Journal of Computational Physics 527 (2025) 113809

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

Multi-frequency progressive refinement for learned inverse scattering

Owen Melia [a,*], Olivia Tsang [a], Vasileios Charisopoulos [b], Yuehaw Khoo [c], Jeremy Hoskins [c], Rebecca Willett [a,b,c]

[a] Department of Computer Science, University of Chicago, Chicago, 60637, IL, USA
[b] Data Science Institute, University of Chicago, Chicago, 60637, IL, USA
[c] Computational and Applied Mathematics, Department of Statistics, University of Chicago, Chicago, 60637, IL, USA

ARTICLE INFO

Dataset link: https://github.com/meliao/mfisnets, https://doi.org/10.5281/zenodo.14514353

Keywords:
Machine learning
Inverse scattering
Algorithm unrolling
Deep learning
Inverse problems

ABSTRACT

Interpreting scattered acoustic and electromagnetic wave patterns is a computational task that enables remote imaging in a number of important applications, including medical imaging, geophysical exploration, sonar and radar detection, and nondestructive testing of materials. However, accurately and stably recovering an inhomogeneous medium from far-field scattered wave measurements is a computationally difficult problem, due to the nonlinear and non-local nature of the forward scattering process. We design a neural network, called Multi-Frequency Inverse Scattering Network (MFISNet), and a training method to approximate the inverse map from far-field scattered wave measurements at multiple frequencies. We consider three variants of MFISNet, with the strongest performing variant inspired by the recursive linearization method — a commonly used technique for stably inverting scattered wavefield data — that progressively refines the estimate with higher frequency content. MFISNet outperforms past methods in regimes with high-contrast, heterogeneous large objects, and inhomogeneous unknown backgrounds.

1. Introduction

Wave scattering is an important imaging technology with applications in medical and seismic imaging, sonar and radar detection, and nondestructive testing of materials. In this setting, a known source transmits incident waves through a penetrable medium, and due to an inhomogeneity in the spatial region of interest, the incident waves are scattered. Several receivers measure the scattered wave field at distant locations. We are interested in the inverse wave scattering problem: given a set of scattered wave field measurements, we want to recover the inhomogeneity in the spatial region of interest that produced the measurements. In this paper, we focus on the inverse wave scattering problem with unknown medium and full-aperture measurements at multiple incident wave frequencies. This problem is characterized by a highly nonlinear forward measurement operator, making the recovery of the scattering potential challenging. We propose a machine learning solution to this problem: given a training set of pairs of scattering potentials and scattered wavefield measurements, we seek to approximate the inversion map with a deep neural network that predicts a scattering potential from scattered wavefield measurements at multiple frequencies. We design a new training method and a new neural network architecture to achieve this goal.

* Corresponding author.
E-mail addresses: meliao@uchicago.edu (O. Melia), oortsang@uchicago.edu (O. Tsang), vchariso@uchicago.edu (V. Charisopoulos), ykhoo@uchicago.edu (Y. Khoo), jeremyhoskins@uchicago.edu (J. Hoskins), willett@g.uchicago.edu (R. Willett).

https://doi.org/10.1016/j.jcp.2025.113809
Received 10 June 2024; Received in revised form 14 January 2025; Accepted 30 January 2025

Available online 3 February 2025

The aforementioned inverse wave scattering problem has been widely studied. While the measurement operator is known to be injective when there are infinitely many sensors positioned in a ring around the scattering potential [8], computational approaches must always operate in the ill-posed case where finite receivers are present. Thus, past research has focused on optimization approaches to solving the inverse problem. Simple gradient-based optimization approaches to this problem face two major difficulties: computing a gradient requires solving a partial differential equation (PDE), which can be computationally expensive; additionally, the nonlinearity of the forward model induces a non-convex objective function. Therefore, convergence of local search methods such as gradient descent is not guaranteed without careful initialization.

A classical strategy to alleviate the optimization challenges associated with a nonlinear forward model is to use a linear approximation. This linear approximation allows one to formulate the inverse problem as a linear least squares problem, which is relatively easy to solve. A well-known method inverting this global linear approximation is the so-called *filtered back-projection (FBP) method* [22]. While this method is relatively easy to implement, it suffers from modeling errors and produces inaccurate reconstructions. To remedy this, many machine learning approaches are aimed at constructing data-driven approximations of the inverse map by designing architectures to imitate FBP [11,15,18]. At a high level, these works approximate the two key components of FBP – namely, an application of the adjoint of a linearization of the forward operator followed by a filtering step – by suitably chosen neural network blocks. As a result, they suffer from similar drawbacks as the FBP method: in particular, they provide low-quality reconstructions of high-contrast scattering potentials, especially in the presence of unknown inhomogeneous backgrounds or measurement noise. A natural alternative is integrating machine-learning models into other iterative methods, which are computationally demanding relative to FBP but can provide higher quality reconstructions.

A standard approach, which has been successful in the strongly nonlinear scattering regime, is to use data collected at multiple incoming wave frequencies. Recursive linearization methods [1,3,6] use multi-frequency measurements to solve a sequence of subproblems, starting at the lowest frequency to provide an initial estimate of the scattering potential and refining that estimate at progressively higher frequencies using warm-started local search methods. Algorithms in this family offer two benefits: first, they alleviate the need for careful initialization since the loss landscape of the lowest frequency sub-problem is typically well-behaved; second, they greatly reduce the number of PDE solves by relying on first-order approximations of the forward model that are relatively inexpensive to invert. However, these methods require measurements at a large number of incident wave frequencies and still involve solving large-scale PDEs and least-squares problems for each frequency. This requires, for example, multiple CPU core-hours to recover a single image, even with a state-of-the-art PDE solver [3].

In light of these advances, we propose a new architecture and training method inspired by the recursive linearization algorithm. Our primary approach is based on a residual update architecture and training method that ensures specific network blocks solve specific sub-problems. In addition, we introduce two new methods of extending FBP-inspired neural networks to the multi-frequency setting. We call our networks "Multi-Frequency Inverse Scattering Networks", or "MFISNets". We note that our methods are based on a neural network architecture from [11] but could, in principle, use any other neural network architectures designed for the inverse scattering problem in the single-frequency setting.

1.1. Contributions & paper outline

In Section 2, we formally define the inverse scattering problem and the machine learning objective. We present standard results about inverse scattering and survey related work in Section 3. In Section 4, we review the recursive linearization algorithm and introduce our method, MFISNet-Refinement. Finally, we introduce our other two methods, MFISNet-Fused and MFISNet-Parallel, and present a numerical evaluation of our methods in Section 5. Our main contributions can be summarized as follows:

1. We introduce "MFISNet-Refinement", short for "Multi-Frequency Inverse Scattering Network with Refinement", a neural network architecture and training method that is inspired by recursive linearization algorithms [1,3,6]. (Section 4)
2. We show that our network achieves lower errors than single-frequency methods [11] and multi-frequency methods [18] in the literature as well as the other newly-introduced MFISNets in a high-contrast, noiseless, full-aperture setting. (Section 5.3)
3. We demonstrate numerically that our method is robust to moderate measurement noise. (Section 5.4)
4. We consider alternative training strategies and find that MFISNet-Refinement is robust to the choice of training method, suggesting the majority of improvement is due to the architecture. (Section 5.5)
5. We publicly release our code https://github.com/meliao/mfisnets and dataset https://doi.org/10.5281/zenodo.14514353.

2. Problem setup and notation

The forward model for our imaging setup is implicitly defined by a PDE problem involving the Helmholtz equation. See Fig. 1 for a diagram of the geometry of the problem. Let $x \in \mathbb{R}^2$ be the spatial variable. Suppose $u_{\text{in}}(x; s) = e^{ikx \cdot s}$ is an incoming plane wave with direction $s \in \mathbb{S}^1$, wavelength λ, and angular wavenumber $k = 2\pi/\lambda$. We normalize the problem's units so this wave travels at speed $c_0 \equiv 1$ in free space. The incoming wave interacts with a real-valued scattering potential $q(x)$ to produce an additive perturbation, called the scattered wave field $u_{\text{scat}}[q](x; s)$. We define $q(x) = c_0^2/c^2(x) - 1$ where $c(x)$ is the wave speed at x. The total wave field $u[q](x; s) = u_{\text{scat}}[q](x; s) + u_{\text{in}}(x; s)$ solves the following inhomogeneous Helmholtz equation, and the scattered wave field satisfies the Sommerfeld radiation boundary condition:

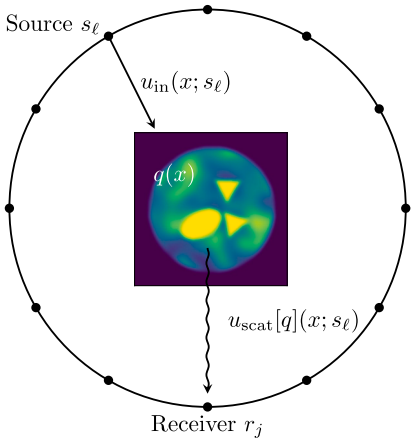


Fig. 1. Geometry of the inverse scattering problem. An incident plane wave $u_{\text{in}}(x; s_\ell)$ coming from source direction s_ℓ interacts with the scattering potential $q(x)$. The resulting scattered wave field $u_{\text{scat}}[q](x; s_\ell)$ is recorded by a receiver r_j.

$$\begin{cases} \Delta u[q](x;s) + k^2(1+q(x))u[q](x;s) = 0 & x \in \mathbb{R}^2; \\ \dfrac{\partial u_{\text{scat}}[q](x;s)}{\partial \|x\|_2} - iku_{\text{scat}}[q](x;s) = o(\|x\|_2^{-1/2}), & \text{as } \|x\|_2 \to \infty. \end{cases} \tag{1}$$

We assume $q(x)$ is supported on a square domain $\Omega = [-0.5, 0.5]^2$, and we work with $q \in \mathbb{R}^{N_q \times N_q}$, the discretization of $q(x)$ onto a regular grid with (N_q, N_q) grid points. We place the receivers equally spaced around a large ring of radius $R \gg 1$, centered at the origin. We identify individual receivers by their unit-vector directions $r_j \in \mathbb{S}^1$. We compute the solution for the same set of N_r receiver points and N_s incoming source directions, where $N_r = N_s$ and the grid points are equally distributed about the unit circle. This results in a set of (N_r, N_s) observations, $\{u_{\text{scat}}[q](Rr_j; s_\ell)\}_{j,\ell \in [N_r] \times [N_s]}$, which we arrange in a data array $d_k \in \mathbb{C}^{N_r \times N_s}$. We call the mapping from q to d_k the forward model with incoming wave frequency k:

$$\left(d_k\right)_{j,\ell} = \mathcal{F}_k[q]_{j,\ell} \equiv u_{\text{scat}}[q](Rr_j; s_\ell) \tag{2}$$

Because we are interested in multi-frequency algorithms, we are interested in observations of the forward model evaluated on the same q but with a set of incoming wave frequencies $[k_1, \ldots, k_{N_k}]$. In particular, our goal is to approximate the following mapping:

$$\left[\mathcal{F}_{k_1}[q], \ldots, \mathcal{F}_{k_{N_k}}[q]\right] \mapsto q. \tag{3}$$

Our goal is to train a neural network g_θ with parameters θ to approximate the mapping: $g_\theta(\mathcal{F}_{k_1}[q], \ldots, \mathcal{F}_{k_{N_k}}[q]) \approx q$. Given a distribution $\mathcal{D}$ over scattering potentials q, we draw a training set of n independent samples from $\mathcal{D}$ to generate data to train the neural network. After evaluating the forward model nN_k times, we have a training set

$$\mathcal{D}_n := \left\{ (q^{(j)}, \mathcal{F}_{k_1}[q^{(j)}], \ldots, \mathcal{F}_{k_{N_k}}[q^{(j)}]) \right\}_{j=1}^{n} \tag{4}$$

We evaluate networks in this setting by measuring the relative ℓ_2 error:

$$\text{RelativeL2Error}(g_\theta) = \mathbb{E}_{q \sim \mathcal{D}} \left[\frac{\left\| g_\theta\left(\mathcal{F}_{k_1}[q], \ldots, \mathcal{F}_{k_{N_k}}[q]\right) - q \right\|_2}{\|q\|_2} \right] \tag{5}$$

In practice, we approximate the expected relative ℓ_2 error in (5) by an empirical mean over a held-out test set of 1,000 samples drawn independently from $\mathcal{D}$.

3. Background and related work

3.1. Background

In this section, we review standard results about $\mathcal{F}_k$ relevant to our study. In particular, we focus on a linear approximation of $\mathcal{F}_k$ that gives insights into the inverse scattering problem.

The first result is that $\mathcal{F}_k$ becomes more nonlinear as the magnitude of the scatterer $\|q\|_2$ or the wavenumber k increases. Indeed, the solution to (1) can be equivalently defined as the solution to the Lippmann-Schwinger integral equation:

$$u_{\text{scat}}[q](x;s) = k^2 \int_\Omega G_k(\|x-x'\|_2) q(x') (u_{\text{in}}(x';s) + u_{\text{scat}}[q](x';s)) \mathrm{d}x' \tag{6}$$

where G_k is the Green's function for the homogeneous Helmholtz operator. This recursive equation provides a nonlinear map from $q(x)$ to $u_{\text{scat}}[q](\cdot;s)$ and therefore $\mathcal{F}_k[q]$.

One way to view this nonlinearity is to interpret the Lippmann-Schwinger equation as a power series in $q(x)$ by iteratively substituting the value of $u_{\text{scat}}[q](x)$ into its appearance on the right-hand side of (6). For example, performing this substitution once yields a linear and a quadratic term in $q(x)$ that are independent of $u_{\text{scat}}[q]$, as well as a "remainder" term involving the unknown $u_{\text{scat}}[q]$ that accounts for higher-order terms:

$$\begin{aligned} u_{\text{scat}}[q](x;s) = {} & k^2 \int_\Omega G_k(\|x-x'\|_2) q(x') u_{\text{in}}(x';s) \mathrm{d}x' \\ & + k^4 \int_\Omega G_k(\|x-x'\|_2) q(x') \int_\Omega G_k(\|x'-x''\|_2) q(x'') u_{\text{in}}(x'';s) \mathrm{d}x'' \mathrm{d}x' \\ & + \text{Higher-order terms in } k \text{ and } q(x). \end{aligned} \tag{7}$$

This power series does not converge for general $q(x)$, but it helps illustrate which parts of the problem drive the nonlinearity of the operator $\mathcal{F}_k$: as $\|q\|_2$ or k grow, the size of these nonlinear terms will also grow, and as a result $\mathcal{F}_k$ becomes increasingly nonlinear.

The next result is that, under a linear approximation, the far-field measurements are diffraction-limited and can only capture frequency components of q up to $2k$. Equivalently, the measurements depend on q to a spatial resolution of $\lambda/2$. We consider the first-order *Born approximation* [4], which approximates (6) by dropping the $u_{\text{scat}}[q](x';s)$ term from the right-hand side. This is further simplified with an approximation of the Green's function in the far-field limit [4], yielding

$$d_k(r,s) \approx k^2 \int_\Omega e^{-ik(r-s)\cdot x'} q(x') \mathrm{d}x'. \tag{8}$$

We will refer to this linear approximation of the map from $q(x)$ to $d_k(r,s)$ as F_k. Note that $F_k q$ is proportional to the Fourier Transform of q evaluated at frequency vectors of the form $k(r-s)$. Since $r, s \in \mathbb{S}^1$ range over the unit circle, the frequency vectors $k(r-s)$ take on values throughout a disk with radius $2k$ centered at the origin. Thus, evaluations of the linearized forward model, $F_k q$, only contain the low-frequency components of q, while high-frequency components of q are in the kernel of the linearized forward model F_k [6].

Owing to its simplicity, (8) is often used as inspiration for the design of neural network architectures approximating the inverse map $d_k \mapsto q$. The networks emulate the FBP method [22], which produces an estimate $\hat{q}$ of the scattering potential q as

$$\hat{q} = (F_k^* F_k + \mu I)^{-1} F_k^* d_k, \tag{9}$$

where F_k^* is the adjoint of F_k and μ is a regularization parameter that stabilizes the inversion of $F_k^* F_k$. The operator $(F_k^* F_k + \mu I)^{-1}$ can be implemented as a two-dimensional spatial convolution [11,15,18], while novel network architectures have been proposed to emulate $F_k^* \in \mathbb{C}^{N_q^2 \times N_r N_s}$ in a parameter-efficient manner. In particular, [11] and [30] suggest leveraging the rotational equivariance of the forward model to emulate F_k^* with one-dimensional convolutions, after applying a far-field scaling and an appropriate coordinate transformation in [11, Equation 6]). We refer to the network described by Fan and Ying [11] as FYNet.

3.2. Related work

Deep learning has revolutionized linear inverse problems in imaging, advancing methods for superresolution, inpainting, deblurring, and medical imaging. Many of these advances stem from methods combining deep neural networks with optimization algorithms. For example, the *deep unrolling* paradigm [21] performs a fixed number of steps of an iterative algorithm and replaces certain operations with learnable mappings, which are parameterized by neural networks whose weights are learned from data. Components that remain fixed throughout training may reflect prior knowledge of problem parameters, such as explicit knowledge of the forward measurement model. In this setting, the network is usually trained end-to-end by minimizing the Euclidean distance between the network outputs and the true data. Another paradigm, called *plug-and-play denoising* [26], suggests that general image denoisers can be used in place of proximal operators for regularization functions, an important subroutine in many optimization routines for linear inverse problems in imaging. In this setting, neural network blocks are often trained to solve a different task, such as denoising corrupted signals, and then used inside the inversion algorithm. Ongie et al. [24] provides a review of deep learning for inverse problems in imaging.

Several works in the wave scattering literature attempt to solve the inverse scattering problem by augmenting an optimization algorithm with components learned from data. At inference time, these methods require running an iterative optimization algorithm. Kamilov et al. [14] develops a plug-and-play algorithm for inverse scattering, and show that various off-the-shelf denoisers can be applied as proximal operators. Zhao et al. [31] use an encoder network paired with a network emulating the forward model and suggest optimizing a latent representation of the scattering potential using stochastic gradient descent. When only phaseless measurements of the scattered wave field u_{scat} are available, [9] propose a network unrolling proximal gradient descent, where the proximal operator is a neural network learned from data. For the inverse obstacle scattering problem in two dimensions, [32] propose using a fully-connected neural network to warm-start a Gauss-Newton algorithm. Ding et al. [10] train a neural network

to approximately invert a forward scattering process depending on temporal data, and use this approximate inverse as a nonlinear preconditioner for a nonlinear least squares optimization routine. In concurrent work, Zhang et al. [29] use diffusion sampling to reconstruct scattering potentials and quantify uncertainty of the reconstructions.

Other methods propose to learn the inverse map directly from data. Recently, neural networks that are approximately invariant to discretization size have been proposed as methods of learning maps between general function spaces [19,20] and these general-purpose networks have been applied to inverse scattering [23]. Other networks have been designed to invert the forward scattering model in particular; see [5] for a broad review of such approaches. In our work, we are particularly interested in FBP-inspired methods. The work of [15] leverages the approximate low-rank structure of scattering operators to design their SwitchNet network. Fan and Ying [11] propose a data transformation that facilitates emulating the adjoint of the linearized forward model via 1D convolutions. In our work, we consider how to combine multiple such network blocks to invert the multi-frequency forward map in (3). One way of combining these blocks is to learn each adjoint operator F_k^* as a separate neural network block and combine data to jointly emulate the learned filtering operators $(F_k^* F_k + \mu I)^{-1}$. This strategy is employed in [30], which is similar to our MFISNet-Parallel, but uses a different parameterization for the layers emulating F_k^* and $(F_k^* F_k + \mu I)^{-1}$. Another strategy is to use the Wide-Band Butterfly Network [17,18], which hierarchically merges information from different frequencies in the network block emulating F_k^*. We provide more detail and commentary about methods of combining network blocks to form multi-frequency networks in Section 5.2.

Finally, we note in passing that the design of our method is inspired by *homotopy methods* [28]. These methods solve a sequence of sub-problems of increasing difficulty, gradually transforming a simple (but uninformative) optimization problem to the optimization problem of interest and using solutions to a given sub-problem to warm-start local search methods for subsequent sub-problems. Such a sequence can be constructed explicitly (*e.g.*, by varying regularization levels) or implicitly; for example, *curriculum learning* [2] progressively adjusts the training data distribution from "easy" to "hard" samples and has been used to train physics-informed neural networks in challenging problem settings [12,16].

4. Recursive linearization and our method

We propose a neural network that learns to approximate the multi-frequency inversion map from training data. To design the network and training algorithm, we draw inspiration from the recursive linearization method for inverse scattering, which we briefly review below.

4.1. Recursive linearization

Recursive linearization is a classical method for solving the inverse scattering problem, introduced by [6]. In spite of the nonlinearity of the true forward scattering model described in (6), recursive linearization breaks the inverse problem into a series of simpler problems, each of which corresponds to a linear inverse problem. In this section we discuss the intuition behind this strategy.

Recall from our discussion in Section 3.1 that the forward map evaluated at low incident wave frequencies k acts approximately like a low-pass filter with cutoff frequency $2k$. At first glance, this suggests that observing $\mathcal{F}_k[q]$ for a high value of k is sufficient for high-resolution recovery of q. However, when viewed from an optimization perspective, it becomes clear that this problem is increasingly challenging for large values of k. For example, one might consider the nonlinear least-squares problem

$$\operatorname*{argmin}_{\hat{q}} \|d_k - \mathcal{F}_k[\hat{q}]\|_2^2. \tag{10}$$

To illustrate the challenges for the optimization formulation with increasing values of k, we consider a simple example where q is known to be a Gaussian bump with a given spread parameter and unknown amplitude. Given observations $d_k = \mathcal{F}_k[q]$ and a numerical PDE solver to calculate $\mathcal{F}_k[\cdot]$, one could estimate the amplitude of q by solving a minimization problem similar to (10), but only searching over the unknown amplitude. In Fig. 2, we plot this objective as a function of the amplitude of $\hat{q}$ for a range of values of k. For large values of k, the objective function is highly oscillatory and contains many spurious local minima. Typically this suggests that it can be challenging to locate the global minimum unless there is a scheme to initialize estimates close enough to the global minimum to avoid getting stuck elsewhere. However, Fig. 2 also shows that the objective function oscillates much more slowly at the smallest value of k while sharing the same global minimum. This suggests that the low-frequency observations can be used to get close to the global minimum, even though they are diffraction-limited and cannot resolve high-frequency components.

The recursive linearization algorithm leverages this insight by solving a sequence of inversion problems at increasing wave frequencies k. Crucially, each sub-problem uses the output of the previous sub-problem to initialize a new optimization problem. This method was introduced in [6] and further developed in [1,3]. At iteration t, the algorithm uses the previous estimate $\hat{q}_{k_{t-1}}$ along with a new set of observations d_{k_t}, and it calculates an update δq that minimizes the ℓ_2 distance in measurement space:

$$\operatorname*{argmin}_{\delta q} \|d_{k_t} - \mathcal{F}_{k_t}[\hat{q}_{k_{t-1}} + \delta q]\|_2^2 \tag{11}$$

The value of $\hat{q}_{k_{t-1}}$ may make it possible to avoid spurious local minima, but this problem is still difficult since an iterative optimizer would require solving a PDE at each of its iterations. However, $\mathcal{F}_{k_t}[\hat{q}_{k_{t-1}} + \delta q]$ is well-approximated by a first-order Taylor expansion about $\hat{q}_{k_{t-1}}$ when δq is small or when it does not contain low-frequency information [6]. This motivates the following surrogate for the optimization problem in (11):

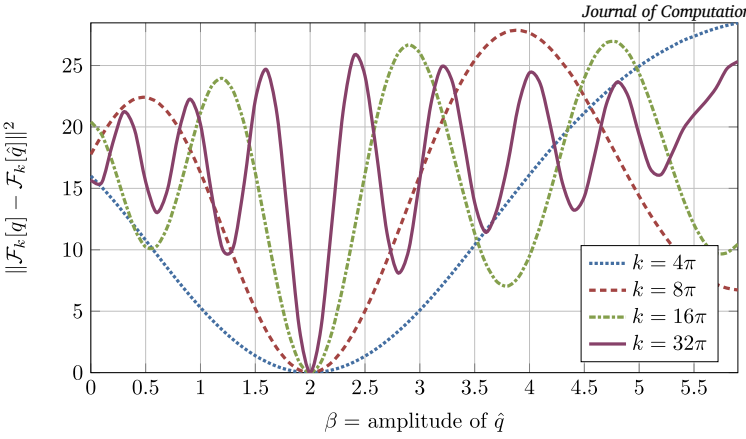


Fig. 2. Even in a highly stylized setting, accurately and reliably inverting $\mathcal{F}_k$ is difficult for high frequencies k. Suppose the ground-truth scattering potential is $q(x) = \beta \exp(-\frac{\|x\|^2}{2\sigma^2})$, a Gaussian bump with known spread parameter $\sigma = 0.1$ but unknown amplitude β. We show the optimization landscape that arises from searching over different amplitudes. At low incident wave frequencies, this optimization landscape is smooth and has a large basin of attraction. However, as the incident wave frequency increases, the optimization landscape becomes highly oscillatory, requiring a nearly exact initialization to guarantee convergence to the ground truth. In experimental settings, the parameterization of q is often high-dimensional, which requires higher frequency data to resolve high-frequency information in q. This numerical example was inspired by [1].

Algorithm 1: Recursive linearization for inverse scattering based on [1,3,6,7].

Input: Multi-frequency data $\{d_{k_1}, d_{k_2}, \dots, d_{k_{N_k}}\}$

1 $\hat{q}_{k_1} \leftarrow \left(F_{k_1}^* F_{k_1} + \mu I\right)^{-1} F_{k_1}^* d_{k_1}$

2 **for** $t = 2, \dots, N_k$ **do**

3 Compute $\mathcal{F}_{k_t}[\hat{q}_{k_{t-1}}]$ and $D\mathcal{F}_{k_t}[\hat{q}_{k_{t-1}}]$

4 $\delta q_{k_t} \leftarrow \underset{\delta q}{\operatorname{argmin}} \ \|d_{k_t} - \left(\mathcal{F}_{k_t}[\hat{q}_{k_{t-1}}] + D\mathcal{F}_{k_t}[\hat{q}_{k_{t-1}}]\delta q\right)\|_2^2$

5 $\hat{q}_{k_t} \leftarrow \hat{q}_{k_{t-1}} + \delta q_{k_t}$

Result: Final estimate $\hat{q}_{k_{N_k}}$

$$\underset{\delta q}{\operatorname{argmin}} \ \|d_{k_t} - \left(\mathcal{F}_{k_t}[\hat{q}_{k_{t-1}}] + D\mathcal{F}_{k_t}[\hat{q}_{k_{t-1}}]\delta q\right)\|_2^2, \tag{12}$$

where $D\mathcal{F}_{k_t}[\hat{q}_{k_{t-1}}]$ denotes the Fréchet derivative of the forward model at $\hat{q}_{k_{t-1}}$. The action of $D\mathcal{F}_{k_t}[\hat{q}_{k_{t-1}}]$ and its adjoint, $D\mathcal{F}_{k_t}^*[\hat{q}_{k_{t-1}}]$, can be computed using the adjoint-state method [1,3]. The resulting algorithm is akin to a Gauss-Newton method; critically, each sub-problem of the form shown in (12) is a linear least-squares problem. We outline a sketch of the recursive linearization algorithm in Algorithm 1.

The recursive linearization algorithm is very demanding computationally. Each iteration requires solving N_s large-scale PDEs and a high-dimensional least-squares problem, which quickly creates a large computational burden when producing high-resolution solutions. In a classical setting without machine learning, the frequencies should be spaced close to each other for best results. Chen [7] uses $k = 1, 2, \dots, 9$ in their numerical experiments, while [3] uses $k = 1, 1.25, \dots, 70$, which they report takes around 40-50 hours per sample to produce a single 241×241 pixel image.

Although recursive linearization is computationally expensive as stated, we believe that one of the key features of recursive linearization is the way that it breaks the recovery process into multiple steps, each of which refines the estimate from the previous step using data of a higher frequency. We will refer to this step-wise recovery strategy as progressive refinement.

Progressive refinement facilitates the recovery process, since each step is only responsible for a correction to the estimate of the scattering potential within a frequency band. Focusing on this strategy also allows us to look for machine learning methods that do not explicitly emulate $\mathcal{F}_k[\cdot]$ or $D\mathcal{F}_k[\cdot]$, which are expensive to compute.

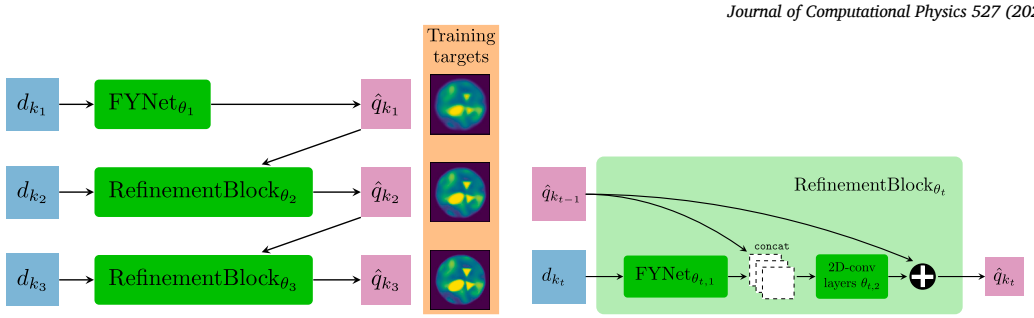
To this end, we consider a generalization of recursive linearization where we replace lines 3 and 4 in Algorithm 1 by describing the inner loop as a generic refinement step:

$$\delta q_{k_t} = \text{RefinementStep}_{k_t}(\hat{q}_{k_{t-1}}, d_{k_t}) \qquad \text{for } t = 2, \dots, N_k \tag{13}$$

where $\text{RefinementStep}_{k_t}(\hat{q}_{k_{t-1}}, d_{k_t})$ refers to the update calculated for estimate $\hat{q}_{k_t}$ given data d_{k_t} and can be implemented using a neural network. We will propose and discuss a network architecture in the next section.

4.2. Our method

We use Algorithm 1 as inspiration for the design of our neural network architecture and training method. In particular, we focus on the following two crucial aspects of Algorithm 1:



(a) MFISNet-Refinement Architecture (b) Refinement Block

Fig. 3. Our MFISNet-Refinement architecture is designed to emulate the recursive linearization algorithm. Fig. 3a shows that our network proceeds by making an initial low-frequency reconstruction and then making a series of updates given higher-frequency data and an estimate of the scattering potential. The network is trained to match the intermediate reconstructions to low-pass filtered versions of scattering potentials from the training set. One example collection of such filtered scattering potentials is shown. Fig. 3b shows that our refinement block is a simple extension of the FYNet architecture from [11]. By using a skip connection in this block, we ensure the network only needs to predict an update to the estimated scattering potential.

Progressive refinement: The algorithm builds intermediate estimates of the scattering potential which are progressively refined with the introduction of new data.

Homotopy through frequency: The iterative refinements from the first step form a homotopy from low to high-frequency measurements. As a result, updates at step t contain high-frequency information relative to k_{t-1}.

To emulate the progressive refinement structure, we propose a network with a residual structure and skip connections. The network comprises multiple blocks, one for each incident wave frequency k_t, $t = 1, \dots, N_k$. The input to each block is measurement data d_{k_t} collected at a particular incident wave frequency k_t. The input passes through an FYNet block [11], which approximately inverts the forward model. The output of the FYNet block is then concatenated with the output of the previous block, $\hat{q}_{k_{t-1}}$, and the concatenation is passed to 2D convolutional layers for a second filtering step. Finally, a skip connection adds $\hat{q}_{k_{t-1}}$ to the output of the last convolutional layer of the block, producing the next estimate $\hat{q}_{k_t}$. The network's architecture is shown in Fig. 3; we call the resulting network "MFISNet-Refinement." Note that, under this construction, the FYNet blocks could be replaced by any other neural network architecture designed for the single-frequency inverse scattering problem.

To emulate the homotopy through frequency, we design a training method to ensure each successive block adds higher-frequency information to the estimate of the scattering potential. Under the Born approximation (8), we know d_k contains information about q up to frequency limit $2k$. This suggests that given data d_{k_t}, we should be able to reconstruct the frequency components of q up to $2k_t$. To reflect this, we train the output of block t with the following loss function:

$$L_t(\hat{q}_{k_t}; q) = \mathbb{E}_{q\sim D_n}\left[\|\hat{q}_{k_t} - \mathsf{LPF}_{2k_t} q\|^2\right] \tag{14}$$

In (14), LPF_{2k_t} is a low-pass filter with approximate cutoff frequency $2k_t$, implemented as a Gaussian filter to avoid ringing artifacts; its frequency response is given in the Fourier domain by

$$\widetilde{\mathsf{LPF}}_{f_{\mathsf{cut}}}(f) := \exp\left(-\frac{1}{2}\left(\frac{f}{f_{\mathsf{cut}}/\sqrt{2\log 2}}\right)^2\right), \tag{15}$$

given a target cutoff frequency f_{cut}. Note that we shrink the standard deviation by a factor of $\sqrt{2\log 2}$ so that the filter's half-width at half-maximum matches f_{cut}.

To train the network, we first adjust the weights of each block in a sequential fashion and then perform a final training step which fine-tunes all of the blocks jointly; the training procedure is summarized in Algorithm 2.

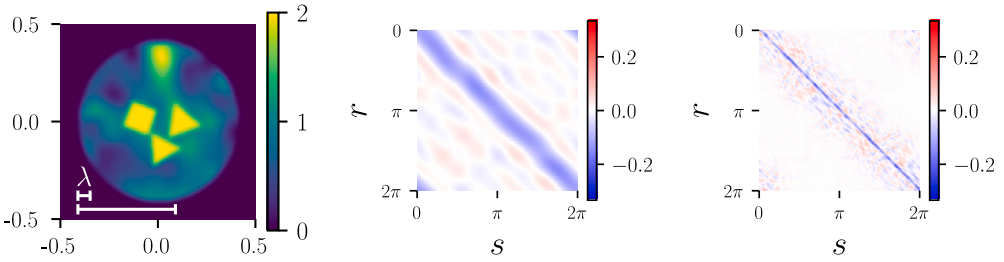
4.3. Implementation of FYNet

We implement the inversion network described in [11] and call it FYNet. This method suggests applying the far-field scaling and a transformation from the receiver and source direction coordinates (r, s) to a new set of variables as summarized in [11, Equation (6)], which we perform using bicubic interpolation. We describe this transformation in Equation (16). We split the complex-valued input into real and imaginary parts along a channel dimension. To implement the action of the adjoint operator F_k^*, we use a composition of three 1-dimensional convolutional layers. We implement the convolutional layers as learnable in the Fourier domain because the expression for F_k^* derived in [11] is local in frequency, but not space. To implement the filtering operator $\left(F_k^* F_k + \mu I\right)^{-1}$, we use a composition of three 2-dimensional convolutional layers. All layers but the last one use ReLU activations. This network outputs an estimate of the scattering potential on a regular polar grid, in coordinates (ρ, ϕ). Following [11], we train the network by minimizing the difference between predictions and targets on the polar grid. We transform to Cartesian coordinates for visualization and computing final test statistics.

Algorithm 2: Training procedure.

Input: Randomly-initialized neural network parameters $\{\theta_1, \ldots, \theta_{N_k}\}$;
Training data samples $\mathcal{D}_n := \left\{ (q^{(j)}, d^{(j)}_{k_1}, \ldots, d^{(j)}_{k_{N_k}}) \right\}_{j=1}^{n}$.

for $t = 1, ..., N_k$ **do**
  Set θ_t as trainable, and freeze all other weights
  if $t < N_k$ **then**
    Train θ_t by optimizing L_t // Equation (14)
  else
    Train θ_t by optimizing $\|\hat{q}_{k_{N_k}} - q\|_2^2$
Set all weights as trainable
Train all weights by optimizing $\|\hat{q}_{k_{N_k}} - q\|_2^2$
Result: Trained neural network parameters $\{\theta_1, ..., \theta_{N_k}\}$.



(a) Scattering potential q (b) Measured d_k, $k = 4\pi$ (c) Measured d_k, $k = 32\pi$

Fig. 4. Fig. 4a shows a typical example from our distribution of scattering potentials, drawn from the test set. Our distribution of scattering potentials has a random low-frequency background field, occluded by piecewise constant geometric shapes. The bottom-left corner shows the wavelength of two incident waves with frequencies $k = 4\pi$ and $k = 32\pi$. Figs. 4b and 4c show the output of the forward model applied to this scattering potential, using these incident frequencies. The real part of u_{scat} is shown. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

5. Experiments

In this section, we describe the setting and results for our numerical investigation of the efficacy of our proposed method.

5.1. Dataset and data generation

Distribution of scattering potentials We define a distribution of scattering potentials $\mathcal{D}$ which has nonzero spatial support on the disk of radius 0.4, with a smoothly varying random background occluded by three randomly placed and randomly sized piecewise-constant shapes. We normalize the scattering potential so the background has minimum and maximum values 0 and 2 respectively, and we normalize the piecewise-constant shapes to have value 2. Fig. 4a shows one such scattering potential from our distribution.

Notably, the contrast of these scattering potentials $\|q\|_\infty = 2$, which is much larger than the contrast used in distributions to evaluate other machine learning methods in the shape reconstruction regime [11,18]. The high-contrast regime is an important experimental setting because it ensures the nonlinearity of the forward model, which is the difficult and interesting problem setting, is captured. The non-constant background also adds to the difficulty of the problem by increasing $\|q\|_2$, which adds to the nonlinearity of the forward model. It also adds much more entropy to $\mathcal{D}$. We use this model to reflect experimental conditions in imaging tasks, wherein backgrounds are rarely known, constant, or homogeneous.

Implementation of $\mathcal{F}$ To implement the forward model, we implement a numerical PDE solver to compute solutions of (1). We implement this by discretizing the scattering domain Ω with a (N_q, N_q) regular grid, with $N_q = 192$. We transform (1) into the Lippmann-Schwinger integral equation and recast the latter as a sparse linear system, which we accelerate using fast Fourier transforms and hardware acceleration. We solve this linear system with GMRES [25] implemented by SciPy [27] to a relative tolerance of 10^{-2}. This formulation allows us to compute the solution u_{scat} on a large, distant ring placed at radius $R = 100$. We compute the solution at $N_r = 192$ equally-spaced positions on this ring, and we repeat this process for each of the $N_s = 192$ equally-spaced source directions. The sources and receivers are located on the same grid. The runtime for evaluating the full forward model for a given q at the lowest and highest incident wave frequencies considered requires 5 and 220 seconds respectively, using one NVIDIA® A40 GPU. Figs. 4b and 4c show the solution u_{scat} produced by our implementation for low and high incident wave frequencies, respectively.

Fan and Ying [11] use a coordinate transformation of the far-field scattering data from coordinates (r, s) to (m, h), where

$$m := \frac{r+s}{2} \quad \text{and} \quad h := \frac{r-s}{2} \tag{16}$$

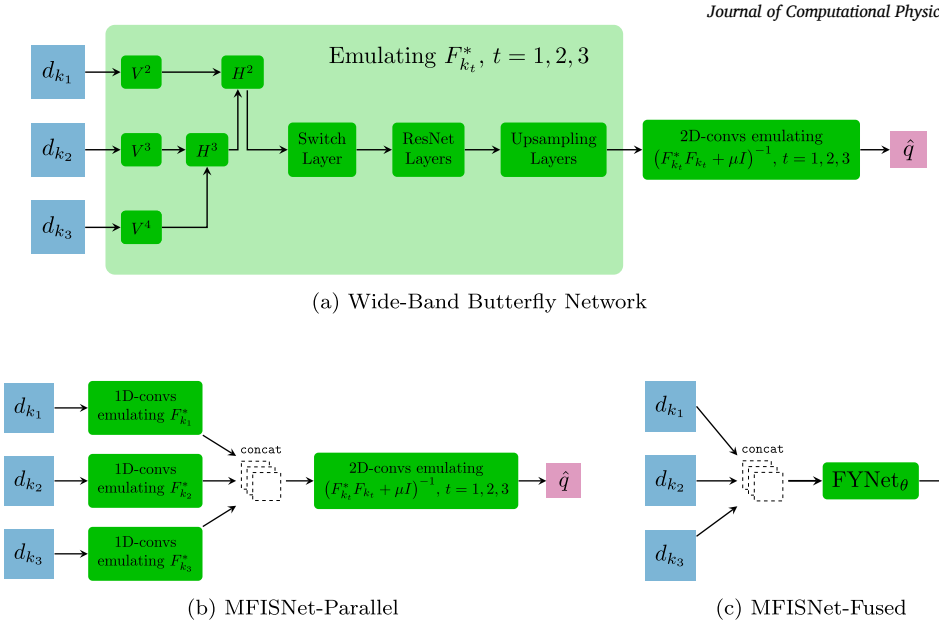


Fig. 5. Alternative neural network architectures for learning the multi-frequency inverse map. All blocks in dark green contain trainable parameters.

as described in [11, Equation (6)]. The variable m ranges from 0 to 2π, while h ranges from $-\pi/2$ to $\pi/2$. As in [11], we choose $N_m = 192$ and $N_h = 96$ to keep the angular sampling frequency fixed for all angular variables considered in the problem. We perform this transformation using bicubic interpolation, and the resulting grid has dimensions $(N_m, N_h) = (192, 96)$.

The FYNet blocks reconstruct images on a regular polar grid with $(N_\rho, N_\phi) = (96, 192)$ pixels, and the radial dimension of our polar grid extends to $\rho_{\max} = 0.5$. Finally, we use bicubic interpolation to transform the model's outputs to the (N_q, N_q) Cartesian grid for final visualization and error measurement.

5.2. Alternative multi-frequency methods

We wish to compare our method, MFISNet-Refinement, with other methods of learning an inverse to the multi-frequency forward map. We design two new methods of extending FBP-inspired single-frequency architectures to the multi-frequency setting. We use the FYNet architecture [11] to instantiate all three of our MFISNet methods, which allows us to focus on the effects caused by different methods of combining multi-frequency data. We show the architectures in Fig. 5. We also compare our method with the Wide-Band Butterfly Network [17,18]. For broader context, we refer to [18] for comparisons between the Wide-Band Butterfly Network and classical multi-frequency methods that do not involve any machine learning such as Full Waveform Inversion (FWI) and a Least-Squares (LS) scheme. The authors show that the Wide-Band Butterfly Network achieves better accuracy than either FWI or LS with much faster inference times with minimal frequency or hyperparameter tuning.

MFISNet-fused In MFISNet-Fused, we use an FYNet architecture that is constrained to learn all of the adjoint operators $F^*_{k_1}, \ldots, F^*_{k_{N_k}}$ jointly. We concatenate the inputs $[d_{k_1}, \ldots, d_{k_{N_k}}]$ along a new channel dimension, so the input array has shape $(N_m, N_h, N_k, 2)$. The concatenated input is then passed into a standard FYNet architecture, with the first layer having slightly wider convolutional channels. The shape of the weights in the 2D convolutional layers is constant in the problem dimensions, so the number of parameters in this network is dominated by the 1D convolutional weights and scales as $O(N_k^2 N_h)$.

MFISNet-parallel In MFISNet-Parallel, we use an extension of FYNet that allows each adjoint operator $F^*_{k_1}, \ldots, F^*_{k_{N_k}}$ to be learned individually. Each input d_{k_t} is input to a unique 1D CNN which emulates $F^*_{k_t}$. After the adjoint operators are learned separately, the results are concatenated along a channel dimension, and the filtering operators $\left(F^*_{k_t} F_{k_t} + \mu I\right)^{-1}$ are emulated jointly by 2D CNN layers. Again, the number of 2D CNN weights does not scale with the problem dimensions, so the number of parameters in this network is dominated by the N_k 1D CNN blocks and scales as $O(N_k N_h)$.

Wide-band butterfly network The Wide-Band Butterfly Network is introduced and defined in [17,18]. Similar to MFISNet-Fused, this architecture also jointly parameterizes the adjoint operators $F^*_{k_1}, \ldots, F^*_{k_{N_k}}$, but it leverages the complementary low-rank property of

Table 1
When holding the number of forward model evaluations $= nN_k$ constant, methods trained on more frequencies outperform methods with fewer frequencies. The final column reports the relative ℓ_2 error mean $\pm$ one standard deviation computed over 1,000 held-out test samples. The lowest mean for each incident frequency set is marked in boldface font.

Performance Comparison (Noiseless)

N_k	$[k_1, k_2, \ldots]$	n	Method Name	Relative ℓ_2 Error
1	$[32\pi]$	10,000	FYNet	0.159 ± 0.033
2	$[16\pi, 32\pi]$	5,000	MFISNet-Fused	0.158 ± 0.030
			MFISNet-Parallel	$\mathbf{0.144 \pm 0.029}$
			MFISNet-Refinement (Ours)	0.152 ± 0.032
3	$[8\pi, 16\pi, 32\pi]$	3,333	Wide-Band Butterfly Network	0.156 ± 0.037
			MFISNet-Fused	0.121 ± 0.024
			MFISNet-Parallel	0.107 ± 0.021
			MFISNet-Refinement (Ours)	$\mathbf{0.094 \pm 0.018}$
4	$[4\pi, 8\pi, 16\pi, 32\pi]$	2,500	MFISNet-Fused	0.103 ± 0.020
			MFISNet-Parallel	0.106 ± 0.021
			MFISNet-Refinement (Ours)	$\mathbf{0.082 \pm 0.019}$
5	$[2\pi, 4\pi, 8\pi, 16\pi, 32\pi]$	2,000	MFISNet-Fused	0.115 ± 0.022
			MFISNet-Parallel	0.109 ± 0.021
			MFISNet-Refinement (Ours)	$\mathbf{0.087 \pm 0.018}$

$F^*_{k_t}$ [15] to hierarchically merge the data using a butterfly network. For this network, we use code provided by the authors.[1] The reference implementation is limited to using data at three incident frequencies, so we only present results in this setting.

5.3. Stabilizing reconstruction by adding frequencies

First, we test whether the intuition built in Section 4 is true in a machine learning context. We test whether machine learning methods that operate on data with multiple incoming wave frequencies are more accurate and stable than single-frequency machine learning methods. To make this comparison fair, we create a sequence of training datasets with number of incident wave frequencies $N_k \in \{1, \ldots, 5\}$ and keep the amount of training data, nN_k, constant for each dataset by suitably adjusting the number of training samples n.

For the $N_k = 1$ dataset, we train an FYNet model, and for the other datasets, we train our three models: MFISNet-Fused, MFISNet-Parallel, and MFISNet-Refinement. For each model, we train for a fixed number of epochs and choose the model weights at the epoch at which the error measured on a validation set of size $n/10$ is minimized. We also use the validation set to search over various hyperparameters, such as the size of 1D and 2D convolutional kernels, the number of channels in the convolutional layers, and optimization hyperparameters, such as step size and weight decay. For the Wide-Band Butterfly Network, we search over the rank of the butterfly factorization, as well as optimization hyperparameters, such as initial learning rate, batch size, and learning rate decay (Appendix B).

We present the results of this experiment in Tables 1 and 2 and Figs. 6 and 7. See also Appendix D for additional empirical results, which include visualizations of predictions on more held-out test samples, and visualizations of predictions of MFISNet-Parallel and MFISNet-Fused. The relatively poor performance of FYNet confirms our belief that we are in a challenging nonlinear problem regime. As more frequencies are added, the multi-frequency methods improve. The performances of MFISNet-Fused and MFISNet-Parallel are comparable, indicating that the distinction between learning adjoint operators separately or jointly does not have a large effect in this setting. For $N_k \geq 3$, the tested methods are uniformly outperformed by our method, MFISNet-Refinement. As we keep increasing N_k, the performance of MFISNet-Refinement plateaus. We hypothesize that our method could be improved by choosing a different set of incident wave frequencies, possibly linearly-spaced in a smaller frequency band. The optimal choice of frequencies could also be learned from data in a reinforcement learning setting [13].

In the case of $N_k = 3$, the Wide-Band Butterfly Network underperforms the other methods. We hypothesize that the weaker performance of the Wide-Band Butterfly Network may be attributed to several factors: on the one hand, the butterfly factorization was inspired by analysis in a weak (linear) scattering regime, but our experiments are in a strong (nonlinear) scattering regime. Also, the Wide-Band Butterfly Network was previously tested in low-contrast settings with sub-wavelength scatterers and a known background, while we are in an experimental setting with high contrast and an unknown, inhomogeneous background.

5.4. Measurement noise

We now turn to the question of robustness against measurement noise. We repeat the experiment above but train and test the models using noisy inputs. We assume an additive noise model similar to [3]. Given a clean input $d_k \in \mathbb{C}^{N_m \times N_h}$ and a desired noise-to-signal ratio δ, we define a noisy input $\tilde{d}_k$ as

[1] https://github.com/borongzhang/ISP_baseline.

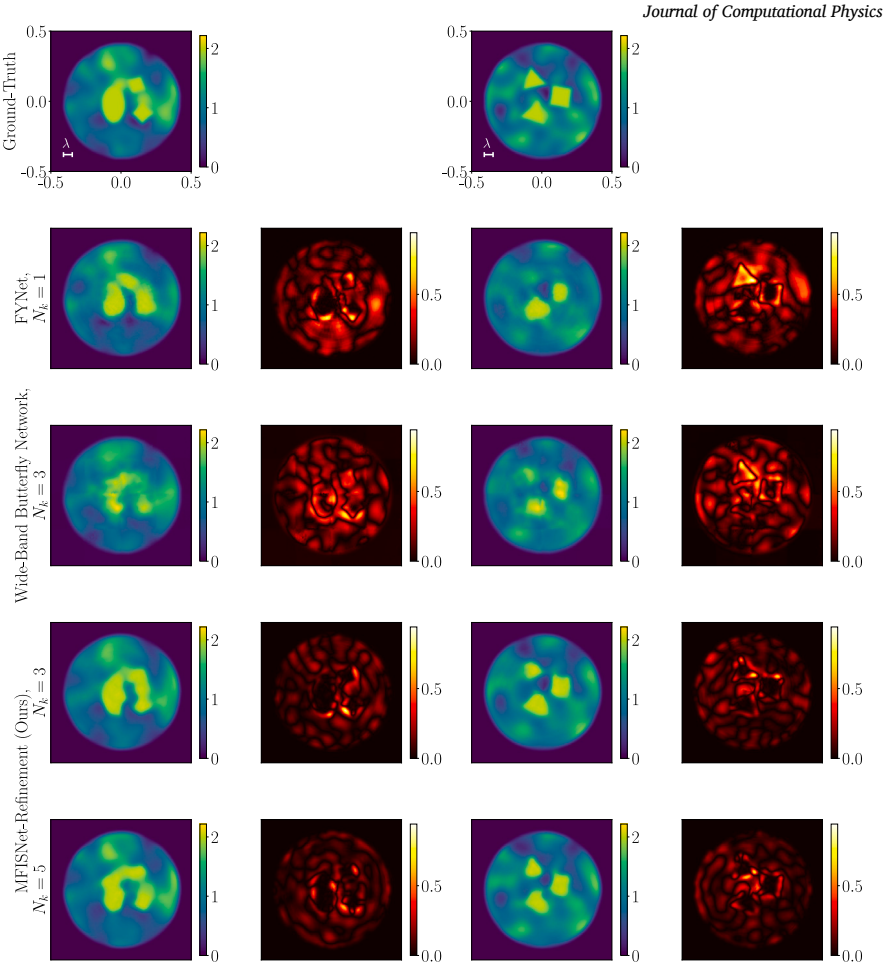


Fig. 6. Sample predictions from models trained on different datasets. Predictions and errors on two held-out test samples are shown. The first row shows the ground-truth scattering potential; in this plot we show the wavelength corresponding to the maximum frequency $k = 32\pi$. The remaining rows show predictions and errors for FYNet, Wide-Band Butterfly Network, MFISNet-Refinement ($N_k = 3$), and MFISNet-Refinement ($N_k = 5$). See Appendix D for additional samples and outputs from MFISNet-Parallel and MFISNet-Fused.

Table 2
The time required to train and evaluate machine learning models. In different settings of N_k, we report the training time for each method. We also report the time required to make predictions on the entire set of 1,000 held-out test samples, performed in ten batches of size 100. As we increase N_k, the number of samples n decreases, so MFISNet-Fused and MFISNet-Parallel see an increase in training speed. For MFISNet-Refinement, as we increase the number of frequencies, we also increase the number of training epochs. The two effects, smaller dataset and more epochs, approximately negate each other, so the training time for MFISNet-Refinement remains approximately constant for different N_k. The testing times for the FYNet and MFISNet models are all approximately equal because they are dominated by a final polar-to-Cartesian coordinate transformation step, which is performed on a single CPU core but can in principle be accelerated.

N_k	n	Method Name	Training Time (seconds)	Testing Time (seconds)
1	10,000	FYNet	870.3	27.3
3	3,333	Wide-Band Butterfly Network	6,453.6	29.9
		MFISNet-Fused	326.2	27.8
		MFISNet-Parallel	256.1	27.8
		MFISNet-Refinement (Ours)	2,033.0	28.2
5	2,000	MFISNet-Fused	306.3	27.6
		MFISNet-Parallel	182.2	27.8
		MFISNet-Refinement (Ours)	2,095.5	28.8

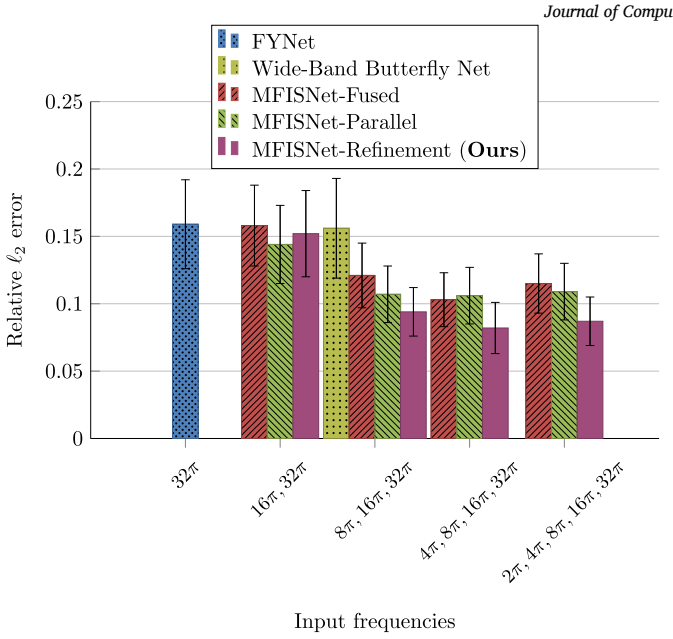


Fig. 7. Illustration of the results of Table 1. The advantage of the MFISNet-Refinement model grows as more low-frequency data is used, even as the total volume of training data is held constant regardless of the number of frequencies.

Table 3
Model evaluation on noisy train and test inputs with $\delta = 0.1$ (see (17) for a description of the noise model). The final column reports the relative ℓ_2 error mean ± one standard deviation computed over 1,000 held-out test samples. The lowest mean for each incident frequency set is marked in boldface font.

Performance Comparison (Noisy, $\delta = 0.1$)

N_k	$[k_1, k_2, \ldots]$	n	Method Name	Relative ℓ_2 Error
1	$[32\pi]$	10,000	FYNet	0.163 ± 0.031
2	$[16\pi, 32\pi]$	5,000	MFISNet-Fused	0.164 ± 0.035
			MFISNet-Parallel	$\mathbf{0.148 \pm 0.029}$
			MFISNet-Refinement (Ours)	0.151 ± 0.031
3	$[8\pi, 16\pi, 32\pi]$	3,333	Wide-Band Butterfly Network	0.156 ± 0.037
			MFISNet-Fused	0.125 ± 0.025
			MFISNet-Parallel	0.110 ± 0.023
			MFISNet-Refinement (Ours)	$\mathbf{0.097 \pm 0.019}$
4	$[4\pi, 8\pi, 16\pi, 32\pi]$	2,500	MFISNet-Fused	0.102 ± 0.021
			MFISNet-Parallel	0.108 ± 0.019
			MFISNet-Refinement (Ours)	$\mathbf{0.086 \pm 0.019}$
5	$[2\pi, 4\pi, 8\pi, 16\pi, 32\pi]$	2,000	MFISNet-Fused	0.112 ± 0.022
			MFISNet-Parallel	0.103 ± 0.020
			MFISNet-Refinement (Ours)	$\mathbf{0.090 \pm 0.017}$

$$\tilde{d}_k = d_k + \sigma(Z_1 + iZ_2), \tag{17}$$

where $\sigma = \delta \dfrac{\|d_k\|_2}{\sqrt{2N_m N_h}}$;

and $[Z_j]_{m,h} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,1)$ for $j = 1, 2$ and $(m, h) \in [N_m] \times [N_h]$.

Under this noise model, the expected noise-to-signal ratio is $\mathbb{E}_{Z \sim \mathcal{N}(0,I)}[\|\tilde{d}_k - d_k\|/\|d_k\|] \approx \delta$. We do not alter the scattering potentials q_i in any way. We train and test all models using noisy inputs, and report the results of this experiment in Table 3. For the Wide-Band Butterfly Network, which takes d_k inputs in the original (r, s) coordinates, we add noise to the input in its original coordinates and divide by $\sqrt{2N_r N_s}$ instead. This results in an equivalent noise profile thanks to the normalization that is grid-invariant. We note that the models lose between 1% - 2% accuracy on test set, suggesting the methods are relatively robust to the presence of measurement noise. Again, our method, MFISNet-Refinement, outperforms all other methods at most values N_k. We present the results of this experiment in Table 3 and visualize the results in Fig. 8.

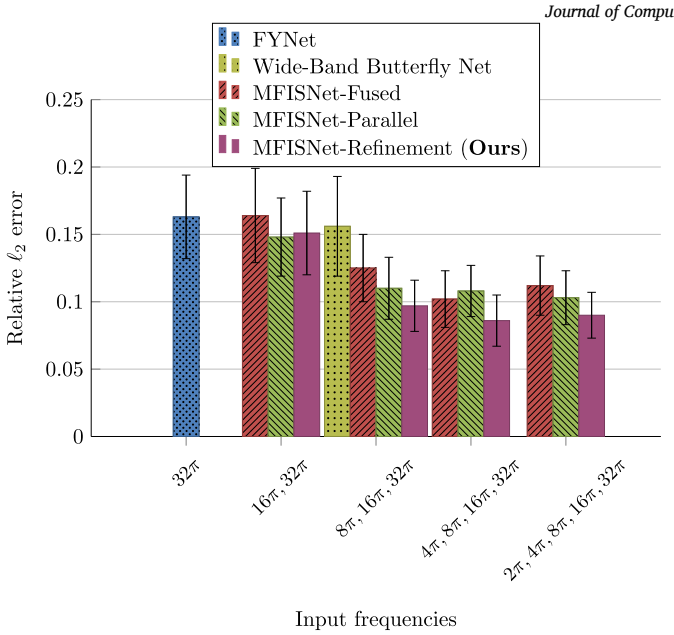


Fig. 8. Illustration of results in the noisy measurement regime (Table 3). The relative performance of the models remains the same as in the noiseless case.

5.5. Investigating the training method

Our method unfolds the reconstruction problem into a sequence of simpler frequency-dependent refinement steps. This emulates the sequential, frequency-dependent structure of the recursive linearization method. Compared to existing methods, our method introduces both a new residual architecture and a new sequential training procedure, so a natural question would be how important each of these factors is to our method's success. To test this question, we investigate two different changes to our training method which remove the progressive refinement structure. We describe each adjustment below. The results are presented in Table 4. In this experiment, we find the accuracy of our method is relatively robust to perturbations of the training method presented in Algorithm 2. This indicates the advantage of our approach is primarily driven by the residual architecture.

No homotopy through frequency Rather than sequentially training each network block after the previous blocks have been optimized, we jointly train all of the blocks by optimizing the loss function

$$\|\hat{q}_{k_{N_k}} - q\|_2^2 + \sum_{t=1}^{N_k-1} \gamma^{N_k-t} \|\hat{q}_{k_t} - \mathsf{LPF}_{2k_t} q\|_2^2 \tag{18}$$

Here γ is a hyperparameter which controls the relative importance of the different loss terms. We tuned over a few choices of γ; see Table B.11 for details.

No progressive refinement Instead of using the intermediate loss terms $\|\hat{q}_{k_t} - \mathsf{LPF}_{2k_t} q\|_2^2$, designed to promote specific network blocks learning different parts of the reconstruction, we train the network by only optimizing the final loss term $\|\hat{q}_{k_{N_k}} - q\|_2^2$. This is the standard training loss used in most other works, including [11,15,18].

5.6. Investigating the speed of training convergence

While Tables 1 and 3 show that MFISNet-Refinement is more accurate than other networks in most settings, Table 2 shows that training our network is slower than the baseline MFISNet-Parallel and MFISNet-Fused architectures. This is because our MFISNet-Refinement architecture is more complex than MFISNet-Parallel or MFISNet-Fused, and our training procedure (Algorithm 2) takes a block-wise approach which requires many training epochs. In this section, we conduct experiments to investigate whether we can reduce the time spent training MFISNet-Refinement. Because the results presented in Section 5.5 suggest that all three training methods considered, Algorithm 2, "No Homotopy through Frequency", and "No Progressive Refinement", produce models which are approximately similar in accuracy, we consider training with all of these methods.

We introduce another training method, designed to mimic Algorithm 2 but converge in fewer training epochs. This method, which we call "Algorithm 2 + Warm-Start Initialization", uses the previous block's parameters θ_{t-1} to initialize the weights of the next block θ_t. This "warm start" is inspired by the fact that neighboring blocks in our MFISNet-Refinement architecture are designed to learn similar functions; each block is meant to increase the resolution of the estimated scattering potential by a small amount. Thus, the

Table 4
Alternative training methods, designed to remove parts of the recursive linearization structure, produce models of similar accuracy to the training procedure discussed in Algorithm 2. This indicates the advantage of our MFISNet-Refinement method is driven primarily by its residual architecture. We describe the alternate training methods in Section 5.5. The optimal hyperparameters for these models are listed in Appendix B.

Training Method	$[k_1, k_2, \ldots]$	n	Relative ℓ_2 Error
No Progressive Refinement	$[16\pi, 32\pi]$	5,000	$\mathbf{0.152 \pm 0.030}$
No Homotopy through Frequency			$\mathbf{0.152 \pm 0.030}$
Algorithm 2			$\mathbf{0.152 \pm 0.032}$
No Progressive Refinement	$[8\pi, 16\pi, 32\pi]$	3,333	$\mathbf{0.094 \pm 0.019}$
No Homotopy through Frequency			0.097 ± 0.018
Algorithm 2			$\mathbf{0.094 \pm 0.018}$
No Progressive Refinement	$[4\pi, 8\pi, 16\pi, 32\pi]$	2,500	0.091 ± 0.020
No Homotopy through Frequency			0.086 ± 0.018
Algorithm 2			$\mathbf{0.082 \pm 0.019}$
No Progressive Refinement	$[2\pi, 4\pi, 8\pi, 16\pi, 32\pi]$	2,000	$\mathbf{0.082 \pm 0.018}$
No Homotopy through Frequency			0.087 ± 0.016
Algorithm 2			0.087 ± 0.018

Table 5
Using warm-starting in Algorithm 2 reduces the training runtime between 20 and 35%. In this table, we report time and epochs required to train MFISNet-Refinement models with different training methods, and the relative ℓ_2 errors of each fully-trained model. The warm-start initialization method and the convergence criterion are described in Section 5.6.

Training Method	$[k_1, k_2, \ldots]$	n	Number of Epochs	Training Time (seconds)	Relative ℓ_2 Error
No Progressive Refinement	$[8\pi, 16\pi, 32\pi]$	3,333	90	1,200.6	0.096 ± 0.020
No Homotopy through Frequency			55	920.3	0.105 ± 0.020
Algorithm 2			215	1,226.4	0.094 ± 0.019
Algorithm 2 + Warm-Start Init.			195	802.8	0.098 ± 0.019
No Progressive Refinement	$[2\pi, 4\pi, 8\pi, 16\pi, 32\pi]$	2,000	45	811.6	0.095 ± 0.019
No Homotopy through Frequency			50	997.8	0.091 ± 0.017
Algorithm 2			260	1,417.5	0.087 ± 0.017
Algorithm 2 + Warm-Start Init.			210	1041.9	0.105 ± 0.020

weights learned by the previous block may be a good starting point for the optimization of the next block. We give full pseudocode for this algorithm in Appendix C.

We set a early stopping criterion, which terminates training if the relative ℓ_2 error has not decreased by at least 10^{-3} over the past 15 epochs. We use this early stopping condition to terminate the "No Homotopy through Frequency" and "No Progressive Refinement" training methods. Similarly, we use this early stopping criterion to terminate each block in Algorithm 2. In Table 5, we measure the number of epochs used and runtime of each training method. We observe that the warm-start initialization decreases the number of epochs and training time of Algorithm 2. For $N_k = 3$ frequencies, this method converges faster than the other three methods. In both cases, using the warm start initialization incurs a slight decrease in accuracy while greatly increasing the speed of training convergence. Different early stopping criteria could achieve different tradeoffs between training speed and accuracy, and we leave the investigation of such criteria to future work.

The time required to complete an epoch of training for Algorithm 2 with and without warm-starting is much shorter than that of the "No Progressive Refinement" and "No Homotopy through Frequency" methods. This is because for the majority of epochs in Algorithm 2, only one block is updated at a time, making the backpropagation of gradients much faster. In the "No Progressive Refinement" and "No Homotopy through Frequency" methods, all trainable parameters are updated in each epoch.

6. Conclusion

This paper investigates the use of multi-frequency data in deep learning approaches to the inverse medium scattering problem in a highly nonlinear, full-aperture regime. We review standard optimization results for this problem, identify recursive linearization as an algorithm particularly well-suited for this problem, and use this insight to design a neural network architecture and training method. We experimentally evaluate our proposed approach, comparing against novel and previously-published methods for combining multi-frequency data. In these comparisons, we find our method outperforms the other methods across a wide range of data settings, including with and without measurement noise.

This work also leaves open important questions about machine learning in different multi-frequency data settings. We leave the investigation of these important problems to future work. The first setting is seismic imaging, where sources and receivers are located on one side of the scattering potential, resulting in limited-aperture measurements. Another setting to consider is full-aperture measurements, with a small fixed or frequency-dependent number of source and receiver directions, as N_s and N_r drive real-world

costs when implementing an imaging system. Real-world imaging systems often encounter noise that is not well-approximated by an additive zero-mean Gaussian noise model; characterizing the robustness of deep learning methods in more realistic noise settings is important future work. Finally, we suggest a distribution of scattering potentials with an unknown, smoothly varying background occluded by strongly scattering shapes. We note that an important open problem is to learn to *segment* the reconstruction into disjoint regions, containing only background or only the strong scatterers.

CRediT authorship contribution statement

Owen Melia: Writing – original draft, Software, Methodology, Data curation, Conceptualization. **Olivia Tsang:** Writing – original draft, Software, Methodology, Data curation, Conceptualization. **Vasileios Charisopoulos:** Writing – review & editing, Supervision, Methodology, Conceptualization. **Yuehaw Khoo:** Writing – review & editing, Supervision, Methodology, Conceptualization. **Jeremy Hoskins:** Writing – review & editing, Supervision, Methodology, Conceptualization. **Rebecca Willett:** Writing – review & editing, Supervision, Project administration, Methodology, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors would like to thank Borong Zhang for sharing his implementation of the Wide-Band Butterfly Network. OM, OT, VC, and RW gratefully acknowledge the support of AFOSR FA9550-18-1-0166 and NSF DMS-2023109. RW, OM, OT, and YK gratefully acknowledge the support of DOE DE-SC0022232. YK gratefully acknowledges the support of NSF DMS-2339439. The team gratefully acknowledges the support of the Margot and Tom Pritzker Foundation.

Appendix A. Distribution of scattering potentials

To generate samples from $\mathcal{D}$, our distribution of scattering potentials, we draw a random smoothly-varying background and three shapes with random sizes, positions, and rotations. This section provides details about the generation of these scattering objects.

The random low-frequency backgrounds were generated by drawing random Fourier coefficients and filtering out the high frequencies using $\mathsf{LPF}_{7.1\pi}$. The resulting background was transformed to Cartesian coordinates, and then shifted and scaled so the maximum value was 2.0 and the minimum value was 0.0. The background was truncated to 0.0 outside of the disk of radius 0.4. Three shapes were randomly chosen among equilaterial triangles, squares, and ellipses. The three shapes had randomly-chosen centers and rotations, constrained to be non-overlapping and fit inside the disk of radius 0.4. The side lengths of the squares and triangles were uniformly sampled from $[0.1, 0.15]$. The major axis lengths of the ellipses were uniformly sampled from $[0.1, 0.15]$, and the minor axis lengths were uniformly sampled from $[0.05, 0.1]$. Finally, $\mathsf{LPF}_{32\pi}$ was applied to the scattering potential.

Appendix B. Hyperparameter search

For our hyperparameter searches, we trained models on a grid of hyperparameters and evaluated them on a validation set every 5 epochs. We found the epoch and hyperparameter setting which produced the lowest error on the validation set, and used those model weights for final evaluation on a held-out test set.

B.1. FYNet and MFISNet models

We train all models using the Adam algorithm, with a batch size of 16 samples. All of the FYNet, MFISNet-Parallel, MFISNet-Fused, and MFISNet-Refinement models tested have 3 1D convolutional layers followed by 3 2D convolutional layers; ReLU activations are used between layers. In the FYNet, MFISNet-Parallel and MFISNet-Refinement models, we use 1D and 2D convolutional kernels with 24 channels following [11]; in the MFISNet-Fused models, we use 1D and 2D convolutional kernels with $24N_k$ channels to adjust for the increased input size. To train MFISNet-Fused, MFISNet-Parallel and MFISNet-Refinement, we search over a grid of architecture and optimization hyperparameters. We report the optimal hyperparameters we found in Tables B.6 to B.9 and B.11. See also Table B.10.

The FYNet model and all MFISNet models use no input or output normalization. The parameters for the 1D convolutional layers were initialized by drawing from a uniform distribution $U[0, 2/d_{in}]$, where d_{in} is the input dimension for a given layer. The parameters for the 2D convolutional layers were initialized by the standard initialization scheme for 2D convolutional layers in `Pytorch`, We performed preliminary investigation into different initialization schemes and found that the initialization scheme had a smaller effect than architecture or optimization hyperparameters. As a result, we decided to restrict our hyperparameter search to the architecture and optimization hyperparameters.

1d kernel size This is the number of frequency components in the 1D convolutional filters emulating F_k^*. We search over values $\{20, 40, 60\}$.

Table B.6
Optimal hyperparameters for FYNet.

Hyperparameter	Data Setting (FYNet)	
	$\delta = 0.0; N_k = 1$	$\delta = 0.1; N_k = 1$
1d kernel size	60	60
2d kernel size	5	5
Weight decay	0.0	0.0
Learning Rate	1×10^{-3}	1×10^{-3}

Table B.7
Optimal hyperparameters for MFISNet-Refinement.

Hyperparameter	Data Setting (MFISNet-Refinement)							
	$N_k = 2$ $\delta = 0.0$	$N_k = 3$	$N_k = 4$	$N_k = 5$	$N_k = 2$ $\delta = 0.1$	$N_k = 3$	$N_k = 4$	$N_k = 5$
1d kernel size	40	20	40	40	40	20	40	20
2d kernel size	7	5	5	5	5	5	5	5
Weight decay	1×10^{-3}	0.0	1×10^{-3}	0.0	0.0	0.0	1×10^{-3}	1×10^{-3}
Learning rate	5×10^{-4}	5×10^{-4}	1×10^{-3}	5×10^{-4}	1×10^{-3}	5×10^{-4}	1×10^{-4}	1×10^{-4}
LR decrease	0.25	1.0	0.25	1.0	0.25	1.0	1.0	1.0

Table B.8
Optimal hyperparameters for MFISNet-Parallel.

Hyperparameter	Data Setting (MFISNet-Parallel)							
	$N_k = 2$ $\delta = 0.0$	$N_k = 3$	$N_k = 4$	$N_k = 5$	$N_k = 2$ $\delta = 0.1$	$N_k = 3$	$N_k = 4$	$N_k = 5$
1d kernel size	40	20	20	20	40	40	20	20
2d kernel size	5	7	5	7	5	5	7	7
Weight decay	0.0	1×10^{-3}	0.0	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-3}	0.0
Learning rate	5×10^{-4}	5×10^{-4}	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-3}	5×10^{-4}

Table B.9
Optimal hyperparameters for MFISNet-Fused.

Hyperparameter	Data Setting (MFISNet-Fused)							
	$N_k = 2$ $\delta = 0.0$	$N_k = 3$	$N_k = 4$	$N_k = 5$	$N_k = 2$ $\delta = 0.1$	$N_k = 3$	$N_k = 4$	$N_k = 5$
1d kernel size	40	20	20	40	20	20	20	40
2d kernel size	5	5	5	5	7	5	5	5
Weight decay	0	1×10^{-3}	0	0	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-3}
Learning Rate	5×10^{-4}	5×10^{-4}	1×10^{-4}	1×10^{-3}	1×10^{-3}	5×10^{-4}	1×10^{-4}	1×10^{-3}

2d kernel size This is the size (in pixels) of the 2D convolutional kernel used in the layers emulating $\left(F_k^* F_k + \mu I\right)^{-1}$. We search over values $\{5, 7\}$.

Weight decay The weight decay parameter adds an ℓ_2 weight regularization term to the loss function. This hyperparameter determines the coefficient of this regularization term. We search over values $\{0.0, 1 \times 10^{-3}\}$.

Learning rate This is the step size for the Adam optimization algorithm. We search over values $\{1 \times 10^{-4}, 5 \times 10^{-4}, 1 \times 10^{-3}\}$.

LR decrease For MFISNet-Refinement models only, we decrease the learning rate each time we begin training a new network block. This parameter determines the multiplicative decrease that we apply to the learning rate. We search over values $\{1.0, 0.25\}$.

B.2. Wide-band butterfly network

To find the optimal Wide-Band Butterfly Network, we optimized over the following hyperparameters. We defined the grid of hyperparameters by taking the original hyperparameter from [18] and adding both higher and lower values, where possible. See Table B.12 for the selected values.

Table B.10
Optimal hyperparameters for the MFISNet-Refinement models trained with the "No Progressive Refinement" training condition (Section 5.5). All of the models were trained on noiseless training samples.

Hyperparameter	Data Setting (MFISNet-Refinement) with No Progressive Refinement Training			
	$N_k = 2$	$N_k = 3$	$N_k = 4$	$N_k = 5$
1d kernel size	40	20	40	40
2d kernel size	5	5	5	7
Weight decay	0.0	1×10^{-3}	1×10^{-3}	0.0
Learning rate	5×10^{-4}	5×10^{-4}	1×10^{-4}	1×10^{-4}

Table B.11
Optimal hyperparameters for the MFISNet-Refinement models trained with the "No Homotopy through Frequency" training condition (Section 5.5). Here, γ is the factor which weights different loss terms (cf. (18)). We searched over values $\gamma = \{0.9, 1.0, 1.1\}$. All of the models were trained on noiseless training samples.

Hyperparameter	Data Setting (MFISNet-Refinement) with No Homotopy through Frequency Training			
	$N_k = 2$	$N_k = 3$	$N_k = 4$	$N_k = 5$
1d kernel size	40	20	40	40
2d kernel size	5	7	7	7
Weight decay	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-3}
Learning rate	5×10^{-4}	5×10^{-4}	5×10^{-4}	1×10^{-4}
γ	1.0	1.0	1.1	1.1

Table B.12
Optimal hyperparameters for Wide-Band Butterfly Networks.

Hyperparameter	Data Setting	
	$\delta = 0.0; N_k = 3$	$\delta = 0.1; N_k = 3$
Rank	50	50
Initial Learning Rate	5×10^{-4}	5×10^{-4}
Learning Rate Decay	0.85	0.95
Sigma	1.5	1.5
Batch Size	16	32

Rank This parameter controls the rank of the compression of local patches in the butterfly factorization. Increasing this rank parameter increases the number of learnable parameters in the part of the network emulating F_k^*. We found that increasing the rank decreased the train and validation errors, and we increased the rank until we were unable to fit the model and data onto a single GPU. We searched over values $\{2, 3, 5, 10, 15, 20, 30, 50\}$.

Initial Learning Rate We decrease the learning rate by a multiplicative factor after 2,000 minibatches, as suggested by [18]. This parameter is the initial learning rate for the optimization algorithm. We searched over values $\{5 \times 10^{-4}, 1 \times 10^{-3}, 5 \times 10^{-3}\}$.

Learning Rate Decay This is the multiplicative decay parameter for the learning rate schedule. We searched over values $\{0.85, 0.95\}$.

Sigma [18] suggest training the network to match slightly-filtered versions of the ground-truth q. This is performed by applying a Gaussian filter to the targets $q^{(i)}$ before training. Sigma is the standard deviation of this Gaussian filter. We do not blur the targets in the test set. We searched over values $\{0.75, 1.125, 1.5\}$.

Batch Size This is the number of samples per minibatch. We searched over values $\{16, 32\}$.

Appendix C. Training method with warm-start initialization

In this section, we give details of the "Algorithm 2 + Warm-Start Initialization" training procedure described in Section 5.6. This training procedure leverages the intuition that sequential blocks in our MFISNet-Refinement architecture learn similar functions. In Algorithm 3 we give full pseudocode for the warm-start training procedure. In this pseudocode, we refer to the trainable parameters

in block t as θ_t. If $t > 1$, these parameters contain the parameters for the FYNet block, which we call $\theta_{t,1}$ and the parameters for the extra filtering layers, which we call $\theta_{t,2}$.

Algorithm 3: Training procedure.

Input: Training data samples $D_n := \left\{ (q^{(j)}, d^{(j)}_{k_1}, \dots, d^{(j)}_{k_{N_k}}) \right\}_{j=1}^{n}$.

for $t = 1, \dots, N_k$ **do**
  if $t = 1$ **then**
    Initialize $t_{1,1}$ randomly
  else if $t = 2$ **then**
    Initialize $\theta_{2,1}$ with $\theta_{1,1}$
    Initialize $\theta_{2,2}$ randomly
  else
    Initialize $\theta_{t,1}$ with $\theta_{t-1,1}$
    Initialize $\theta_{t,2}$ with $\theta_{t-1,2}$
  Set θ_t as trainable, and freeze all other weights
  if $t < N_k$ **then**
    Train θ_t by optimizing L_t // Equation (14)
  else
    Train θ_t by optimizing $\|\hat{q}_{k_{N_k}} - q\|_2^2$
Set all weights as trainable
Train all weights by optimizing $\|\hat{q}_{k_{N_k}} - q\|_2^2$

Result: Trained neural network parameters $\{\theta_1, \dots, \theta_{N_k}\}$.

Appendix D. Additional empirical results

In this section, we illustrate the predictions generated by the different models used in our experiments on randomly-selected test samples from our dataset. For each prediction, we include the associated error plot. In Fig. D.9, we provide more samples comparing FYNet, Wide-Band Butterfly Network, and MFISNet-Refinement. In Fig. D.10, we show sample outputs from MFISNet-Fused and MFISNet-Parallel.

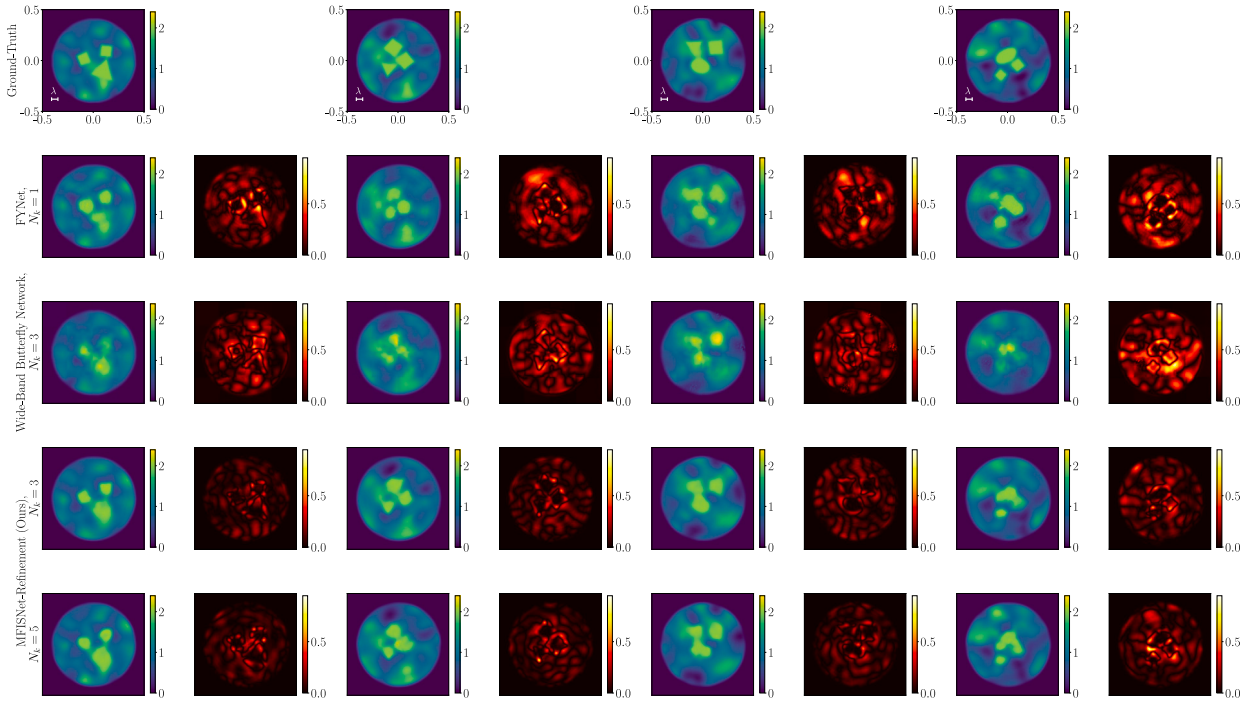


Fig. D.9. Sample predictions from four randomly-selected test samples. The first row shows the ground-truth scattering potential; in this plot we show the wavelength corresponding to the maximum frequency $k = 32\pi$.

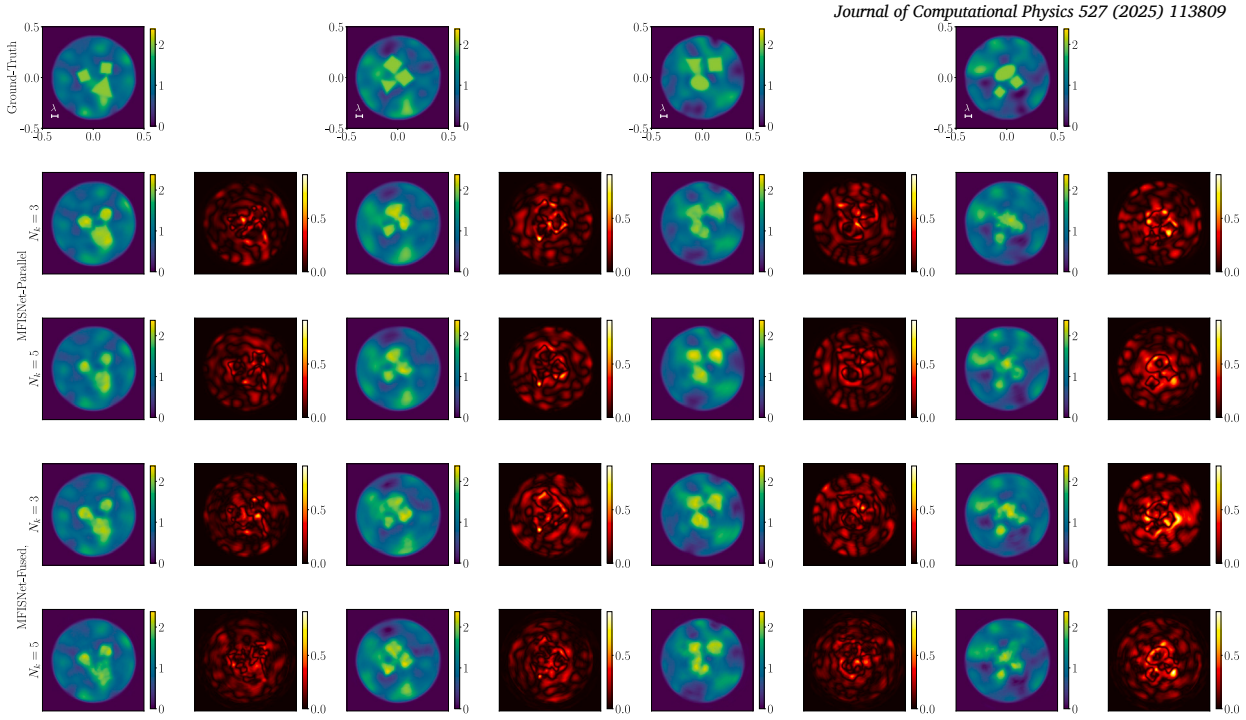


Fig. D.10. Sample predictions from four randomly-selected test samples. The first row shows the ground-truth scattering potential; in this plot we show the wavelength corresponding to the maximum frequency $k = 32\pi$.

Data availability

Our publicly-available GitHub repository https://github.com/meliao/mfisnets contains the following:

- Code for defining and training MFISNet-Refinement, MFISNet-Parallel, and MFISNet-Fused.
- Code for generating the dataset used in our experiments.

Our publicly-available dataset of 10,000 training samples, 1,000 validation samples, and 1,000 test samples can be downloaded from https://doi.org/10.5281/zenodo.14514353.

References

[1] G. Bao, J. Liu, Numerical solution of inverse scattering problems with multi-experimental limited aperture data, SIAM J. Sci. Comput. 25 (2003) 1102–1117, https://doi.org/10.1137/S1064827502409705, publisher: Society for Industrial and Applied Mathematics.
[2] Y. Bengio, J. Louradour, R. Collobert, J. Weston, Curriculum learning, in: Proceedings of the 26th Annual International Conference on Machine Learning, Association for Computing Machinery, New York, NY, USA, 2009, pp. 41–48.
[3] C. Borges, A. Gillman, L. Greengard, High resolution inverse scattering in two dimensions using recursive linearization, SIAM J. Imaging Sci. 10 (2017) 641–664, https://doi.org/10.1137/16M1093562.
[4] M. Born, E. Wolf, A.B. Bhatia, Principles of Optics: Electromagnetic Theory of Propagation, Interference and Diffraction of Light, 7th (expanded) ed, Cambridge University Press, Cambridge [England], 1999.
[5] X. Chen, Z. Wei, M. Li, P. Rocca, A review of deep learning approaches for inverse scattering problems (Invited review), Prog. Electromagn. Res. 167 (2020) 67–81, https://doi.org/10.2528/PIER20030705.
[6] Y. Chen, Recursive Linearization for Inverse Scattering, Technical Report YALEU/DCS/RR-1088, Yale University, 1995.
[7] Y. Chen, Inverse scattering via Heisenberg's uncertainty principle, Inverse Probl. 13 (1997) 253, https://doi.org/10.1088/0266-5611/13/2/005.
[8] D. Colton, R. Kress, Looking back on inverse scattering theory, SIAM Rev. 60 (2018) 779–807, https://doi.org/10.1137/17M1144763.
[9] S. Deshmukh, A. Dubey, R. Murch, Unrolled optimization with deep learning-based priors for phaseless inverse scattering problems, IEEE Trans. Geosci. Remote Sens. 60 (2022) 1–14, https://doi.org/10.1109/TGRS.2022.3214495, conference Name: IEEE Transactions on Geoscience and Remote Sensing.
[10] W. Ding, K. Ren, L. Zhang, Coupling deep learning with full waveform inversion, http://arxiv.org/abs/2203.01799, 2022, arXiv:2203.01799 [cs, math].
[11] Y. Fan, L. Ying, Solving inverse wave scattering with deep learning, Ann. Math. Sci. Appl. 7 (2022) 23–48.
[12] Y. Huang, W. Hao, G. Lin, Hompinns: homotopy physics-informed neural networks for learning multiple solutions of nonlinear elliptic differential equations, Comput. Math. Appl. 121 (2022) 62–73, https://doi.org/10.1016/j.camwa.2022.07.002.
[13] H. Jiang, Y. Khoo, H. Yang, Reinforced inverse scattering, SIAM J. Sci. Comput. 46 (2024) B884–B902, https://doi.org/10.1137/22M153207X.
[14] U.S. Kamilov, H. Mansour, B. Wohlberg, A plug-and-play priors approach for solving nonlinear imaging inverse problems, IEEE Signal Process. Lett. 24 (2017) 1872–1876, https://doi.org/10.1109/LSP.2017.2763583, conference Name: IEEE Signal Processing Letters.
[15] Y. Khoo, L. Ying, SwitchNet: a neural network model for forward and inverse scattering problems, SIAM J. Sci. Comput. 41 (2019) A3182–A3201, https://doi.org/10.1137/18M1222399, publisher: Society for Industrial and Applied Mathematics.

[16] A. Krishnapriyan, A. Gholami, S. Zhe, R. Kirby, M.W. Mahoney, Characterizing possible failure modes in physics-informed neural networks, in: M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, J.W. Vaughan (Eds.), Advances in Neural Information Processing Systems, Curran Associates, Inc., 2021, pp. 26548–26560, https://proceedings.neurips.cc/paper_files/paper/2021/file/df438e5206f31600e6ae4af72f2725f1-Paper.pdf.

[17] M. Li, L. Demanet, L. Zepeda-Núñez, Accurate and robust deep learning framework for solving wave-based inverse problems in the super-resolution regime, http://arxiv.org/abs/2106.01143, 2021, arXiv:2106.01143 [cs, math, stat].

[18] M. Li, L. Demanet, L. Zepeda-Núñez, Wide-band butterfly network: stable and efficient inversion via multi-frequency neural networks, Multiscale Model. Simul. 20 (2022) 1191–1227, https://doi.org/10.1137/20M1383276, publisher: Society for Industrial and Applied Mathematics.

[19] Z. Li, N.B. Kovachki, K. Azizzadenesheli, B. liu, K. Bhattacharya, A. Stuart, A. Anandkumar, Fourier neural operator for parametric partial differential equations, in: International Conference on Learning Representations, 2021, https://openreview.net/forum?id=c8P9NQVtmnO.

[20] L. Lu, P. Jin, G. Pang, Z. Zhang, G.E. Karniadakis, Learning nonlinear operators via deeponet based on the universal approximation theorem of operators, Nat. Mach. Intell. 3 (2021) 218–229, https://doi.org/10.1038/s42256-021-00302-5.

[21] V. Monga, Y. Li, Y.C. Eldar, Algorithm unrolling: interpretable, efficient deep learning for signal and image processing, IEEE Signal Process. Mag. 38 (2021) 18–44.

[22] F. Natterer, The Mathematics of Computerized Tomography, Society for Industrial and Applied Mathematics, 2001.

[23] Y.Z. Ong, Z. Shen, H. Yang, Integral autoencoder network for discretization-invariant learning, J. Mach. Learn. Res. 23 (2022) 1–45.

[24] G. Ongie, A. Jalal, C.A. Metzler, R.G. Baraniuk, A.G. Dimakis, R. Willett, Deep learning techniques for inverse problems in imaging, IEEE J. Sel. Areas Inf. Theory 1 (2020) 39–56, https://doi.org/10.1109/JSAIT.2020.2991563, conference Name: IEEE Journal on Selected Areas in Information Theory.

[25] Y. Saad, M.H. Schultz, GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Stat. Comput. 7 (1986) 856–869, https://doi.org/10.1137/0907058.

[26] S.V. Venkatakrishnan, C.A. Bouman, B. Wohlberg, Plug-and-play priors for model based reconstruction, in: 2013 IEEE Global Conference on Signal and Information Processing, 2013, pp. 945–948.

[27] P. Virtanen, R. Gommers, T.E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S.J. van der Walt, M. Brett, J. Wilson, K.J. Millman, N. Mayorov, A.R.J. Nelson, E. Jones, R. Kern, E. Larson, C.J. Carey, İ. Polat, Y. Feng, E.W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E.A. Quintero, C.R. Harris, A.M. Archibald, A.H. Ribeiro, F. Pedregosa, P. van Mulbregt, SciPy 1.0 Contributors, SciPy 1.0: fundamental algorithms for scientific computing in Python, Nat. Methods 17 (2020) 261–272, https://doi.org/10.1038/s41592-019-0686-2.

[28] L.T. Watson, R.T. Haftka, Modern homotopy methods in optimization, Comput. Methods Appl. Mech. Eng. 74 (1989) 289–305, https://doi.org/10.1016/0045-7825(89)90053-4.

[29] B. Zhang, M. Guerra, Q. Li, L. Zepeda-Núñez, Back-projection diffusion: solving the wideband inverse scattering problem with diffusion models, http://arxiv.org/abs/2408.02866, 2024, arXiv:2408.02866 [cs, math].

[30] B. Zhang, L. Zepeda-Núñez, Q. Li, Solving the wide-band inverse scattering problem via equivariant neural networks, http://arxiv.org/abs/2212.06068, 2023, arXiv:2212.06068 [cs, math].

[31] Q. Zhao, Y. Ma, P. Boufounos, S. Nabi, H. Mansour, Deep born operator learning for reflection tomographic imaging, in: ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2023, pp. 1–5.

[32] M. Zhou, J. Han, M. Rachh, C. Borges, A neural network warm-start approach for the inverse acoustic obstacle scattering problem, J. Comput. Phys. 490 (2023) 112341, https://doi.org/10.1016/j.jcp.2023.112341.