific computing [15, 18–20, 40, 41, 65], and that generalises the fundamental problems of S. Smale on existence of algorithms [24, 25, 100–102] and work by C. McMullen [86, 87] and P. Doyle & C. McMullen [48].

(I) **(Neural networks may exist, but cannot be computed, even for well-conditioned problems).** The answer to Question I above is, in general, 'no', even for well-conditioned problems. Mappings that take training data

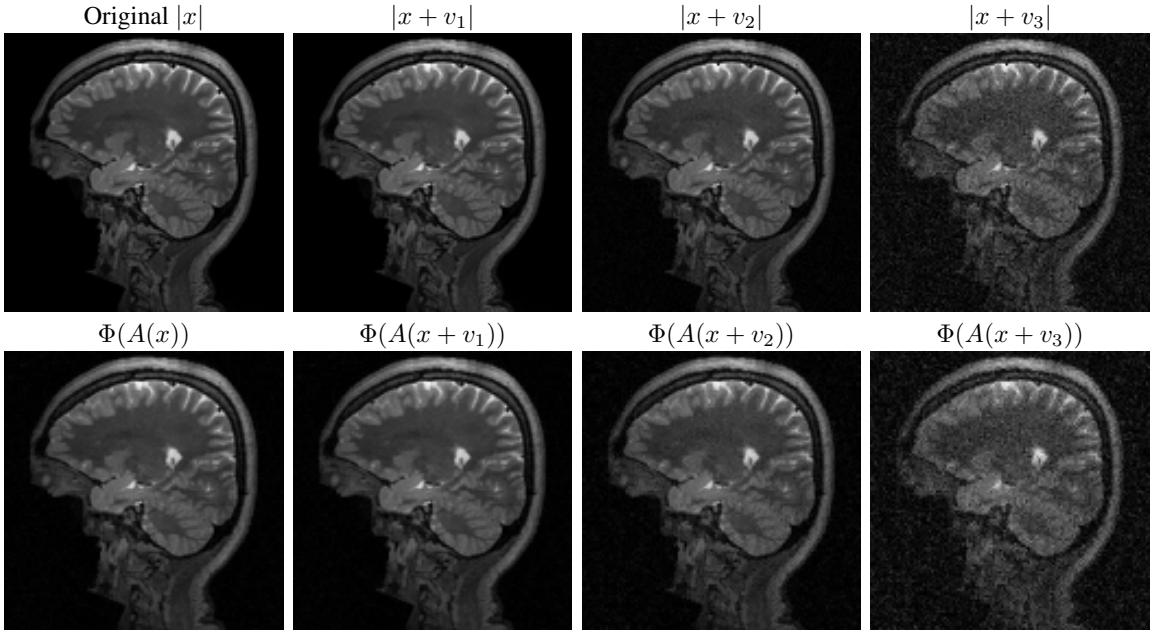

FIGURE 2. (**The FIRENET is stable to worst-case perturbations**). Using the same method as in Figure 1, we compute perturbations v_j in the image domain, to simulate worst-case effect for the FIRENET $\Phi\colon \mathbb{C}^m \to \mathbb{C}^N$. Here x and $A \in \mathbb{C}^{m\times N}$ are the same image and sampling matrix as in Figure 1. Moreover, for each $j = 1, 2, 3$ we have ensured that $\|v_j\|_{l^2} \geq \|r_j\|_{l^2}$, where the r_j's are the perturbations from Figure 1 (we have denoted the perturbations for FIRENET by v_j to emphasise that these adversarial perturbations are sought for FIRENET and have nothing to do with the perturbations in Figure 1). The top row shows the perturbed images $|x+v_j|$, $j = 0, 1, 2, 3$ (assuming $v_0 = 0$), and the bottom row shows the network's reconstruction from the perturbed measurements $A(x+v_j)$. Experimental details are in §4.

to NNs may exist, however, no algorithm that computes approximations of the NNs from the training data exists. This statement is made precise in Theorems 2.1 and 2.2, and is valid for any model of computation.

(II) **(Randomised algorithms do not help in solving the issue).** The answer to Question I is still 'no' for any randomised algorithm. That is, as Theorem 2.2 reveals, replacing a deterministic algorithm with a randomised algorithm (common randomised algorithms in DL include stochastic gradient descent) will not yield the desired error with probability better than coin-flipping.

(III) **(Algorithms may compute neural networks to $K-1$ digits of accuracy, but not K).** As Theorem 2.2 reveals, the answer to Question I above depends on the desired accuracy. For any integer $K > 1$ there exist classes of problems for which there is a mapping taking the training data to the NN, however, there will only exist an algorithm that can compute an approximation to the NN to $K-1$ correct digits, and no algorithm – even randomised – can compute an approximation to K correct digits.

(IV) **(Algorithms may exist, but any algorithm will require arbitrary large training data).** Theorem 2.2 also shows that there are classes of problems where one can compute an approximation to the desired NN with $K-1$ correct digits, however, for any $M \in \mathbb{N}$ and any algorithm, there will be a problem so that the algorithm requires more than M training data to provide an approximation with at least $K-1$ correct digits. However, for the same class of problems, there exists an algorithm providing $K-2$ correct digits using only one training datum. Condition numbers (see §8.1) of the problems in Theorem 2.2 are all bounded by 1.

(V) **(Algorithms computing stable and accurate neural networks exist only in specific cases).** (I)–(IV) demonstrate that it is only in specific cases that there exists an algorithm that can compute the desired NN, despite the fact that one can prove that the desired NN exists. This implies that there is a theory classifying which NNs can be computed by algorithms and which conditions are sufficient and necessary for the existence of algorithms. We initiate this classification theory below in Theorems 5.5 and 5.10.

(VI) **(Unrolling optimisation algorithms as neural networks will, in general, not converge).** The concept of unrolling optimisation algorithms as a sequence of NNs is highly popular (see §7). Due to the specific choices of mappings in Theorems 2.1 and 2.2 (see §2.1), our impossibility results imply that such sequences of NNs will in general never converge. Indeed, potentially surprisingly, the objective function values of iterates will converge, yet the unrolled NN's output will in general never converge to the solution set.

(VII) **(Under specific conditions, unrolling yields stable and accurate neural networks).** Theorems 5.5 and 5.10 show that under specific conditions that typically are present in, for example, Magnetic Resonance Imaging (MRI), the technique of unrolling optimisation algorithms will yield stable and accurate NNs for certain inverse problems, hence providing an algorithm for constructing these NNs. Such NNs outperform state-of-the-art unstable (displayed in Figure 1) trained NNs based on DL, such as AUTOMAP, and they can withstand adversarial attacks, see Figure 2. When these specific conditions are present, the unrolling NNs can even be used to stabilise any unstable NN, see Figure 3 for an example.

(VIII) **(Specific conditions yield unrolled neural networks with exponential convergence in the number of layers and fast transforms to implement the linear maps).** Given specific conditions, the unrolling procedure can be manipulated through a careful restart scheme to yield exponential convergence: one can obtain an error of ϵ by using a NN with order $|\log(\epsilon)|$ layers. Moreover, the linear maps in the layers of the NN can be implemented using fast transforms in many applications (e.g. those in §5.3). Hence, one obtains fast algorithms to produce stable and accurate NNs that can be executed very efficiently. We demonstrate this in §6, where only on the order of 10 layers are needed to obtain accurate recovery of undersampled noisy images.

(IX) **(Stability/accuracy trade-off and avoiding overperformance).** There is a trade-off between stability and accuracy in DL, with limits on how well a stable NN can perform in inverse problems (e.g. if a NN is trained to accurately recover vectors whose difference lies close to the kernel of the forward map, the recovery is necessarily unstable - see §3). Figure 5 demonstrates this with a U-net trained on images consisting of ellipses and which is quite stable. However, when a detail not in the training set is added, it washes it out almost entirely. This is a problem in real-world clinical practice - Facebook and NYU's 2019 FastMRI challenge reported that networks that performed well in terms of standard image quality metrics were prone to false negatives, failing to reconstruct small, but physically-relevant image abnormalities [74]. The 2020 version of the challenge subsequently focused on pathologies and also included a generalisation track, noting, "Our challenge confirmed areas in need of research, particularly those along the lines of evaluation metrics, error characterization, and AI-generated hallucinations" [91]. In contrast, as demonstrated in §4, our NNs offer a blend of both stability and accuracy. However, they are by no means the end of the story. Tracing out the optimal stability vs. accuracy trade-off curve is crucial for applications and will no doubt require a myriad of different techniques to be developed to tackle different problems.

(X) **(Numerical stability)** It is important to understand the difference between the stability (or conditioning) of a map and the stability of its numerical implementation (see, for example, the recent work in [23] which considers common functions used in classification methods). As well as the former, our results cover the latter by performing an error analysis of the forward pass of the NNs, assuming each layer is computed with an error (see Remark 5.3). We show that only low precision is needed and that worst-case errors can only ever accumulate slowly as the number of layers increase.

We call our neural networks **F**ast **I**terative **RE**started **NET**works, or **FIRENETs**. The code and data, used to produce all figures in this manuscript are available from:

`www.github.com/Comp-Foundations-and-Barriers-of-AI/firenet`.

1.3. **Notation.** We briefly collect some basic notation, further notation will be introduced throughout where appropriate. We use $\mathcal{N}_{m,N}$ to denote the class of neural networks (NNs) from $\mathbb{C}^m$ to $\mathbb{C}^N$ (see §5.1 for the precise definition). Given a metric space $(\mathcal{M}, d)$, $x \in \mathcal{M}$ and $X \subset \mathcal{M}$, $d(x, X) = \operatorname{dist}(x, X) = \inf_{y \in X} d(x, y)$. For a matrix $A \in \mathbb{C}^{m \times N}$, the norm $\|A\|$ refers to the operator norm of A when $\mathbb{C}^m$ and $\mathbb{C}^N$ are equipped with the standard l^2-norm. For $x \in \mathbb{C}^N$ and $p \in [1, \infty]$, $\|x\|_{l^p}$ refers to the l^p-norm of x. For

a set of indices S and vector x, x_S is the vector defined by $(x_S)_j = x_j$ if $j \in S$ and $(x_S)_j = 0$ if $j \notin S$. Complex rationals $\mathbb{Q} + i\mathbb{Q}$ are denoted by $\mathbb{Q}[i]$. We use $\square$ to denote the end of a proof and $\boxtimes$ to denote the end of a remark.

2. Main Results I: Fundamental barriers and existence of algorithms

The canonical inverse problem studied in this paper is to solve an underdetermined system of equations:

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

Here $A \in \mathbb{C}^{m\times N}$ represents a model of typically undersampled sampling ($m < N$), such as a subsampled discrete Fourier transform as in Magnetic Resonance Imaging (MRI). Problem (2.1) forms the basis for much of inverse problems and image analysis. The possibility of $y \neq Ax$ models noise or perturbations.

2.1. Existence of NNs is not enough, algorithms may not compute them sufficiently accurately.

To demonstrate that the results we introduce are present in applications, we consider basic mappings used in modern mathematics of information, inverse problems and optimisation. Given a matrix $A \in \mathbb{C}^{m\times N}$ and a vector $y \in \mathbb{C}^m$, we consider the following three minimisation problems:

$$(P_1) \quad \operatorname{argmin}_{x\in\mathbb{C}^N} F_1^A(x) \coloneqq \|x\|_{l^1_w}, \text{ such that } \|Ax - y\|_{l^2} \leq \epsilon, \tag{2.2}$$

$$(P_2) \quad \operatorname{argmin}_{x\in\mathbb{C}^N} F_2^A(x, y, \lambda) \coloneqq \lambda\|x\|_{l^1_w} + \|Ax - y\|^2_{l^2}, \tag{2.3}$$

$$(P_3) \quad \operatorname{argmin}_{x\in\mathbb{C}^N} F_3^A(x, y, \lambda) \coloneqq \lambda\|x\|_{l^1_w} + \|Ax - y\|_{l^2}, \tag{2.4}$$

known respectively as (quadratically constrained) basis pursuit [1, 16, 35], unconstrained LASSO [33, 66] and unconstrained square-root LASSO [17, 109]. In applications, one is often interested in the solution of (2.1) rather than the problems (P_j). However, sparse regularisation is often used as a benchmark method for (2.1) and we prove impossibility results for approximations of the solution maps of these sparse regularisation problems. The solution maps of (P_j) are considerably simpler than the problem (2.1) (which is, in general, ill-posed) and an impossibility result for these simpler problems is a striking computational barrier given that one can prove the existence of accurate neural networks (Theorem 2.1).

The parameters λ and ϵ are positive rational numbers, and the weighted l^1_w norm is given by $\|x\|_{l^1_w} \coloneqq \sum_{l=1}^N w_l|x_l|$, where each weight w_j is a positive rational. Throughout, we use the following notation:

$$\Xi(A, y) \text{ is the set of minimisers for } (P_j) \text{ given input } A \in \mathbb{C}^{m\times N}, y \in \mathbb{C}^m, \tag{2.5}$$

where, for notational convenience, we have suppressed the dependence on ϵ or λ (which are usually fixed parameters) and the index j. In certain cases, we will write Ξ_j to specify minimisers of problem (P_j). Let

$$A \in \mathbb{C}^{m\times N}, \quad \mathcal{S} = \{y_k\}_{k=1}^R \subset \mathbb{C}^m, \quad R < \infty.$$

We consider the following key question:

> *Given a collection Ω of such pairs $(A, \mathcal{S})$, does there exists a neural network approximating the mapping Ξ, and if so, can such an approximation be trained by an algorithm?*

To make this question precise, we first note that A and the elements in $\mathcal{S}$ will typically never be exact, but can be approximated to arbitrary precision. For example, this would be the case if A was a subsampled discrete cosine transform. Thus, we can access approximations $\{y_{k,n}\}_{k=1}^R \subset \mathbb{Q}[i]^m$ and $A_n \in \mathbb{Q}[i]^{m\times N}$ such that

$$\|y_{k,n} - y_k\| \leq 2^{-n}, \quad \|A_n - A\| \leq 2^{-n}, \quad \forall n \in \mathbb{N}. \tag{2.6}$$

(The bounds 2^{-n} are chosen for convenience and can be replaced by any other sequence of positive rationals converging to zero.) We also assume access to $\{x_{k,n}\}_{k=1}^R \subset \mathbb{Q}[i]^N$ such that

$$\inf_{x^*\in\Xi(A_n,y_{k,n})} \|x_{k,n} - x^*\| \leq 2^{-n}, \quad \forall n \in \mathbb{N}. \tag{2.7}$$

Hence, the training set associated with $(A, \mathcal{S}) \in \Omega$ for training a suitable NN must be

$$\iota_{A,\mathcal{S}} \coloneqq \{(y_{k,n}, A_n, x_{k,n}) \mid k = 1, \dots, R, \text{ and } n \in \mathbb{N}\}. \tag{2.8}$$

Thus, given a collection of $(A, \mathcal{S})$, we denote the class of all such admissible training data by

$$\Omega_{\mathcal{T}} \coloneqq \{\iota_{A,\mathcal{S}} \text{ as in (2.8)} \mid (A, \mathcal{S}) \in \Omega, \text{ (2.6) and (2.7) hold}\}.$$

Precise statements addressing the above question are summarised in the following theorems (the first follows directly from universal approximation theorems).

Theorem 2.1 (**Neural networks exist for Ξ**). *Consider the problem (P_j) ($j = 1, 2, 3$) for fixed dimensions $m < N$ and parameters λ or ϵ. Then, for any family Ω of such $(A, \mathcal{S})$ described above, there exists a mapping*

$$\mathcal{K}\colon \Omega_{\mathcal{T}} \to \mathcal{N}_{m,N}, \quad \mathcal{K}(\iota_{A,\mathcal{S}}) = \varphi_{A,\mathcal{S}}, \quad \textit{such that} \quad \varphi_{A,\mathcal{S}}(y) \in \Xi(A, y), \quad \forall y \in \mathcal{S}.$$

In words, $\mathcal{K}$ maps the training data $\Omega_{\mathcal{T}}$ to NNs that solve the optimisation problem (P_j) for each $(A, \mathcal{S}) \in \Omega$.

Despite the existence of NNs guaranteed by Theorem 2.1, the problem of computing such a NN from training data is a most delicate issue, as described in the following theorem (proven in §8).

Theorem 2.2 (**Despite existence, neural networks may only be computed to a certain accuracy**). *For $j = 1, 2$ or 3, consider the optimisation problem (P_j) for fixed parameters $\lambda \in (0, 1]$ or $\epsilon \in (0, 1/2]$ and $w_l = 1$, where $N \geq 2$ and $m < N$. Let $K > 2$ be a positive integer and let $L \in \mathbb{N}$. Then there exists a class Ω of elements $(A, \mathcal{S})$ as in (2.5), with the following properties. The class Ω is well-conditioned with condition numbers of the matrices AA^* and the solution maps Ξ, as well as the feasibility primal local condition number (see §8.1), all bounded by 1 independent of all parameters. However, the following hold:*

(i) There does not exist any algorithm that, given a training set $\iota_{A,\mathcal{S}} \in \Omega_{\mathcal{T}}$, produces a NN $\phi_{A,\mathcal{S}}$ with

$$\min_{y \in \mathcal{S}} \inf_{x^* \in \Xi(A,y)} \|\phi_{A,\mathcal{S}}(y) - x^*\|_{l^2} \leq 10^{-K}, \quad \forall (A, \mathcal{S}) \in \Omega. \tag{2.9}$$

Furthermore, for any $p > 1/2$, no probabilistic algorithm (BSS, Turing or any model of computation) can produce a NN $\phi_{A,\mathcal{S}}$ such that (2.9) holds with probability at least p.

(ii) There does exist a deterministic Turing machine that, given a training set $\iota_{A,\mathcal{S}} \in \Omega_{\mathcal{T}}$, produces a NN $\phi_{A,\mathcal{S}}$ with

$$\max_{y \in \mathcal{S}} \inf_{x^* \in \Xi(A,y)} \|\phi_{A,\mathcal{S}}(y) - x^*\|_{l^2} \leq 10^{-(K-1)}, \quad \forall (A, \mathcal{S}) \in \Omega. \tag{2.10}$$

However, for any probabilistic Turing machine $(\Gamma, \mathbb{P})$, $M \in \mathbb{N}$ and $p \in \left[0, \frac{N-m}{N+1-m}\right)$ that produces a NN $\phi_{A,\mathcal{S}}$, there exists a training set $\iota_{A,\mathcal{S}} \in \Omega_{\mathcal{T}}$ such that for all $y \in \mathcal{S}$,

$$\mathbb{P}\Big(\inf_{x^* \in \Xi(A,y)} \|\phi_{A,\mathcal{S}}(y) - x^*\|_{l^2} > 10^{1-K} \textit{ or the training data size needed to construct } \phi_{A,\mathcal{S}} > M\Big) > p. \tag{2.11}$$

(iii) There does exist a deterministic Turing machine that, given a training set $\iota_{A,\mathcal{S}} \in \Omega_{\mathcal{T}}$ and using only L training data from each $\iota_{A,\mathcal{S}}$, produces a NN $\phi_{A,\mathcal{S}}(y)$ such that

$$\max_{y \in \mathcal{S}} \inf_{x^* \in \Xi(A,y)} \|\phi_{A,\mathcal{S}}(y) - x^*\|_{l^2} \leq 10^{-(K-2)}, \quad \forall (A, \mathcal{S}) \in \Omega. \tag{2.12}$$

Remark 2.3 (Meaning of the notation $(\Gamma, \mathbb{P})$)**.** The notation $(\Gamma, \mathbb{P})$ in (ii) is used to denote a (possibly) randomised algorithm Γ and its law $\mathbb{P}$. This includes scenarios such as stochastic gradient descent, random selection of training data, random computation with training data etc. The precise setup is detailed in §8.1. ⊠

Remark 2.4 (Generalisations of Theorem 2.2)**.** For simplicity, we have stated Theorem 2.2 for errors measured in the l^2-norm and the case of unweighted l^1 regularisation (all the $w_l = 1$) in the problems (P_j). However, the proof can be adapted, and similar results hold for any norm replacing the l^2-norm, and any non-singular weighted l^1 regularisation. Moreover, result (i) in Theorem 2.2 holds regardless of the model of computation, even if we allowed real number arithmetic (see Definition 8.3). For further details on the precise

$\text{dist}(\Psi_{A_n}(y_n), \Xi_3(A,y))$	$\text{dist}(\Phi_{A_n}(y_n), \Xi_3(A,y))$	$\|A_n - A\| \leq 2^{-n}$ $\|y_n - y\|_{l^2} \leq 2^{-n}$	10^{-K}	Ω_K
0.2999690	0.2597827	$n = 10$	10^{-1}	$K = 1$
0.3000000	0.2598050	$n = 20$	10^{-1}	$K = 1$
0.3000000	0.2598052	$n = 30$	10^{-1}	$K = 1$
0.0030000	0.0025980	$n = 10$	10^{-3}	$K = 3$
0.0030000	0.0025980	$n = 20$	10^{-3}	$K = 3$
0.0030000	0.0025980	$n = 30$	10^{-3}	$K = 3$
0.0000030	0.0000015	$n = 10$	10^{-6}	$K = 6$
0.0000030	0.0000015	$n = 20$	10^{-6}	$K = 6$
0.0000030	0.0000015	$n = 30$	10^{-6}	$K = 6$

TABLE 1. (**Impossibility of computing approximations of the existing neural network to arbitrary accuracy**). We demonstrate the impossibility statement (i) from Theorem 2.2 on FIRENETs Φ_{A_n}, and trained LISTA networks Ψ_{A_n}. The table shows the shortest l^2 distance between the output from the networks, and the true solution of the problem (P_3), with $w_l = 1$ and $\lambda = 1$, for different values of n and K. Note that none of the trained networks can compute the existing correct NN (that exists by Theorem 2.1 and coincides with Ξ_3) to 10^{-K} digits accuracy, while all of them are able to compute approximations that are accurate to 10^{-K+1} digits (for the input class Ω_K). This agrees exactly with Theorem 2.2.

setup, including the definition of condition numbers, which are standard in the literature, see §8.1. Finally, the theorem remains true if we restrict ourselves to real-valued matrices and vectors. ⊠

2.1.1. **The neural network exists, but no algorithm can compute it – Numerical example**. To highlight the impossibility (Theorem 2.2) of computing neural networks – despite their existence by Theorem 2.1 – we consider the following numerical example. Consider the problem (P_3), with $w_l = 1$ and $\lambda = 1$. Theorem 2.2 is stated for a specific input class Ω, and in this example we will consider three different such classes Ω_K, depending on the accuracy parameter K. In Theorem 2.2, we required that $K > 2$, to ensure that $K - 2 > 0$ but this is not necessary to show the impossibility statement (i), so we consider $K = 1, 3, 6$.

To show that it is impossible to compute neural networks which can solve (P_3) to arbitrary accuracy we consider (untrained) FIRENETs Φ_{A_n} and (trained) learned ISTA (LISTA) networks Ψ_{A_n} based on the architecture choice from [39]. That is, networks of the form

$$\Psi_{A_n}(y) = x_T, \quad \text{where} \quad x_{i+1} = S_{\theta_i}\left(x_i + W_i^\top (x_i - A_n^\top y)\right), \quad \text{for } i = 0, \ldots, T-1, \text{ and } x_0 = 0, \tag{2.13}$$

for real-valued inputs y and matrices A_n. Here S_θ is the soft thresholding operator with threshold $\theta \geq 0$ and the weights $(\theta_i, W_i)_{i=0}^{T-1} \subset \mathbb{R}_{\geq 0} \times \mathbb{R}^{m \times N}$ are learned from the data using a squared l^2 norm loss function $(T = 10)$. The LISTA networks are trained to high accuracy on training data on the form (2.8) with $R = 8000$ training samples and n given as in Table 1. In all cases $N = 20$, $m = N - 1$ and the $x_{k,n}$'s minimising (P_3) with input data $(y_{k,n}, A_n)$, are all 6-sparse. We highlight that the choice of N, m and sparsity in this example is chosen to allow for fast training. Other choices are certainly possible.

In Table 1, we show the distance $\text{dist}(\Psi_{A_n}(y_n), \Xi_3(A,y))$ and $\text{dist}(\Phi_{A_n}(y_n), \Xi_3(A,y))$ for both the trained LISTA networks Ψ and the FIRENETs Φ. Both network types are given input data (y_n, A_n), approximating the true data (y, A). As is clear from the table, none of the networks are able to compute an approximation to the true minimiser in $\Xi_3(A,y)$ to K digits accuracy. However, both networks are able to compute an approximation with $K - 1$ digits accuracy. These observations agree with Theorem 2.2. Further details on how the input data and neural networks in this experiment are constructed can be found in §8.4.

3. Main Results II: Algorithms compute stable and accurate NNs in specific cases

Theorem 2.2 shows that the optimisation problems (P_1), (P_2), and (P_3) (defined in (2.2), (2.3), and (2.4)) cannot, in general, be solved by an algorithm. Hence any attempt at solving the *general* inverse problem (2.1) by such methods, without additional assumptions, is doomed to fail. This is not just the case for reconstruction using sparse regularisation. In fact, any stable and accurate reconstruction procedure must be "kernel aware" (see, for example, [59]) - a reconstruction method lacks kernel awareness (and thus has a large local Lipschitz constant) if it approximately recovers two vectors x and x' whose difference lies close to the null space of A.

3.1. The subtlety and difficulty of removing instabilities and the need for additional assumptions.

As discussed in §1 (see also §7 on previous work), the instability problem is the current Achilles' heel of modern AI and DL. For example, adversarial training has become a standard attempt to remedy instabilities (see, for example, [95]). However, such a strategy may yield poor performance. Indeed, consider the following optimisation problem which seeks to generate a reconstruction in the form of a NN, given samples $\Theta = \{(y_s, x_s) : s = 1, ..., R, Ax_s = y_s\}$ and $\epsilon, \lambda > 0$:

$$\min_{\phi \in \mathcal{N}_{m,N}} \frac{1}{R} \sum_{s=1}^{R} \max_{\|z\|_{l^2} \leq \epsilon} \left\{ \|x_s - \phi(y_s)\|_{l^2}^2 + \lambda \|x_s - \phi(y_s + z)\|_{l^2}^2 \right\}. \tag{3.1}$$

In other words, for each training point $(y, x) \in \Theta$ we find the worst-case perturbation z in the ϵ-ball around y (measured by the l^2-norm). This is a simplified model of what one might do using Generative Adversarial Networks (GANs) to approximate adversarial perturbations [10, 58]. For simplicity, assume that A has full row rank m and that we have access to exact measurements $y_s = Ax_s$. Suppose that our sample is such that $\min_{i \neq j} \|y_i - y_j\|_{l^2} > 2\epsilon$. Any ϕ that minimises (3.1) must be such that $\phi(y_s + z) = x_s$ for all z with $\|z\|_{l^2} \leq \epsilon$. Such networks can easily be constructed using, say, ReLU activation functions. Now suppose that x_2 is altered so that $x_1 - x_2$ lies in the kernel of A. Then for any minimiser ϕ, we must have

$$\phi(y_1 + z) = \phi(y_2 + z) = \frac{x_1 + x_2}{2}, \quad \forall z \text{ with } \|z\|_{l^2} \leq \epsilon,$$

and hence we can never be more than $\|x_1 + x_2\|_{l^2}/2$ accurate over the whole test sample. Given these types of examples and Theorem 2.2, we arrive at the following question:

> *Are there sufficient conditions on the matrix A that imply the existence of an algorithm that can compute a neural network that is both accurate and stable for the problem* (2.1)*?*

3.2. A sufficient condition yielding an algorithm computing accurate and stable neural networks.

Precise theorems can be found in §5, and for ease of exposition, we present simplified versions here. To state these, we need the concept of sparsity in levels from compressed sensing. This local sparsity structure was introduced in [5], and was demonstrated empirically to play a key role in the quality of the image recovery via the "flip test" in [5, 14, 97]. The sparsity in levels model is crucial in demonstrating that compressed sensing and sparse regularisation is near-optimal for image recovery [3]. The key issue is that natural images are not sparse in X-lets (wavelets, curvelets, shearlets etc.): they are sparse in levels. Thus, sparsity in levels is needed since the classical theoretical model in compressed sensing, sparsity in one level, does not account for the recovery often found in practice for problems such as the Fourier-wavelet problem. (The main problem for sparsity in one level in this example is that the Fourier-wavelet matrix is coherent [5].) For example, the seminal work of Lustig, Donoho & Pauly on compressed sensing for MRI [81] observed both poor recovery from uniform random sampling and the improvement offered by variable density sampling.

Definition 3.1 (Sparsity in levels)**.** *Let $\mathbf{M} = (M_1, ..., M_r) \in \mathbb{N}^r$, $1 \leq M_1 < ... < M_r = N$, and $\mathbf{s} = (s_1, ..., s_r) \in \mathbb{N}_0^r$, where $s_k \leq M_k - M_{k-1}$ for $k = 1, ..., r$ ($M_0 = 0$). $x \in \mathbb{C}^N$ is $(\mathbf{s}, \mathbf{M})$-sparse in levels if*

$$|\mathrm{supp}(x) \cap \{M_{k-1} + 1, ..., M_k\}| \leq s_k, \quad k = 1, ..., r.$$

The total sparsity is $s = s_1 + ... + s_r$. *We denote the set of* $(\mathbf{s}, \mathbf{M})$*-sparse vectors by* $\Sigma_{\mathbf{s},\mathbf{M}}$. *We also define the following measure of distance of a vector* x *to* $\Sigma_{\mathbf{s},\mathbf{M}}$ *by*

$$\sigma_{\mathbf{s},\mathbf{M}}(x)_{l^1_w} = \inf\{\|x - z\|_{l^1_w} : z \in \Sigma_{\mathbf{s},\mathbf{M}}\}.$$

Since its introduction, this model has been used to explain the effectiveness of compressed sensing in real life applications [14, 72, 110]. For simplicity, we will assume throughout that each $s_k > 0$ and that

$$w_i = w_{(j)}, \quad \text{if } M_{j-1} + 1 \leq i \leq M_j. \tag{3.2}$$

In other words, the weights in the l^1_w norm are constant in each level. For an image c which is compressible in the wavelet basis, $\sigma_{\mathbf{s},\mathbf{M}}(x)_{l^1_w}$ is expected to be small if x is the vector of wavelet coefficients and the levels correspond to wavelet levels, see [46] and [83, Ch. 9]. In general, the weights are a prior on anticipated support of the vector [54] and we discuss some specific choices in §5.3 (these can be chosen optimally).

The key "kernel aware" property that guarantees algorithms that compute stable and accurate NNs (with uniform recovery guarantees) for the inverse problem (2.1), is Definition 3.2, first used in the context of compressed sensing in [14] for the unweighted l^1 norm, and extended to l^1_w in [1]. In §5.3, we give examples common in applications where this condition holds.

Definition 3.2 (weighted rNSP in levels)**.** *Let* $(\mathbf{s}, \mathbf{M})$ *be local sparsities and sparsity levels respectively. For weights* $\{w_i\}_{i=1}^N$ $(w_i > 0)$*, we say that* $A \in \mathbb{C}^{m\times N}$ *satisfies the weighted robust null space property in levels (weighted rNSPL) of order* $(\mathbf{s}, \mathbf{M})$ *with constants* $0 < \rho < 1$ *and* $\gamma > 0$ *if for any* $(\mathbf{s}, \mathbf{M})$ *support set* Δ*,*

$$\|x_\Delta\|_{l^2} \leq \rho \|x_{\Delta^c}\|_{l^1_w} / \sqrt{\xi} + \gamma \|Ax\|_{l^2}, \qquad \textit{for all } x \in \mathbb{C}^N.$$

A simplified version of Theorem 5.5 is (constants are independent of n and given in Theorem 5.5):

Simplified version of Theorem 5.5 (Computing stable and accurate NNs with exponential convergence). *There exists an algorithm such that for any input sparsity parameters* $(\mathbf{s}, \mathbf{M})$*, weights* $\{w_i\}_{i=1}^N$*,* $A \in \mathbb{C}^{m\times N}$ *(with the input* A *given by* $\{A_l\}$*) satisfying the rNSPL with constants* $0 < \rho < 1$ *and* $\gamma > 0$ *(also input), and input parameters* $n \in \mathbb{N}$ *and* $\{\delta, b_1, b_2\} \subset \mathbb{Q}_{>0}$*, the algorithm outputs a neural network* ϕ_n *with* $\mathcal{O}(n)$ *layers and the following property. For any* $x \in \mathbb{C}^N$ *and* $y \in \mathbb{C}^m$ *with*

$$\sigma_{\mathbf{s},\mathbf{M}}(x)_{l^1_w} + \|Ax - y\|_{l^2} \lesssim \delta, \quad \|x\|_{l^2} \lesssim b_1, \quad \|y\|_{l^2} \lesssim b_2, \quad \textit{we have}$$

$$\|\phi_n(y) - x\|_{l^2} \lesssim \delta + e^{-n} \quad \textit{(exponential convergence in } n\textit{).}$$

(This is also often called "linear convergence" or "geometric convergence".)

Hence, up to the small error term $\sigma_{\mathbf{s},\mathbf{M}}(x)_{l^1_w}$, and in the limit $n \to \infty$ (with exponential convergence), we recover x stably with an error proportional to the measurement error $\|Ax - y\|_{l^2}$. Further interpretations of the terms in this bound can be found in Remark 5.3. Theorem 5.5 also bounds the error when we only approximately apply the nonlinear maps of the NNs: in other words, we also gain a form of stability of the forward pass of the NN. In addition to providing stability, it is precisely the rNSPL that allows us to prove exponential convergence through a careful restarting and reweighting scheme. We therefore call our NNs Fast Iterative REstarted NETworks (FIRENETs). Note that, even when ignoring issues such as stability and inexact arithmetic, a naive unrolling of iterative methods commonly used in compressed sensing gives a slow first-order convergence rate with error bounds $\mathcal{O}(\delta + n^{-1})$ (and in certain regimes second-order $\mathcal{O}(\delta + n^{-2})$): see Remark 5.7. In contrast to naive unrolling and previous approaches, we use novel restarting tools and analysis of the square-root LASSO problem (P_3) to gain good theoretical bounds (accuracy and stability).

Remark 3.3 (Logarithmic grid search when parameters are unknown)**.** In the case that we do not know ρ or γ (the constants in the definition of rNSPL), we can perform a log-scale grid search for suitable parameters. Given a total budget of $\mathcal{O}(n \log(n))$ layers, we can still gain exponential convergence in n by choosing the parameters in the grid search that lead to the vector with minimal F_3^A (the objective function of (P_3)). In some cases, such as Theorem 5.10, it is possible to prove probabilistic results where ρ and γ are known. ⊠

In §5.3, we apply Theorem 5.5 to examples in compressive imaging (see §6 for computational examples), with A arising from multilevel random subsampling (Definition 5.8) of Fourier and Walsh (or binary) measurements, and sparsity in levels measured in terms of wavelet coefficients. Theorem 5.10 gives precise bounds on the number of samples needed, m, and the number of layers of a NN needed to achieve a specified accuracy:

Simplified version of Theorem 5.10. *We show that stable and accurate (with exponential convergence in the number of layers) neural networks for image reconstruction can be constructed via an algorithm, with the number of samples needed the same, up to logarithmic terms, as the current best-known oracle estimators.*

4. FIRENET: BALANCING THE TRADE-OFF BETWEEN STABILITY AND ACCURACY

As demonstrated in [9], current DL methods for image reconstruction can be unstable in the sense that (1) a tiny perturbation, in either the image or sampling domain, can cause severe artefacts in the reconstructed image (instability - see Figure 1), and/or (2) a tiny detail in the image domain might be washed out in the reconstructed image (lack of accuracy), resulting in potential false negatives. Inevitably, there is a stability-accuracy trade-off, for this type of linear inverse problem, making it impossible for any reconstruction method to become arbitrarily stable without sacrificing accuracy or visa versa (see §3). In this section, we show that the NNs computed by our algorithm (FIRENETs) are stable with respect to adversarial perturbations and accurate for images which are sparse in wavelets. As most images are sparse in wavelets, these networks also show great generalisation properties to unseen images.

Computing worst-case (adversarial) perturbations – First we describe the algorithm developed in [9] for computing perturbations meant to simulate worst-case effect in terms of reconstruction artefacts. It is this algorithm which has been used to compute the perturbed images, seen in Figures 1, 2, 4, and 5. The algorithm does the following. Given an image $x \in \mathbb{C}^N$ and a NN $\phi \in \mathcal{N}_{m,N}$, designed for image reconstruction from samples y provided by a specific sampling modality described by the matrix A, the algorithm searches for a perturbation of the image that makes the most severe change in the output of the network while still keeping the perturbation small. The algorithm seeks a vector $r \in \mathbb{C}^N$ such that $\|\phi(y+Ar)-\phi(Ax)\|_{l^2}$ is large, while $\|r\|_{l^2}$ is small. Specifically, consider the optimisation problem

$$Q_y^\phi(r) = \frac{1}{2}\|\phi(y+Ar)-x\|_{l^2}^2 - \frac{\lambda}{2}\|r\|_{l^2}^2, \quad r^*(y) \in \operatorname*{argmax}_r Q_y^\phi(r). \tag{4.1}$$

The problem (4.1) seeks perturbations in the image domain since this provides an easy way to compare the original image and deduce whether the reconstruction of the perturbed image is acceptable/unacceptable. Of course, we could have just as easily considered perturbations in the sampling domain instead.

The non-concavity of Q_y^ϕ means that finding a global maximiser of (4.1) is very difficult (if not impossible), even for small m and N. The test aims to locate local maxima of (4.1) by using a gradient search. A natural method to find local maxima is gradient ascent with momentum. This uses the gradient of Q_y^ϕ (which can easily be written down) along with two parameters $\gamma > 0$ (the momentum) and $\eta > 0$ (the learning rate) in each step towards a local maximum. Namely, $r(0)$ is initialised randomly and then we update the perturbation at the jth step via $v(j+1) = \gamma v(j) + \eta \nabla_r Q_y^\phi(r(j))$ and $r(j+1) = r(j) + v(j+1)$. The final perturbation is taken after M steps, where typically we run 10-100 steps, seeking the perturbation which causes the worst reconstructed image. Just as in the case when training NNs using stochastic gradient descent, choosing the parameters γ and η is an art of engineering, and the optimal choices of γ, η are based on empirical testing.

Worst-case (adversarial) perturbations for AUTOMAP and FIRENETs – Figure 1 in the introduction shows the algorithm applied to the AUTOMAP [113] network used for MRI reconstruction with 60% subsampling. The network weights are provided by the authors of [113] and had been trained on de-identified brain images from the MGH–USC HCP dataset [49], where the image measurements $y = Ax + e$ were contaminated with small Gaussian noise e. The image x seen in Figure 1 is taken from the mentioned dataset, the algorithm is run on the AUTOMAP network to find a sequence of perturbations $|r_1| < |r_2| < |r_3|$. In order to illustrate the smallness of the perturbations, we have visualised $|x + r_j|$ in the first row of Figure 1. As can

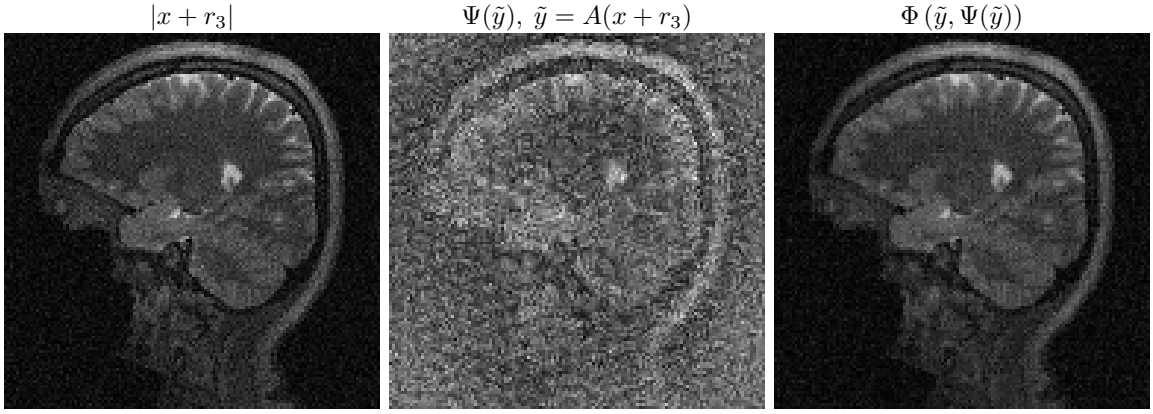

FIGURE 3. (**Adding a few FIRENET layers at the end of AUTOMAP makes it stable**). The FIRENET $\Phi\colon \mathbb{C}^m \times \mathbb{C}^N \to \mathbb{C}^N$ takes as input measurements $y \in \mathbb{C}^m$ and an initial guess for x, which we call $x_0 \in \mathbb{C}^N$. We now concatenate a 25-layer ($p = 5$, $n = 5$) FIRENET Φ and the AUTOMAP network $\Psi\colon \mathbb{C}^m \to \mathbb{C}^N$, by using the output from AUTOMAP as initial guess x_0, i.e., we consider the neural network mapping $y \mapsto \Phi(y, \Psi(y))$. In this experiment we consider the perturbed image $x + r_3$ from Figure 1 and the perturbed measurements $\tilde{y} = A(x + r_3)$ (here A is as in Figure 1). As can be seen from the figure, the new network is stable with respect to AUTOMAP's worst-case perturbation r_3. Note that in all other experiments we use the initial guess $x_0 = 0$, and consider Φ as a mapping $\Phi\colon \mathbb{C}^m \to \mathbb{C}^N$.

be seen from the second row in the figure, the network reconstruction completely deforms the image and the reconstruction is severely unstable (similar results for other networks are demonstrated in [9]).

In contrast, we have applied the same algorithm, but now for the new NNs (FIRENETs) reported in this paper. Figure 2 shows the algorithm applied to the constructed FIRENETs described by Theorems 5.5 and 5.10 (we have renamed the perturbations v_j to emphasise the fact that these perturbations are sought for the new NNs and have nothing to do with the adversarial perturbations in Figure 1). We now see that despite the search for adversarial perturbations, the reconstruction remains stable. The error in the reconstruction was also found to be at most of the same order of the perturbation (as expected from the stability in Theorems 5.5 and 5.10). In applying the test to FIRENETs, we tested/tuned the parameters in the gradient ascent algorithm considerably (much more so than was needed for applying the test to AUTOMAP, where finding instabilities was straightforward) in order to find the worst reconstruction results, yet the reconstruction remained stable. Finally, it should be mentioned that this search algorithm is just one form of test and it is likely that there are many other tests for creating instabilities for NNs for inverse problems. This highlights the importance of results such as Theorems 5.5 and 5.10, which guarantee stability regardless of the perturbation.

Stabilising unstable NNs with FIRENETs – Our NNs also act as a stabiliser. For example, Figure 3 shows the adversarial example for AUTOMAP (taken from Figure 1), but now shows what happens when we take the reconstruction from AUTOMAP as an input to our FIRENETs. Here we are using the fact that we can view our networks as approximations of unrolled (or unfolded) and restarted iterative methods, allowing us to use the output of AUTOMAP as the initial image for the reconstruction. We see that FIRENETs fix the output of AUTOMAP and stabilise the reconstruction.

Generalisation – To demonstrate the generalisation properties of our NNs, Figure 4 shows the stability test applied to FIRENETs for a range of images. This shows stability across different types of images and highlights an important fact. Namely, methods based on the conditions in §3 allow great generalisation properties and avoid time-consuming and expensive retraining of NNs for different classes of images.

The accuracy/stability trade-off and false negatives – It is easy to produce a perfectly stable network: the zero network is the obvious candidate! However, this network would obviously have poor performance and produce many false negatives. The challenge is to simultaneously ensure performance and stability. Figure 5 highlights this issue. Here we have trained two NNs to recover a set of ellipses images from noise-free and

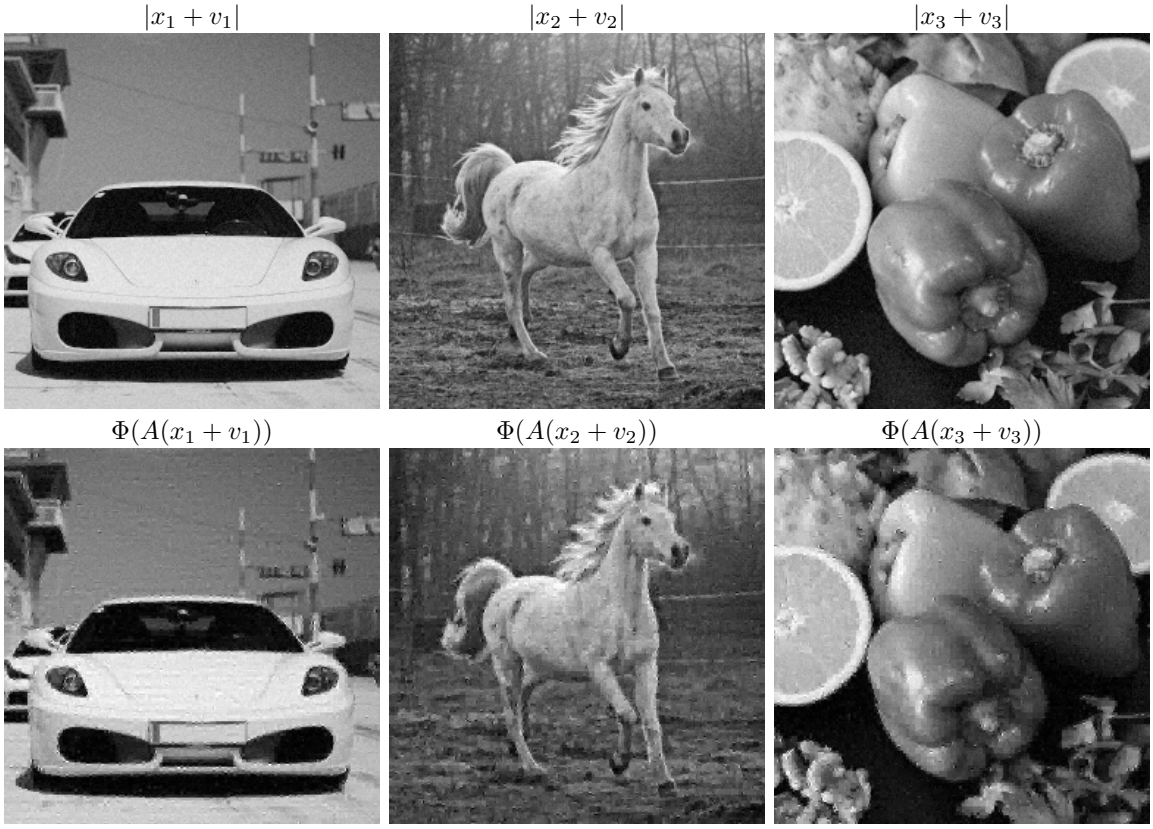

FIGURE 4. (**FIRENET withstand worst-case perturbations and generalises well**). To show that FIRENET generalises well and is stable, we consider three different images x_j, $j = 1, 2, 3$. For each image x_j we compute a perturbation v_j meant to simulate worst-case effect for a FIRENET Φ with $n = 5$ and $p = 5$. The first row shows the perturbed images $x_j + v_j$, whereas the second row shows the FIRENET reconstructions from data $A(x_j + v_j)$. Here $A \in \mathbb{C}^{m\times N}$ is a subsampled discrete Fourier transform with $m/N = 0.25$ and $N = 256^2$. The perturbations v_j have magnitude $\|Av_j\|_{l^2}/\|Ax_j\|_{l^2} \geq 0.05$ in the measurement domain.

noisy Fourier measurements. The noise-free measurements are generated as $y = Ax$, where $A \in \mathbb{C}^{m\times N}$ is a subsampled discrete Fourier transform, with $m/N = 0.15$ and $N = 1024^2$. The noisy measurements are generated as $y = Ax + ce$, where A is as before, and the real and imaginary components of $e \in \mathbb{C}^m$ are drawn from a zero mean and unit variance normal distribution $\mathcal{N}(0, 1)$, and $c \in \mathbb{R}$ is drawn from the uniform distribution $\mathrm{Unif}([0, 100])$. The noise $ce \in \mathbb{C}^m$, is generated on the fly during the training process.

The trained networks are using a standard benchmarking architecture for image reconstruction, and maps $y \mapsto \phi(A^*y)$, where $\phi\colon \mathbb{C}^N \to \mathbb{R}^N$ is a trainable U-net NN [71, 80]. Training networks with noisy measurements, using for example this architecture, have previously been used as an example of how to create NNs which are robust towards adversarial attacks [55]. As we can see from Figure 5 (bottom row) this is the case, as it does indeed create a NN which is stable with respect to worst-case perturbations. However, a key issue is that it is also producing false negatives due to its inability to reconstruct details. Similarly, as reported in the 2019 FastMRI challenge, trained NNs that performed well in terms of standard image quality metrics were prone to false negatives: they failed to reconstruct small, but physically-relevant image abnormalities [74]. Pathologies, generalisation and AI-generated hallucinations were subsequently a focus of the 2020 challenge [91]. FIRENET, on the other hand, has a guaranteed performance (on images being sparse in wavelet bases) and stability, given specific conditions on the sampling procedure. The challenge

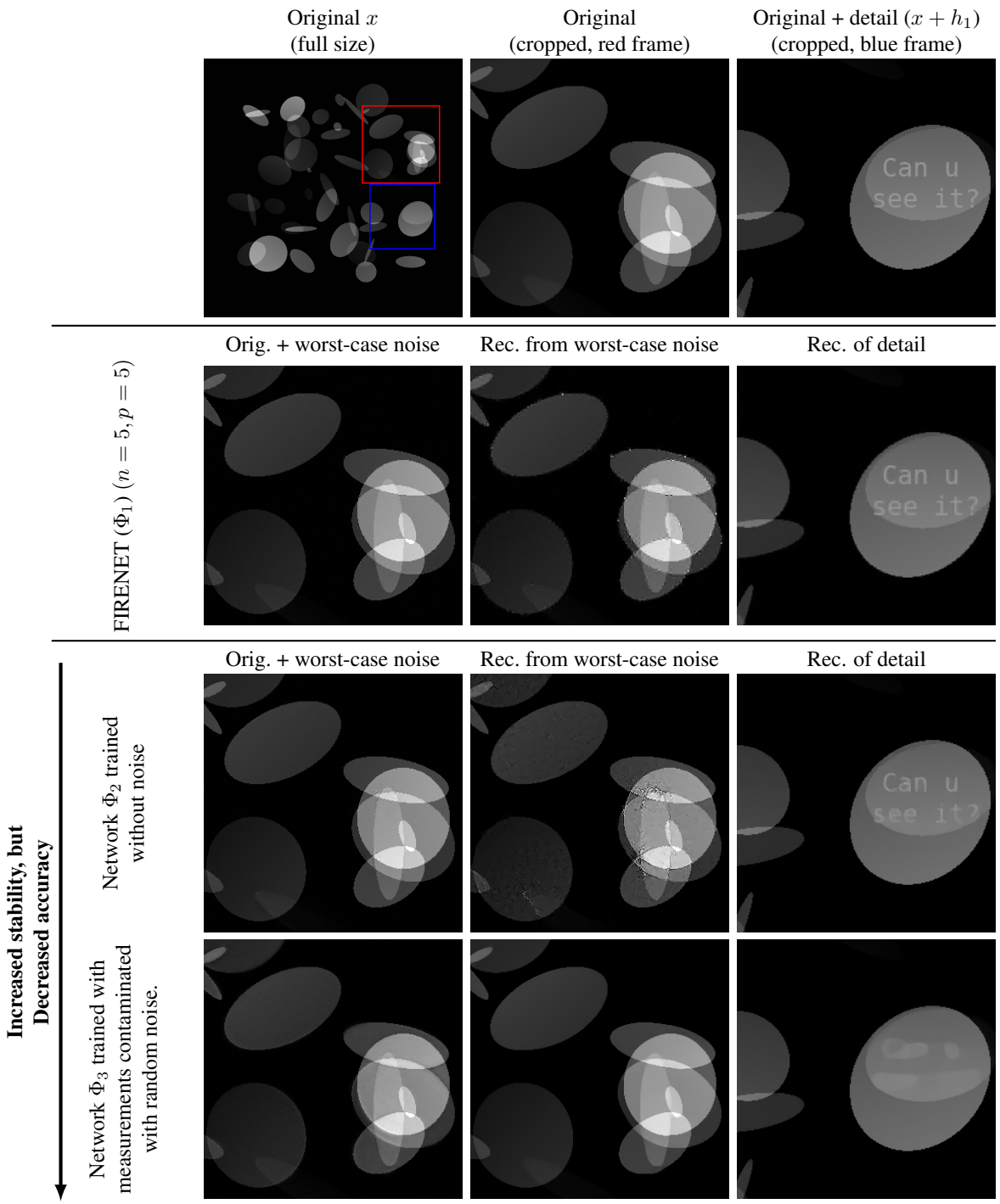

FIGURE 5. (**Trained neural networks with limited performance can be stable**). We examine the accuracy/stability trade-off for linear inverse problems by considering three reconstruction networks $\Phi_j \colon \mathbb{C}^m \to \mathbb{C}^N$, $j = 1, 2, 3$. Here Φ_1 is a FIRENET, whereas Φ_2 and Φ_3 are trained NNs, trained without and with noisy measurements, respectively (see §4 for details). For each network, we compute a perturbation $w_j \in \mathbb{C}^N$ meant to simulate the worst-case effect, and we show a cropped version of the perturbed images $x + w_j$ in the first column (row 2-4). In the middle column (row 2-4), we show the reconstructed images $\Phi_j(A(x + w_j))$ from each of the networks. In the last column (row 2-4) we test the networks ability to reconstruct a tiny detail h_1, in the form of the text "can u see it?". As we see from the figure, the network trained on noisy measurements is stable to worst-case perturbations, but it is not accurate. Conversely, the network trained without noise is accurate but not stable. The FIRENET is balancing this trade-off and is accurate for images which are sparse in wavelets, and stable to worst-case perturbations.

is to determine the optimal balance between accuracy/stability, a problem that is well known in numerical analysis.

5. Main Results III: Precise formulations of Theorem 5.5 and Theorem 5.10

5.1. **Neural networks and notational conventions.** To state our theorems, we need to be precise about the definition of a NN. For introductions to the field of DL and NNs, we refer the reader to [68, 78] and [92], respectively, and the references therein. To capture standard architectures used in practice such as skip connections, we consider the following definition of a NN. Without loss of generality and for ease of exposition, we also work with complex-valued NNs. Such networks can be realised by real-valued NNs by splitting into real and imaginary parts. A NN is a mapping $\phi\colon \mathbb{C}^m \to \mathbb{C}^N$ that can be written as a composition

$$\phi(y) = V_T(\rho_{T-1}(...\rho_1(V_1(y)))), \quad \text{where:}$$

- Each V_j is an affine map $\mathbb{C}^{N_{j-1}} \to \mathbb{C}^{N_j}$ given by $V_j(x) = W_j x + b_j(y)$ where $W_j \in \mathbb{C}^{N_j \times N_{j-1}}$ and the $b_j(y) = R_j y + c_j \in \mathbb{C}^{N_j}$ are affine functions of the input y.
- Each $\rho_j\colon \mathbb{C}^{N_j} \to \mathbb{C}^{N_j}$ is one of two forms:
  (i) There exists an index set $I_j \subset \{1, ..., N_j\}$ (possibly a strict subset) such that ρ_j applies a possibly non-linear function $f_j : \mathbb{C} \to \mathbb{C}$ element-wise on the input vector's components with indices in I_j:

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

  (ii) There exists a possibly non-linear function $f_j : \mathbb{C} \to \mathbb{C}$ such that, after decomposing the input vector x as $(x_0, X^\top, Y^\top)^\top$ ($\top$ denotes transpose) for scalar x_0 and $X \in \mathbb{C}^{m_j}$ ($Y \in \mathbb{C}^{N_j - 1 - m_j}$), we have

$$\rho_j : \begin{pmatrix} x_0 \\ X \\ Y \end{pmatrix} \to \begin{pmatrix} 0 \\ f_j(x_0)X \\ Y \end{pmatrix}. \tag{5.1}$$

The affine dependence of $b_j(y)$ on y allows skip connections from the input to the current level as in standard definitions of feed-forward NNs [99, p. 269], and the above architecture has become standard [45, 63, 71].

Remark 5.1 (On the use of multiplication)**.** The use of non-linear functions of the form (ii) may be re-expressed using the following element-wise squaring trick:

$$\begin{pmatrix} x_0 \\ X \\ Y \end{pmatrix} \to \begin{pmatrix} f_j(x_0) \\ X \\ Y \end{pmatrix} \to \begin{pmatrix} f_j(x_0)\mathbf{1} \\ X \\ f_j(x_0)\mathbf{1} + X \\ Y \end{pmatrix} \to \begin{pmatrix} f_j(x_0)^2\mathbf{1} \\ X^2 \\ [f_j(x_0)\mathbf{1} + X]^2 \\ Y \end{pmatrix} \to \begin{pmatrix} 0 \\ \frac{1}{2}\left[[f_j(x_0)\mathbf{1} + X]^2 - f_j(x_0)^2\mathbf{1} - X^2\right] \\ Y \end{pmatrix},$$

where $\mathbf{1}$ denotes a vector of ones of the same size as X (so that $f_j(x_0) \to f_j(x_0)\mathbf{1}$ is a linear map). However, this is not done in practice since (5.1) is directly trainable via backpropagation. ⊠

Note that we do not allow the matrices W_j to depend on y. We denote the collection of all NNs of the above form by $\mathcal{N}_{\mathbf{D},T,q}$, where the vector $\mathbf{D} = (N_0 = m, N_1, ..., N_T = N)$ denotes the dimensions in each layer, T denotes the number of layers and q denotes the number of different non-linear functions applied (including the count of different I_j and m_j). In general, we will require that the layer sizes N_j do not grow with j so that the size of each layer is of the same order as the sampling matrix A.

We consider stable reconstruction from noisy undersampled measurements, as in (2.1), and NNs that can be constructed via algorithms. To make this precise, we assume that we have access to a sequence of matrices $A_l \in \mathbb{Q}[i]$ such that $\|A - A_l\| \leq q_l$ for some known null sequence $\{q_l\}$. This is consistent with the training set given by (2.8). To construct NNs via an algorithm, care must be taken with the non-linear activation functions. We assume that for $\theta \in \mathbb{Q}_{>0}$ we have access to a routine "sqrt_θ" such that $|\mathrm{sqrt}_\theta(x) - \sqrt{x}| \leq \theta$ for all $x \in \mathbb{R}_{\geq 0}$. In what follows, the non-linear maps f_j used in the NNs are either arithmetic or constructed

using arithmetic operations and sqrt_θ. We always ensure that sqrt_θ acts on non-negative real numbers and on rational inputs if the input to the NN is rational. We refer to the pair (ϕ, θ) as a NN.

Remark 5.2 (Approximating $\sqrt{\cdot}$ with neural networks)**.** On any bounded set (for bounded input our constructed NNs only require the routine sqrt_θ on a bounded set), we can construct an approximation to $\sqrt{\cdot}$ using standard non-linear activation functions such as ReLU (more efficient approximations may be achieved by using other activation functions such as rational maps [29]). The choice of the square root function is somewhat arbitrary, but simplifies our proof of Theorem 5.5. Similar results hold for other activation functions. ⊠

Remark 5.3 (An interpretation of θ)**.** As well as being necessary from a foundations point of view, an important interpretation of θ is numerical stability, or accumulation of errors, of the forward pass of the NN. Larger values of θ show greater stability when applying the NN in finite precision. We prove results of the form

$$\|\phi_n(y) - x\|_{l^2} \leq \varepsilon + c_1(A, x)\|Ax - y\|_{l^2} + c_2(A, x)\upsilon^n, \quad \forall x \in S \subset \mathbb{C}^N, y \in \mathbb{C}^m, \tag{5.2}$$

where (ϕ_n, θ_n) is a (sequence of) NN(s) with $\mathcal{O}(n)$ layers that is computed by an algorithm, $\upsilon \in (0, 1)$ describes the exponential rate of convergence in the number of layers, $\varepsilon > 0$ (in our results ε will be related to the distance to vectors that are sparse in levels: $\sigma_{\mathbf{s},\mathbf{M}}(x)_{l^1_w}$ in Definition 3.1), and $\theta_n^{-1} = \theta^{-1}$ is bounded independent of n. Up to the error tolerance ε, the constant $c_1(A, x)$ can be thought of as an asymptotic *local Lipschitz constant* for the NNs as $n \to \infty$, and thus measures *stability of inexact input* y. In practice one would use floating point arithmetic to approximate square roots. Hence, the boundedness of θ_n^{-1} is a *numerical notion of stability* - the accuracy needed for approximating square roots (and the non-linear maps) does not become too great and errors do not accumulate as n increases. Moreover, in practice the value of θ^{-1} needed is well below what is achieved using standard floating-point formats. ⊠

5.2. **The construction of stable and accurate neural networks.** The main result of this section, Theorem 5.5, uses the concept of sparsity in levels and weighted robust null space property in levels defined in §3. We also define the following quantities:

$$\xi = \xi(\mathbf{s}, \mathbf{M}, w) \coloneqq \sum_{k=1}^{r} w_{(k)}^2 s_k, \quad \zeta = \zeta(\mathbf{s}, \mathbf{M}, w) \coloneqq \min_{k=1,\dots,r} w_{(k)}^2 s_k, \quad \kappa = \kappa(\mathbf{s}, \mathbf{M}, w) \coloneqq \xi/\zeta.$$

Unless there is ambiguity, we will drop the $(\mathbf{s}, \mathbf{M}, w)$ from the notation of these parameters. Recall the setup throughout this paper of a matrix $A \in \mathbb{C}^{m\times N}$ $(m < N)$, where we have access to an approximation sequence A_l such that $\|A - A_l\| \leq q_l$ with known $q_l \to 0$ as $l \to \infty$. In this regard, the following simple perturbation lemma is useful (whose proof is given in §9).

Lemma 5.4 (**The weighted rNSP in levels is preserved under perturbations or approximations**)**.** *Assume that (3.2) holds and that A satisfies the weighted rNSPL of order $(\mathbf{s}, \mathbf{M})$ with constants $0 < \rho < 1$ and $\gamma > 0$. Let $\hat{A}$ be an approximation of A such that $\|\hat{A} - A\| < (1 - \rho)\gamma^{-1}\left(1 + \frac{\sqrt{\xi}}{\min_{k=1,\dots,r} w_{(k)}}\right)^{-1}$. Then $\hat{A}$ satisfies the weighted rNSPL of order $(\mathbf{s}, \mathbf{M})$ with new constants*

$$\hat{\rho} = \frac{\rho + \gamma\sqrt{\xi}\|\hat{A} - A\|/\min_{k=1,\dots,r} w_{(k)}}{1 - \gamma\|\hat{A} - A\|}, \quad \hat{\gamma} = \frac{\gamma}{1 - \gamma\|\hat{A} - A\|}.$$

Lemma 5.4 says that if A satisfied the weighted rNSPL of order $(\mathbf{s}, \mathbf{M})$, then so does A_l for large enough l. Moreover, given the sequence $\{q_l\}$, we can compute how large l must be and the new constants. For ease of exposition, we drop the notational hats from these constants. We can now state our main result, proven in §9.

Theorem 5.5 (**Stable and accurate neural networks with uniform recovery guarantees can be constructed**)**.** *There exists an algorithm such that for any input sparsity parameters $(\mathbf{s}, \mathbf{M})$, weights $\{w_i\}_{i=1}^N$, $A \in \mathbb{C}^{m\times N}$ (with the input A given by $\{A_l\}$) satisfying the rNSPL with constants $0 < \rho < 1$ and $\gamma > 0$ (also input), and input parameters $n \in \mathbb{N}$, $\{\delta, b_1, b_2\} \subset \mathbb{Q}_{>0}$ and $\upsilon \in (0, 1) \cap \mathbb{Q}_{>0}$, the algorithm outputs a neural*

network ϕ_n such that the following holds. For

$$C_1 = \left(\frac{1+\rho}{2} + (3+\rho)\frac{\kappa^{1/4}}{4}\right)\left(\frac{3+\rho}{1-\rho}\right) \sim \frac{\kappa^{1/4}}{1-\rho}, \qquad C_2 = 2\left(\frac{3+\rho}{1-\rho} + \frac{7+\rho}{1-\rho}\frac{\kappa^{1/4}}{2}\right)\gamma \sim \frac{\kappa^{1/4}\gamma}{1-\rho},$$

(1) (Size) $\phi_n \in \mathcal{N}_{\mathbf{D}_{(n,p)},3np+1,3}$ with $\mathbf{D}_{(n,p)} = (m, \underbrace{2N+m, 2(N+m), 2N+m+1}_{np \text{ times}}, N)$, where $p \in \mathbb{N}$ with the bound $p \leq \left\lceil \frac{3C_2\|A\|}{\upsilon} \right\rceil$. Moreover, $\theta^{-1} \sim p^2(1+\|w\|_{l^2})\max\left\{1, \frac{\|w\|_{l^2}}{\|A\|\gamma\sqrt{\xi}}\right\}$.

(2) (Exponentially Convergent, Uniform and Stable Recovery) For any pair $(x,y) \in \mathbb{C}^N \times \mathbb{C}^m$ with

$$\frac{2C_1}{C_2\sqrt{\xi}} \cdot \sigma_{\mathbf{s},\mathbf{M}}(x)_{l^1_w} + 2\|Ax-y\|_{l^2} \leq \delta, \quad \|x\|_{l^2} \leq b_1, \quad \|y\|_{l^2} \leq b_2,$$

we have the following exponentially convergent, uniform and stable recovery guarantees:

$$\|\phi_n(y) - x\|_{l^2} \leq \frac{2C_1}{\sqrt{\xi}} \cdot \sigma_{\mathbf{s},\mathbf{M}}(x)_{l^1_w} + 2C_2 \cdot \|Ax-y\|_{l^2} + \left(\frac{1+\upsilon}{1-\upsilon}\right) C_2 \cdot \delta + b_2 C_2 \cdot \upsilon^n, \tag{5.3}$$

$$\|\phi_n(y) - x\|_{l^1_w} \leq \left(\frac{3+\rho}{1-\rho}\right)\frac{\sqrt{\xi}}{C_1}\left(\frac{2C_1}{\sqrt{\xi}} \cdot \sigma_{\mathbf{s},\mathbf{M}}(x)_{l^1_w} + 2C_2 \cdot \|Ax-y\|_{l^2} + \left(\frac{1+\upsilon}{1-\upsilon}\right) C_2 \cdot \delta + b_2 C_2 \cdot \upsilon^n\right). \tag{5.4}$$

Remark 5.6 (The optimal choice of υ)**.** For a total budget of $T = 3pn+1$ layers,

$$\upsilon^n = \exp\left(\frac{(T-1)}{3}\left\lceil \frac{3C_2\|A\|}{\upsilon}\right\rceil^{-1} \log(\upsilon)\right)$$

If we ignore the ceiling function, the optimal choice is $\upsilon = e^{-1}$ (strictly speaking Theorem 5.5 is only stated for rational υ, but we can easily approximate e^{-1}). This yields the error term $\upsilon^n = \exp\left(-\frac{(T-1)}{3}\lceil 3C_2 e\|A\|\rceil^{-1}\right)$ and exponential convergence in the number of layers T. This is not optimal. For example, a study of the proof of Theorem 5.5 shows that we can replace $3C_2$ in the exponential by

$$2\left(\frac{1+\rho}{1-\rho} + \frac{3+\rho}{1-\rho}\frac{\kappa^{1/4}}{2}\right)\gamma + \epsilon$$

for arbitrary $\epsilon > 0$. Suppose that we want $b_2C_2 \cdot r^n \sim \delta$, then the number of layers required is proportional to $C_2\|A\|\log(b_2\delta^{-1})$, and only grows logarithmically with the precision δ^{-1}. This is made precise in Theorem 5.10, where we apply Theorem 5.5 to examples in compressive imaging. ⊠

The proof of Theorem 5.5 uses the optimisation problem (P_3) (defined in (2.2)), in the construction of ϕ_n. It is also possible to prove similar results using (P_1) and (P_2), but we do not provide the details. The NNs constructed are approximations of unrolled primal-dual iterations for (P_3), with a careful restart scheme to ensure exponential convergence in the number of layers. Pseudocode is provided in 6, as well as computational experiments. The bounds in (5.3) and (5.4) are not quite optimal. If we were able to work in exact arithmetic (taking $\theta \to 0$ and $A_l \to A$), we obtain slightly smaller constants, though these do not affect the asymptotic rates. The pseudocode is written with $\theta \to 0$ and $A_l \to A$, and slightly different parameters accordingly. The approach of unrolling iterative methods as NNs has a rich history in DL, as discussed in §7.

Remark 5.7 (What happens without restart or with unknown δ?)**.** Without the restart scheme, the convergence in the number of layers scales as $\mathcal{O}(n^{-1})$. However, one can get rid of the assumption $2C_1/(C_2\sqrt{\xi})\sigma_{\mathbf{s},\mathbf{M}}(x)_{l^1_w} + 2\|Ax-y\|_{l^2} \leq \delta$. (The assumption is to ensure that the reweighting of the restarts do not become too small - in practice, we found that this was not an issue and the assumption was not needed, with (up to small constants) δ replaced by $2C_1/(C_2\sqrt{\xi})\sigma_{\mathbf{s},\mathbf{M}}(x)_{l^1_w} + 2\|Ax-y\|_{l^2}$ in (5.3). See also the discussion in §6.) More precisely, the proof of Theorem 5.5 can be adapted to show the following. For an additional input $\beta \in \mathbb{Q}_{>0}$ (and without inputs b_2, δ and υ), there exists an algorithm that computes $\widehat{\phi}_n \in \mathcal{N}_{\mathbf{D}_n,3n+1,3}$ with

$$\mathbf{D}_n = (m+N, \underbrace{2N+m, 2(N+m), 2N+m+1}_{n \text{ times}}, N),$$

such that for any $x, x_0 \in \mathbb{C}^N$ with $\|x\|_{l^2} \leq b_1$ and all $y \in \mathbb{C}^m$, the following reconstruction guarantees hold:

$$\|\widehat{\phi}_n(y, x_0) - x\|_{l^2} \leq \frac{2C_1}{\sqrt{\xi}} \sigma_{\mathbf{s},\mathbf{M}}(x)_{l^1_w} + 2C_2 \left[\|Ax - y\|_{l^2} + \frac{\|A\|}{n} \left(\frac{\|x - x_0\|^2_{l^2}}{\beta} + \beta \right) \right], \tag{5.5}$$

$$\|\widehat{\phi}_n(y, x_0) - x\|_{l^1_w} \leq \left(\frac{3+\rho}{1-\rho} \right) \frac{\sqrt{\xi}}{C_1} \left(\frac{2C_1}{\sqrt{\xi}} \sigma_{\mathbf{s},\mathbf{M}}(x)_{l^1_w} + 2C_2 \left[\|Ax - y\|_{l^2} + \frac{\|A\|}{n} \left(\frac{\|x - x_0\|^2_{l^2}}{\beta} + \beta \right) \right] \right). \tag{5.6}$$

Here, x_0 should be interpreted as an initial guess (an arbitrary input to the NNs) and β should be interpreted as a scaling parameter, with optimal scaling $\beta \sim \|x - x_0\|_{l^2}$. A good choice for β is $\|x\|_{l^2}$, or, when this is unknown, $\|y\|_{l^2}/\|A\|$. For completeness, we provide a proof sketch of (5.5) and (5.6) at the end of §9.3. ⊠

Algorithm unrolling is particularly well-suited to scenarios where it is difficult to collect large training samples. However, training a finite fixed number of layers typically incurs the same stability and generalisation issues mentioned above. Moreover, learning the weights and biases usually prevents the convergence analysis of standard (unlearned) iterative methods carrying over. In particular, there is no guarantee of objective function minimisation (let alone convergence of the iterated arguments) or any form of convergence as the number of layers increases. A subtle, yet fundamental, point regarding iterative methods, whether they are unrolled as a NN and supplemented with learned parameters or not, is the following. Theorem 2.2 states that, in general, the optimisation problems (P_1), (P_2), and (P_3) are non-computable. This is despite the fact that there are many results in the literature describing rates of convergence for iterative methods. The resolution of this apparent puzzle is that convergence results regarding iterative methods are typically given in terms of the *objective function* that is being minimised (see also Theorem 9.5, which we use to prove Theorem 5.5). As the proof of Theorem 5.5 shows, it is crucial to have conditions such as the rNSPL to convert these objective function bounds to the desired error bounds on the distance to the *minimisers* or vector x. Moreover, this property has the key effect of allowing exponential convergence through restarting and reweighting.

5.3. **Examples in compressive imaging.** As an example application of Theorem 5.5, we consider the case of Fourier and Walsh sampling, using the Haar wavelets as the sparsifying transform. Our results can be generalised to the infinite-dimensional setting with the use of higher-order Daubechies wavelets (though the results are more complicated to write down), and we refer the reader to [4] for compressed sensing in infinite dimensions. We first define the concept of multilevel random subsampling [5].

Definition 5.8 (Multilevel random subsampling)**.** *Let $\mathbf{N} = (N_1, \dots, N_l) \in \mathbb{N}^l$, where $1 \leq N_1 < \cdots < N_l = N$ and $\mathbf{m} = (m_1, \dots, m_l) \in \mathbb{N}^l$ with $m_k \leq N_k - N_{k-1}$ for $k = 1, \dots, l$, and $N_0 = 0$. For each $k = 1, \dots, l$, let $\mathcal{I}_k = \{N_{k-1} + 1, \dots, N_k\}$ if $m_k = N_k - N_{k-1}$ and if not, let $t_{k,1}, \dots, t_{k,m_k}$ be chosen uniformly and independently from the set $\{N_{k-1} + 1, \dots, N_k\}$ (with possible repeats), and set $\mathcal{I}_k = \{t_{k,1}, \dots, t_{k,m_k}\}$. If $\mathcal{I} = \mathcal{I}_{\mathbf{N},\mathbf{m}} = \mathcal{I}_1 \cup \cdots \cup \mathcal{I}_l$ we refer to $\mathcal{I}$ as an $(\mathbf{N}, \mathbf{m})$-multilevel subsampling scheme.*

Definition 5.9 (Multilevel subsampled unitary matrix)**.** *A matrix $A \in \mathbb{C}^{m \times N}$ is an $(\mathbf{N}, \mathbf{m})$-multilevel subsampled unitary matrix if $A = P_{\mathcal{I}} D U$ for a unitary matrix $U \in \mathbb{C}^{N \times N}$ and $(\mathbf{N}, \mathbf{m})$-multilevel subsampling scheme $\mathcal{I}$. Here D is a diagonal scaling matrix with diagonal entries*

$$D_{ii} = \sqrt{\frac{N_k - N_{k-1}}{m_k}}, \quad i = N_{k-1} + 1, ..., N_k, \quad k = 1, ..., l$$

and $P_{\mathcal{I}}$ denotes the projection onto the linear span of the subset of the canonical basis indexed by $\mathcal{I}$.

Throughout this subsection, we let $K = 2^r$ for $r \in \mathbb{N}$, and consider vectors on $\mathbb{C}^K$ or d-dimensional tensors on $\mathbb{C}^{K \times \cdots \times K}$. To keep consistent notation with previous sections, we set $N = K^d$ so that the objective is to recover a vectorised $x \in \mathbb{C}^N$. The following can also be generalised to rectangles (i.e. $\mathbb{C}^{2^{r_1} \times \cdots \times 2^{r_d}}$ with possibly different $r_1, ..., r_d$) or dimensions that are not powers of two.

Let $V \in \mathbb{C}^{N \times N}$ be either the matrix $F^{(d)}$ or $W^{(d)}$, corresponding to the $d-$dimensional discrete Fourier or Walsh transform (see §10.1). In the Fourier case, we divide the different frequencies $\{-K/2 + 1, \dots, K/2\}^d$ into dyadic bands. For $d = 1$, we let $B_1 = \{0, 1\}$ and $B_k = \{-2^{k-1}+1, \dots, -2^{k-2}\} \cup \{2^{k-2}+1, \dots, 2^{k-1}\}$ for $k = 2, \dots, r$. In the Walsh case, we define the frequency bands $B_1 = \{0, 1\}$ and $B_k = \{2^{k-1}, \dots, 2^k -$

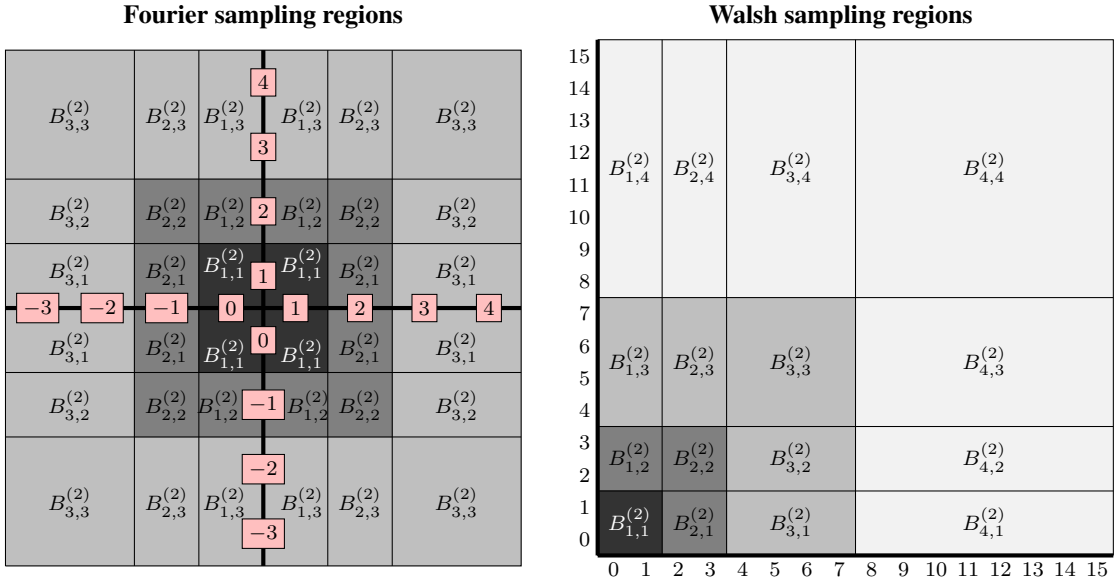

FIGURE 6. The different sampling regions used for the sampling patterns for Fourier (left, $r = 3$) and Walsh (right, $r = 4$). The axis labels correspond to the frequencies in each band and the annular regions are shown as the shaded greyscale regions.

$1\}$ for $k = 2, \ldots, r$ in the one-dimensional case. In the general d-dimensional case for Fourier or Walsh sampling, we set

$$B_{\mathbf{k}}^{(d)} = B_{k_1} \times \ldots \times B_{k_d}, \quad \mathbf{k} = (k_1, \ldots, k_d) \in \mathbb{N}^d.$$

For a d-dimensional tensor $c \in \mathbb{C}^{K \times \cdots \times K}$, we assume we can observe subsampled measurements of $V\mathrm{vec}(c)$, where $\mathrm{vec}(c) \in \mathbb{C}^N$ is a vectorised version of c. To recover a sparse representation, we consider the Haar wavelet coefficients. We denote the discrete Haar Wavelet transform by $\Phi \in \mathbb{C}^{N \times N}$, and note that $\Psi^* = \Psi^{-1}$ since Ψ is unitary. In other words, we consider a multilevel subsampled unitary matrix (Definition 5.9), with $U = V\Psi^*$. Given $\{m_{\mathbf{k}=(k_1,\ldots,k_d)}\}_{k_1,\ldots,k_d=1}^{r}$, we use a multilevel random sampling such that $m_{\mathbf{k}}$ measurements are chosen from $B_{\mathbf{k}}^{(d)}$ according to Definition 5.8. This corresponds to $l = r^d$ and the N_i's can be chosen given a suitable ordering of the Fourier/Walsh basis. The sparsity in levels structure (Definition 3.1) is chosen to correspond to the r wavelet levels. A pictorial representation is given in Figure 6. Finally, we define

$$\mathcal{M}_{\mathcal{F}}(\mathbf{s}, \mathbf{k}) \coloneqq \sum_{j=1}^{\|\mathbf{k}\|_{l^\infty}} s_j \prod_{i=1}^{d} 2^{-|k_i - j|} + \sum_{j=\|\mathbf{k}\|_{l^\infty}+1}^{r} s_j 2^{-2(j-\|\mathbf{k}\|_{l^\infty})} \prod_{i=1}^{d} 2^{-|k_i - j|} \tag{5.7}$$

$$\mathcal{M}_{\mathcal{W}}(\mathbf{s}, \mathbf{k}) \coloneqq s_{\|\mathbf{k}\|_{l^\infty}} \prod_{i=1}^{d} 2^{-|k_i - \|\mathbf{k}\|_{l^\infty}|}. \tag{5.8}$$

For notational convenience, we also define $\mathcal{Z} = \max\left\{1, \frac{\max_{j=1,\ldots,r} w_{(j)}\sqrt{(M_j - M_{j-1})}}{\sqrt{\xi(\mathbf{s},\mathbf{M},w)}}\right\}$.

We now state the main theorem of this section (proven in §10), which states how many samples are needed and the number of layers of the NN needed, which only depends logarithmically on the error δ, a consequence of the exponential convergence in Theorem 5.5. We discuss the sampling conditions below.

Theorem 5.10. *Consider the above setup of recovering a d-dimensional tensor $c \in \mathbb{C}^{K^d}$ ($N = K^d$) from subsampled Fourier or Walsh measurements Vc, such that A is a multilevel subsampled unitary matrix with respect to $U = V\Psi^*$. Let $\epsilon_{\mathbb{P}} \in (0, 1)$ and $\mathcal{L} = d \cdot r^2 \cdot \log(2m) \cdot \log^2(s \cdot \kappa(\mathbf{s}, \mathbf{M}, w)) + \log(\epsilon_{\mathbb{P}}^{-1})$. Suppose that:*

- *(a) In the Fourier case*
$$m_{\mathbf{k}} \gtrsim \kappa(\mathbf{s}, \mathbf{M}, w) \cdot \mathcal{M}_{\mathcal{F}}(\mathbf{s}, \mathbf{k}) \cdot \mathcal{L}. \tag{5.9}$$
- *(b) In the Walsh case*
$$m_{\mathbf{k}} \gtrsim \kappa(\mathbf{s}, \mathbf{M}, w) \cdot \mathcal{M}_{\mathcal{W}}(\mathbf{s}, \mathbf{k}) \cdot \mathcal{L}. \tag{5.10}$$

Then with probability at least $1 - \epsilon_{\mathbb{P}}$, A satisfies the weighted rNSPL of order $(\mathbf{s}, \mathbf{M})$ with constants $\rho = 1/2$ and $\gamma = \sqrt{2}$. The conclusion of Theorem 5.5 then holds for the uniform recovery of the Haar wavelet coefficients
$$x = \Psi c \in \mathbb{C}^N. \tag{5.11}$$
Moreover, for any $\delta \in (0,1)$, let $\mathcal{J}(\delta, \mathbf{s}, \mathbf{M}, w)$ be the collection of all $y \in \mathbb{C}^m$ with $y = P_{\mathcal{I}} DV c + e$ where
$$\|c\|_{l^2} \leq 1, \quad \max\left\{ \frac{\sigma_{\mathbf{s},\mathbf{M}}(\Psi c)_{l^1_w}}{\sqrt{\xi}}, \|e\|_{l^2} \right\} \leq \delta. \tag{5.12}$$
Then we construct via an algorithm, a neural network $\phi \in \mathcal{N}_{\mathbf{D},3n+1,3}$ such that with probability at least $1-\epsilon_{\mathbb{P}}$,
$$\|\phi(y) - c\|_{l^2} \lesssim \kappa^{1/4}\delta, \quad \forall y = P_{\mathcal{I}} DV c + e \in \mathcal{J}(\delta, \mathbf{s}, \mathbf{M}, w). \tag{5.13}$$
The network parameters are
$$\mathbf{D} = (m, \underbrace{2N+m, 2(N+m), 2N+m+1}_{n \text{ times}}, N), \quad n \leq \left\lceil \log\left(\delta^{-1}\mathcal{Z}\right) \kappa^{1/4} \mathcal{Z} \right\rceil. \tag{5.14}$$

The sampling conditions (5.9) and (5.10) are optimised by minimising $\kappa(\mathbf{s}, \mathbf{M}, w)$. Up to a constant scale, this corresponds to the choice $w_{(j)} = \sqrt{s/s_j}$ and
$$n = \left\lceil \log\left(\delta^{-1} \max_{j=1,\dots,r} \sqrt{\max\left\{1, \frac{M_j - M_{j-1}}{r s_j}\right\}} \right) r^{1/4} \max_{j=1,\dots,r} \sqrt{\max\left\{1, \frac{M_j - M_{j-1}}{r s_j}\right\}} \right\rceil.$$
Up to log-factors, the measurement condition then becomes equivalent to the currently best-known oracle estimator (where one assumes apriori knowledge of the support of the vector) [2, Prop. 3.1]. For Fourier measurements, we can interpret the condition as follows. For $d = 1$, this estimate yields the sampling estimates
$$m_k \gtrsim \left(\sum_{j=1}^{k} s_j 2^{-|k-j|} + \sum_{j=k+1}^{r} s_j 2^{-3|k-j|} \right) \cdot r \cdot \mathcal{L}.$$
In other words, up to logarithmic factors and exponentially small terms, s_j measurements are needed in each level. Furthermore, if $s_1 = \ldots = s_r = s_*$ and $d = 2$ then (5.9) holds if
$$m_{(k_1,k_2)} \gtrsim s_* 2^{-|k_1 - k_2|} \cdot r \cdot \mathcal{L}. \tag{5.15}$$
Another interpretation is gained by considering
$$m_k = \sum_{\|\mathbf{k}\|_{l^\infty} = k} m_{\mathbf{k}}, \quad k = 1, \dots, r,$$
the number of samples per annular region. We then have
$$m_k \gtrsim 3^d d \left(s_k + \sum_{l=1}^{k-1} s_l 2^{-(k-l)} + \sum_{l=k+1}^{r} s_l 2^{-3(l-k)} \right) \cdot r \cdot \mathcal{L}, \tag{5.16}$$
which is the same estimate as the one-dimensional case for bounded d. Note that the number of samples required in each annular region is (up logarithmic factors) proportional to the corresponding sparsity s_k with additional exponentially decaying terms dependent on $s_l, l \neq k$. In the case of Walsh sampling, (5.15) remains the same whereas (5.16) becomes $m_k \gtrsim 2^d \cdot d \cdot r \cdot \mathcal{L} \cdot s_k$, with no terms from the sparsity levels $s_l, l \neq k$.

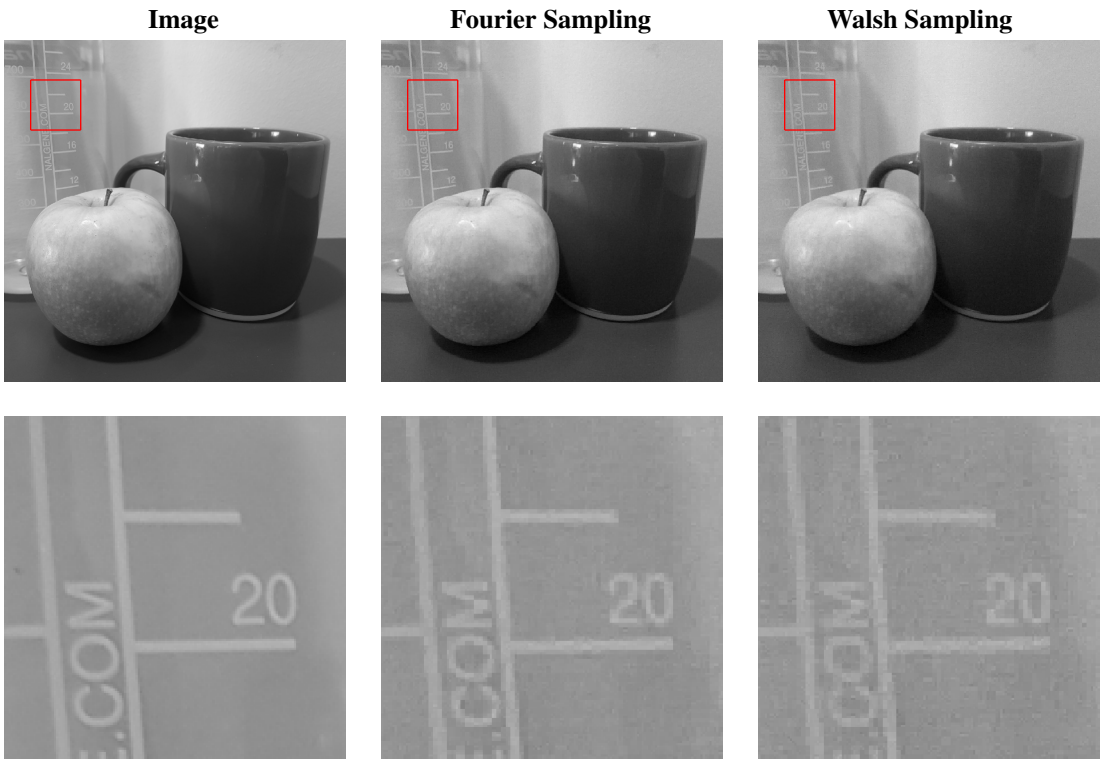

FIGURE 7. Left: The true image. Middle: Reconstruction from noisy Fourier measurements. Right: Reconstruction from noisy Walsh measurements. Both images were reconstructed using only a 15% sampling rate according to the sampling patterns in Figure 6 and $n = p = 5$. The top row shows the full image and the bottom row shows a zoomed in section (corresponding to the red boxes in the top row).

6. FIRENET: EXAMPLE OF THE EXPONENTIAL CONVERGENCE AND PSEUDOCODE

We now provide a computational experiment to demonstrate Theorem 5.10 (and Theorem 5.5). Note that the matrix A and its adjoint can be implemented rapidly using the fast Fourier transform (or fast Walsh–Hadamard transform). We take the image shown in Figure 7, a subsampling rate of only 15%, and corrupt the measurements by adding 2% Gaussian noise. Figure 7 shows the reconstructions using Fourier and Walsh sampling and Haar wavelets. Similar results hold for other wavelets, such as Daubechies wavelets with a larger number of vanishing moments. In fact, the reconstruction results are better than those shown for the Haar wavelet system. We have chosen to show the Haar wavelet results because this is the system for which Theorem 5.10 is stated. Pseudocode for the reconstruction is shown in Algorithm 1. For the reconstruction, we take $\lambda = 0.00025, \tau = \sigma = 1$, $p = 5$ and the weights as discussed in §5.3. In the spirit of no parameter tuning, the weights were selected based on a standard phantom image, and not the image we use to test the algorithm. These parameters are certainly not optimal, and instead were chosen simply to emphasise that we have deliberately avoided parameter tuning. Moreover, we found that the choice of δ in the algorithm was of little consequence, so have taken $\delta = 10^{-9}$.

Figure 8 shows the convergence in the number of inner iterations (or, equivalently, n - the total number of inner iterations is np, and hence we have not specified n, which is typically chosen to be 5). We show the error between the constructed image after j iterations (denoted by c_j) and the true image (denoted by c), as well as the convergence of the objective function which we denote by F in the figure caption. To compute the minimum of F, denoted F^*, we ran several thousand iterates of the non-restarted version of the algorithm so that the error in the value of F^* is at least an order of magnitude smaller than the shown values of $F(c_j) - F^*$. Whilst the objective function is guaranteed to converge to the minimum value when computing F^* this way, there is no guarantee that the vectors computed by the non-restarted version converge

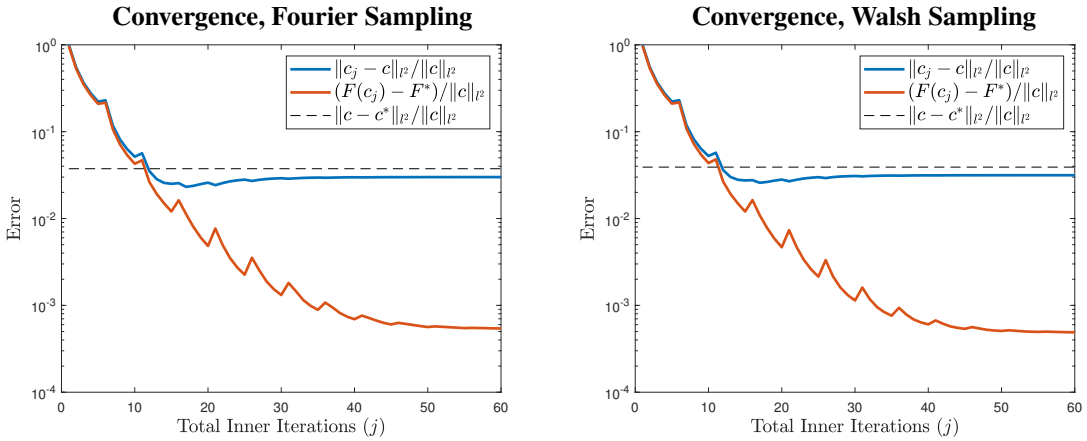

FIGURE 8. The convergence of the algorithm in the number of inner iterations. The dashed line shows $\|c - c^*\|_{l^2}/\|c\|_{l^2}$. In both cases, the error between the reconstruction and the image decreases exponentially until this bound is reached. The objective function gap decreases exponentially slightly beyond this point, demonstrating that the robust null space property (in levels) controls the l^2-norm difference between vectors (locally around c^*) down to the error $\|c - c^*\|_{l^2}$ (see the bound (9.18) in our proof).

Algorithm 1: `FIRENETcomp` constructs a FIRENET which corresponds to n iterations of `InnerIt` with a rescaling scheme. We write the output as the map ϕ_n to emphasise that `FIRENETcomp` defines a NN. `InnerIt` performs p iterations of Chambolle and Pock's primal-dual algorithm for square-root LASSO (the order of updates is swapped compared to [37]). The functions φ_s and ψ^1 are proximal maps:

$$[\varphi_s(x)]_j = \max\left\{0, 1 - \frac{s}{|x_j|}\right\} x_j, \quad \psi^1(y) = \min\left\{1, \frac{1}{\|y\|_{l^2}}\right\} y.$$

Both of these are approximated by NNs in our proof.

Function `FIRENETcomp` $(A, p, \tau, \sigma, \lambda, \{w_j\}_{j=1}^N, \epsilon_0, \delta, n)$

Initiate with $\phi_0 \equiv 0$ (other initial vectors can also be chosen).
(NB: ϵ_0 should be of the same order as $\|y\|_{l^2}$ for inputs $y \in \mathbb{C}^m$.)
for $k = 1, ..., n$ **do**
$\epsilon_k = e^{-1}(\delta + \epsilon_{k-1})$,
$\beta_k = \frac{\epsilon_k}{2\|A\|}$
$\phi_k(\cdot) = p\beta_k \cdot$ `InnerIt` $\left(\frac{\cdot}{p\beta_k}, \frac{\phi_{k-1}(\cdot)}{p\beta_k}, A, p, \sigma, \tau, \lambda, \{w_j\}_{j=1}^N\right)$
end
return: FIRENET $\phi_n : \mathbb{C}^m \to \mathbb{C}^N$
end

Function `InnerIt` $(y, x_0, A, p, \tau, \sigma, \lambda, \{w_j\}_{j=1}^N)$

Set $B = \mathrm{diag}(w_1, ..., w_N) \in \mathbb{C}^{N \times N}$.
Initiate with $x^0 = x_0$, $y^0 = 0 \in \mathbb{C}^m$ (the superscripts denote indices not powers).
for $k = 0, ..., p-1$ **do**
$x^{k+1} = B\varphi_{\tau\lambda}(B^{-1}(x^k - \tau A^* y^k))$
$y^{k+1} = \psi^1(y^k + \sigma A(2x^{k+1} - x^k) - \sigma y)$
end
$X = \sum_{k=1}^p \frac{x^k}{p}$
return: $X \in \mathbb{C}^N$ (ergodic average of p iterates)
end

to a minimiser, as demonstrated by the non-computability results in Theorem 2.2. However, in this case, the non-restarted version converged to a vector c^* up to an error much smaller than $\|c - c^*\|_{l^2}$. Hence $\|c - c^*\|_{l^2}$ indicates the minimum error we can expect from using (P_3) to recover the image.

The figure shows the expected exponential convergence, as the number of inner iterations increases, of the objective function values as well as c_j to c until the error is of the order $\|c - c^*\|_{l^2}$. This corresponds to an initial phase of exponential convergence, where the υ^{-n} term (with $\upsilon = e^{-1}$) is dominant in Theorem 5.5, followed by a plateau to the minimal error $\|c - c^*\|_{l^2}$ (shown as the dotted line). This plateau occurs due to inexact measurements (the noise) and the fact that the image does not have exactly sparse wavelet coefficients. This corresponds to the robust null space property (in levels) only being able to bound the distance $\|c - c_j\|_{l^2}$ up to the same order as $\|c - c^*\|_{l^2}$. In other words, we can only accelerate convergence up to this error bound. The error plateau disappears in the limit of exactly sparse vectors and zero noise (in the limit $\delta \downarrow 0$ in Theorem 5.5), and one gains exponential convergence down to essentially machine precision. Finally, the acceleration is of great practical interest. Rather than the several hundreds (or even thousands) of iterations that are typically needed for solving compressed sensing optimisation problems with first-order iterative methods, we obtain optimal accuracy in under 20 iterations. This was found for a range of different images, subsampling rates etc. The fact that so few layers are needed, coupled with the fast transforms for implementing the affine maps in the NNs, makes the NNs very computationally efficient and competitive speed-wise with state-of-the-art DL.

7. Connections with previous work

The paper touches many different areas of mathematics and AI; thus we have divided the areas into the following subsections: (i) Computational barriers in DL, foundations and the SCI hierarchy; (ii) Instabilities in DL; (iii) DL in inverse problems; and (iv) Compressed sensing, optimisation and unrolling of algorithms.

(i) ***Computational barriers in DL, foundations and the SCI hierarchy***: The SCI hierarchy and its foundations framework provide the basis for the techniques for the computational barriers and foundations (what is and what is not computationally possible) results in DL proved in this paper. The SCI hierarchy has recently been used to solve longstanding questions on the existence of algorithms [15,18–20,40,41,65]. It was introduced in [65] and is an effective tool to establish the boundaries of what computers can achieve in scientific computing and also in computer-assisted proofs, see for example the work by C. Fefferman & L. Seco (Dirac-Schwinger conjecture) [50, 51] and T. Hales et. al (Kepler's conjecture/Hilbert's 18th problem) [61, 62] that implicitly prove results in the SCI hierarchy (see [18] for details). The SCI hierarchy generalises S. Smale's seminal work [100, 102] with L. Blum, F. Cucker, M. Shub [24, 25] and his program on the foundations of scientific computing and existence of algorithms pioneered by C. McMullen [86,87] and P. Doyle & C. McMullen [48]. In particular, the work by F. Cucker [42] can be viewed as an early version of the SCI hierarchy.

In [18] J. Ben-Artzi, O. Nevanlinna, M. Seidel and two of the authors established a collection of techniques for proving sharp results on the existence of algorithms that form a basis for techniques in the SCI hierarchy framework. These ideas were extended to randomised algorithms in [15]. The SCI hierarchy has also been used in signal processing and sampling theory by H. Boche & V. Pohl in [26]. Recent results using the SCI hierarchy include the work by S. Olver & M. Webb [111] on computing spectral measures and the work by J. Ben-Artzi, M. Marletta & F. Rösler [19, 20] on computing resonances. There is a vast literature on impossibility results regarding the existence of algorithms for different problems in mathematics. The seminal work of S. Weinberger [112] is a great example. These results can also be interpreted as classification results in the SCI hierarchy, as the framework is flexible and can encompass any model of computation.

Our results connect with the vast literature in approximation theory starting with the universal approximation theorem and many follow-up papers establishing the great approximation qualities of NNs, see, for example, the work by H. Bölcskei, P. Grohs, G. Kutyniok & P. Petersen [27] and the results by I. Daubechies, R. DeVore, S. Foucart, B. Hanin & G. Petrova [44] (the recent article [45] of R. DeVore, B. Hanin & G. Petrova surveys current challenges of NN approximation). What all these results have in common is that

there is no given algorithm constructing the approximating NNs that are proven to exist. The literature is full of existence results that will typically not imply the existence of algorithms computing the NNs.

(ii) ***Instabilities in DL***: Initiated by the work of C. Szegedy et. al. [105], there is now a vast literature on the instability phenomenon in DL in a wide variety of applications [7, 9, 36, 52, 70, 90, 105] ranging from image recognition and classification, via audio and speech recognition to automatic diagnosis in medicine, image reconstruction and inverse problems. Thus we can only highlight a small subset here. A significant development was the *DeepFool* research programme and software package of S. Moosavi–Dezfooli, A. Fawzi & P. Frossard [90], which was followed by the construction of so-called *universal adversarial perturbations* [89]. The construction of and mitigation against *adversarial attacks* is now an active area of research. To the best of our knowledge, [9] and [70] were the first works to demonstrate the instability phenomenon for inverse problems in imaging. There is also an increasing amount of work dedicated to the important problem of numerical instability in ML, see, for example, the recent work of P. Blanchard, D. Higham & N. Higham [23].

(iii) ***DL in inverse problems***: The work of K. Jin, M. McCann, E. Froustey & M. Unser [71] was influential in highlighting the promise of DL for inverse problems in imaging. This is now a rapidly evolving area of research, which we will not attempt to summarise. See [12] for an overview of current techniques. Note that sparse regularisation has been used as the basis for some DL technique, e.g. by using DL to recover the parts of an image that sparse regularisation cannot such as in [31], or by designing NN architectures through the process of *unrolling* an optimisation algorithm (see, e.g., [12]) discussed below. Another approach is to learn variational regularisers, see, for example, [75].

(iv) ***Compressed sensing, optimisation and unrolling of algorithms***: Our positive results rely on theory from the compressed sensing literature initiated by E. Candes, J. Romberg & T. Tao [34] and D. Donoho [47]. In particular, our results rely on the many results on structured sampling in structured compressed sensing, see the work by B. Adcock et. al. [2, 5, 6], A. Bastounis et. al. [14], J. Bigot, C. Boyer & P. Weiss [22, 30] and G. Kutyniok & W. Lim [77]. Moreover, our results are closely related to the work of A. Ben-Tal & A. Nemirovski [21], who were one of the first to realise how key assumptions in compressed sensing – such as the *robust nullspace property* – help bound the error of the approximation to a minimiser (produced by an optimisation algorithm) in terms of error bounds on the approximation to the objective function. Our construction of stable and accurate NNs uses the optimisation problem (P_3) and approximations of unrolled (or unfolded) primal-dual iterations with a restart scheme: see the discussion at the start of §9. Unrolling iterative methods as NNs was first developed by Gregor & LeCun in [60] who considered LISTA, a learned version of ISTA, for the recovery of sparse vectors. This work has been followed by theoretical grantees [39, 79], ensuring the existence of NN with linear convergence towards the minimiser. Yet, neither [39] nor [79], use the theoretically correct weights, as these can only be computed as solutions of intractably large optimisation problems. Moreover, it is not clear whether the needed assumptions on A ever hold in practice. Many other algorithms have been unrolled as NNs. Typically, such approaches train the weights and biases, and even the activation functions of the NNs (see also the comments after Remark 5.7). We shall not attempt a broad survey, and instead point the reader to the papers of M. McCann, K. Jin & M. Unser [85] and V. Monga, Y. Li & Y. Eldar [88] for up to date reviews.

To prove convergence of vectors (as opposed to objective functions), it is crucial to have conditions such as the robust null space property (in levels) of Definition 3.2. Moreover, this property enables a restart scheme to achieve exponential convergence in the number of layers. Typically, exponential convergence for restart schemes requires a Łojasiewicz-type inequality of the form $\gamma d(x, X^*)^\nu \leq f(x) - f^*$, where f is the objective function and $d(x, X^*)$ denotes the distance to the set of minimisers [28, 73]. For example, V. Roulet, N. Boumal & A. d'Aspremont [98] achieve exponential convergence, using the restarted NESTA algorithm [16], for exact recovery (noiseless) of sparse vectors if A satisfies the null space property. By taking into account computability and construction of NNs, and allowing noise, approximate sparsity, sparsity in levels and complex-valued vectors and matrices, our techniques are more general than [98].

8. Proof of Theorem 2.2 and tools from the SCI hierarchy

In this section, we prove Theorem 2.2. To do this, we need some analytical results regarding phase transitions of solutions of $(P_1), (P_2)$ and (P_3) which are given in §8.2. However, before analysing these phase transitions, we need some preliminary definitions regarding algorithms, inexact inputs, and condition numbers. There are two main reasons for this framework. First, because our definitions are general, they lead to stronger impossibility results than when restricted to specific models of computation. Second, our framework greatly simplifies the proofs and makes it clear what the key mechanisms behind the proofs are (§8.2 describes this in terms of phase transitions of minimisers). The following discussions are self-contained.

8.1. Algorithmic preliminaries: a user-friendly guide.

We begin with a definition of a computational problem, which is deliberately general in order to capture any computational problem.

Definition 8.1 (Computational problem)**.** *Let Ω be some set, which we call the domain, and 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. When it is clear what $\mathcal{M}$ and Λ are, we write $\{\Xi, \Omega\}$ for brevity.*

Remark 8.2 (Multivalued problems)**.** In some cases, such as when considering the optimisation problems (P_j) that may have more than one solution, we consider $\Xi(\iota) \subset \mathcal{M}$. With an abuse of notation, we then set $d(x, \Xi(\iota)) = \operatorname{dist}(x, \Xi(\iota)) = \inf_{y \in \Xi(\iota)} d(x, y)$ and this distinction will be made clear from context. ⊠

The set Ω is the set of objects that give rise to our computational problems. The problem function $\Xi : \Omega \to \mathcal{M}$ is what we are interested in computing. Finally, the set Λ is the collection of functions that provide us with the information we are allowed to read as input to an algorithm. For example, Ω could consist of a collection of matrices A and data y in (2.1), Λ could consist of the pointwise entries of the vectors and matrices in Ω, Ξ could represent the solution set (with the possibility of more than one solution as in Remark 8.2) of any of the problems (P_j) and $(\mathcal{M}, d)$ could be $\mathbb{C}^N$ with the usual Euclidean metric (or any other suitable metric).

Given the definition of a computational problem, we need the definition of a general algorithm, whose conditions hold for any reasonable notion of a deterministic algorithm. Throughout this paper, we deal with the case that $\Lambda = \{f_j\}_{j \in \beta}$, where β is some (at most) countable index set.

Definition 8.3 (General Algorithm)**.** *Given a computational problem $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$, a general algorithm is a mapping $\Gamma : \Omega \to \mathcal{M}$ such that for each $\iota \in \Omega$*

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

If, in addition, there exists a canonical ordering $\Lambda_\Gamma(\iota) = \{f^\Gamma_{\iota,1} = f_{k_1}, ..., f^\Gamma_{\iota,S_\Gamma(\iota)} = f_{k_{S_\Gamma(\iota)}}\}$, where $S_\Gamma(\iota) = |\Lambda_\Gamma(\iota)|$, such that if $\kappa \in \Omega$ and $f^\Gamma_{\iota,j}(\iota) = f^\Gamma_{\iota,j}(\kappa)$ for all $j \leq r < S_\Gamma(\iota)$, then $f^\Gamma_{\iota,j} = f^\Gamma_{\kappa,j}$ for all $j \leq r+1$, then we call Γ a Sequential General Algorithm. In this case, we use the notation $k_j(\Gamma, \iota)$ to denote the ordered indices corresponding to the evaluation functions that the algorithm reads.

The three properties of a general algorithm are the most basic natural properties we would expect any deterministic computational device to obey. The first condition says that the algorithm can only take a finite amount of information, though it is allowed adaptively to choose, depending on the input, the finite amount of information that it reads. The second condition ensures that the algorithm's output only depends on its input, or rather the information that it has accessed (or "read"). The final condition is very important and ensures that the algorithm produces outputs and accesses information consistently. In other words, if it sees the same information for two different inputs, then it cannot behave differently for those inputs. Note that the definition of a general algorithm is more general than the definition of a Turing machine [106] or a Blum–Shub–Smale (BSS) machine [24], which can be thought of as digital and analog computational devices respectively. In particular, a general algorithm has no restrictions on the operations allowed. The extra

condition for a sequential general algorithm is satisfied by any algorithm defined by a computational machine with input of readable information (one should think of the ordered indices of the evaluation functions as corresponding to sequentially reading the tape which encodes the input information). Hence, a sequential general algorithm is still more general than a Turing or a BSS machine. Complete generality in Definition 8.3 is used for two primary reasons:

(i) *Strongest possible bounds:* Since Definition 8.3 is completely general, the lower bounds hold in any model of computation, such as a Turing machine or a BSS machine. On the other hand, the algorithms we construct in this paper are made to work using only arithmetic operations over the rationals. Hence, we obtain the strongest possible lower bounds and the strongest possible upper bounds.

(ii) *Simplified exposition:* Using the concept of a general algorithm considerably simplifies the proofs of lower bounds and allows us to see precisely the mechanisms behind the proofs.

Next, we consider the definition of a randomised general algorithm, which again is more general than a probabilistic Turing or probabilistic BSS machine. Randomised algorithms are widely used in practice in areas such as optimisation, algebraic computation, machine learning, and network routing. In the case of Turing machines, it is currently unknown, in the sense of polynomial runtime, whether randomisation is beneficial from a complexity class viewpoint [11, Ch. 7], however, rather intriguingly this is not the case for BSS machines [24, Ch. 17] (some of the proofs in this reference are non-constructive - it is an open problem whether any probabilistic BSS machine can be simulated by a deterministic machine having the same machine constants and with only a polynomial slowdown). Nevertheless, randomisation is an extremely useful tool in practice. From a machine learning point of view, we also want to consider randomised algorithms to capture procedures such as stochastic gradient descent which are commonly used to train NNs. As developed in [15], the concept of a general algorithm can be extended to a randomised general algorithm. This concept allows for universal impossibility results regardless of the computational model.

Definition 8.4 (Randomised General Algorithm)**.** *Given a computational problem $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$, a Randomised General Algorithm (RGA) Γ^{ran} is a collection X of general algorithms $\Gamma : \Omega \to \mathcal{M}$, 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:*

(1) For each $\iota \in \Omega$, the mapping $\Gamma^{\mathrm{ran}}_\iota : (X, \mathcal{F}) \to (\mathcal{M}, \mathcal{B})$ defined by $\Gamma^{\mathrm{ran}}_\iota(\Gamma) = \Gamma(\iota)$ is a random variable, where $\mathcal{B}$ is the Borel sigma-algebra on $\mathcal{M}$.

(2) For each $n \in \mathbb{N}$ and $\iota \in \Omega$, the set $\{\Gamma \in X : \sup\{m \in \mathbb{N} : f_m \in \Lambda_\Gamma(\iota)\} \leq n\} \in \mathcal{F}$.

(3) For each $\iota_1, \iota_2 \in \Omega$ and $E \in \mathcal{F}$, such that for every $\Gamma \in E$ we have $f(\iota_1) = f(\iota_2)$ for every $f \in \Lambda_\Gamma(\iota_1)$, then $\mathbb{P}_{\iota_1}(E) = \mathbb{P}_{\iota_2}(E)$.

With slight abuse of notation, we denote the family of randomised general algorithms by RGA.

The first two conditions are measure theoretic to avoid pathological cases and ensure that "natural sets" one might define for a random algorithm (such as notions of stopping times) are measurable. These conditions hold for all standard probabilistic machines (such as a Turing or BSS machine). The third condition ensures consistency, namely, that in the case of identical evaluations, the laws of the output cannot change. Finally, we will use the standard definition of a probabilistic Turing machine (which is a particular case of Definition 8.4). However, to make sense of probabilistic Turing machines in our context (in particular, to restrict operations to the rationals which can be encoded by the natural numbers), we must define the notion of inexact input.

Suppose we are given a computational problem $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$, and that $\Lambda = \{f_j\}_{j\in\beta}$, where we remind the reader that β is some index set that can be finite or countably infinite. However, obtaining f_j may be a computational task on its own, which is exactly the problem in most areas of computational mathematics. In particular, for $\iota \in \Omega$, $f_j(\iota)$ could be the number $e^{\frac{\pi}{j}i}$ for example. Hence, we cannot access or store $f_j(\iota)$ on a computer, but rather $f_{j,n}(\iota)$ where $f_{j,n}(\iota) \to f_j(\iota)$ as $n \to \infty$. This idea is formalised in the definition below, however, to put this in perspective it is worth mentioning the Solvability Complexity Index (SCI) hierarchy.

Remark 8.5 (The Solvability Complexity Index (SCI) hierarchy)**.** The SCI of a computational problem is the smallest number of limits needed in order to compute the solution. The full hierarchy is described in [18], and the mainstay of the hierarchy are the Δ_k^α classes. The α denotes the model of computation. Informally, we have the following description. Given a collection $\mathcal{C}$ of computational problems, then

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

The above hierarchy gives rise to the concept of 'Δ_1-information.' That is, in informal terms, the problem of obtaining the inexact input to the computational problem is a Δ_1 problem. One may think of an algorithm taking the number $\exp(1)$ or $\sqrt{2}$ as input. Indeed, one can never produce an exact version of these numbers to the algorithm, however, one can produce an approximation to an arbitrarily small error.

Definition 8.6 (Δ_1-information)**.** *Let $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$ be a computational problem. We say that Λ has Δ_1-information if each $f_j \in \Lambda$ is not available, however, 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 $\iota \in \Omega$. Finally, if $\widehat{\Lambda}$ is a collection of such functions described above such that Λ has Δ_1-information, we say that $\widehat{\Lambda}$ provides Δ_1-information for Λ. Moreover, we denote the family of all such $\widehat{\Lambda}$ by $\mathcal{L}^1(\Lambda)$.*

We want to have algorithms that can handle all computational problems $\{\Xi, \Omega, \mathcal{M}, \widehat{\Lambda}\}$ whenever $\widehat{\Lambda} \in \mathcal{L}^1(\Lambda)$. In order to formalise this, we define what we mean by a computational problem with Δ_1-information.

Definition 8.7 (Computational problem with Δ_1-information)**.** *A computational problem where Λ has Δ_1-information is denoted by $\{\Xi, \Omega, \mathcal{M}, \Lambda\}^{\Delta_1} \coloneqq \{\widetilde{\Xi}, \widetilde{\Omega}, \mathcal{M}, \widetilde{\Lambda}\}$, where*

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

Moreover, if $\widetilde{\iota} = \{f_{j,n}(\iota)\}_{j,n\in\beta\times\mathbb{N}} \in \widetilde{\Omega}$ then we define $\widetilde{\Xi}(\widetilde{\iota}) = \Xi(\iota)$ and $\widetilde{f}_{j,n}(\widetilde{\iota}) = f_{j,n}(\iota)$. We also set $\widetilde{\Lambda} = \{\widetilde{f}_{j,n}\}_{j,n\in\beta\times\mathbb{N}}$. Note that $\widetilde{\Xi}$ is well-defined by Definition 8.1 of a computational problem and the definition of $\widetilde{\Omega}$ includes all possible instances of Δ_1-information $\widehat{\Lambda} \in \mathcal{L}^1(\Lambda)$.

We can now define a probabilistic Turing machine for $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$, where the algorithm Γ is executed by a Turing machine [106], that has an oracle tape consisting of $\{\widetilde{\iota}_f\}_{f\in\widetilde{\Lambda}}$. In what follows, we have deliberately not written down the (lengthy) definition of a Turing machine (found in any standard text [11]), which one should think of as an effective algorithm or computer programme (the famous Church–Turing thesis).

Definition 8.8. *Given the definition of a Turing machine, a probabilistic Turing machine for $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$ is a Turing machine that has an oracle tape consisting of $\{\widetilde{\iota}_f\}_{f\in\widetilde{\Lambda}}$ (for $\widetilde{\iota} \in \widetilde{\Omega}$), with an additional read-only tape containing independent binary random numbers (0 or 1 with equal probability), and which halts with probability one and outputs a single element of $\mathcal{M}$. The law of such a machine will be denoted by $\mathbb{P}$. With an abuse of notation, we sometimes denote the probabilistic Turing machine by $(\Gamma, \mathbb{P})$.*

Remark 8.9 (Where does the output live?)**.** Strictly speaking, when we say that the output of a probabilistic Turing machine lies in $\mathcal{M}$, we mean that the output corresponds, via an encoding, to an element of a subset of $\mathcal{M}$ such as $(\mathbb{Q} + i\mathbb{Q})^N \subset \mathbb{C}^N$. However, we follow the usual convention of suppressing such encodings.⊠

One should think of Definition 8.8 as an algorithm for the computational problem with inexact input, but with the additional ability to generate random numbers (corresponding to the binary input tape) and execute commands based on the sequence of random numbers that are generated. The reader should intuitively think of this as a computer program with a random number generator. For equivalent definitions and the

basic properties of such machines, see [11]. For simplicity, we have only considered probabilistic Turing machines that halt with probability one, though extensions can be made to non-halting machines. Note that Definition 8.8 is a special case of Definition 8.4, where $\mathbb{P}_\iota = \mathbb{P}$ is fixed across different ι. In particular, given a probabilistic Turing machine, the sigma-algebra and probability distribution generated by the standard product topology on $\{0,1\}^{\mathbb{N}}$ induce the relevant collection X of Turing machines and sigma-algebra $\mathcal{F}$, as well as $\mathbb{P}$.

Finally, we recall standard definitions of condition used in optimisation [24, 32]. The classical condition number of an invertible matrix A is given by $\mathrm{Cond}(A) = \|A\|\|A^{-1}\|$. For different types of condition numbers related to a possibly multivalued (signified by the double arrow) mapping $\Xi : \Omega \subset \mathbb{C}^n \rightrightarrows \mathbb{C}^N$ we need to establish what types of perturbations we are interested in. For example, if Ω denotes the set of diagonal matrices (which we treat as elements of $\mathbb{C}^n$ for some n), we may not be interested in perturbations in the off-diagonal elements as they will always be zero. In particular, we may only be interested in perturbations in the coordinates that are varying in the set Ω. Thus, given $\Omega \subset \mathbb{C}^n$ we define the active coordinates of Ω to be $\mathrm{Act} = \mathrm{Act}\,(\Omega) = \{j \,:\, \exists\, x, y \in \Omega, x_j \neq y_j\}$. Moreover, for $\nu > 0$ (including the obvious extension to $\nu = \infty$),

$$\Omega_\nu = \left\{x \,:\, \exists\, y \in \Omega \text{ such that } \|x-y\|_{l^\infty} \leq \nu, x_{\mathrm{Act}^c} = y_{\mathrm{Act}^c}\right\}.$$

In other words, Ω_ν is the set of ν-perturbations along the non-constant coordinates of elements in Ω. We can now recall some of the classical condition numbers from the literature [24, 32].

(1) *Condition of a mapping*: Let $\Xi : \Omega \subset \mathbb{C}^n \rightrightarrows \mathbb{C}^m$ be a linear or non-linear mapping, and suppose that Ξ is also defined on Ω_ν for some $\nu > 0$. Then,

$$\mathrm{Cond}\,(\Xi, \Omega) = \sup_{x\in\Omega} \lim_{\epsilon\to 0^+} \sup_{\substack{x+z\in\Omega_\nu \\ 0<\|z\|_{l^2}\leq\epsilon}} \left\{\frac{\mathrm{dist}(\Xi(x+z), \Xi(x))}{\|z\|_{l^2}}\right\},$$

where we allow for multivalued functions by defining $\mathrm{dist}(\Xi(x+z), \Xi(z)) = \inf_{w_1\in\Xi(x+z), w_2\in\Xi(z)} \|w_1 - w_2\|_{l^2}$ (see Remark 8.2). We will use this notion of condition number for (P_1), (P_2) and (P_3).

(2) *Distance to infeasibility - the Feasibility Primal condition number*: For the problem (P_1) of basis pursuit (for (P_2) and (P_3) the following condition number is always zero) we set

$$\nu(A,y) = \sup\left\{\epsilon \geq 0 \,:\, \|\hat{y}\|_{l^2}, \|\hat{A}\| \leq \epsilon, (A+\hat{A}, y+\hat{y}) \in \Omega_\infty \Rightarrow (A+\hat{A}, y+\hat{y}) \text{ are feasible inputs to } \Xi_{P_1}\right\},$$

and define the *Feasibility Primal* (FP) local condition number $C_{\mathrm{FP}}(A,y) \coloneqq \frac{\max\left\{\|y\|_{l^2}, \|A\|\right\}}{\nu(A,y)}$. We then define the FP global condition number via $C_{\mathrm{FP}}\left(\Xi_{P_1}, \Omega\right) \coloneqq \sup_{(A,y)\in\Omega} C_{\mathrm{FP}}(A,y)$.

8.2. **Phase transitions.** To prove Theorem 2.2, we use the following lemmas which describe phase transitions of the minimisers of the respective optimisation problems (e_j correspond to the canonical basis of $\mathbb{C}^N$).

Lemma 8.10 (Phase transition for basis pursuit). *Let $N \geq 2$ and consider the problem (P_1) for*

$$A = \begin{pmatrix} \frac{w_1}{\rho_1} & \frac{w_2}{\rho_2} & \cdots & \frac{w_N}{\rho_N} \end{pmatrix} \in \mathbb{C}^{1\times N}, \quad y = 1, \quad \epsilon \in [0,1),$$

where $\rho_j > 0$ for $j = 1, ..., N$. Then the set of solutions is given by

$$\sum_{j=1}^N \left[t_j(1-\epsilon)\frac{\rho_j}{w_j}\right] e_j, \quad \textit{s.t.} \quad t_j \in [0,1], \sum_{j=1}^N t_j = 1 \quad \textit{and} \quad t_j = 0 \quad \textit{if} \quad \rho_j > \min_k \rho_k. \tag{8.1}$$

Proof. Let $\hat{x}_j = x_j w_j \rho_j^{-1}$, then the optimisation problem becomes

$$\mathrm{argmin}_{\hat{x}\in\mathbb{C}^N} f(\hat{x}) \coloneqq \sum_{j=1}^N \rho_j |\hat{x}_j| \text{ such that } \left|1 - \sum_{j=1}^N \hat{x}_j\right| \leq \epsilon. \tag{8.2}$$

Since $\epsilon < 1$ and the $(\rho_1, \rho_2, ...\rho_N)$ weighted l^1 norm is convex, it follows that the solution must lie on the hypersurface segment $\hat{x}_1 + \hat{x}_2 + ... + \hat{x}_j = 1 - \epsilon$ for $\hat{x}_j \in \mathbb{R}_{\geq 0}$. We now claim that if $\hat{x}$ is a solution of

(8.2), and $\rho_j > \min_k \rho_k$, then $\hat{x}_j = 0$. Suppose for a contradiction that there exists a solution $\hat{x}$ of (8.2) where $\hat{x}_j > 0$ and $\rho_j > \min_k \rho_k$. Pick any l such that $\rho_l = \min_k \rho_k$, then $\hat{x} + \hat{x}_j(e_l - e_j)$ is feasible with $f(\hat{x} + \hat{x}_j(e_l - e_j)) < f(\hat{x})$, a contradiction. Similarly, if x is of the form (8.1), then $f(\hat{x}) = (1-\epsilon)\min_k \rho_k$. In particular, the objective function is constant over the set of all such vectors and the result follows. □

Lemma 8.11 (Phase transition for LASSO). *Let $N \geq 2$ and consider the problem (P_2) for*

$$A = \lambda \begin{pmatrix} \frac{w_1}{\rho_1} & \frac{w_2}{\rho_2} & \cdots & \frac{w_N}{\rho_N} \end{pmatrix} \in \mathbb{C}^{1\times N}, \quad y = 1,$$

where $0 < \rho_j < 2$ for $j = 1, ..., N$. Then the set of solutions is given by

$$\left(1 - \frac{\min_k \rho_k}{2}\right) \sum_{j=1}^{N} \frac{\rho_j t_j}{\lambda w_j} e_j, \quad \textit{s.t.} \quad t_j \in [0,1], \sum_{j=1}^{N} t_j = 1 \quad \textit{and} \quad t_j = 0 \quad \textit{if} \quad \rho_j > \min_k \rho_k. \tag{8.3}$$

Proof. Let $\hat{x}_j = x_j \lambda w_j \rho_j^{-1}$, then the optimisation problem becomes $\operatorname{argmin}_{\hat{x}\in\mathbb{C}^N} f(\hat{x}) \coloneqq |1 - \sum_{j=1}^N \hat{x}_j|^2 + \sum_{j=1}^N \rho_j |\hat{x}_j|$. It is clear that any optimal solution must be real and hence we restrict our argument to real $\hat{x}$. Define the 2^N quadrant subdomains $D_{k_1,...,k_N} = \{\hat{x}_j \cdot (-1)^{k_j} > 0\}$ for $k_j \in \{0,1\}$, and notice that

$$\nabla f(\hat{x}) = \begin{bmatrix} -2(1 - \sum_{j=1}^N \hat{x}_j) + (-1)^{k_1}\rho_1 \\ \vdots \\ -2(1 - \sum_{j=1}^N \hat{x}_j) + (-1)^{k_N}\rho_N \end{bmatrix}, \quad \text{for } \hat{x} \in D_{k_1,...,k_N}.$$

We first look for stationary points of the objective function in the subdomains $D_{k_1,...,k_N}$. The condition for a stationary point in the interior of such a domain leads to the constraint that $k_1 = k_2 = ... = k_N$. If $k_1 = k_2 = ... = k_N = 1$, then $\nabla f = 0$ leads to the contradiction $\rho_j = 2(\hat{x}_1 + ... + \hat{x}_N) - 2 < 0$. Finally, in the case (and only in the case) of $\rho_1 = \rho_2 = ... = \rho_N$, there is a hypersurface segment of stationary points in $D_{0,0,...,0}$ given by $\hat{x}_1 + ... + \hat{x}_N = 1 - \rho_1/2$ (recall that we assumed $\rho_1 < 2$ so this segment exists).

First, consider the case that $\rho_1 = ... = \rho_N$. Then any optimal solution must either lie on the boundary of some $D_{k_1,...,k_N}$ or on the hypersurface segment $\hat{x}_1 + ... + \hat{x}_N = 1 - \rho_1/2$ in $D_{0,0,...,0}$. A simple case by case analysis now yields that the solutions $\hat{x}$ are given by convex combinations of $(1 - \rho_j/2)e_j$ for $j = 1, ..., N$. Now consider the case that not all of the ρ_j are equal. Then any optimal solution must lie on the boundary of some $D_{k_1,...,k_N}$. A simple case by case analysis now yields that the solutions $\hat{x}$ are given by convex combinations of $(1 - \rho_j/2)e_j$ for j such that $\rho_j = \min_k \rho_k$. Rescaling back to x gives the result. □

Lemma 8.12 (Phase transition for square-root LASSO). *Let $N \geq 2$ and consider the problem (P_3) for*

$$A = \lambda \begin{pmatrix} \frac{w_1}{\rho_1} & \frac{w_2}{\rho_2} & \cdots & \frac{w_N}{\rho_N} \end{pmatrix} \in \mathbb{C}^{1\times N}, \quad y = 1,$$

where $0 < \rho_j < 1$ for $j = 1, ..., N$. Then the set of solutions is given by

$$\sum_{j=1}^{N} \frac{\rho_j t_j}{\lambda w_j} e_j, \quad \textit{s.t.} \quad t_j \in [0,1], \sum_{j=1}^{N} t_j = 1 \quad \textit{and} \quad t_j = 0 \quad \textit{if} \quad \rho_j > \min_k \rho_k. \tag{8.4}$$

Proof. Let $\hat{x}_j = x_j \lambda w_j \rho_j^{-1}$, then the optimisation problem becomes $\operatorname{argmin}_{\hat{x}\in\mathbb{C}^N} f(\hat{x}) \coloneqq |1 - \sum_{j=1}^N \hat{x}_j| + \sum_{j=1}^N \rho_j |\hat{x}_j|$. It is clear that any optimal solution must be real and hence we restrict our argument to real $\hat{x}$. The objective function is piecewise affine and since $\rho_j < 1$, the gradient of f is non-vanishing on the interior of any of the domains $D_{k_1,...,k_N} = \{\hat{x}_j \cdot (-1)^{k_j} > 0\}$ for $k_j \in \{0,1\}$. It follows that the optimal solutions must lie on the boundaries of the domains $D_{k_1,...,k_N}$. A simple case by case analysis shows that the solutions $\hat{x}$ are given by convex combinations of e_j for j such that $\rho_j = \min_k \rho_k$. Rescaling back to x gives the result. □

We will also need the following propositions, which give useful criteria for impossibility results.

Proposition 8.13. *Let $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$ be a computational problem. Suppose that there are two sequences $\{\iota_n^1\}_{n\in\mathbb{N}}$, $\{\iota_n^2\}_{n\in\mathbb{N}} \subset \Omega$ satisfying the following conditions:*

(a) There are sets $S^1, S^2 \subset \mathcal{M}$ *and* $\kappa > 0$ *such that* $\inf_{x_1 \in S^1, x_2 \in S^2} d(x_1, x_2) > \kappa$ *and* $\Xi(\iota_n^j) \subset S^j$ *for* $j = 1, 2$.

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

Then, if we consider $\{\Xi, \Omega, \mathcal{M}, \Lambda\}^{\Delta_1}$*, we have the following:*

(i) For any sequential general algorithm Γ *and* $M \in \mathbb{N}$*, there exists* $\iota \in \Omega$ *and* $\widehat{\Lambda} \in \mathcal{L}^1(\Lambda)$ *such that*

$$\operatorname{dist}(\Gamma(\iota), \Xi(\iota)) > \kappa/2 \quad \textit{or} \quad S_\Gamma(\iota) > M.$$

(ii) If there is an $\iota^0 \in \Omega$ *such that for every* $f \in \Lambda$ *we have that (b) is satisfied with* $c_f = f(\iota^0)$*, then for any RGA* Γ *and* $p \in [0, 1/2)$*, there exists* $\iota \in \Omega$ *and* $\widehat{\Lambda} \in \mathcal{L}^1(\Lambda)$ *such that* $\mathbb{P}_\iota(\operatorname{dist}(\Gamma(\iota), \Xi(\iota)) \geq \kappa/2) > p$.

Proof. Without loss of generality, we assume that $\Omega = \{\iota_n^1\}_{n \in \mathbb{N}} \cup \{\iota_n^2\}_{n \in \mathbb{N}}$. Part (ii) follows immediately from a Proposition of [15], so we only prove part (i). Let Γ be a sequential general algorithm and $M \in \mathbb{N}$. We will construct the required $\widehat{\Lambda} \in \mathcal{L}^1(\Lambda)$ inductively. By Definition 8.3 and the setup of Δ_1-information for Λ, there exists some $f_{k_1} \in \Lambda$ and $n_1 \in \mathbb{N}$ such that for all $\iota \in \Omega$ and for all $\widehat{\Lambda} \in \mathcal{L}^1(\Lambda)$, we have $f_{\iota,1}^\Gamma = f_{k_1,n_1}$. We set $f_{k_1,n_1}(\iota_m^j) = c_{f_{k_1}}$ for all $m \geq n_1$ and choose $f_{k_1,n_1}(\iota_m^j)$ consistently for $m < n_1$. Again by Definition 8.3 and the setup of Δ_1-information for Λ, it follows that there exists $f_{k_2} \in \Lambda$ and $n_2 \in \mathbb{N}$ (which without loss of generality $\geq n_1$) such that for all $m \geq n_1$, either $\Lambda_\Gamma(\iota_m^j) = \{f_{k_1,n_1}\}$ or $f_{\iota_m^j,2}^\Gamma = f_{k_2,n_2}$. In the latter case, we set $f_{k_2,n_2}(\iota_m^j) = c_{f_{k_2}}$ for all $m \geq n_2$ and choose $f_{k_2,n_2}(\iota_m^j)$ consistently for $m < n_2$. We continue this process for a maximum of M steps up to $f_{\iota_m^j,\min\{M,|\Lambda_\Gamma(\iota_m^j)|\}}^\Gamma$ as follows. At the qth step after defining f_{k_q,n_q}, by Definition 8.3 and the setup of Δ_1-information for Λ, it follows that there exists $f_{k_{q+1}} \in \Lambda$ and $n_{q+1} \in \mathbb{N}$ (which without loss of generality $\geq n_q$) such that for all $m \geq n_q$, either $\Lambda_\Gamma(\iota_m^j) \subset \{f_{k_1,n_1}, ..., f_{k_q,n_q}\}$ or $f_{\iota_m^j,q+1}^\Gamma = f_{k_{q+1},n_{q+1}}$. In the latter case, we set $f_{k_{q+1},n_{q+1}}(\iota_m^j) = c_{f_{k_{q+1}}}$ for all $m \geq n_{q+1}$ and choose $f_{k_{q+1},n_{q+1}}(\iota_m^j)$ consistently for $m < n_{q+1}$. We can then choose the rest of the function values to obtain $\widehat{\Lambda}$.

Given this $\widehat{\Lambda} \in \mathcal{L}^1(\Lambda)$, suppose for a contradiction that for any $\iota \in \Omega$, $\operatorname{dist}(\Gamma(\iota), \Xi(\iota)) \leq \kappa/2$ and $S_\Gamma(\iota) \leq M$. Without loss of generality, we assume that the above construction is carried out for M steps. It follows that we must have $f(\iota_{n_M}^1) = f(\iota_{n_M}^2)$ for all $f \in \widehat{\Lambda}_\Gamma(\iota_{n_M}^1)$. By (ii) and (iii) of Definition 8.3, it follows that $\Gamma(\iota_{n_M}^1) = \Gamma(\iota_{n_M}^2)$. Let $\epsilon > 0$ be arbitrary. Since $\operatorname{dist}(\Gamma(\iota), \Xi(\iota)) \leq \kappa/2$ for all $\iota \in \Omega$, there exists $s_j \in S^j$ such that $d(\Gamma(\iota_{n_M}^j), s_j) < \kappa/2 + \epsilon$. It follows that

$$\inf_{x_1 \in S^1, x_2 \in S^2} d(x_1, x_2) \leq d(s_1, s_2) \leq d(\Gamma(\iota_{n_M}^1), s_1) + d(\Gamma(\iota_{n_M}^2), s_2) < \kappa + 2\epsilon.$$

Since $\epsilon > 0$ was arbitrary, we have $\inf_{x_1 \in S^1, x_2 \in S^2} d(x_1, x_2) \leq \kappa$, the required contradiction. □

Proposition 8.14. *Let* $\{\Xi, \Omega, \mathcal{M}, \Lambda\}$ *be a computational problem and* $u \geq 2$ *be a positive integer. Suppose that there are* u *sequences* $\{\iota_n^j\}_{n \in \mathbb{N}} \subset \Omega$*, for* $j = 1, ..., u$*, satisfying the following conditions:*

(a) There are sets $S^j \subset \mathcal{M}$*, for* $j = 1, ..., u$*, and* $\kappa > 0$ *such that* $\inf_{x_j \in S^j, x_k \in S^k} d(x_j, x_k) > \kappa$ *for any* $j \neq k$ *and* $\Xi(\iota_n^j) \subset S^j$ *for* $j = 1, ..., u$.

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

Then, if we consider $\{\Xi, \Omega, \mathcal{M}, \Lambda\}^{\Delta_1}$*, for any halting probabilistic Turing machine* $(\Gamma, \mathbb{P})$*,* $M \in \mathbb{N}$ *and* $p \in [0, \frac{u-1}{u})$*, there exists* $\iota \in \Omega$ *and* $\widehat{\Lambda} \in \mathcal{L}^1(\Lambda)$ *such that* $\mathbb{P}\Big(\operatorname{dist}(\Gamma(\iota), \Xi(\iota)) > \kappa/2 \quad \textit{or} \quad S_\Gamma(\iota) > M\Big) > p$.

Proof. Without loss of generality, we can assume that $\Omega = \cup_{j=1}^u \{\iota_n^j\}_{n \in \mathbb{N}}$. Let $(\Gamma, \mathbb{P})$ be a halting probabilistic Turing machine and $M \in \mathbb{N}$, $p \in [0, (u-1)/u)$. Suppose for a contradiction that for all $\iota \in \Omega$ and all $\widehat{\Lambda} \in \mathcal{L}^1(\Lambda)$, we have $\mathbb{P}(\operatorname{dist}(\Gamma(\iota), \Xi(\iota)) > \kappa/2 \text{ or } S_\Gamma(\iota) > M) \leq p$. We will construct the required $\widehat{\Lambda} \in \mathcal{L}^1(\Lambda)$ inductively. Let $\beta \in (0, 1)$ be such that $(1-\beta)^M - p > 1/u$. Such a β exists since $p \in [0, (u-1)/u)$. Since the Turing machine must halt with probability one, there exists finite sets $K_1, N_1 \subset \mathbb{N}$ such that with probability (w.r.t. $\mathbb{P}$) at least $1 - \beta$, for all $\iota \in \Omega$ and for all $\widehat{\Lambda} \in \mathcal{L}^1(\Lambda)$, it holds that $f_{\iota,1}^\Gamma = f_{k_1,n_1}$ for some $k_1 \in K_1$ and $n_1 \in N_1$. We set $f_{k_1,n_1}(\iota_m^j) = c_{f_{k_1}}$ for all $m \geq \max\{n_1 : n_1 \in N_1\}$ and choose $f_{k_1,n_1}(\iota_m^j)$ consistently otherwise.

We continue this process inductively for M steps up to $f^{\Gamma}_{\iota^j_m,\min\{M,|\Lambda_\Gamma(\iota^j_m)|\}}$ as follows. At the qth step after defining f_{k_q,n_q} for $k_q \in K_q$ and $n_q \in N_q$, it follows (since the Turing machine halts with probability one) that there exists finite sets $K_{q+1}, N_{q+1} \subset \mathbb{N}$ with the following property. Let E_{q+1} be the event that for all $\iota^j_m \in \Omega$ with $m \geq \max\{n : n \in N_1 \cup ... \cup N_q\}$, either $f^{\Gamma}_{\iota^j_m,q+1} = f_{k_{q+1},n_{q+1}}$, for some $k_{q+1} \in K_{q+1}$ and $n_{q+1} \in N_{q+1}$, or $|\Lambda(\iota^j_m)| \leq q$. Then $\mathbb{P}(E_{q+1}|\cap_{k\leq q} E_k) \geq 1-\beta$. We then set $f_{k_{q+1},n_{q+1}}(\iota^j_m) = c_{f_{k_{q+1}}}$ for all $m \geq \max\{n : n \in N_1 \cup ... \cup N_{q+1}\}$ and choose $f_{k_{q+1},n_{q+1}}(\iota^j_m)$ consistently otherwise. This ensures the existence of K_{q+2} and N_{q+2}. After the Mth step, we can choose the rest of the function values to obtain $\widehat{\Lambda}$.

It follows that for $m \geq \max\{n : n \in N_1 \cup ... \cup N_M\}$, the outputs $\Gamma(\iota^j_m)$ conditional on the event $E_1 \cap ... \cap E_M \cap \{S_\Gamma(\cdot) \leq M\}$ are equal for $j = 1, ..., u$. Since $\inf_{x_j \in S^j, x_k \in S^k} d(x_1, x_2) > \kappa$ for $j \neq k$, it follows that the events $F_j := \{\text{dist}(\Gamma(\iota^j_m), \Xi(\iota^j_m)) \leq \kappa/2\} \cap \{S_\Gamma(\iota^j_m) \leq M\} \cap E_1 \cap ... \cap E_M$, $j = 1, ..., u$, are disjoint. Moreover, using the fact that $\mathbb{P}(A \cap B) = \mathbb{P}(A) + \mathbb{P}(B) - \mathbb{P}(A \cup B)$,

$$\begin{aligned}\mathbb{P}(F_j) &\geq \mathbb{P}(\{\text{dist}(\Gamma(\iota^j_m), \Xi(\iota^j_m)) \leq \kappa/2\} \cap \{S_\Gamma(\iota^j_m) \leq M\}) + \mathbb{P}(E_1 \cap ... \cap E_M) - 1 \\ &\geq \mathbb{P}(\{\text{dist}(\Gamma(\iota^j_m), \Xi(\iota^j_m)) \leq \kappa/2\} \cap \{S_\Gamma(\iota^j_m) \leq M\}) + (1-\beta)^M - 1 \geq (1-\beta)^M - p > 1/u.\end{aligned}$$

But this contradicts the disjointness of the F_j's. □

8.3. **Proof of Theorem 2.2.**

Proof of Theorem 2.2. We will argue for $m = 1$ and construct such an Ω for this case. The general case of $m > 1$ follows by embedding our construction for $A \in \mathbb{C}^{1\times(N+1-m)}$ in the first row of matrices and vectors of the form

$$\hat{A} = \begin{pmatrix} A & 0 \\ 0 & \alpha I \end{pmatrix}, \quad \hat{y} = (y, 0)^\top, \quad (A, y) \in \Omega,$$

where $I \in \mathbb{C}^{(m-1)\times(m-1)}$ denotes the $(m-1)\times(m-1)$ identity matrix and $\alpha = \alpha(A)$ is chosen such that $\hat{A}\hat{A}^*$ is a multiple of the identity. In particular, such an embedding does not effect the relevant condition numbers (it is straightforward to see that the matrix norm, distance to infeasibility for (P_1) and condition numbers of the mappings are all unchanged). For the classes we consider, the setup of Theorem 2.2 coincides with the Δ_1-information model discussed in §8.1. In particular, we can use Lemmas 8.10, 8.11 and 8.12 to derive the relevant $x_{s,n}$'s in (2.7). This means that we can apply Propositions 8.13 and 8.14 with the metric corresponding to the l^2-norm. Recall that for this theorem, we assume that $w_1 = w_2 = ... = w_N = 1$.

Step 1: Proof for (P_1). First, consider the class defined by

$$\Omega_1 = \left\{(A(\gamma_1;\rho), y) : A(\gamma_1;\rho) \coloneqq \gamma_1 \begin{pmatrix} \frac{1}{\rho_1} & \frac{1}{\rho_2} & \cdots & \frac{1}{\rho_N} \end{pmatrix}, y = 1, \rho_j \in [1-2\delta, 1-\delta]\right\},$$

for fixed $\gamma_1 > 10$ and $\delta \in (0, 1/4)$. We choose γ_1 and δ such that

$$\frac{1-\epsilon}{\gamma_1} \cdot \sup_{\rho_j,\rho_k \in [1-2\delta,1-\delta], j\neq k} \|\rho_j e_j - \rho_k e_k\|_{l^2} = 3 \cdot 10^{-K}, \tag{8.5}$$

$$\frac{1-\epsilon}{\gamma_1} \cdot \inf_{\rho_j,\rho_k \in [1-2\delta,1-\delta], j\neq k} \|\rho_j e_j - \rho_k e_k\|_{l^2} > 2 \cdot 10^{-K}, \tag{8.6}$$

where the e_j denote the canonical basis of $\mathbb{C}^N$. Note that we can ensure $\gamma_1 > 10$ since $\epsilon \leq 1/2$ and $K > 2$. If $\rho_j \in [1-2\delta, 1-\delta)$, for $j = 1, 2$, then by (a simple rescale of) Lemma 8.10,

$$\Xi_{P_1}(A(\gamma_1;(\rho_1, 1-\delta, ..., 1-\delta)), 1) = \frac{(1-\epsilon)\rho_1}{\gamma_1} e_1, \Xi_{P_1}(A(\gamma_1;(1-\delta, \rho_2, 1-\delta, ..., 1-\delta)), 1) = \frac{(1-\epsilon)\rho_2}{\gamma_1} e_2.$$

Since (8.6) holds, it follows by selecting appropriate sequences ι^j_n for choices of $\rho_j = \rho^n_j \uparrow 1-\delta$ that the conditions of Proposition 8.13 hold for Ω_1 with

$$S^j = \left\{\frac{1-\epsilon}{\gamma_1}\rho e_j : \rho \in [1-2\delta, 1-\delta]\right\}, \quad \kappa = 2 \cdot 10^{-K}. \tag{8.7}$$

Moreover, the condition for part (ii) of Proposition 8.13 also holds with $\iota^0 = (A(\gamma_1;(1-\delta, 1-\delta, ..., 1-\delta)), 1)$.

Now suppose for a contradiction that there exists a (halting) RGA (with input $\iota_{A,\mathcal{S}}$) and $p > 1/2$ that produces a NN ϕ_A such that $\min_{y\in\mathcal{S}_A} \inf_{x^*\in\Xi_{P_1}(A,y)} \|\phi_A(y) - x^*\|_{l^2} \leq 10^{-K}$ holds with probability at least p for all $(A, y) \in \Omega_1$. Then there exists a (halting) RGA, Γ, taking $\iota_{A,\mathcal{S}}$ as input that computes a solution of (P_1) to K correct digits with probability at least p on each input in Ω_1. However, this contradicts Proposition 8.13 (ii). Next, consider the class defined by

$$\Omega_2 = \left\{(A(\gamma_2;\rho), y) : A(\gamma_2;\rho) = \gamma_2 \begin{pmatrix} \frac{1}{\rho_1} & \frac{1}{\rho_2} & \cdots & \frac{1}{\rho_N} \end{pmatrix}, y = 1, \rho_j \in [1-2\delta, 1-\delta], \rho_j \neq \rho_k \text{ if } j \neq k\right\},$$

where $\gamma_2 = \gamma_1/10 > 1$ so that

$$\sup_{\rho_j,\rho_k\in[1-2\delta,1-\delta],j\neq k} \|\rho_j e_j - \rho_k e_k\|_{l^2} = 3 \cdot 10^{-K+1} \cdot \frac{\gamma_2}{1-\epsilon}, \tag{8.8}$$

$$\inf_{\rho_j,\rho_k\in[1-2\delta,1-\delta],j\neq k} \|\rho_j e_j - \rho_k e_k\|_{l^2} > 2 \cdot 10^{-K+1} \cdot \frac{\gamma_2}{1-\epsilon}. \tag{8.9}$$

By extending the argument above to $u = N + 1 - m = N$ (recall without loss of generality that $m = 1$) sequences and sets S^j defined as in (8.7), the conditions of Proposition 8.14 hold with $\kappa = 2 \cdot 10^{-K+1}$. Now suppose that there exists a (halting) probabilistic Turing machine $(\Gamma, \mathbb{P})$, $M \in \mathbb{N}$ and $p \in \left[0, \frac{N-m}{N+1-m}\right)$, such that for any $(A, 1) \in \Omega_2$, Γ computes a NN ϕ_A with

$$\mathbb{P}\left(\inf_{x^*\in\Xi_{P_1}(A,y)} \|\phi_A(y) - x^*\|_{l^2} > 10^{1-K} \text{ or the sample size needed to construct } \phi_A > M\right) \leq p.$$

Then there exists a (halting) probabilistic Turing machine that computes a solution of (P_1) to $K - 1$ correct digits on each input in Ω_2 with sample size at most M with probability at least $1-p$. However, this contradicts Proposition 8.14.

We now set $\Omega = \Omega_1 \cup \Omega_2$. Note that the negative statements of part (i) and (ii) follow from the above arguments by considering restrictions to Ω_1 and Ω_2 respectively. Hence, we are left with proving the condition number bounds, part (iii) and the positive part of part (ii). First, note that $\mathrm{Cond}(AA^*) = 1$ for any $(A, y) \in \Omega$. For any $(A, y) \in \Omega$, we have $\nu(A, y) = \|A\| \geq 1 = \|y\|_{l^2}$ and hence $C_{\mathrm{FP}}(\Xi_{P_1}, \Omega) \leq 1$. To bound the final condition number, first note that if $\rho_j, \rho'_j \leq 1$, then $\|\rho - \rho'\|_{l^2} \leq \|\sum_{j=1}^N (\frac{1}{\rho_j} - \frac{1}{\rho'_j}) e_j\|_{l^2}$. Let $(A(\gamma_1;\rho), 1) \in \Omega_1$, then if $(A(\gamma_1;\rho'), 1) \in \Omega_1$ with $\Delta(\rho, \rho') \coloneqq \gamma_1 \|\sum_{j=1}^N (\frac{1}{\rho_j} - \frac{1}{\rho'_j}) e_j\|_{l^2}$ sufficiently small,

$$\mathrm{dist}(\Xi_{P_1}(A(\gamma_1;\rho'), 1), \Xi_{P_1}(A(\gamma_1;\rho), 1)) \leq \frac{1-\epsilon}{\gamma_1} \|\rho - \rho'\|_{l^2}.$$

It follows that

$$\lim_{\beta\downarrow 0} \sup_{\substack{(A(\gamma_1;\rho'),1)\in\Omega_1 \\ \Delta(\rho,\rho')\leq\beta}} \frac{\mathrm{dist}(\Xi_{P_1}(A(\gamma_1;\rho'), 1), \Xi_{P_1}(A(\gamma_1;\rho), 1))}{\Delta(\rho, \rho')} \leq \frac{1-\epsilon}{\gamma_1^2} < 1.$$

A similar argument holds for $(A(\gamma_2;\rho), 1) \in \Omega_2$, and hence $\mathrm{Cond}(\Xi_{P_1}, \Omega) \leq 1$.

We now prove the positive parts of (ii) and (iii). We begin with (ii) and describe the algorithm informally, noting that the output of the algorithm, $\Gamma(A)$, yields a NN which maps $y = 1$ to $\Gamma(A) \in \mathbb{C}^N$. Given an input $(A, y) \in \Omega$, the algorithm first tests the size of $A_{1,1}$ to determine whether $(A, y) \in \Omega_1$ or $(A, y) \in \Omega_2$. Explicitly, we note that $A_{1,1}$ is positive and bounded away from 0. Hence, with one sample from $\iota_{A,\mathcal{S}}$ we can determine $A_{1,1}$ to an accuracy of at least $0.01 \cdot A_{1,1}$ and such that, simultaneously, the corresponding approximation of $A_{1,1}^{-1}$ is accurate to at least 10^{-K}. If $(A, y) \in \Omega_1$, then $A_{1,1} \in \gamma_1 \cdot [1, 2]$ whereas if $(A, y) \in \Omega_2$, then $A_{1,1} \in \gamma_2 \cdot [1, 2]$. Since $\gamma_2 = \gamma_1/10$, this level of accuracy is enough to determine whether $(A, y) \in \Omega_1$ or $(A, y) \in \Omega_2$. Next, if the algorithm determines $(A, y) \in \Omega_1$, it outputs the corresponding approximation of $(1 - \epsilon)A_{1,1}^{-1} e_1 = \frac{1-\epsilon}{\gamma_1} \rho_1 e_1$ correct to 10^{-K} in the l^2-norm from the sample. Since

$$\sup_{\rho_j,\rho_k\in[1-2\delta,1-\delta],j\neq k} \|\rho_j e_j - \rho_k e_k\|_{l^2} = 3 \cdot 10^{-K} \cdot \frac{\gamma_1}{1-\epsilon},$$

it follows that $\inf_{x^* \in \Xi_{P_1}(A,y)} \|\Gamma(A) - x^*\|_{l^2} \leq 4 \cdot 10^{-K} < 10^{-K+1}$. On the other hand, if the algorithm determines $(A, y) = (A(\gamma_2; \rho), y) \in \Omega_2$, then we know that all of the ρ_j are distinct. The algorithm continues to sample $\iota_{A,\mathcal{S}}$ until we determine j such that $\rho_j = \min_k \rho_k$. It then outputs an approximation of $(1 - \epsilon)A_{1,j}^{-1} e_j = \Xi_{P_1}(A, y)$ correct to 10^{-K} in the l^2-norm. Such as approximation can be computed using $\iota_{A,\mathcal{S}}$. It then follows that $\|\Gamma(A) - \Xi_{P_1}(A, y)\|_{l^2} \leq 10^{-K} < 10^{-K+1}$ and this finishes the proof of (ii). Finally, to prove (iii), note that the arguments above show that, given an input $(A, y) \in \Omega$, we can use one sample ($L = 1$) of $\iota_{A,\mathcal{S}}$ to compute an approximation of $A_{1,1}^{-1}$ with error bounded by 10^{-K}. We simply set $\Gamma(A)$ to be $(1 - \epsilon)e_1$ multiplied by the approximation of $A_{1,1}^{-1}$. Using (8.5) and (8.8), it follows that

$$\inf_{x^* \in \Xi_{P_1}(A,y)} \|\Gamma(A) - x^*\|_{l^2} \leq 3 \cdot 10^{-K+1} + 10^{-K} < 10^{-K+2}.$$

Step 2: Proof for (P_2). This is almost identical step 1 with replacing Lemma 8.10 with Lemma 8.11. The other changes are replacing ϵ with the suitable $\rho_i/2$ in the solution of each LASSO problem, including the additional scale λ in the definition of the matrices (see Lemma 8.11) and choosing $\gamma_1 \lambda > 10$ and $\delta \in (0, 1/4)$ such that (recall that $\lambda \leq 1$)

$$\sup_{\rho_j, \rho_k \in [1-2\delta, 1-\delta], j \neq k} \frac{1}{\lambda \gamma_1} \cdot \|\rho_j(1 - \rho_j/2)e_j - \rho_k(1 - \rho_k/2)e_k\|_{l^2} = 3 \cdot 10^{-K} \tag{8.10}$$

$$\inf_{\rho_j, \rho_k \in [1-2\delta, 1-\delta], j \neq k} \frac{1}{\lambda \gamma_1} \cdot \|\rho_j(1 - \rho_j/2)e_j - \rho_k(1 - \rho_k/2)e_k\|_{l^2} > 2 \cdot 10^{-K}. \tag{8.11}$$

Let $f(x) = (1 - x/2)x$, then for $x \in [0, 1]$, $|f'(x)| \leq 1$. It follows that

$$\left\| \sum_{j=1}^{N} \left(\rho_j(1 - \rho_j/2) - \rho_j'(1 - \rho_j'/2) \right) e_j \right\|_{l^2} \leq \left\| \sum_{j=1}^{N} \left(\frac{1}{\rho_j} - \frac{1}{\rho_j'} \right) e_j \right\|_{l^2}.$$

Hence, for sufficiently small δ, for any $(A(\gamma_1; \rho), 1) \in \Omega_1$ (recall the additional factor of λ) and ρ' sufficiently close to ρ with $(A(\gamma_1; \rho'), 1) \in \Omega_1$,

$$\frac{\operatorname{dist}(\Xi_{P_2}(A(\gamma_1; \rho'), 1), \Xi_{P_2}(A(\gamma_1; \rho), 1))}{\gamma_1 \lambda \left\| \sum_{j=1}^{N} \left(\frac{1}{\rho_j} - \frac{1}{\rho_j'} \right) e_j \right\|_{l^2}} \leq \frac{1}{\gamma_1^2 \lambda^2} < 1,$$

with the same bound holding for Ω_2. It follows that $\operatorname{Cond}(\Xi_{P_2}, \Omega) \leq 1$.

Step 3: Proof for (P_3). This is almost identical step 2 with replacing Lemma 8.11 with Lemma 8.12, and deleting the corresponding factors of $1 - \rho_j/2$. □

8.4. **Details on numerical example in §2.1.** In this section, we elaborate on the numerical example in §2.1. The example is a simplification of the arguments found in the proof of Theorem 2.2 that uses Lemma 8.12 extensively. In our experiment, we use $N_1 = 2$ and $\lambda = 1$, but for full generality, we do not keep these parameters fixed in the discussion below. We assume throughout that $\lambda \in (0, 1]$ and $N_1 \geq 2$. The experiment is done for real matrices so that the LISTA network architecture can be used.

Let $\gamma > 0$, $\rho \in \mathbb{R}^{N_1} \setminus \{0\}$, and $D \in \mathbb{C}^{N_2+1 \times N_2+1}$ be a unitary discrete cosine transform matrix. Define

$$A(\gamma, \rho) \coloneqq D \begin{pmatrix} a(\gamma, \rho)^\top & 0 \\ 0 & \|a(\gamma, \rho)\|_{l^2} I \end{pmatrix}, \quad \text{where} \quad a(\gamma, \rho)^\top \coloneqq \gamma \begin{pmatrix} \frac{1}{\rho_1} & \cdots & \frac{1}{\rho_{N_1}} \end{pmatrix} \in \mathbb{R}^{1 \times N_1},$$

and $I \in \mathbb{R}^{N_2 \times N_2}$ is the identity matrix. Observe that $A(\gamma, \rho) \in \mathbb{R}^{m \times N}$, with $N = N_1 + N_2$ and $m = N_2 + 1$ and that A has irrational entries (hence only approximations can be used in real-life computations). Furthermore, let $\delta = 1/6$ and $\gamma_K = \frac{\sqrt{2}}{3\lambda}(1 - \delta) \cdot 10^K$, where K is the parameter from Theorem 2.2. Also let

$$y(x^{(2)}) = D \begin{pmatrix} 1 & \|a(\gamma, \rho)\|_{l^2} x^{(2)} \end{pmatrix}^\top \in \mathbb{R}^{N_2+1} \quad \text{for} \quad x^{(2)} \in \mathbb{R}^{N_2} \tag{8.12}$$

and $\Omega_K = \left\{ \left(y(x^{(2)}), A(\gamma_K, \rho) \right) : \rho_j \in [1 - 2\delta, 1 - \delta], x^{(2)} \in \mathbb{R}^{N_2} \right\}$. Next define

$$\rho' = \begin{pmatrix} \rho_1' & 1 - \delta & \cdots & \cdots & 1 - \delta \end{pmatrix} \quad \text{and} \quad \rho^\sharp = \begin{pmatrix} 1 - \delta & \rho_2^\sharp & 1 - \delta & \cdots & 1 - \delta \end{pmatrix}$$

where $\rho_1', \rho_2^\sharp \in [1-2\delta, 1-\delta)$. We let e_i denote the i'th canonical basis vector for $\mathbb{R}^{N_1}$ and define $x' = \frac{\rho_1'}{\lambda\gamma_K} e_1$ and $x^\sharp = \frac{\rho_1^\sharp}{\lambda\gamma_K} e_2$. For this choice of parameters we have from Lemma 8.12 and the fact that D is unitary that $\left(x', x^{(2)}\right)^\top \in \Xi_3\left(A(\gamma_K, \rho'), y(x^{(2)})\right)$ and $\left(x^\sharp, x^{(2)}\right)^\top \in \Xi_3\left(A(\gamma_K, \rho^\sharp), y(x^{(2)})\right)$.

Observe that we can let $A(\gamma_K, \rho')$ and $A(\gamma_K, \rho^\sharp)$ become arbitrary close by letting $\rho_1', \rho_2^\sharp \uparrow 1-\delta$. We will let $\rho_1' = \rho_2^\sharp$, and notice that for this choice the data $y(x^{(2)})$ are the same for both inputs. However, regardless of the choice of $\rho_1', \rho_2^\sharp \in [1-2\delta, 1-\delta)$ the minimisers for the two problems are bounded away from each other. In particular, we have that

$$\inf_{\rho_1', \rho_2^\sharp \in [1-2\delta, 1-\delta]} \frac{1}{\lambda\gamma_K} \|\rho_1' e_1 - \rho_2^\sharp e_2\|_{l^2} > 2 \cdot 10^K \quad \text{and} \quad \sup_{\rho_1', \rho_2^\sharp \in [1-2\delta, 1-\delta]} \frac{1}{\lambda\gamma_K} \|\rho_1' e_1 - \rho_2^\sharp e_2\|_{l^2} = 3 \cdot 10^K$$

which implies that $10^K < \|(x', x^{(2)})^\top - (x^\sharp, x^{(2)})^\top\|_{l^2} < 10^{K+1}$.

In the numerical experiment in §2.1, we take $A = A(\gamma_K, \rho')$ with $\rho_1' = 1 - \delta + 2^{-n-1}$, and approximate this matrix with the matrix $A_n = A(\gamma_K, \rho^\sharp)$, where the parameter $\rho_2^\sharp = 1 - \delta + 2^{-n-1}$. This ensures that $\|A - A_n\| \leq 2^n$. For the trained neural network, we used 8000 triples of the form

$$\iota_{A,\mathcal{S},n} = \left\{ \left(y_{k,n}(x_{k,n}^{(2)}), A_n, (x_n^\sharp, x_{k,n}^{(2)})^\top \right) : k = 1, \ldots, 8000, \text{ and } x^{(2)} \text{ is 5-sparse} \right\}. \tag{8.13}$$

for $n = 10, 20, 30$. Note that it is not necessary to make the $x^{(2)}$ component sparse. This is done merely to make the experiment more realistic, as the main usage of the problems (P_j) are for recovery of sparse vectors.

9. Proof of Theorem 5.5

A roadmap for the proof is as follows. We consider the problem (P_3) and unroll iterations of Chambolle and Pock's primal-dual algorithm [37, 38]. These iterations are approximated by NNs in Theorem 9.5, where we obtain bounds on a rescaled version of the objective function in (9.9). The assumption of weighted rNSPL then allows us to relate the bounds proven in Theorem 9.5 to bounds on the distance of the output of the NN to the wanted vector and also, simultaneously, prove stability. This also allows the acceleration to exponential convergence through a restart scheme (with a reweighting at each restart). We begin with the proof of Lemma 5.4, which allows us to consider the approximation matrices A_l in the construction of the NNs. We also state some results from compressed sensing that are needed in our proofs. We then discuss preliminary results on unrolling iterative algorithms for (P_3), which are used in the proof of Theorem 5.5. When writing out NNs in the proofs, we will use $\xrightarrow{\text{NL}}$ arrows to denote the non-linear maps and $\xrightarrow{\text{L}}$ arrows to denote the affine maps.

9.1. **Some results from compressed sensing.**

Proof of Lemma 5.4. Let Δ be a $(\mathbf{s}, \mathbf{M})$ support set and $x \in \mathbb{C}^N$, then

$$\|x_\Delta\|_{l^2} \leq \frac{\rho\|x_{\Delta^c}\|_{l_w^1}}{\sqrt{\xi}} + \gamma\|Ax\|_{l^2} \leq \frac{\rho\|x_{\Delta^c}\|_{l_w^1}}{\sqrt{\xi}} + \gamma\|\hat{A}x\|_{l^2} + \gamma\|\hat{A} - A\|\|x\|_{l^2}. \tag{9.1}$$

Note that $\min_{k=1,\ldots,r} w_{(k)}^2 \sum_{j\in\Delta^c} |x_j|^2 \leq (\sum_{j\in\Delta^c} |x_j|\, w_j)^2 = \|x_{\Delta^c}\|_{l_w^1}^2$ and hence, (9.1) implies that

$$\|x_\Delta\|_{l^2} \leq \frac{\rho\|x_{\Delta^c}\|_{l_w^1}}{\sqrt{\xi}} + \gamma\|\hat{A}x\|_{l^2} + \gamma\|\hat{A} - A\| \left(\|x_\Delta\|_{l^2} + \frac{\|x_{\Delta^c}\|_{l_w^1}}{\min_{k=1,\ldots,r} w_{(k)}} \right).$$

Rearranging now gives the result. □

The following results are taken from the compressed sensing literature [1].

Lemma 9.1 (rNSPL implies l_w^1 distance bound). *Suppose that A has the weighted rNSPL of order $(\mathbf{s}, \mathbf{M})$ with constants $0 < \rho < 1$ and $\gamma > 0$. Let $x, z \in \mathbb{C}^N$, then*

$$\|z - x\|_{l_w^1} \leq \frac{1+\rho}{1-\rho} \left(2\sigma_{\mathbf{s},\mathbf{M}}(x)_{l_w^1} + \|z\|_{l_w^1} - \|x\|_{l_w^1} \right) + \frac{2\gamma}{1-\rho} \sqrt{\xi} \|A(z-x)\|_{l^2}. \tag{9.2}$$

Lemma 9.2 (rNSPL implies l^2 distance bound). *Suppose that A has the weighted rNSPL of order $(\mathbf{s}, \mathbf{M})$ with constants $0 < \rho < 1$ and $\gamma > 0$. Let $x, z \in \mathbb{C}^N$, then*

$$\|z - x\|_{l^2} \leq \left(\rho + \frac{(1+\rho)\kappa^{1/4}}{2}\right) \frac{\|z - x\|_{l^1_w}}{\sqrt{\xi}} + \left(1 + \frac{\kappa^{1/4}}{2}\right) \gamma \|A(z - x)\|_{l^2}. \tag{9.3}$$

9.2. **Preliminary constructions of neural networks.** When constructing NNs, we will make use of the following maps from $\mathbb{C}^M$ to $\mathbb{C}^M$, defined for various $M \in \mathbb{N}$ and $\beta \in \mathbb{Q}_{>0}$ by $\psi^0_\beta(x) = \max\{0, 1 - \beta/\|x\|_{l^2}\}x$, $\quad \psi^1(x) = \min\{1, \|x\|^{-1}_{l^2}\}x$.

Lemma 9.3. *Let $M \in \mathbb{N}$, $\beta \in \mathbb{Q}_{>0}$ and $\theta \in \mathbb{Q}_{>0}$. Then there exists neural networks $\phi^0_{\beta,\theta}, \phi^1_\theta \in \mathcal{N}_{\mathbf{D},3,2}$ with $\mathbf{D} = (M, 2M, M+1, M)$ such that $\|\phi^0_{\beta,\theta}(x) - \psi^0_\beta(x)\|_{l^2} \leq \theta$ and $\|\phi^1_\theta(x) - \psi^1(x)\|_{l^2} \leq \theta$ for all $x \in \mathbb{C}^M$, and the non-linear maps can be computed from sqrt_θ and finitely many arithmetic operations and comparisons.*

Proof. We deal only with the case of ψ^0_β since the case of ψ^1 is nearly identical. Consider the maps $\phi^0_{\beta,\theta}$:

$$x \xrightarrow{\mathrm{L}} \begin{pmatrix} x \\ x \end{pmatrix} \xrightarrow{\mathrm{NL}} \begin{pmatrix} |x_1|^2 \\ |x_2|^2 \\ \vdots \\ |x_M|^2 \\ x \end{pmatrix} \xrightarrow{\mathrm{L}} \begin{pmatrix} \sum_{j=1}^M |x_j|^2 \\ x \end{pmatrix} \xrightarrow{\mathrm{NL}} \begin{pmatrix} \max\left\{0, 1 - \frac{0}{\mathrm{sqrt}_\theta(\|x\|^2_{l^2})}\right\} \\ \max\left\{0, 1 - \frac{\beta}{\mathrm{sqrt}_\theta(\|x\|^2_{l^2})}\right\} x \end{pmatrix} \xrightarrow{\mathrm{L}} \max\left\{0, 1 - \frac{\beta}{\mathrm{sqrt}_\theta(\|x\|^2_{l^2})}\right\} x.$$

The first, third and final arrows are simple affine maps. The second arrow applies pointwise modulus squaring, which can be done using finitely many arithmetic operations. The penultimate arrow applies a non-linear map which can be computed from one application of sqrt_θ and finitely many arithmetic operations and comparisons. The bound $\|\psi^0_\beta(x) - \phi^0_{\beta,\theta}(x)\|_{l^2} \leq \theta$ follows from a simple case by case analysis. □

The final piece of machinery needed is a NN approximation of applying a pointwise version of ψ^0_β.

Lemma 9.4. *Let $s, \theta \in \mathbb{Q}_{>0}$, $w \in \mathbb{Q}^N_{>0}$ and for $\hat{x} \in \mathbb{C}^N$ consider the minimisation problem*

$$\mathrm{argmin}_{x \in \mathbb{C}^N} \|x\|_{l^1_w} + s\|x - \hat{x}\|^2_{l^2}. \tag{9.4}$$

Let $\tilde{x}_s(\hat{x})$ denote the solution of (9.4). Then, there exists $\phi_{s,\theta} \in \mathcal{N}_{\mathbf{D},2,1}$ such that

$$\|\phi_{s,\theta}(\hat{x}) - \tilde{x}_s(\hat{x})\|_{l^2} \leq \theta \|w\|_{l^2}, \quad \forall \hat{x} \in \mathbb{C}^N \tag{9.5}$$

and $\mathbf{D} = (N, N, N)$. Each affine map in the NN is linear and is an arithmetic function of w. Moreover, the non-linear maps used can be computed from sqrt_θ and finitely many arithmetic operations and comparisons.

Proof. Let $B = \mathrm{diag}(w_1, ..., w_N) \in \mathbb{Q}^{N \times N}$ and consider the function $F(y) = \|By\|_{l^1}/(2s) = \|y\|_{l^1_w}/(2s)$. We write the minimisation problem (9.4) as $\mathrm{prox}_F(\hat{x})$. Given $y \in \mathbb{C}^N$, we identify $y = (y_1, y_2)^\top \in \mathbb{R}^{2N}$.

First, for $\beta > 0$ and $x \in \mathbb{R}^n$ recall that the proximal operator of a multiple of the l^2-norm is

$$\mathrm{prox}_{\beta\|\cdot\|_{l^2}}(x) = \max\{0, 1 - \beta/\|x\|_{l^2}\}x. \tag{9.6}$$

Thus, for $\beta > 0$ we define $\varphi_\beta(y) = (v(y, \beta) * y_1, v(y, \beta) * y_2)^\top$, where $*$ denotes pointwise multiplication and $v(y, \beta)_j = \max\{0, 1 - \beta/\sqrt{y^2_{1,j} + y^2_{2,j}}\}$ for $j = 1, ..., N$. The function φ_β simply corresponds to a proximity map of the l^2-norm applied component-wise to the complexified version of y. Using (9.6), we have

$$\begin{aligned} \mathrm{prox}_F(y) &= \mathrm{argmin}_{z \in \mathbb{C}^N} \frac{1}{2s}\|Bz\|_{l^1} + \frac{1}{2}\|z - y\|^2_{l^2} \\ &= \mathrm{argmin}_{z \in \mathbb{C}^N} \sum_{j=1}^N \left(\frac{B_{jj}}{2s} \sqrt{z^2_{1,j} + z^2_{2,j}} + \frac{1}{2}\left((z_{1,j} - y_{1,j})^2 + (z_{2,j} - y_{2,j})^2\right) \right) \end{aligned}$$

It follows that (in complex vector form) $[\mathrm{prox}_F(y)]_j = \{0, 1 - \frac{B_{jj}/(2s)}{|y_j|}\}y_j, \quad \text{for } j = 1, \ldots, N$. We can therefore write $\mathrm{prox}_F(y) = B\varphi_{(2s)^{-1}}(B^{-1}y)$. We unroll the computation of $\mathrm{prox}_F(\hat{x})$ via:

$$\hat{x} \xrightarrow{\mathrm{L}} B^{-1}\hat{x} \xrightarrow{\mathrm{NL}} \varphi_{(2s)^{-1}}(B^{-1}\hat{x}) \xrightarrow{\mathrm{L}} B\varphi_{(2s)^{-1}}(B^{-1}y).$$

The first arrow is a simple linear map, the second applies $\varphi_{(2s)^{-1}}$ and the third is a linear map. We approximate this by replacing $v(y,\beta)_j y_j$ with $\phi^0_{\beta,\theta}(y_{1,j} + y_{2,j}i)$ (denoting the replacement of $\varphi_{(2s)^{-1}}$ by $\varphi^\theta_{(2s)^{-1}}$) where $\phi^0_{\beta,\theta}$ is the NN from Lemma 9.3 with $M = 1$. This clearly gives $\phi_{s,\theta} \in \mathcal{N}_{\mathbf{D},2,1}$, so we need to only bound the error. From Lemma 9.3 we have

$$\left\|\mathrm{prox}_F(\hat{x}) - B\varphi^\theta_{(2s)^{-1}}(B^{-1}\hat{x})\right\|_{l^2} = \left\|B\left(\varphi_{(2s)^{-1}}(B^{-1}\hat{x}) - \varphi^\theta_{(2s)^{-1}}(B^{-1}\hat{x})\right)\right\|_{l^2} \leq \theta\|w\|_{l^2}.$$

The bound (9.5) now follows. □

The following theorem proves that one can construct NNs with objective function bounds. The proof constructs approximations of unrolled iterations of Chambolle and Pock's primal-dual algorithm [37, 38]. We have used b to denote part of the inputs of the NNs, instead of y, to avoid a clash of notation with the usual notation for primal-dual iterations (y is used to denote a dual variable). The bounds in part 2 of Theorem 9.5 will be combined with results from §9.1 to construct the families of NNs in Theorem 5.5.

Theorem 9.5. *Let $A \in \mathbb{Q}[i]^{m\times N}$ and $\theta \in \mathbb{Q}_{>0}$. Suppose also that $L_A \in \mathbb{Q}_{\geq 1}$ is an upper bound for $\|A\|$, and that $\tau, \sigma \in \mathbb{Q}_{>0}$ are such that $\tau\sigma L_A^2 < 1$. Let $\lambda \in \mathbb{Q}_{>0}$, $w \in \mathbb{Q}^N_{>0}$ and consider the resulting optimisation problem (P_3). Then there exists an algorithm that constructs a sequence of neural networks $\left\{\phi^A_{n,\lambda}, \theta\right\}$ with the following properties:*

(1) (Size) Each $\phi^A_{n,\lambda} : \mathbb{C}^{m+N} \to \mathbb{C}^N$ takes as input data $b \in \mathbb{C}^m$ and an initial guess $x_0 \in \mathbb{C}^N$, both of which are completely general. Also, $\phi^A_{n,\lambda} \in \mathcal{N}_{\mathbf{D}_n,3n+1,3}$ with

$$\mathbf{D}_n = (m+N, \underbrace{2N+m, 2(N+m), 2N+m+1}_{\text{repeated } n \text{ times}}, N).$$

(2) ($\mathcal{O}(n^{-1} + n\theta)$ Error Control) Let

$$C = (1 + \|w\|_{l^2} + 2\sigma\|A\|\|w\|_{l^2})\sqrt{\frac{\tau+\sigma}{1-\tau\sigma L_A^2}}\sqrt{\frac{\tau+\sigma}{\tau\sigma}}, \tag{9.7}$$

then for any inputs $b \in \mathbb{C}^m$ and $x_0 \in \mathbb{C}^N$, there exists a vector $\psi_n(b, x_0) \in \mathbb{C}^N$ with

$$\left\|\psi_n(b,x_0) - \phi^A_{n,\lambda}(b,x_0)\right\|_{l^2} \leq n\theta C \tag{9.8}$$

such that for any $x \in \mathbb{C}^N$ and $\eta \in [0,1]$, it holds that

$$\lambda\|\psi_n(b,x_0)\|_{l^1_w} - \lambda\|x\|_{l^1_w} + \eta\|A\psi_n(b,x_0) - b\|_{l^2} - \|Ax - b\|_{l^2} \leq \frac{1}{n}\left(\frac{\|x - x_0\|^2_{l^2}}{\tau} + \frac{\eta^2}{\sigma}\right). \tag{9.9}$$

Proof. We use the notation $b \in \mathbb{C}^m$ to denote an input vector for our NNs throughout the proof and reserve y to denote dual vectors, consistent with the literature on primal-dual algorithms for saddle point problems.

Step 1: The first step is to consider an equivalent optimisation problem over $\mathbb{R}$ instead of $\mathbb{C}$, and rewrite the problem as a saddle point problem. For $x \in \mathbb{C}^N$, let $x_1 = \mathrm{real}(x)$ and $x_2 = \mathrm{imag}(x)$ and consider $x = (x_1, x_2)^\top$ as a vector in $\mathbb{R}^{2N}$ (and likewise for the dual variables). With an abuse of notation, we use the same notation for complex $x \in \mathbb{C}^N$ and the corresponding vector in $\mathbb{R}^{2N}$, though it will be clear from the context whether we refer to the complex or real case. We let $c = (\mathrm{real}(b), \mathrm{imag}(b))^\top$. Define the matrices

$$K_1 = \begin{pmatrix} \mathrm{real}(A) & -\mathrm{imag}(A) \\ \mathrm{imag}(A) & \mathrm{real}(A) \end{pmatrix} \in \mathbb{R}^{2m\times 2N}, \quad K_2 = \begin{pmatrix} \mathrm{real}(B) & -\mathrm{imag}(B) \\ \mathrm{imag}(B) & \mathrm{real}(B) \end{pmatrix} \in \mathbb{R}^{2N\times 2N},$$

corresponding to multiplication by the matrices A and $B \coloneqq \mathrm{diag}(w_1, ..., w_N)$ respectively. Let $\widetilde{F}_1 : \mathbb{R}^{2N} \to \mathbb{R}$ be defined by $\widetilde{F}_1(x) = \sum_{j=1}^N \sqrt{(K_2x)_j^2 + (K_2x)_{j+N}^2}$ and $\widetilde{F}_3(x) = \lambda\widetilde{F}_1(x)$. Then (P_3) is equivalent to

$\min_{x\in\mathbb{R}^{2N}} \quad \widetilde{F}_3(x) + \|K_1 x - c\|_{l^2}$ and L_A is an upper bound for $\|K_1\|$. The saddle point formulation of the problem is given by

$$\min_{x\in\mathbb{R}^{2N}} \max_{y\in\mathbb{R}^{2m}} \mathcal{L}(x,y) \coloneqq \langle K_1 x, y\rangle + \widetilde{F}_3(x) - f_3^*(y), \tag{9.10}$$

where $f_3^*(y) = \chi_{B_1(0)}(y) + \langle c, y\rangle$, and χ_S denotes the indicator function of a set S, taking the value 0 on S and $+\infty$ otherwise, and $B_1(0)$ denotes the closed l^2 unit ball.

Step 2: We will solve (9.10) by approximating Chambolle and Pock's primal-dual algorithm [37] (with a shift of updates considered in [38]) with a NN. We will write the iteration as an instance of the proximal point algorithm [96] and gain a non-expansive map in a norm which we relate to the standard Euclidean norm.

We start by setting $x^0 = x_0$ (one of the inputs of the NN) and $y^0 = 0$. Recall that for a convex function h, we have that $x = \mathrm{prox}_h(z)$ if and only if $z \in x + \partial h(x)$, where ∂h denotes the subdifferential of h, see, for example, [53, Prop. B.23]. Letting $g = \widetilde{F}_3$ and $f^* = f_3^*$, the exact iterates can be written as

$$\begin{aligned} x^{k+1} &= \mathrm{argmin}_{x\in\mathbb{R}^{2N}}\, g(x) + \frac{1}{2\tau}\|x - (x^k - \tau K_1^* y^k)\|_{l^2}^2 = (I + \tau\partial g)^{-1}(x^k - \tau K_1^* y^k) \\ y^{k+1} &= \mathrm{argmin}_{y\in\mathbb{R}^{2m}}\, f^*(y) + \frac{1}{2\sigma}\|y - (y^k + \sigma K_1(2x^{k+1} - x^k)\|_{l^2}^2 \\ &= (I + \sigma\partial f^*)^{-1}\left[y^k + \sigma K_1(2x^{k+1} - x^k)\right]. \end{aligned} \tag{9.11}$$

Note that the solutions of these proximal mappings are given by Lemmas 9.3 and 9.4 and their proofs, as we describe explicitly below in step 4. The function f^* also depends on the input data b.

Let $z = (x,y)^\top$ and define the matrix

$$M_{\tau\sigma} = \begin{pmatrix} \frac{1}{\tau} I & -K_1^* \\ -K_1 & \frac{1}{\sigma} I \end{pmatrix} \in \mathbb{R}^{2(m+N)\times 2(m+N)},$$

which is positive definite by the assumption $\tau\sigma L_A^2 < 1$ and hence induces a norm denoted by $\|\cdot\|_{\tau\sigma}$. We can write the iterations as (see, for example, [38, Sec. 3])

$$0 \in M_{\tau\sigma}^{-1}\begin{pmatrix} \partial g & K_1^* \\ -K_1 & \partial f^* \end{pmatrix} z^{k+1} + (z^{k+1} - z^k) \Rightarrow z^{k+1} = \left[I + M_{\tau\sigma}^{-1}\begin{pmatrix} \partial g & K_1^* \\ -K_1 & \partial f^* \end{pmatrix}\right]^{-1} z^k.$$

The multi-valued operator

$$M_{\tau\sigma}^{-1}\begin{pmatrix} \partial g & K_1^* \\ -K_1 & \partial f^* \end{pmatrix}$$

is maximal monotone with respect to the inner product induced by $M_{\tau\sigma}$ [96] and hence the iterates are non-expansive in the norm $\|\cdot\|_{\tau\sigma}$. We also have that

$$\|(x,y)^\top\|_{\tau\sigma}^2 \le \frac{\|x\|_{l^2}^2}{\tau} + \frac{\|y\|_{l^2}^2}{\sigma} + 2L_A\|x\|_{l^2}\|y\|_{l^2} \le \left(\frac{L_A}{\nu} + \tau^{-1}\right)\|x\|_{l^2}^2 + \left(L_A\nu + \sigma^{-1}\right)\|y\|_{l^2}^2,$$

for any $\nu > 0$ by the generalised AM–GM inequality. Choosing $\nu = \sigma L_A$ and using $\tau\sigma L_A^2 < 1$, we have that

$$\|(x,y)^\top\|_{\tau\sigma}^2 \le (\tau^{-1} + \sigma^{-1})\|(x,y)^\top\|_{l^2}^2. \tag{9.12}$$

A similar calculation yields that

$$\|(x,y)^\top\|_{l^2}^2 \le \frac{\tau+\sigma}{1-\tau\sigma L_A^2}\|(x,y)^\top\|_{\tau\sigma}^2. \tag{9.13}$$

Step 3: Next, we use convergence guarantees proven in [38] to obtain inequalities that closely resemble (9.9). Define the ergodic averages $X^k = \frac{1}{k}\sum_{j=1}^k x^j, Y^k = \frac{1}{k}\sum_{j=1}^k y^j$. By convexity, the map from $(x^1,y^1)^\top$ to $(X^k,Y^k)^\top$ is also non-expansive in the norm $\|\cdot\|_{\tau\sigma}$. It also holds (see [38] Theorem 1 and remarks) that

$$\mathcal{L}(X^k, y) - \mathcal{L}(x, Y^k) \le \frac{1}{k}\left(\frac{\|x - x_0\|_{l^2}^2}{\tau} + \frac{\|y\|_{l^2}^2}{\sigma}\right), \quad \forall x \in \mathbb{R}^{2N}, \forall y \in \mathbb{R}^{2m}. \tag{9.14}$$

Let y be parallel to $KX^k - c$ such that $\|y\|_{l^2} = \eta \leq 1$, and x be general in (9.14). This gives

$$\widetilde{F}_3(X^k) - \widetilde{F}_3(x) + \langle K_1 X^k - c, y\rangle + \langle c - K_1 x, Y^k\rangle \leq \frac{1}{k}\left(\frac{\|x - x_0\|_{l^2}^2}{\tau} + \frac{\eta^2}{\sigma}\right).$$

Since $\|Y^k\|_{l^2} \leq 1$ (otherwise we gain a contradiction in that the left-hand side of (9.14) is infinite), this implies

$$\widetilde{F}_3(X^k) - \widetilde{F}_3(x) + \eta\|K_1 X^k - c\|_{l^2} - \|K_1 x - c\|_{l^2} \leq \frac{1}{k}\left(\frac{\|x - x_0\|_{l^2}^2}{\tau} + \frac{\eta^2}{\sigma}\right). \tag{9.15}$$

Step 4: The next step is to unroll the iterations in (9.11) as (complex-valued) NNs that approximate the X^k. We unroll via the following steps:

$$\begin{pmatrix} X^k \\ x^k \\ y^k \end{pmatrix} \xrightarrow{\text{L}} \begin{pmatrix} X^k \\ x^k - \tau A^* y^k \\ y^k - \sigma A x^k \end{pmatrix} \xrightarrow{\text{NL}} \begin{pmatrix} X^k \\ x^{k+1} \\ y^k - \sigma A x^k \end{pmatrix} \xrightarrow{\text{L}} \begin{pmatrix} X^{k+1} \\ x^{k+1} \\ u^k \end{pmatrix} \xrightarrow{\text{NL}} \begin{pmatrix} X^{k+1} \\ x^{k+1} \\ y^{k+1} \end{pmatrix},$$

with $u^k = y^k + \sigma A(2x^{k+1} - x^k) - \sigma b$. The first arrow is a simple linear map, the second computes $x^{k+1} = \left(I + \tau\lambda\partial F_1^A\right)^{-1}(x^k - \tau A^* y^k)$. The third is an affine map and the final arrow applies ψ^1 to u^k. We now define the approximations $\tilde{Z}^k$ and $\tilde{z}^k$ (of $Z^k = (X^k, Y^k)^\top$ and $z^k = (x^k, y^k)^\top$ respectively) defined by replacing ψ^1 with ϕ_θ^1 (Lemma 9.3) and the computation of $(I + \tau\lambda\partial F_1^A)^{-1}(x^k - \tau A^* y^k)$ with $\phi_{(2\tau\lambda)^{-1},\theta}(\tilde{x}^k - \tau A^*\tilde{y}^k)$ (Lemma 9.4). We initialise the network with $\tilde{x}^0 = x_0$ and $y^0 = 0$. Since the composition of two affine maps is affine, it follows that the mapping from (b, x_0) to $\tilde{X}^n$ can be realised by $\phi_{n,\lambda}^A \in \mathcal{N}_{\mathbf{D}_n, 3n+1, 3}$ with

$$\mathbf{D}_n = (m + N, \underbrace{2N + m, 2(N + m), 2N + m + 1}_{\text{repeated } n \text{ times}}, N).$$

Clearly, the sequence of NNs are NNs in the sense of §5.1 and can be constructed by an algorithm (see §8.1).

Step 5: Finally, we bound the difference between Z^k and $\tilde{Z}^k$ to deduce (9.8), and the error bound in the objective function using the inequalities in Step 3. We write $\tilde{x}^k = x^k + e_1^k$, $\tilde{y}^k = y^k + e_2^k$ and clearly have that $e_1^0 = 0$ and $e_2^0 = 0$. We can write $\tilde{x}^{k+1} = \phi_{(2\tau\lambda)^{-1}}(\tilde{x}^k - \tau A^*\tilde{y}^k) + e_3^{k+1}$, with $\|e_3^{k+1}\|_{l^2} \leq \theta\|w\|_{l^2}$ by Lemma 9.4. We also have that $\tilde{y}^{k+1} = \psi^1(\tilde{y}^k + \sigma A(2\tilde{x}^{k+1} - \tilde{x}^k) - \sigma b) + e_4^{k+1}$, with $\|e_4^{k+1}\|_{l^2} \leq \theta$ by Lemma 9.3. Since ψ^1 is non-expansive, it follows that $\tilde{y}^{k+1} = \psi^1(\tilde{y}^k + \sigma A(2(\tilde{x}^{k+1} - e_3^{k+1}) - \tilde{x}^k) - \sigma b) + e_5^{k+1}$, with $\|e_5^{k+1}\|_{l^2} \leq \theta(1 + 2\sigma\|A\|\|w\|_{l^2})$. We can then use the fact that the iterates applied with the exact proximal maps are non-expansive in the norm $\|\cdot\|_{\tau\sigma}$, along with (9.13) and (9.12), to conclude that

$$\begin{aligned}
\|X^n - \tilde{X}^n\|_{l^2} &\leq \sqrt{\frac{\tau + \sigma}{1 - \tau\sigma L_A^2}}\|Z^n - \tilde{Z}^n\|_{\tau\sigma} \\
&\leq \sqrt{\frac{\tau + \sigma}{1 - \tau\sigma L_A^2}}\left[\|Z^{n-1} - \tilde{Z}^{n-1}\|_{\tau\sigma} + \theta\sqrt{\frac{\tau + \sigma}{\tau\sigma}}\Big(1 + \|w\|_{l^2} + 2\sigma\|A\|\|w\|_{l^2})\Big)\right] \\
&\leq n\theta(1 + \|w\|_{l^2} + 2\sigma\|A\|\|w\|_{l^2})\sqrt{\frac{\tau + \sigma}{1 - \tau\sigma L_A^2}}\sqrt{\frac{\tau + \sigma}{\tau\sigma}}.
\end{aligned}$$

It follows that (9.8) holds with $\psi_n(b, x_0) = X^n$ and the complex version of (9.15) implies (9.9). □

9.3. **Proof of Theorem 5.5. Step 1:** The first step is to derive a bound on the distance between vectors using the square-root LASSO objective function and rNSPL. For any inputs A (the rational approximations $\{A_l\}$), ρ and γ described in the theorem, we can compute, using Lemma 5.4, a positive integer l in finitely many arithmetic operations and comparisons, such that $A_l \in \mathbb{Q}[i]^{m\times N}$ satisfies the rNSPL with constants

$(1+\rho)/2 \in (0,1)$, $2\gamma > 0$. Lemmas 9.1 and 9.2 therefore imply that for any pair $z_1, z_2 \in \mathbb{C}^N$ we have

$$\|z_1 - z_2\|_{l^1_w} \leq \frac{3+\rho}{1-\rho}\left(2\sigma_{\mathbf{s},\mathbf{M}}(z_2)_{l^1_w} + \|z_1\|_{l^1_w} - \|z_2\|_{l^1_w}\right) + \frac{8\gamma\sqrt{\xi}}{1-\rho}\|A_l(z_1 - z_2)\|_{l^2}, \tag{9.16}$$

$$\|z_1 - z_2\|_{l^2} \leq \left(\frac{1+\rho}{2} + \frac{(3+\rho)\kappa^{1/4}}{4}\right)\frac{\|z_1 - z_2\|_{l^1_w}}{\sqrt{\xi}} + \left(2 + \kappa^{1/4}\right)\gamma\|A_l(z_1 - z_2)\|_{l^2}. \tag{9.17}$$

Combining these two inequalities, we obtain the bound

$$\begin{aligned}\|z_1 - z_2\|_{l^2} &\leq \frac{2C_1}{\sqrt{\xi}}\sigma_{\mathbf{s},\mathbf{M}}(z_2)_{l^1_w} + \frac{C_1}{\sqrt{\xi}}\left(\|z_1\|_{l^1_w} - \|z_2\|_{l^1_w}\right) + C_2\|A_l(z_1 - z_2)\|_{l^2} \\ &\leq \frac{2C_1}{\sqrt{\xi}}\sigma_{\mathbf{s},\mathbf{M}}(z_2)_{l^1_w} + 2C_2\|A_l z_2 - y\|_{l^2} + \frac{C_1}{\lambda\sqrt{\xi}}\left(\lambda\|z_1\|_{l^1_w} - \lambda\|z_2\|_{l^1_w} + \|A_l z_1 - y\|_{l^2} - \|A_l z_2 - y\|_{l^2}\right),\end{aligned} \tag{9.18}$$

where the second inequality follows from the fact that $\|A_l(z_1 - z_2)\|_{l^2} \leq \|A_l z_1 - y\|_{l^2} + \|A_l z_2 - y\|_{l^2}$ and we chose a positive rational $\lambda \leq C_1/(C_2\sqrt{\xi})$ (we will specify how small $|\lambda - C_1/(C_2\sqrt{\xi})|$ must be later, and always assume $\lambda \sim C_1/(C_2\sqrt{\xi})$). For notational convenience, we define

$$G(z_1, z_2, y) \coloneqq \lambda\|z_1\|_{l^1_w} - \lambda\|z_2\|_{l^1_w} + \|A_l z_1 - y\|_{l^2} - \|A_l z_2 - y\|_{l^2}, \tag{9.19}$$

the difference between the values of the objective function F_3^A for arguments z_1 and z_2. We also define

$$c(z, y) \coloneqq \frac{2C_1}{C_2\sqrt{\xi}} \cdot \sigma_{\mathbf{s},\mathbf{M}}(z)_{l^1_w} + 2\|A_l z - y\|_{l^2}. \tag{9.20}$$

It follows from (9.18) and $\lambda \leq C_1/(C_2\sqrt{\xi})$ that

$$\|z_1 - z_2\|_{l^2} \leq \frac{C_1}{\lambda\sqrt{\xi}}\left(c(z_2, y) + G(z_1, z_2, y)\right), \tag{9.21}$$

which also implies the bound $G(z_1, z_2, y) \geq -c(z_2, y)$. These bounds hold for general z_1, z_2 and y.

Step 2: We now apply Theorem 9.5 using a suitable scaling to define a family of parametrised NNs, which we iterate later in the proof (this corresponds to restarting primal-dual iterations with different parameters). Let $\sigma = \tau \in (4\|A_l\|^{-1}/5, 5\|A_l\|^{-1}/6)$ be positive rational numbers. We can compute such parameters by approximating $\|A_l\|$ via any standard algorithm that approximates the largest singular value of a rectangular matrix using finitely many arithmetic operations and comparisons. We now use Theorem 9.5 (with θ specified below) with input $y/(p\beta)$ and $x_0/(p\beta)$ for a given $p \in \mathbb{N}$, and $\beta \in \mathbb{Q}_{>0}$ (which we explicitly define below). Given $\phi^{A_l}_{p,\lambda}(y/(p\beta), x_0/(p\beta))$, Theorem 9.5 ensures the existence of a vector $\psi_p = \psi_p(y/(p\beta), x_0/(p\beta))$ satisfying

$$\left\|\psi_p\left(\tfrac{y}{p\beta}, \tfrac{x_0}{p\beta}\right) - \phi^A_{p,\lambda}\left(\tfrac{y}{p\beta}, \tfrac{x_0}{p\beta}\right)\right\|_{l^2} \leq pC\theta$$

where C is given in (9.7) and

$$\lambda\|\psi_p\|_{l^1_w} - \lambda\left\|\frac{x}{p\beta}\right\|_{l^1_w} + \left\|A\psi_p - \frac{y}{p\beta}\right\|_{l^2} - \frac{1}{p\beta}\|Ax - y\|_{l^2} \leq \frac{1}{p}\left(\frac{\|x(p\beta)^{-1} - x_0(p\beta)^{-1}\|^2_{l^2}}{\tau} + \frac{1}{\sigma}\right) \tag{9.22}$$

for any $x \in \mathbb{C}^N$ (and we have taken $\eta = 1$ in (9.9)). Define the map $H^\beta_p : \mathbb{C}^m \times \mathbb{C}^N \to \mathbb{C}^N$ by

$$H^\beta_p(y, x_0) = p\beta\phi^{A_l}_{p,\lambda}\left(\frac{y}{p\beta}, \frac{x_0}{p\beta}\right).$$

The additional scaling factors can be incorporated so that $H^\beta_p \in \mathcal{N}_{\mathbf{D}_p, 3p+1, 3}$. Rescaling (9.22) yields the existence of a vector $\hat{\psi}_p(y, x_0) \in \mathbb{C}^N$ (where the $\hat{\cdot}$ denotes an appropriate rescaling by multiplying by $p\beta$) such that

$$G\left(\hat{\psi}_p(y, x_0), x, y\right) \leq \frac{5}{4}\left(\frac{\|A_l\|}{p^2\beta}\|x - x_0\|^2_{l^2} + \|A_l\|\beta\right), \tag{9.23}$$

where we have used $\tau^{-1} = \sigma^{-1} \leq 5\|A_l\|/4$. Moreover, the constant C in Theorem 9.5 is bounded by

$$C = (1 + \|w\|_{l^2} + 2\sigma\|A_l\|\|w\|_{l^2}) \sqrt{\frac{\tau+\sigma}{1-\tau\sigma L_A^2}} \sqrt{\frac{\tau+\sigma}{\tau\sigma}} \leq \hat{C}_1(1 + \|w\|_{l^2}), \tag{9.24}$$

for a constant $\hat{C}_1$ that we can explicitly compute. Hence, upon rescaling (9.8), we arrive at

$$\left\| \hat{\psi}_p(y, x_0) - H_p^\beta(y, x_0) \right\|_{l^2} \leq p^2\theta\beta\hat{C}_1(1 + \|w\|_{l^2}).$$

Using Hölder's inequality, this also implies that

$$\left\| \hat{\psi}_p(y, x_0) - H_p^\beta(y, x_0) \right\|_{l^1_w} \leq p^2\theta\beta\hat{C}_1(1 + \|w\|_{l^2})\|w\|_{l^2}.$$

It follows from the reverse triangle inequality that

$$G\left(H_p^\beta(y, x_0), x, y\right) \leq G\left(\hat{\psi}_p(y, x_0), x, y\right) + p^2\theta\beta\hat{C}_1(1 + \|w\|_{l^2})\left(\|A_l\| + \lambda\|w\|_{l^2}\right). \tag{9.25}$$

Using this bound in (9.23), and the fact that $\lambda \lesssim (\gamma\sqrt{\xi})^{-1}$, we can choose $\theta \in \mathbb{Q}_{>0}$ such that

$$\theta^{-1} \lesssim p^2(1 + \|w\|_{l^2}) \max\left\{1, \frac{\lambda\|w\|_{l^2}}{\|A_l\|}\right\} \lesssim p^2(1 + \|w\|_{l^2}) \max\left\{1, \frac{\|w\|_{l^2}}{\|A\|\gamma\sqrt{\xi}}\right\},$$

and, simultaneously,

$$G\left(H_p^\beta(y, x_0), x, y\right) \leq \frac{4}{3}\left(\frac{\|A_l\|}{p^2\beta}\|x - x_0\|_{l^2}^2 + \|A_l\|\beta\right).$$

Combining this with (9.21), we obtain the key inequality

$$G\left(H_p^\beta(y, x_0), x, y\right) \leq \frac{4C_1^2\|A_l\|}{3p^2\beta\lambda^2\xi}\left[c(x, y) + G(x_0, x, y)\right]^2 + \frac{4}{3}\|A_l\|\beta. \tag{9.26}$$

Step 3: In this step, we specify the choice of p and β. So far, we have not used any information regarding the vectors x and y. Recall that for our recovery theorem, we restricted to pairs (x, y) such that

$$\frac{2C_1}{C_2\sqrt{\xi}} \cdot \sigma_{\mathbf{s},\mathbf{M}}(x)_{l^1_w} + 2\|Ax - y\|_{l^2} \leq \delta, \quad \|x\|_{l^2} \leq b_1, \quad \|y\|_{l^2} \leq b_2.$$

Using this, we can choose l larger if necessary such that for any such (x, y), we have the bound

$$c(x, y) \leq \frac{2C_1}{C_2\sqrt{\xi}} \cdot \sigma_{\mathbf{s},\mathbf{M}}(x)_{l^1_w} + 2\|Ax - y\|_{l^2} + 2\|A - A_l\|\|x\|_{l^2} \leq 2\delta.$$

The following lemma shows how to choose β and p to gain a decrease in G by a factor of $\upsilon \in (0, 1)$, up to small controllable error terms.

Lemma 9.6. *Let $\upsilon \in (0, 1) \cap \mathbb{Q}_{>0}$, $\epsilon_0 \in \mathbb{Q}_{>0}$ and choose $\beta \in \mathbb{Q}_{>0}$ such that $8\|A_l\|\beta = 3\upsilon_0\upsilon(\epsilon_0 + 2\delta)$ for some $\upsilon_0 \in [1, 2)$. Then for any x_0 with $G(x_0, x, y) \leq \epsilon_0$ and positive integer $p \geq \left\lceil \frac{8C_1\|A_l\|}{3\upsilon\lambda\sqrt{\xi}\sqrt{(2-\upsilon_0)\upsilon_0}} \right\rceil$ the following bound holds*

$$G\left(H_p^\beta(y, x_0), x, y\right) \leq \upsilon\left(2\delta + \epsilon_0\right). \tag{9.27}$$

Proof. The choice of β ensures that $\frac{4}{3}\|A_l\|\beta \leq \frac{\upsilon_0\upsilon}{2}(2\delta + \epsilon_0)$. Using (9.26), and the fact that $0 \leq c(x, y) + G(x_0, x, y) \leq 2\delta + \epsilon_0$, the bound (9.27) therefore holds if

$$\frac{32C_1^2\|A_l\|^2}{9p^2\upsilon_0\upsilon\lambda^2\xi}\left(2\delta + \epsilon_0\right) \leq \frac{(2 - \upsilon_0)\upsilon}{2}(2\delta + \epsilon_0).$$

Rearranging and taking the square root gives the result, where the ceiling function ensures p is an integer. □

We denote the choice of β in Lemma 9.6 by $\beta(\upsilon, \epsilon_0)$. Since $8/3 < 3$, we can, by taking l larger and by making λ closer to $C_1/(C_2\sqrt{\xi})$ if necessary, and through an appropriate choice of υ_0, ensure that we can compute (using finitely many arithmetic operations and comparisons) a choice $p(\upsilon) \leq \left\lceil \frac{3C_2\|A\|}{\upsilon} \right\rceil$ such that the conclusion of the lemma holds.

Step 4: We are now ready to construct our NNs. Note first that $G(0,x,y) \leq \|y\|_{l^2} \leq b_2$, for any y in our desired input. Given $n \in \mathbb{N}$, we set $\epsilon_0 = b_2$ and for $j = 2, ..., n$ set $\epsilon_j = \upsilon\left(2\delta + \epsilon_{j-1}\right)$. By summing a geometric series, this implies $\epsilon_n \leq \frac{2\upsilon\delta}{1-\upsilon} + \upsilon^n b_2$. We define $\phi_n(y)$ iteratively as follows. We set $\phi_1(y) = H_{p(\upsilon)}^{\beta(\upsilon,\epsilon_0)}(y,0)$ and for $j = 2, ..., n$ we set $\phi_j(y) = H_{p(\upsilon)}^{\beta(\upsilon,\epsilon_{j-1})}(y, \phi_{j-1}(y))$. Clearly this algorithmically constructs a NN ϕ_n. We can concatenate (by combining affine maps) the NNs corresponding to the H_p^β maps to see that $\phi_n \in \mathcal{N}_{\mathbf{D}_{(n,p)},3np+1,3}$. Moreover, Lemma 9.6 implies the bound $G\left(\phi_n(y),x,y\right) \leq \epsilon_n \leq \frac{2\upsilon\delta}{1-\upsilon} + \upsilon^n b_2$. Combining this with (9.18),

$$\|\phi_n(y) - x\|_{l^2} \leq \frac{2C_1}{\sqrt{\xi}}\sigma_{\mathbf{s},\mathbf{M}}(x)_{l_w^1} + 2C_2\|Ax - y\|_{l^2} + 2C_2\|A - A_l\|_{l^2}b_1 + \frac{C_1}{\lambda\sqrt{\xi}}\left(\frac{2\upsilon\delta}{1-\upsilon} + \upsilon^n b_2\right), \quad (9.28)$$

Again, we can apriori choose l and λ to ensure that

$$2C_2\|A - A_l\|_{l^2}b_1 + \frac{C_1}{\lambda\sqrt{\xi}}\left(\frac{2\upsilon\delta}{1-\upsilon} + \upsilon^n b_2\right) \leq C_2\left(\frac{2\upsilon\delta}{1-\upsilon} + \delta + \upsilon^n b_2\right).$$

Applying this bound to (9.28) yields (5.3).

Finally, we argue for the error in the weighted l_w^1-norm. Note that since $\rho < 1$, the choice of λ ensures that $\frac{8\gamma\sqrt{\xi}}{1-\rho} < \frac{3+\rho}{1-\rho}\frac{1}{\lambda}$. It follows from (9.16), using the same argument for the l^2 case, that

$$\|\phi_n(y) - x\|_{l_w^1} \leq \frac{3+\rho}{1-\rho}\left(2\sigma_{\mathbf{s},\mathbf{M}}(x)_{l_w^1} + \frac{2}{\lambda}\|A_l x - y\|_{l^2} + \frac{1}{\lambda}G\left(\phi_n(y),x,y\right)\right), \quad (9.29)$$

Again, we can apriori adjust l and λ as necessary to obtain the bound

$$\|\phi_n(y) - x\|_{l_w^1} \leq \frac{3+\rho}{1-\rho}\left(2\sigma_{\mathbf{s},\mathbf{M}}(x)_{l_w^1} + \frac{2C_2\sqrt{\xi}}{C_1}\|Ax - y\|_{l^2} + \frac{C_2\sqrt{\xi}}{C_1}\left(\frac{2\upsilon\delta}{1-\upsilon} + \upsilon^n b_2\right) + \delta\frac{C_2\sqrt{\xi}}{C_1}\right),$$

where the final term in brackets corresponds to this final approximation. Simplifying this yields (5.4). □

To end this section, we provide a brief proof sketch of the bounds in Remark 5.7. The argument is similar to the proof of Theorem 5.5. We set $\widehat{\phi}_n(y,x_0) = \beta\phi_{n,\lambda}^{A_l}\left(\frac{y}{\beta}, \frac{x_0}{\beta}\right)$, and the arguments in Theorem 5.5 show that we can choose τ, σ, l and θ_n with $\theta_n^{-1} = \mathcal{O}(n^2)$ such that for any $x, x_0 \in \mathbb{C}^N$ and $y \in \mathbb{C}^m$,

$$G\left(\widehat{\phi}_n(y,x_0), x, y\right) \leq \frac{3}{2}\frac{\|A\|}{n}\left(\frac{\|x - x_0\|_{l^2}^2}{\beta} + \beta\right).$$

If $\|x\|_{l^2} \leq b_1$, then we can choose l such that $2\|A - A_l\|b_2 \leq \|A\|\beta/(2n)\min\{C_1/(C_2\lambda\sqrt{\xi}), 1\}$ and hence (5.5) follows from (9.18). Similarly, we can use the corresponding bound (9.29) to show (5.6).

10. Proof of Theorem 5.10

For the benefit of the reader, we first recall the orthonormal bases used. We then provide coherence estimates which are used to obtain bounds on the number of samples needed, and end this section with the proof of Theorem 5.10. It will be convenient to sometimes enumerate the vector or tensor elements starting from 0, or negative numbers. That is for $x \in \mathbb{C}^N$ with $d = 1$ we might denote its elements as $x = (x(0), \ldots, x(N-1))$, or $x = (x(-N/2+1), \ldots, x(N/2))$ and for $d > 1$ its $\boldsymbol{k} = (k_1, \ldots, k_d)$'th element is written as $x(\boldsymbol{k})$. It will always be clear from the context, which range of indices we consider. Furthermore, recall from §5.3 that we let $N = K^d$ and $K = 2^r$ for $r \in \mathbb{Z}_{\geq 0}$. This is assumed throughout this section.

10.1. **Setup: the relevant orthonormal bases.**

Discrete Fourier transform. For a d-dimensional signal $x = \{x(\boldsymbol{t})\}_{t_1,\ldots,t_d=0}^{K-1} \in \mathbb{C}^{K\times\cdots\times K}$ we denote its Fourier transform by $[\mathcal{F}x](\boldsymbol{\omega}) = \frac{1}{N^{1/2}}\sum_{t_1,\ldots,t_d=0}^{K-1} x(\boldsymbol{t})\exp(\frac{2\pi i\boldsymbol{\omega}\cdot\boldsymbol{t}}{K}), \quad \boldsymbol{\omega} \in \mathbb{R}^d$. For discrete computations, it is customary to consider this transform at the integers $\boldsymbol{\omega} \in \{-K/2+1, \ldots, K/2\}^d$ and let $F^{(d)} \in \mathbb{C}^{K^d\times K^d}$

denote the corresponding matrix so that $F^{(d)}\mathrm{vec}(x) = \{[\mathcal{F}x](\boldsymbol{\omega})\}_{\boldsymbol{\omega}\in\{-K/2+1,\dots,K/2\}^d}$ for a suitable vectorisation $\mathrm{vec}(x)$ of x and ordering of the $\boldsymbol{\omega}$'s. Let

$$\vartheta_{\boldsymbol{\omega}} = \left\{N^{-1/2}\exp\left(-2\pi i K^{-1}\boldsymbol{\omega}\cdot\boldsymbol{t}\right) : \boldsymbol{t}\in\{0,\dots,K-1\}^d\right\} \subset \mathbb{C}^{K\times\cdots\times K}.$$

Then

$$\left\{\mathrm{vec}(\vartheta_{\boldsymbol{\omega}}) : \boldsymbol{\omega}\in\{-K/2+1,\dots,K/2\}^d\right\} \tag{10.1}$$

is an orthonormal basis for $\mathbb{C}^{K^d} = \mathbb{C}^N$. Furthermore, recall from §5.3, that we divide the different frequencies into dyadic bands. For $d = 1$ we let $B_1 = \{0, 1\}$ and

$$B_k = \left\{-2^{k-1}+1, ..., -2^{k-2}\right\} \cup \left\{2^{k-2}+1, ..., 2^{k-1}\right\}, \quad k = 2, ..., r.$$

In the general d-dimensional case we set $B_{\mathbf{k}}^{(d)} = B_{k_1}\times ... \times B_{k_d}$ for $\mathbf{k} = (k_1, ..., k_d) \in \mathbb{N}^d$.

Walsh transform.

Definition 10.1. *The Walsh functions $v_n\colon [0,1) \to \{+1, -1\}$ are defined by*

$$v_\omega(z) = (-1)^{\sum_{j=1}^{\infty}(\omega^{(j)}+\omega^{(j+1)})z^{(j)}}, \quad z\in[0,1), \quad \omega\in\mathbb{Z}_{\geq 0}, \tag{10.2}$$

where $(z^{(i)})_{i\in\mathbb{N}}$ denotes the binary expansion of z (terminating if z is a dyadic rational) and we write $\omega = \sum_{j=1}^{\infty}\omega^{(j)}2^{j-1}$ for $\omega^{(j)}\in\{0,1\}$. For $\boldsymbol{z}\in[0,1)^d$ and $\boldsymbol{\omega}\in\mathbb{Z}_{\geq 0}^d$, we let $v_{\boldsymbol{\omega}}(\boldsymbol{z}) = v_{\omega_1}(z_1)\cdots v_{\omega_d}(z_d)$.

For $x\in\mathbb{C}^{K\times\cdots\times K}$ and $K = 2^r$ we let its d-dimensional Walsh transform be denoted by

$$[\mathcal{W}x](\boldsymbol{\omega}) = \frac{1}{N^{1/2}}\sum_{t_1,\dots,t_d=0}^{K-1} x(\boldsymbol{t})v_{\boldsymbol{\omega}}(\boldsymbol{t}/K), \quad \boldsymbol{\omega}\in\{0,\dots,2^r-1\}^d.$$

As in the Fourier case, we let $W^{(d)}\in\mathbb{C}^{N\times N}$ so that $W^{(d)}\mathrm{vec}(x) = \{[\mathcal{W}x](\boldsymbol{\omega})\}_{\boldsymbol{\omega}\in\{0,\dots,K-1\}^d}$ for a suitable vectorisation of x and ordering of the $\boldsymbol{\omega}$'s. We let $\varrho_{\boldsymbol{\omega}} = \{N^{-1/2}v_{\boldsymbol{\omega}}(\boldsymbol{t}/K) : \boldsymbol{t}\in\{0,\dots,K-1\}^d\}\subset\mathbb{C}^{K\times\cdots\times K}$ and note that

$$\left\{\mathrm{vec}(\varrho_{\boldsymbol{\omega}}) : \boldsymbol{\omega}\in\{0,\dots,K-1\}^d\right\} \tag{10.3}$$

is an orthonormal basis for $\mathbb{C}^N$. As in the Fourier case we recall the frequency bands introduced in §5.3. Let $B_1 = \{0, 1\}$ and $B_k = \{2^{k-1}, \dots, 2^k - 1\}$ for $k = 2, \dots, r$ in the one-dimensional case, and $B_{\mathbf{k}}^{(d)} = B_{k_1}\times ... \times B_{k_d}$, $\mathbf{k} = (k_1, ..., k_d)\in\mathbb{N}^d$. Whether the notation refers to the Walsh or Fourier frequency bands will always be clear from the context.

Haar-wavelet transform. On $\mathbb{C}^K$ the Haar wavelet vectors are defined as

$$\psi_{j,p}(i) = \begin{cases} 2^{\frac{j-r}{2}}, & p2^{r-j} \leq i < \left(p+\frac{1}{2}\right)2^{r-j} \\ -2^{\frac{j-r}{2}}, & \left(p+\frac{1}{2}\right)2^{r-j} \leq i < (p+1)2^{r-j} \\ 0, & \text{otherwise,} \end{cases}$$

for $j = 0, ..., r-1$ and $p = 0, ..., 2^j - 1$, and we can define the corresponding scaling vectors as $\varphi_{j,p}(i) = |\psi_{j,p}(i)|$. To simplify the notation we set $\psi_{j,k}^{(0)} = \varphi_{j,k}$ and $\psi_{j,k}^{(1)} = \psi_{j,k}$. For $d > 1$ and $\boldsymbol{q} = (q_1, ..., q_d)\in\{0,1\}^d$, $\boldsymbol{p} = (p_1, \dots, p_d)\in\mathbb{Z}_{\geq 0}^d$ define the tensor product $\psi_{j,\boldsymbol{p}}^{\boldsymbol{q}} = \psi_{j,p_1}^{(q_1)}\otimes ... \otimes \psi_{j,p_d}^{(q_d)}$. Splitting these tensors by scale

$$C_1 = \{\mathrm{vec}(\psi_{0,0}^{\boldsymbol{q}}) : \boldsymbol{q}\in\{0,1\}^d\}, \quad C_j = \{\mathrm{vec}(\psi_{j-1,\boldsymbol{p}}^{\boldsymbol{q}}) : \boldsymbol{q}\in\{0,1\}^d\backslash\{0\}, p_k = 0, ..., 2^{j-1}-1\},$$

for $j = 2, ..., r$, we get that $C_1\cup\cdots\cup C_r$ is an orthonormal basis for $\mathbb{C}^N$. Next, let the vectors in $C_1\cup\cdots\cup C_r$, form the rows of a matrix $\Phi\in\mathbb{C}^{N\times N}$. The matrix Ψ is called the *discrete wavelet transform* (DWT) matrix, and its inverse Ψ^{-1} is called the *inverse discrete wavelet transform* (IDWT) matrix. Notice that since $C_1\cup\cdots\cup C_r$ is an orthonormal basis, we have the relation $\Psi^{-1} = \Psi^*$.

10.2. **Uniform recovery guarantees and coherence estimates.** We express $U = \left[U^{(\mathbf{k},j)}\right]_{\mathbf{k}=1,j=1}^{\|\mathbf{k}\|_{l^\infty}\leq r,r}$ in block form, where the entries in each $U^{\mathbf{k},j}$ consist of the inner products $\langle \varphi, \rho_{\boldsymbol{\omega}} \rangle$ for $\varphi \in C_j$ and where $\rho_{\boldsymbol{\omega}}$ is an element in either (10.1) or (10.3) with $\boldsymbol{\omega} \in B_{\mathbf{k}}^{(d)}$, depending on whether we consider Fourier or Walsh sampling. For this decomposition we define local coherence as follows.

Definition 10.2. *Let $U = \left[U^{(\mathbf{k},j)}\right]_{\mathbf{k}=1,j=1}^{\|\mathbf{k}\|_{l^\infty}\leq r,r}$ be defined as above. Then the $(\mathbf{k},j)$th local coherence of U is*

$$\mu(U^{\mathbf{k},j}) = \left|B_{\mathbf{k}}^{(d)}\right| \max_{p,q} |(U^{\mathbf{k},j})_{pq}|^2, \quad \textit{where } \left|B_{\mathbf{k}}^{(d)}\right| \textit{ is the cardinality of } B_{\mathbf{k}}^{(d)}.$$

Recall from Definition 3.1, that for an $(\mathbf{s},\mathbf{M})$-sparse vector, $s = s_1 + \ldots + s_r$ denotes the total sparsity. Furthermore, $m = \sum_{\mathbf{k}=1}^{\|\mathbf{k}\|_{l^\infty}\leq r} m_{\mathbf{k}}$ denotes the total number of samples in an $(\mathbf{N},\mathbf{m})$-multilevel subsampling scheme. The following shows that to use Theorem 5.5, we need to bound the local coherences of U.

Proposition 10.3 ([1]). *Let $\epsilon_{\mathbb{P}} \in (0,1)$, $(\mathbf{s},\mathbf{M})$ be local sparsities and sparsity levels respectively with $2 \leq s \leq N$, and consider the $(\mathbf{N},\mathbf{m})$-multilevel subsampling scheme to form a subsampled unitary matrix A as in Definitions 5.8 and 5.9. Let*

$$t_j = \min\left\{ \left\lceil \frac{\xi(\mathbf{s},\mathbf{M},w)}{w_{(j)}^2} \right\rceil, M_j - M_{j-1} \right\}, \quad j = 1,...,r, \tag{10.4}$$

and suppose that

$$m_k \gtrsim \mathcal{L}' \cdot \sum_{j=1}^{r} t_j \mu(U^{k,j}), \quad k = 1,...,l \tag{10.5}$$

where $\mathcal{L}' = r \cdot \log(2m) \cdot \log^2(t) \cdot \log(N) + \log(\epsilon_{\mathbb{P}}^{-1})$. Then with probability at least $1-\epsilon_{\mathbb{P}}$, A satisfies the weighted rNSPL of order $(\mathbf{s},\mathbf{M})$ with constants $\rho = 1/2$ and $\gamma = \sqrt{2}$.

The following bound the local coherences of U, with $\mathcal{M}_{\mathcal{F}}(\mathbf{s},\mathbf{k})$ and $\mathcal{M}_{\mathcal{W}}(\mathbf{s},\mathbf{k})$ defined in (5.7) and (5.8).

Lemma 10.4 (**Coherence bound for Fourier case**). *Consider the d-dimensional Fourier–Haar–wavelet matrix with blocks $U^{\mathbf{k},j}$, then the local coherences satisfy*

$$\mu(U^{\mathbf{k},j}) \lesssim 2^{-2(j-\|\boldsymbol{k}\|_{l^\infty})_+} \prod_{i=1}^{d} 2^{-|k_i-j|}, \tag{10.6}$$

where for $t \in \mathbb{R}$, $t_+ = \max\{0,t\}$. It follows that

$$\sum_{j=1}^{r} s_j \mu(U^{\mathbf{k},j}) \lesssim \sum_{j=1}^{\|\mathbf{k}\|_{l^\infty}} s_j \prod_{i=1}^{d} 2^{-|k_i-j|} + \sum_{j=\|\mathbf{k}\|_{l^\infty}+1}^{r} s_j 2^{-2(j-\|\mathbf{k}\|_{l^\infty})} \prod_{i=1}^{d} 2^{-|k_i-j|} = \mathcal{M}_{\mathcal{F}}(\mathbf{s},\mathbf{k}). \tag{10.7}$$

Proof. From the one-dimensional case treated in [6, See proof of Lem. 1], we have

$$\left| \left[\mathcal{F}\psi_{j,p}^{(1)}\right](\omega) \right|^2 \lesssim \begin{cases} 2^{-k}2^{-|k-j|}, & \text{if } j \leq k, \\ 2^{-k}2^{-3|k-j|}, & \text{otherwise} \end{cases},$$

We proceed by showing that $|[\mathcal{F}\psi_{j,p}^{(0)}](\omega)|^2 \lesssim 2^{-k}2^{-|k-j|}$ in the one-dimensional case, before considering d dimensions. Let $\omega \neq 0$ correspond to a frequency in B_k, $j \in \{0,...,r-1\}$ and $p \in \{0,...,2^j-1\}$. Then

$$\left[\mathcal{F}\psi_{j,p}^{(0)}\right](\omega) = 2^{\frac{j}{2}-r} e^{2\pi i \omega p 2^{-j}} \sum_{t=0}^{2^{r-j}-1} e^{2\pi i \omega t 2^{-r}} = 2^{\frac{j}{2}-r} e^{2^{1-j}\pi i \omega p} \frac{1-e^{2\pi i \omega 2^{-j}}}{1-e^{2\pi i \omega 2^{-r}}}.$$

A simple application of the double angle formula then yields

$$\left| \left[\mathcal{F}\psi_{j,p}^{(0)}\right](\omega) \right| \lesssim 2^{\frac{j}{2}-r} \frac{\left|\sin(\pi\omega 2^{-j})\right|}{\left|\sin(\pi\omega 2^{-r})\right|} = \frac{2^{\frac{j}{2}}}{|\omega|} \frac{|\omega 2^{-r}|}{\left|\sin(\pi\omega 2^{-r})\right|} \left|\sin(\pi\omega 2^{-j})\right| \lesssim 2^{\frac{j}{2}-k} \left|\sin(\pi\omega 2^{-j})\right|,$$

where the second inequality follows from $|\omega 2^{-r}| \leq 1/2$ and $2^k \lesssim |\omega|$. If $k > j$, this implies $|[\mathcal{F}\psi_{j,p}^{(0)}](\omega)|^2 \lesssim 2^{-k}2^{-|k-j|}$. If $k \leq j$, we use that $|\sin(\pi t)| \leq \pi|t|,\ \forall t \in \mathbb{R}$ to get $\left|\sin(\pi\omega 2^{-j})\right| \lesssim 2^{k-j}$. Hence,

$$\left|\left[\mathcal{F}\psi_{j,p}^{(0)}\right](\omega)\right|^2 \lesssim 2^{j-2k}2^{2k-2j} = 2^{-j} = 2^{-k}2^{-|k-j|}.$$

If $\omega = 0$ then by definition we have $|[\mathcal{F}\psi_{j,p}^{(0)}](\omega)|^2 \lesssim 2^{-j} = 2^{-k}2^{-|k-j|}$ and hence this bound still holds.

We now consider the general d-dimensional case. The above computations give that

$$\mu(U^{(\mathbf{k},1)}) \lesssim 2^{\sum_{i=1}^d k_i} \max_{\boldsymbol{q}\in\{0,1\}^d} \prod_{i=1}^d \max_{w\in B_{k_i}} \left|\left[\mathcal{F}\psi_{0,0}^{(q_i)}\right](\omega)\right|^2 \lesssim \prod_{i=1}^d 2^{-|k_i-1|}.$$

Similarly for $j > 1$

$$\mu(U^{(\mathbf{k},j)}) \lesssim 2^{\sum_{i=1}^d k_i} \max_{\boldsymbol{q}\in\{0,1\}^d\setminus\{0\}} \prod_{i=1}^d \max_{w\in B_{k_i}} \max_{p_i\in\{0,\dots,2^{j-1}-1\}} \left|\left[\mathcal{F}\psi_{j-1,p_i}^{(q_i)}\right](\omega)\right|^2 \lesssim \max_{\boldsymbol{q}\in\{0,1\}^d\setminus\{0\}} \prod_{i=1}^d 2^{-|k_i-j|-2q_i(j-k_i)_+}.$$

The maximum value of this estimate is obtained when the non-zero component of $\boldsymbol{q}$ corresponds to the maximum value of k_i. This gives precisely (10.6). □

Before proceeding with the Walsh–Haar–wavelet case, we recall the following lemma [8].

Lemma 10.5. *Let ω and $j \geq 0$ be integers so that $2^j \leq \omega < 2^{j+1}$ and let $\Delta_k^j = [k2^{-j}, (k+1)2^{-j})$ for $k \in \mathbb{Z}_{\geq 0}$. Then v_ω is constant on each of the intervals Δ_k^{j+1}, $k \in \{0, \dots, 2^{j+1}-1\}$. Each of the intervals Δ_k^j can be decomposed into the intervals Δ_{2k}^{j+1} and Δ_{2k+1}^{j+1}, where v_ω is equal to 1 on exactly one of them and equal to -1 on the other. When $\omega = 0$, we have $v_\omega \equiv 1$.*

Lemma 10.6 (Coherence bound for Walsh case). *Consider the d-dimensional Walsh–Haar–wavelet matrix with blocks $U^{(\mathbf{k},j)}$, then the local coherences satisfy*

$$\mu(U^{(\mathbf{k},j)}) \lesssim \begin{cases} \prod_{i=1}^d 2^{-|k_i-j|} & \text{if } k_i \leq j \text{ for } i = 1, ..., d \text{ with at least one equality,} \\ 0 & \text{otherwise} \end{cases}. \tag{10.8}$$

It follows that

$$\sum_{j=1}^r s_j \mu(U^{(\mathbf{k},j)}) \lesssim s_{\|\mathbf{k}\|_{l^\infty}} \prod_{i=1}^d 2^{-|k_i - \|\mathbf{k}\|_{l^\infty}|} = \mathcal{M}_{\mathcal{W}}(\mathbf{s}, \mathbf{k}). \tag{10.9}$$

Proof. We begin with some computations in the one-dimensional case. Let $I_{j,p} = \{p2^{r-j}, \dots, (p+1)2^{r-j} - 1\}$. We recall that that $\text{supp}(\psi_{j,p}^{(0)}) = \text{supp}(\psi_{j,p}^{(1)}) = I_{j,p}$. Using Lemma 10.5 it is clear that for $2^m \leq \omega < 2^{m+1}$, ϱ_ω is constant on $I_{m+1,k}$, for $k \in \{0, \dots, 2^{m+1}-1\}$ and that for any pair $I_{m+1,2t}$, $I_{m+1,2t+1}$, ϱ_ω changes sign. For $\omega = 0$, we have that ϱ_ω is all constant. Keeping track of the supports gives the relations

$$\left|\left\langle \psi_{j,p}^{(0)}, \varrho_\omega \right\rangle\right| = \begin{cases} 2^{-j/2} & \text{if } \omega < 2^j \\ 0 & \text{otherwise} \end{cases}, \quad \text{and} \quad \left|\left\langle \psi_{j,p}^{(1)}, \varrho_\omega \right\rangle\right| = \begin{cases} 2^{-j/2} & \text{if } 2^j \leq \omega < 2^{j+1} \\ 0 & \text{otherwise} \end{cases}.$$

In particular, we can rewrite this as $|\langle \psi_{j,p}^{(0)}, \varrho_\omega \rangle| = 2^{-j/2} = 2^{-k/2}2^{-(j-k)/2}$ if $\omega \in B_k$, $k \leq j$ and 0 otherwise, and note that $|\langle \psi_{j,p}^{(1)}, \varrho_\omega \rangle| = 2^{-j/2}$ if $\omega \in B_{j+1}$ and 0 otherwise.

Turning to the general d-dimensional case. The above computations immediately give that $\mu(U^{(\mathbf{k},1)}) \lesssim \prod_{i=1}^d \delta_{k_i,1}$, where $\delta_{i,j}$ is the Kronecker-delta. Similarly for $j > 1$

$$\begin{aligned}\mu(U^{(\mathbf{k},j)}) &= \left|B_{\mathbf{k}}^{(d)}\right| \max_{\boldsymbol{q}\in\{0,1\}^d\setminus\{0\}} \prod_{i=1}^d \max_{\omega_i\in B_{k_i}} \max_{p_i\in\{0,\dots,2^{j-1}-1\}} \left|\langle \varrho_{\omega_i}, \psi_{j-1,p_i}^{(q_i)}\rangle\right|^2 \\ &\lesssim 2^{\sum_{i=1}^d k_i} \max_{\boldsymbol{q}\in\{0,1\}^d\setminus\{0\}} \prod_{i=1}^d \left(\delta_{q_i,0}\delta_{k_i<j} 2^{-k_i-|k_i-j|} + \delta_{q_i,1}\delta_{k_i,j} 2^{-k_i}\right) \\ &\lesssim \max_{\boldsymbol{q}\in\{0,1\}^d\setminus\{0\}} \prod_{i=1}^d \left(\delta_{q_i,0}\delta_{k_i<j} 2^{-|k_i-j|} + \delta_{q_i,1}\delta_{k_i,j}\right).\end{aligned}$$

This estimate is zero unless $k_i \leq j$ and at least one of the k_i is equal to j. In this case the maximum corresponds to $q_i = 1$ if $k_i = j$ and $q_i = 0$ otherwise. This gives precisely (10.8). □

10.3. **Proof of Theorem 5.10.** For the benefit of the reader, we recall that $A = P_{\mathcal{I}} DV\Psi$. We apply Proposition 10.3, noting that the t_j in (10.4) satisfy

$$t_j \lesssim \frac{\xi(\mathbf{s},\mathbf{M},w)}{w_{(j)}^2} \leq s_j \cdot \kappa(\mathbf{s},\mathbf{M},w), \qquad t \lesssim s\cdot\kappa(\mathbf{s},\mathbf{M},w). \tag{10.10}$$

Therefore $\sum_{j=1}^r t_j \mu(U^{\mathbf{k},j}) \lesssim \kappa(\mathbf{s},\mathbf{M},w) \sum_{j=1}^r s_j \mu(U^{\mathbf{k},j})$. Combining with (10.10), note that (10.5) holds if

$$m_{\mathbf{k}} \gtrsim \kappa(\mathbf{s},\mathbf{M},w) \cdot \left(\sum_{j=1}^r s_j \mu(U^{\mathbf{k},j})\right) \cdot \mathcal{L}, \quad \text{where} \tag{10.11}$$

$$\mathcal{L} = \frac{r\cdot\log(2m)}{\log(2)} \cdot \log^2\left(s\cdot\kappa(\mathbf{s},\mathbf{M},w)\right)\cdot\log(N) + \log(\epsilon_{\mathbb{P}}^{-1}) = d\cdot r^2\cdot\log(2m)\cdot\log^2\left(s\cdot\kappa(\mathbf{s},\mathbf{M},w)\right) + \log(\epsilon_{\mathbb{P}}^{-1}),$$

since $N = 2^{r\cdot d}$. In the Fourier sampling case, by Lemma 10.4, (10.11) holds if (5.9) holds. Similarly, in the Walsh sampling case, by Lemma 10.6, (10.11) holds if (5.10) holds. By Proposition 10.3, with probability at least $1-\epsilon_{\mathbb{P}}$, A satisfies the weighted rNSPL of order $(\mathbf{s},\mathbf{M})$ with constants $\rho = 1/2$ and $\gamma = \sqrt{2}$. The conclusion of Theorem 5.5 then holds for the uniform recovery of the Haar wavelet coefficients $x = \Psi c \in \mathbb{C}^N$.

For the final part, we use Theorem 5.5. The only difference is that we have to compose the NNs with (an approximation of) the matrix Ψ^* to recover approximations of c from approximations of $x = \Psi c$. Recall that

$$\mathcal{Z} = \max\left\{1, \frac{\max_{j=1,\dots,r} w_{(j)}\sqrt{(M_j - M_{j-1})}}{\sqrt{\xi(\mathbf{s},\mathbf{M},w)}}\right\}$$

and set $n_0 = \left\lceil \log\left(\delta^{-1}\mathcal{Z}\right)\kappa^{1/4}\mathcal{Z}\right\rceil$. Let p be as in Theorem 5.5 and let $n_1 \in \mathbb{Z}_{\geq 0}$ such that $n_0 = n_1 p + n_2$ for $n_1 \in \{0,\dots,p-1\}$ (the n from the statement of the theorem corresponds to $n_1 p$). Set $\phi(y) = \Psi^*\left[\phi_{n_1}(y,0)\right]$, where ϕ_{n_1} denotes the NN from Theorem 5.5 with $b_1 = 1$, $b_2 = \|A\| + \delta$ and $\upsilon = e^{-1}$. Strictly speaking, we need to approximate $\|A\|$ and e^{-1}, and also apply a rational approximation of the matrix Ψ^* instead of Ψ^*, but we have avoided this extra notational clutter (the associated approximation errors can be made smaller than $\kappa^{1/4}\delta$ since the vectors we apply the matrix to are uniformly bounded). Now suppose that $y = P_{\mathcal{I}} DVc + e \in \mathcal{J}(\delta,\mathbf{s},\mathbf{M},w)$, and notice that for $\|c\|_{l^2} \leq 1$ we have that $\|y\|_{l^2} \leq \|A\| + \|e\|_{l^2} \leq b_2$ since Ψ is an isometry. Then, since $C_1, C_2 \sim \kappa^{1/4}$ (using that $\kappa \geq 1$), (5.3) implies that

$$\|\phi(y) - c\|_{l^2} = \|\phi_{n_0}(y,0) - \Psi c\|_{l^2} \lesssim \kappa^{1/4}\delta + b_2\kappa^{1/4}e^{-n_1}.$$

The theorem follows if we can prove that $b_2 e^{-n_1} \lesssim \delta$.

Let $\mathbf{t}$ be as in (10.4) and let $\Delta_1, \Delta_2, \dots$ be a partition of $\{1,\dots,N\}$ such that each support set is $(\mathbf{t},\mathbf{M})$-sparse. We can choose such as partition with at most

$$\max_{j=1,\dots,r}\left\lceil\frac{M_j - M_{j-1}}{t_j}\right\rceil \lesssim \max_{j=1,\dots,r}\left\lceil\frac{M_j - M_{j-1}}{\min\{\xi(\mathbf{s},\mathbf{M},w)/w_{(j)}^2, M_j - M_{j-1}\}}\right\rceil$$

sets. The proof of Proposition 10.3 shows that A satisfies the RIPL of order $(\mathbf{t}, \mathbf{M})$ and hence for any $x \in \mathbb{C}^N$,

$$\|Ax\|_{l^2} \leq \sum_i \|A(x_{\Delta_i})\|_{l^2} \lesssim \sum_i \|x_{\Delta_i}\|_{l^2} \lesssim \max_{j=1,\dots,r} \sqrt{\left\lceil \frac{M_j - M_{j-1}}{\min\{\xi(\mathbf{s}, \mathbf{M}, w)/w_{(j)}^2, M_j - M_{j-1}\}} \right\rceil} \|x\|_{l^2},$$

where we have used Hölder's inequality in the last step. It follows that $\|A\| \lesssim \mathcal{Z}$ and hence that $p \lesssim \kappa^{1/4}\mathcal{Z}$ and $b_2 \lesssim \mathcal{Z}$. This implies that $n_1 \gtrsim \log\left(\delta^{-1}\mathcal{Z}\right)$ and $b_2 e^{-n_1} \lesssim \delta$, completing the proof. □

ACKNOWLEDGMENTS

An unrolled version of Chambolle and Pock's primal-dual algorithm [53, Ch. 15.3] was developed in TensorFlow by Krisitian M. Haug, prior to this work. We are grateful to Kristian, for sharing his code and allowing us to use parts of in when implementing the FIRENETs. We would also like to thank Ben Adcock and Alex Townsend for useful discussions. This work was supported by a Research Fellowship at Trinity College, Cambridge (M.J.C.), and a Leverhulme Prize and a Royal Society University Research Fellowship (A.C.H.).

REFERENCES

[1] B. Adcock, V. Antun, and A. C. Hansen. Uniform recovery in infinite-dimensional compressed sensing and applications to structured binary sampling. *arXiv preprint arXiv:1905.00126*, 2019.

[2] B. Adcock, C. Boyer, and S. Brugiapaglia. On oracle-type local recovery guarantees in compressed sensing. *Inf. Inference*, 2018.

[3] B. Adcock, S. Brugiapaglia, and M. King-Roskamp. Do log factors matter? On optimal wavelet approximation and the foundations of compressed sensing. *arXiv preprint arXiv:1905.10028*, 2019.

[4] B. Adcock and A. C. Hansen. Generalized sampling and infinite-dimensional compressed sensing. *Found. Comput. Math.*, 16(5):1263–1323, 2016.

[5] B. Adcock, A. C. Hansen, C. Poon, and B. Roman. Breaking the coherence barrier: A new theory for compressed sensing. In *Forum of Mathematics, Sigma*, volume 5. Cambridge University Press, 2017.

[6] B. Adcock, A. C. Hansen, and B. Roman. A note on compressed sensing of structured sparse wavelet coefficients from subsampled Fourier measurements. *IEEE Signal Process Lett.*, 23(5):732–736, 2016.

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

[8] V. Antun. Coherence estimates between Hadamard matrices and Daubechies wavelets, 2016. Master's thesis, *University of Oslo*.

[9] 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. *PNAS*, 2020.

[10] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 214–223, 2017.

[11] S. Arora and B. Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

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

[13] N. Baker et al. Workshop report on basic research needs for scientific machine learning: Core technologies for artificial intelligence. Technical report, USDOE Office of Science (SC), Washington, DC (United States), 2019.

[14] 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 J. Imaging Sci.*, 10(1):335–371, 2017.

[15] A. Bastounis, A. C. Hansen, and V. Vlačić. The extended Smale's 9th problem - on computational barriers and paradoxes in estimation, regularisation, learning and computer-assisted proofs. *Preprint*, 2020.

[16] S. Becker, J. Bobin, and E. J. Candès. NESTA: A fast and accurate first-order method for sparse recovery. *SIAM J. Imaging Sci.*, 4(1):1–39, 2011.

[17] A. Belloni, V. Chernozhukov, and L. Wang. Square-root lasso: pivotal recovery of sparse signals via conic programming. *Biometrika*, 98(4):791–806, 2011.

[18] J. Ben-Artzi, M. J. Colbrook, A. C. Hansen, O. Nevanlinna, and M. Seidel. Computing Spectra – On the Solvability Complexity Index hierarchy and towers of algorithms. *arXiv:1508.03280v5*, 2020.

[19] J. Ben-Artzi, M. Marletta, and F. Rösler. Computing scattering resonances. *arXiv:2006.03368*, 2020.

[20] J. Ben-Artzi, M. Marletta, and F. Rösler. Computing the sound of the sea in a seashell. *arXiv:2009.02956*, 2020.

[21] A. Ben-Tal and A. Nemirovski. Lectures on modern convex optimization–2020/2021.

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

[23] P. Blanchard, D. J. Higham, and N. J. Higham. Accurately computing the log-sum-exp and softmax functions. *IMA J. Numer. Anal.*, 2020. draa038.

[24] L. Blum, F. Cucker, M. Shub, and S. Smale. *Complexity and Real Computation*. Springer-Verlag New York, Inc., 1998.

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

[26] H. Boche and V. Pohl. The solvability complexity index of sampling-based Hilbert transform approximations. In *2019 13th International conference on Sampling Theory and Applications (SampTA)*, pages 1–4. IEEE, 2019.

[27] H. Bölcskei, P. Grohs, G. Kutyniok, and P. Petersen. Optimal approximation with sparsely connected deep neural networks. *SIAM J. Math. Data Sci.*, 1:8–45, 2019.

[28] J. Bolte, A. Daniilidis, and A. Lewis. The łojasiewicz inequality for nonsmooth subanalytic functions with applications to subgradient dynamical systems. *SIAM J. Optim.*, 17(4):1205–1223, 2007.

[29] N. Boullé, Y. Nakatsukasa, and A. Townsend. Rational neural networks. *arXiv preprint arXiv:2004.01902*, 2020.

[30] C. Boyer, J. Bigot, and P. Weiss. Compressed sensing with structured sparsity and structured acquisition. *Appl. Comput. Harmon. Anal.*, 46(2):312 – 350, 2019.

[31] T. A. Bubba, G. Kutyniok, M. Lassas, M. Maerz, W. Samek, S. Siltanen, and V. Srinivasan. Learning the invisible: a hybrid deep learning-shearlet framework for limited angle computed tomography. *Inverse Problems*, 35(6):064002, 2019.

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

[33] E. J. Candes and Y. Plan. A Probabilistic and RIPless Theory of Compressed Sensing. *IEEE Trans. Inf. Theory*, 57(11):7235–7254, 2011.

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

[35] E. J. Candes, J. K. Romberg, and T. Tao. Stable signal recovery from incomplete and inaccurate measurements. *Commun. Pure Appl. Math.*, 59(8):1207–1223, 2006.

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

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

[38] A. Chambolle and T. Pock. On the ergodic convergence rates of a first-order primal–dual algorithm. *Math. Program.*, 159(1-2):253–287, 2016.

[39] X. Chen, J. Liu, Z. Wang, and W. Yin. Theoretical linear convergence of unfolded ISTA and its practical weights and thresholds. In *Advances in Neural Information Processing Systems*, pages 9061–9071, 2018.

[40] M. J. Colbrook and A. C. Hansen. On the infinite-dimensional QR algorithm. *Numer. Math.*, 143(1):17–83, 2019.

[41] M. J. Colbrook, B. Roman, and A. C. Hansen. How to compute spectra with error control. *Phys. Rev. Lett.*, 122(25):250201, 2019.

[42] F. Cucker. The arithmetical hierarchy over the reals. *J. Logic Comput.*, 2(3):375–395, 1992.

[43] G. E. Dahl, D. Yu, L. Deng, and A. Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Trans. Audio Speech Lang. Process.*, 20(1):30–42, 2011.

[44] I. Daubechies, R. DeVore, S. Foucart, B. Hanin, and G. Petrova. Nonlinear approximation and (deep) relu networks. *ArXiv*, abs/1905.02199, 2019.

[45] R. DeVore, B. Hanin, and G. Petrova. Neural network approximation. *arXiv preprint arXiv:2012.14501*, 2020.

[46] R. A. DeVore. Nonlinear approximation. *Acta Numer.*, 7:51–150, 1998.

[47] D. L. Donoho. Compressed sensing. *IEEE Trans. Inf. Theory*, 52(4):1289–1306, 2006.

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

[49] Q. Fan et al. MGH–USC human connectome project datasets with ultra-high b-value diffusion MRI. *Neuroimage*, 124:1108–1114, 2016.

[50] C. Fefferman and L. Seco. On the energy of a large atom. *Bull. Amer. Math. Soc. (N.S.)*, 23(2):525–530, 1990.

[51] C. Fefferman and L. Seco. Interval arithmetic in quantum mechanics. In *Applications of interval computations (El Paso, TX, 1995)*, volume 3 of *Appl. Optim.*, pages 145–167. Kluwer Acad. Publ., Dordrecht, 1996.

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

[53] S. Foucart and H. Rauhut. *A mathematical introduction to compressive sensing*. Birkhäuser Basel, 2013.

[54] M. P. Friedlander, H. Mansour, R. Saab, and Ö. Yilmaz. Recovering compressively sampled signals using partial support information. *IEEE Trans. Inf. Theory*, 58(2):1122–1134, 2012.

[55] M. Genzel, J. Macdonald, and M. März. Solving inverse problems with deep neural networks–robustness included? *arXiv preprint arXiv:2011.04268*, 2020.

[56] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

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

[58] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

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

[60] K. Gregor and Y. LeCun. Learning fast approximations of sparse coding. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pages 399–406, 2010.

[61] T. Hales et al. A formal proof of the Kepler conjecture. *Forum Math. Pi*, 5:e2, 29, 2017.

[62] T. C. Hales. A proof of the Kepler conjecture. *Annals of Mathematics (2)*, 162(3):1065–1185, 2005.

[63] 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. *Magn. Reson. Med.*, 79(6):3055–3071, 2018.

[64] R. Hamon, H. Junklewitz, and I. Sanchez. Robustness and explainability of artificial intelligence. *JRC Technical Report*, 2020.

[65] A. C. Hansen. On the solvability complexity index, the n-pseudospectrum and approximations of spectra of operators. *J. Amer. Math. Soc.*, 24(1):81–124, 2011.

[66] T. Hastie, R. Tibshirani, and M. Wainwright. *Statistical learning with sparsity: the lasso and generalizations*. CRC press, 2015.

[67] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[68] C. F. Higham and D. J. Higham. Deep learning: An introduction for applied mathematicians. *SIAM Rev.*, 61(4):860–891, 2019.

[69] G. Hinton et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Process Mag.*, 29(6):82–97, 2012.

[70] 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.

[71] 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.

[72] A. Jones, A. Tamtögl, I. Calvo-Almazán, and A. Hansen. Continuous compressed sensing for surface dynamical processes with helium atom scattering. *Sci. Rep.*, 6:27776, 2016.

[73] H. Karimi, J. Nutini, and M. Schmidt. Linear convergence of gradient and proximal-gradient methods under the Polyak-łojasiewicz condition. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 795–811. Springer, 2016.

[74] F. Knoll et al. Advancing machine learning for MR image reconstruction with an open competition: Overview of the 2019 fastMRI challenge. *Magn. Reson. Med.*, 2020.

[75] E. Kobler, A. Effland, K. Kunisch, and T. Pock. Total deep variation: A stable regularizer for inverse problems. *arXiv preprint arXiv:2006.08789*, 2020.

[76] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[77] G. Kutyniok and W.-Q. Lim. Optimal compressive imaging of fourier data. *SIAM J. Imag. Sci.*, 11(1):507–546, 2018.

[78] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521:436 EP –, 05 2015.

[79] J. Liu, X. Chen, Z. Wang, and W. Yin. ALISTA: Analytic weights are as good as learned weights in LISTA. In *International Conference on Learning Representations*, 2018.

[80] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[81] M. Lustig, D. Donoho, and J. M. Pauly. Sparse MRI: The application of compressed sensing for rapid MR imaging. *Magn. Reson. Med.*, 58(6):1182–1195, 2007.

[82] J. Ma, R. P. Sheridan, A. Liaw, G. E. Dahl, and V. Svetnik. Deep neural nets as a method for quantitative structure–activity relationships. *JCIM*, 55(2):263–274, 2015.

[83] S. Mallat. *A wavelet tour of signal processing: The sparse way*. Academic Press, third edition, 2008.

[84] Y. V. Matiyasevich. *Hilbert's tenth problem*. MIT Press, 1993.

[85] 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.

[86] C. McMullen. Families of rational maps and iterative root-finding algorithms. *Annals of Mathematics (2)*, 125(3):467–493, 1987.

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

[88] V. Monga, Y. Li, and Y. C. Eldar. Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing. *arXiv preprint arXiv:1912.10557*, 2019.

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

[90] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2574–2582, 2016.

[91] M. J. Muckley et al. State-of-the-art Machine Learning MRI Reconstruction in 2020: Results of the Second fastMRI Challenge. *arXiv preprint arXiv:2012.06318*, 2020.

[92] A. Pinkus. Approximation theory of the MLP model in neural networks. *Acta Numer.*, 8:143–195, 1999.

[93] R. B. Platte, L. N. Trefethen, and A. B. Kuijlaars. Impossibility of fast stable approximation of analytic functions from equispaced samples. *SIAM Rev.*, 53(2):308–318, 2011.

[94] B. Poonen. Undecidable problems: a sampler. *Interpreting Gödel: Critical Essays*, pages 211–241, 2014.

[95] A. Raj, Y. Bresler, and B. Li. Improving robustness of deep-learning-based image reconstruction. In *International Conference on Machine Learning*, pages 7932–7942. PMLR, 2020.

[96] R. T. Rockafellar. Monotone operators and the proximal point algorithm. *SIAM J. Control Optim.*, 14(5):877–898, 1976.

[97] B. Roman, A. Hansen, and B. Adcock. On asymptotic structure in compressed sensing. *arXiv preprint arXiv:1406.4178*, 2014.

[98] V. Roulet, N. Boumal, and A. d'Aspremont. Computational complexity versus statistical performance on sparse recovery problems. *Inf. Inference*, 9(1):1–32, 2020.

[99] S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

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

[101] S. Smale. On the efficiency of algorithms of analysis. *Bull. Amer. Math. Soc. (N.S.)*, 13(2):87–121, 1985.

[102] S. Smale. Complexity theory and numerical analysis. *Acta Numer.*, 6:523–551, 1997.

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

[104] R. Strack. Imaging: AI transforms image reconstruction. *Nat. Methods*, 15(5):309, 2018.

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

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

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

[108] I. Y. Tyukin, D. J. Higham, and A. N. Gorban. On adversarial examples and stealth attacks in artificial intelligence systems. *arXiv preprint arXiv:2004.04479*, 2020.

[109] S. A. van de Geer. *Estimation and testing under sparsity*. Springer, 2016.

[110] Q. Wang, M. Zenge, H. E. Cetingul, E. Mueller, and M. S. Nadar. Novel sampling strategies for sparse MR image reconstruction. *Proc. Int. Soc. Mag. Res. in Med.*, 2014.

[111] M. Webb and S. Olver. Spectra of Jacobi operators via connection coefficient matrices. *arXiv preprint arXiv:1702.03095*, 2017.

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

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