WIDE-BAND BUTTERFLY NETWORK: STABLE AND EFFICIENT INVERSION VIA MULTI-FREQUENCY NEURAL NETWORKS.*

MATTHEW LI†, LAURENT DEMANET‡, AND LEONARDO ZEPEDA-NÚÑEZ§

Abstract.
We introduce an end-to-end deep learning architecture called the *wide-band butterfly network* (`WideBNet`) for approximating the inverse scattering map from wide-band scattering data. This architecture incorporates tools from computational harmonic analysis, such as the butterfly factorization, and traditional multi-scale methods, such as the Cooley-Tukey FFT algorithm, to drastically reduce the number of trainable parameters to match the inherent complexity of the problem. As a result `WideBNet` is efficient: it requires fewer training points than off-the-shelf architectures, and has stable training dynamics, thus it can rely on standard weight initialization strategies. The architecture automatically adapts to the dimensions of the data with only a few hyper-parameters that the user must specify. `WideBNet` is able to produce images that are competitive with optimization-based approaches, but at a fraction of the cost, and we also demonstrate numerically that it learns to super-resolve scatterers in the full aperture scattering setup.

1. Introduction. There is nowadays extensive documentation on the remarkable ability of neural networks to approximate high-dimensional, non-linear maps provided that enough data is available [58]. In many applications the process of discovering such approximations simply involves enriching the network models, i.e. making them wider and/or deeper, until favourable stationary points arise in the empirical loss landscape. This practice can be partially justified by the asymptotic capacity of neural networks to approximate functions to within arbitrary accuracy, assuming only mild regularity conditions [22, 49, 67]. Oftentimes, however, this strategy results in models that are vastly overparametrized even when compared to the already massive datasets that are necessary for training. For reasons we outline below, these approximation-theoretic results also obscure many pre-asymptotic complications that are particularly acute when neural networks are applied to scientific applications. In these instances the neural architecture often requires to be specifically tailored to the task at hand in order to satisfy the stricter requirements of scientific computing.

In this paper we focus on the problem of high-resolution imaging of scatterers arising from wave-based inverse problems. This task naturally arises in many scientific applications: e.g. biomedical imaging [77], synthetic aperture radar [20], non-destructive testing [71], and geophysics [75]. This problem also prototypically exhibits two challenges that are commonly encountered in scientific machine learning. First: obtaining the training data in this setting – whether synthetically or experimentally – comes at considerable expense, which bottlenecks the size of the models that can be reliably trained to satisfy the stringent accuracy requirements. This necessitates the use of unconventional architectures that are bespoke to each problem. Second: wave scattering involves *non-smooth data* that are recordings of highly oscillatory, broadband, scattered waveforms. These highly oscillatory (i.e. high-frequency) signals are

*Submitted to the editors DATE.
Funding: The authors thank Total SA for support. LD is also supported by AFOSR grant FA9550-17-1-0316. L.Z.-N. is also supported in part by the Wisconsin Alumni Research Fund, the National Science Foundation under the grant DMS-2012292, and NSF TRIPODS award 1740707.

†Computational Science and Engineering, Massachusetts Institute of Technology, Cambridge MA 02139 (mtcli@mit.edu)
‡Department of Mathematics and Earth Resources Lab, Massachusetts Institute of Technology, Cambridge MA 02139 (laurent@math.mit.edu)
§Department of Mathematics, University of Wisconsin-Madison, Madison WI 53706 (zepeda-nunez@wisc.edu)

known to impede the training dynamics of many machine learning algorithms [86] and thus require new strategies to mitigate their effect.

Existing methods for scientific machine learning address the issue of data scarcity by "incorporating underlying physics" into the design of neural architectures. In instances where the problem data is *smooth*, this demonstrably reduces the total number of trainable weights which, in turn, reduces the number of training data required. Broadly categorized, these designs manifest as either: *(i)* explicitly enforcing physical symmetries into the network [90, 93, 94], *(ii)* exploiting signal invariances and equivariances when processing the data [12], *(iii)* directly embedding the governing differential equations into the objective function [51, 74], or *(iv)* imposing information flow (i.e. connectivity) within the architecture according to multiscale interactions inherent to the physics of the data generating process [37, 36, 54]. Surprisingly, in addition to lowering data requirements these strategies are also observed to improve on the testing accuracy of comparable conventional models which are trained on a larger set of training points [46, 67, 92].

In comparison, not much is known about designing architectures for processing *non-smooth data* such as high-frequency waves. Here the same challenge that confounds the original inverse problem – namely, the processing of highly oscillatory signals – similarly obstructs direct application of machine learning methods. This idea is formalized by the "F-principle" conjecture [86] which documents the relation between machine learning methods and Fourier analysis. Specifically, it is empirically observed that models with fully-connected and convolutional architectures preferentially capture the low-frequency features of the target function. On the other hand, considerable expense (with respect to model size and/or data) is needed to learn high-frequency features [65]. Some examples even demonstrate that training can completely fail when the target function lacks low-frequency content even if highly expressive models are used [15, 85]. The F-principle thus demonstrates that although neural networks are universal approximators in an asymptotic sense, new strategies are needed to account for the issue with high frequencies if tractably computable models are to be obtained.

We note that in our application the forward and inverse maps are intrinsically oscillatory on account of the physics of wave propagation. This can be seen as an immediate consequence of the *dispersion relation*,

$$\lambda f = c, \tag{1.1}$$

which describes the inverse scaling of the frequency f of propagating waves to their spatial wavelength λ by a factor of the local wavespeed c. This dispersion relation, in conjunction with rudimentary signal processing, effectively establishes that images generated by back-propagating the recorded waves into the medium are constrained to a resolution limit of $\lambda/2$, i.e. the classical diffraction limit [41]. High resolution imaging of scatterers thus seemingly necessitates the use of high frequency waves to probe the media.

1.1. Our Contributions. We introduce a custom architecture for the inverse wave scattering problem which we call `WideBNet`. We demonstrate that our architecture overcomes the major deficiencies outlined above for traditional architectures. Specifically, `WideBNet` relies on ideas from the butterfly factorization [61] to capture the Fourier Integral Operators (FIOs) underlying the physics of wave-scattering – as a result, fewer training data points are needed. Moreover, it addresses the high frequency limitations identified by F-principle by mimicking the Cooley-Tukey algorithm [27] to process multi-frequency data only at localized length scales – this

effectively renders each frequency slice as *locally* low-frequency information. These design choices afford `WideBNet` the following benefits compared to off-the-shelf deep learning models:

Training Efficiency The architecture builds upon the butterfly factorization and thus systematically adapts to the input size of the data, i.e. the number of pixels in the image. As a result, the degrees of freedom in the model scale near-linearly with the input size, and the depth of the network scales logarithmically with the input size[1]. This makes training our network data-efficient as there are relatively fewer degrees of freedom.

Training Stability `WideBNet` avoids empirically observed shortcomings with other network architectures that rely on the butterfly factorization. For example, in [60] the authors prove that butterfly-networks are capable of efficiently approximating generic FIO's, but report that learning such operators requires an accurate initialization to avoid local minima; this is typically not easily obtainable for most FIOs, including our application. Similarly [54] introduces a butterfly-network for single frequency inversion but requires increasing the width of their network (so that the degrees of freedom no longer scale linearly) to overcome local minima. In contrast, empirically we observe that `WideBNet` does not require specialized initialization strategies, it does not routinely get stuck in local minima, and it does not exhibit exploding/vanishing gradients. We speculate that the training stability of `WideBNet` can be attributed to its use of multi-frequency data that is banded to appropriate length scales to avoid the F-principle limitations.

Imaging Super-resolution In our numerical results we demonstrate that our network superresolves scatterers, i.e. produces sharp images of sub-wavelength features[2] such as diffraction corners, in addition to producing competitive images when compared against classical inversion methods in the traditional super-diffraction regime.

Hyper-parameter Efficiency It is efficient to tune the hyper-parameters of `WideBNet` as there are only a few which are used to describe the architecture. We note that in numerical examples we observe strong robustness to variations in these hyper-parameters. This indicates that relatively little effort is needed on the user's part to optimally tune our architecture.

A detailed discussion of the `WideBNet` architecture, as well as implementation notes, can be found in Section 3. Meanwhile, we briefly sketch the intuition behind the design choice here. The idea to embed the butterfly factorization into the architecture is to effectively furnish our network with a strong prior on the physics of wave scattering. Indeed, we provide compelling numerical evidence that it is necessary to manually encode the long range "non-local" interactions between scatterers and sources that are inherent to the wave kernel. Mathematically these interactions are known to described as the action of an FIO [48], which can be discretely represented

[1] When compared to other machine learning based approaches, we note that a comparable implementation using fully connected networks results in models with degrees of freedom that scale cubically with the size of the input, i.e. the number of pixels in the image, and are thus prohibitively expensive to train. Conversely, a purely convolutional neural network implementation for the task requires far deeper networks (or far wider filters) to properly capture the long-range interactions governed by the underlying wave physics. Such deep networks are known to exhibit issues with exploding/vanishing gradients leading to unstable training dynamics [7] . While we do not discount the possibility of other hybridized (fully connected + convolutional) architectures which achieve the same task, we emphasize that these architectures would not be immediately transferable for different image and data resolution requirements.

[2] We plan to further investigate and document this super-resolution phenomenon in forthcoming work.

in a complexity-optimal manner by means of the butterfly factorization [61] and the butterfly algorithm [16, 70, 11].

However we stress that the marriage of the butterfly factorization with network architectures is *not* the original contribution of this work; butterfly-like architectures have been previously proposed by other authors, albeit with different goals [60, 54], and we review these contributions below in Section 1.2. Instead, our contribution is the *combination of this network architecture with multifrequency data*. This data assimilation strategy takes cues from the Cooley-Tukey algorithm and is done, in part, to address the F-principle. For reference, one notable strategy for avoiding the F-principle involves partitioning the model into disjoint Fourier segments and frequency down-shifting accordingly [14], but this introduces costly convolutions in data-space and requires a dense data sampling strategy that scales unfavourably with dimensionality. Our network improves on this approach by exploiting the duality between frequency f and wavelength λ, as described by the dispersion relation in (1.1), to introduce data only at their local length scales. This effectively performs frequency downshifting by *spatial downsampling*. This strategy is easily accommodated by the butterfly architecture as these multiscale interactions are already implicitly present in its formulation.

Outline. The remainder of this document is structured as follows. We close this section with relevant background material on existing algorithms for inverse scattering and relevant machine learning based approaches for general inverse problems in Section 1.2. Section 2 describes the technical details of the underlying physical model, provides background on the problem to solve and the algorithmic ideas behind the network. In Section 3 we present in detail the network architecture. Finally, in Section 4 we present and discuss the numerical results.

1.2. Related Literature.

1.2.1. Classical Approaches. One of the earliest modalities in imaging is travel-time tomography [69, 45, 5], in which the travel time of a wave passing between two points is used to reconstruct the medium wave-speed [78]. Travel-time tomography is a rather mature technique, which can even be easily and cost effectively implemented in portable ultra-sound devices [23]. However, its resolution deteriorates greatly when dealing with highly heterogeneous media and in the presence of multiple scattering.

In response to these drawbacks, several techniques were developed such as reverse time migration [6], linear sampling method [24], decomposition methods [57] among many other. See [26] and [83] for excellent historical reviews.

Finally, a high-resolution technique, called full-waveform inversion (FWI) [80] was developed in the late 80s, which has been shown empirically capable of handling multiple scattering. FWI solves a constrained optimization problem in which the misfit between the real data and synthetic data coming from the numerical solution of the PDE are minimized. This technique, coupled with large computing power, has been successful at recovering the properties of the sub-surface [73]. Nowadays, it is considered the gold standard in geophysical exploration [82].

Despite its enormous success, FWI still suffers from three significant challenges: prohibitive computational cost, cycle-skipping and limited resolution. The prohibitive computational cost is linked to the cost of computing the gradient within the optimization loop, which requires a large amount of wave solves. The resulting complexity

of each iterations is quadratic[3] [10] with respect to number of unknowns to recover. Progress in this direction has focused on developing fast PDE solvers [91, 35] which are necessary to compute the gradient. In addition, a large number of iterations is usually required for convergence. This prohibitive computational cost has hampered the application of this vastly superior technique to domains where images are required on-the-fly, such as biomedical imaging.

Cycle-skipping is the undesirable convergence to spurious local minima, specially in the absence of low-frequency data. In a nutshell, the low frequencies help determine the kinematically relevant, low-wavenumber components of the material properties, which are in turn needed to avoid convergence of FWI to spurious local minima. However, in practice, acquiring low-frequency data from the field is a challenging and expensive task. Progress in this area has focused on regularizing the optimization routine to handle the lack of low-frequency data [79, 81], using a smooth initial guess from travel-time tomography [3], or extrapolating the low-frequency component from higher frequency data [63]. Finally, the resolution of the output of FWI is usually constrained by the Shannon-Nyquist scaling, i.e., the finest details available in the output are bound by the shortest wavelength at which data is available. This is important for accurately imaging discontinuities [4, 13, 31, 30], which in return, are crucial for properly interpreting geophysical formations, for properly detecting cracks in materials, and for detecting and interpreting anomalies in biomedical imaging.

1.3. Machine Learning Approaches. Besides the classical, PDE constrained optimization approaches, several recent methodologies based on machine learning for more general inverse problems have been proposed lately.

In [19] authors used the recently introduced paradigm of physics informed neural networks (PINN) to solve for inverse problems in optics. Aggarwal et al. introduce a model-based image reconstruction framework [2] for MRI reconstruction. The formulation contains a novel data-consistency step that performs conjugate gradient iterations inside the unrolled algorithm. Gilton et al. proposed in [42] a novel network based on Neumann series coupled with a hand-crafted preconditioner for linear inverse problems, which recast an unrolled algorithm as elements of a Neumann series. In [66] Mao et al. use a deep encoder-decoder network reminiscent of U-nets [76] for image denoising, using symmetric skip connections.

In [38] the authors proposes a rotationally equivariant network for inverse scattering, that is only valid for homogeneous media; the same type of ideas is applied to travel-time tomography [40] and optical tomography [39].

Among the more general field of computational harmonic analysis, to which the butterfly algorithm is connected, we have several other applications. Networks based on the Short-time Fourier transform [88, 87] has been used for hierarchically decomposing signals in a non-linear fashion. Networks based on the scattering transform has been proposed [12] to take in account translation invariance in images. In [89] the authors introduced another framework based on frames for inverse problems, which was applied to computer tomography denoising [53].

In addition, machine learning recently has been used for super resolution in the signal processing context [18] and image processing. Recently newly developed frameworks such as generative adversarial networks (GANs) [43, 44], and variational autoencoders (VAEs) [56, 33] have been used for super resolution in the context of image processing [52, 59, 68]. These techniques provides a end-to-end map that relies on the

[3]Using state-of-the-art sparse direct solvers. It can be further reduced to $\mathcal{O}(N^{3/2})$ using state-of-the-art preconditioner, but with substantially larger constants.

statistical properties of the images to super-resolve them.

The method introduced in this manuscript follows ideas similar to the ones in [37, 36, 29], where authors introduce tools from numerical analysis into deep learning. They build on the sparse matrix factorizations that result from exploiting low-rank interactions arising from the underlying physics of the problem. These factorization are translated into the machine learning context: each matrix factor becomes a layer in the network wherein the sparsity pattern informs the connectivity between layers, and the matrix entries themselves are viewed as learnable weights. In particular, the authors translate hierarchical matrices ($\mathcal{H}$-matrices), which are factorizations of operators into low-rank and permutations matrices, into individual layers in neural network architectures. Although these networks are well suited for smooth data with compressible long range interactions, which is the underlying motivation for the $\mathcal{H}$-matrices, they are not well suited for wave-scattering problems where the data is highly oscillatory, and where the long-range interactions are not typically compressible.

Instead, the correct idea for capturing wave propagation is the choice of the butterfly factorization, as motivated by their use for representing FIOs. In fact, architectures based on butterfly algorithm have been previously proposed, albeit with different goals as the one considered in this paper. In [28] the authors recover the butterfly structure of certain linear operators, from permutation operations. In [54] the authors use a one-level butterfly network with applications to inverse scattering, though critically they require a super-linear scaling in their number of parameters. In [60] the authors propose a mono-chromatic butterfly network similar to the architecture used in this case, which was later simplified in [84]. In [29], the authors use the backbone of the butterfly structure to learn fast matrix approximations, with a clever variational relaxation strategy for learning the permutation factors. However, as mentioned in the prequel, none of these works address the use of butterfly factorizations for super-resolution in wave-based imaging which requires stable training over a wideband dataset.

2. Background. In this section we briefly review concepts from classical imaging (see [25] for further details) and their connection with fast numerical methods. We also provide a succinct description of the butterfly factorization and Cooley-Tukey FFT algorithm to motivate the discussion of our architecture in Section 3.

2.1. Underlying Physical Model. We consider the time-harmonic wave equation with constant-density acoustic physics, also called the Helmholtz equation, with frequency ω and squared slowness m, given by

$$(\Delta + \omega^2 m(\mathbf{x}))u(\mathbf{x}) = 0 \tag{2.1}$$

with radiating boundary conditions. We further suppose the slowness admits a scale separation into

$$m(\mathbf{x}) = m_0(\mathbf{x}) + \eta(\mathbf{x}),$$

where m_0 corresponds to the smooth background slowness, assumed known, and η the rough perturbation that we wish to recover. If the background slowness is constant and normalized[4] so that

$$m(\mathbf{x}) = 1 + \eta(\mathbf{x}),$$

[4]This assumption is only made to make the presentation more transparent.

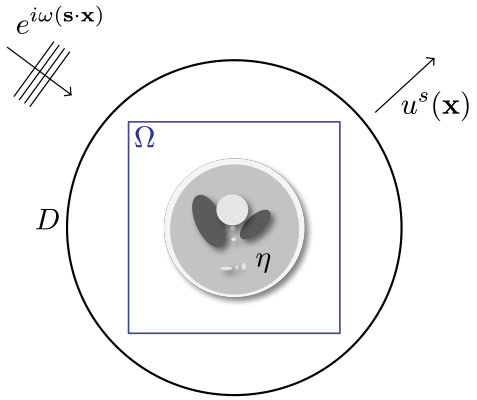

FIG. 1. *Setup for the inverse scattering problem. In this case we probe the medium with a planewave with direction* $\mathbf{s}$*, and we sample the scattered field on the disk* D*.*

then solutions to (2.1) can be expressed in the form

$$u(\mathbf{x}) = e^{i\omega(\mathbf{s}\cdot\mathbf{x})} + u^{sc}(\mathbf{x}),$$

where $e^{i\omega(\mathbf{s}\cdot\mathbf{x})}$ is the incoming plane wave, with propagating direction $\mathbf{s}$, that we use to "probe" the perturbation, and $u^{sc}(\mathbf{x})$ is the scattered field produced by the interaction of the perturbation with the impinging wave. The scattered field satisfies

$$\begin{cases} \left(\Delta + \omega^2(1+\eta(\mathbf{x})\right) u^{sc}(\mathbf{x}) = -\omega^2\eta(\mathbf{x})e^{i\omega(\mathbf{s}\cdot\mathbf{x})} \quad \text{for } \mathbf{x}\in\mathbb{R}^2, \\ \lim_{|\mathbf{x}|\to\infty} |\mathbf{x}|^{1/2}\left(\dfrac{\partial}{\partial|\mathbf{x}|} - \mathrm{i}\omega\right) u^{\mathrm{s}}(\mathbf{x}) = 0, \end{cases} \tag{2.2}$$

following the setup depicted in Fig. 1. Just as in Fig. 1 we select the detector manifold D to be a circle of radius R that is sufficiently large enough to invoke far-field approximations. For each incoming direction $\mathbf{s}\in\mathbb{S}^1$ the data is given by sampling the scattered field with receiver elements that are located on D and indexed by $\mathbf{r}\in\mathbb{S}^1$. This yields the far-field pattern given by

$$\Lambda_{\mathbf{s},\mathbf{r}} = u^{sc}(R\mathbf{r})$$

where $\mathbf{s}$ denotes the incoming probing direction as defined in (2.2). We call $\mathcal{F}^\omega[\eta]$ the *forward map* relating the perturbation η to its corresponding far-field pattern.

Accordingly we can cast the inverse problem for recovering the rough perturbation as

$$\eta^* = \mathrm{argmin}_\mu \|\mathcal{F}^\omega[\mu] - \Lambda_{\mathbf{s},\mathbf{r}}\| \tag{2.3}$$

where $\Lambda_{\mathbf{s},\mathbf{r}}$ is the measured data. It will be instructive to linearize $\mathcal{F}^\omega$ to shed light on the essential difficulties of this problem. Using the classical Born approximation in (2.2) we obtain that

$$u^{sc}(\mathbf{x}) = \omega^2 \int_{\mathbb{R}^2} \Phi^\omega(\mathbf{x},\mathbf{y})\eta(\mathbf{y})e^{i\omega(\mathbf{s}\cdot\mathbf{y})}d\mathbf{y}, \tag{2.4}$$

where Φ^ω is the Green's function of the two-dimensional Helmholtz equation in homogeneous media, i.e., Φ^ω satisfies

$$\begin{cases} \left(\Delta + \omega^2\right) \Phi^\omega(\mathbf{x}, \mathbf{y}) = -\delta(\mathbf{x}, \mathbf{y}) & \text{for } \mathbf{x} \in \mathbb{R}^2, \\ \lim\limits_{|\mathbf{x}|\to\infty} |\mathbf{x}|^{1/2} \left(\dfrac{\partial}{\partial |\mathbf{x}|} - \mathrm{i}\omega \right) \Phi^\omega(\mathbf{x}, \mathbf{y}) = 0. \end{cases} \tag{2.5}$$

Furthermore, we can use the classical far-field asymptotics of the Green's function to express

$$u^{sc}(R\mathbf{r}) = -\omega^2 \frac{e^{i\omega R}}{\sqrt{R}} \int_{\mathbb{R}^2} \eta(y) e^{i\omega(\mathbf{s}-\mathbf{r})\cdot\mathbf{y})} d\mathbf{y} + \mathcal{O}(R^{-3/2}). \tag{2.6}$$

Thus, up to a re-scaling and a phase change, the far-field pattern defined in (2.7) can be approximately written as a Fourier transform of the perturbation, i.e.,

$$\Lambda_{\mathbf{s},\mathbf{r}}(\omega) \approx F^\omega \eta = -\omega^2 \frac{e^{i\omega R}}{\sqrt{R}} \int_{\mathbb{R}^2} e^{i\omega(\mathbf{s}-\mathbf{r})\cdot\mathbf{y}} \eta(y) d\mathbf{y} \tag{2.7}$$

is the linearized forward operator acting on the perturbation.

Solving the inverse problem (2.3) using the linearized operator in (2.7) results in an explicit solution given by the normal equation

$$\eta^* = \left((F^\omega)^* F^\omega + \epsilon I \right)^{-1} (F^\omega)^* \Lambda_{\mathbf{s},\mathbf{r}}, \tag{2.8}$$

which is often also referred to as filtered back-projection [25]. The constant ϵ is a small regularization parameter that remedies the ill-conditioning of $(F^\omega)^* F^\omega$.

Performing the inversion numerically requires discretizing the wavespeed and the sampling geometry. We discretize Ω using $N = n_x \times n_z$ degrees of freedom following the Nyquist sampling rate of $n_x \sim n_z \sim \omega$. The scattered data $\Lambda_{\mathbf{s},\mathbf{r}}$ is discretized into an $n_{src} \times n_{rcv}$ matrix.

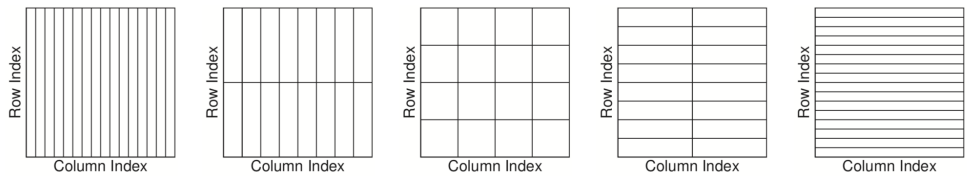

FIG. 2. *Sketch of a matrix exhibiting a complementary low-rank property. Each of the blocks induced by the different partitions has the same ϵ-rank.*

After discretization and a change of variables, $(F^\omega)^*$ in (2.8) is a Fourier transform (which itself is a FIO), and F^*F is a pseudo-differential operator, which, in addition, is translation invariant, and $(F^*F + \epsilon I)^{-1}$ can be reduced to a convolution-type operator.

Remark: Thus far we have assumed that we probe the perturbation η using only a monochromatic time-harmonic wave with fixed frequency ω. As mentioned in the introduction this is known to be ill-posed and data at additional frequencies is required to stabilize the reconstruction [50]. In particular, a time-domain formulation

known as the *imaging condition* yields a more stable reconstruction using the full frequency bandwidth; this formula can be formally stated as

$$\eta^* = \int_{\mathbb{R}} \left((F^\omega)^* F^\omega + \epsilon I \right)^{-1} (F^\omega)^* \Lambda_{\mathbf{s},\mathbf{r}}(\omega) d\alpha(\omega), \tag{2.9}$$

where $d\alpha(\omega)$ is a density related to the frequency content of the probing wavelet. When the density is well approximated by a discrete measure then

$$\eta^* \approx \sum_{i=1}^{N_{\text{freqs}}} \left((F^{\omega_i})^* F^{\omega_i} + \epsilon I \right)^{-1} (F^{\omega_i})^* \Lambda_{\mathbf{s},\mathbf{r}}(\omega_i) \alpha(\omega_i), \tag{2.10}$$

over a discrete set of frequencies $\{\omega_i\}_{i=1}^{N_{\text{freqs}}}$. We note that the selection of these frequencies, in addition to the optimal ordering in which the summation is computed under an iterative regime, remains an open question and an area of active research [10].

2.2. Butterfly Factorization and Fourier Integral Operator. When the scattered field is given by (2.7) then one could apply the fast Fourier transform [27] to compute the estimate (2.10) in quasi-linear time. However, with a heterogeneous background the linearized forward map is instead given by a more general representation usually known as a Fourier integral operator (FIO), which has the form

$$(F^\omega \eta)(\mathbf{x}) = \int_{\mathbb{R}^2} a(\mathbf{x}, \mathbf{y}) e^{i\omega\phi(\mathbf{x},\mathbf{y})} \eta(\mathbf{y}) d\mathbf{y}. \tag{2.11}$$

Here $\phi(\mathbf{x}, \mathbf{y})$ is referred to as the phase (or travel-time) function while a is typically a very smooth function that encodes the amplitude[5]. The work of [16, 72, 70] recognized that even in this more generalized instance the application of F^ω and its adjoint can be computed in with optimal complexity by means of the butterfly algorithm. The butterfly algorithm is a multi-scale algorithm which takes advantage of the *complementary low-rank property* of the discretized operator depicted in Fig. 2. In its original form the algorithm relies on explicit knowledge of the phase function; later, in [61] the authors introduced the butterfly factorization, which approximates the discretized operator (2.11) by the multiplication of sparse matrices with a *specific* sparsity pattern[6] as shown in Fig. 3.

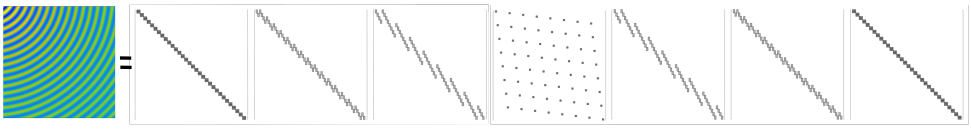

FIG. 3. *Sketch of the butterfly factorization, where the matrix at the left is factorized in sequence of very sparse matrices with a distinct sparsity pattern, induced by Fig. 2.*

In a nutshell, the butterfly factorization approximately factorizes a matrix A that satisfies the complementary low-rank property in $L + 3$ sparse matrices following:

$$A \approx A_{butterfly} = U^L G^{L-1} \cdots G^{L/2} S^{L/2} \left(H^{L/2} \right)^* \cdots \left(H^{L-1} \right)^* \left(V^L \right)^*, \tag{2.12}$$

[5]The principal symbol, a, is usually either independent of weakly dependent of ω.

[6]This pattern is for the one-dimensional butterfly factorization, which already captures the key algorithmic ideas while keeping the presentation clean of ordering issues that arises in higher dimension.

where U^L and V^L are block diagonal matrices, $S^{L/2}$ is a weighted permutation matrix, usually called a *switch matrix*, and L is the number of levels in the factorization, which is usually a power of two.

We can interpret the factors in (2.12) following the original butterfly algorithm. V^L extracts a local representation of the vector, then each factor H^ℓ compresses two neighboring local representations, i.e., decimates by a factor of two the number of local representations, while increasing the amount of information in each presentation. The switch matrix $S^{L/2}$ quickly redistribute the information contained in each local representation. The factors G^ℓ decompress the information contained in each representation at each stage, i.e., the local representations are split in two by each factor increasing the spacial resolution, and finally the factor U^L, transforms the local representations to the sampling points.

For the sake of completeness we provide a formal argument to show that the FIO in (2.11) satisfies the complementary rank property (see [16] for a more comprehensive argument). Suppose that we have two points $\mathbf{x}_0$ and $\mathbf{y}_0$ in the evaluation and integration region respectively. We define two neighborhoods around each point, such that $|\mathbf{x}-\mathbf{x}_0| < d_x$ and $|\mathbf{y}-\mathbf{y}_0| < d_y$. We then seek to find the largest values of d_x and d_y such that we can efficiently approximate

$$\int_{|\mathbf{y}-\mathbf{y}_0|<d_y} a(\mathbf{x},\mathbf{y}) e^{i\omega\phi(\mathbf{x},\mathbf{y})}\eta(\mathbf{y})d\mathbf{y}, \tag{2.13}$$

using a separable function. The principal symbol, $a(\mathbf{x},\mathbf{y})$ is supposed to be smooth and independent of ω (or weakly dependent), so we can focus our discussion to the oscillatory term $e^{i\omega\phi(\mathbf{x},\mathbf{y})}$.

Using a Taylor expansion we have that

$$\begin{aligned}\phi(\mathbf{x},\mathbf{y}) =& \phi(\mathbf{x}_0,\mathbf{y}_0) + \partial_{\mathbf{x}}\phi(\mathbf{x}_0,\mathbf{y}_0)\cdot(\mathbf{x}-\mathbf{x}_0) + \partial_{\mathbf{y}}\phi(\mathbf{x}_0,\mathbf{y}_0)\cdot(\mathbf{y}-\mathbf{y}_0)\\ &+(\mathbf{x}-\mathbf{x}_0)^T\cdot\partial^2_{\mathbf{x}}\phi(\mathbf{x}_0,\mathbf{y}_0)\cdot(\mathbf{x}-\mathbf{x}_0) + (\mathbf{y}-\mathbf{y}_0)^T\cdot\partial^2_{\mathbf{y}}\phi(\mathbf{y}_0,\mathbf{y}_0)\cdot(\mathbf{y}-\mathbf{y}_0)\\ &+2(\mathbf{x}-\mathbf{x}_0)^T\cdot\partial^2_{\mathbf{x},\mathbf{y}}\phi(\mathbf{x}_0,\mathbf{y}_0)\cdot(\mathbf{y}-\mathbf{y}_0) + \mathcal{O}(d_x d_y)\end{aligned}$$

Clearly the first five terms provide separable expressions, the sixth term can be easily bounded producing

$$e^{i\omega\phi(x,y)} = e^{i\omega\psi(x)}e^{i\omega\xi(y)}(1+\mathcal{O}(\omega d_x d_y)) \tag{2.14}$$

thus as long as $d_x d_y \leq \omega^{-1}$, then $e^{i\omega\phi(x,y)}$ can be locally approximated by a separable function. In the discrete case this property is translated to the fact that the multiplication of the height and the width of each block has a constant ϵ-rank, which is exactly the complementary low-rank property showcased in Fig. 2.

Remark: We point out that there exist three different types of butterfly factorizations. The left one sided, the right one sided, and the two sided (see [64] for a review). In this work we focus on the two-sided version, which provides the best complexity. It is possible to "neuralize" the other two types of factorizations, which yield a specific type of CNN networks with sparse channel connections as shown in [84].

2.3. Cooley-Tukey Algorithm. The Cooley-Tukey FFT algorithm [27] is one of the most important algorithms in the 20th century [21]. It aims to compute the discrete Fourier transform (DFT) of a signal $\{x_n\}_{n=0}^{N-1}$ given by

$$\hat{x}(k) = \sum_{k=0}^{N-1} x_n e^{-\frac{2\pi i}{N}nk}, \tag{2.15}$$

in $N \log N$ time. The algorithm leverages the algebraic structure of the N-th complex roots of the unit to recursively split the computation. The simplest version of the algorithm is called the radix-2 decimation-in-time FFT, which computes the DFT of both even-indexed and odd-indexed inputs, which are then merged to produce the final result. In particular, for the first level the DFT is rearranged as

$$\begin{aligned}\hat{x}(k) &= \sum_{m=0}^{N/2-1} x_{2m} e^{-\frac{2\pi i}{N/2} mk} + e^{-\frac{2\pi i}{N} k} \sum_{m=0}^{N/2-1} x_{2m+1} e^{-\frac{2\pi i}{N/2} mk}, \\ &= \hat{x}_e(k) + e^{-\frac{2\pi i}{N} k} \hat{x}_o(k),\end{aligned}$$

where $\hat{x}_e(k)$ and $\hat{x}_o(k)$ stand for the even and odd downsampled DFTs respectively. However, given that we are using decimated DFTs this expression is only valid for $k = 0, ...N/2 - 1$. Thus, in order to obtain the full length DFT, one can use the periodicity of the complex exponential, and we have that

$$\hat{x}(k) = \hat{x}_e(k) + e^{-\frac{2\pi i}{N} k} \hat{x}_o(k), \tag{2.16}$$

$$\hat{x}(k + N/2) = \hat{x}_e(k) - e^{-\frac{2\pi i}{N} k} \hat{x}_o(k). \tag{2.17}$$

2.4. Wide-Band Butterfly Algorithm. For the sake of simplicity we motivate the idea behind this paper, which is the multiscale decomposition of the butterfly factorization, by using the Cooley-Tukey FFT algorithm. We point out that the same argument can be obtained from a rather involved analysis of the original butterfly algorithm. In particular, one can follow the description of the algorithm in [32] to show that if we build a compressed FIO, as the one in (2.11), at frequency ω using the butterfly algorithm, then most of the computation can be reused to build the same FIO, but at frequency $\omega/2$.

The cornerstone of the approach is to leverage the recursive nature of the FFT algorithm to reuse most of the algorithm pipeline when computing the FFT of decimated signals, or in the case of (2.11) at lower frequencies. We focus our attention on two operations: computing the DFT of a decimated signal using the FFT for a non-decimated signal, computing the same DFT using a decimated algorithm, but keeping a non-decimated resolution. These two operations will be key when designing our network.

From (2.16) and (2.17) we clearly see that we can compute the DFT of a decimated signal, using the regular FFT algorithm. One only needs to interweave the original signal with zeros, then apply the FFT for the longer signal, and then truncate half of the resulting vector. This means that after a modification of the input we can reuse the algorithmic pipeline from a non-decimated FFT.

Furthermore, if we compute the DFT of a decimated signal, but want to keep the full frequency resolution of the non-decimated one, then (2.16) and (2.17) provides an answer to that: one needs to repeat the result from the decimated signal. This *upscaling* operation will be key when designing the network in Section 3.

These operations follow the same principle behind the wide-band butterfly network. If we want to implement (2.10), we would need to build a network to process the data at each frequency independently. However, using the argument above one can use the recursive decomposition to process the frequencies jointly. In particular, if we want to process data, say at half frequency, i.e., $\omega/2$, then the complementary low-rank conditions states that $d_x d_y \leq 2\omega^{-1}$. If we suppose, in addition, that the

evaluation grid remains constant[7] then d_y can be twice as large, thus inducing a different factorization. However, as mentioned above, each factor in H^ℓ factor in the butterfly factorization (see (2.12)) down-samples the local representation in y, while increasing the resolution in x. This means, that after a small modification at the beginning, followed by an upscaling operation similar to the one in (2.16) and (2.17) when the odd signal is zero, one can reuse the rest of the network, which is idea behind merging the networks to treat the different frequencies jointly at the appropriate scale.

3. `WideBNet` Architecture. We provide a self-contained overview of the network architecture in this section. The material here is tailored towards a machine-learning audience with no prior exposure to the butterfly factorization. Indeed, beyond the salient aspects which we summarize below, implementing `WideBNet` becomes essentially algorithmic since the network structure and connectivity are determined once the dimensions (i.e., grid size) of the data are specified. For clarity our discussion focuses only on two-dimensional scattering as it captures the essential complexities of the problem; adapting the architecture to higher dimensions proceeds in a straightforward manner.

We split the discussion into several parts. In Section 3.1 we define the sampling and formatting of the input data. Section 3.2 provides the overarching ideas of the architecture presented in a modular manner, followed by the definition of each of such module in terms of layers of a neural network. Specific details of these layers are further elaborated in their respective Sections 3.3, 3.4, and 3.5. Lastly in Section 3.6 we provide the number of parameters (trainable weights) required for the network. Pseudo-code for `WideBNet` is provided in Listing 1 below, but the pseudo-code for each specialized layer is located in their corresponding subsection.

```
def wbnn():
  # inputs:  Λ^L,...,Λ^{L/2}
  # output:  [?, 1, 4^L, 1]

  y = V^L(Λ^L)
  for l in range(L-1), L/2-1, -1):
    y = H^l(y, V^l(Λ^l))
  y = SwitchResnet(y)
  for l in range(L/2, L, +1):
    y = G^l(y)
  y = U^L(y)
  y = CNN(y)
  return y
```

LISTING 1

Pseudo code for the `WideBNet` , where each module is explained in detail in Sections 3.4, 3.3, and 3.5.

3.1. Input formatting. We assume the scatterers (discretized over an $n_x \times n_z$ grid) and the scattered data (an $n_{\text{src}} \times n_{\text{rcv}}$ matrix for each frequency ω) are represented using complete quad-trees with L levels[8] with leaf size s. In other words,

[7] This assumption is a direct consequence of (2.10), where the resolution of the perturbation to be reconstructed is fixed.

[8] We require that L is divisible by 2. This is a minor restriction and can be accommodated by e.g. zero padding of the data or by interpolating the data. While the total depth of both quad-trees must be the same, it is not necessary for them to have the same leaf size. However, for ease of

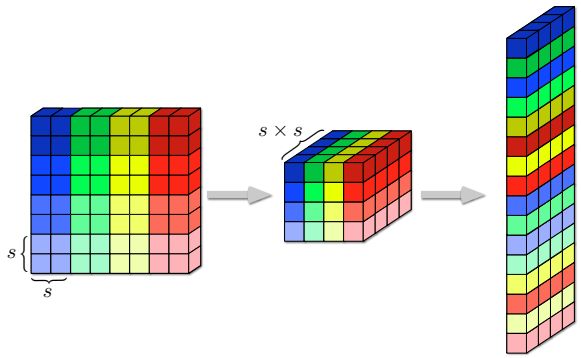

FIG. 4. *Transformation from the image of dimensions* $[2^\ell s,\ 2^\ell s]$ *to the tensorized form of size* $[2^\ell,\ 2^\ell,\ s^2]$*, and then to the flatted tensor using Morton ordering resulting on a tensor of dimensions* $[4^\ell,\ s^2]$.

we require a discretization into $n = 2^\ell s$ points for each matrix dimension, and the level $\ell \in [L/2,\ L]$ indexes the size of the contiguous $2^{L-\ell}s \times 2^{L-\ell}s$ sub-matrices. The choice of L and s is informed by the inherent wavelength scaling in each problem such that the data inside each $s \times s$ voxel (equivalently, each sub-matrix at the highest resolution $\ell = L$) is non-oscillatory, i.e. contains a fixed amount of oscillations.

Following the Tensorflow convention of `[height, width, channels]` we reshape these quad-trees into three-tensors of size $[2^\ell,\ 2^\ell,\ s^2]$ as shown in Fig. 4. The first two dimensions of the tensor contain the geometrical information, and the last contains a local representation. In fact, the data describing the local representation inside each voxel corresponds to *channels*. In addition, we refer to slices along the height and width dimensions, i.e. the geometrical dimensions, as *patches*, e.g. a 1×1 patch of data describes slices with dimension $[1,\ 1,\ s^2]$. At the highest spatial resolution `WideBNet` operates on 1×1 patches, and, as we discuss shortly, at the lowest spatial resolution it operates on $2^{L/2} \times 2^{L/2}$ patches.

For the purpose of describing the operation in terms of linear algebra, it is convenient to view these three-tensors as reshaped two-tensors of size $[4^\ell,\ s^2]$ as depicted in Fig. 4; this flattening proceeds according to a natural ordering of quad-trees known as "Morton-ordering" or "Z-ordering". We refer to [62] for more details.

The input data $\Lambda_{s,r}(\omega) \in \mathbb{C}^{n_{\text{src}} \times n_{\text{rcv}}}$ is a function of the probe frequency ω in (2.2). To stabilize the inverse problem it is necessary for the input data to be collected from a wideband of frequencies $\Omega = [\omega_{\text{low}},\ \omega_{\text{high}}]$. We separate this bandwidth using a dyadic partition containing $L/2 + 1$ intervals: for $L/2 \le \ell \le L$ we label the intervals $\Omega^\ell = (\omega_{\text{low}} + 2^{L-\ell-1}\Delta\omega,\ \omega_{\text{low}} + 2^{L-\ell}\Delta\omega]$ where $\Delta\omega = \omega_{\text{high}} - \omega_{\text{low}}$. Within each interval Ω^ℓ we sample n_ω^ℓ frequencies, not necessarily equispaced, and with slight abuse of notation denote the resulting dataset as $\Lambda_{s,r}^\ell \in \mathbb{C}^{n_{\text{src}} \times n_{\text{rcv}} \times n_\omega^\ell}$. Following the quad-tree structure we reshape each data tensor $\Lambda_{s,r}^\ell$ into a three-tensor of size $[2^\ell,\ 2^\ell,\ n_\omega^\ell s^2]$, i.e. we concatenate all the multifrequency data collected from bandwidth Ω^ℓ along the channel dimension. The input to `WideBNet` thus consists of the collection $\{\Lambda_{s,r}^\ell\}_{L/2 \le \ell \le L}$.

3.2. Architecture Overview. We start by recalling that we aim to build an operator that emulates (2.10), i.e., for a set of given frequencies $\{\omega_\ell\}_{\ell=L/2}^L$ we seek a

presentation, our discussion focuses exclusively on this case.

network based on the discrete imaging condition

$$\sum_{\ell=L/2}^{L} \alpha(\omega_\ell) \left((F^{\omega_\ell})^* F^{\omega_\ell} + \epsilon I \right)^{-1} (F^{\omega_\ell})^* \Lambda^\ell_{\mathbf{s},\mathbf{r}}. \tag{3.1}$$

In the monochromatic case, we know that each operator can be written as

$$\left((F^{\omega})^* F^{\omega} + \epsilon I \right)^{-1} (F^{\omega})^* = A_{\texttt{conv}} A_{\texttt{butterfly}} \tag{3.2}$$

where $A_{\texttt{conv}}$ is a translation invariant operator and $A_{\texttt{butterfly}}$ accepts a factorization similar to

$$A_{\texttt{butterfly}} = U^L G^{L-1} \dots G^{L/2} S^{L/2} H^{L/2} \dots H^{L-1} V^L. \tag{3.3}$$

In the linear case, $A_{\texttt{butterfly}} \in \mathbb{C}^{4^\ell s^2 \times 4^\ell s^2}$ with $U^\ell \in \mathbb{C}^{4^\ell s^2 \times 4^\ell r}$, $V^\ell \in \mathbb{C}^{4^\ell r \times 4^\ell s^2}$, and all remaining factors of size $4^\ell r \times 4^\ell r$, where the inputs are supposed to be morton-flattened as mentioned above.

A schematic diagram of `WideBNet` is shown in Fig. 8 complemented by the pseudo code in Alg. 1. At a high level, `WideBNet` consists of $L+3$ specialized layers that are non-linear analogues of the butterfly factors[9] of $A_{\texttt{butterfly}}$ in (3.3) where $S^{L/2}$ is replaced by a SWITCH-RESNET module. These layers ultimately send data into a CNN module, which corresponds to $A_{\texttt{conv}}$, that is intended to mimic the effects of the regularized pseudo-inverse in sharpening the image/estimate.

The main contribution of `WideBNet` is in how multi-frequency datasets[10] are assimilated by exploiting the connection between spatial resolution and frequency in wave-scattering problems. In particular, we stress that it is both

(i) the connectivity/permutations inside the specialized layers $\{H^\ell\}$ and $\{G^\ell\}$ that process the wideband data, as well as,

(ii) the non-linearities induced by the middle SWITCH-RESNET layer,

that are crucial towards achieving stable training dynamics, as well as image super-resolution[11].

By choosing a dyadic partition of the frequency band `WideBNet` exploits the inherent multiscale nature of the $\{H^\ell\}$ layers. Analogous to their butterfly factorization namesakes, each $\{H^\ell\}$ layer only locally interpolates the data over voxel patches of effectively $2^{L-\ell} \times 2^{L-\ell}$, i.e. the effective length scales at this layer are of order $2^{L-\ell}$. It follows from the dispersion relation in wave-scattering that only data from bandwidth Ω^ℓ are informative at this length-scale[12]. As mentioned in Section 2.3 this strategy of dyadically partitioning the bandwidth to localize spatial information is employed by the Cooley-Tukey FFT algorithm to achieve quasi-linear time complexity [27]; in our setting this strategy affords us significant reductions in the number of trainable weights in the network.

We now proceed with a detailed discussion of each layer following Alg. 1. In what follows we compare each `WideBNet` layer to their analogous matrix factor from the butterfly factorization in (3.3).

[9] For ease of comparison we retain the transpose $\cdot^*$ in our naming conventions but note that transposition is no longer well defined in the non-linear setting.

[10] If only data at a single frequency is provided, i.e. using solely $\Omega^\ell_{r,s} \in \mathbb{C}^{n \times n \times n^\ell_\omega}$ with $n^\ell_\omega = 1$ as the input, then the network reduces to an equivalent `BNet` [60] or `SwitchNet` [54] networks.

[11] It is know that successful estimators for super-resolution must necessarily involve non-linear combinations of *wideband* data [34].

[12] The $\{G^\ell\}$ layers have a similar multiresolution property. This suggests that data from bandwidth Ω^ℓ should also be fed into G^ℓ similar to the U-Net [76] architecture; however, numerical results demonstrate that this additional complexity is unnecessary.

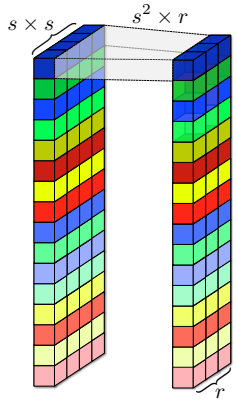

FIG. 5. *Sketch of the compression carried in the V^L layer, from the points contained in a leaf of size $s \times s$ (see Fig. 4) to a local representation of rank r. The grey polygon represent the connections between the two layers.*

3.3. U^L and V^ℓ layers. The V^ℓ factor in (3.3) represents a block diagonal matrix with block size $r \times s^2$ as shown in Fig. 5. This operator takes input data (viewed as a complete quad-tree) and compresses leaf nodes at level L, each with $s \times s$ degrees of freedom, into 1×1 patches with $\sqrt{r} \times \sqrt{r}$ degrees of freedom. Similarly, the U^L factor in (3.3) is also block diagonal except with block sizes of $s^2 \times r$. This operator "samples" the local representation back to its nominal dimensions. In both instances the compression/decompression is essentially lossless provided the number of levels L is properly adapted to the probe frequency ω. We emphasize again that this follows as a consequence of the *dispersion relation* in wave-scattering: provided these parameters are chosen correctly, then over $s \times s$ length scales the data is non-oscillatory (i.e. sub-wavelength) and therefore admits a low-rank representation with rank r.

`WideBNet` also exploits this relation between spatial resolution and frequency. However, a key point of departure from the butterfly factorization is that here the input data is wideband and thus contains multiple length scales (wavelengths). This motivates the introduction of auxiliary layers V^ℓ for $L/2 \le \ell \le L$ whose inputs are assumed to be sampled from bandwidth Ω^ℓ. Each V^ℓ layer compresses the input data *at level* ℓ such that nodes with $2^{L-\ell}s \times 2^{L-\ell}s$ degrees of freedom are mapped into 1×1 patches with $\sqrt{r} \times \sqrt{r}$ degrees of freedom; this also has the interpretation of spatial downsampling. Note that the dyadic scaling in the definition of Ω^ℓ is critical in maintaining the balance between spatial resolution and frequency.

When the input data $\Lambda^\ell_{s,r}$ from bandwidth Ω^ℓ is represented as a three-tensor of dimension $[2^\ell,\ 2^\ell,\ n^\ell_\omega]$, each V^ℓ layer can be implemented as a `LocallyConnected2D` layer in Tensorflow with rn^ℓ_ω channels and both the kernel size and stride as $2^{L-\ell} \times 2^{L-\ell}$. Note this coincides with V^ℓ butterfly factor when $l = L$ and $n^\ell_\omega = 1$. The U^L layer can also be implemented as `LocallyConnected2D` layer with rank s^2 and 1×1 kernel size and stride; the input to this layer is assumed to be of dimension $[2^\ell,\ 2^\ell,\ c]$ with c input channels.

```
def V^ℓ(X):
  # input:  [?, 2, 4^L, s^2]
  # output:  [?, 2, 4^L, rn_{ω_ℓ}]
```

```
  y_re, y_im = X[:,0,:,:], X[:,1,:,:]

  x_re = LC1D[rn_ω_ℓ,1,1](y_re) + LC1D[rn_ω_ℓ,1,1](y_im)
  x_im = LC1D[rn_ω_ℓ,1,1](y_re) + LC1D[rn_ω_ℓ,1,1](y_im)

  return tf.stack([x_re, x_im], axis=1)
```

LISTING 2

Pseudo code for the V^ℓ module. where `LCN[a,b,c] == LocallyConnected1D(filters=a, kernel_size=b, strides=c)`

```
def U(X):
  # input:  [?, 2, 4^L, ·]
  # output:  [?, 2, 4^L, s^2]
  y_re, y_im = X[:,0,:,:], X[:,1,:,:]

  x_re = LC1D[s^2,1,1](y_re) + LC1D[s^2,1,1](y_im)
  x_im = LC1D[s^2,1,1](y_re) + LC1D[s^2,1,1](y_im)

  return tf.stack([x_re, x_im], axis=1)
```

LISTING 3

`LCN[a,b,c] == LocallyConnected1D(filters=a, kernel_size=b, strides=c)`

Remark: A major application of the butterfly factorization is for applying FIOs in linear-time complexity; the rank r then depends on the error tolerance but generally requires that $r \ll s^2$. This represents a significant philosophical difference in how in r is determined in our machine learning setting – it does not matter if $r \geq s^2$ so as long as the learned model achieves its intended task. Nevertheless, numerical results show that *(i)* it suffices to choose $r \ll s^2$, and, moreover, that *(ii)* generalization is largely insensitive to the choice of r.

3.4. H^ℓ and G^ℓ layers. The H^ℓ and G^ℓ factors in (3.3) continue the theme of multiscale processing. When viewed as matrices, both H^ℓ and G^ℓ are block diagonal with block size $4^{L-\ell}r \times 4^{L-\ell}r$. Equivalently, when the input is formatted as a complete quad-tree, this implies both are *local operators* which process the nodes on the tree at length scale l to map each $2^{L-\ell}s \times 2^{L-\ell}s$ patches. Within each block there is further structure to the operators, as Figure 6 demonstrates. For each H^ℓ each sub-block has the interpretation of *aggregating* information, whereas each G^ℓ achieves the dual task of *spreading* information. We stress, however, that the action of this is entirely local within each patch. In either case the key observation is that by permuting each nodes following a set pattern each operator becomes block-diagonal with block size 4^ℓ, for all $L/2 \leq \ell \leq L$.

Each G^ℓ layer mimics the behaviour of their counterparts can be be implemented using the LOCALLYCONNECTED2D layer with 4×4 kernel sizes and stride 4. The number of channels is $\sum_{i=l+1}^{\ell} rn_{\omega_i}$ for symmetry.

The H^ℓ layers differ in that they process two inputs: one the output of the V^ℓ layer of dimension $[2^{L-\ell}, 2^{L-\ell}, rn_{\omega_\ell}]$, the other the output from the previous layer of dimension $[2^\ell, 2^\ell, c]$ for some channel size c. To process the dimensions of both we first upscale each patch with redundant information to convert the data into $[2^\ell, 2^\ell, rn_{\omega_\ell}]$. Then this is concatenated with the other dataset to form a tensor of size $[2^\ell, 2^\ell, c + rn_{\omega_\ell}]$.

The ordering of the concatenation along the channel dimension does not matter so as long as it is performed consistently.

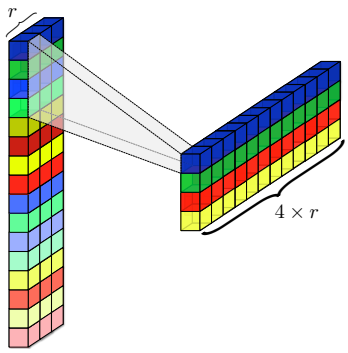

FIG. 6. *Sketch of the application of the H^ℓ layer. The layer decimates by a factor of four the number of neurons in the spatial dimension, while increasing four times the number of channels. From Fig. 4 we can observe that the decimation is equivalent to decimate by a factor two in each of the first two dimensions, which follows from the Z-ordering.*

```
def H^ℓ(X, W):
  # input X: [?, 2, 4^L, ·]
  # input W: [?, 2, 4^L, ·]
  # output:  [?, 2, 4^L, r∑_{i=ℓ}^{L} n_{ω_i}]

  X̃ = UpSampling2D[·, ·](X̃)

  X = tf.stack([X, X̃], axis=-1)

  y_re, y_im = X[:,0,:,:], X[:,1,:,:]

  y_re = permute_dim(y_re, level=ℓ)
  y_im = permute_dim(y_im, level=ℓ)

  x_re = LC1D[4r∑_{i=ℓ}^{L} n_{ω_i},4,4](y_re) + LC1D[4r∑_{i=ℓ}^{L} n_{ω_i},4,4](y_im)
  x_im = LC1D[4r∑_{i=ℓ}^{L} n_{ω_i},4,4](y_re) + LC1D[rn_{ω_ℓ},1,1](y_im)

  return tf.stack([x_re, x_im], axis=1)
```

LISTING 4

LCN[a,b,c] == LocallyConnected1D(filters=a, kernel_size=b, strides=c)

3.5. Switch-Resnet layer. We retain the permutation pattern of the switch layer as this is responsible for capturing the inherent non-locality of wave scattering (e.g. a point scatterer generates a diffraction pattern that is measured by all receivers in our geometry). We illustrate this pattern in Fig. 7.

The input this level serves as a condensed representation of the measured data. It is at this level that we non-linearly process the multifrequency dataset; we speculate that this also essential in facilitating the model to produce super-resolved images. We achieve this using a residual network to refine each channel locally following each resnet unit. The pseudocode is provided in Alg. 12.

```
def SwitchResnet():
  # input
  # output
```

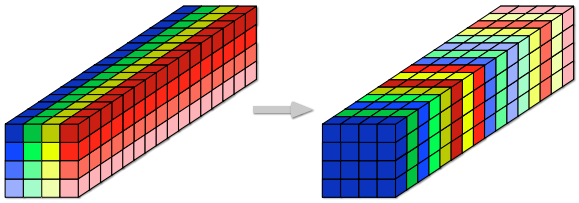

FIG. 7. *Sketch of the permutation of information of at the Σ layer. In this case we consider that $r = 1$, and that $L = 4$. In this case the tensor at the middle level is $[2^{L/2}, 2^{L/2}, r4^{L/2}] = [4, 4, 16]$. The last dimension is the number of channels, in this case the information contained at each geometric position is distributed along all the positions.*

```
# apply switch permutation
for kk in range(N_resnet):
  y = LCD1D(ReLu(LC1D(y)) + y

  if kk < N_resnet:
    y = ReLu(y)
```

LISTING 5

LCN[a,b,c] == LocallyConnected1D(filters=a, kernel_size=b, strides=c)

3.6. `WideBNet` Parameter Count. An estimate of how the number of parameters (i.e. trainable weights) scales is

$$\text{d.o.f.}(\texttt{WideBNet}\ ) \approx 4^{\ell} r^2 \left(\sum_{l=L/2}^{\ell} n_{\omega_\ell}^2 + \sum_{l=L/2}^{\ell} \left(\sum_{i=l}^{\ell} n_{\omega_i} \right)^2 \right).$$

When only a single frequency is sampled in each sub-band, i.e. $n_{\omega_\ell} = 1$ for all l, then this total becomes $\mathcal{O}\left(N(\log N + \log^3 N)\right)$. Note this is essentially linear in the total degrees of freedom in the data (N) up to polylogarithmic factors. Furthermore, note if naïvely L separate single channel `WideBNet` networks were used to compute (2.10) this would correspond to complexity $\mathcal{O}\left(N \log N^2\right)$; the multifrequency assimilation only exceeds this with mild oversampling by a logarithmic factor.

Lastly, we note the effect of the partitioning of the frequencies. If all the frequencies were ingested at length scale L then the scaling becomes $\mathcal{O}\left(N(\log N^2 + \log N^3)\right)$. While to leading order this presents the same asymptotic scaling, in terms of practical considerations this presents as substantial increase in the number of trainable parameters.

4. Numerical Results. Synthetic data was generated using numerical finite differencing for (2.2) over the computational domain $[-0.5, 0.5]^{\otimes 2}$. The domain was discretized with an equispaced mesh of $n_x = 80$ by $n_z = 80$ points which corresponds to a quad-tree partitioning into $L = 4$ levels with leaf size $s = 5$. Training data was generated using a second-order finite difference scheme while *testing* data was computed with fourth-order finite differences. The use of higher quality simulations for testing serves to validate that `WideBNet` predictions do not depend on computational artifacts such as e.g. numerical dispersion. The radiating boundary conditions for Eq 2.2 were implemented using perfectly matched layers (PML) with a quadratic

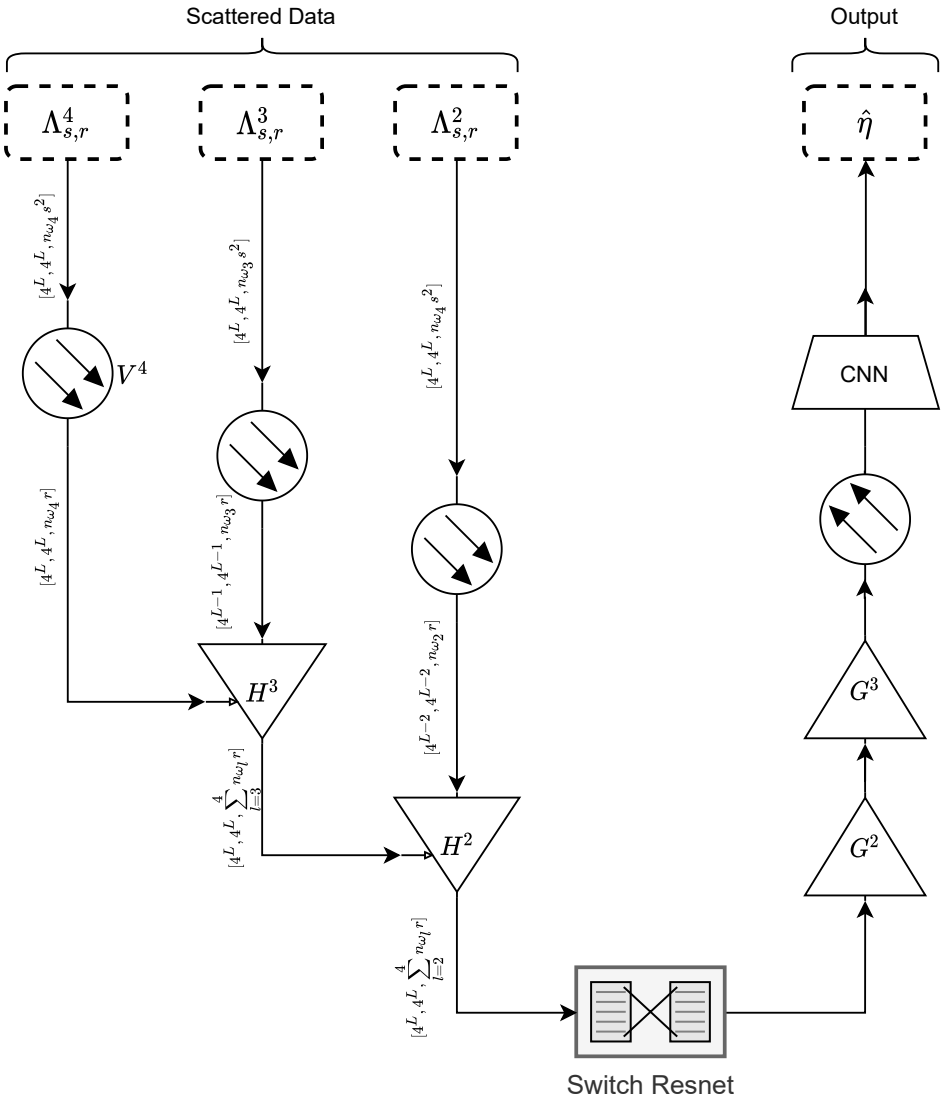

FIG. 8. *Diagram of* `WideBNet` *for data with* $L = 4$ *levels, leaf size* s*, and rank* r*.*

profile with intensity 80 [8]. The width of the PML was chosen to span at least one wavelength at the lowest frequency.

Unless specified otherwise the dataset consisted of $n_f = 3$ source frequencies at 2.5, 5, and 10 Hz. In a homogeneous background with velocity $c_0 = 1$ this corresponds to 8 points-per-wavelength (PPW) at the highest frequency. Receivers were located at equi-angular intervals around a circle of radius $r = 0.5$ with the recorded data computed by linearly interpolating the scattered field. We used $N_{\text{rcv}} = 80$ receivers and sources for all experiments. For a homogeneous background the direct wave is given analytically (see (2.2)). In these instances the directions of arrival $\mathbf{s} \in \mathbb{S}^1$ were

aligned with the receiver geometry, i.e. incident from 80 equiangular directions. However for inhomogeneous media the direct waves had to be computed numerically. This was achieved by using numerical Dirac deltas as source functions. These sources were localized on a circle of radius $r = 1$ at 80 equiangular intervals and the computational domain was extended to $[-1, 1]^{\otimes 2}$ using the same grid spacing Δx and Δz as before. The resulting scattered field was computed by differencing the solutions to (2.2) with and without scatters. The acquisition geometry was fixed for all frequencies.

Scatterers were selected from a dictionary of simple, convex, geometric objects such as squares, triangles, and gaussian bumps. The characteristic lengths of the square and triangular scatterers were measured with respect to their base, rather than the diameter of the smallest enclosing ball, whereas the characteristic length of the gaussian was taken to be its standard deviation. In each data point the number of scatterers was determined by uniformly sampling from $\{2, 3, 4\}$ objects, and their locations were uniformly distributed inside a circle of radius $r = 0.35$. No restrictions were enforced against overlapping scatterers. In all experiments the amplitude of each scatterer was fixed to 0.2; we leave to future work how the training data can be augmented to account for variations in amplitudes.

`WideBNet` was implemented in Tensorflow [1] and trained with the pixelwise sample loss function

$$\sum_{x=\text{pixel in image}} \left\| (K_{\text{high}} * \eta)(x) - \texttt{WideBNet}\ [\Lambda^{L}_{s,r}, \ldots, \Lambda^{L/2}_{s,r}](x) \right\|_2^2, \tag{4.1}$$

where η denotes the sample realization of the scatterer wavefield and $\{\Lambda^{\ell}_{s,r}\}_{L/2 \le l \le L}$ the partitioned multifrequency data. This objective function was chosen to promote the recovery of an image that is *smoother* than the reference image by a factor of a two dimensional convolution with high-pass filter K_{high}. Critically we still remain in the super-resolution regime when the support of filter K_{high} is significant smaller than the Nyquist limit of $\lambda_{min}/2$[13] as the smoothed image still contains sub-wavelength features. This strategy was inspired by the work of [17] who relied on this insight for theoretical proofs on recoverability limits in superresolution. In our experiments we selected K_{high} to be a Gaussian kernel with characteristic width of 0.75 grid points (compare this the diffraction limit in our bandwidth of 4 pixels). This smoothing was observed to be integral in promoting stable training dynamics. We also report the the image-wise relative error

$$\frac{\left\| K_{\text{high}} * \eta - \texttt{WideBNet}\ [\Lambda^{L}_{s,r}, \ldots, \Lambda^{L/2}_{s,r}] \right\|_2^2}{\| K_{\text{high}} * \eta \|_2^2}. \tag{4.2}$$

Note that we do not normalize the norms in either (4.1) or (4.2) by the grid lengths Δx and Δz.

The training and testing split was 21000 points and 4000 points[14], respectively, with batch size 32. Note, in comparison, an instance of `WideBNet` with $N_{\text{CNN}} = 3$ convolutional layers and $N_{\text{RNN}} = 3$ residual layers contains 200000 trainable parameters meaning our models are still in the massively over-parameterized regime. Unless specified otherwise the testing set follows the same distribution (e.g. scatterer types) as the training set. The initial learning rate (i.e. step size) was universally set

[13] The ratio of these two quantities is the so-called *super-resolution factor*.

[14] a single "data point" has dimension $n_x \times n_z \times n_f$

to 5e-3 across all experiments. The learning schedule was set according to Tensorflow's [1] implementation of `ExponentialDecay` with a decay rate of 0.95 after every 2000 plateaus steps with stair-casing. We chose the Adam optimizer [55] and terminated training after 150 epochs. No special initialization strategy was required and the network weights were randomly initialized with `glorot_uniform` – we did not observe the training instabilities with random initialization that were thoroughly documented in [84] for general butterfly networks. All computations were done with `float32` half-precision. Note that no effort was taken to optimize these hyper-parameters using an external validation set.

4.1. Homogeneous Background. In this section we present numerical results for `WideBNet` models trained with scattered data that propagated through a known homogeneous background medium of wavespeed $c_0 = 1$. Each row of Figure 11 depicts `WideBNet` predictions on testing data across a variety of scatterer configurations. With the exception of Figure 11c the data was sampled from the bandwidth of 2.5, 5 and 10 Hz which implies a limiting wavelength of 8 PPW. Figures 11a and 11b involve a multiscale dictionary of scatterers with characteristic lengths ranging from 3, 5, and 10 pixels; these correspond to the sub-wavelength, wavelength, and super-wavelength regimes, respectively. We observe that `WideBNet` correctly localizes each scatterer in addition to resolving sub-wavelength features such as e.g. the corners of the triangles. Figure 11d similarly depicts a heterogeneous dictionary but with rotated triangles of fixed sidelength 5 pixels. These experiments demonstrate that `WideBNet` performs a far more complex signal processing task beyond joint blind deconvolution with point super-resolution. In Figure 11c the same experiment was repeated but with a bandwidth that was shifted to 1.25, 2.5 and 5 Hz so that the limiting wavelength increases to 16 PPW; in this regime all scatterers are sub-wavelength. Nevertheless, `WideBNet` still produces images that are qualitatively comparable to the higher bandwidth experiments. This suggests that our algorithm has a high super resolution factor. For completeness we include results in Figure 11e for point scatterers that were originally proposed for super-resolution by Donoho [34].

Table 2 summarizes the training and testing loss for various scatterer configurations. Each row corresponds to a separate experiment with triangular ($\triangle$), square ($\square$), or gaussian ($\bigcirc$) scatterers. The numbers in the parentheses correspond to the characteristic length, in pixels, with multiple numbers indicating a multiscale dataset.

Several trends can be observed from this table. In all configurations there is no evidence of overfitting; indeed, the generalization gap, defined to be the difference between the testing and training errors, is on average less than an order of magnitude. Furthermore, both qualitatively and quantitatively there is no significant difference between datasets with a fixed characteristic length versus the multiscale datasets. This demonstrates robustness to the choice of the scatterer dictionary. However, we observe that gaussian scatterers outperform other shapes across all metrics, perhaps owing to their smoothness. Surprisingly in testing the pixelwise error tends to decrease with decreasing length scale; we conjecture the exact scaling may depend on the perimeter to area ratio of the polygons.

4.1.1. Effect of Switch Layer. In Section 3 we emphasized the importance of the `switch` permutation pattern in representing the local-to-global physics of wave scattering. Figure 13 gives weight to this claim by comparing the predictive ability of `WideBNet` models trained with and without the inclusion of the `switch` permutation layer. Critically all other configurations were held equal with both models containing the same number of trainable weights.

TABLE 1
Effect of frequency partition

		Pixel-wise Squared Loss		Image-wise Relative Loss	
	DOF	Train	Test	Train	Test
AllFreq	2746368	2.92E-06	4.81E-06	1.26E-05	1.72E-05
MultiFreq	1913856	4.06E-06	6.40E-06	1.72E-05	2.27E-05

Figure 13 demonstrates that the predictions *without* the permutation layer are of noticeably poorer quality. However, the switch-less configuration manages to localize scatterers and even reproduces subwavelength features to an extent, particularly when the scatterers are well separated as in Figure 13(b). However, Figures 13(c) and (d) exposes the deficiencies of this model in the presence of overlapping scatterers. From a physical perspective this is unsurprising as colliding scatterers generate sub-wavelength diffraction patterns which complicates the imaging problem. These complication appear to be remedied by the inclusion of the switch permutation layer. Although the switchless configuration manages to produce reasonable images, we conjecture that this is because the model is "reasonably deep" at this length scale. We suspect the predictive abilities will quickly deteriorate as $L \to \infty$ since the depth of the network only scales linearly as $\Theta(L)$.

4.1.2. Partitioning of frequencies. Table 1 reports on the difference between two competing frequency partitioning strategies: "AllFreq", in which the data from the entire bandwidth is fed into `WideBNet` at level L, versus "MultiFreq" wherein the data is only processed at the appropriate length scale ℓ. Qualitatively both strategies produce comparable images that are sharp and resolve the sub-wavelength features. In fact, quantitatively the "AllFreq" strategy produces marginally lower losses (though within the same order of magnitude). However, as noted in Table 1 that the degrees of freedom of "AllFreq" far exceed that of "MultiFreq"; although both strategies have the same asymptotic storage complexity of $\mathcal{O}(N \log^3(N))$ (see Section 3.6), practically speaking the constant differs by a substantial amount in favour of "MultiFreq".

We report that we were unable to successfully train a model by mimicking (2.10) directly, i.e. training single channel `WideBNet` models for each frequency independently, then merging their predictions via a CNN module. This is perhaps unsurprising since it is known that super-resolution algorithms require non-linear synthesis of multi-frequency data to succeed. Whereas in both "MultiFreq" and "AllFreq" this is achieved by the switch-resnet module, in this naive strategy the synthesis is performed only at the end by the CNN layers. In comparison to the optimal storage complexity of $\mathcal{O}(N \log^2 N)$ in this naïve strategy, note that mildly overparametrizing by a small logarithmic factor provides significant training stability to the inverse problem.

4.1.3. Training Curves & Hyper-Parameter Sensitivity.

Training Curves. Figure 10(a) reports the training errors for models trained on datasets containing 5000, 10000, 15000, and 21000 datapoints. The trained models were evaluated on a *fixed* testing set of 3000 points (i.e. the same testing set is to compare all experiments). All remaining hyperparameters such as the learning rate and number of epochs were held the same as discussed in the beginning of Section 4. Note in all cases we remain in the over-parametrized regime since the number of

datapoints is far fewer than the number of degree of freedom. Nevertheless with only a few samples `WideBNet` stably achieves a pixel-wise loss on the order of 10^{-5}.

We observe in Figure 10 that both training and testing errors decrease with increasing training points, as expected. However these training/testing curves quickly saturate and the differences fall less than an order order of magnitude. Furthermore, the empirical generalization gap, taken to be the difference between the testing curve (dashed lines) and the training curve (solid line) remains within the same order of magnitude as the number of points is increased. These points demonstrate that our model *(i)* generalizes with relatively scant training points, and *(ii)* saturates its model capacity quickly, which is an indication that the architecture is well adapted to the task.

Sensitivity to the rank r. While the data essentially specifies the architecture through requirements on the level L and leaf size s, it remains up to the user to select the rank r. We reiterate that this choice serves as a significant departure from the numerical analysis perspective of the Butterfly factorization; whereas in the original context it is essential to have the scaling $r \ll s^2$ for the purpose of fast matrix-vector multiplication[15], in the current machine learning context there is no restriction against choosing $r \geq s^2$. Nevertheless, as Figure 10b demonstrates, a large over-parameterization with respect to r is unnecessary. Indeed, while the training metrics monotonically decrease as the model capacity increases with rank, we observe that testing errors remain relatively saturated. This suggests that performance of `WideBNet` is largely insensitive to the rank and the network topology plays a more significant role.

Moreover, these results indicate that that allowing for a non-uniform rank for each patch may not yield be a fruitful exercise. Or, conversely, if the intent is to compress the model further to e.g. fit on mobile devices [9], this also suggests that tenable strategy may be to prune a trained model by adaptive patch-wise rank reduction. We leave this to future work.

Effect of CNN and ResNet Layers. Beyond the selection of the rank r the only remaining hyper-parameters that determine the `WideBNet` architecture are the number of CNN layers N_{CNN} and the number of residual layers N_{RNN} in the switch module. Figure 14 reports on the sensitivity of the `WideBNet` model to these parameters. Evidently from Figure 14b we conclude that the predictive performance is unaffected by the number of post-processing CNN layers. A similar conclusion can be drawn about the number of residual layers from Figure 14a; note the fluctuations in the training and testing curves are negligible in magnitude.

4.2. Heterogeneous Background. In this section we present numerical results with scattering data from a known *inhomogeneous* background medium. The variations in the background wavespeed introduce significant complications to the inverse problem. For instance, homogeneous backgrounds afford symmetries such as rotational equivariance which can be exploited for efficient network design, see e.g. [40]; in an inhomogeneous background this assumption is no longer valid. The physics of wave propagation through inhomogeneous media also complicates the signal processing probme as it gives rise to multi-pathing as well as multiple arrivals due to interior scattering. While the architecture and data formatting remain unchanged, the complexity of the inverse problem for localizing scatterers, let alone super-resolution,

[15]Typically the rank is determined by computations of SVDs so that ϵ is close to be machine zero. Analytical relations between ϵ and r are kernel dependent and is known explicitly only in few cases.

TABLE 2

Training and testing errors for various datasets. Each experiment consisted of 21000 training points and `WideBNet` *was evaluated against an independent testing dataset with 3000 points. The data was generated using a homogeneous background wavefield $c_0 = 1$ and and data was sampled at 2.5, 5, and 10 Hz (i.e. the effective wavelength was 8 PPW). Each row denotes a separate experiment with the scatterers comprising of triangles (△), squares (□), and gaussians (◯). The numbers in the parentheses indicate the characteristic lengths of each scatterer; multiple numbers indicate a heterogeneous dataset of scatterer sizes, while (rot,5) indicates a dataset with rotated scatterers. In general we observe that* `WideBNet` *does not overfit the data. Surprisingly, on average the testing pixel-wise error decreases with decreasing lengthscale.*

	Pixel-wise Squared Loss		Image-wise Relative ross	
Scatterer	Train	Test	Train	Test
△ (3,5,10)	4.06E-06	6.40E-06	5.38E-04	7.12E-04
□ (3,5,10)	7.12E-04	1.13E-05	4.63E-04	6.24E-04
◯ (3,5,10)	1.24E-06	2.01E-06	1.89E-05	2.71E-05
△ (rot,5)	3.03E-06	4.09E-06	5.52E-04	7.32E-04
△ (10)	2.47E-06	2.51E-05	9.26E-05	8.17E-04
△ (5)	1.14E-06	7.19E-06	2.11E-04	1.24E-03
△ (3)	4.35E-06	4.23E-06	2.62E-03	2.62E-03
□ (10)	2.63E-06	7.92E-05	4.90E-05	1.24E-03
□ (5)	1.24E-06	2.09E-05	1.13E-04	1.75E-03
□ (3)	1.19E-05	1.19E-05	3.77E-03	3.80E-03
◯ (3)	9.89E-08	2.61E-06	5.97E-06	1.30E-04
◯ (2)	3.19E-07	4.84E-07	4.35E-05	6.28E-05
◯ (1)	5.86E-07	7.52E-07	4.71E-04	5.87E-04

increases in this setting.

We tested the algorithm for two heterogeneous backgrounds: (i) a smooth linearly increasing background medium with wavespeed $c = 0.5$ at the top and $c = 1.5$ at the bottom, and (ii) layered background medium with wavespeeds $c = 1$, $c = 2$, and $c = 4$. The results of trained `WideBNet` models on testing data are shown in Figure 12. We observe in Figure 12(b) that `WideBNet` manages to process the multiple arrivals to image the triangular scatterers. However, surprisingly, it does significantly poorer for the smoothly varying background. Explaining this discrepancy remains an open problem.

Remark: The notion of resolution becomes ambiguous for inhomogeneous medium as the wavelength changes with background medium $c(x, z)$ following the dispersion relation in (1.1). Nevertheless across the range of background velocities the scatterers still contain sub-wavelength features such as e.g. the corners.

4.2.1. Comparison versus FWI. In order to make the comparison fair, we use FWI implemented in PySIT [47] to invert the same perturbation. We start with a homogeneous background. Given the non-convexity of the objective function, we use a frequency sweep following standard practices in the geophysical community. We tested a dozen of different combinations, and we picked the two that produced the best images. In particular, we used two typical strategies for the frequency sweeps. In each strategy we use three stages in which data at different frequency is fed to the optimization loop. In the first strategy, we run the optimization routine for the lowest frequency data available, then progressively add higher frequency data at each stage. In the second strategy, at each stage we process data only at a certain frequency, without combining them, but we use the guess at the end of one stage as the initial guess for the next one: in the first stage we process the lowest frequency data, we save the final answer which will be used as an initial guess for the next stage, which will process data in the immediately higher frequency-band, and we repeat until data at all frequencies are processed.

We let the optimization run until the residue stagnates at around 10^{-6}, which is roughly after 20 iterations per stage. Both strategies provide similar results as shown in Fig. 9. We can observe that the reflectors are properly placed but the result from the neural network provides a better localization with far fewer oscillatory artifacts. In addition, `WideBNet` outperforms FWI in term of the relative ℓ^2 error of the reconstructions, which is 0.0013 for `WideBNet` compared to 0.0151, and 0.0141, for the first and second strategy, respectively.

We point out that procuring these images for FWI was labor- and time-intensive. It took roughly one week to test all the different frequency sweeps and the full computation, performed in PySIT [47], took roughly eight hours in a 16 core workstation with an AMD 2950X CPU and 128 Gb of RAM. In contrast, the training stage for `WideBNet` took in total 12 hours, and the inference takes a fraction of a second, running on a Nvidia GTX 1080Ti graphics card.

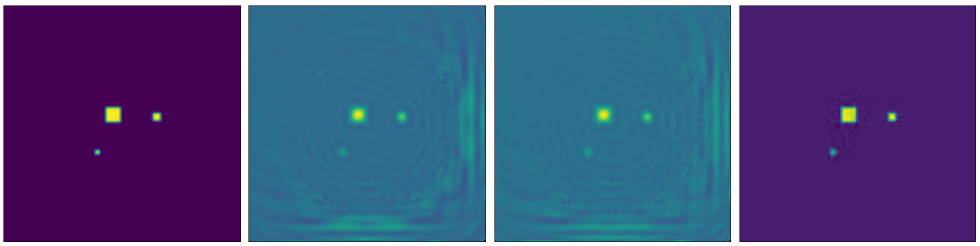

FIG. 9. *a) Original wavespeed to be reconstructed, b) result using the first strategy with three frequencies and* 20 *iterations of LBFGS at each set of frequencies following* $[\{1.25Hz : 20\ it\}, \{1.25Hz, 2.5Hz : 20\ it\}, \{1.25Hz, 2.5Hz, 5Hz : 20\ it\}]$*, c) result using the second strategy with three frequencies and* 20 *iterations of LBFGS at each set of frequencies following* $[\{1.25Hz : 20\ it\}, \{2.5Hz : 20\ it\}, \{5Hz : 20\ it\}]$*, d) result using* `WideBNet` *.*

5. Conclusion & Future Work. In this manuscript we have designed an end-to-end architecture that is specifically tailored for solving the inverse scattering problem. We have shown that by assimilating multi-frequency data and coupling them through non-linearities we can produce images that solve the inverse scattering problem. Our tool produces results which are competitive with optimization-based approaches, but at a fraction of the cost. More critically, we have demonstrated that our architecture design and data assimilation strategy avoids three known shortcom-

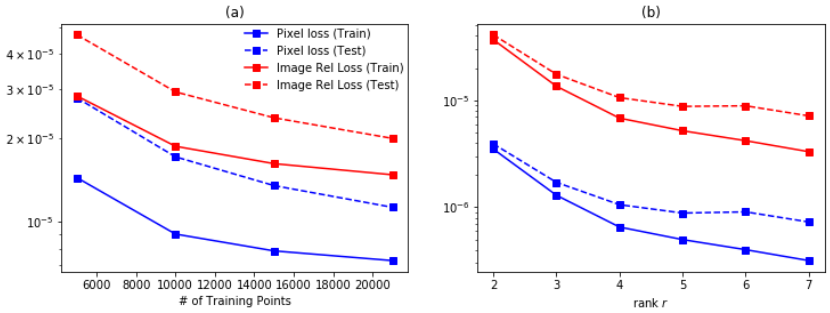

FIG. 10. *Sensitivity to hyper-parameters: number of training points and the rank r. (a) Performance of* `WideBNet` *with increasing number of training points. Note the same testing dataset, consisting of* 3000 *points, was used for all experiments. Observe that the generalization gap (the distance between the dashed and solid lines) remains asymptotically as both training and testing errors saturate. This exhibits that we are saturating the model capacity. (b) Performance of* `WideBNet` *with increasing rank r. The testing dataset was fixed for all experiments. Note that for leaf size $s = 5$ the maximum rank of the linearized model is $r_{max} = 25$. Although the training error decreases with increasing rank, we observe that the testing error beings to plateau beyond $r = 3$.*

ings with conventional architectures and also other butterfly-based networks: (i) by incorporating tools from computational harmonic analysis, such as the butterfly factorization, and multi-scale methods, such as the Cooley-Tukey FFT algorithm, we are able to drastically reduce the number of trainable parameters to match the inherent complexity of the problem and lower the training data requirements, (ii) our network has stable training dynamics and does not encounter issues such poorly conditioned gradients or poor local minima, and (iii) our network can be initialized using standard off-the-shelf technologies.

In addition, we have shown that our network recovers features below the diffraction limit of general, albeit fixed, class of scatterers. Even though there is an underlying assumption on the distribution of the scatterers we do not explicitly exploit it. Thus one future research direction is to use the current architecture within a VAE or GAN framework, to fully capture the underlying distribution, and to further study the limits of the current architecture to image sub-wavelength features. Following the same approach one can seek to extend the applicability of the current architecture to the cases where there is noise in signal, or uncertainty on the background medium.

Acknowledgments. We thank Yuehaw Khoo, Lexing Ying, Guillaume Bal, Yingzhou Li, Zhilong Fang, Pawan Bhawardarj, and Nori Nakata for fruitful discussions.

REFERENCES

[1] M. ABADI, A. AGARWAL, P. BARHAM, E. BREVDO, Z. CHEN, C. CITRO, G. S. CORRADO, A. DAVIS, J. DEAN, M. DEVIN, S. GHEMAWAT, I. GOODFELLOW, A. HARP, G. IRVING, M. ISARD, Y. JIA, R. JOZEFOWICZ, L. KAISER, M. KUDLUR, J. LEVENBERG, D. MANÉ, R. MONGA, S. MOORE, D. MURRAY, C. OLAH, M. SCHUSTER, J. SHLENS, B. STEINER, I. SUTSKEVER, K. TALWAR, P. TUCKER, V. VANHOUCKE, V. VASUDEVAN, F. VIÉGAS, O. VINYALS, P. WARDEN, M. WATTENBERG, M. WICKE, Y. YU, AND X. ZHENG, *TensorFlow: Large-scale machine learning on heterogeneous systems*, 2015, https://www.tensorflow.org/. Software available from tensorflow.org.

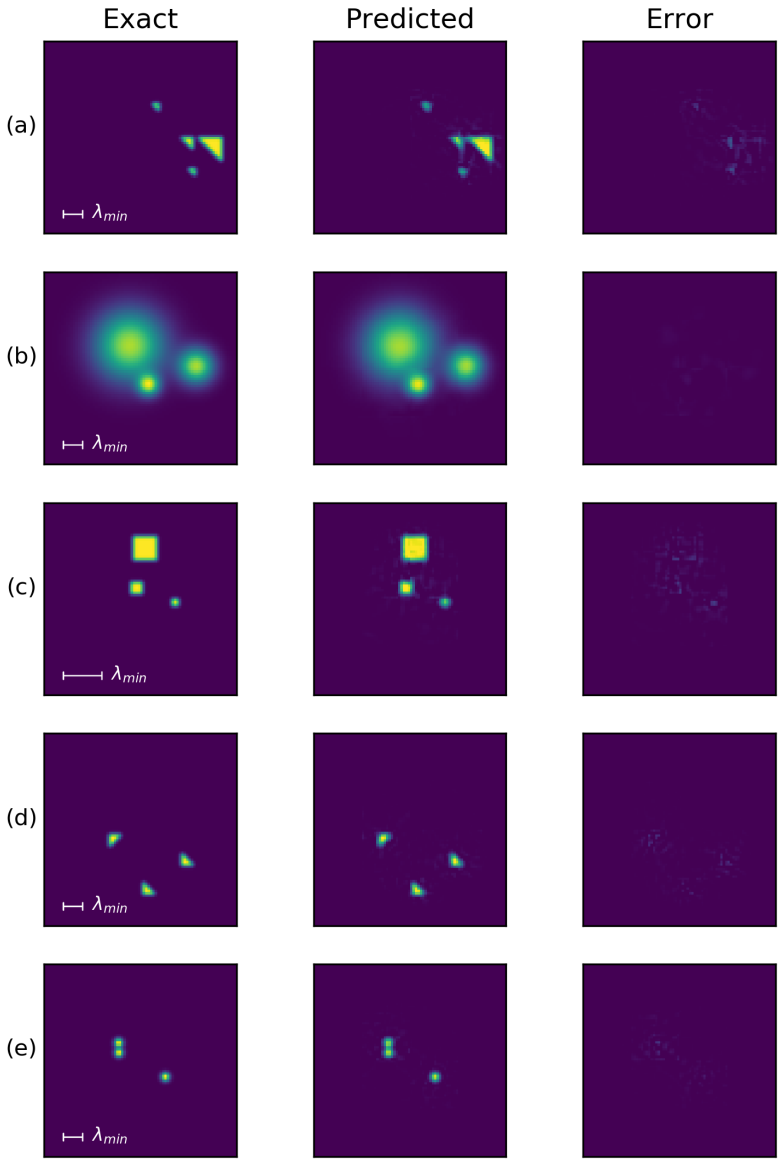

FIG. 11. *Visualization of* `WideBNet` *predictions on a testing set. The first column is the exact solution, the second column the output of* `WideBNet` *, and the third column the pointwise error. The colour scales in each row are normalized with respect to the first column. (a) with* $\triangle$ *(3,5,10). (b) same as above but with the Gaussian dataset. (c) heterogeneous squares but with a lower bandwidth (1.25, 2, and 5 Hz) so the effective wavelength is* 16 *PPW. (d) rotated dictionary. (e) gaussian scatterers with characteristic length* 1*.*

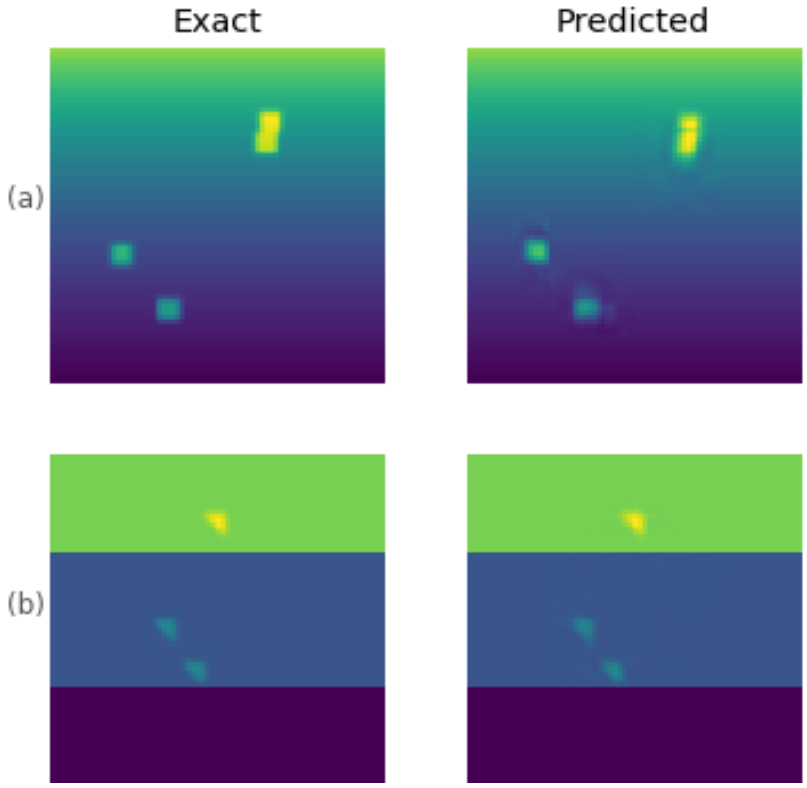

FIG. 12. *Visualization of* `WideBNet` *predictions on a testing set with inhomogeneous backgrounds. The colour scales of each row is normalized to the first column. The background medium is assumed known. (a) With a lineary increasing gradient in the background. (b) A layered medium increasing velocity with depth.*

[2] H. K. AGGARWAL, M. P. MANI, AND M. JACOB, *Modl: Model-based deep learning architecture for inverse problems*, IEEE Transactions on Medical Imaging, 38 (2019), pp. 394–405.

[3] T. ALKHALIFAH, *Scattering-angle based filtering of the waveform inversion gradients*, Geophys. J. Int., 200 (2014), pp. 363–373, https://doi.org/10.1093/gji/ggu379, https://doi.org/10.1093/gji/ggu379, https://arxiv.org/abs/http://oup.prod.sis.lan/gji/article-pdf/200/1/363/2231058/ggu379.pdf.

[4] D. ATKINSON AND N. D. APARICIO, *An inverse problem method for crack detection in viscoelastic materials under anti-plane strain*, Int. J. Eng. Sci., 35 (1997), pp. 841 – 849, https://doi.org/https://doi.org/10.1016/S0020-7225(97)80003-1, http://www.sciencedirect.com/science/article/pii/S0020722597800031.

[5] G. BACKUS AND F. GILBERT, *The Resolving Power of Gross Earth Data*, Geophys. J. Int., 16 (1968), pp. 169–205, https://doi.org/10.1111/j.1365-246X.1968.tb00216.x, https://doi.org/10.1111/j.1365-246X.1968.tb00216.x, https://arxiv.org/abs/http://oup.prod.sis.lan/gji/article-pdf/16/2/169/5891044/16-2-169.pdf.

[6] E. BAYSAL, D. D. KOSLOFF, AND J. W. C. SHERWOOD, *Reverse time migration*, GEOPHYSICS, 48 (1983), pp. 1514–1524, https://doi.org/10.1190/1.1441434, https://doi.org/10.1190/1.1441434, https://arxiv.org/abs/https://doi.org/10.1190/1.1441434.

[7] Y. BENGIO, P. SIMARD, AND P. FRASCONI, *Learning long-term dependencies with gradient descent is difficult*, IEEE Transactions on Neural Networks, 5 (1994), pp. 157–166, https://doi.org/10.1109/72.279181, https://doi.org/10.1109/72.279181.

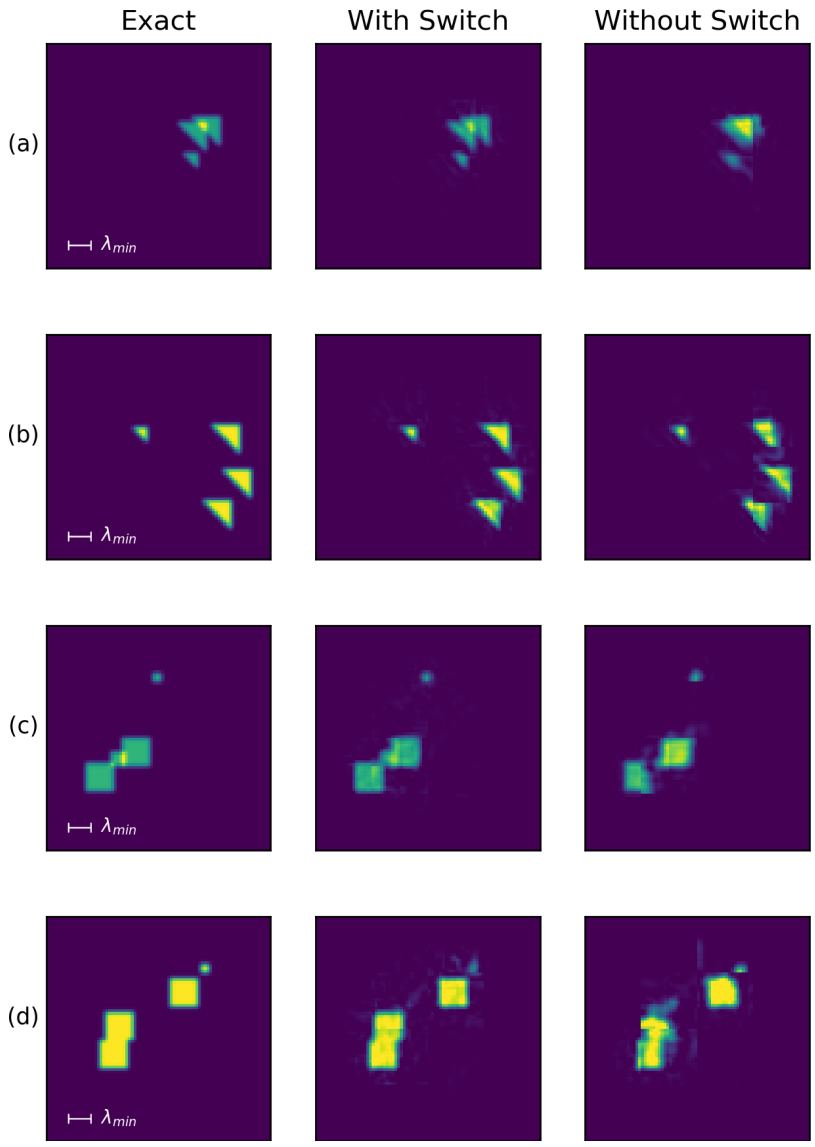

FIG. 13. *Visualization the effect of removing the switch permutation layer. The colour scale of each row is normalized to the first column. We observe that while* `WideBNet` *-without-switch manages to localize the scatterers, it is unable to fully resolve all subwavelength features.*

[8] J.-P. BÉRENGER, *A perfectly matched layer for the absorption of electromagnetic waves*, J. Comput. Phys., 114 (1994), pp. 185–200.

[9] D. BLALOCK, J. J. G. ORTIZ, J. FRANKLE, AND J. GUTTAG, *What is the state of neural network pruning?*, 2020, https://arxiv.org/abs/2003.03033.

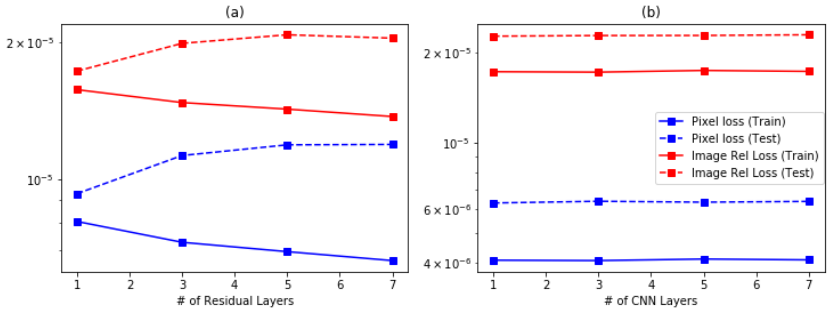

FIG. 14. *Sensitivity to hyper-parameters: number of residual layers and number of convolution post-processing layers. The testing set of* 3000 *points was fixed across all experiments. (a) The training error decreases with increasing residual layers, but the testing error increases. Note however the variation is negligible. (These experiments all had three convolution layers) . (b)* `WideBNet` *exhibits nearly complete insensitivity to the number of CNN post processing layers. (this experiment was with three residual layers)*

[10] C. BORGES, A. GILLMAN, AND L. GREENGARD, *High resolution inverse scattering in two dimensions using recursive linearization*, SIAM J. Imaging Sci., 10 (2017), pp. 641–664, https://doi.org/10.1137/16M1093562, https://doi.org/10.1137/16M1093562, https://arxiv.org/abs/https://doi.org/10.1137/16M1093562.

[11] S. BÖRM, C. BÖRST, AND J. M. MELENK, *An analysis of a butterfly algorithm*, Comput. Math. Appl., 74 (2017), pp. 2125 – 2143, https://doi.org/https://doi.org/10.1016/j.camwa.2017.05.019, http://www.sciencedirect.com/science/article/pii/S0898122117303085. Advances in Mathematics of Finite Elements, honoring 90th birthday of Ivo Babuška.

[12] J. BRUNA AND S. MALLAT, *Invariant scattering convolution networks*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 35 (2013), pp. 1872–1886.

[13] M. BURGER AND S. J. OSHER, *A survey on level set methods for inverse problems and optimal design*, Eur. J. Appl. Math., 16 (2005), p. 263–301, https://doi.org/10.1017/S0956792505006182.

[14] W. CAI, X. LI, AND L. LIU, *Phasednn - a parallel phase shift deep neural network for adaptive wideband learning*, 2019, https://arxiv.org/abs/1905.01389.

[15] W. CAI AND Z.-Q. J. XU, *Multi-scale deep neural networks for solving high dimensional PDEs*, ArXiv e-prints, [cs.LG] 1910.11710, https://arxiv.org/abs/arXiv:1910.11710.

[16] E. CANDÈS, L. DEMANET, AND L. YING, *A fast butterfly algorithm for the computation of fourier integral operators*, Multiscale Model. Sim., 7 (2009), pp. 1727–1750, https://doi.org/10.1137/080734339, http://dx.doi.org/10.1137/080734339, https://arxiv.org/abs/http://dx.doi.org/10.1137/080734339.

[17] E. J. CANDÈS AND C. FERNANDEZ-GRANDA, *Super-resolution from noisy data*, Journal of Fourier Analysis and Applications, 19 (2013), pp. 1229–1254, https://doi.org/10.1007/s00041-013-9292-3, https://doi.org/10.1007/s00041-013-9292-3.

[18] E. J. CANDÈS AND C. FERNANDEZ-GRANDA, *Towards a mathematical theory of super-resolution*, Comm. Pure and Appl. Math., 67 (2014), pp. 906–956, https://doi.org/10.1002/cpa.21455, http://dx.doi.org/10.1002/cpa.21455.

[19] Y. CHEN, L. LU, G. E. KARNIADAKIS, AND L. D. NEGRO, *Physics-informed neural networks for inverse problems in nano-optics and metamaterials*, Opt. Express, 28 (2020), pp. 11618–11633, https://doi.org/10.1364/OE.384875, http://www.opticsexpress.org/abstract.cfm?URI=oe-28-8-11618.

[20] M. CHENEY, *A mathematical tutorial on synthetic aperture radar*, SIAM Rev., 43 (2001), pp. 301–312, https://doi.org/10.1137/S0036144500368859, https://doi.org/10.1137/S0036144500368859, https://arxiv.org/abs/https://doi.org/10.1137/S0036144500368859.

[21] B. A. CIPRA, *The best of the 20th century: Editors name top 10 algorithms*, SIAM News, 33 (2000), pp. 1–2.

[22] N. COHEN, O. SHARIR, AND A. SHASHUA, *On the expressive power of deep learning: A tensor analysis*, in Conference on Learning Theory, 2016, pp. 698–728.

[23] A. COLLI, D. PRATI, M. FRAQUELLI, S. SEGATO, P. P. VESCOVI, F. COLOMBO, C. BALDUINI,

S. Della Valle, and G. Casazza, *The use of a pocket-sized ultrasound device improves physical examination: Results of an in- and outpatient cohort study*, PLOS ONE, 10 (2015), pp. 1–10, https://doi.org/10.1371/journal.pone.0122181, https://doi.org/10.1371/journal.pone.0122181.

[24] D. Colton and A. Kirsch, *A simple method for solving inverse scattering problems in the resonance region*, Inverse Problems, 12 (1996), pp. 383–393, https://doi.org/10.1088/0266-5611/12/4/003, https://doi.org/10.1088%2F0266-5611%2F12%2F4%2F003.

[25] D. Colton and R. Kress, *Integral Equation Methods in Scattering Theory*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2013, https://doi.org/10.1137/1.9781611973167, http://epubs.siam.org/doi/abs/10.1137/1.9781611973167, https://arxiv.org/abs/http://epubs.siam.org/doi/pdf/10.1137/1.9781611973167.

[26] D. Colton and R. Kress, *Inverse Acoustic and Electromagnetic Scattering Theory*, Springer-Verlag New York, New York, PA, 3 ed., 2013, https://doi.org/10.1007/978-1-4614-4942-3.

[27] J. W. Cooley and J. W. Tukey, *An algorithm for the machine calculation of complex Fourier series*, Math. Comput., 19 (1965), pp. 297–301, http://www.jstor.org/stable/2003354.

[28] T. Dao, A. Gu, M. Eichhorn, A. Rudra, and C. Ré, *Learning fast algorithms for linear transforms using butterfly factorizations*, Proceedings of machine learning research, 97 (2019), pp. 1517–1527.

[29] T. Dao, A. Gu, M. Eichhorn, A. Rudra, and C. Ré, *Learning fast algorithms for linear transforms using butterfly factorizations*, 2019, https://arxiv.org/abs/1903.05895.

[30] M. de Buhan and M. Darbas, *Numerical resolution of an electromagnetic inverse medium problem at fixed frequency*, Comput. Math. Appl., 74 (2017), pp. 3111 – 3128, https://doi.org/10.1016/j.camwa.2017.08.002.

[31] M. de Buhan and M. Kray, *A new approach to solve the inverse scattering problem for waves: combining the TRAC and the adaptive inversion methods*, Inverse Probl., 29 (2013), p. 085009, https://doi.org/10.1088/0266-5611/29/8/085009.

[32] L. Demanet and L. Ying, *Fast wave computation via Fourier integral operators*, Math. Comput., 81 (2012), pp. 1455–1486.

[33] C. Doersch, *Tutorial on variational autoencoders*, arXiv:1606.05908.

[34] D. L. Donoho, *Superresolution via sparsity constraints*, SIAM Journal on Mathematical Analysis, 23 (1992), pp. 1309–1331, https://doi.org/10.1137/0523074, https://doi.org/10.1137/0523074, https://arxiv.org/abs/https://doi.org/10.1137/0523074.

[35] B. Engquist and L. Ying, *Sweeping preconditioner for the Helmholtz equation: moving perfectly matched layers*, Multiscale Model. Sim., 9 (2011), pp. 686–710.

[36] Y. Fan, J. Feliu-Fabà, L. Lin, L. Ying, and L. Zepeda-Núñez, *A multiscale neural network based on hierarchical nested bases*, Research in the Mathematical Sciences, 6 (2019), p. 21, https://doi.org/10.1007/s40687-019-0183-3.

[37] Y. Fan, L. Lin, L. Ying, and L. Zepeda-Núñez, *A multiscale neural network based on hierarchical matrices*, arXiv:1807.01883.

[38] Y. Fan and L. Ying, *Solving inverse wave scattering with deep learning*, arXiv:1911.13202.

[39] Y. Fan and L. Ying, *Solving optical tomography with deep learning*, arXiv:1910.04756.

[40] Y. Fan and L. Ying, *Solving traveltime tomography with deep learning*, arXiv:1911.11636.

[41] J. Garnier and G. Papanicolaou, *Passive Imaging with Ambient Noise*, Cambridge Monographs on Applied and Computational Mathematic, Cambridge University Press, 2016, https://books.google.com/books?id=9kfGCwAAQBAJ.

[42] D. Gilton, G. Ongie, and R. Willett, *Neumann networks for inverse problems in imaging*, ArXiv e-prints, [cs.CV] 1901.03707, https://arxiv.org/abs/arXiv:1901.03707.

[43] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*, MIT Press, 2016.

[44] 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 27, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, eds., Curran Associates, Inc., 2014, pp. 2672–2680, http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

[45] B. Gutenberg, *Ueber erdbebenwellen. vii a. beobachtungen an registrierungen von fernbeben in göttingen und folgerung über die konstitution des erdkörpers (mit tafel)*, Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse, 1914 (1914), pp. 125–176, http://eudml.org/doc/58907.

[46] K. He and J. Sun, *Convolutional neural networks at constrained time cost*, 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), (2015), pp. 5353–5360.

[47] R. J. Hewett and L. Demanet, *Pysit*, 2013, http://pysit.org/.

[48] L. Hörmander, *The Analysis of Linear Partial Differential Operators. IV: Fourier Integral Operators*, vol. 63 of Classics in Mathematics, Springer, Berlin, 2009.

[49] K. Hornik, M. Stinchcombe, and H. White, *Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks*, Neural networks, 3 (1990), pp. 551–560.
[50] P. Hähner and T. Hohage, *New stability estimates for the inverse acoustic inhomogeneous medium problem and applications*, SIAM Journal on Mathematical Analysis, 33 (2001), pp. 670–685, https://doi.org/10.1137/S0036141001383564.
[51] M. Innes, A. Edelman, K. Fischer, C. Rackauckas, E. Saba, V. B. Shah, and W. Tebbutt, *A differentiable programming system to bridge machine learning and scientific computing*, 2019, https://arxiv.org/abs/1907.07587.
[52] P. Isola, J. Zhu, T. Zhou, and A. A. Efros, *Image-to-image translation with conditional adversarial networks*, (2017), pp. 5967–5976, https://doi.org/10.1109/CVPR.2017.632.
[53] E. Kang, W. Chang, J. Yoo, and J. C. Ye, *Deep convolutional framelet denosing for low-dose ct via wavelet residual network*, IEEE Transactions on Medical Imaging, 37 (2018), pp. 1358–1369.
[54] Y. Khoo and L. Ying, *SwitchNet: A neural network model for forward and inverse scattering problems*, SIAM J. Sci. Comput., 41 (2019), pp. A3182–A3201, https://doi.org/10.1137/18M1222399, https://doi.org/10.1137/18M1222399, https://arxiv.org/abs/https://doi.org/10.1137/18M1222399.
[55] D. Kingma and J. Ba, *Adam: A method for stochastic optimization*, in The 3rd International Conference for Learning Representations (ICLR), 2015. arXiv:1412.6980v8.
[56] D. P. Kingma and M. Welling, *Auto-encoding variational bayes*, arXiv:1312.6114.
[57] A. Kirsch and N. Grinberg, *The Factorization Method for Inverse Problems*, Oxford University Press, Oxford, first ed., 2008.
[58] Y. LeCun, Y. Bengio, and G. Hinton, *Deep learning*, Nature, 521 (2015), p. 436.
[59] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, *Photo-realistic single image super-resolution using a generative adversarial network*, in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), July 2017, pp. 105–114, https://doi.org/10.1109/CVPR.2017.19.
[60] Y. Li, X. Cheng, and J. Lu, *Butterfly-Net: Optimal function representation based on convolutional neural networks*, arXiv preprint arXiv:1805.07451, (2018).
[61] Y. Li, H. Yang, E. Martin, K. Ho, and L. Ying, *Butterfly factorization*, Multiscale Model. Sim., 13 (2015), pp. 714–732, https://doi.org/10.1137/15M1007173, https://doi.org/10.1137/15M1007173, https://arxiv.org/abs/https://doi.org/10.1137/15M1007173.
[62] Y. Li, H. Yang, and L. Ying, *Multidimensional butterfly factorization*, Applied and Computational Harmonic Analysis, 44 (2018), pp. 737 – 758, https://doi.org/https://doi.org/10.1016/j.acha.2017.04.002, http://www.sciencedirect.com/science/article/pii/S1063520317300271.
[63] Y. E. Li and L. Demanet, *Full-waveform inversion with extrapolated low-frequency data*, Geophysics, 81 (2016), pp. R339–R348, https://doi.org/10.1190/geo2016-0038.1, https://doi.org/10.1190/geo2016-0038.1, https://arxiv.org/abs/https://doi.org/10.1190/geo2016-0038.1.
[64] Y. Liu, X. Xing, H. Guo, E. Michielssen, and X. S. Ghysels, P. Li, *Butterfly factorization via randomized matrix-vector multiplications*, arXiv:2002.03400.
[65] T. Luo, Z. Ma, Z.-Q. J. Xu, and Y. Zhang, *Theory of the frequency principle for general deep neural networks*, ArXiv e-prints, [cs.LG] 1906.09235, https://arxiv.org/abs/arXiv:1906.09235.
[66] X. Mao, C. Shen, and Y.-B. Yang, *Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections*, in Advances in Neural Information Processing Systems 29, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, eds., Curran Associates, Inc., 2016, pp. 2802–2810.
[67] H. Mhaskar, Q. Liao, and T. Poggio, *Learning functions: When is deep better than shallow*, arXiv preprint arXiv:1603.00988, (2016).
[68] M. Mirza and S. Osindero, *Conditional generative adversarial nets*, 2014, http://arxiv.org/abs/1411.1784. cite arxiv:1411.1784.
[69] R. D. Oldham, *The constitution of the interior of the earth, as revealed by earthquakes*, Quarterly Journal of the Geological Society, 62 (1906), pp. 456–475, https://doi.org/10.1144/GSL.JGS.1906.062.01-04.21, https://jgs.lyellcollection.org/content/62/1-4/456, https://arxiv.org/abs/https://jgs.lyellcollection.org/content/62/1-4/456.full.pdf.
[70] M. O'Neil, F. Woolfe, and V. Rokhlin, *An algorithm for the rapid evaluation of special function transforms*, Appl. Comput. Harmon. A., 28 (2010), pp. 203 – 226, https://doi.org/https://doi.org/10.1016/j.acha.2009.08.005, http://www.

sciencedirect.com/science/article/pii/S1063520309000888. Special Issue on Continuous Wavelet Transform in Memory of Jean Morlet, Part I.

[71] J. R. Pettit, A. E. Walker, and M. J. S. Lowe, *Improved detection of rough defects for ultrasonic nondestructive evaluation inspections based on finite element modeling of elastic wave scattering*, IEEE T. Ultrason. Ferr., 62 (2015), pp. 1797–1808, https://doi.org/10.1109/TUFFC.2015.007140, http://dx.doi.org/10.1109/TUFFC.2015.007140.

[72] J. Poulson, L. Demanet, N. Maxwell, and L. Ying, *A parallel butterfly algorithm*, SIAM J. Sci. Comput., 36 (2014), pp. C49–C65, https://doi.org/10.1137/130921544, http://dx.doi.org/10.1137/130921544, https://arxiv.org/abs/http://dx.doi.org/10.1137/130921544.

[73] R. G. Pratt, *Seismic waveform inversion in the frequency domain; part 1: Theory and verification in a physical scale model*, Geophysics, 64 (1999), pp. 888–901, https://doi.org/10.1190/1.1444597, http://Geophysics.geoscienceworld.org/content/64/3/888.abstract, https://arxiv.org/abs/http://Geophysics.geoscienceworld.org/content/64/3/888.full.pdf+html.

[74] M. Raissi and G. E. Karniadakis, *Hidden physics models: Machine learning of nonlinear partial differential equations*, J. Comput. Phys., 357 (2018), pp. 125 – 141, https://doi.org/10.1016/j.jcp.2017.11.039.

[75] N. Rawlinson, S. Pozgay, and S. Fishwick, Phys. Earth Planet. Int., 178 (2010), pp. 101–135, https://doi.org/10.1016/j.pepi.2009.10.002.

[76] O. Ronneberger, P. Fischer, and T. Brox, *U-Net: Convolutional Networks for Biomedical Image Segmentation*, Springer International Publishing, Cham, 2015, pp. 234–241, https://doi.org/10.1007/978-3-319-24574-4_28, https://doi.org/10.1007/978-3-319-24574-4_28.

[77] H. Schomberg, *An improved approach to reconstructive ultrasound tomography*, J. of Phys. D: Appl. Phys., 11 (1978), pp. L181–L185, https://doi.org/10.1088/0022-3727/11/15/004, https://doi.org/10.1088%2F0022-3727%2F11%2F15%2F004.

[78] P. Stefanov, G. Uhlmann, A. Vasy, and H. Zhou, *Travel time tomography*, Acta Math. Sin., 35 (2019), pp. 1085–1114, https://doi.org/10.1007/s10114-019-8338-0, https://doi.org/10.1007/s10114-019-8338-0.

[79] W. W. Symes and J. J. Carazzone, *Velocity inversion by differential semblance optimization*, Geophysics, 56 (1991), pp. 654–663, https://doi.org/10.1190/1.1443082, https://doi.org/10.1190/1.1443082, https://arxiv.org/abs/https://doi.org/10.1190/1.1443082.

[80] A. Tarantola, *Inversion of seismic reflection data in the acoustic approximation*, Geophysics, 49 (1984), pp. 1259–1266, https://doi.org/10.1190/1.1441754, http://dx.doi.org/10.1190/1.1441754, https://arxiv.org/abs/http://dx.doi.org/10.1190/1.1441754.

[81] T. van Leeuwen and F. J. Herrmann, *Mitigating local minima in full-waveform inversion by expanding the search space*, Geophys. J. Int., 195 (2013), pp. 661–667, https://doi.org/10.1093/gji/ggt258, https://doi.org/10.1093/gji/ggt258, https://arxiv.org/abs/http://oup.prod.sis.lan/gji/article-pdf/195/1/661/1674252/ggt258.pdf.

[82] J. Virieux, A. Asnaashari, R. Brossier, L. Métivier, A. Ribodetti, and W. Zhou, *6. An introduction to full waveform inversion*, 2017, pp. R1–1–R1–40, https://doi.org/10.1190/1.9781560803027.entry6, https://library.seg.org/doi/abs/10.1190/1.9781560803027.entry6, https://arxiv.org/abs/https://library.seg.org/doi/pdf/10.1190/1.9781560803027.entry6.

[83] J. Virieux and S. Operto, *An overview of full-waveform inversion in exploration geophysics*, Geophysics, 74 (2009), pp. WCC1–WCC26, https://doi.org/10.1190/1.3238367, http://dx.doi.org/10.1190/1.3238367, https://arxiv.org/abs/http://dx.doi.org/10.1190/1.3238367.

[84] Z. Xu, Y. Li, and X. Cheng, *Butterfly-Net2: Simplified Butterfly-Net and Fourier transform initialization*, vol. 107 of Proceedings of Machine Learning Research, Princeton University, Princeton, NJ, USA, 20–24 Jul 2020, PMLR, pp. 431–450, http://proceedings.mlr.press/v107/xu20b.html.

[85] Z.-Q. J. Xu, *Frequency principle in deep learning with general loss functions and its potential application*, ArXiv e-prints, [cs.LG] 1811.10146, https://arxiv.org/abs/arXiv:1811.10146.

[86] Z.-Q. J. Xu, Y. Zhang, and Y. Xiao, *Training behavior of deep neural network in frequency domain*, in Neural Information Processing, T. Gedeon, K. W. Wong, and M. Lee, eds., Cham, 2019, Springer International Publishing, pp. 264–274.

[87] S. Yao, S. Hu, Y. Zhao, A. Zhang, and T. Abdelzaher, *Deepsense: A unified deep learning framework for time-series mobile sensing data processing*, in Proceedings of the 26th International Conference on World Wide Web, WWW '17, Republic and Canton of Geneva, CHE, 2017, International World Wide Web Conferences Steering Committee, p. 351–360, https://doi.org/10.1145/3038912.3052577, https://doi.org/10.1145/3038912.3052577.

[88] S. Yao, A. Piao, W. Jiang, Y. Zhao, H. Shao, S. Liu, D. Liu, J. Li, T. Wang, S. Hu, L. Su, J. Han, and T. Abdelzaher, *Stfnets: Learning sensing signals from the time-frequency perspective with short-time fourier neural networks*, in The World Wide Web Conference, WWW '19, New York, NY, USA, 2019, Association for Computing Ma-

chinery, p. 2192–2202, https://doi.org/10.1145/3308558.3313426, https://doi.org/10.1145/3308558.3313426.

[89] J. C. Ye, Y. Han, and E. Cha, *Deep convolutional framelets: A general deep learning framework for inverse problems*, SIAM Journal on Imaging Sciences, 11 (2018), pp. 991–1048, https://doi.org/10.1137/17M1141771, https://doi.org/10.1137/17M1141771, https://arxiv.org/abs/https://doi.org/10.1137/17M1141771.

[90] L. Zepeda-Núñez, Y. Chen, J. Zhang, W. Jia, L. Zhang, and L. Lin, *Deep Density: circumventing the Kohn-sham equations via symmetry preserving neural networks*. https://www.math.wisc.edu/~lzepeda/Deep-Density.pdf, 2019.

[91] L. Zepeda-Núñez and L. Demanet, *The method of polarized traces for the 2D Helmholtz equation*, J. Comput. Phys., 308 (2016), pp. 347 – 388, https://doi.org/http://dx.doi.org/10.1016/j.jcp.2015.11.040, http://www.sciencedirect.com/science/article/pii/S0021999115007809.

[92] J. Zhang, L. Zepeda-Núñez, Y. Yao, and L. Lin, *Learning the mapping* $\mathbf{x} \mapsto \sum_{i=1}^{d} x_i^2$*: the cost of finding the needle in a haystack*, Comm. App. Math. Comp., (2020).

[93] L. Zhang, J. Han, H. Wang, R. Car, and W. E, *Deep potential molecular dynamics: A scalable model with the accuracy of quantum mechanics*, Physical Review Letters, 120 (2018), p. 143001.

[94] L. Zhang, J. Han, H. Wang, R. Car, and W. E, *Deepcg: Constructing coarse-grained models via deep neural networks*, J. Chem. Phys., 149 (2018), p. 034101, https://doi.org/10.1063/1.5027645.

Appendix A. Permutation and Switch Indices.

```
def π^l():
  # indices inside each 4^{L-l} × 4^{L-l} block
  Δ = 4^{L-l-1}

  # [0,Δ,2Δ,3Δ,0,Δ,2Δ,3Δ,4Δ,...]
  π^l = np.file(np.arange(4)*Δ, Δ)

  # + [0,0,0,0,1,1,1,1,...,Δ, Δ,Δ,Δ]
  π^l += np.repeat(np.arange(Δ), 4)

  # indices for entire block diagonal matrix
  π^l = np.tile(π^l, 4^l)
  π^l += np.repeat(np.arange(4^l)*4^{L-l}, 4^{L-l})

  return π^l
```

LISTING 6

LCN[a,b,c] == LocallyConnected1D(filters=a, kernel_size=b, strides=c)

```
def switchidx():
  # input
  # output

  π_switch = np.arange(2^L)*(2^L)

  π_switch = np.tile(π_switch), 2^L)
  π_switch += np.repeat(np.arange(2^L), 2^L)

  return π_switch
```

LISTING 7

LCN[a,b,c] == LocallyConnected1D(filters=a, kernel_size=b, strides=c)