Coupling Deep Learning with Full Waveform Inversion

Wen Ding[*]   Kui Ren[†]   Lu Zhang[‡]

Abstract

Full waveform inversion (FWI) aims at reconstructing unknown physical coefficients in wave equations using the wave field data generated from multiple incoming sources. In this work, we propose an offline-online computational strategy for coupling classical least-squares based computational inversion with modern deep learning based approaches for FWI to achieve advantages that can not be achieved with only one of the components. In a nutshell, we develop an offline learning strategy to construct a robust approximation to the inverse operator and utilize it to design a new objective function for the online inversion with new datasets. We demonstrate through numerical simulations that our coupling strategy improves the computational efficiency of FWI with reliable offline training on moderate computational resources (in terms of both the size of the training dataset and the computational cost needed).

Key words. full waveform inversion, computational inverse problem, deep learning, neural networks, preconditioning, data-driven inversion

AMS subject classifications 2020. 35R30, 49N45, 65M32, 74J25, 78A46, 86A22

1 Introduction

Full waveform inversion (FWI) refers to the process of extracting information on physical parameters of wave equations from data related to the solutions to the wave equations [3, 9, 10, 12, 15, 18, 21, 37, 41, 43, 44, 45, 52, 58, 59, 60]. In seismic imaging, this is manifested as the problem of reconstructing the speed distribution of seismic waves in the interior of the Earth from measured wave field data on the Earth surface. The sources of the measured waves could come either from nature, such as earthquakes, or from geophysical exploration activities by humankind, such as air guns and seismic vibrators. We refer interested readers

---

[*]Department of Applied Physics and Applied Mathematics, Columbia University, New York, NY 10027; wd2288@columbia.edu

[†]Department of Applied Physics and Applied Mathematics, Columbia University, New York, NY 10027; kr2002@columbia.edu

[‡]Department of Applied Physics and Applied Mathematics, Columbia University, New York, NY 10027; lz2784@columbia.edu

to [13, 25, 42, 61] and references therein for overviews on the recent development in the field of FWI for geophysical applications. While the term FWI was mainly coined in the seismic imaging community, FWI also has a wide range of applications in other imaging applications, such as in medical ultrasound imaging [5, 8, 27, 30, 34, 38, 40, 62]. From the practical point of view, the main difference between geophysical and medical FWI is that the quality of the dataset collected in medical applications, both in terms of the variety of source-detector configurations can be arranged and in terms of the frequency contents of the incident sources, is much richer than that of the geophysical FWI dataset.

For the sake of concreteness, let us consider the simplest model of acoustic wave propagation in a heterogeneous medium Ω with wave speed field $m(\mathbf{x}) > 0$. The wave field u solves

$$
\begin{aligned}
\frac{1}{m^2(\mathbf{x})} \frac{\partial^2 u}{\partial^2 t} - \Delta u &= 0, & \text{in } (0, +\infty) \times \Omega \\
\frac{\partial u}{\partial \mathbf{n}} &= h(t, \mathbf{x}), & \text{on } (0, +\infty) \times \partial\Omega
\end{aligned}
\tag{1}
$$

with an appropriate initial condition. Here, $\mathbf{n}$ is the unit outward normal vector of the domain boundary at $\mathbf{x} \in \partial\Omega$. The data that we measure is time traces of the solution to the wave equation (1) at a set of detector locations, say $\Gamma \subset \mathbb{R}^d$, for a period of time, say T, that is,

$$
g := u(t, \mathbf{x})|_{(0,T] \times \Gamma}.
\tag{2}
$$

The objective of FWI in this setting is to recover the unknown wave speed field m in the wave equation (1) from the measured data $u(t, \mathbf{x})_{|(0,T] \times \Gamma}$ collected in a multi-source multi-detector configuration. This is a challenging inverse problem that has rich mathematical and computational content. The main computational strategy, due to the lack of explicit/semi-explicit reconstruction methods, in solving the FWI inverse problem as well as many other model-based inverse problems, is the classical L^2 least-squares formulations where we search for the inverse solution by minimizing the L^2 mismatch between model predictions and observed data. To formulate this more precisely, we assume that we collect data from N_s acoustic sources $\{h_s\}_{s=1}^{N_s}$, and we denote by $f(m; h_s)$ the forward model that takes m to the corresponding wave field data g_s ($1 \leq s \leq N_s$). Then the inverse problem of reconstructing m from measured data g_s^δ aims at solving the following operator equation:

$$
\mathbf{f}(m) = \mathbf{g}^\delta
\tag{3}
$$

where

$$
\mathbf{f}(m) := \begin{pmatrix} f(m; h_1) \\ \vdots \\ f(m; h_s) \\ \vdots \\ f(m; h_{N_s}) \end{pmatrix} \quad \text{and} \quad \mathbf{g}^\delta := \begin{pmatrix} g_1^\delta \\ \vdots \\ g_s^\delta \\ \vdots \\ g_{N_s}^\delta \end{pmatrix}.
$$

The superscript δ denotes the fact that the datum g is polluted by measurement noise. The classical L^2 least-squares method performs the reconstruction by searching for m that

minimizes the mismatch functional (with the possibility of adding a regularization term):

$$\Psi(m) := \frac{1}{2}\|\mathbf{f}(m) - \mathbf{g}^\delta\|^2_{[L^2((0,T]\times\Gamma)]^{N_s}} . \tag{4}$$

This is a challenging numerical optimization problem that has been extensively studied in the past three decades. Many novel methods have been developed to address two of the main challenges: (i) the high computational cost needed to reconstruct high-resolution images of m, and (ii) the abundance of local minimizers (due to the non-convexity of the least-squares functional) that trap iterative reconstruction algorithms; see for instance [13, 25, 51] for a detailed explanation of those challenges among others.

In recent years, there are great interest in the FWI community to use deep learning techniques, based on neural networks, to replace the classical least-squares based inversion methods [1, 4, 16, 22, 23, 28, 31, 32, 36, 39, 49, 54, 55, 56, 57, 63, 65, 66, 67, 68]. Assume that we are given a set of sampled data

$$\{\mathbf{g}_j := (g_{j1}, \cdots, g_{js}, \cdots, g_{jN_s})^\mathfrak{T}, m_j\}_{j=1}^N , \tag{5}$$

where $\{m_j\}_{j=1}^N$ are a set of N velocity profiles sampled from a given distribution and $\{\mathbf{g}_j\}_{j=1}^N$ are the corresponding wave field predictions generated from N_s sources $\{h_s\}_{s=1}^{N_s}$ with the model $\mathbf{g} = \mathbf{f}(m)$. Deep learning methods try to train a neural network, denote by $\mathbf{f}_\theta^{-1}(\mathbf{g})$, with θ denoting the set of parameters (that is, the weight matrices and the bias vectors) of the neural networks, that represents the inverse operator $\mathbf{f}^{-1}$. A training process based on the L^2 loss functional can be formulated as:

$$\widehat{\theta} = \underset{\theta \in \Theta}{\arg\min}\, \mathfrak{L}(\theta) \quad \text{with} \quad \mathfrak{L}(\theta) := \frac{1}{2N}\sum_{j=1}^N \|m_j - \mathbf{f}_\theta^{-1}(\mathbf{g}_j)\|^2_{L^2(\Omega)}$$

where Θ represent the space of parameters of the network, and a regularization term can be added to the loss function $\mathfrak{L}(\theta)$ to help stabilize the training process. The number of samples N needs to be large enough in order for $\mathfrak{L}(\theta)$ to be a good approximation to the expectation of the mismatch over the distribution: $\mathbb{E}_m[\|m - \mathbf{f}_\theta^{-1}(\mathbf{g}(m))\|^2_{L^2(\Omega)}]$. Many other types of loss functions can be used, but we will not dive into this direction. Note that since we know the forward operator $\mathbf{f}$ and are only interested in learning its inverse operator, the datasets used in the training process are synthetic: for each data point $(\mathbf{g}_j, m_j)$, $\mathbf{g}_j$ is constructed by solving the wave equation (1) with the given speed field m_j and evaluate (2).

Numerical experiments, such as those documented in [4, 32, 36, 54, 63, 65, 66, 67, 68], showed that, with sufficiently large training datasets, it is possible to train highly accurate inverse operators that can be used to directly map measured wave field data into the velocity field. This, together with the recent success in learning inverse operators for other inverse problems (see for instance [2, 7, 11, 24, 47, 53] for some examples) has led many to believe, probably overly optimistically, that one can completely replace classical computational inversion with offline deep learning.

Despite the tremendous success in deep learning for FWI, it is still computational challenging to train a once-for-all inverse machine $\mathbf{f}_\theta^{-1}$. First, with the aim of reconstructing

3

high-resolution images of the velocity field $m(\mathbf{x})$, the size of the neural networks to be constructed as a discrete representation of $\mathbf{f}_\theta^{-1}$ is prohibitively large. Second, it is well known that $\mathbf{f} : m \mapsto \mathbf{g}$ is a smoothing operator (between appropriate spaces; see for instance [29] and references therein for more precise mathematical characterization of the statement). The inverse operator is therefore de-smoothing. Learning such operators requires the ability to capture precisely high-frequency information in the training data, and this is very hard to do in the training process as deep neural networks tend to capture low-frequency components of the data much more efficiently than the high-frequency components [46, 50, 64]. On top of the above, the inverse operator $\mathbf{f}_\theta^{-1}$ we learned from model-generated data very often has limited generalization, making it challenging to apply the operator to new measured datasets.

In this work, we propose an offline-online computational strategy for coupling classical least-squares based computational inversion with deep learning based approaches for FWI to achieve advantages that can not be achieved with only one of the components. Roughly speaking, we utilize offline trained approximate inverse of the operator $\mathbf{f}$ to precondition online least-squares based numerical reconstructions. Instead of pursuing high-quality training of highly accurate inverse operator, we train neural networks that only capture the main features in the velocity field. This relaxes dramatically the requirement on both the size of the dataset and the computational resources needed in the training process, and the trained model is more generalizable to other classes of velocity models. Meanwhile, the offline trained approximate inverse is sufficient as a nonlinear preconditioner to improve the speed of convergence of the classical least-squares based FWI numerical reconstruction in the online stage of the inversion.

The rest of the paper is organized as follows. We first describe the proposed coupling strategy in Section 2 in the abstract setting. We then present some preliminary understanding on the training and reconstruction stage of the method in Section 3. In Section 4 we discuss the details of the implementation of the strategy. Extensive numerical simulations are presented in Section 5 to demonstrate the performance of the learning-inversion coupling. Concluding remarks are offered in Section 6.

2 Coupling learning with FWI

Our main objective here is to couple the deep learning based image reconstruction approach with the classical least-squares based image reconstruction method for FWI. More precisely, we utilize the approximate inverse we learned with neural networks to construct a new objective function for least-squares based FWI reconstruction from measured data.

2.1 Robust offline learning of main features

In the offline learning stage, we use deep learning to train an approximate inverse of the operator $\mathbf{f}$. As we outlined in the previous section, our main argument is that the learning process can only be performed reliably on a small amount of dominant features of the velocity

field. First, resolving all details of the velocity field requires over-sized neural networks that demand an exceedingly large amount of training data, not to mention that such networks are computationally formidable to train reliably. Second, large neural networks or large sizes display serious frequency bias in picking up frequency contents in the training datasets [64], making it inefficient in fitting high-frequency components of the velocity field. Despite all the challenges in resolving high-frequency features, it has been shown in different scenarios that learning low-frequency components of the velocity profile can be done in a robust manner [35, 48, 56]. This means that if we take the Fourier representation, the lower Fourier modes of the inverse operator can be learned stably. This good low-frequency approximate inverse is our main interest in the learning stage (even though an accurate inverse itself would be better if one can realistically have it).

Let $\mathfrak{M}$ be the feature map we selected, and $\mathfrak{m}$ the corresponding feature vector, that is,

$$\mathfrak{M} : m(\mathbf{x}) \in \mathcal{M} \longmapsto \mathfrak{m} \in \mathbb{M},$$

where $\mathcal{M} \subseteq L^2(\Omega)$ is the class of velocity field that we are interested in and $\mathbb{M}$ the space of the feature vectors. Motivated by the analysis of weighted optimization in [19, 20], we train a network, which we still denote as $\mathbf{f}_\theta^{-1} : \mathbf{g} \mapsto \mathfrak{m}$, using the synthetic dataset (5), through the optimization problem

$$\widehat{\theta} = \arg\min_{\theta \in \Theta} \mathcal{L}(\theta) \quad \text{with} \quad \mathcal{L}(\theta) := \frac{1}{2N} \sum_{j=1}^{N} \left\| \boldsymbol{\mu} \circledast \left(\mathfrak{m}_j - \mathbf{f}_\theta^{-1}(\mathbf{g}_j) \right) \right\|_{\mathbb{M}}^2 \tag{6}$$

where the weight vector $\boldsymbol{\mu}$ is selected to weight the loss heavily on the features we are interested in while damping the features that are hard to learn stably. The $\circledast$ is used to denote the componentwise product between the vectors involved. The selection of the feature vectors as well as the weighting vector $\boldsymbol{\mu}$ will be discussed in Section 5 in more detail. For the purpose of illustrating the main idea, let us point out that one example is to think of (6) as the equivalence of

$$\widehat{\theta} = \arg\min_{\theta \in \Theta} \frac{1}{2N} \sum_{j=1}^{N} \int_\Omega \left(\int_\Omega \mu(\mathbf{x} - \mathbf{y}) \big(m_j(\mathbf{y}) - \mathbf{f}_\theta^{-1}(\mathbf{g}_j)(\mathbf{y}) \big) d\mathbf{y} \right)^2 d\mathbf{x}$$

in the Fourier domain, i.e. when the features we use are Fourier modes, with $\boldsymbol{\mu}$ the Fourier transform of the kernel $\mu(\mathbf{x})$. If we take μ to be a smoothing kernel, such as a Gaussian kernel, $\boldsymbol{\mu}$ will decay fast with the increase of the frequency. In such a case, the learning problem (6) focuses on the lower Fourier modes of the velocity field m.

Weighted optimization schemes of the form (6) with weight $\boldsymbol{\mu}$ to emphasize dominant features in the learning problems have been extensively studied in the learning and inverse problems community; see [20] and references therein. When the feature we selected are Fourier basis, it has been shown that correct selecting of the weight $\boldsymbol{\mu}$ in the training scheme can lead to more robust learning results for a class of models $\mathbf{f}_\theta^{-1}$ following certain distributions, sometimes at the expense of learning accuracy, with better generalization capabilities [20]. This is the main motivation for us to adopt this strategy for our purpose in this research.

2.2 New objective function for online inversion

In the online reconstructions stage, we utilize the approximate inverse we trained to construct a new objective function for FWI image reconstruction from given noisy data $\mathbf{g}^\delta$. More precisely, instead of solving the model (3), we aim at solving the modified model

$$\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}\big(\mathbf{f}(m)\big) = \widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}(\mathbf{g}^\delta) \tag{7}$$

where

$$\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1} := \mathfrak{M}^{-1} \circ \mathbf{f}_{\widehat{\theta}}^{-1} : \ \mathbf{g} \mapsto m$$

is the learned approximate to $\mathbf{f}^{-1}$ (while $\mathbf{f}_{\widehat{\theta}}^{-1} : \ \mathbf{g} \mapsto \mathfrak{m}$ is the learned representation in $\mathfrak{M}$).

The least-squares formulation for the reconstruction problem now takes the form

$$\widehat{m} = \underset{m \in \mathcal{M}}{\arg\min}\, \Phi(m)\,, \tag{8}$$

with

$$\Phi(m) := \frac{1}{2}\|\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}\big(\mathbf{f}(m)\big) - \widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}(\mathbf{g}^\delta)\|_{L^2(\Omega)}^2 + \frac{\gamma}{2}\|\mathfrak{M}^{-1}(\boldsymbol{\mu}^{-1} \circledast \mathfrak{m})\|_{L^2(\Omega)}^2\,. \tag{9}$$

The last term in the objective functional is a Tikhonov regularization functional that imposes a smoothness constraint on the target velocity field. This smoothness constraint is selected such that it is consistent with the training process. The natural initial guess for any iterative solution scheme for this minimization problem is $m_0 := \widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}(\mathbf{g}^\delta)$.

Let us emphasize that there is a significant difference between the L^2 objective function $\Phi(m)$ we introduced in (9), ignoring the regularization term, and the standard L^2 objective function $\Psi(m)$ defined in (4). Our objective function $\Phi(m)$ measures the mismatch between the approximations of predicted velocity field and the true velocity field corresponding to the measured data, while the standard objective function $\Psi(m)$ measures the mismatch between predicted wave field data with measured wave field data. In other words, our objective function works on the parameter space (also called the model space in the FWI literature, that is, the space of the velocity field) while the standard objective function is defined on the signal space (that is the space of wave field signals at the detectors). With reasonably-trained $\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}$, the functional $\Phi(m)$ has advantageous landscape for optimization purpose as we will demonstrate in the numerical simulations in Section 5.

2.3 The benefits of the coupling approach

The offline-online coupling scheme we proposed allowed us to focus on training a robust approximate inverse instead of the exact inverse. This makes the learning process more stable and also requires less computational resources (in terms of the amount of data, the size of the network, and the computational cost for optimization) than training an accurate inverse. Moreover, the sacrifice in accuracy brings better generalizability for the learned approximate inverse. On the computational side, the trained approximate inverse serves as a "preconditioner" for the inversion process. It can not only provide a good initial guess for the reconstruction but also simplify the landscape of the optimization problem.

We finish this section with the following remark. In the ideal case when all the operators involved are invertible as they should be, the solution to (7) is identical to the solution to (3), assuming that $\mathbf{g}$ indeed lives in the range of $\mathbf{f}$. Therefore, our formulation does not change the true solution to the original inverse problem. However, as we will see, the new formulation utilizes the result of learning to facilitate the FWI reconstruction in terms of saving computational cost as well as making the optimization landscape more desirable.

3   Formal understanding of the coupling

We now attempt to gain a more systematic understanding of the coupling strategy. As we have argued in the previous sections, it is computationally challenging to train neural networks that are accurate approximations of the inverse operator and are very generalizable at the same time. However, there is certainly some dominant information in the inverse operator that we could extract with learning and this is the approximate inverse that we are interested in constructing.

3.1   Elements of network training

Due to the fact that the training data we have are generated from exactly the same operator we are trying to represent with the neural network, the learning process we have is much more under control than those purely data-driven learning problems in applications. Here we highlight a few critical issues in the learning process without getting into the details of the implementation of the learning algorithm.

Sampling training data.   To learn the inverse operator, we need to pay attention to both its input space and its output space. While our focus will be to learn the low-frequency component of the inverse operator, we want the training data to include as much high-frequency information as possible to gain generalization capability in the input space. Let $\mathbf{K}_{\text{out}}$ be the frequency range for the network output that we are interested in recovering and $\mathbf{K}_{\text{in}}$ the frequency range of the velocity fields that generated the wavefield data. We construct the training dataset as

$$\{m_j(\mathbf{x}), \mathbf{g} := \mathbf{f}(m_j(\mathbf{x}) + \widetilde{m}_j(\mathbf{x}))\}_{j=1}^N$$

where $\widetilde{m}_j(\mathbf{x})$ are selected such that $\mathcal{F}(\widetilde{m}_j)(\mathbf{k}) = 0 \ \forall \mathbf{k} \in \mathbf{K}_{\text{out}}$, and $\mathcal{F}(\widetilde{m}_j)(\mathbf{k}) \neq 0 \ \forall \mathbf{k} \in \mathbf{K}_{\text{in}} \backslash \mathbf{K}_{\text{out}}$ ($\mathcal{F}(m)$ denoting the Fourier transform of m). In other words, we train the network with input wavefield data having richer frequency content of the velocity field than the output velocity field. This constructing enrich the frequency content of the input data but does not increase the computational cost of the training process.

The well-known result on the differentiability of the data $\mathbf{g}$ with respect to m, quoted in the proposition below, indicates that the input space of the inverse operator, i.e. the range of the forward operator, is quantitatively smoother than the output space (the velocity space)

that we are working with. Therefore, a well-trained network approximation should have good interpolation ability in applications when the space of velocity field we are interested in working with is sufficiently smooth.

Proposition 3.1 ([6, 17, 29]). *Let Ω be a smooth domain and $h(t, \mathbf{x})$ be the restriction of a $\mathcal{C}^1$ function on $\partial\Omega$. Assume further that $m \in \mathcal{C}^2(\bar{\Omega}) \cap [\underline{m}, \overline{m}]$ for some $0 < \underline{m} < \overline{m} < +\infty$. Then the map: $\mathbf{f}(m) : m \longmapsto \mathbf{g}$ is Fréchet differentiable at any $m \in \mathcal{C}^2(\bar{\Omega}) \cap [\underline{m}, \overline{m}]$.*

The result is standard. We refer interested reader to [6, 17, 29] and references therein for more precise formulations of it in different scenarios. This result also ensures that if we can train a stable network, then the learning quality is guaranteed; see Lemma 3.3 below.

Network training error. Our main objective of this work is to focus the learning process on the low-frequency content of the output of the inverse operator. We do this with the weighted optimization scheme (6) by selecting weight $\boldsymbol{\mu}$ that penalizes heavily the low-frequency component of the mismatch of true data and the network prediction. The impact of such weighting schemes on the learning results have been analyzed extensively; see [19, 20] and reference therein. We illustrate this in an extremely simplified setting. Let $\mathbf{F} := (\mathbf{f}^{-1})'(m_0)$ be the linearization of $\mathbf{f}^{-1}$ at m_0 for a one-dimensional medium. Assume that the learning loss function $\mathcal{L}(\theta)$ in (6) is minimized to the order of ε^2 in the training process. Then on the leading order, the trained $\mathbf{F}$ satisfies

$$\boldsymbol{\mu} \circledast (\mathbf{m} - \mathbf{F}\mathbf{G}) \sim \mathcal{O}(\varepsilon),$$

where $\mathbf{m} = [\mathbf{m}_1, \cdots, \mathbf{m}_N]$ is the matrix whose columns are vectors of the Fourier coefficients of the training velocity samples $\{m_j\}_{j=1}^N$, $\mathbf{G} = [\mathbf{g}_1, \cdots, \mathbf{g}_N]$ is a matrix whose columns are vectors of the input data, and $\mathcal{O}(\varepsilon)$ is a diagonal matrix of size order ε. The trained linearized inverse operator, when applied to a new input data $\mathbf{g}^\delta$, gives the result

$$\mathbf{F}\mathbf{g}^\delta \sim \left(\mathbf{m} - \boldsymbol{\mu}^{-1} \circledast \mathcal{O}(\varepsilon)\right)\mathbf{G}^{\mathfrak{T}}(\mathbf{G}\mathbf{G}^{\mathfrak{T}})^{-1}\mathbf{g}^\delta.$$

The nature of $\boldsymbol{\mu}$ clearly indicates that the relative error in the learned output is larger in the high-frequency Fourier modes.

3.2 Inversion with accurate training

As we discussed in the previous section, when the network is trained so that $\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1} = \mathbf{f}^{-1}$, the objective function $\Phi(m)$, defined in (9), in the reconstruction stage is a convex functional of m. When the learning is not perfect but sufficiently accurate, the functional $\Phi(m)$ still has an advantageous landscape. This is given in the following result.

Lemma 3.2. *Let $\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1} : \mathbf{g} \in [L^2([0, T] \times \Gamma)]^{N_s} \mapsto m \in L^2(\Omega)$ be an approximation to $\mathbf{f}^{-1}$ with Fréchet derivative at $\mathbf{g}$ given as $d\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}[\mathbf{g}]$. Assume that*

$$\sup_m \|\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}\big(\mathbf{f}(m)\big) - m\|_{L^2(\Omega)} \le \epsilon \tag{10}$$

8

and

$$A := 1 + \sup_{\mathbf{g}} \|d\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}[\mathbf{g}]\|_{\mathcal{L}([L^2([0,T]\times\Gamma)]^{N_s};L^2(\Omega))} < +\infty \tag{11}$$

for some $\epsilon > 0$ and $\mathbf{g}^{\boldsymbol{\delta}} = \mathbf{f}(m) + \boldsymbol{\delta}$ for some $\boldsymbol{\delta}$ with $\|\boldsymbol{\delta}\|_{[L^2([0,T]\times\Gamma)]^{N_s}}$ sufficiently small. Then we have that

$$\left| \|\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}\big(\mathbf{f}(m)\big) - \widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}(\mathbf{g}^{\boldsymbol{\delta}})\|_{L^2(\Omega)} - \|m - m_0\|_{L^2(\Omega)} \right| \le 2\epsilon + A\|\boldsymbol{\delta}\|_{[L^2([0,T]\times\Gamma)]^{N_s}} . \tag{12}$$

Proof. We denote by $r(m) = \widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}\big(\mathbf{f}(m)\big) - m$. We then have, by Taylor's theorem, that

$$\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}(\mathbf{g}^{\boldsymbol{\delta}}) = \widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}(\mathbf{f}(m_0) + \boldsymbol{\delta}) = m_0 + r(m_0) + d\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}[\mathbf{f}(m_0)]\big(\boldsymbol{\delta}\big) + o(\boldsymbol{\delta}),$$

where $\lim_{\boldsymbol{\delta}\to 0} \dfrac{\|o(\boldsymbol{\delta})\|_{L^2(\Omega)}}{\|\boldsymbol{\delta}\|_{[L^2([0,T]\times\Gamma)]^{N_s}}} = 0$. We therefore have

$$\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}\big(\mathbf{f}(m)\big) - \widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}(\mathbf{g}^{\boldsymbol{\delta}}) = m - m_0 + r(m) - r(m_0) - d\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}[\mathbf{f}(m_0)]\big(\boldsymbol{\delta}\big) + o(\boldsymbol{\delta}) .$$

We can now use the triangle inequality to conclude that

$$\left| \|\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}\big(\mathbf{f}(m)\big) - \widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}(\mathbf{g}^{\boldsymbol{\delta}})\|_{L^2(\Omega)} - \|m - m_0\|_{L^2(\Omega)} \right|$$

$$\le \|r(m) - r(m_0) - d\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}[\mathbf{f}(m_0)]\big(\boldsymbol{\delta}\big) + o(\boldsymbol{\delta})\|_{L^2(\Omega)}$$

$$\le \|r(m)\|_{L^2(\Omega)} + \|r(m_0)\|_{L^2(\Omega)} + \|d\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}[\mathbf{f}(m_0)]\big(\boldsymbol{\delta}\big)\|_{L^2(\Omega)} + \|o(\boldsymbol{\delta})\|_{L^2(\Omega)}$$

$$\le 2\epsilon + A\|\boldsymbol{\delta}\|_{[L^2([0,T]\times\Gamma)]^{N_s}}$$

where the last step comes from the assumptions in (10) and (11). The proof is complete. $\quad\square$

This result says that the new objective function $\Phi(m)$ in (9) behaves similarly to the quadratic functional $\|m - \widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}(\mathbf{g})\|_{L^2(\Omega)}^2$ provided that the trained $\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}$ is accurate enough. It is clear that we can replace the strong assumption on the accuracy of $\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}$, $\sup_m \|\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}\big(\mathbf{f}(m)\big) - m\|_{L^2(\Omega)} \le \epsilon$, with the weaker assumption $\|\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}\big(\mathbf{f}(m)\big) - m\|_{L^2(\Omega)} \le \epsilon\|m\|_{L^2(\Omega)}$, in which case the 2ϵ term in the bound (12) will be replaced by $\epsilon(\|m\|_{L^2(\Omega)} + \|m_0\|_{L^2(\Omega)})$. The conclusion still holds.

Due to the smoothing property of the forward operator as given in Proposition 3.1, the stability of the trained inverse operator, measured by the boundedness of its Fréchet derivative, is enough to ensure the accuracy of the neural network reconstruction. Therefore, if we could train network approximations with such stability property, they have good generalization capabilities in the output space.

Lemma 3.3. *Let $m, m_0 \in \mathcal{C}^2(\Omega) \cap [\underline{m}, \overline{m}]$ for some $0 < \underline{m} < \overline{m} < +\infty$. Then, when $\|m - m_0\|_{L^2(\Omega)}$ is sufficiently small, there exists a constant $\mathfrak{c}$ such that*

$$\|\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}\big(\mathbf{f}(m)\big) - \widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}\big(\mathbf{f}(m_0)\big)\|_{L^2(\Omega)} \le \mathfrak{c}\|m - m_0\|_{L^2(\Omega)} \tag{13}$$

Proof. By Proposition 3.1, the map $m \mapsto \mathbf{g} := \mathbf{f}(m)$ is Fréchet differentiable with the derivative at m in direction $\widetilde{m}$ denoted as $d\mathbf{f}[m](\widetilde{m})$. By Taylor's theorem, we have

$$\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}\big(\mathbf{f}(m)\big) = \widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}\big(\mathbf{f}(m_0) + d\mathbf{f}[m_0](m - m_0) + o(m - m_0)\big)$$
$$= \widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}\big(\mathbf{f}(m_0)\big) + d\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}[\mathbf{f}(m_0)]\big(d\mathbf{f}[m_0](m - m_0)\big) + \widetilde{o}(m - m_0),$$

where $\lim\limits_{m \to m_0} \dfrac{\|o(m-m_0)\|_{[L^2([0,T]\times\Gamma)]^{N_s}}}{\|m-m_0\|_{[L^2(\Omega)}} = \lim\limits_{m \to m_0} \dfrac{\|\widetilde{o}(m-m_0)\|_{L^2(\Omega)}}{\|m-m_0\|_{[L^2(\Omega)]}} = 0$. We therefore have

$$\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}\big(\mathbf{f}(m)\big) - \widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}(\mathbf{f}(m_0)) = d\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}[f(m_0)]\big(d\mathbf{f}[m_0](m - m_0)\big) + \widetilde{o}(m - m_0)\,.$$

The bound in (13) then follows from the assumption (11). $\qquad\square$

When the class of velocity models is sufficiently nice, for instance, when each $m(\mathbf{x})$ can be represented with a small number of Fourier coefficients in a narrow frequency band, one can hope that accurate training is achievable. When this is the case, Lemma 3.2 and Lemma 3.3 ensure that the learned model can be utilized to facilitate the FWI reconstruction with the new dataset.

3.3 Computational simplifications

The reconstruction stage of the coupling can be greatly simplified when the training of the neural network approximation is sufficiently accurate.

First, the coupling method will degenerate to a deep learning based method when we have confidence in our ability in training an accurate deep neural network representation of the inverse operator in FWI. Indeed, when $\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1} = \mathbf{f}^{-1}$, that is, $\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}$ is exactly the inverse, the reconstruction step (8) simplifies to

$$\widehat{m} = \underset{m \in \mathcal{M}}{\arg\min} \, \frac{1}{2}\|m - \widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}(\mathbf{g}^\delta)\|_{L^2(\Omega)}^2 + \frac{\gamma}{2}\|\nabla m\|_{L^2(\Omega)}^2\,,$$

assuming, only for the sake of simplifying the notation, that the weighting operator $\mu(\mathbf{x}-\mathbf{y})$ is taken as an integral operator such that $\boldsymbol{\mu}^{-1}(\mathbf{k}) = \mathbf{k}$. This gives a fast inversion for the new data and immediately leads to the optimal selection of the regularization parameter when the regularization term is not too complicated. In this case, we simply did a post-process on the deep learning reconstruction given by the operator $\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}$. The solutions to this are explicitly given as

$$\widehat{m} = (\mathcal{I} + \gamma\Delta)^{-1}\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}(\mathbf{g}^\delta)\,,$$

where $\mathcal{I}$ is the identity and Δ is the Laplacian operator. Therefore, m is simply a smoothed version of the result produced by the trained neural network, $\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}(\mathbf{g}^\delta)$. The exact form of the smoothing effect depends on the selection of μ.

Second, when we can not train an accurate $\mathbf{f}^{-1}$, but can train a good approximation to the inverse, that is, when the operator $\mathcal{I} - \widehat{\mathbf{f}}_{\widehat{\theta}}^{-1} \circ \mathbf{f}$ is not zero but small in an appropriate

operator norm, the FWI problem (7) can be solved by using Neumann series. More precisely, we can rewrite (7) as

$$m - K(m) = \widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}(\mathbf{g}^\delta), \quad K := \mathcal{I} - \widehat{\mathbf{f}}_{\widehat{\theta}}^{-1} \circ \mathbf{f}$$

whose solution can be expressed in a Neumann series as

$$\widehat{m} = (\mathcal{I} - K)^{-1}(\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}(\mathbf{g}^\delta)) = \sum_{j=0}^{\infty} K^j\left(\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}(\mathbf{g}^\delta)\right). \tag{14}$$

The better the approximation $\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}$ is to $\mathbf{f}^{-1}$, the faster the series converges. For the training we had, see more discussion in Section 5, a few terms of the Neumann series often provide sufficient accuracy for the reconstruction.

Let us emphasize that by the informal analysis in Section 3.1, the error in the learning implies roughly that $|\mathcal{F}(m - K(m))(\mathbf{k})| \sim \zeta(\mathbf{k})\|m\|_{L^2(\Omega)}$ with $\zeta(\mathbf{k})$ large for large $|\mathbf{k}|$. Due to the fact that the operator norm of $\mathcal{I} - K$ is bounded below by $\max_{\mathbf{k}} \zeta(\mathbf{k})$, this means that the convergence speed of the Neumann series is controlled by the worst training error in the (high-frequency) Fourier modes.

3.4 Utilizing learning outside of training domain

It is important to point out that the weight $\boldsymbol{\mu}$ in the weighted training scheme (6) should be selected to emphasize the low-frequency components of the output and penalize the high-frequency components. It should not completely remove the high-frequency components. If it does, then the high-frequency components of the velocity field in the reconstruction stage can not be recovered with the optimization problem (8). This is an obvious yet important observation that we summarize as a lemma to emphasize it.

Lemma 3.4. *Let $\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}$ be such that for any m, $\mathcal{F}[\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}(\mathbf{f}(m))](\mathbf{k}) = \mathbf{0} \ \forall |\mathbf{k}| > k_0$, and $\widehat{m}$ be reconstructed from (8) with a gradient-based iterative scheme or the Neumann series method in (14). Then $\mathcal{F}[\widehat{m}](\mathbf{k}) = \mathbf{0} \ \forall |\mathbf{k}| > k_0$.*

Proof. Under the assumption on $\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}$, it is straightforward to check that $\mathcal{F}(m_0)(\mathbf{k}) = 0$ $(m_0 := \widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}(\mathbf{g}^\delta)) \ \forall |\mathbf{k}| > k_0$, and $\mathcal{F}(K^j m_0)(\mathbf{k}) = 0 \ \forall |\mathbf{k}| > k_0$, for any $j \geq 1$. Therefore $\mathcal{F}(\widehat{m})(\mathbf{k}) = 0 \ \forall |\mathbf{k}| > k_0$. Let m_ℓ be the ℓ-th iteration of a gradient based iterative scheme, then $\mathcal{F}(r(m_\ell))(\mathbf{k}) = 0 \ (r(m) := \widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}\left(\mathbf{f}(m)\right) - \widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}(\mathbf{g}^\delta)) \ \forall |\mathbf{k}| > k_0$. This leads to the fact that $\mathcal{F}\left(d\Phi[m_\ell](\delta m)\right)(\mathbf{k}) = 0$ for any δm. Therefore, $\mathcal{F}(m_{\ell+1})(\mathbf{k}) = 0 \ \forall |\mathbf{k}| > k_0$. The rest of the proof follows from an induction. $\qquad \square$

For any velocity field that can be written as $m_b + \delta m$ with m_b the prediction of the trained neural network and δm outside of the range of the neural network but either has small amplitude (compared to that of m) or has large amplitude by small support compared

11

to the size of the domain (in which case δm is very localized), we can recover δm with an additional linearized reconstruction step. We linearize the inverse problem around the network prediction $\mathbf{f}_{\widehat{\theta}}^{-1}(\mathbf{g}^\delta)$. The reconstruction can be performed with a classical migration scheme, or equivalently by minimizing the following quadratic approximation to the functional (9):

$$\Psi_Q(m) = \frac{1}{2}\left\|\mathbf{f}(m_b) + d\mathbf{f}[m_b]\Big(m - m_b\Big) - \mathbf{g}^\delta\right\|_{[L^2((0,T]\times\Gamma)]^{N_s}}^2 + \frac{\gamma}{2}\|\nabla m\|_{L^2(\Omega)}^2, \qquad (15)$$

where $m_b := \mathbf{f}_{\widehat{\theta}}^{-1}(\mathbf{g}^\delta)$.

4 Computational implementation

We now provide some details on the implementation of the coupling framework we outlined in the previous section. For computational simplicity, we focus on the implementation in two spatial dimensions even though the methodology itself is independent of the dimension of the problem.

4.1 Computational setup
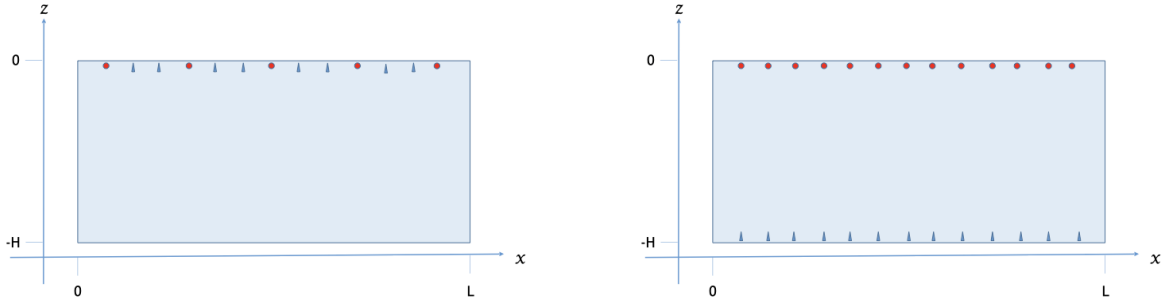
Figure 1: The two-dimensional computational domain $\Omega = (0, L) \times (-H, 0)$ for wave propagation. Periodic boundary conditions are imposed on the left and right boundaries. In geophysical applications, sources and detectors are placed on the top boundary (left) while in medical ultrasound applications, sources (red dots) and detectors (blue triangles) can be placed on both the top and the bottom boundaries (right).

For the purpose of being concrete, we first describe briefly the geometrical setting under which we implement the learning and reconstruction algorithms. Let $\mathbf{x} = (x, z)$. The computational domain of interests is $\Omega = (0, L) \times (-H, 0)$. We impose periodic boundary conditions on the left and right boundaries of the domain. Probing sources and detectors are placed on the top and bottom boundaries $\Gamma_t = (0, L) \times \{0\}$ and $\Gamma_b = (0, L) \times \{-H\}$, depending on the exact applications we have in mind. In geophysical applications, source and detectors are both placed on the top boundary while in medical ultrasound type of applications, source and detectors could be placed on the opposite sides; see Figure 1 for

12

an illustration. Under this setup, the wave equation (1) with a source $h(t, x)$ on the top boundary and a reflective bottom boundary takes the form

$$
\begin{aligned}
\frac{1}{m^2}\frac{\partial^2 u}{\partial^2 t} - \Delta u &= 0, & \text{in } & (0, T] \times \Omega, \\
u(0, x, z) = \frac{\partial u}{\partial t}(0, x, z) &= 0, & (x, z) \in & (0, L) \times (-H, 0), \\
u(t, 0, z) &= u(t, L, z), & (t, z) \in & (0, T] \times (-H, 0), \\
\frac{\partial u}{\partial z}(t, x, -H) &= 0, & (t, x) \in & (0, T] \times (0, L), \\
\frac{\partial u}{\partial z}(t, x, 0) &= h(t, x), & (t, x) \in & (0, T] \times (0, L).
\end{aligned}
\tag{16}
$$

Similar equations can be written down for other types of source-detector configurations.

4.2 The neural network for learning

With the above computational setup, we can generate the training dataset (5) by solving the wave equation (16) with given source functions. We will describe in detail how the training dataset is generated, including the spatial-temporal discretization of the wave equation (16).
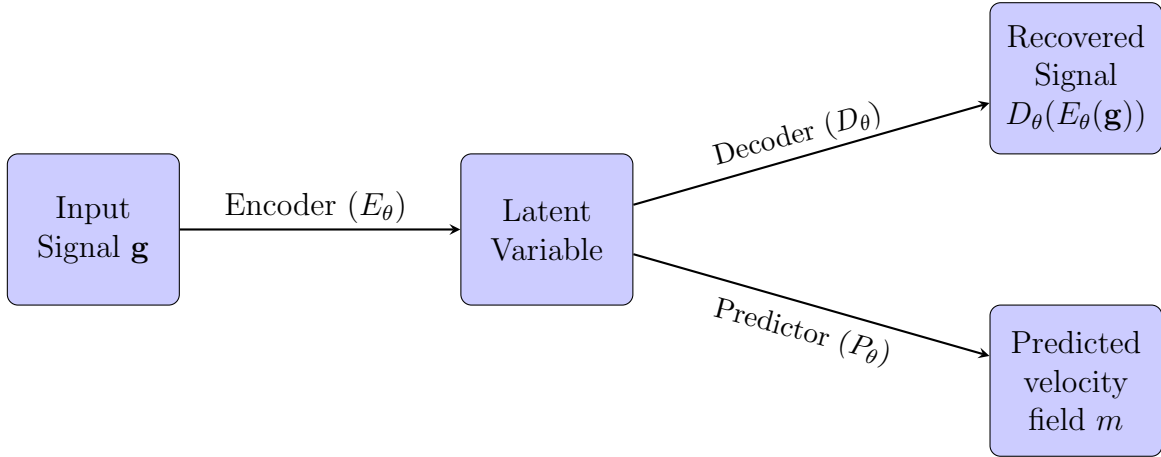

Figure 2: Network flow for learning the approximate inverse operator. Training objective is to select θ such that $\mathbf{g} = D_\theta(E_\theta(\mathbf{g}))$ and $m = P_\theta(E_\theta(\mathbf{g}))$ for every datum pair $(\mathbf{g}, m)$.

We construct an autoencoder network scheme to represent the inverse operator. The learning architecture contains three major substructures: an encoder network E_θ, a decoder network D_θ, and an additional predictor network P_θ; see Figure 2 for an illustration of the network flow. More information on the construction of the encoder, the decoder and the predictor is documented in Appendix C. The encoder-decoder substructure is trained to regenerate the input data, while the predictor reads the latent variable to predict velocity field m. In terms of the input-output data, the network training aims at finding the network parameter θ such that

$$
\mathbf{g}_j = D_\theta(E_\theta(\mathbf{g}_j)) \quad \text{and} \quad m_j = P_\theta(E_\theta(\mathbf{g}_j)), \quad 1 \le j \le N.
\tag{17}
$$

This is done by a minimization algorithm that minimizes a combined ℓ^1-ℓ^2 loss function with the ℓ^1 loss for the encoder-decoder substructure while ℓ^2 for the encoder-predictor substructure. More precisely, we train the network by solving

$$\widehat{\theta} = \arg\min_{\theta \in \Theta} \widetilde{\mathcal{L}}(\theta),$$

with

$$\widetilde{\mathcal{L}}(\theta) := \frac{1}{N} \sum_{j=1}^{N} \|\mathbf{g}_j - D_\theta(E_\theta(\mathbf{g}_j))\|_{\ell^1} + \frac{1}{2N} \sum_{j=1}^{N} \|\boldsymbol{\mu} \circledast (\mathfrak{m}_j - P_\theta(E_\theta(\mathbf{g}_j)))\|_{\mathbb{M}}^2.$$

(18)

While the ℓ^1 loss for the encoder-decoder substructure is standard in the learning literature, the second part of the loss function is simply what we introduced in (6). Once the training is performed, the approximated inverse is taken as

$$\mathbf{f}_{\widehat{\theta}}^{-1} := P_{\widehat{\theta}} \circ E_{\widehat{\theta}}.$$

Let us emphasize that the main motivation for us to adopt this autoencoder framework, instead of directly training a network for $\mathbf{f}_{\widehat{\theta}}^{-1}$, is to take advantage of the commonly observed capability of autoencoders to identify lower dimension features from high-dimensional input data. That is, very often, one can train the autoencoder such that the latent variable $E_\theta(\mathbf{g})$ contains most of useful information in $\mathbf{g}$ but has much lower dimension than $\mathbf{g}$. This lowers the dimension of the predictor network and therefore makes it easier to train the overall network. Moreover, the weighted optimization we used in the encoder-predictor substructure further stabilizes the learning process by focusing on matching the lower-frequency components of the output.

4.3 Learning-assisted FWI inversion

To implement the preconditioned FWI reconstruction method, that is, the solution to the least-squares optimization problem (8), we tested two different algorithms.

Quasi-Newton method with adjoint state. We implemented a quasi-Newton method based on the BFGS gradient update rule [26] for the numerical reconstruction. This BFGS optimization algorithm itself is standard, so we will not describe it in details here. The algorithm requires the gradient of the objective function $\Phi(m)$ defined in (9). We evaluate the gradient with a standard adjoint state method. The procedure is documented in Algorithm 1 of Appendix A. The main complication that the learning stage brings into the adjoint state calculation is that we will need the transpose of the gradient of the neural network with respect to its input. This imposes restrictive accuracy requirements on the training of the neural network in the sense that we need the network to learn not only the map from measurement to the velocity field but also the derivative of the operator.

14

Neumann series method. The Neumann series method based on (14) is more training friendly since it does not require the adjoint operator of the learned approximate inverse $\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}$. We implemented a J-term truncated Neumann series approximation

$$\widehat{m} = \sum_{j=0}^{J-1} K^j \left(\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}(\mathbf{g}^\delta)\right). \tag{19}$$

The computational procedure is summarized in Algorithm 2 of Appendix B.

5 Numerical experiments

We now present some numerical simulations to illustrate some of the main characters of the proposed framework of coupling deep learning with model-based FWI reconstruction. We fix the computational domain to be $\Omega = [0,1] \times [-1,0]$, that is, $L = H = 1$. In this proof-of-concept study, we use acoustic source functions that can generate data at all frequencies. We leave it as future work to consider the situation where low-frequency wavefield data are impossible to measure, in applications such as seismic imaging.

5.1 Velocity feature models

In this work, we consider two different feature models for the output velocity field of the neural network.

Generalized Fourier feature model. In the first model, we represent $m(\mathbf{x})$ as linear combinations of the Laplace-Neumann eigenfunctions on the computational domain Ω. To be precise, let $(\lambda_{\mathbf{k}}, \varphi_{\mathbf{k}})$ $(\mathbf{k} = (k_x, k_z) \in \mathbb{N}_0 \times \mathbb{N}_0)$ be the eigenpair of the eigenvalue problem:

$$-\Delta\varphi = \lambda\varphi, \quad \text{in }\ \Omega, \qquad \mathbf{n} \cdot \nabla\varphi = 0, \quad \text{on }\ \partial\Omega.$$

where $\mathbf{n}(\mathbf{x})$ is the unit outward normal vector of the domain boundary at $\mathbf{x} \in \partial\Omega$. Then $\lambda_{\mathbf{k}} = (k_x\pi)^2 + (k_z\pi)^2$, and

$$\varphi_{\mathbf{k}}(x,z) = \cos(k_x\pi x)\cos(k_z\pi z).$$

In our numerical simulations, we take

$$m(\mathbf{x}) = \sum_{k_x, k_z = 0}^{M} \mathfrak{m}(\mathbf{k})\ \varphi_{\mathbf{k}}(x,z), \tag{20}$$

for some given M. The generation of the random coefficients $\mathfrak{m}(\mathbf{k})$ will be described in detail in the next section.

Gaussian mixture model. The second feature model we take is the Gaussian mixture model. More precisely, we represent $m(\mathbf{x})$ as a superposition of Gaussian functions:

$$m(\mathbf{x}) = m_0 + \sum_{k=1}^{M} c_k e^{-\frac{1}{2}(\mathbf{x}-\mathbf{x}_0^k)^{\mathfrak{T}} \Sigma_k^{-1}(\mathbf{x}-\mathbf{x}_0^k)}. \tag{21}$$

With a small number of highly localized Gaussians, successful reconstruction of such a model could provide inside on source locating problems in seismic applications [14]. This is the main motivation for us to consider this model.

5.2   Learning dataset generation

To generate training data, we generate a set of velocity fields and then solve the wave equation model (16) with source functions $\{h_s\}_{s=1}^{N_s}$ to get the corresponding wave field data at the detectors.

Generating velocity fields. We first construct a set of N random velocity fields $\{m_j\}_{j=1}^{N}$ using the representation (20) or (21). We do this by randomly choosing the coefficients $\{\mathfrak{m}(\mathbf{k})\}_{\mathbf{k}\in\mathbb{N}_0\times\mathbb{N}_0}$ from the uniform distribution $\mathcal{U}[-0.5, 0.5]$ when considering the model (20) and the coefficients c_k from $\mathcal{U}[0, 5]$, $\mathbf{x}_0^k$ from $\mathcal{U}(-H, 0) \times \mathcal{U}(0, L)$, $(\Sigma_k)_{ij}$ from $\mathcal{U}[0, 0.2] + 0.1$ and $m_0 = 10$ when using the model (21). To mimic frequency content of realistic velocity fields, we force the coefficient $\mathfrak{m}(\mathbf{k})$ in the random Fourier model (20) to decay asymptotically as

$$\mathfrak{m}(\mathbf{k}) \sim \mathfrak{m}(\mathbf{k})[(k_x + 1)(k_z + 1)]^{-\alpha}, \quad \text{for large } |\mathbf{k}| = \sqrt{k_x^2 + k_z^2} \tag{22}$$

with $\alpha \geq 0$ given in the concrete examples later.

To make sure that the velocity fields we generated are physically meaningful, we rescale them so that the velocity lives in a range $[\underline{m}, \overline{m}]$ ($0 < \underline{m} < \overline{m} < +\infty$). The linear rescaling is done through the operation

$$m(\mathbf{x}) \leftarrow \frac{\overline{m} - \underline{m}}{m^* - m_*} m(\mathbf{x}) + \frac{\underline{m}m^* - \overline{m}m_*}{m^* - m_*}, \tag{23}$$

where $m^* := \max_{\mathbf{x}} m(\mathbf{x})$ and $m_* := \min_{\mathbf{x}} m(\mathbf{x})$.

In Figure 3 we show some typical samples of the velocity field generated from the aforementioned process. The top panel of Figure 3 shows the surface plots of 4 different randomly generated velocity fields using the model (20) with $M = 4$. The bottom panel presents the surface plots of 4 random realizations of the velocity field given by the model (21) with $M = 2$. Random noise at different levels will be added to the sampled velocity fields to study the generalization of the learning scheme we have. The exact level of noise will be given later in concrete examples.
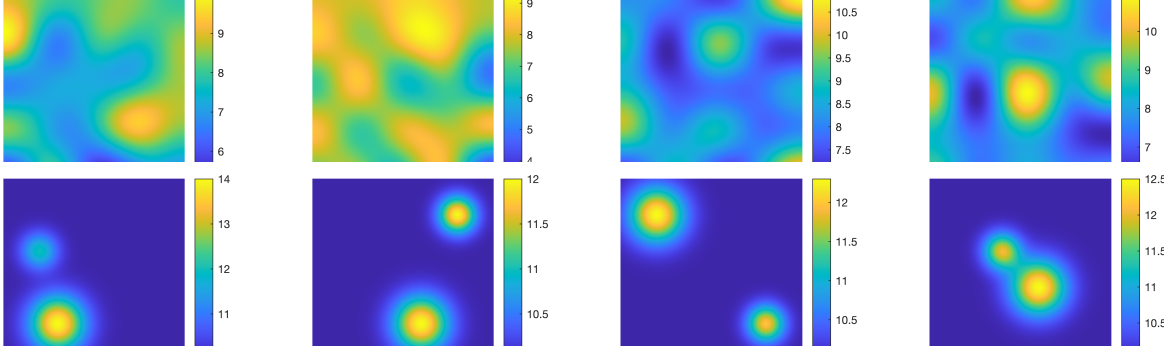
Figure 3: Random samples of the velocity field for training of the neural networks. Top row: velocity fields generated from (21) with $M = 4$; bottom row: velocity fields generated from (20) with $M = 2$.

L	H	K	Δt	t_0	j	T
1	1	50	0.0005	0	20	0.5

Table 1: Values of parameters in the spatial and temporal discretization of the wave equation and the time node of the recorded wave signal.

Finite difference scheme for the wave equation. We use the time-domain stagger-grid finite difference scheme that adopts a second-order in both the time and the spatial directions to solve the wave equation (16). Precisely, the discretization is performed with elements over the Cartesian grids formed by $(x_k, z_l) = (k\Delta x, l\Delta z), k, l = 0, 1, ..., K$ with $\Delta x = L/K$ and $\Delta z = H/K$. The receivers are equally placed at the bottom surface, coinciding with the grid points, as documented in the right panel of Figure 1, namely, there are $K + 1$ receivers for each velocity model. We then record the wave signal starting at time t_0 and take another shot every $j\Delta t$ until the termination time T, here, j is a positive integer and Δt is the uniform time step size for the forward wave solver. As an example for illustration, we take

$$h(t, x) = e^{\frac{-(x-0.6)^2}{0.01}} - e^{\frac{-(x-0.3)^2}{0.01}} \tag{24}$$

to be the top source in (16) and present the recorded time series wave signals in Figure 4. Table 1 summarizes the parameters we used to generate these wavefield signals.

Figure 4, from the left panel to the right panel, shows the time series wave signals at the bottom surface generated from the velocity model satisfying (21) with $M = 2$, and the velocity model satisfying (20) with $M = 4$, respectively; from the top panel to the bottom panel are the wave signals without noise, with 10% multiplication Gaussian noise, and with 10% additive Gaussian noise, respectively.

Last, we note that to obtain a reliable learning dataset, one needs to guarantee the stability of the time integrator when solving (16). Recall that the second order time-domain
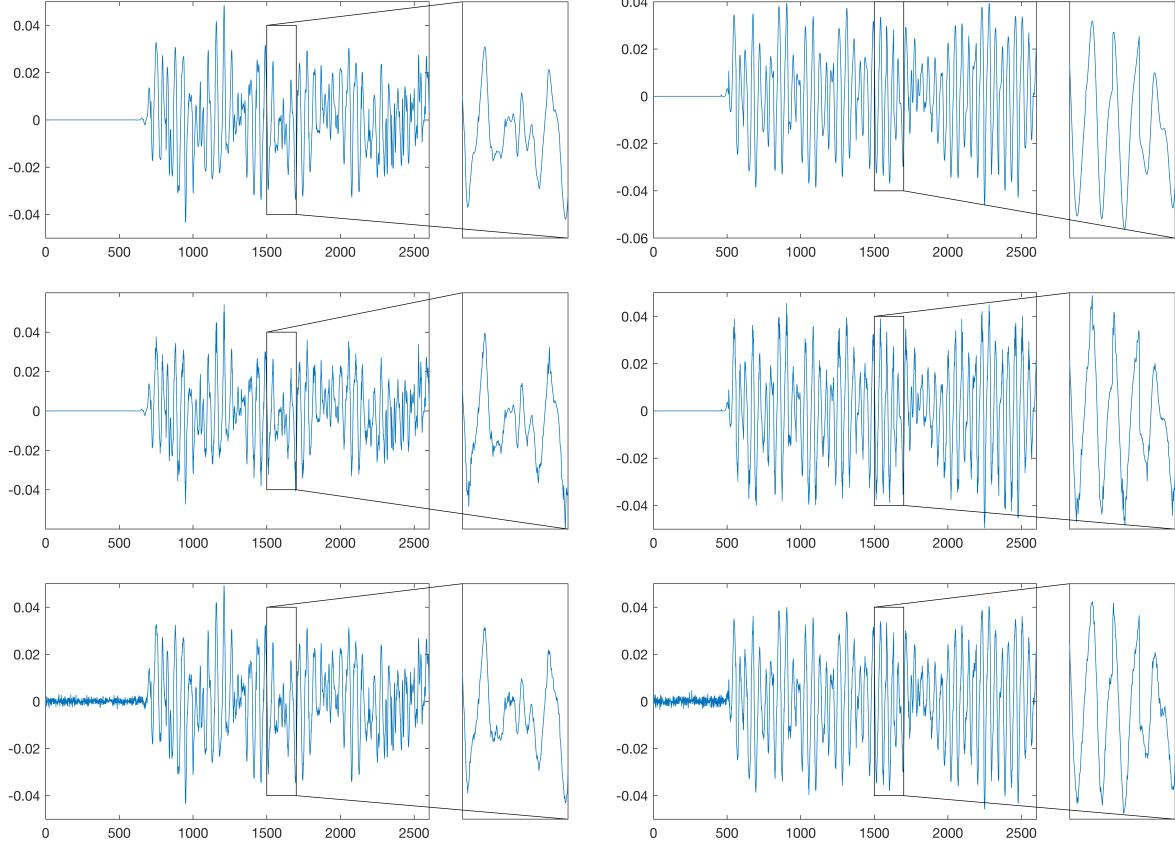
Figure 4: The left panel presents time series wave signals at the bottom surface generated from a velocity model satisfying (21) with $M = 4$, while the right panel shows time series wave signals at the bottom surface generated from a velocity model constructed by (20) with $M = 2$. From the top to the bottom are time series wave signals without noise, with 10% multiplication Gaussian noise and with 10% additive Gaussian noise, respectively.

stagger-grid finite difference forward wave solver is stable under the following CFL condition

$$\Delta t \leq \frac{\min\{\Delta x, \Delta z\}}{\sqrt{2}\max_{\mathbf{x}}\{m(\mathbf{x})\}}. \tag{25}$$

To guarantee the stability of the forward solver for all velocity samples, we force

$$\Delta t = \Delta t^* < \frac{\min\{\Delta x, \Delta z\}}{\sqrt{2}\max_{\mathbf{x}}\{\overline{m}(\mathbf{x})\}},$$

where $\overline{m}$ is used in the scaling (23), for the data generation of the offline training stage. In this work, we set $\Delta t^* = 0.0005$ as shown in Table 1 based on our setting.

5.3   Training and testing performance

We now present a systematic numerical exploration on the training and testing performance of the offline training stage. Given that the training and application of the Gaussian mixture

velocity model (21) with a small amount of Gaussians functions is extremely successful (due to the smallness of the parameter space) according to our numerical experience, we will focus on the training of the generalized Fourier velocity model (20).

Training dataset size. We first emphasize that the training results we show in this section are obtained on a very small dataset in the following sense. The number of data points in the artificial dataset $\{\mathbf{g}_j, m_j\}_{j=1}^N$ is small with $N = 10^6$. Moreover, for each m_j, we collect the wavefield from $N_s = 3$ illumination sources and $N_d = 51$ detectors. Those source-detector pairs are a subset of the source-detector pairs for the dataset we used in the reconstruction step. Moreover, at each detector, we use only data at 51 time steps out of the 1000 time steps in the numerical solutions. This small dataset is used so that we can handle the computational cost of the training process with our limited computing resources. It is also intentionally done to demonstrate that one can train reasonable approximate inverse with a significantly smaller dataset if one is willing to sacrifice a little of the training accuracy.

Training-testing dataset split. We perform a standard training-validation cycle on the neural network approximate inverse. Before the training process starts, we randomly split the artificial dataset of $N = 10^6$ data points into a training dataset and a testing dataset. The training dataset takes 80% of the original dataset, while the test dataset takes the rest 20% of the data points. The training dataset and the validation dataset have no intersection, namely, no data points in the validation set are present in the training dataset.

5.3.1 Random Fourier velocity model: case of non-decaying coefficients

We start with the most challenging scenario where we train the neural network to approximate the inverse operator for the velocity model (20) with randomly generated Fourier coefficients without any decay requirement on the coefficients, that is, we set the decay rate $\alpha = 0$ in (22). This is an extremely challenging case because the effective parameter space of this class of velocity models grows exponentially with respect to the number of Fourier models we have in the model. Ideally, one would need an exponentially large training dataset in order to have reasonable training results. However, due to the smooth property of the map $\mathbf{f} : m \mapsto \mathbf{g}$, we demonstrate below that with a relatively small dataset, and a very limited number of source detector pairs and time shots, our training result is fairly encouraging.

In Figure 5, we show three randomly selected velocity fields (m) from the testing dataset, the corresponding neural network predictions $(\widetilde{m} = \widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}(\mathbf{f}(m)))$, and the error in the prediction $(m - \widetilde{m})$. The largest number of Fourier modes allowed in these learning processes is 10, meaning that $0 \leq k_x, k_z \leq 9$ in the velocity model (20). The training output is a 10×10 matrix containing the content of $\mathfrak{m}(\mathbf{k})$ in (20). The output space is therefore 100-dimensional. A naive visual inspection of the results in Figure 5 shows that the training process is quite successful as the testing errors seem to be pretty reasonable, especially given that our training dataset is fairly small (0.8×10^6 data points to be precise). While it is expected that when the number of Fourier modes allowed in the velocity model is very
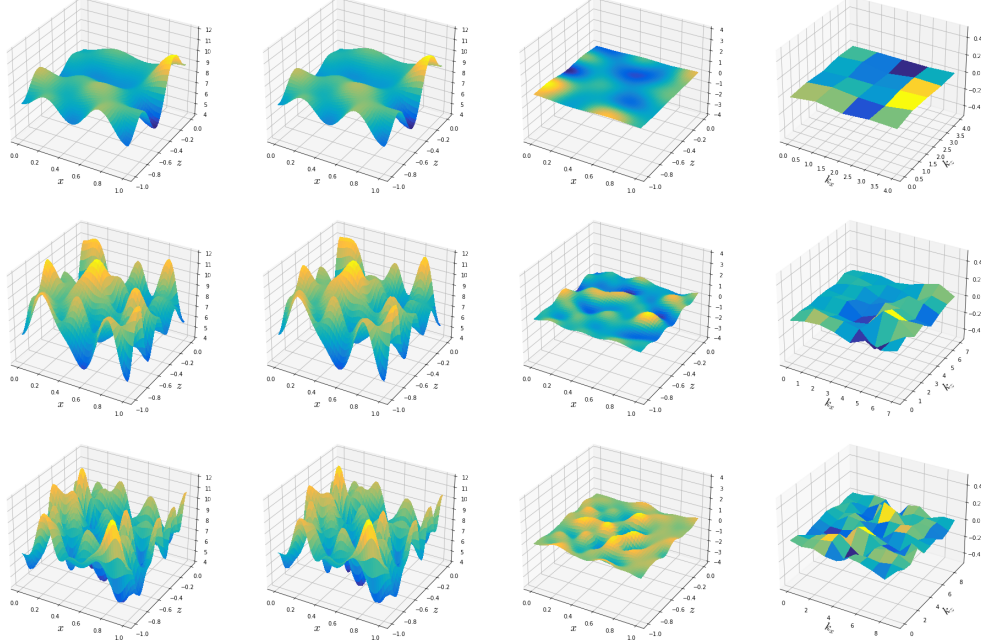
19

Figure 5: Three randomly selected velocity fields from the testing dataset: 5×5 coefficients Fourier model, 8×8 coefficients Fourier model, 10×10 coefficients Fourier model. All cases have decay rate $\alpha = 0$ (column 1), the corresponding predictions by the trained neural network (column 2), the error of the prediction (column 3), and the error in the neural network prediction $(\widetilde{m}(\mathbf{x}))$ in the Fourier domain $(\mathfrak{m}(\mathbf{k}) - \widetilde{\mathfrak{m}}(\mathbf{k}))$ (column 4).

large, the validation error will be sufficiently large if we keep the training sample size, we do observe that validation error is quite small in general for cases when less than 10×10 Fourier modes are pursued in the learning process. Increasing computational power would certainly improve training quality.

Let us remark that our training results indeed show that we have better accuracy in learning the low-frequency components of the inverse operator as we discussed in the previous sections of the work. In the right column of Figure 5, we provide the Fourier coefficients of the errors in the network prediction. In all velocity fields, we see clearly larger errors in the higher-frequency components of the network velocity recovery. This is a universal phenomenon that we observed over the testing dataset.

To dive a little more into the training quality and the optimization landscape after applying our neural network preconditioner, we offer in Figure 6 the training-validation loss curves for a typical learning experiment. We observe very similar curves for training and validating with the velocity model (20) with different total numbers of Fourier modes. We measure the training accuracy quantitatively with the size of the operator $\mathcal{I} - \widehat{\mathbf{f}}_{\widehat{\theta}}^{-1} \circ \mathbf{f}$. More precisely, we evaluate the three main quantities for a data point $(\mathbf{g}, m)$ in the testing dataset:

(i) The error in the network prediction of Fourier modes of m:

$$\Delta \mathfrak{m}(\mathbf{k}) := \mathfrak{m}(\mathbf{k}) - \mathbf{f}_{\widehat{\theta}}^{-1} \circ \mathbf{f}(m) \,.$$
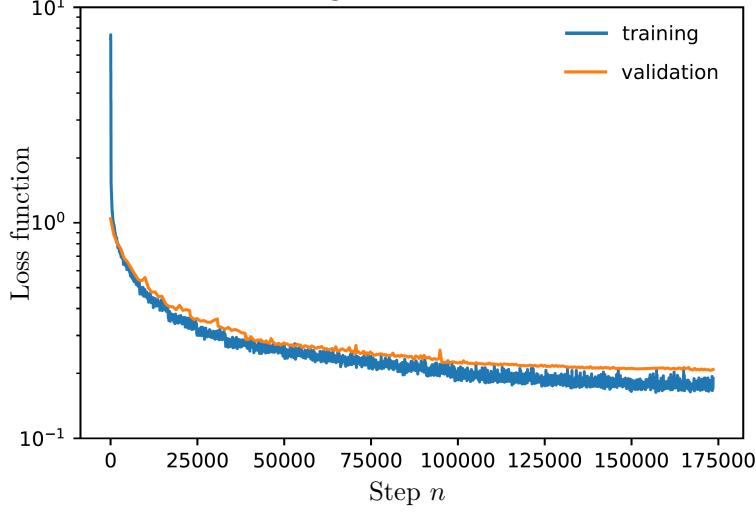
Figure 6: Training and validation loss curves for a typical learning experiment. Very similar curves are observed for each of the learning experiments we performed.

(ii) The landscape of the classical functional $\Psi(m)$ evaluated along a line in the direction of a given Fourier mode of m, $\varphi_{\mathbf{k}}$, passing through two different points $m = \mathbf{f}^{-1}(\mathbf{g})$ and $m_{\text{net}} = \widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}(\mathbf{g})$:

$$\Psi_{\mathcal{O}}(h; \mathbf{k}) := \|\mathbf{g} - \mathbf{f}(m_0 + h\varphi_{\mathbf{k}})\|_{[L^2((0,T]\times\Gamma)]^{N_s}}^2, \quad m_0 = \mathcal{O}(m), \quad \mathcal{O} \in \{\mathcal{I}, \widehat{\mathbf{f}}_{\widehat{\theta}}^{-1} \circ \mathbf{f}\}.$$

(iii) The landscape under our preconditioner, the new mismatch function $\Phi(m)$ evaluated as in (ii):

$$\Phi_{\mathcal{O}}(h; \mathbf{k}) := \|\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}(\mathbf{g}) - \widehat{\mathbf{f}}_{\widehat{\theta}}^{-1} \circ \mathbf{f}(m_0 + h\varphi_{\mathbf{k}})\|_{L^2(\Omega)}^2, \quad m_0 = \mathcal{O}(m), \quad \mathcal{O} \in \{\mathcal{I}, \widehat{\mathbf{f}}_{\widehat{\theta}}^{-1} \circ \mathbf{f}\}.$$

When a perfect learning is performed, we would have $\Delta\mathfrak{m}(\mathbf{k}) = \mathfrak{o}$, $\Psi_{\mathcal{I}}(h; \mathbf{k}) = \Psi_{\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}\circ\mathbf{f}}(h; \mathbf{k})$, and $\Phi_{\mathcal{I}}(h; \mathbf{k}) = \Phi_{\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}\circ\mathbf{f}}(h; \mathbf{k})$ for any $(\mathbf{g}, m)$ in the training dataset, and $\Delta\mathfrak{m}(\mathbf{k})$ small, $\Psi_{\mathcal{I}}(h; \mathbf{k}) \approx \Psi_{\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}\circ\mathbf{f}}(h; \mathbf{k})$, and $\Phi_{\mathcal{I}}(h; \mathbf{k}) \approx \Phi_{\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}\circ\mathbf{f}}(h; \mathbf{k})$ for any $(\mathbf{g}, m)$ in the testing dataset.

In Figure 7, we show plots of $\Delta\mathfrak{m}(\mathbf{k})$ (left column), $\Psi_{\mathcal{I}}(h; \mathbf{k})$ (red line) and $\Psi_{\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}\circ\mathbf{f}}(\mathbf{k})$ (blue line) (middle column), and $\Phi_{\mathcal{I}}(h; \mathbf{k})$ (red line) and $\Phi_{\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}\circ\mathbf{f}}(\mathbf{k})$ (blue line) (right column), for four randomly selected $(\mathbf{g}, m)$ pairs in the testing dataset. Shown are results for $\mathbf{k} = (2, 3)$ and $\mathbf{k} = (1, 1)$. Very similar behavior are observed along other coordinates $\varphi_{\mathbf{k}}$.

The plots in Figure 7 provide a quantitative description of the accuracy of the trained network. They clearly indicates that the trained $\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}$ is indeed a good approximation to $\mathbf{f}^{-1}$. Moreover, a comparison of the second column and the third column gives the impression that along with the coordinates we plotted, the new objective functional Φ in (9) has a much better landscape than the classical Ψ in (4). This is what we observed in other coordinates that are not shown here as well. Therefore, the trained neural network $\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}$ can work as a nonlinear preconditioner to improve convexify of the optimization landscape. Moreover, the plots provided a good indication that the trained network is fairly generalizable in the
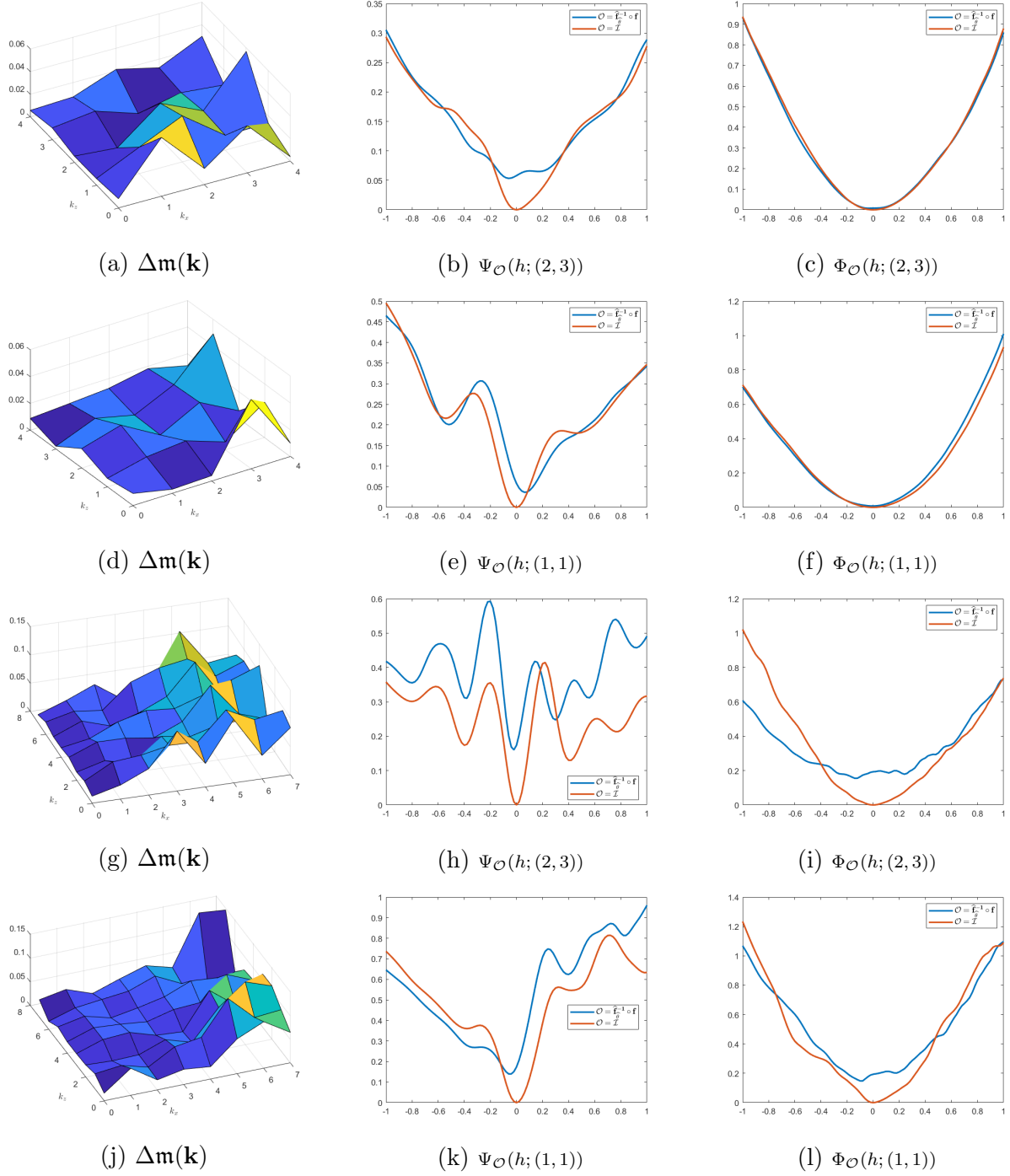
21

(a) $\Delta\mathfrak{m}(\mathbf{k})$ (b) $\Psi_{\mathcal{O}}(h;(2,3))$ (c) $\Phi_{\mathcal{O}}(h;(2,3))$

(d) $\Delta\mathfrak{m}(\mathbf{k})$ (e) $\Psi_{\mathcal{O}}(h;(1,1))$ (f) $\Phi_{\mathcal{O}}(h;(1,1))$

(g) $\Delta\mathfrak{m}(\mathbf{k})$ (h) $\Psi_{\mathcal{O}}(h;(2,3))$ (i) $\Phi_{\mathcal{O}}(h;(2,3))$

(j) $\Delta\mathfrak{m}(\mathbf{k})$ (k) $\Psi_{\mathcal{O}}(h;(1,1))$ (l) $\Phi_{\mathcal{O}}(h;(1,1))$

Figure 7: Plots of $\Delta\mathfrak{m}(\mathbf{k})$ (first column), $\Psi_{\mathcal{O}}(h;\mathbf{k})$ (second column), and $\Phi_{\mathcal{O}}(h;\mathbf{k})$ (third column) for four different $(\mathbf{g}, m)$ pairs in the testing dataset. The velocity model for rows 1-2 has $M = 4$ and that for the plots in rows 3-4 has $M = 7$.

following sense. The Fourier coefficients (including $\mathfrak{m}_{(2,3)}$ and $\mathfrak{m}_{(1,1)}$ shown in the plots) in the training dataset are all randomly drawn in the interval $[-0.5, 0.5]$. Here in the plots, we consider the coefficient values in the range $[-1, 1]$. The agreement of the red and blue

22

lines outside of the training value range $[-0.5, 0.5]$, that is, in the range $[-1, -0.5) \cup (0.5, 1]$, suggests that the trained neural network can be used in a region of coefficient values that is far larger than its training domain.

5.3.2 Random Fourier velocity model: case of decaying coefficients

While training the neural network for an approximate inverse $\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}$ on a large space of velocity field is extremely useful for generalization purpose, it also poses great challenges when the number of Fourier modes included in the model gets very large. Not only will we need an exponentially larger training dataset, but also the training process takes exponentially growing computational power. This is what we observed in our numerical experiments. In this section, we show some training-validation results for the velocity model (20) with decaying Fourier coefficients following the pattern we imposed in (22). We present results from two different cases: the slow decay case with $\alpha = 1/2$ and the fast decay case with $\alpha = 1$.

In Figure 8, we show typical validation results on five randomly selected velocity profiles in the testing dataset. The top two rows are the results for the training of velocity model with $\alpha = 0$, the third row is the case of $\alpha = 1/2$ while the bottom two rows are for the case of $\alpha = 1$. In both cases, the training is successful as can be seen from the relatively small errors in the predictions. Plots of the functionals $\Psi_{\mathcal{I}}$ and $\Psi_{\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1} \circ \mathbf{f}}$ show similar patterns as those in Figure 7. We omit those to space. Moreover, prediction errors in the Fourier domain display very similar behavior as observed in the previous subsection: the error is higher for high-frequency components and lower for low-frequency components.

To study the generalization capability of the learned network, we validate the learning with on dataset generated from a different velocity model, that is consider the case where training and testing data samples are from different classes. In Figure 9, we train a neural network to recover the first 10×10 Fourier coefficient of the velocity field and validate the trained neural network on a dataset generated from velocity models that contains 20×20 random Fourier modes. The decay rate in this particular case is $\alpha = 1$ but similar results are observed for $\alpha = 1/2$ as well. The validation results demonstrate that the trained network is reasonably generalizable in the setting that we considered.

5.3.3 Mesh-based velocity model

In the last training-validation numerical experiment, we demonstrate that the phenomena observed in the previous subsections are not particularly due to the Fourier parameterization of the velocity field in (20) that we used. Indeed, the results are more related to our method of training. Here we perform the same type of training on a neural network whose output is the velocity field represented on a 51×51 uniform mesh on the domain Ω. The output space is therefore much larger compared to the training in the case of the random Fourier velocity model. However, the training result, after projecting into the Fourier space, has almost identical properties as what we observed in the random Fourier model. In Figure 10, we show the out-of-domain validation result for the training. The velocity fields that generated
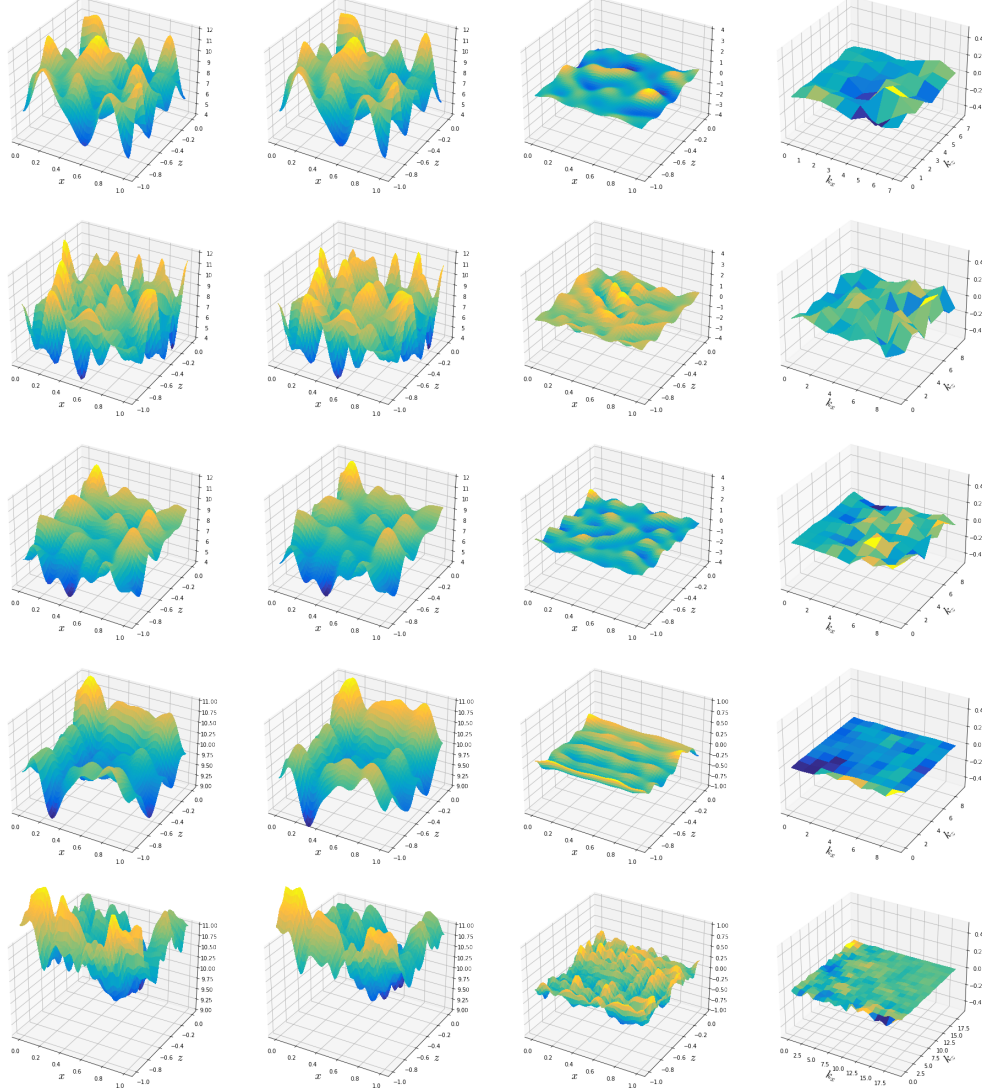
Figure 8: Validation results on four typical velocity fields in the testing dataset. Row 1 is the results for 8 Fourier velocity model with $\alpha = 0$, Row 2 is the results for 10 Fourier velocity model with $\alpha = 0$, row 3 is the results for 10 Fourier velocity model with $\alpha = 1/2$, row 4 is the results for 10 Fourier velocity model with $\alpha = 1$ while row 5 is are results for 20 Fourier velocity model with $\alpha = 1$. Show from left to right are: the true velocity field, neural network prediction, the error of the prediction, and the error of the prediction in the Fourier domain.

the training dataset have 10×10 Fourier modes while the velocity fields in the validation dataset have 20×20 Fourier modes (but represented on a 51×51 uniform mesh), both generated with $\alpha = 1$. The relatively small validation errors again indicate that the training is fairly successful and reasonably generalizable. The computational cost, in this case, is much larger than those in the previous subsections since the neural network has a larger size due to the increased size of the network output.
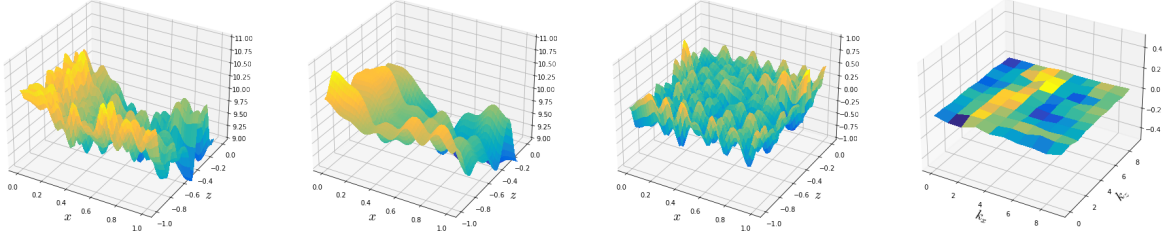
Figure 9: Instance of validation of learning results on a different class of velocity models for the case of $\alpha = 1$. Shown from left to right are: true velocity field, the neural network prediction, the error in the prediction, and the error in the Fourier domain.
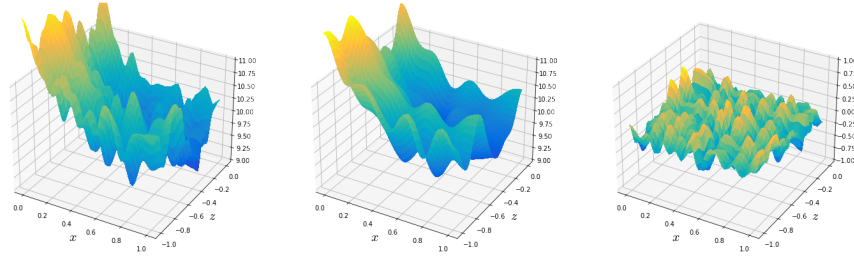

Figure 10: Out-of-domain validation of a training result with mesh-based velocity representation. Shown from left to right are: the true velocity field, the network prediction, and the error in the prediction.

5.4 Learning-assisted FWI reconstruction

In this section, we present inversion results for some simulated datasets to verify the efficiency and stability of the proposed coupling method. All simulations on the inversion stage are conducted on a quadcore Intel Core i7 with 16 GB RAM.

5.4.1 Convexity of the new objective function

Lemma 3.2 indicates that if we have relatively accurate training, the new objective function for our coupled reconstruction scheme behaves similarly to the functional $\|m - m_0\|^2_{L^2(\Omega)}$, m_0 being the true solution. Figure 7 provided some evidence of this in the training of the random Fourier model. In the one coefficient case, plots in Figure 7 clearly show that the new objective function is almost convex. We now present some numerical evidence in the case of the Gaussian mixture velocity model. In particular, we are interested in seeking convexity with respect to the location of a Gaussian perturbation. More precisely, the velocity field $m(\mathbf{x})$ is set to be a single Gaussian model with $M = 1$ in (21), that is,

$$m(\mathbf{x}) = m_0 + c_1 e^{-\frac{1}{2}(\mathbf{x}-\mathbf{x}_0^1)^{\mathfrak{T}} \Sigma_1^{-1}(\mathbf{x}-\mathbf{x}_0^1)}, \quad \mathbf{x}_0^1 = (x_0^1, z_0^1), \quad \Sigma_1 = \begin{pmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_1^2 \end{pmatrix}.$$

where the background velocity m_0, the amplitude c_1, and the variance σ_1 are fixed to be $(m_0, c_1, \sigma_1) = (10, 5, 0.1)$. We then present the objective functions $\Psi(m)$ and $\Phi(m)$ ($\gamma = 0$)

with respect to the location (x_0^1, z_0^1) in Figure 11. The setting of the offline training stage for generating $\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}$ is the same as those in Section 5.2.

Figure 11 presents the landscapes of objective functions $\Psi(m)$ and $\Phi(m)$ ($\gamma = 0$) with fixed (m_0, c_1, σ_1). In particular, we set $(x_0^1, z_0^1) = (0.5, -0.5)$ as the ground true velocity model which generates the wave signal $\mathbf{g}$. From Figure 11, we observe that, (i) the classical objective function $\Psi(m)$ is not a convex function, and its landscape shows that the optimization can be easily trapped into a local minimum if the initial model is not carefully chosen; (ii) the new objective function $\Phi(m)$ ($\gamma = 0$) for the proposed coupling method becomes more convex which is consistent with Lemma 3.2. In addition, we note that when the initial model is close enough to the exact model (located at the convex region of the misfit function), the global minimum is guaranteed and one can also expect a fast convergence. In fact, a good initial model under the setting of the proposed coupling scheme can be easily obtained by adding a small perturbation to $\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}(\mathbf{g})$ as indicated by Neumann series (14).

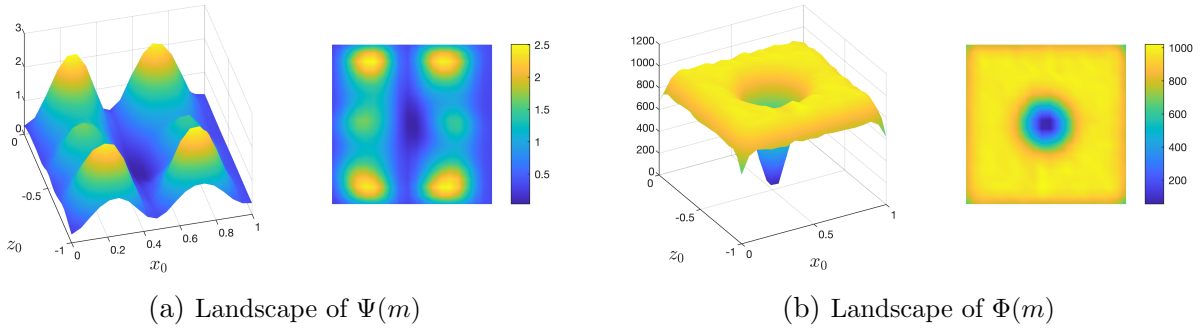(a) Landscape of $\Psi(m)$ (b) Landscape of $\Phi(m)$

Figure 11: The landscape of the classical (left) and new (right) objective functions for the location of a Gaussian perturbation of the velocity field.

5.4.2 Inversion for the velocity model (21) with $M = 2$

The first inversion example was performed to recover the following mixed Gaussian velocity model (21) with $M = 2$ and $m_0 = 10$,

$$m(\mathbf{x}) = 10 + \sum_{k=1}^{2} c_k e^{-\frac{1}{2}(\mathbf{x}-\mathbf{x}_0^k)^T \Sigma_k^{-1}(\mathbf{x}-\mathbf{x}_0^k)}, \quad \mathbf{x}_0^k = (x_0^k, z_0^k), \quad \Sigma_k = \begin{pmatrix} \sigma_k^2 & 0 \\ 0 & \sigma_k^2 \end{pmatrix}. \qquad (26)$$

We use the same offline training settings as those for the mixed Gaussian wave signal generation in Section 5.2 to construct $\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}$ for the online inversion stage. However, to generate versatile wave signals at the bottom surface to recover the features $\{c_1, c_2, \sigma_1, \sigma_2, x_0^1, x_0^2, z_0^1, z_0^2\}$, we enforce three different top sources $h_i(x), i = 1, 2, 3$ with

$$h_1(x) = e^{-\frac{(x-0.8)^2}{0.01}} - e^{-\frac{(x-0.2)^2}{0.01}}, \quad h_2(x) = e^{-\frac{(x-0.4)^2}{0.01}} - e^{-\frac{(x-0.7)^2}{0.01}},$$

and

$$h_3(x) = e^{-\frac{(x-0.6)^2}{0.01}} - e^{-\frac{(x-0.3)^2}{0.01}},$$

	no noise			10% multiplicative noise			10% additive noise		
J	L^2 error	L^∞ error	CPU time(s)	L^2 error	L^∞ error	CPU time(s)	L^2 error	L^∞ error	CPU time(s)
1	1.48e-01	1.12e-00	0	2.54e-01	1.98e-00	0	2.60e-01	1.60e-00	0
20	1.10e-04	1.11e-03	6.83	1.71e-03	1.39e-02	6.71	5.79e-03	4.27e-02	6.89
40	3.21e-06	2.60e-05	13.80	4.84e-05	5.48e-04	13.60	5.88e-03	4.28e-02	13.38
60	3.09e-06	2.41e-05	20.34	5.78e-06	4.38e-05	20.97	5.89e-03	4.28e-02	20.51
80	2.86e-06	2.66e-05	27.74	4.63e-06	4.36e-05	28.52	5.89e-03	4.28e-02	27.56

Table 2: L^2/L^∞ reconstruction errors, and the CPU time for the inversion stage with different J-term truncated Neumann series approximation, as well as different noise level/form for the reconstruction of the mixed Gaussian (26).

rather than one single external top source in Section 5.2.

For the inversion stage, we implement a J-term truncated Neumann series approximation (19) to obtain the reconstructed velocity image. Note that $J = 1$ corresponds to the reconstructed velocity image from the offline training stage. We also add the Gaussian noise with zeros mean and 10% standard derivation to test the stability of the proposed coupling scheme. Figure 12 presents the reconstructed images. Precisely, the first three columns show the surface plots of the exact velocity field, the neural network prediction velocity field from the offline training stage, and the reconstructed velocity field with $J = 20$ from the online inversion stage, while the last column displays the difference between the exact velocity field (first column) and the reconstructed velocity field (third column). From the top row to the bottom row of Figure 12, we present the results from the noise-free wave signal, the wave signal with 10% multiplication Gaussian noise, and the wave signal with 10% additive Gaussian noise, respectively. We see that the online inversion stage improves the accuracy of the reconstructions for all cases. Table 2 lists the L^2/L^∞ errors on the velocity field for the entire computational domain, as well as the CPU time for various implementations with different values of J. There, we note that for the wave signals without noise and with 10% multiplication Gaussian noise, both L^2 and L^∞ reconstruction errors dropped by a factor $\sim 10^4$ within 30 seconds; for the wave signal with 10% additive Gaussian noise, it seems that there is no improvement to add more Neumann terms in (19) once the L^2 error reduces to 5.89×10^{-3} and L^∞ error reduces to 4.28×10^{-2}; for this type of the situation, we can use the reconstruction from adding Neumann terms as an initial guess for a gradient-based optimization scheme to further improve the resolution of the reconstruction, see Section 4.3.

5.4.3 Inversion for the velocity model (20) with $M = 4$

For the second inversion example, we work on reconstructing the features of the following velocity model

$$m(\mathbf{x}) = \sum_{k_x, k_z=0}^{4} \mathfrak{m}(\mathbf{k}) \cos(k_x \pi x) \cos(k_z \pi z), \quad \mathbf{k} = (k_x, k_z) \tag{27}$$
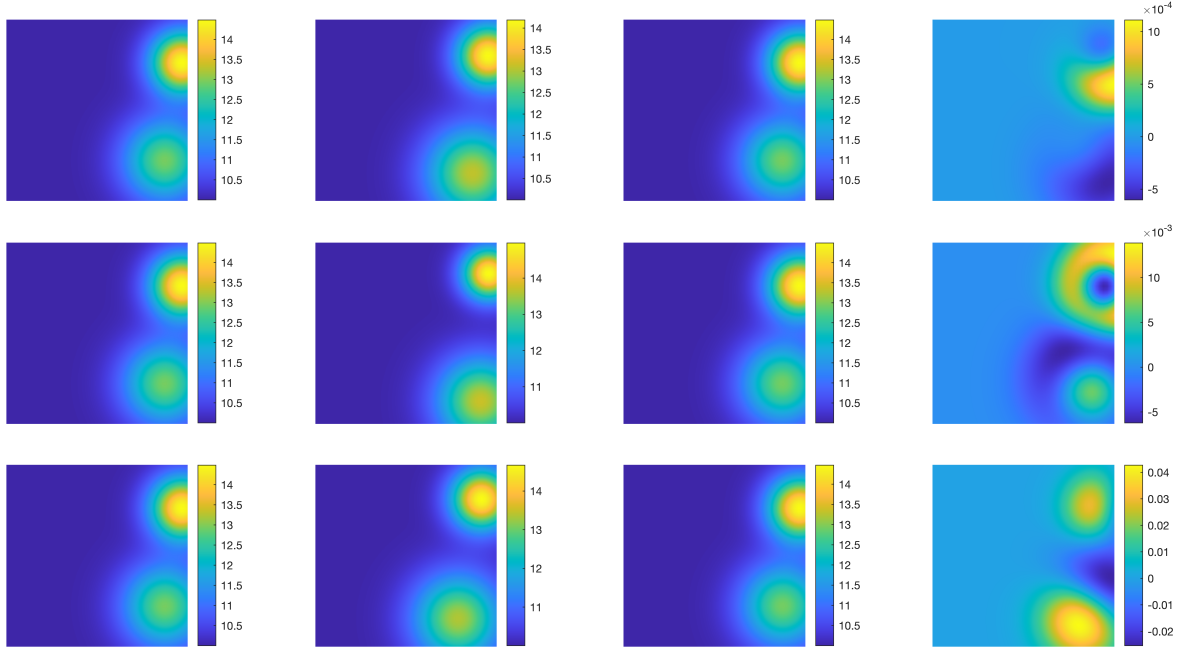
Figure 12: The reconstructed velocity images for the mixed Gaussian (26). From the left to the right are the ground true velocity field, the reconstructed velocity field with $J = 1$, the reconstructed velocity field with $J = 20$, and the difference between the ground true velocity field and the reconstructed velocity field with $J = 20$ (first column - third column). From the top to the bottom are the results from the noise-free wave signal, the wave signal with 10% multiplication Gaussian noise, and the wave signal with 10% additive Gaussian noise.

with $\alpha = 0$ in (22), namely, we don't consider any decay on the coefficients for this example. In addition, we use the same training settings as those in Section 5.2 for the Fourier wave signal generation. But for the external top sources $h_i(x)$, we choose them to be the same as the sources in Section 5.4.2 to generate resourceful training samples for the construction of $\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}$.

For the online inversion stage, we also implement a J-term truncated Neumann series approximation (19) to recover the velocity model. To test the stability of the proposed coupled scheme, as in Section 5.4.2, we add the Gaussian noise with zeros mean and 10% standard derivation to the wave signals. Figure 13 presents the surface plots of the reconstructed velocity images with $J = 20$, as well as the surface plots for the difference between the reconstructed image and the ground true velocity model. The layout of Figure 13 is the same as the one in Figure 12. We observe that the training prediction is stable with respect to the noise, see the second column of Figure 13 and L^2/L^∞ errors when $J = 1$ in Table 3. In addition, we note that the inversion stage can significantly improve the accuracy of the reconstruction. For the data without noise, the errors dropped by a factor $\sim 10^7$ within 30 seconds; even for the data with 10% Gaussian noise, the errors also dropped by a factor $\sim 10^3$ within 30 seconds.
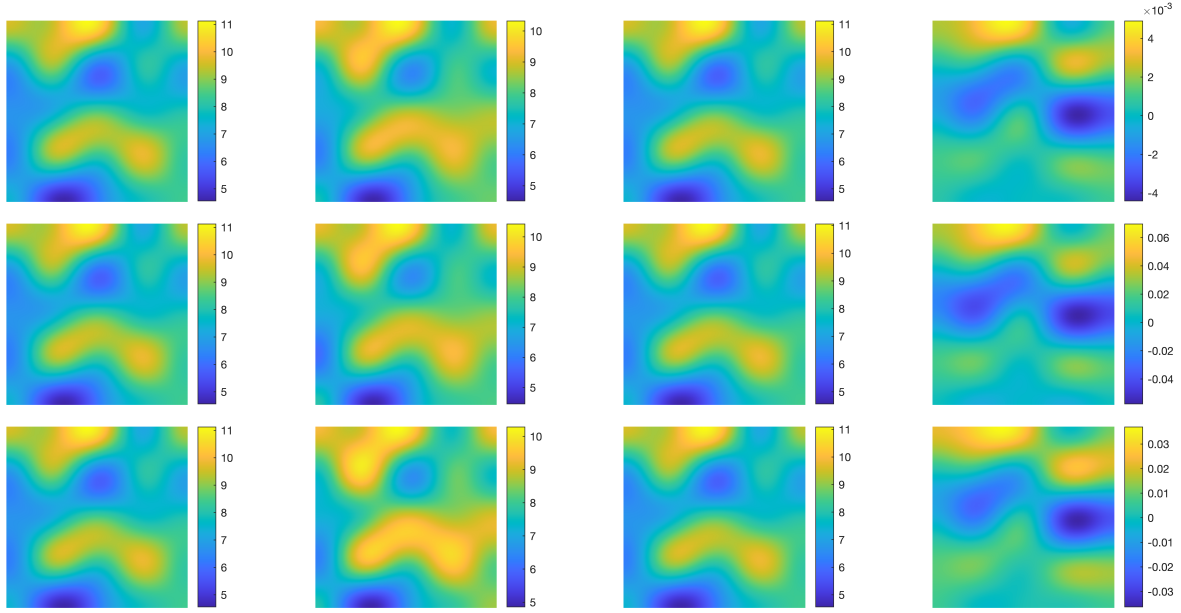
Figure 13: The reconstructed velocity images for the general Fourier type (20) with $M = 4$. From the top to the bottom are for the velocity reconstruction without noise, with 10% multiplication Gaussian noise, with 10% additive Gaussian noise, respectively. While from the left to the right are the ground true velocity field, the reconstructed velocity field from the neural network in the offline training stage, and the reconstructed velocity image with $J = 20$, respectively.

5.4.4 Inversion for the velocity model (20) with $M = 7$

For the third example, we consider a velocity model which contains 8 Fourier modes in each direction, namely,

$$m(x, z) = \sum_{k_x, k_z = 0}^{7} \mathfrak{m}(\mathbf{k}) \cos(k_x \pi x) \cos(k_z \pi z). \tag{28}$$

The spatial and temporal discretization, as well as the rules for data generation, the choice of the top source $h_i(x)$ are the same as the example in Section 5.4.3.

For the inversion stage, we again implement a J-term truncated Neumann series approximation (19) to obtain the reconstructed velocity image. Figure 14 presents the surface plots of the reconstructed velocity images with various values of J in the online inversion stage. Precisely, each row of Figure 14 corresponds to one velocity model; from the left to the right are the ground true velocity field, the reconstructed velocity image with $J = 1$, the reconstructed velocity image with $J = 20$, and the reconstructed velocity image with $J = 50$, respectively. We note that the online inversion stage improves the accuracy of the reconstruction for all cases which verifies the effectiveness of the proposed coupling scheme.

29

	no noise			10% multiplicative noise			10% additive noise		
J	L^2 error	L^∞ error	CPU time(s)	L^2 error	L^∞ error	CPU time(s)	L^2 error	L^∞ error	CPU time(s)
1	1.78e-01	8.46e-01	0	1.79e-00	7.25e-00	0	2.64e-01	1.16e-00	0
20	8.52e-04	4.90e-03	6.08	1.23e-02	6.97e-02	5.81	7.13e-03	3.70e-02	5.87
40	1.49e-05	8.41e-05	11.15	4.93e-03	2.76e-02	11.24	1.69e-03	9.26e-03	11.73
60	2.60e-07	1.53e-06	17.97	3.06e-03	1.71e-02	17.42	6.81e-04	3.77e-03	17.67
80	2.16e-08	1.34e-07	22.74	2.19e-03	1.22e-02	23.25	2.05e-04	1.12e-03	23.13

Table 3: L^2/L^∞ reconstruction errors, and the CPU time for the inversion stage with different J-term truncated Neumann series approximation, as well as different noise level/form for the reconstruction of the Fourier model (27).
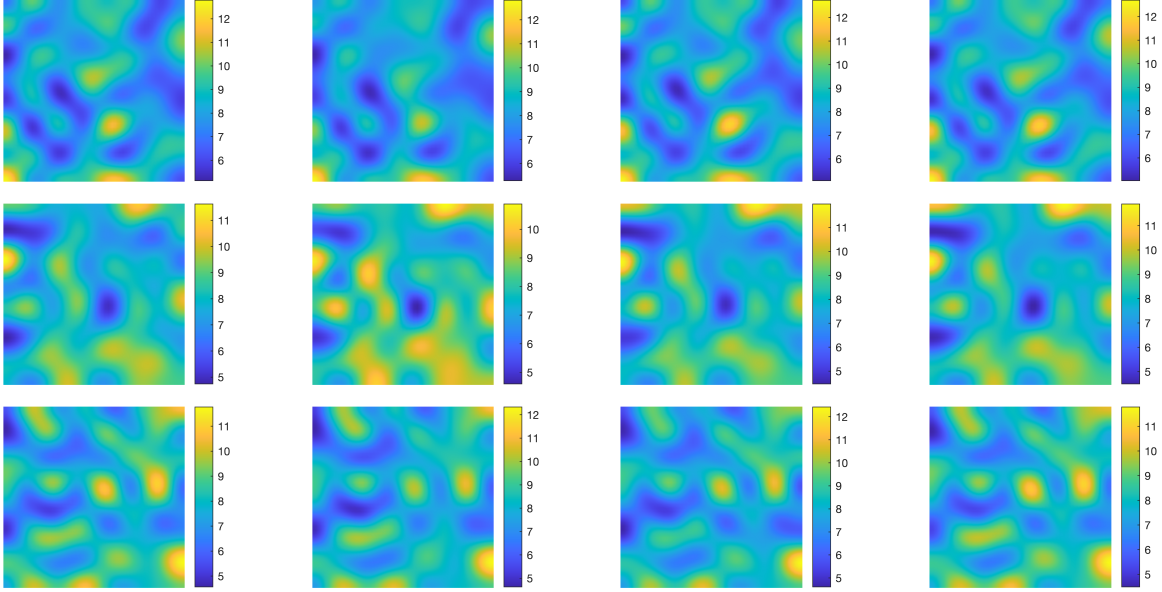

Figure 14: The reconstructed velocity images for the Fourier model (28). Each row corresponds to the reconstruction of one velocity field. From the left to the right are the ground true velocity field, the reconstructed velocity field with $J = 1$, the reconstructed velocity field with $J = 20$, and the reconstructed velocity field with $J = 50$, respectively.

5.4.5 Inversion for the velocity model outside of the training domain

In the last example, we test the proposed coupling scheme on a velocity model which is outside of the training domain. Precisely, the design of the offline training stage is the same as the one in Section 5.4.3, namely, we focus on learning the first 5 Fourier modes along each direction during the training. However, our goal in this example is to reconstruct the following velocity model,

$$m(x, z) = \begin{cases} 8.4, & (x, z) \in [0.22, 0.74] \times [-0.52, -0.5], \\ 7.6, & \text{otherwise,} \end{cases}, \quad (x, z) \in [0, 1] \times [-1, 0], \quad (29)$$

30

which is apparently outside of the training domain containing many high frequency components. To reconstruct (29), we first implement the J-term truncated Neumann series approximation (14) with $J = 20$ to obtain the low frequency part of the velocity model (29), then use it as the initial guess of a quasi-Newton algorithm based on the BFGS gradient update rule to minimize (4). In addition, to recover the high-frequency components of the velocity field, except the 51 receivers at the bottom surface in the training stage, we place another 51 receivers at the top surface when minimizing (4), and enforce 7 different top sources $h_i(x), i = 1, 2 \cdots, 7$ with h_1, h_2, h_3 being the same as the top sources in the training stage, and

$$h_4(x) = e^{-\frac{(x-0.7)^2}{0.01}} - e^{-\frac{(x-0.2)^2}{0.01}}, \quad h_5(x) = e^{-\frac{(x-0.3)^2}{0.01}} - e^{-\frac{(x-0.9)^2}{0.01}},$$

$$h_6(x) = e^{-\frac{(x-0.2)^2}{0.01}} - e^{-\frac{(x-0.5)^2}{0.01}}, \quad h_7(x) = e^{-\frac{(x-0.1)^2}{0.01}} - e^{-\frac{(x-0.6)^2}{0.01}}.$$

Figure 15 presents the surface plots of the reconstructed velocity images with both noise-free data and the data with Gaussian noises. Precisely, the top row shows the reconstructed velocity from noise-free data, the middle row displays the reconstructed velocity from the data with 10% multiplication Gaussian noise, and the bottom row presents the reconstructed velocity from the data with 10% additive Gaussian noise; while from the left to the right columns are the ground true velocity field, the reconstructed velocity image with $J = 1$, the reconstructed velocity image with $J = 20$ (initial guess), and the reconstructed velocity image by minimizing (4), respectively. We note that adding several terms to the Neumann series approximation can lead to a relatively good reconstruction for the low-frequency components of the velocity field (29) for all cases (noise-free data and the data with Gaussian noise) by comparing the reconstruction results in column 2 and column 3. Then solving an extra classical minimization problem as documented in Section 3.4 helps grab the high-frequency components of the velocity field as shown in the last column.

6 Concluding remarks

We presented in this work an offline-online computational strategy for coupling deep learning methods with classical model-based iterative reconstruction schemes for the FWI problem. The main advantage of the coupling lies in two aspects. First, the coupling requires much less rigorous training for the learning part than a purely learning based approach. This makes the learning of the approximate inverse operator much more realistic with limited computational resources. Second, the offline learning can still significantly reduce the online reconstruction with new datasets when used as a nonlinear preconditioner. The numerical simulations we performed demonstrated the feasibility of such a coupled approach.

There are many important issues in the current direction that need to be more rigorously investigated. One particular aspect is to develop a mathematical characterization of the training error in the learning process and study its impact on the reconstruction step. A second aspect is to improve the learning algorithm to learn more features in the inverse operator. As we reasoned in the paper, it is extremely challenging to learn all the details in the inverse operator. However, we believe that one could do much better than the numerical
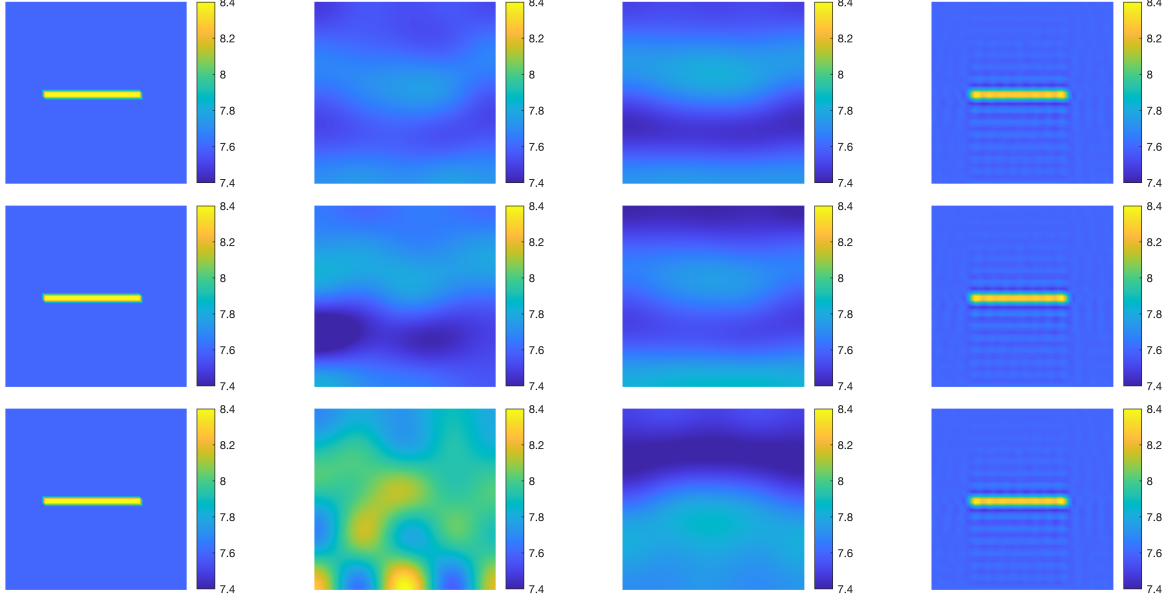
Figure 15: The reconstructed velocity images for the velocity model (29). From the top to the bottom are the reconstruction with noise-free signal, the signal with 10% multiplication Gaussian noise, and the signal with 10% additive Gaussian noise, respectively. From the left to the right are the ground true velocity field, the reconstructed velocity field with $J = 1$, the reconstructed velocity field with $J = 20$, and the reconstructed velocity field by minimizing (4), respectively.

experiments in this paper where we pursue only a very small amount of features in the learning process. Searching for better feature models for the velocity field as well as the time traces of the wavefield is also an important task with the potential to significantly improve the performance of the learning procedure.

Acknowledgments

A   Adjoint state gradient calculation

We summarize here the calculation of the Fréchet derive of the objective function $\Phi(m)$ defined in (9) with respect to the velocity field m.

Following Proposition 3.1, the Fréchet differentiability of the map $\mathbf{f}(m)$ with respect to m is well-established under reasonable assumptions on the smoothness of the domain, the regularity of the incident wave source h and the regularity of the velocity field m. With the assumption we have on the differentiability of the trained network $\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}$, the Fréchet differentiability of $\Phi(m)$ in (9) is ensured.

To simplify the notation, we denote by

$$r(m) := \widehat{f}_{\widehat{\theta}}^{-1}(f(m)) - \widehat{f}_{\widehat{\theta}}^{-1}(g^{\delta}) \tag{30}$$

the data residual and $\Gamma \subset \Omega$ the subset of the boundary of the domain where acoustic data are measured. To be concrete, we take the regularization functional to be the $\mathcal{H}^1$ semi-norm of the unknown m. We also assume that the velocity is known on the boundary of the domain so that the perturbation $\delta m|_{\partial\Omega} = 0$. This assumption simplified the calculations below but is by no means essential.

Taking the derivative of $\Phi(m)$ with respect to m in the direction δm, we have

$$\Phi'(m)[\delta m] = \int_{\Omega} r(m) \big(\widehat{f}_{\widehat{\theta}}^{-1}\big)'(f(m)) \Big[f'(m)[\delta m]\Big] d\mathbf{x} + \gamma \int_{\Omega} \nabla m \cdot \nabla \delta m \, d\mathbf{x}. \tag{31}$$

Let $\big(\widehat{f}_{\widehat{\theta}}^{-1}\big)'^* : L^2(\Omega) \mapsto L^2((0,T] \times \Gamma)$ be the adjoint of the operator $\big(\widehat{f}_{\widehat{\theta}}^{-1}\big)'(f(m))$. Using the assumption that $\delta m|_{\partial\Omega} = 0$, we can then write the above result as

$$\Phi'(m)[\delta m] = \int_0^T \int_\Gamma \big(\widehat{f}_{\widehat{\theta}}^{-1}\big)'^*[r(m)] f'(m)[\delta m] \, dS(\mathbf{x}) dt - \gamma \int_\Omega \Big(\Delta m\Big) \delta m \, d\mathbf{x}. \tag{32}$$

This can be written into the following form with the adjoint operator of $f'(m)$, $f'^* : L^2((0,T] \times \Gamma) \mapsto L^2(\Omega)$:

$$\Phi'(m)[\delta m] = \int_\Omega f'^* \Big[\big(\widehat{f}_{\widehat{\theta}}^{-1}\big)'^*[r(m)]\Big] \delta m \, d\mathbf{x} - \gamma \int_\Omega \Big(\Delta m\Big) \delta m \, d\mathbf{x}. \tag{33}$$

The adjoint operator f'^* can be found in the standard way. We document the calculation for the specific two-dimensional setup we have as follows.

For the wave equation (16), we can formally differentiate u with respect to m to have that u' solves

$$
\begin{aligned}
\frac{1}{m^2}\frac{\partial^2 u'}{\partial^2 t} - \Delta u' &= 2\frac{\delta m}{m^3}\frac{\partial^2 u}{\partial^2 t}, & \text{in } (0,T] \times \Omega \\
u'(0,x,z) = \frac{\partial u'}{\partial t}(0,x,z) &= 0, & (x,z) \in (0,L) \times (-H,0) \\
u'(t,0,z) &= u'(t,L,z), & (t,z) \in (0,T] \times (-H,0) \\
\frac{\partial u'}{\partial z}(t,x,-H) &= 0, & (t,x) \in (0,T] \times (0,L) \\
\frac{\partial u'}{\partial z}(t,x,0) &= 0, & (t,x) \in (0,T] \times (0,L)
\end{aligned}
\tag{34}
$$

Let us define the adjoint problem

$$
\begin{aligned}
\frac{1}{m^2}\frac{\partial^2 w}{\partial^2 t} - \Delta w &= 0, & &\text{in } (0,T] \times \Omega \\
w(T,x,z) = \frac{\partial w}{\partial t}(T,x,z) &= 0, & &(x,z) \in (0,L) \times (-H,0) \\
w(t,0,z) = w(t,L,z) &= 0, & &(t,z) \in (0,T] \times (-H,0) \\
\frac{\partial w}{\partial x}(t,0,z) + \frac{\partial w}{\partial x}(t,L,z) &= 0, & &(t,z) \in (0,T] \times (-H,0) \\
\frac{\partial w}{\partial z}(t,x,-H) &= 0, & &(t,x) \in (0,T] \times (0,L) \\
\frac{\partial w}{\partial z}(t,x,0) &= \left(\widehat{f}_{\widehat{\theta}}^{-1}\right)'^*[r(m)], & &(t,x) \in (0,T] \times (0,L)
\end{aligned}
\tag{35}
$$

We can then multiply the equation for u' by w and the equation for w by u' and use integration by part to show that

$$
\Phi'(m)[\delta m] = - \int_\Omega \frac{2}{m^3}\left(\int_0^T \frac{\partial w}{\partial t}\frac{\partial u}{\partial t}\,dt \right)\delta m\, d\mathbf{x} - \gamma \int_\Omega \left(\Delta m\right)\delta m\, d\mathbf{x}.
\tag{36}
$$

When the data in the inversion are collected from N_s different incoming sources $\{h_s\}_{s=1}^{N_s}$, the forward map $\mathbf{f}(m)$ and the data $\mathbf{g}^\delta$ defined in (3). Let u_s ($1 \le s \le N_s$) be solution to (16) with source h_s, and w_s be the solution to the adjoint equation (35) with the s-th component of $\left(\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}\right)'^*[\mathbf{r}(m)]$, here

$$
\mathbf{r}(m) = \widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}(\mathbf{f}(m)) - \widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}(\mathbf{g}^\delta),
\tag{37}
$$

then derivative of $\Phi(m)$ can be computed as

$$
\Phi'(m)[\delta m] = - \int_\Omega \frac{2}{m^3}\left(\sum_{s=1}^{N_s} \int_0^T \frac{\partial w_s}{\partial t}\frac{\partial u_s}{\partial t}\,dt \right)\delta m\, d\mathbf{x} - \gamma \int_\Omega \left(\Delta m\right)\delta m\, d\mathbf{x}.
\tag{38}
$$

Algorithm 1 Gradient Calculation with Adjoint State

1: **for** $s = 1$ to N_s **do**
2:     Solve (16) with h_s for u_s
3:     Evaluate the $f(m; h_s)$ component of $\mathbf{f}(m)$
4: **end for**
5: Evaluate $\mathbf{r}(m)$ according to (37) with the network $\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}$
6: Evaluate $\left(\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}\right)'^*[\mathbf{r}(m)]$ with the neural network
7: **for** $s = 1$ to N_s **do**
8:     Solve (35) with the s-th component of $\left(\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}\right)'^*[\mathbf{r}(m)]$ as the source term for w_s
9: **end for**
10: Evaluate $\Phi'(m)$ according to (38)

The computational procedure is summarized in Algorithm 1. The main difference between the calculation here and the adjoint calculation for a standard FWI gradient calculation is that we need to use the network $\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}$ to backpropagate the data into the velocity field in Line 5 of Algorithm 1 to compute the residual, and then use the adjoint of the network operator (transpose of the gradient in the discrete case), $\left(\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}\right)'^*$, to map the residual $\mathbf{r}(m)$ to the source of the adjoint wave equation in Line 6.

B Inversion with truncated Neumann series

The truncated Neumann series reconstruction (14) can be implemented with only the forward wave simulation and the learned neural network (without the need for the gradient of the learned operator). Let us define

$$m_0 := \widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}(\mathbf{g}^\delta), \quad R_J := \sum_{j=0}^{J-1} K^j(m_0),$$

with K defined in (14). We can then verify that

$$R_{J+1}(m_0) = (I + K\sum_{j=0}^{J-1} K^j)(m_0) = m_0 + KR_J(m_0) = m_0 + R_J(m_0) - \widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}(\mathbf{f}(R_J(m_0))). \quad (39)$$

This leads to the computational procedure summarized in Algorithm 2. The main differ-

---

Algorithm 2 Reconstruction with J-Term Truncated Neumann Series

---

1: Evaluate $m_0 := \widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}(\mathbf{g}^\delta)$ with the learned neural network
2: Set $m \leftarrow m_0$;
3: **for** $j = 1$ to $j = J - 1$ **do**
4:     **for** $s = 1$ to N_s **do**
5:         Solve (16) with (m, h_s) for u_s
6:         Evaluate the $f(m; h_s)$ component of $\mathbf{f}(m)$
7:     **end for**
8:     Update $m \leftarrow m_0 + m - \widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}(\mathbf{f}(m))$
9: **end for**

---

ence between the calculation here and the adjoint calculation for a standard FWI gradient calculation is that we only need to evaluate the network $\widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}$ to project the data back into the velocity field to compute the residual $m_0 - \widehat{\mathbf{f}}_{\widehat{\theta}}^{-1}(\mathbf{f}(m))$, and update the current result m.

C Network structure and training

For the sake of reproducibility of our research, we provide here the structures of the encoder, decoder and predictor networks we used in the encoder-decoder-predictor training framework described in Section 4.2; see Figure 16.
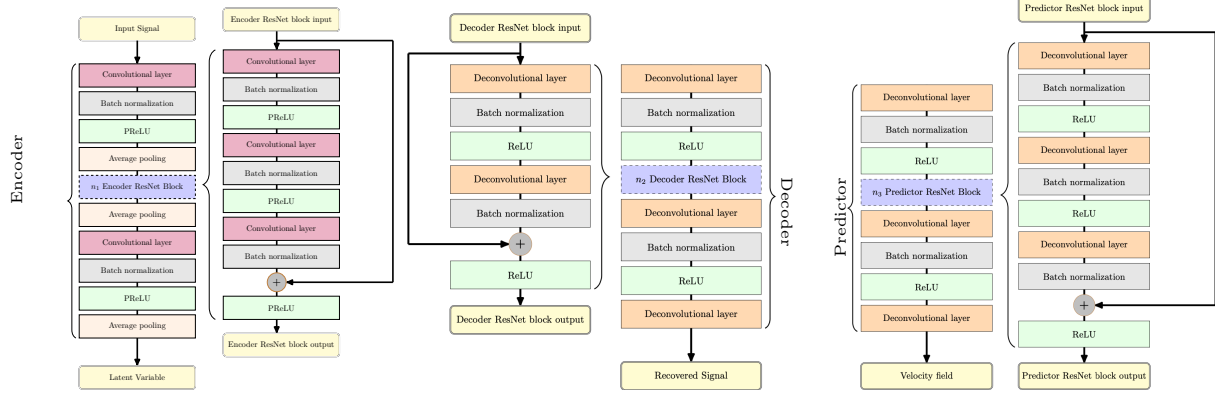
Figure 16: Network structures of the encoder, decoder and predictor networks.

The different layers of the networks are all standard as indicated by their names. In our implementation, the input of the neural network is the a $N_t \times N_d \times N_s$ tensor representing the solution of (16) for N_s sources, at N_d detector points $\{\mathbf{x}_d\}_{d=1}^{N_d}$, and on N_t time instances $\{t_j\}_{j=1}^{N_t}$: $u(t_i, \mathbf{x}_j; h_s)$, $i = 1, \cdots, N_t$, $j = 1, \cdots, N_d$, and $s = 1, \cdots, N_s$. The network outputs the recovered input (from the decoder and the reconstructed velocity field from the predictor. When the output velocity field is represented with the Fourier basis, the output of the predictor is an $M \times M$ matrix whose ij-element is $\mathfrak{m}(k_i, k_j)$ $(0 \leq k_i, k_j \leq M)$.

Besides the sizes of the network input (that is, the input of the encoder) and the network output (that is, the output of the predictor), the key parameters of the overall network are: (i) the size of the latent variables, and (ii) the number of ResNet blocks in each of the sub-networks (n_1, n_2 and n_3). In our implementation, we tested the network structure with different numbers of ResNet blocks. The training results are not sensitive to the selection of such numbers (which controls the size of the overall network). In the numerical simulations we presented in the paper, we use $n_1 = 10$, $n_2 = 5$ and $n_3 = 10$. The computational code we used for the numerical simulations in this paper, implemented using Python, are deposited at https://github.com/wending1/FWI_Deep_Learning.

The network training is achieved with the Adam optimizer [33]. The learning rate is initially set to be $5 * 10^{-4}$, and decays by a factor of 1.2 for every 5 epoch. The batch size is chosen to be 128. We stop the training after 50 epoch.

References

[1] A. ADLER, M. ARAYA-POLO, AND T. POGGIO, *Deep learning for seismic inverse problems: Toward the acceleration of geophysical analysis workflows*, IEEE Signal Processing Magazine, 38 (2021), pp. 89–119.

[2] J. ADLER AND O. ÖKTEM, *Solving ill-posed inverse problems using iterative deep neural networks*, Inverse Problems, 33 (2017). 124007.

[3] V. Akçelik, G. Biros, and O. Ghattas, *Parallel multiscale Gauss-Newton-Krylov methods for inverse wave propagation*, in Proceedings of the 2002 ACM/IEEE Conference on Super-computing, 2002, pp. 1–15.

[4] M. Araya-Polo, A. Adler, S. Farris, and J. Jennings, *Fast and accurate seismic tomography via deep learning*, in Deep Learning: Algorithms and Applications, Springer, 2020, pp. 129–156.

[5] E. Bachmann and J. Tromp, *Source encoding for viscoacoustic ultrasound computed to-mography*, J. Acoust. Soc. Am., 147 (2020), pp. 3221–3235.

[6] G. Bao and W. W. Symes, *On the sensitivity of hyperbolic equation to the coefficient*, Comm. in P.D.E., 21 (1996), pp. 395–422.

[7] G. Bao, X. Ye, Y. Zang, and H. Zhou, *Numerical solution of inverse problems by weak adversarial networks*, Inverse Problems, 36 (2020). 115003.

[8] S. Bernard, V. Monteiller, D. Komatitsch, and P. Lasaygues, *Ultrasonic computed tomography based on full-waveform inversion for bone quantitative imaging*, Phys. Med. Bio., 62 (2017), pp. 7011–7035.

[9] L. Borcea, V. Druskin, A. Mamonov, and M. Zaslavsky, *Untangling the nonlinearity in inverse scattering with data-driven reduced order models*, Inverse Problems, 34 (2018). 065008.

[10] R. Brossier, S. Operto, and J. Virieux, *Which data residual norm for robust elastic frequency-domain full waveform inversion*, Geophysics, 75 (2010), pp. R37–R46.

[11] T. A. Bubba, M. Galinier, M. Lassas, M. Prato, L. Ratti, and S. Siltanen, *Deep neural networks for inverse problems with pseudodifferential operators: An application to limited-angle tomography*, SIAM J. Imaging Sci., 14 (2021), pp. 470–505.

[12] C. Bunks, F. M. Saleck, S. Zaleski, and G. Chavent, *Multiscale seismic waveform inversion*, Geophysics, 50 (1995), pp. 1457–1473.

[13] C. Burstedde and O. Ghattas, *Algorithmic strategies for full waveform inversion: 1D experiments*, Geophysics, 74 (2009), pp. 37–46.

[14] J. Chen, Y. Chen, H. Wu, and D. Yang, *The quadratic Wasserstein metric for earthquake location*, J. Comput. Phys., 373 (2018), pp. 188–209.

[15] K. Chen and M. D. Sacchi, *Time-domain elastic Gauss-Newton full-waveform inversion: a matrix-free approach*, Geophys. J. Int., 223 (2020), pp. 1007–1039.

[16] G. Côrte, J. Dramsch, H. Amini, and C. MacBeth, *Deep neural network application for 4D seismic inversion to changes in pressure and saturation: Optimizing the use of synthetic training datasets*, Geophysical Prospecting, 68 (2020), pp. 2164–2185.

[17] T. Dierkes, O. Dorn, F. Natterer, V. Palamodov, and H. Sieschott, *Fréchet derivatives for some bilinear inverse problems*, SIAM J. Appl. Math., 62 (2002), pp. 2092–2113.

[18] B. Engquist, B. D. Froese, and Y. Yang, *Optimal transport for seismic full waveform inversion*, Commun. Math. Sci., 14 (2016), pp. 2309–2330.

[19] B. Engquist, K. Ren, and Y. Yang, *The quadratic Wasserstein metric for inverse data matching*, Inverse Problems, 36 (2020), p. 055001. arXiv:1911.06911.

[20] ——, *A generalized weighted optimization method for computational learning and inversion*, ICLR 2022, (2022). arXiv:2201.09223.

[21] I. Epanomeritakis, V. Akcelik, O. Ghattas, and J. Bielak, *A Newton-CG method for large-scale three-dimensional elastic full-waveform seismic inversion*, Inverse Problems, 24 (2008). 034015.

[22] J. Fang, H. Zhou, Y. E. Li, Q. Zhang, L. Wang, P. Sun, and J. Zhang, *Data-driven low-frequency signal recovery using deep-learning predictions in full-waveform inversion*, Geophysics, 85 (2020), pp. A37–A43.

[23] S. Farris, M. Araya-Polo, J. Jennings, B. Clapp, and B. Biondi, *Tomography: a deep learning vs full-waveform inversion comparison*, in Proceedings, First EAGE Workshop on High Performance Computing for Upstream in Latin America, vol. 2018, 2018, pp. 1–5.

[24] J. Feliu-Fabà, Y. Fan, and L. Ying, *Meta-learning pseudo-differential operators with deep neural networks*, J. Comput. Phys., 404 (2020). 109309.

[25] A. Fichtner, *Full Seismic Waveform Modelling and Inversion*, Springer-Verlag, Berlin, 2011.

[26] R. Fletcher, *Practical methods of optimization*, John Wiley & Sons, 2013.

[27] L. Guasch, O. Calderón Agudo, M. X. Tang, P. Nachev, and M. Warner, *Full-waveform inversion imaging of the human brain*, Digit. Med., 3 (2000). 28.

[28] W. Hu, Y. Jin, X. Wu, and J. Chen, *Progressive transfer learning for low-frequency data prediction in full-waveform inversion*, Geophysics, 86 (2021), pp. R369–R382.

[29] V. Isakov, *Inverse Problems for Partial Differential Equations*, Springer-Verlag, New York, second ed., 2006.

[30] A. Javaherian, F. Lucka, and B. T. Cox, *Refraction-corrected ray-based inversion for three-dimensional ultrasound tomography of the breast*, Inverse Problems, 36 (2020). 125010.

[31] Z. Jia, R. Guo, M. Li, G. Wang, Z. Liu, and Y. Shao, *3-d model-based inversion using supervised descent method for aspect-limited microwave data of metallic targets*, IEEE Trans. Geosci. Remote Sens., (2021), pp. 1–10.

[32] V. Kazei, O. Ovcharenko, P. Plotnitskii, D. Peter, X. Zhang, and T. Alkhalifah, *Mapping full seismic waveforms to vertical velocity profiles by deep learning*, Geophysics, 86 (2021), pp. 1–50.

[33] D. Kingma and J. Ba, *Adam: A method for stochastic optimization*, arXiv:1412.6980, (2014).

[34] F. Li, U. Villa, S. Park, and M. A. Anastasio, *Three-dimensional stochastic numerical breast phantoms for enabling virtual imaging trials of ultrasound computed tomography*, arXiv:2106.02744, (2021).

[35] Z. Lin, R. Guo, M. Li, A. Abubakar, T. Zhao, F. Yang, and S. Xu, *Low-frequency data prediction with iterative learning for highly nonlinear inverse scattering problems*, IEEE Trans. Microw. Theory Tech., (2021).

[36] B. Liu, S. Yang, Y. Ren, X. Xu, P. Jiang, and Y. Chen, *Deep-learning seismic full-waveform inversion for realistic structural models*, Geophysics, 86 (2021), pp. R31–R44.

[37] Q. Liu, S. Beller, W. Lei, D. Peter, and J. Tromp, *Preconditioned BFGS-based uncertainty quantification in elastic full waveform inversion*, arXiv:2009.12663, (2020).

[38] F. Lucka, M. Pérez-Liva, B. E. Treeby, and B. T. Cox, *High resolution 3D ultrasonic breast imaging by time-domain full waveform inversion*, arXiv:2102.00755, (2021).

[39] S. Mache, P. K. Pokala, K. Rajendran, and C. S. Seelamantula, *DuRIN: A deep-unfolded sparse seismic reflectivity inversion network*, arXiv:2104.04704, (2021).

[40] T. P. Matthews, J. Poudel, L. Li, L. V. Wang, and M. A. Anastasio, *Parameterized joint reconstruction of the initial pressure and sound speed distributions for photoacoustic computed tomography*, SIAM J. Imaging Sci., 11 (2018), pp. 1560–1588.

[41] L. Métivier, A. Allain, R. Brossier, Q. Mérigot, E. Oudet, and J. Virieux, *Optimal transport for mitigating cycle skipping in full waveform inversion: a graph space transform approach*, Geophysics, 83 (2018), pp. R515–R540.

[42] R. Modrak and J. Tromp, *Seismic waveform inversion best practices: regional, global and exploration test cases*, Geophys. J. Int., 206 (2016), pp. 1864–1889.

[43] R.-E. Plessix, *A review of the adjoint-state method for computing the gradient of a functional with geophysical applications*, Geophys. J. Int., 167 (2006), pp. 495–503.

[44] 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.

[45] R. G. Pratt, C. Shin, and G. J. Hicks, *Gauss-Newton and full Newton methods in frequency-space seismic waveform inversion*, Geophys. J. Int., 133 (1998), pp. 341–362.

[46] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, and A. Courville, *On the spectral bias of neural networks*, in Proceedings of the 36th International Conference on Machine Learning, vol. 97, 2019, pp. 5301–5310.

[47] M. Raissi, P. Perdikaris, and G. E. Karniadakis, *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations*, J. Comput. Phys., 378 (2019), pp. 686–707.

[48] Y. Ren, L. Nie, S. Yang, P. Jiang, and Y. Chen, *Building complex seismic velocity models for deep learning inversion*, IEEE Access, 9 (2021), pp. 63767–63778.

[49] R. Rojas-Gomez, J. Yang, Y. Lin, J. Theiler, and B. Wohlberg, *Physics-consistent data-driven waveform inversion with adaptive data augmentation*, IEEE Geosci. Remote. Sens. Lett., (2020), pp. 1–5.

[50] B. Ronen, D. Jacobs, Y. Kasten, and S. Kritchman, *The convergence rate of neural networks for learned functions of different frequencies*, in Advances in Neural Information Processing Systems, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds., vol. 32, 2019.

[51] F. Santosa and W. W. Symes, *Analysis of Least-squares velocity inversion*, Society of Exploration Geophysicists, 1989.

[52] L. Sirgue and R. G. Pratt, *Efficient waveform inversion and imaging: A strategy for selecting temporal frequencies*, Geophysics, 69 (2004), pp. 231–248.

[53] J. D. Smith, K. Azizzadenesheli, and Z. E. Ross, *Eikonet: Solving the eikonal equation with deep neural networks*, arXiv:2004.00361, (2020).

[54] C. Song and T. Alkhalifah, *Wavefield reconstruction inversion via physics-informed neural networks*, arXiv:2104.06897, (2021).

[55] B. Sun and T. Alkhalifah, *ML-misfit: learn a robust misfit function for full-waveform inversion using machine learning*, arXiv:2002.03163v2, (2020).

[56] H. Sun and L. Demanet, *Extrapolated full-waveform inversion with deep learning*, Geophysics, 85 (2020), pp. R275–R288.

[57] J. Sun, K. A. Innanen, and C. Huang, *Physics-guided deep learning for seismic inversion with hybrid training and uncertainty analysis*, Geophysics, 86 (2021), pp. R303–R317.

[58] W. W. Symes, *Migration velocity analysis and waveform inversion*, Geophysical Prospecting, 56 (2008), pp. 765–790.

[59] J. Tromp, C. Tape, and Q. Liu, *Seismic tomography, adjoint methods, time reversal and banana-doughnut kernels*, Geophys. J. Int., 160 (2005), pp. 195–216.

[60] J. Virieux, A. Asnaashari, R. Brossier, L. Métivier, A. Ribodetti, and W. Zhou, *An introduction to full waveform inversion*, in Encyclopedia of Exploration Geophysics, 2014, pp. R1–1–R1–40.

[61] J. Virieux and S. Operto, *An overview of full-waveform inversion in exploration geophysics*, Geophysics, 74 (2009), pp. WCC1–WCC26.

[62] J. Wiskin, B. Malik, D. Borup, N. Pirshafiey, and J. Klock, *Full wave 3D inverse scattering transmission ultrasound tomography in the presence of high contrast*, Scientific Reports, 10 (2020). 20166.

[63] Y. Wu and Y. Lin, *InversionNet: An efficient and accurate data-driven full waveform inversion*, IEEE Transactions on Computational Imaging, 6 (2020), pp. 419–433.

[64] Z. J. Xu, Y. Zhang, and Y. Xiao, *Training behavior of deep neural network in frequency domain*, arXiv:1807.01251v3, (2018).

[65] F. Yang and J. Ma, *Deep-learning inversion: A next-generation seismic velocity model building method*, Geophysics, 84 (2019), pp. R583–R599.

[66] S. Yu and J. Ma, *Data-driven geophysics: from dictionary learning to deep learning*, arXiv:2007.06183, (2020).

[67] W. Zhang and J. Gao, *Deep-learning full-waveform inversion using seismic migration images*, IEEE Trans. Geosci. Remote Sens., (2021), pp. 1–18.

[68] Z. Zhang and Y. Lin, *Data-driven seismic waveform inversion: A study on the robustness and generalization*, IEEE Trans. Geosci. Remote Sens., 58 (2020), pp. 6900–6913.