Physics-guided deep learning for seismic inversion: hybrid training and uncertainty analysis

Jian Sun*†, Kris Innanen*, Chao Huang‡

ABSTRACT

The determination of subsurface elastic property models is crucial in quantitative seismic data processing and interpretation. This problem is commonly solved by deterministic physical methods, such as tomography or full waveform inversion (FWI). However, these methods are entirely local, and require accurate initial models. *Deep learning* represents a plausible class of methods for seismic inversion, which may avoid some of the issues of purely descent-based approaches. However, any generic deep learning network capable of relating each elastic property cell value to each sample in a seismic dataset would require a very large number of degrees of freedom. Two approaches might be taken to train such a network. First, by invoking a massive and exhaustive training dataset; second, by working to reduce the degrees of freedom by enforcing physical constraints on model-data relationship. The second approach is referred to as "physics-guiding". Based on recent progress of wave theory-designed (i.e., physics-based) network, we propose a hybrid network design, involving both deterministic, physics-based modelling and data-driven deep learning components. From an optimization standpoint, both a data-driven model misfit (i.e., standard deep learning), and now a physics-guided data residual (i.e., a wave propagation network), are simultaneously minimized during the training of the network. An experiment is carried out to analyze the trade-off between two types of losses. Synthetic velocity building is employed to examine the capacity of hybrid training. Comparisons demonstrate that, given the same training dataset, the hybrid-trained network outperforms the traditional fully data-driven network. Additionally, a comprehensive error analysis is performed to quantitatively compare the fully data-driven and hybrid physics-guided approaches. The network is applied to SEG salt model data and the uncertainty is analyzed, in order to further examine the benefits of hybrid training.

INTRODUCTION

Estimation of subsurface structures and physical parameters using seismic data can be carried out based on fully deterministic models of wave propagation. This category of approach includes, for instance, traditional tomography (Woodward et al., 2008) for velocity model building by migrating seismograms with traveltime information, and full waveform inversion (FWI) for the high resolution reconstruction of subsurface velocity, or elastic property models (Tarantola, 1984; Virieux and Operto, 2009). These approaches are powerful, and in particular in the case of the latter, represent the state of the art in seismic inversion. However, they tend to be computationally complex, with results and computational requirements that are both very sensitive to parameterization, data completeness, initial model, and numerical optimization approach.

*CREWES Project, University of Calgary
†College of Earth and Mineral Science, The Pennsylvania State University
‡School of Ocean and Earth Science, Tongji University

Deep learning, a recent extension of machine learning technology, has drawn significant recent attention across various fields, including seismology. Deep learning applications to problems of image processing include classification (Krizhevsky et al., 2012; Simonyan and Zisserman, 2014), segmentation(Long et al., 2015; Ronneberger et al., 2015; Häggström et al., 2019), denoising (Zhang et al., 2017), and superresolution (Kim et al., 2016). They are often carried out with convolutional neural networks (CNNs), a popular class of network which can be formulated with a relatively small number of trainable parameters. For example, Simonyan and Zisserman (2014) observed unusually high performance in a CNN called VGG16, which included a convolutional encoder and a classifier, for image recognition; Ronneberger et al. (2015) designed a CNN with an encoder and decoder architecture in an U-shape (Unet), and reported high accuracy in its application to biomedical image segmentation tasks.

Approaches of this general type have been applied successfully in geophysical disciplines also. However, here we encounter challenges of the kind discussed by Karpatne et al. (2018), because we tend to have (1) large and rich, but not exhaustive, datasets, and (2) deterministic physical models, in the form of partial differential equations, which accurately relate data to subsurface properties under certain assumptions. In the relatively rare circumstance in which exhaustive data are available for a given geophysical problem, successes are striking. For instance, Araya-Polo et al. (2017) and Wu et al. (2019) demonstrated that a CNN can be trained to predict the presence of faults; Ovcharenko et al. (2019) applied low-frequency extrapolation from multi-offset seismic data using a CNN; Kaur et al. (2019) formulated elastic wave mode-separation in heterogeneous anisotropic media using a generative adversarial network (GAN); Das et al. (2019) and Das and Mukerji (2020) successfully performed seismic impedance inversion and petrophysical properties prediction using a CNN; Yang and Ma (2019) set up a velocity building procedure for subsurface salt bodies using an Unet, and discussed the influence of low-frequencies and noise; Fabien-Ouellet and Sarkar (2020) developed a deep recurrent neural network (RNN) for velocity estimation using common-midpoint (CMP) gathers; fine results of tomography using deep learning can also be found (Araya-Polo et al., 2018; Adler et al., 2019; Araya-Polo et al., 2020). These approaches are fully data driven, and thus rely entirely on the completeness of training data. Supplying such data could be difficult or impossible for the solution of a range of important geophysical problems, to the point where they would seem to be essentially inaccessible by a deep learning approach. For instance, if a FWI-like problem was broached, in which each sample of a large seismic data set must be mapped to each element of a multidimensional elastic property model, the number of degrees of freedom within the deep learning network would be exceedingly large. Training data of the extent needed to constrain such a network are unlikely to be available.

However, these issues, which derive from item (1) above, i.e., the requirement of large and rich training dataset, can be addressed to a great degree by making use of item (2), i.e., deterministic physical models. That is, the degrees of freedom of an otherwise vast and complex network can be reduced by constraining the network with physical rules. For instance, Wu and McMechan (2019) showed that an FWI procedure can be set up within a CNN framework with a multi-grid parameterization of the subsurface model, using a multi-layer neural network. Meanwhile, Sun et al. (2020) and Richardson (2018) discussed a theory-guided waveform inversion using a RNN designed to emulate wave propagation;

the former authors further confirmed that the training stage within such a waveform RNN is equivalent to the optimization of standard, gradient-based FWI. Zhang et al. (2020) extended this to incorporate more complete, multidimensional elastic and viscoelastic, models and parameter estimations. These networks, as initially realized, were sufficiently well-constrained by the rules of acoustic or elastic wave propagation, that they were successfully trained with a single training data set (hence their resemblance to standard FWI). Ideally, such networks would allow for a greater role for training when additional data exist. For instance, Biswas et al. (2019) successfully set up a physics-guided CNN for seismic impedance inversion without labeled data by incorporating a convolutional operation to the retrieved reflectivity. However, difficulties will be encountered with such a physics-guided scheme applied in waveform inversion problem, which will be discussed in this paper. Similar to the schematic flow shown in Alfarraj and AlRegib (2019) for impedance inversion, we examine the opportunity deep learning networks offer for hybridizing data-driven training with waveform inversion, by constructing a convolutional neural network in which both data-driven model misfits and physics-based data residuals are simultaneously minimized, which has not been broached in the literature.

This paper is organized as follows. First, we review the deep learning formulation and solution of seismic inverse problems, and then discuss the possible scheme of hybridizing the traditional deep learning network and physics-guiding framework. Second, we introduce the architecture of our proposed network and discuss the influence of trade-off between the model misfit and data residual in a 2D synthetic salt velocity model building environment. After that, we compare the performance of networks trained in a fully data-driven and a hybrid physics-guided modes, respectively. Finally, the quantitative error analysis, generalization ability of the network, and predictive uncertainty analysis are implemented to further comprehensively examine the network's ability.

METHODOLOGY

Physics-based deterministic solution

Sun et al. (2020) demonstrated that in the mapping between velocity and wave data, the physics of wave propagation can be explicitly enforced within a wave theory-designed (i.e., physics-based) RNN. Such a network is optimized by a crosscorrelation, in time, of the backpropagated data residuals and the second-order partial derivative of the forward-modeled states, in a manner equivalent to a gradient-based FWI. During the training, the physics-based RNN optimizes the input by minimizing the data discrepancy in the context of the physical rules.

Given an estimated set of subsurface model parameters $\tilde{\mathbf{m}}$, the physics-based data residual can be obtained by forward-propagating information through a physics-based waveform RNN operator $\mathcal{F}(\cdot)$. The normalized data residual in an $\ell 2$-norm is in this case

$$\mathcal{L}_d^k(\tilde{\mathbf{m}}, \mathbf{d}) = \frac{1}{Tkn_s} \sum_{i=1}^{k} \sum_{s=1}^{n_s} \sum_{t=0}^{T} \left\| \frac{\mathbf{d}_i^{s,t} - \mathcal{F}^{s,t}(\tilde{\mathbf{m}}_i)}{max(\mathbf{d}_i) + \epsilon} \right\|^2, \tag{1}$$

where k denotes the number of velocity models, t is the time step in the RNN, i.e., seismic sample interval, n_s is the number of sources per subsurface model, $\mathbf{d}$ represents the

observed seismic data, and $\tilde{\mathbf{d}}_i^{s,t} = \mathcal{F}^{s,t}(\tilde{\mathbf{m}}_i)$ represents the waveform RNN forward propagation. ϵ is a small value to prevent dividing by zeros. Optimal inputs are those subsurface physics parameters which minimize equation 1, written as

$$\tilde{\mathbf{m}} = \arg\min_{\tilde{\mathbf{m}}} \tilde{\mathcal{L}}_d^k(\tilde{\mathbf{m}}, \mathbf{d}) = \arg\min_{\tilde{\mathbf{m}}} \frac{1}{Tkn_s} \sum_{i=1}^{k} \sum_{s=1}^{n_s} \sum_{t=0}^{T} \left\| \frac{\mathbf{d}_i^{s,t} - \mathcal{F}^{s,t}(\tilde{\mathbf{m}}_i)}{max(\mathbf{d}_i) + \epsilon} \right\|^2 \tag{2}$$

In gradient-descent training, the model update is based on the gradient of $\mathcal{L}_d(\tilde{\mathbf{m}}, \mathbf{d})$ with respect to the model parameters $\tilde{\mathbf{m}}$:

$$\delta\tilde{\mathbf{m}} = -\frac{\partial \tilde{\mathcal{L}}_d^k}{\partial \tilde{\mathbf{m}}} = -\left[\frac{\partial \tilde{\mathcal{L}}_d^k}{\partial \tilde{\mathbf{d}}_i^{s,t}}\right] \frac{\partial \tilde{\mathbf{d}}_i^{s,t}}{\partial \tilde{\mathbf{m}}}, \tag{3}$$

where $\left[\partial \tilde{\mathcal{L}}_d^k / \partial \tilde{\mathbf{d}}_i^{s,t}\right]$ is determined by the detailed structure of the forward operator $\mathcal{F}(\cdot)$, and contains all partial derivatives via the chain rule. For instance, with the temporal second-order finite difference operator in Figure 1, $\left[\partial \tilde{\mathcal{L}}_d^k / \partial \tilde{\mathbf{d}}_i^{s,t}\right]$ contains three contributions at time steps $t, t + \Delta t, t + 2\Delta t$, respectively:

$$\left[\frac{\partial \tilde{\mathcal{L}}_d^k}{\partial \tilde{\mathbf{d}}_i^{s,t}}\right] = \left[\frac{\partial \tilde{\mathcal{L}}_d^k}{\partial \tilde{\mathbf{d}}_i^{s,t+2\Delta t}}\right] \frac{\partial \tilde{\mathbf{d}}^{s,t+2\Delta t}}{\partial \tilde{\mathbf{d}}_i^{s,t}} + \left[\frac{\partial \tilde{\mathcal{L}}_d^k}{\partial \tilde{\mathbf{d}}_i^{s,t+\Delta t}}\right] \frac{\partial \tilde{\mathbf{d}}^{s,t+\Delta t}}{\partial \tilde{\mathbf{d}}_i^{s,t}} + \frac{\partial \tilde{\mathcal{L}}_d^k}{\partial \tilde{\mathbf{d}}_i^{s,t}}, \tag{4}$$

where $0 \leq t \leq T$, T being the maximum time of wave propagation, and where the initial conditions for the RNN backpropagation are assumed to be zeros, i.e., $\left[\partial \tilde{\mathcal{L}}_d^k / \partial \tilde{\mathbf{d}}_i^{s,t}\right]_{t=T+\Delta T, T+2\Delta T} = 0$. A detailed derivation of the gradient for a full time sequence (i.e., time steps from 0 to T) using the chain rule appears elsewhere (Sun et al., 2020).

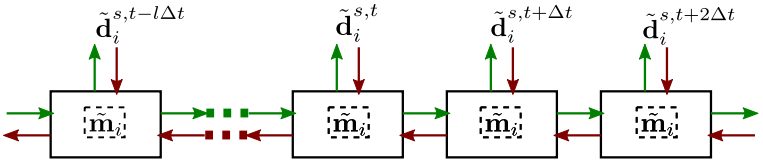

FIG. 1. Truncated structure of waveform RNN,where the green (red) arrows delineate the forward (backward) propagation direction.

The auto-differential backpropagation method for neural networks requires storage of all related internal states, which is computationally expensive, especially for RNN backpropagation. Therefore, a truncated backpropagation, instead of the full time sequence backpropagation, is commonly applied in minimizing the RNN loss function shown in equation 2.

For a full time sequence RNN ($0 \rightarrow T$), initial conditions are applied at time steps $T + 1$, and $T + 2$ which are initialized as zeros. The truncated approach instead divides the RNN cells into several time segments, such as $0 \rightarrow (l-1)\Delta t$, $l\Delta t \rightarrow (2l-1)\Delta t$, $\cdots$, $t - l\Delta t \rightarrow t$, $\cdots$, $nl\Delta t \rightarrow T$, and zero initial conditions are applied at each time breaking steps $l\Delta t$, $(l+1)\Delta t$, $2l\Delta t$, $(2l+1)\Delta t$, $\cdots$, $t + \Delta t$, $t + 2\Delta t$, $\cdots$, $T + \Delta t$, $T + 2\Delta t$. For example, considering a time segment $t - l\Delta t$ to t (see Figure 1). The backpropagation is

performed from t to $t - l\Delta t$. According to equation 4, the partial derivatives of the data residual at time step t comprise three partitions. However, we lack information at time step $t + \Delta t$ and $t + 2\Delta t$ in a truncated RNN, so these two partial derivative contributions are initialized to zero, i.e., $\left[\partial\tilde{\mathcal{L}}_d^k/\partial\tilde{\mathbf{d}}_i^{s,t}\right]_{t=t+\Delta t,t+2\Delta t} = 0$. The truncated approach is in this sense only approximately equivalent to the full time sequence RNN backpropagation; it does, however, permit a trade-off to be made between the speed of convergence and memory requirements.

The physics-based RNN has been demonstrated in Sun et al. (2020) and its schematic workflow is shown in Figure 2a. However, two major disadvantages remain in such a framework: 1) a good initial model is required to avoid converging to an undesired local minimum; 2) the training process has to be repeated to solve for different models. In this paper we extend these ideas, specifically adding a data-driven deep learning component to the physics-based network, to address these issues. Next, we will briefly review the deep learning solution of seismic inverse problems and discuss its formulation.

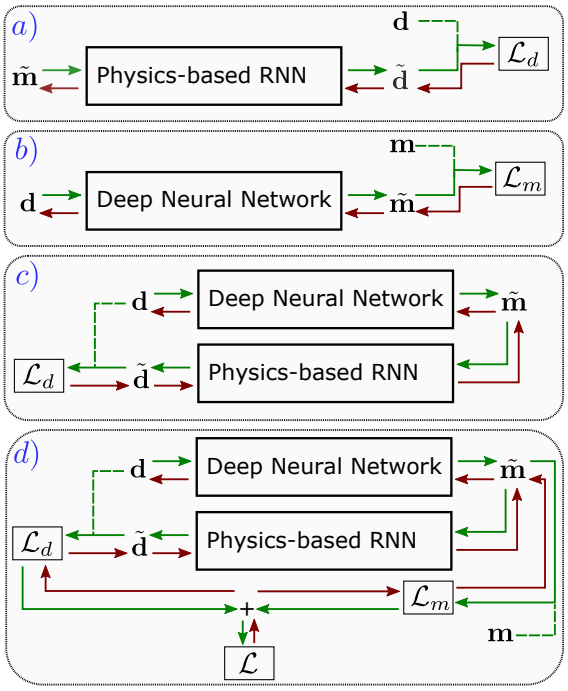

FIG. 2. The schematic workflows: a) a physics-based RNN, b) a data-driven deep neural network, c) a physics-guided neural network, d) a hybrid physics-guided neural network, where the green (red) arrows delineate the forward (backward) propagation and dashed green lines indicate labeled data. **d** and $\tilde{\mathbf{d}}$ represent observed and estimated seismic data, respectively; **m** and $\tilde{\mathbf{m}}$ are the ground-truth and estimated velocity models, respectively.

Data-driven deep learning solution

Let the seismic inverse problem as solved within a neural network be the determination of the medium property model vector **m** from the observed seismic data **d**:

$$\tilde{\mathbf{m}} = \Omega\left(\mathbf{d}; \Theta\right), \tag{5}$$

where the operator $\Omega(\cdot)$ contains the full nonlinear relationship between the input data and the estimated subsurface property $\tilde{\mathbf{m}}$, and involves a sequence of nonlinear transforms with trainable weights Θ. In a neural network, the nonlinearity is commonly included using activation mapping, such as a rectified linear unit, or ReLU (Nair and Hinton, 2010). $\Omega(\cdot)$ is the inverse of a forward operator $\mathcal{F}(\cdot)$ which contains all wave propagation rules as well as source and acquisition information, i.e., $\mathbf{d} = \mathcal{F}(\mathbf{m})$. For a seismic velocity model building problem, the inputs are seismic shot gathers and the outputs are velocity models. The best-fit mapping from input to output is determined in a training stage, during which all trainable parameters Θ are estimated using a training dataset. This is carried out in a numerical optimization procedure, in which an objective function measuring the model misfit (i.e., the discrepancy between the estimated velocity $\tilde{\mathbf{m}}$ and the ground-truth velocity model $\mathbf{m}$) is minimized. In the case of an $\ell 2$-norm, for instance, this objective function is

$$\mathcal{L}_m^n(\mathbf{m}, \mathbf{d}) = \frac{1}{n} \sum_{i=0}^{n} \|\mathbf{m}_i - \tilde{\mathbf{m}}_i\|^2 = \frac{1}{n} \sum_{i=0}^{n} \|\mathbf{m}_i - \Omega(\mathbf{d}_i; \Theta)\|^2 , \tag{6}$$

where the subscript m is a reminder that the function pertains to model rather than data misfit; n is the number of paired samples in the training data distribution.

To manage the large number of training samples normally involved, and also the memory limitations of either graphic processing units (GPU) or tensor processing units (TPU), a mini-batch strategy is applied in both training and validation stages. Rather than minimizing $\mathcal{L}_m^n$ with n covering the entire training dataset in a single optimization, the training samples are randomly divided into batches, each containing k samples (k is referred to as the batch size). The weights are calculated by minimizing the objective function over each batch, via:

$$\hat{\Theta} = \arg\min_{\Theta} \mathcal{L}_m^k(\mathbf{m}, \mathbf{d}) = \arg\min_{\Theta} \frac{1}{k} \sum_{i=1}^{k} \|\mathbf{m}_i - \Omega(\mathbf{d}_i; \Theta)\|^2. \tag{7}$$

A single epoch of training is defined as a single passing of all training batches forward and backward through the neural network. In velocity model building problems, model parameters have a large range of possible values which can be plausibly taken on, and velocity models may be of high and low variance. Elements within the random batches in these situations can make unequal contributions to the objective function, with high-variance velocity models tending to dominate in the convergence process. To balance the contributions of the samples in a single batch, in our general network we adopt a normalized objective function:

$$\hat{\Theta} = \arg\min_{\Theta} \tilde{\mathcal{L}}_m^k(\mathbf{m}, \mathbf{d}) = \arg\min_{\Theta} \frac{1}{k} \sum_{i=1}^{k} \left\| \frac{\mathbf{m}_i - \Omega(\mathbf{d}_i; \Theta)}{max(\mathbf{m}_i) + \epsilon} \right\|^2. \tag{8}$$

The flowchart describing the training of a neural network in a traditional and fully data-driven way is shown in Figure 2b, where the green arrows describe the forward propagation of information through the neural network and the red arrows represent the backpropagation for parameter updating. Unlike the physics-based network that updates the subsurface parameters directly, the training of a deep neural network searches for the best-fit mapping between input and output by adjusting internally weighted connections.

Physics-guided neural network solution

Though the deep neural network can be a powerful tool to solve seismic inversion problems, its training process requires vast quantities of shot-and-model pairs illuminating diverse subsurface structures. Building such a training dataset is impractical. Therefore, incorporating the known physics knowledge into the training process of a neural network will preserve advantages of both deep learning and physics-based deterministic methods. Successful implementation of such a scheme can be found in Biswas et al. (2019), where a CNN is trained for seismic impedance inversion by incorporating a simple convolutional operation between wavelet and retrieved reflectivity. We can adopt a similar strategy to solve the waveform inversion problem by combining a deep neural network with a physics-based RNN, which allows parameter updating is performed by the backpropagation of both using automatic differential technique. In this physics-guided framework, raw shot gathers are input into the deep neural network; the estimated subsurface model parameters by the deep neural network are then used as inputs for the physics-based RNN, from which we obtain simulated seismic records. The physics-guided optimization procedure of a neural network contains five steps:

1) Feeding shot gathers $\mathbf{d}$ into the neural network to estimate subsurface models $\tilde{\mathbf{m}} = \Omega\left(\mathbf{d}; \Theta\right)$
2) Feeding estimated subsurface models $\tilde{\mathbf{m}}$ into a physics-based RNN to calculate predicted shot gathers $\tilde{\mathbf{d}} = \mathcal{F}(\tilde{\mathbf{m}}) = \mathcal{F}(\Omega\left(\mathbf{d}; \Theta\right))$
3) Calculating the data misfit between the input shot gathers (i.e., observation) $\mathbf{d}$ and prediction $\tilde{\mathbf{d}}$ using equation 2
4) Backpropagating the data misfit to update parameters Θ
5) Repeating above steps until convergence

The process of training a physics-guided neural network (PGNN) is illustrated in Figure 2c. In contrast, a network simultaneously trained by both model misfit and data residual which we will refer to as hybrid physics-guided neural network (H-PGNN), is illustrated in Figure 2d. The model misfit is calculated between the network estimated results and the ground-truth velocity models, and the data residual is computed between the simulated seismic records and raw shot gathers. This permits an objective function to be formed from a weighted sum of model misfit (equation 8) and data residual (equation 2):

$$\mathcal{L}^k = \lambda_m \tilde{\mathcal{L}}_m^k + \lambda_d \tilde{\mathcal{L}}_d^k, \tag{9}$$

where λ_m and λ_d moderates trade-off between model and data misfits, respectively. Both fully data-driven and purely physics-guided training as end-members can be produced by assigning different combinations of two trade-off parameters, i.e., ($\lambda_m = 1$, $\lambda_d = 0$) for data-driven and ($\lambda_m = 0$, $\lambda_d = 1$) for physics-guided, which are two special cases of the hybrid objectives.

For the hybrid training scheme, λ_m and λ_d must be selected to balance the contributions from model misfit and physics-based data residual. In previous work (Sun et al., 2020)

on the seismic velocity inversion problem, to achieve the fast and stable convergence, a suitable range of learning rate for the Adaptive Momentum, or *Adam*, algorithm (Kingma and Ba, 2014) is $\alpha \in [10, 100]$, in contrast to the more common range $[0, 1]$ used in deep learning optimizations. It means that, λ_d/λ_m needs to be in range of $[10, 100]$ to balance the gradients obtained using model misfit and data residual, respectively. In the experiments to follow, with trial and error, we choose $\lambda_d = 40$ which will remain to be constant during training, and let λ_m to be adjustable.

Network architecture

We select an encoder-decoder CNN design as the primary architecture. In it, we expect the encoder accepts seismic shot gathers or their weighted summations as input and generates low-dimensional feature maps through a sequence of convolution blocks and maxpooling layers. Then, the decoder transforms these feature maps into the estimated velocity model using a sequence of convolution blocks and transposed convolution layers (this latter process is more commonly known as "deconvolution" in the deep learning community, but to avoid confusion with the geophysical term we will continue to refer to it as "transposed convolution"). The architecture of the CNN is plotted in Figure 3. There are four partitions in the proposed network, including an initial convolution, an encoder network, a decoder network, and a sequence of re-scaling and clamping processes. The initial convolution (plotted as the pink column in Figure 3) contains a single convolutional layer with 16 convolutional filters of size $1 \times 1 \times 10$. Thus the proposed network accepts 10 shot gathers per model in the velocity model building. The purpose of this initial convolution layer is to generate feature maps with an appropriate number of channels for the encoder network. In other words, the encoder network will be trained to accept 16 feature maps as inputs which are weighted combinations of 10 shot gathers in our experiment. One can easily replace the initial convolution layer with other feature generators and then only fine-tune the feature generator using transfer learning strategy when the number of shot gathers is not 10.

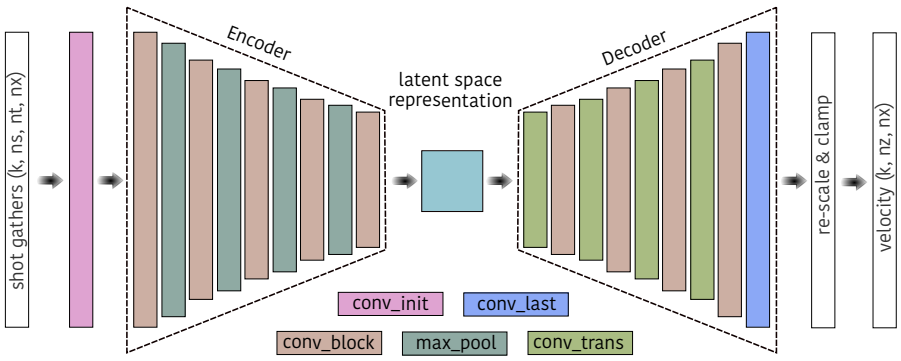

FIG. 3. The architecture of our proposed encoder-decoder convolution neural network (CNN), where k represents the batch size, n_s represents number of shot gathers, n_t represents the number of samples in time, n_z and n_x represents the number of pixels in vertical and horizontal, respectively. The structures of $conv\_block$ and $conv\_last$ are illustrated in Figure 4 and detailed parameters are contained in Table 1.

After the initial convolution, the generated feature maps are fed into the encoder partition, which is comprised of five convolutional blocks (plotted as the brown columns in

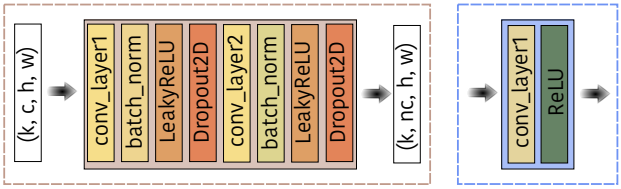

FIG. 4. Left: the detailed structure of convolutional block ($conv\_block$) shown as the brown columns in Figure 3, where (k, c/nc, h, w) denotes (batch size, input/output channel, height, width), respectively. For encoder nc=$2 * c$ and for decoder nc=$c//2$. Right: the structure of the last convolutional combination ($conv\_last$) in decoder plotted as the blue column in Figure 3.

Figure 3) and four maxpooling layers (plotted as the dark-blue columns in Figure 3). These convolutional blocks share the same structure, including two convolution layers, two batch normalization layers, two activation layers using a leaky rectifier unit (LeakyReLU) with a constant negative leaking slope 0.1, and two spatial dropout layers with a constant probability of retention ($p = 0.8$), as illustrated in the left panel of Figure 4. In the encoder, the number of output channels (nc) generated by a convolutional block is twice that of its input channels (c), i.e., $nc = 2c$, while the feature map size remains. The sequence of convolutional blocks and maxpooling layers implies that, by passing through the encoder, the dimension of the input is significantly reduced, but the output is richer because of the increased number of channels. The encoder thus transforms the high-dimensional input feature maps into low-dimensional, detailed latent space representations.

Next, these low-dimensional feature maps in latent space are used as input for the decoder, which is a sequential list of four transposed convolution layers (plotted as the dark-green columns in Figure 3) with four cross-interacting convolution blocks, following by a final convolution combination (plotted as the light-blue column in Figure 3). In the decoder, the output channel from a convolution block is half of the input channel (i.e., $nc = c//2$), but the same size remains in its other dimensions. The last convolution combination contains a single convolution layer with one filter of size $1 \times 1 \times 32$, followed by a ReLU layer, as illustrated in the right panel of Figure 4.

It is found that scaling labeled data and the output of the network into a small range may accelerate the convergence process, especially with the *Adam* optimizer which provides 'normalized' (i.e., the ratio of the first and the second momentum) parameter updates, since all trainable parameters are randomly initialized in range of $[0, 1]$. Therefore, in experiments of this paper, the ground-truth velocity models are scaled using the global minimum and maximum of velocity values $\mathbf{m}_{min}$ and $\mathbf{m}_{max}$, i.e., $(\mathbf{m} - \mathbf{m}_{min})/(\mathbf{m}_{max} - \mathbf{m}_{min})$. The model misfit is calculated between the scaled ground-truth and the network output using equation 8. To obtain the estimated velocity model, a reverse scaling procedure is required to assign each pixel of output into to a velocity value, i.e., $\tilde{\mathbf{m}} = \tilde{\mathbf{m}} * (\mathbf{m}_{max} - \mathbf{m}_{min}) + \mathbf{m}_{min}$. Finally, a clamping step (shown in bottom of Table 1) is performed to reduce the uncertainty of the estimated velocity models and to stabilized the following physics-based wave propagation. Detailed descriptions of layer parameters are contained in Table 1.

Operation Layer		Number of Filters	Size of Each Filter	Stride	Padding	Output Size (nc x h x w)
Input Data	shot gathers	-	-	-	-	10x400x100
Initial Convolution	conv_init	16	1x1x10	1x1	0	16x400x100
Encoder	conv_block	32	3x3x16	1x1	1x1	32x400x100
		32	3x3x32	1x1	1x1	32x400x100
	max_pool	1	4x2	4x2	2x0	32x101x50
	conv_block	64	3x3x32	1x1	1x1	64x101x50
		64	3x3x64	1x1	1x1	64x101x50
	max_pool	1	3x2	2x2	1x0	64x51x25
	conv_block	128	3x3x64	1x1	1x1	128x51x25
		128	3x3x64	1x1	1x1	128x51x25
	max_pool	1	4x2	4x2	0x0	128x12x12
	conv_block	256	3x3x128	1x1	1x1	256x12x12
		256	3x3x128	1x1	1x1	256x12x12
	max_pool	1	2x2	2x2	0x0	256x6x6
	conv_block	512	3x3x256	1x1	1x1	512x6x6
		512	3x3x512	1x1	1x1	512x6x6
Decoder	conv_trans	512	2x2x512	2x2	0x0	512x12x12
	conv_block	256	3x3x512	1x1	1x1	256x12x12
		256	3x3x256	1x1	1x1	256x12x12
	conv_trans	256	3x3x256	2x2	0x0	256x25x25
	conv_block	128	3x3x256	1x1	1x1	128x25x25
		128	3x3x128	1x1	1x1	128x25x25
	conv_trans	128	2x2x128	2x2	0x0	128x50x50
	conv_block	64	3x3x128	1x1	1x1	64x50x50
		64	3x3x64	1x1	1x1	64x50x50
	conv_trans	64	2x2x64	2x2	0x0	64x100x100
	conv_block	32	3x3x64	1x1	1x1	32x100x100
		32	3x3x32	1x1	1x1	32x100x100
	conv_last	1	1x1x32	1x1	0x0	1x100x100
re-scale & clamp	re-scale	$\tilde{\mathbf{m}} = \tilde{\mathbf{m}} * (\mathbf{m}_{max} - \mathbf{m}_{min}) + \mathbf{m}_{min}$				
	clamp	$\tilde{\mathbf{m}} = \begin{cases} \mathbf{m}_{min} & \text{if } \tilde{\mathbf{m}} \leq \mathbf{m}_{min} \\ \tilde{\mathbf{m}} & \text{otherwise} \\ \mathbf{m}_{max} & \text{if } \tilde{\mathbf{m}} \geq \mathbf{m}_{max} \end{cases}$				

Table 1. Detailed parameters of our proposed CNN with architecture shown in Figure 3.

NUMERICAL EXAMPLES

In this section, we carry out a simulated velocity building exercise for a subsurface salt body using the basic network architecture set out in the previous section. The implementation of the physics-based RNN inversion are not performed independently in this paper since we aim to build an end-to-end velocity reconstruction framework. Instead, it is treated as a partition of the PGNN or the H-PGNN to provide gradient contribution from data discrepancies and details of implementing the physics-based RNN inversion are elaborated in the previous work of Sun et al. (2020). In the following contexts, we will investigate the possible strategy of training the proposed encoder-decoder CNN in both physics-guided ($\lambda_m = 0$ and $\lambda_d = 1$) and hybrid-guided ($\lambda_m \neq 0$ and $\lambda_d = 40$) modes, and discuss the influence of two trade-off parameters. Additionally, the fully data-driven ($\lambda_m = 1$ and $\lambda_d = 0$) training will also be implemented. The goal of training a network in a fully data-driven mode is to create a benchmark deep learning solution. This will both highlight the potential strengths of a deep learning technique, but also to highlight some practical challenges. To be consistent with traditional deep learning, a fully data-driven trained network will be simply referred to as CNN in later contexts.

Data preparation

Optimizing the network illustrated in Figure 3 in a fully data-driven mode requires a very large number of training samples. Here, $16,000$ synthetic velocity models with a grid size of 101×101 and 10m cells are created. Each model contains a single salt body with an arbitrary shape and location. The salt velocity is fixed at 4500m/s. The backgrounds are random realizations of 5-to-12 laterally-distorted layers with velocity values ranging from 1500-4000m/s. These synthetic velocity models are divided into the training (10,000 samples), validation (3,000 samples), and test (3,000 samples) datasets. Ten representative examples of synthetic velocity models are illustrated in Figure 5 .

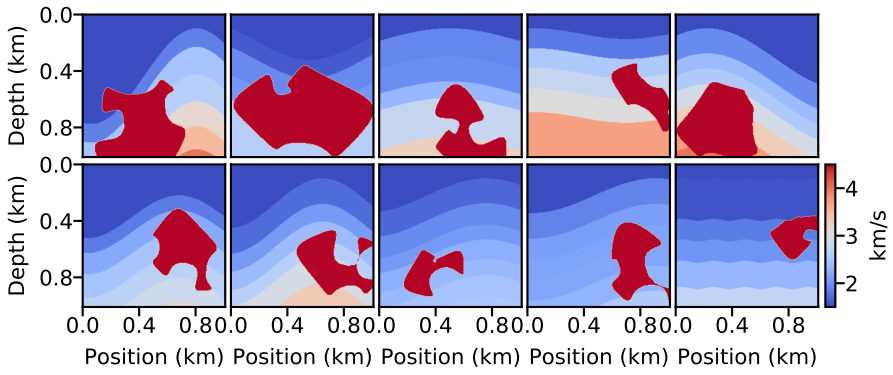

FIG. 5. Ten representative examples of synthetic velocity models.

To generate the corresponding shot gathers as inputs of the proposed CNN, we adopt a second-order, both in spatial and time dimensions, finite difference (Smith, 1985) operator for the forward wave propagation simulation using physics-based RNN. For each velocity model, ten shot gathers are collected with shot interval 100m. All ten sources are located at the surface starting from 50m and ending at 950m (from left to right). A minimum phase

wavelet with 12Hz dominant frequency (shown in Figure 6) and a simple sponge absorbing boundary condition are used for the synthetic seismic record collection. The time sample interval during wave propagation is 1.5ms and 800 time samples are recorded. However, to reduce memory use and to fit the input size of our designed CNN, we further downsample all shot gathers into 400 time samples with 3ms interval. In Figure 7, one representative input example containing ten shot gathers using the upper-left velocity model in Figure 5 are plotted. The ten raw shot gathers for each velocity model are treated as distinct input channels in the proposed CNN; no preprocessing is applied.

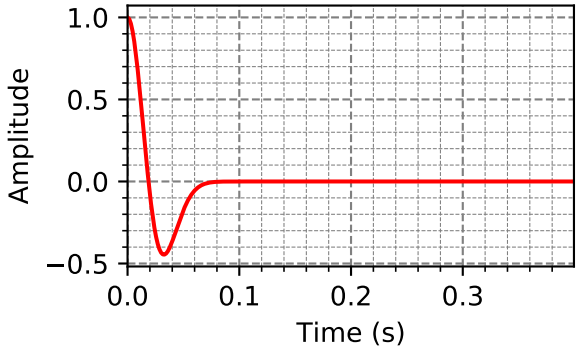

FIG. 6. A minimum phase wavelet adopted for synthetic data generation.

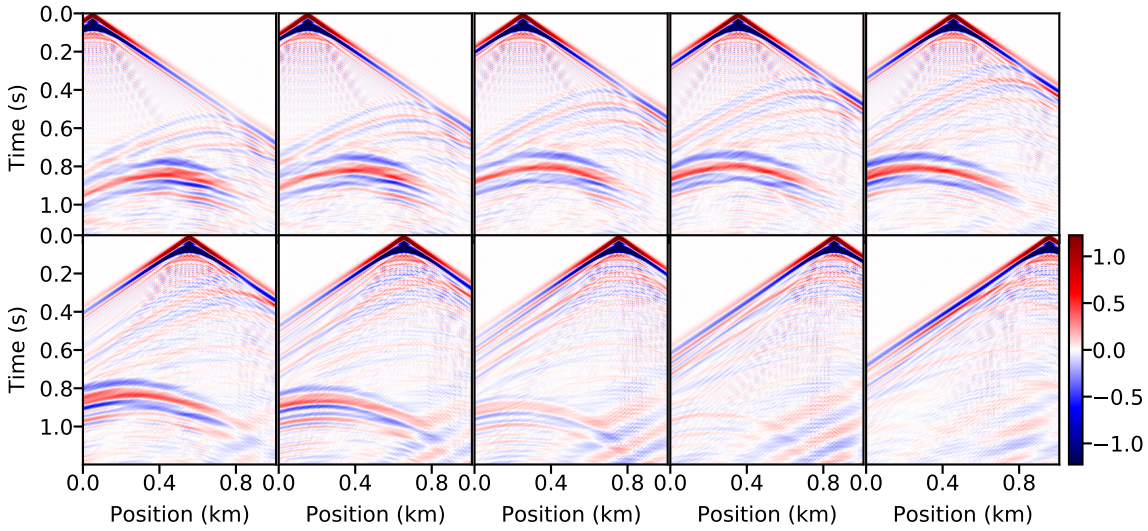

FIG. 7. One representative input example containing ten shot gathers using the velocity model shown in the upper-left corner in Figure 5.

As discussed in previous section, to accelerate the convergence process during training, all velocity models are scaled into the range $[0, 1]$ using the global minimum and maximum values: $\mathbf{m} = (\mathbf{m} - \mathbf{m}_{min})/(\mathbf{m}_{max} - \mathbf{m}_{min})$ where $\mathbf{m}_{min} = 1500$m/s and $\mathbf{m}_{min} = 4500$m/s. The model misfit is calculated between the scaled velocity profiles and the output of the decoder network using equation 8; the data residual is calculated between the ground-truth and estimated shot gathers using equation 2.

Parameters / Mode	Number of samples in Train (Val)	Batch-size in Train (Val)	Trade-off (λ_m, λ_d)	($\mathcal{L}_m^T$, $\mathcal{L}_m^V$) ($\times 10^{-1}$)	($\mathcal{L}_d^T$, $\mathcal{L}_d^V$) ($\times 10^{-4}$)
CNN	320 (120)	16 (64)	(1, 0)	(0.42, 0.37)	(1.48, 1.28)
PGNN	320 (120)	16 (32)	(0, 1)	(1.97, 1.98)	(1.76, 0.84)
H-PGNN$^{\frac{0.1}{40}}$	320 (120)	16 (32)	(0.1, 40)	(1.96, 1.98)	(1.71, 0.84)
H-PGNN$^{\frac{1}{40}}$	320 (120)	16 (32)	(1, 40)	(2.03, 2.80)	(0.89, 0.76)
H-PGNN$^{\frac{10}{40}}$	320 (120)	16 (32)	(10, 40)	(2.04, 3.29)	(1.12, 1.13)

Table 2. Parameter settings in different training modes for hyperparameter selection, where $\mathcal{L}_m^T$, $\mathcal{L}_m^V$, $\mathcal{L}_d^T$, and $\mathcal{L}_d^V$ columns represent the smallest model misfit and data residual on training and validation sets within 30 epochs training.

Hyperparameter selection

All training processes are carried out on 10 NVIDIA Telsa P100-PCIe GPU accelerators with 12GB memory using the *Adam* optimizer with the learning rate 0.01. Due to the computational burden of implementing PGNN and H-PGNN, only a small portion of training (320 samples) and validation (120 samples) sets are extracted from the intact training ($10,000$ samples) and validation ($3,000$ samples) datasets, respectively, and utilized to analyze the hyperparameter selection. Though training the proposed CNN with such a small quantity of samples may not be able to achieve its best performance and be readily overfitted, it provides us an insight of the training behavior under the selected combination of hyperparameters.

To investigate the influences of both model misfit and data residual to training, we select three different combinations for trade-off parameters (λ_m, λ_d), including (0.1, 40), (1, 40), and (10, 40). The proposed network is trained in H-PGNN mode with three selected combinations of trade-off parameters, respectively. In addition, the CNN and PGNN training modes are also employed to train the network as two benchmarks. Both model misfit and data residual on training and validation sets are monitored for all training modes at every epoch. The training processes are either stopped by reaching the maximum epoch 30 or early manually terminated when the overfitting phenomenon occurs. The parameter settings for each training mode and their best achieved model misfits and data residuals on training and validation datasets are illustrated in Table 2. Besides that, a truncated time length of 50, which is the maximum length the computational memory of our equipment allows for, is chosen for the truncated RNN backpropagation in the PGNN and H-PGNN modes.

All loss variations of both model misfits and data residuals are plotted in Figure 8, where the solid lines represent losses' behaviors on training dataset and the dashed lines delineate losses' variations on validation dataset. The blue lines represent the variations of model misfits and data residuals using a network trained in the CNN mode. With a small amount of training examples, we observe that the behavior of the model misfit on the validation dataset is not inconsistent with the decreasing trend of the training model misfit. This implies that, the convergence direction in a fully data-driven training may easily manipulated by the diversity of the training dataset. In addition, the inconsistent trends between model misfits and data residuals are also detected in the CNN mode, which

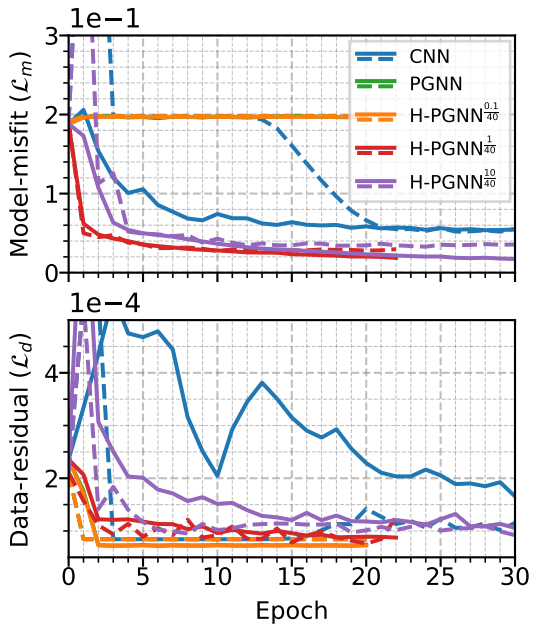

FIG. 8. Statistical comparisons of model misfits and data residuals in different training modes with varying trade-off parameters, where the solid lines represent losses' behaviors on training dataset and the dashed lines delineate losses' variations on validation dataset.

indicate the convergence process in a fully data-driven mode does not obeys physics rules.

In both PGNN and H-PGNN$^{\frac{0.1}{40}}$ (i.e., $\lambda_m = 0.1$, $\lambda_d = 40$) modes (shown in green and orange in Figure 8), due to a lack of accurate initial models, the network failed to escape from a local minimum of data residual. To meet the requirement of good starting points by PGNN/H-PGNN, we need to increase the contribution from model misfit by increasing λ_m, while λ_d remains to be consistent to 40. The red lines in Figure 8 demonstrate the losses' variations in the H-PGNN$^{1/40}$ (i.e., $\lambda_m = 1$, $\lambda_d = 40$) mode. The result shows that both model misfit and data residual are smoothly reduced, and the variation tendency on training and validation datasets are also well matched. However, a slight overfitting phenomenon is observed around 15 epochs in the model misfit recording. As increasing λ_m to 10 (purple lines in Figure 8), both model misfit and data residual are still able to be reduced but with more fluctuations. Besides, the overfitting issue also occurs around 15 epoch in the model misfit variation with the setting of $\lambda_m = 10$.

Therefore, from what we have examined, the combination of ($\lambda_m = 1$, $\lambda_d = 40$) seems to be a suitable choice as the initial setting for H-PGNN training in later experiments. To mitigate the overfitting issue, instead of a fixed setting, λ_m will be reset from 1 to 0.1 once the model misfit have been reduced by 80%. Here, the 80% is found by trial and error to ensure accurate initial models for H-PGNN while not quickly overfitting the model misfit. Beyond that, we reset λ_m to be 0.1 instead of zero is to prevent the convergence falling into a local minimum mislead by data residual. With a varying λ_m, the H-PGNN allows the model misfit leading the training process during the first few epochs and expects the data residual (i.e., physics rules) dominating the rest convergences.

Parameters / Mode	Number of samples in Train (Val, Test)	Batch-size in Train (Val, Test)	Trade-off (λ_m, λ_d)	Training time-cost (per epoch 10GPUs)
CNN-10k	10000 (3000, 3000)	32 (64, 64)	(1, 0)	2.58mins
CNN-2k	2000 (600, 3000)	32 (64, 64)	(1, 0)	0.75mins
H-PGNN-2k	2000 (600, 3000)	16 (32, 64)	(1→0.1, 40)	20.3mins

Table 3. Parameter settings for different training modes, where the training time-cost is the computational time of computing a single epoch includes one training and one validation processes.

Training and validation

Since the PGNN does not converge without accurate initial models, in the following experiences, only CNN and H-PGNN ($\lambda_m : 1 \rightarrow 0.1$, $\lambda_d = 40$) will be performed. To fully explore the benefits of the H-PGNN, we select a portion set including $2,000$ examples from the intact training dataset ($10,000$ examples) to train the network in the H-PGNN mode, namely H-PGNN-2k in Table 3. Similarly, 600 validation examples are randomly selected from the intact validation dataset ($3,000$ examples) to monitor the overfitting phenomenon during training. The same selected training and validation portions are also used to train the network in a fully data-driven mode, namely CNN-2k in Table 3. Furthermore, the CNN is also trained using the intact training and validation datasets as a comparison, namely CNN-10k in Table 3. Number of samples in training and validation datasets and batch sizes for CNN-2k, CNN-10k, H-PGNN-2k are summarized in Table 3. The validation process is performed after every training epoch. In each epoch, the model misfits are computed for training and validation datasets in CNN-2k, CNN-2k, and H-PGNN-2k, respectively. However, the data residuals are only calculated and monitored in CNN-2k and H-PGNN-2k because of the computational burden. The computational time cost for a single epoch in each mode is shown in Table 3. Note that, for CNN-2k, the time cost of computing data residuals is not taken into account. Finally, the same truncated time length of 50 is used for H-PGNN-2k training.

Both CNN-2k and CNN-10k trainings are stopped after 100 epochs. In Figure 9, we plot the monitored model misfits and the data residuals using CNN-2k, CNN-10k, and H-PGNN-2k, respectively, where the solid lines represent the training losses and the dashed lines delineate the validation losses. As mentioned previously, no data residual is monitored for CNN-10k due to the computational burden. The model misfits using CNN-2k are shown in blue lines in the top panel of Figure 9. We observe that training and validation model misfits using CNN-2k are comparably decreased and tend to be flatten after 100 epochs given $2,000$ training examples. However, the decreasing trend of model misfit using CNN-2k does not reflect on the data residual. In the bottom panel of Figure 9, the training and validation data residuals by CNN-2k reaches its minimum at the second epoch, and fall into a local minimum after that. This further confirms that the model misfit objective function only provides a pixel-wise optimization, and physics constraints are not being invoked. The results also indicate that our current training dataset ($2,000$ examples) are not enough to achieve the best performance of the proposed architecture. Namely, the fully data-driven training requires a large amount of labeled examples. This deduction is confirmed by the model misfit variation using CNN-10k. As plotted in green lines in the top panel of Figure 9, by simply increasing the number of training examples, the CNN-10k is able to smoothly converge to a much smaller error.

Finally, the proposed network is also trained in the H-PGNN-2k mode using the same training and validation dataset as used in the CNN-2k, and the training is stopped after 67 epochs. Following the discussion of trade-off parameters selection, a constant value (40) is chosen for λ_d, while the λ_m is initially set to 1 and then reset to 0.1 after the model misfit is reduced by 80% (referred to the third epoch in this case). In other words, after the third epoch, we expect the starting of an approximate physics-driven optimization stage. We point out that, although the model misfit dominates starting point acts something like an initial velocity model building stage, and the data-misfit ending point in isolation acts something like an FWI regimen, the mid-range iterations, in which both act simultaneously, is unique to this hybrid approach.

The recorded model misfits and data residuals using H-PGNN-2k on both training and validation datasets at each epoch are plotted in orange lines in Figure 9. Given the same training dataset, the model misfit using H-PGNN-2k converges much faster than CNN-2k, and a much smaller error is achieved. In addition, the convergence of data residual using H-PGNN-2k are much smoother than the one by CNN-2k. Comparing to CNN-10k, a comparable level of model misfit is obtained by H-PGNN-2k with slightly faster convergence. The results demonstrate that incorporating physical knowledge may reduce the strict requirement of big data by the traditional deep learning and gain a stable convergence process, which is essential in geophysical inversion problem.

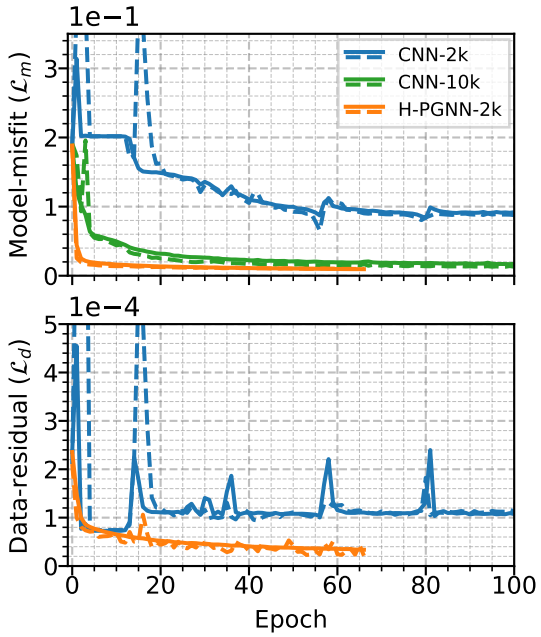

FIG. 9. Statistical comparisons of model misfits and data residuals in CNN-2k, CNN-10k, and H-PGNN-2k modes, where the solid (dashed) lines represent losses' variations on training (validation) dataset.

To intuitively examine the performance of networks trained in different modes, five velocity models are extracted from the test dataset as representative examples of velocity building using the CNN-2k, CNN-10k, and H-PGNN-2k, respectively, given raw shot gathers as input. In Figure 10, five ground-truth velocity models are plotted in the first row, and the associated velocity predictions using CNN-2k, CNN-10k, and H-PGNN-2k

are plotted from the second row to the bottom. Generally, approximate locations of the salt bodies and the inside velocity values are predicted correctly by the CNN-2k. However, CNN-2k failed to predict the shape of the salt body and the background information. With increased training samples, the salt body boundaries have been reasonably approximated by CNN-10k, with the background trends and model character largely intact. We conclude that the CNN-10k, trained by the pure model misfit objective function with $10,000$ training examples, has the capacity to capture some of the property and positioning information contained in the raw shot gathers. The estimated velocity models using H-PGNN-2k are illustrated in the bottom row of Figure 9. Compared to the results derived from CNN-2k and CNN-10k, the salt bodies predicted by H-PGNN-2k have more highly resolved shapes, and clearly defined boundaries, reduced smoothing. The results also exhibit better definition within background irregular layers, and in facts resolves some of interfaces, which was not achieved with CNNs. In Figure 11 the estimated model misfits using CNN-2K, CNN-10k, and H-PGNN-2k respectively are plotted in different rows. We observe that, given the same training dataset, involving the physics-based objective function significantly improves the accuracy of estimated velocity models.

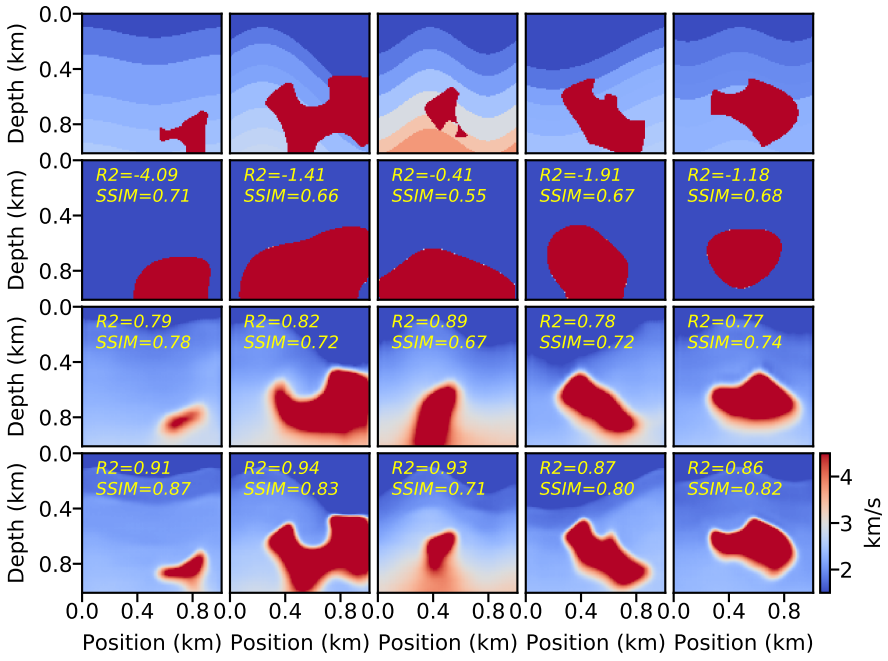

FIG. 10. Comparison of the ground-truth (the first row) and the estimated velocity models using CNN-2k, CNN-10k, and H-PGNN-2k (from the second row to the bottom), respectively.

Furthermore, to investigate the physical characteristics of the estimated velocities, the forward wave propagation is implemented using all velocity models shown in Figure 10 with shot locations placed on the surface at the middle of the model. The corresponding shot gathers are plotted in Figure 12, from top to bottom, associated with the ground-truth, CNN-2k, CNN-10k, and H-PGNN-2k, respectively. It is noticeable that the simulated shot gathers using estimated velocities by CNN-10k and H-PGNN-2k are better matched with the ground truth shot gathers than these shot profiles related to CNN-2k, which only capture the direct waves and partial primary events reflected by the top boundaries of the salt body. The data residuals of the middle shot gathers are also plotted in Figure 13, alongside the

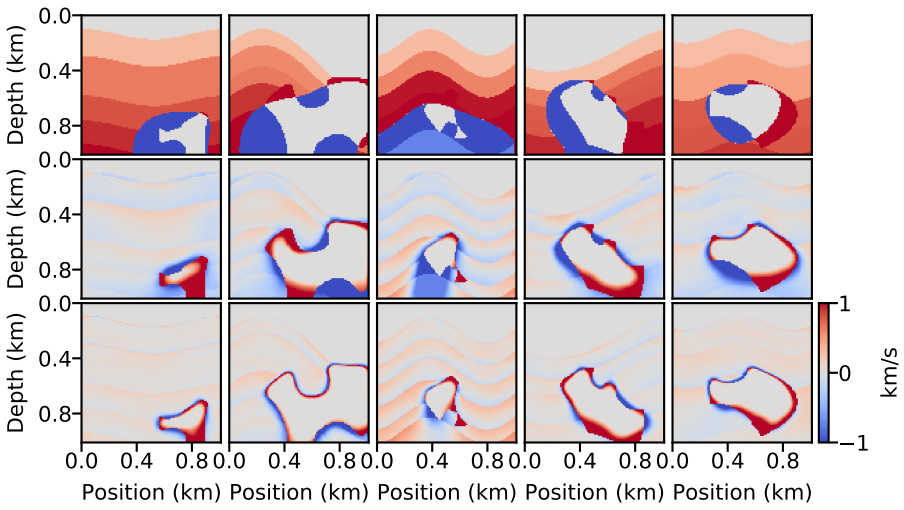

FIG. 11. The model prediction errors using CNN-2k, CNN-10k, and H-PGNN-2k (from top to bottom), respectively.

final estimated velocity models by CNN-2k, CNN-10k, and H-PGNN-2k. The significantly reduced discrepancies of data residuals are observed in profiles using H-PGNN-2k.

DISCUSSION

It appears from our analysis that the data-driven training requires a vast quantity of labeled examples to accurately extract the main subsurface parameter information influencing the waveforms observed in surface seismic data. Physics-guided network with hybrid training appears to be able to supply a significant jump in accuracy and resolution under the same training distribution, which our experiments indicate would only be possible in a data-driven scheme through a large inflation of the training data. Further quantitative comparisons between the data-driven and the physics-guiding are required to completely flesh out the trade-offs between the two elements of the hybrid approach, and how a physics-based objective function operates within the optimization of an otherwise "traditional" neural network. In this section, we discuss the difference between data-driven and hybrid training in three different aspects, including the quantitative error analysis on the entire test dataset, the investigation of network's generalization ability using SEG salt models, and the uncertainty analysis of network's estimations.

Quantitative error analysis

We can begin this process relatively simply. We carry out the velocity building process on the entire test dataset, which comprises 3000 velocity models, using CNN-2k, CNN-10k, and H-PGNN-2k, respectively. After acquiring the estimated velocity models, the absolute value of model misfits are computed and forward wave propagation is performed to calculate the absolute data residuals. In addition, two other metrics, structural similarity index measure (i.e., SSIM, Wang et al., 2004) and coefficient of determination (R2-score) are adopted to provide a general idea of the prediction accuracy. The absolute model misfit, the absolute data residual, the SSIM, and the R2-score are referred to as four metrics, and

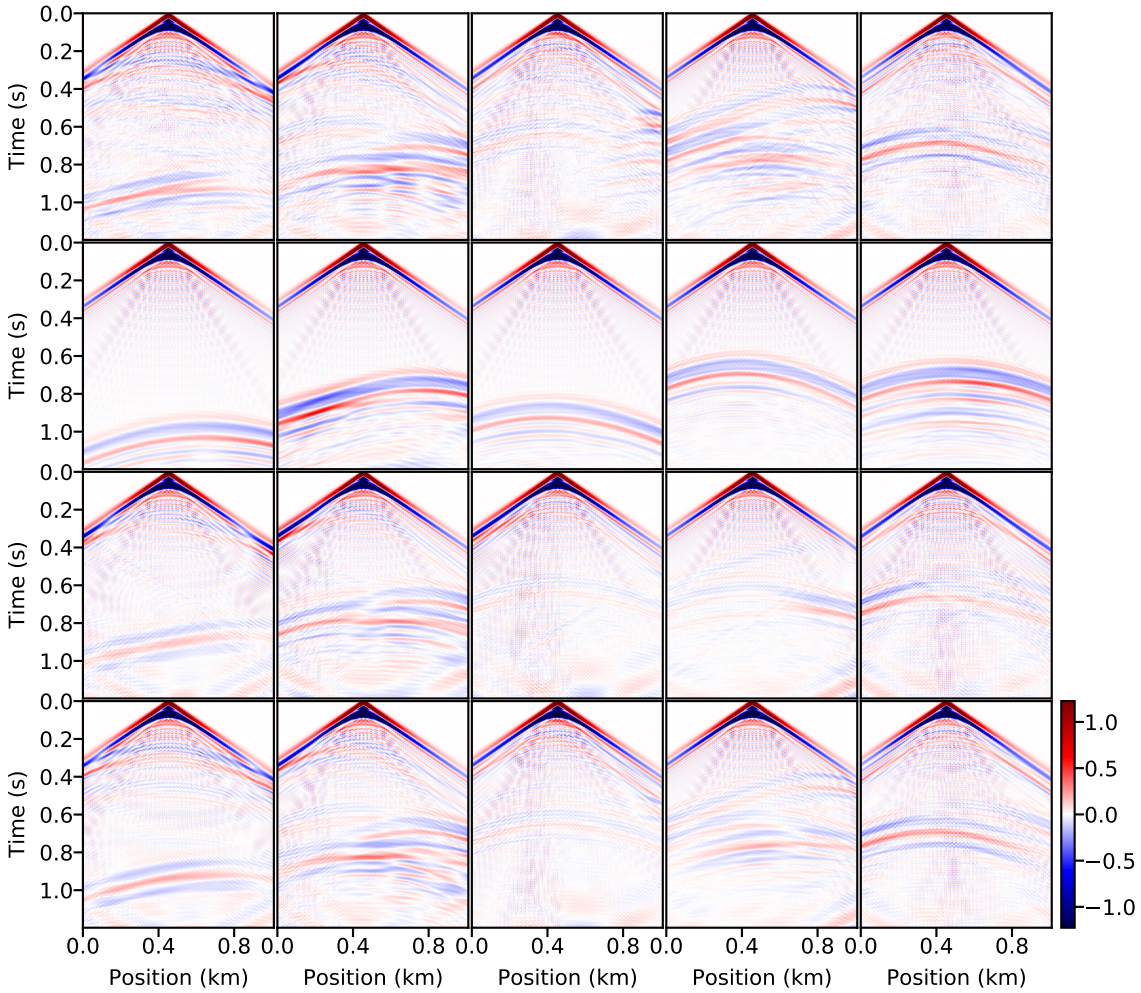

FIG. 12. Comparison of the middle-located shot gathers using velocity models shown in Figure 10. From top to bottom, they are related to ground-truth, CNN-2k, CNN-10k, and H-PGNN-2k, respectively.

prediction errors of model and data are computed in percentage.

We extract $1,000$ velocity models from the test dataset and their associated four metrics using CNN-2k, CNN-10k, and H-PGNN-2k, respectively, are plotted in the top row of Figure 14. We observe that the absolute model misfits using CNN-2k are significantly large spreading from 15% to 35%, while the pixel-wise accuracy of velocity models estimated by CNN-10k and H-PGNN-2k maintain in a high standard, i.e., for most of cases, their model misfits are less than 10%. From the perspective of the data residual, noticeable advantages are observed in the network trained by the H-PGNN-2k comparing to CNNs. Similar patterns are also found in SSIM-R2-score plots.

Finally, the mean and standard deviation of four metrics are calculated on the entire test dataset. The results are plotted in the bottom row of Figure 14. Four metrics associated with CNN-2k, CNN-10k, and H-PGNN-2k are plotted in blue, orange, and green ellipses, respectively, where the centers of the ellipses are associated with the averages of four met-

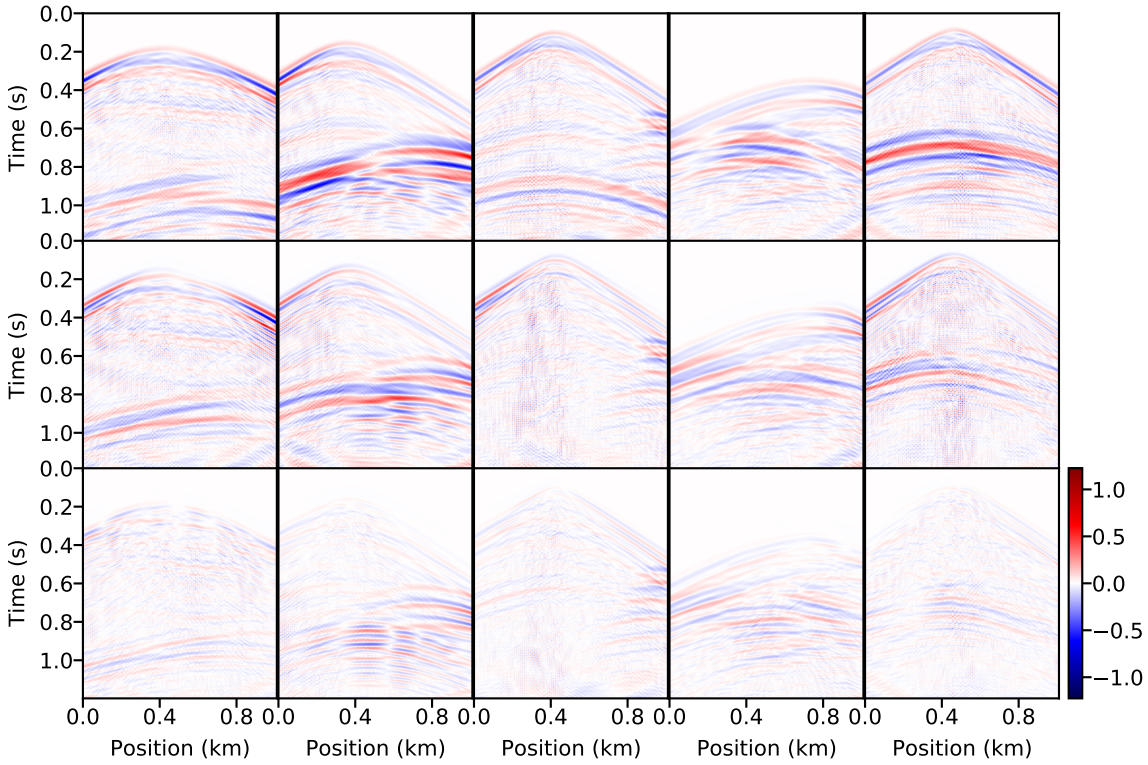

FIG. 13. The data prediction errors of the middle-located shot gathers using the estimated velocity model by CNN-2k, CNN-10k, and H-PGNN-2k (from top to bottom), respectively.

rics over the entire validation dataset. The heights (width) of the ellipses represent twice the standard deviations of four metrics. The network's performances of using CNN-2k, CNN-10k, and H-PGNN-2k are distinguishable. The comparison of the blue (CNN-2k) and the orange (CNN-10k) ellipses furthermore affirms that enlarging the diversity and distribution of the training dataset may gain a better performance. In the bottom-left panel of Figure 14, comparing to the blue ellipse associated with CNN-2k, a smaller standard deviation of model misfits using CNN-10k (i.e., the height of the orange ellipse) indicates a large and diverse training ensembles may enhance the network's robustness with an increased stability for handling daedal examples. However, it is worth mentioning that the standard deviation of the data residuals using CNN-10k is comparable to that of using CNN-2k ($\sim 7\%$), though the mean of the data residuals are highly reduced. This implies, in a fully data-driven mode, enlarging the diversity and the number of training examples does not ensures the network's prediction obeys the physical rules.

However, by involving a physics constraint in the objective function as H-PGNN-2k, prominent advantages are detected in Figure 14. Comparable level of model misfits using CNN-10k and H-PGNN-2k shows that the training requirement of big data may be considerably reduced by incorporating known physics knowledge, which may also improves the data residuals to some extent. In addition, smaller standard deviations of both model misfits and data residuals demonstrate the H-PGNN-2k has a better generalization ability of handling distinguishable cases. In the bottom-right panel, a higher SSIM using H-PGNN-2k indicates the H-PGNN-2k are able to estimate the subsurface structure with preciser reso-

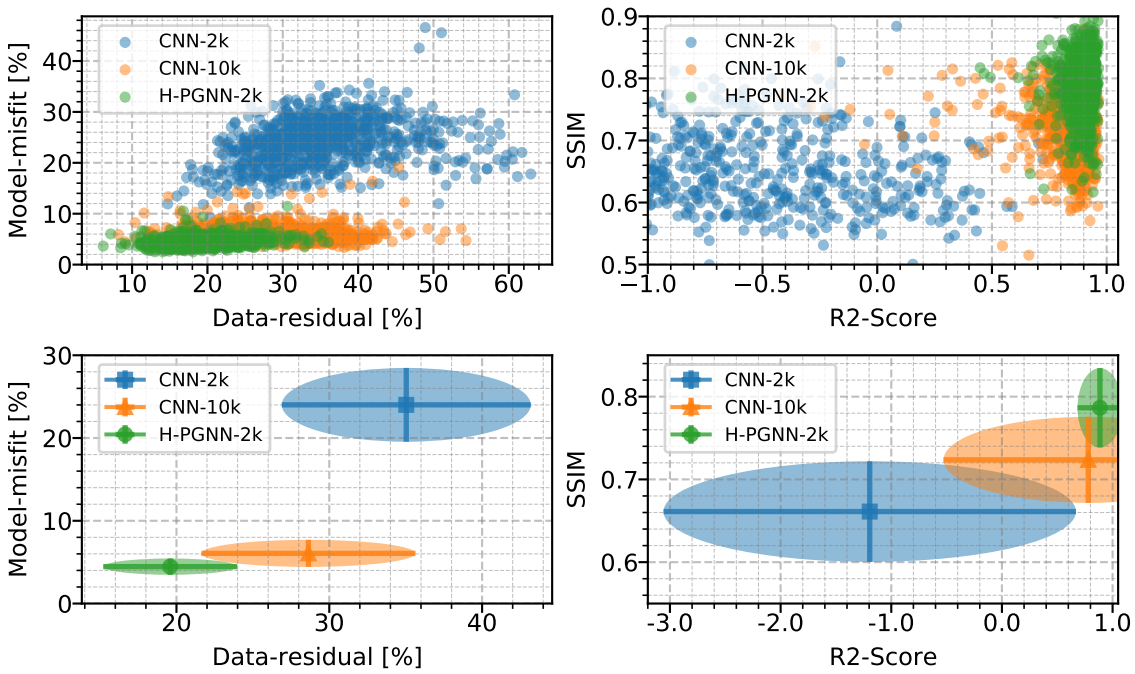

FIG. 14. The comparison of the absolute model misfits, the absolute data residuals, SSIM, and R2-score on the entire test dataset using CNN-2k, CNN-10k, and H-PGNN-2k, respectively. Top: four metrics' distributions of $1,000$ representative velocity models extracted from the test dataset; Bottom: the mean and standard deviations of four metrics on the entire test dataset, where the centers of the ellipses associate with the mean values of four metrics, and the height (width) delineates twice the standard deviations of four metrics, respectively.

lution than the results by CNN-2k and CNN-10k. A smaller standard deviation of R2-score by H-PGNN-2k demonstrates the higher correlation between the estimated velocity profiles and the ground-truth models, which implies the stability of the H-PGNN-2k.

Generalization ability

In order to examine the ability of a trained network to be applied in more general instances, we employ the 3D SEG salt model which has never been seen in the intact training dataset. Ten 2D velocity models are extracted from the 3D SEG salt volume, and plotted in the first row of Figure 15. Next, ten shot gathers placed on the surface with the same geometry setting shown in previous section are generated using each velocity profile. By feeding these raw shot gathers into the trained networks, the velocity building for ten extracted SEG salt models are implemented using CNN-2k, CNN-10k, and H-PGNN-2k, respectively, in which their corresponding estimated velocities are plotted from the second row to the bottom of Figure 15. The estimated velocity models in the second row of Figure 15 demonstrate that the CNN-2k has the ability of predicting approximate locations of the salt body with imprecise boundaries and background information. Besides that, strong anomalies are also exposed in CNN-2k's predictions. On the other hand, CNN-10k and H-PGNN-2k made a step forward in casting the shape of the salt body and in providing more information of background layers. Additionally, anomalies are significantly reduced in estimated models associating with CNN-10k and H-PGNN-2k. Comparing to CNN-10k,

velocity model estimated by H-PGNN-2k have less smoothing effects on the boundaries of the salt body.

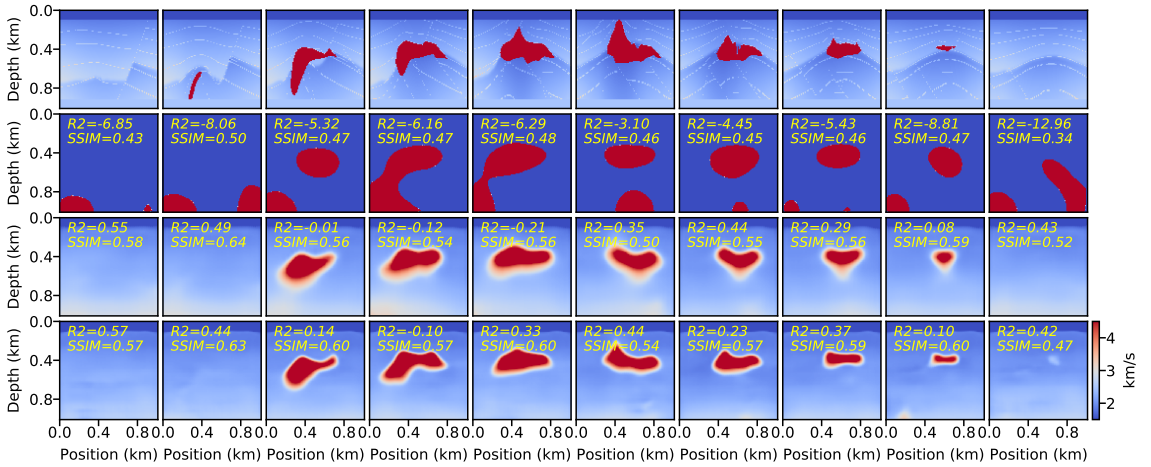

FIG. 15. Comparison of the ground-truth (the first row) and the estimated SEG salt models using CNN-2k, CNN-10k, and H-PGNN-2k (from the second row to the bottom) respectively.

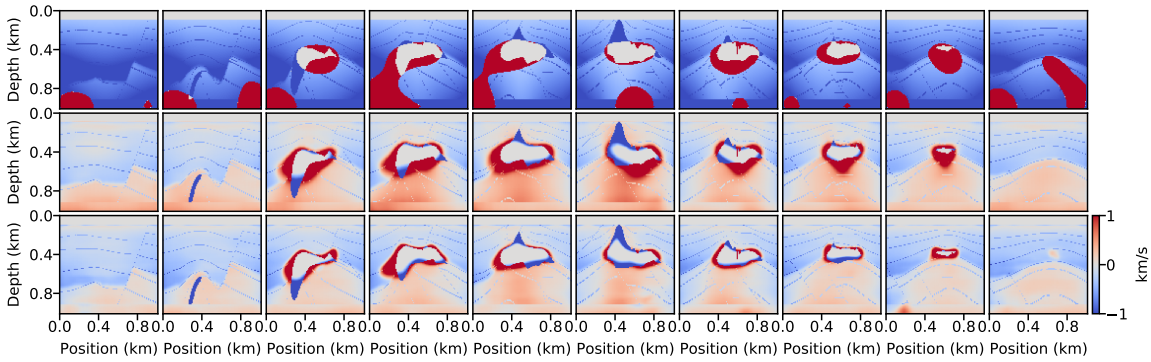

FIG. 16. The model misfits between ground-truth SEG salt models and estimated velocities using, from top to bottom, CNN-2k, CNN-10k, and H-PGNN-2k, respectively.

The model misfits between the ground-truth SEG salt models and estimated profiles are plotted in Figure 16, from top to bottom, associated with CNN-2k, CNN-10k, and H-PGNN-2k, respectively. The middle-located shot gathers using velocity models shown in Figure 15 are calculated and plotted in Figure 17 to examine the physical meaning of these estimations. The corresponding data residuals are illustrated in Figure 18 in the same layout.

Predictive uncertainty analysis

Unlike Bayesian statistics, which gives the distribution of parameters to delineate the underlying phenomenon between the desired target and our prior knowledge, the supervised deep neural network establishes a deterministic projection from the input to output determined by given training dataset, and therefore does not capture the prediction uncertainty. In order to measure the uncertainty of predicted results, we need to build the

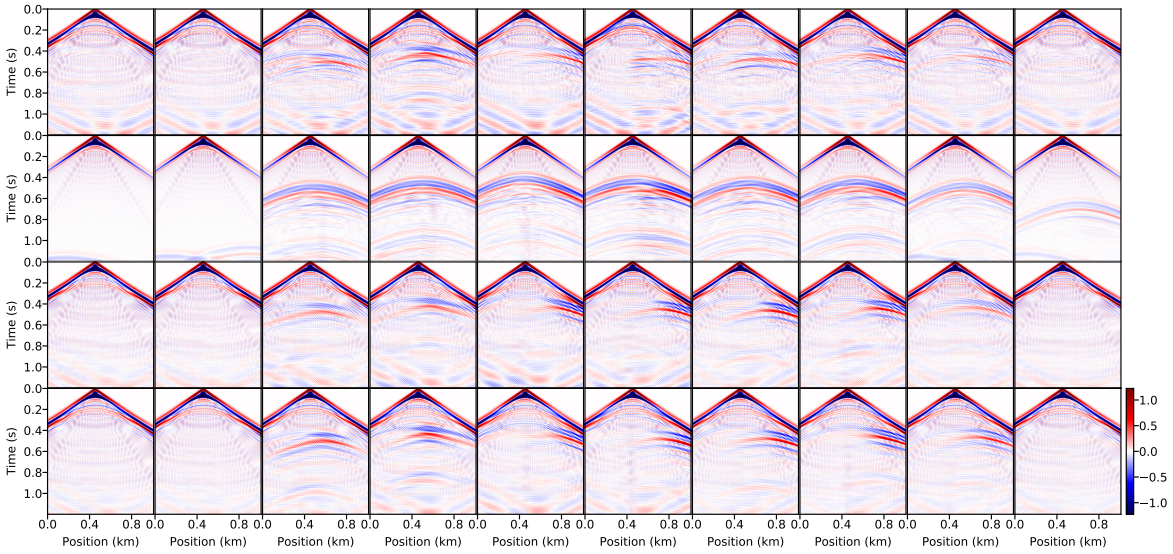

FIG. 17. Comparison of the middle-located shot gathers associated with velocity models shown in Figure 15. From top to bottom, they are related to ground-truth, CNN-2k, CNN-10k, and H-PGNN-2k, respectively.

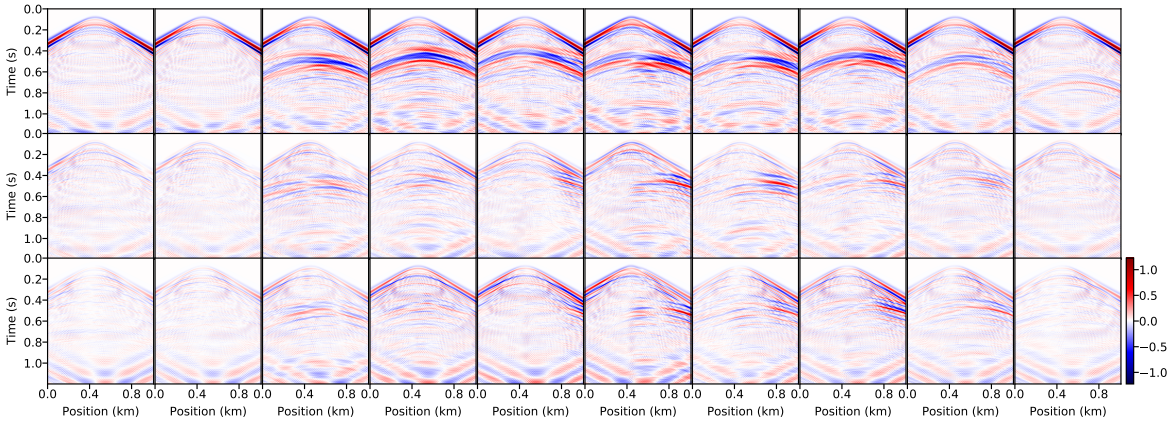

FIG. 18. The data prediction errors of the middle-located shot gathers using the estimated SEG-salt models by CNN-2k, CNN-10k, and H-PGNN-2k (from top to bottom) respectively.

distribution of the network's trainable parameters, instead of deterministic weights. Deep ensembling is a powerful technique where a large ensemble of networks are trained independently using respective datasets. Thus, a collective predictions can be obtained using trained network ensembles to measure the predictive uncertainty. However, training a large number of networks usually comes with a prohibitive computational cost.

Gal and Ghahramani (2016) introduced an approximate Bayesian interference by including dropout training and validating in deep neural networks. Intuitively, by randomly zeroes some elements of hidden feature maps with probability ($p = 0.2$ in our experiments) using samples from a Bernoulli distribution, those elements that are not dropped out define a new network. In other words, the training with dropout layer can be considered as training a large ensemble of different networks simultaneously. In addition, the mini-batch strategy ensures different training examples are utilized for training different networks. The key of

measuring predictive uncertainty using such a technique is activating dropout in both training and validation stages. During training, it ensures training a large ensemble of networks which Gal and Ghahramani (2016) demonstrated is an approximation of deep ensembling Bayesian model. In validation, by implementing the trained network with dropout multiple times, we can obtain a set of predictive realizations.

To evaluate the uncertainty of predicted results, we first run 100 realizations with dropout on the synthetic velocity model using CNN-2k, CNN-10k, and H-PGNN-2k, respectively. The mean and standard deviation of 100 generated realizations are calculated. The results are plotted in Figure 19, where, from left to right, they are associated with CNN-2k, CNN-10k, and H-PGNN-2k, respectively; from top to bottom, they are corresponding to the ground-truth, the mean of 100 predictive realizations, the model misfit between the ground-truth and the averaged realization, and the standard deviation of 100 predictive realizations. In the second and the third rows, the averaged predictions' precision using different networks are consistent with our previous observation, where the H-PGNN-2k outperforms CNN-2k and CNN-10k. In the last row of Figure 19, the standard deviations of 100 predictive realizations using CNN-10k and H-PGNN-2k are smaller than 300m/s at most of areas, except at the salt body boundary, which reveals the network's robustness and stability. Similarly, we compute the mean and standard deviation of the middle-located shot gathers using 100 predictive velocity models for different modes, and results are plotted in Figure 20 with the same layout of Figure 19.

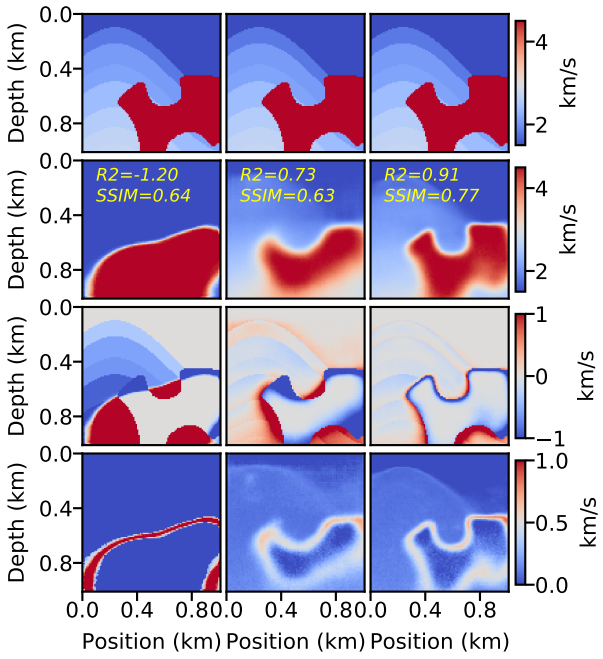

FIG. 19. The uncertainty and average of 100 realizations using, from left to right, CNN-2k, CNN-10k, and H-PGNN-2k, respectively. From top to bottom, plots are listed as: the ground-truth model, the mean of 100 realizations predicted using different trained networks, the model misfits between the ground-truth and averaged realization, and the standard derivation of 100 realizations.

Moreover, the uncertainty analysis using dropout on ten SEG salt models are also performed using H-PGNN-2k and 100 realizations for each velocity profile are generated, respectively. In Figure 21, the ground-truth and the mean of 100predictive realizations are

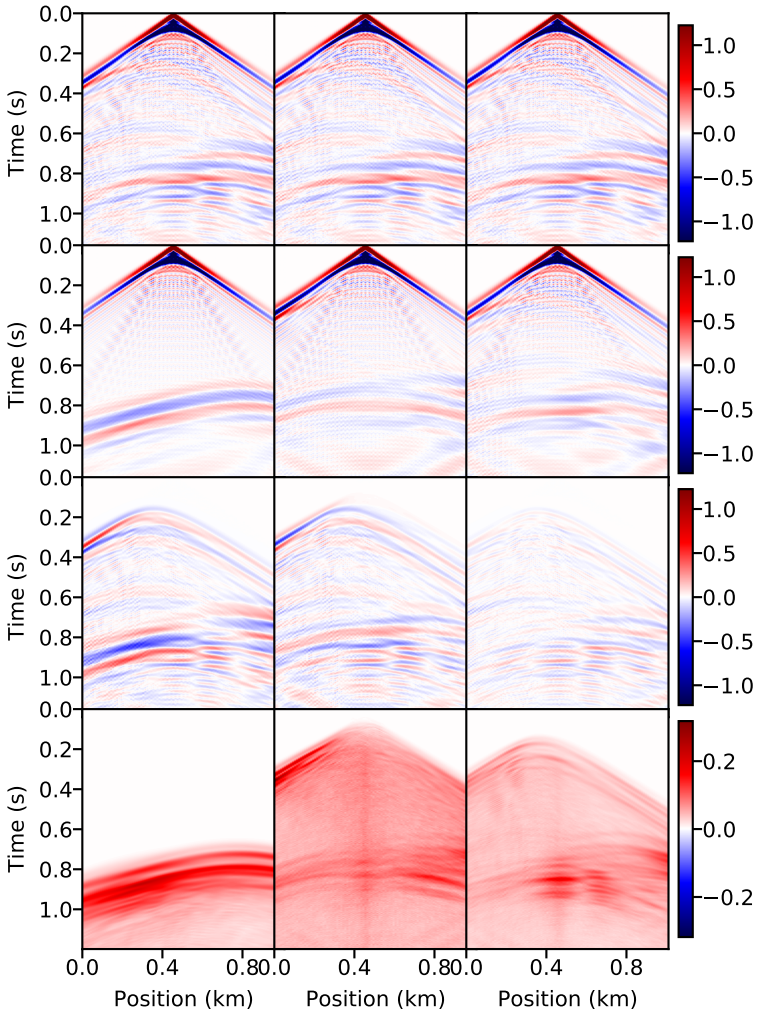

FIG. 20. The uncertainty and average of the middle-located shot gathers using 100 estimated realizations in different modes: from left to right, CNN-2k, CNN-10k, and H-PGNN-2k, respectively. From top to bottom, plots are listed as: the ground-truth, the mean of the middle-located shot gathers using 100 estimated realizations, the data residuals between the ground-truth and the average shot gathers, and the standard derivation of the middle-located shot gathers using 100 realizations.

plotted in the first and second rows, respectively; the model misfits between the ground-truth and the averaged realization are illustrated in the third row; the standard deviations of 100 predictive realizations on SEG salt models are shown in the bottom row. Recall that the entire 3D SEG salt volume has never been utilized in any training processes. The standard deviations of 100 realizations for ten SEG salt models are smaller than 500m/s at most of areas, which confirm the stability and the generalization ability of the H-PGNN-2k.

We surmise that involving the physics-based objective function in the training of a neural network, we gain more information from the same training dataset. However, weaknesses of the physics-based approach, familiar to FWI practitioners, such as illumination issues, and susceptibility to missing or incomplete data, are incorporated at the same time. For example, the high uncertainty of predictive realizations shown in the last row of Figure21 shows the H-PGNN-2k struggles to delineate beneath salt bodies; this is a fa-

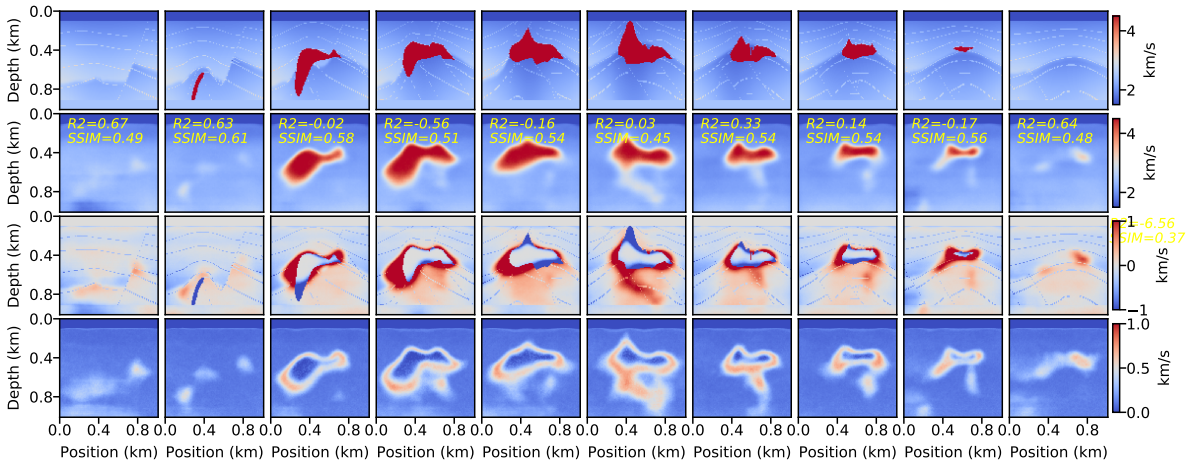

FIG. 21. The uncertainty and average of 100 realizations for ten SEG salt models using H-PGNN-2k. From top to bottom, plots are listed as: the ground-truth model, the mean of 100 predictive realizations, the model misfits between the ground-truth and averaged realization, and the standard derivation of 100 realizations.

miliar feature in imaging based on wave methods (Liu et al., 2011; Malcolm et al., 2009). Whether increasing the training aspect of a network like the one we discuss could mitigate these shortcomings, and if so how to select for them, are questions of ongoing research.

CONCLUSIONS

Deep learning, as a class of artificial intelligence methods, has produced a wide and growing range of solutions for imaging processing problems. However, the accuracy and stability of the underlying supervised neural networks is highly dependent on the scope of the training data, and more practically on the availability of the computational resources required to carry out sufficient training. In large seismic imaging and inversion problems, exhaustive training data sets are largely unavailable. Furthermore, seismic data being the projection of a field, which satisfies to some degree a known partial differential equation, is strongly suggestive that a full data-driven network, trained "from scratch", is not necessary to explain much of the data variance. Therefore, to better employ deep learning techniques in solving geoscience problems, including physics-guiding, and keeping the training data volumes manageably small, is a sensible strategy.

In this paper, we use waveform-based velocity building of salt bodies as a way to enumerate the positive and negative features of a fully data-driven CNN. Our experimentation indicates that, with a small amount of training examples, a fully data-driven CNN is capable of capturing the locations of subsurface salt bodies, but failing on casting the salt body boundaries and background information. The natural route is to explore a significant increase in the training of this CNN and results shows that sufficiently improves the CNN's capacity of extracting the main features of subsurface salt body models. However, recent work in theory-guided machine learning in seismic inversion is suggestive of a second possibility.

To improve the network while keeping the training dataset size and extent unchange, we introduce "physics-guiding", analyzed in isolation in a separate communication, which

involves data residuals, and a wave propagation model to explain them, in the training stage. The aim is to ensure that, while "learning", the neural network penalizes updates which are inconsistent with wave propagation within the current iterate of velocity model. One serious obstacle faced by physics-guided network designers is the requirement for accurate initialization. Whereas, the data-driven approach is seen in these early examples to proceed without the explicit definition of a nearby starting model. By taking advantages of both, we propose the hybrid training strategy to ensure the physical meaning of estimated profiles which further stabilizes the wave propagation. Beyond that, the selection of the trade-off parameters between model misfit and data residual is discussed. Comparing results of this constrained network to those of the data-driven CNN, we observe a strong up-tick in accuracy, and resolution. Furthermore, a comprehensive error analysis, generalization ability evaluation using SEG salt models, and predictive uncertainty analysis using dropout are performed to quantitatively analyze the physics-guided network's performance.

ACKNOWLEDGMENTS

We thank the sponsors of CREWES for continued support. This work was carried out in collaboration with CREWES and as an outgrowth of research funded by CREWES industrial sponsors and NSERC (Natural Science and Engineering Research Council of Canada) through the grant CRDPJ 461179-13 and CRDPJ 543578-19.

REFERENCES

Adler, A., Araya-Polo, M., and Poggio, T., 2019, Deep recurrent architectures for seismic tomography, *in* 81st EAGE Conference and Exhibition 2019, vol. 2019, European Association of Geoscientists & Engineers, 1–5.

Alfarraj, M., and AlRegib, G., 2019, Semi-supervised learning for acoustic impedance inversion, *in* SEG Technical Program Expanded Abstracts 2019, Society of Exploration Geophysicists, 2298–2302.

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

Araya-Polo, M., Dahlke, T., Frogner, C., Zhang, C., Poggio, T., and Hohl, D., 2017, Automated fault detection without seismic processing: The Leading Edge, **36**, No. 3, 208–214.

Araya-Polo, M., Jennings, J., Adler, A., and Dahlke, T., 2018, Deep-learning tomography: The Leading Edge, **37**, No. 1, 58–66.

Biswas, R., Sen, M. K., Das, V., and Mukerji, T., 2019, Prestack and poststack inversion using a physics-guided convolutional neural network: Interpretation, **7**, No. 3, SE161–SE174.

Das, V., and Mukerji, T., 2020, Petrophysical properties prediction from pre-stack seismic data using convolutional neural networks: Geophysics, **85**, No. 5, 1–85.

Das, V., Pollack, A., Wollner, U., and Mukerji, T., 2019, Convolutional neural network for seismic impedance inversion: Geophysics, **84**, No. 6, R869–R880.

Fabien-Ouellet, G., and Sarkar, R., 2020, Seismic velocity estimation: A deep recurrent neural-network approach: Geophysics, **85**, No. 1, U21–U29.

Gal, Y., and Ghahramani, Z., 2016, Dropout as a bayesian approximation: Representing model uncertainty in deep learning, *in* international conference on machine learning, 1050–1059.

Häggström, I., Schmidtlein, C. R., Campanella, G., and Fuchs, T. J., 2019, Deeppet: A deep encoder–decoder network for directly solving the pet image reconstruction inverse problem: Medical image analysis, **54**, 253–262.

Karpatne, A., Ebert-Uphoff, I., Ravela, S., Babaie, H. A., and Kumar, V., 2018, Machine learning for the geosciences: Challenges and opportunities: IEEE Transactions on Knowledge and Data Engineering.

Kaur, H., Fomel, S., and Pham, N., 2019, Elastic wave-mode separation in heterogeneous anisotropic media using deep learning, *in* SEG Technical Program Expanded Abstracts 2019, Society of Exploration Geophysicists, 2654–2658.

Kim, J., Kwon Lee, J., and Mu Lee, K., 2016, Accurate image super-resolution using very deep convolutional networks, *in* Proceedings of the IEEE conference on computer vision and pattern recognition, 1646–1654.

Kingma, D. P., and Ba, J., 2014, Adam: A method for stochastic optimization: arXiv preprint arXiv:1412.6980.

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

Liu, Y., Chang, X., Jin, D., He, R., Sun, H., and Zheng, Y., 2011, Reverse time migration of multiples for subsalt imaging: Geophysics, **76**, No. 5, WB209–WB216.

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

Malcolm, A. E., Ursin, B., and Maarten, V., 2009, Seismic imaging and illumination with internal multiples: Geophysical Journal International, **176**, No. 3, 847–864.

Nair, V., and Hinton, G. E., 2010, Rectified linear units improve restricted boltzmann machines, *in* Proceedings of the 27th international conference on machine learning (ICML-10), 807–814.

Ovcharenko, O., Kazei, V., Kalita, M., Peter, D., and Alkhalifah, T., 2019, Deep learning for low-frequency extrapolation from multioffset seismic data: Geophysics, **84**, No. 6, R989–R1001.

Richardson, A., 2018, Seismic full-waveform inversion using deep learning tools and techniques: arXiv preprint arXiv:1801.07232.

Ronneberger, O., Fischer, P., and Brox, T., 2015, U-net: Convolutional networks for biomedical image segmentation, *in* International Conference on Medical image computing and computer-assisted intervention, Springer, 234–241.

Simonyan, K., and Zisserman, A., 2014, Very deep convolutional networks for large-scale image recognition: arXiv preprint arXiv:1409.1556.

Smith, G. D., 1985, Numerical solution of partial differential equations: finite difference methods: Oxford university press.

Sun, J., Niu, Z., Innanen, K. A., Li, J., and Trad, D. O., 2020, A theory-guided deep-learning formulation and optimization of seismic waveform inversion: Geophysics, **85**, No. 2, R87–R99.

Tarantola, A., 1984, Inversion of seismic reflection data in the acoustic approximation: Geophysics, **49**, No. 8, 1259–1266.

Virieux, J., and Operto, S., 2009, An overview of full-waveform inversion in exploration geophysics: Geophysics, **74**, No. 6, WCC1–WCC26.

Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P., 2004, Image quality assessment: from error visibility to structural similarity: IEEE transactions on image processing, **13**, No. 4, 600–612.

Woodward, M. J., Nichols, D., Zdraveva, O., Whitfield, P., and Johns, T., 2008, A decade of tomography: Geophysics, **73**, No. 5, VE5–VE11.

Wu, X., Shi, Y., Fomel, S., Liang, L., Zhang, Q., and Yusifov, A. Z., 2019, Faultnet3d: predicting fault probabilities, strikes, and dips with a single convolutional neural network: IEEE Transactions on Geoscience and Remote Sensing, **57**, No. 11, 9138–9155.

Wu, Y., and McMechan, G. A., 2019, Parametric convolutional neural network-domain full-waveform inversion: Geophysics, **84**, No. 6, R881–R896.

Yang, F., and Ma, J., 2019, Deep-learning inversion: A next-generation seismic velocity model building method: Geophysics, **84**, No. 4, R583–R599.

Zhang, K., Zuo, W., Chen, Y., Meng, D., and Zhang, L., 2017, Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising: IEEE Transactions on Image Processing, **26**, No. 7, 3142–3155.

Zhang, T., Innanen, K., Sun, J., and Trad, D., 2020, Numerical analysis of a deep learning formulation of elastic full waveform inversion: SEG Technical Program Expanded Abstracts 2020.