SCIENTIFIC REPORTS
natureresearch

OPEN

Deep learning for irregularly and regularly missing data reconstruction

Xintao Chai[1], Hanming Gu[1*], Feng Li[2], Hongyou Duan[3], Xiaobo Hu[3] & Kai Lin[1]

Deep learning (DL) is a powerful tool for mining features from data, which can theoretically avoid assumptions (e.g., linear events) constraining conventional interpolation methods. Motivated by this and inspired by image-to-image translation, we applied DL to irregularly and regularly missing data reconstruction with the aim of transforming incomplete data into corresponding complete data. To accomplish this, we established a model architecture with randomly sampled data as input and corresponding complete data as output, which was based on an encoder-decoder-style U-Net convolutional neural network. We carefully prepared the training data using synthetic and field seismic data. We used a mean-squared-error loss function and an Adam optimizer to train the network. We displayed the feature maps for a randomly sampled data set going through the trained model with the aim of explaining how the missing data are reconstructed. We benchmarked the method on several typical datasets for irregularly missing data reconstruction, which achieved better performances compared with a peer-reviewed Fourier transform interpolation method, verifying the effectiveness, superiority, and generalization capability of our approach. Because regularly missing is a special case of irregularly missing, we successfully applied the model to regularly missing data reconstruction, although it was trained with irregularly sampled data only.

Deep learning (DL)[1] is a branch of machine learning (ML) that addresses the question of how to build computers that intelligently improve through experience[2]. Recently, DL or ML, in general, enjoyed an explosive growth and showed great promise in various areas, e.g., biology[3,4], image reconstruction[5,6], and solid earth geoscience[7]. DL is powerful for mining features or relationships from data, which is invaluable in the context of big data, as it extracts high-level information from huge volumes of data. Please refer to Goodfellow *et al.*[8] for a good textbook of DL. One of the most popular DL technologies is the convolutional neural network (CNN), which is at the core of most state-of-the-art DL solutions for numerous tasks[9]. In recent years, deep CNNs have had stunning successes, surpassing human accuracy for hard problems such as visual recognition[6].

In exploration seismology, DL or ML has been widely used in fault detection[10], structural interpretation[11], inversion[12], and data interpolation[13–15], to name a few. A more tremendous trend of developments has recently come about through the use of DL not for image analysis but for image transformation. In these cases, CNNs are trained to transform one type of image into another. Many geophysical problems can be posed as transforming an input profile into a corresponding output profile (e.g., denoising: transforming noisy data to noise-free data). Inspired by Isola *et al.*[16], where DL is investigated as a general-purpose solution to image-to-image translation problems, we apply DL to missing data reconstruction with the aim of transforming an input incomplete data set into a corresponding complete data set, which is an important ongoing research topic in exploration seismology.

Physical (e.g., the presence of obstacles, no-permit areas, and hardware problems with geophones/hydrophones/air-guns) and economic constraints lead seismic data to be incomplete or sparsely sampled during data acquisition[17]. However, many important techniques cannot adequately handle irregular sampling and rely on uniformly and densely sampled, unaliased input data (e.g., 2D/3D surface-related multiple elimination, amplitude-variation-with-offset analysis, and reservoir characterization). The performance of multichannel data processing depends heavily on the spatial sampling intervals. Too large an interval leads to aliasing, adversely resulting in poor resolution. Therefore, the missing data should be reconstructed[18].

[1]China University of Geosciences (Wuhan), Institute of Geophysics and Geomatics, DeepResearch Group, Center for Wave Propagation and Imaging, Wuhan, Hubei, China. [2]Sinopec Henan Oilfield Branch Company, Nanyang, Henan, China. [3]Henan Oilfield Exploration and Development Research Institute, Zhengzhou, Henan, China. *email: hmgu@cug.edu.cn

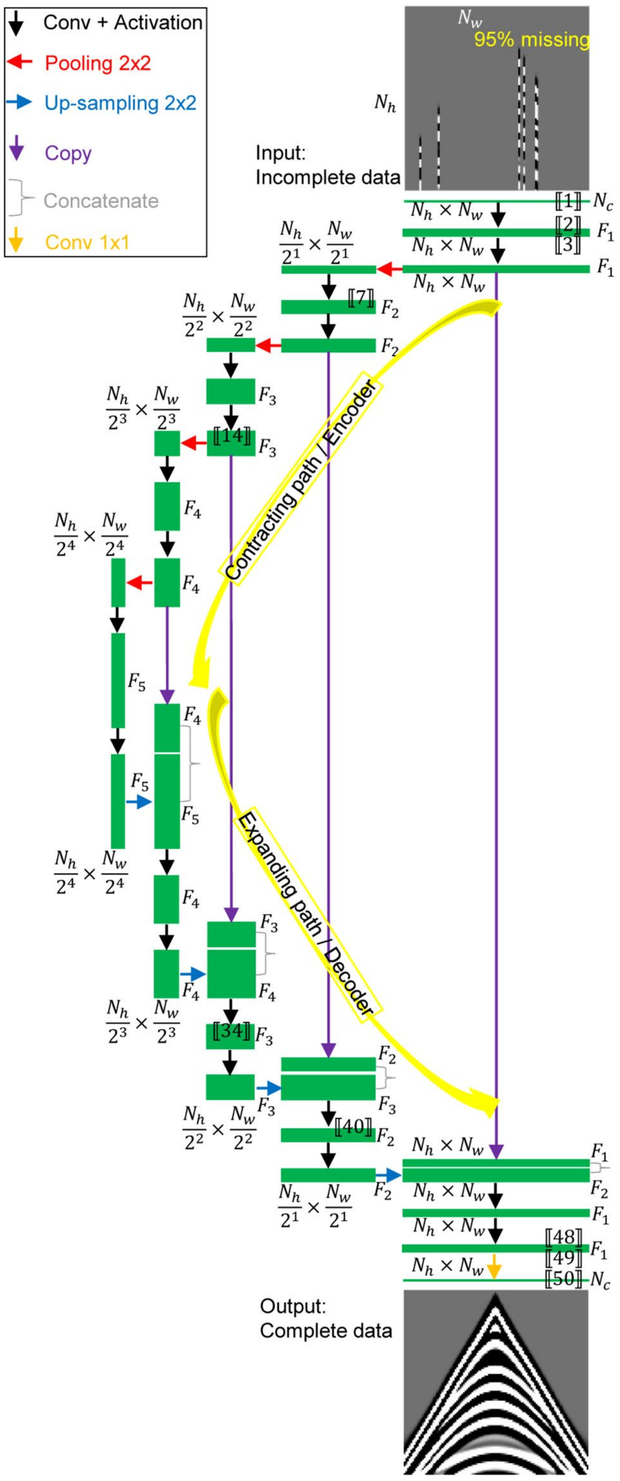

Figure 1. Network architecture at an example of $\frac{N_h}{2^4} \times \frac{N_w}{2^4}$ pixels in the lowest resolution. N_h, N_w, and N_c are the height, width, and the number of channels of the input data, respectively. Blocks show the calculated feature hierarchy. Each green box denotes multiple feature maps, and the number of feature maps (i.e., $F_{i,i\in[1,5]}$) is marked on the right of the box. The height-width-size of a feature map is given around the box. The boxes with the same height have the same number of feature maps. The boxes with the same width indicate the same height-width-size of feature maps. The arrows and the right curly brace denote different operations. Numbers in [[·]] are labelled according to Table 1, which are in line with those shown in Fig. 2.

The missing data problem can be classified into two categories: regularly missing and irregularly missing. Regularly missing means the data are equidistantly or periodically missing at a constant rate in uniform grids. Irregularly missing means the data are randomly missing on uniform grids[19]. Seismic data are often irregularly

Layer number	Layer name	Number of trainable parameters
2	Conv01	$(K_hK_w \times N_c + 1) \times F_1 = 1664$
4	Conv02	$(K_hK_w \times F_1 + 1) \times F_1 = 102464$
7	Conv03	$(K_hK_w \times F_1 + 1) \times F_2 = 204928$
9	Conv04	$(K_hK_w \times F_2 + 1) \times F_2 = 409728$
12	Conv05	$(K_hK_w \times F_2 + 1) \times F_3 = 819456$
14	Conv06	$(K_hK_w \times F_3 + 1) \times F_3 = 1638656$
17	Conv07	$(K_hK_w \times F_3 + 1) \times F_4 = 3277312$
19	Conv08	$(K_hK_w \times F_4 + 1) \times F_4 = 6554112$
22	Conv09	$(K_hK_w \times F_4 + 1) \times F_5 = 13108224$
24	Conv10	$(K_hK_w \times F_5 + 1) \times F_5 = 26215424$
28	Conv11	$(K_hK_w \times (F_5 + F_4) + 1) \times F_4 = 19661312$
30	Conv12	$(K_hK_w \times F_4 + 1) \times F_4 = 6554112$
34	Conv13	$(K_hK_w \times (F_4 + F_3) + 1) \times F_3 = 4915456$
36	Conv14	$(K_hK_w \times F_3 + 1) \times F_3 = 1638656$
40	Conv15	$(K_hK_w \times (F_3 + F_2) + 1) \times F_2 = 1228928$
42	Conv16	$(K_hK_w \times F_2 + 1) \times F_2 = 409728$
46	Conv17	$(K_hK_w \times (F_2 + F_1) + 1) \times F_1 = 307264$
48	Conv18	$(K_hK_w \times F_1 + 1) \times F_1 = 102464$
50	Conv19	$(1 \times 1 \times F_1 + 1) \times N_c = 65$

Table 1. Model summary for the model architecture shown in Fig. 1, where $N_c = 1$, $F_1 = 64$, $F_2 = 128$, $F_3 = 256$, $F_4 = 512$, $F_5 = 1024$, and $K_hK_w = K_h \times K_w$. $K_h = K_w = 5$ denote the height and width of the convolution kernel, respectively. The first layer is an input layer. The layers numbered 3, 5, 8, 10, 13, 15, 18, 20, 23, 25, 29, 31, 35, 37, 41, 43, 47, and 49 are (18) activation layers. The layers numbered 6, 11, 16, and 21 are (4) max-pooling layers. The layers numbered 26, 32, 38, and 44 are (4) up-sampling layers. The layers numbered 27, 33, 39, and 45 are (4) concatenate layers. The total trainable parameters are 87,149,953.

and sparsely sampled along the spatial coordinates, leading to suboptimal processing and imaging results. This work primarily concentrates on solving the irregularly missing data problem. Solving the regularly missing data reconstruction problem is an unexpected harvest.

Based on a variety of principles and assumptions, important advances have been made in seismic data reconstruction. Some of them addressed interpolating regularly sampled data[13–15,20], while some of them attacked non-uniformly sampled interpolation[21]. There are techniques developed for both irregularly and regularly missing data reconstruction[22]. A complete and detailed discussion of previous publications is beyond the scope of this work. We only review some key viewpoints and literature closely related to the subject of this work. The methods based on classical signal-processing principles use specific properties of seismic data as a priori information for interpolation. Signal-processing reconstruction techniques via transforming the data to other domains and prediction-error filtering generally assume that the data are composed of a superposition of a few plane waves. The sparseness, band limitation, and low-rank assumptions also underlie some of these methods.

Naghizadeh and Innanen[23] addressed seismic data interpolation using a fast-generalized Fourier transform (FGFT). They utilized the FGFT to identify the space-wavenumber evolution of spatial signals at each temporal frequency, and a least-squares fitting scheme to retrieve the optimal FGFT coefficients representative of the desired interpolated data. For randomly sampled data, they sought a sparse representation of FGFT coefficients to retrieve the missing pixels. To interpolate the regularly sampled data at a given frequency, they used a mask function derived from the FGFT coefficients of the low frequencies. This makes the FGFT interpolation method a good competitor, which is used for comparison in our work.

Of the multitudinous methods for seismic data interpolation, few take advantage of recent developments in DL or ML. Jia and Ma[13] proposed a method for reconstructing seismic data from regularly under-sampled traces based on a classic ML method of support vector regression. Jia *et al.*[14] proposed an intelligent interpolation method for regularly sampled data by Monte Carlo ML. Wang *et al.*[15] proposed a DL-based approach for regularly sampled seismic data antialiasing interpolation. Based on CNNs, Wang *et al.*[15] designed eight-layer residual networks (ResNets) with a better back-propagation property for interpolation, which extract feature maps of the training data in a non-linear way. For the methods of Jia and Ma[13], Jia *et al.*[14], and Wang *et al.*[15], in the training process, to generate the input of the designed network, calculation of the initial pre-interpolation data using a bicubic method is required, which affects the final performance of their methods.

With the DL theory described in Goodfellow *et al.*[8], we applied DL to both irregularly and regularly missing data reconstruction, where we defined intelligent data-to-data translation as the task of translating incomplete data into complete data (without pre-interpolation). DL allows trainable models composed of multiple layers to learn representations of data with multiple levels of abstraction[1]. DL is a representation-learning method with multiple levels of representation obtained by composing non-linear modules, in which each transforms the representation at one level into a representation at a higher, slightly more abstract level[8]. With the composition of enough such layers, and given sufficient training data, very complicated functions/relations can be learned.

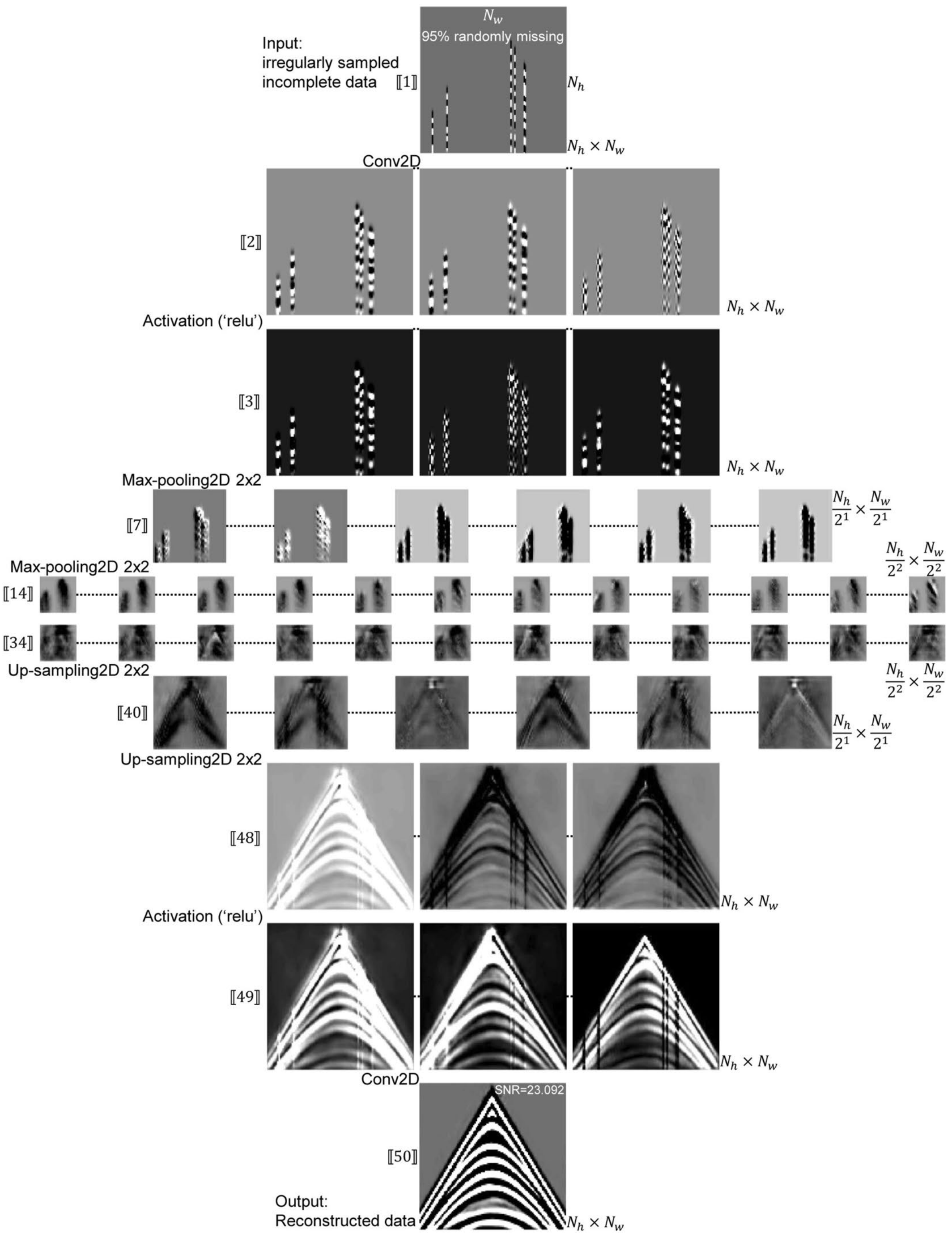

Figure 2. Feature maps for a randomly sampled data set going through the trained model. Numbers in [[·]] are labelled according to Table 1, which are consistent with those shown in Fig. 1. The symbol "..." implies the omitted feature maps.

Moreover, as a motivation, DL can theoretically avoid some assumptions restricting conventional interpolation methods (e.g., assumptions of linear events, sparsity, band limitation, and low-rank).

The rest of the paper is organized as follows. In the "Methods" section, first, we briefly transcribe some basic DL theory; next, with the incomplete data as the model input and the corresponding complete data as the model output, we elaborate the established model architecture in detail, which is based on an U-Net convolutional network[4]; then, we provide detailed training analysis including the loss function definition, optimizer used, evaluation metrics, training data preparation, and parameter setup. In the "Results" section, to make the results more convincing and to validate the generalization capacity of the trained model, we test the model's performance using several typical data sets (i.e., a synthetic training data set, a synthetic test data set, a physical modelling data set, the Mobil Viking graben line 12 data set, the F3 data set, a fault data set from the GeoFrame software, and a data set from the North Sea). The trained model is used to accomplish both irregularly and regularly missing

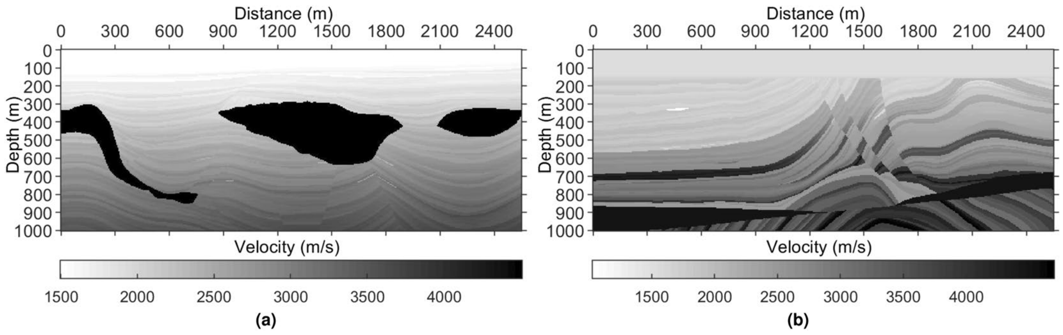

Figure 3. Velocity models used to generate the synthetic data. Each velocity model has its own density model. (**a**) Adapted Pluto 1.5 model. (**b**) Down-sampled Marmousi2 model. The lateral spacing dx and the depth spacing dz are 5 m.

data reconstruction. After discussing some practical aspects and extensions of this work, we summarize some concluding remarks.

Methods

Basic theory.

For the sake of brevity, we refer the reader to Goodfellow *et al.*[8] for a detailed description of the DL terminologies. DL is a computational tool to learn complex motifs from data. DL uses multiple processing layers to discover patterns and structures in the data. Each layer learns features from the data that subsequent layers build on. Non-linear parameterized processing layers are combined to progressively transform an input $\mathbf{X}$ into the desired output $\mathbf{Y}_{\text{ref}}$, typically attaining only approximations $\mathbf{Y}_{\text{pre}}$. Deep artificial neural networks (ANNs) are black-box models whose operation is opaque and difficult to interpret. The adjective "deep" refers to the large number of stacked layers required for building a universal function approximator f as follows:

$$\mathbf{Y}_{\text{pre}} = f(\mathbf{X};\, \theta), \tag{1}$$

where θ denotes the parameters (including but not limited to, weights $\mathbf{W}$ and biases $\mathbf{b}$ of the convolution kernels) in the CNNs.

We regard ANNs as bridges connecting the input $\mathbf{X}$ and the desired output $\mathbf{Y}_{\text{ref}}$. ANNs with a sufficient number of parameters can theoretically approximate any function. For fitting the desired mapping, the model needs to go through a training process, which begins with a random choice for θ. The training process can be considered as an optimization problem composed by finding a set of the internal parameters θ through the minimization of the discrepancy between $\mathbf{Y}_{\text{pre}}$ and $\mathbf{Y}_{\text{ref}}$ (quantified by the loss function ϕ, which will be explained later) for all the samples fed into the model[12].

The parameters θ are iteratively updated to minimize the loss using gradient descent and to improve the accuracy of the model prediction. Each layer of the model is differentiable, meaning that it is known how changes in θ cause changes in output values. The back-propagation (BP) algorithm[24] uses the chain rule to efficiently compute all partial derivatives, or gradients, with just one forward pass through the model followed by a backward pass. The training process is accomplished if the loss function achieves an acceptable level. The optimizer for the loss function is an important requirement for training a model.

Model architecture.

There is no hard and fast rule for how many layers are needed to constitute ANNs, but most researchers agree that no less than three are required. Figure 1 shows a schematic of the established model architecture, which belongs to a specific family of neural network (NN) architectures known as U-Net, a generic DL solution for various tasks[4]. Specific to our mission, the model input data include one horizontal (spatial) dimension and one vertical (temporal) dimension. In seismic terminology, N_h and N_w denote the sampling points of the input data along the time and space axes, respectively. The number of channels N_c equals 1 for seismic data. A data sample with N_c channels is fed into the model on the top. The model input data $\mathbf{X}$ are randomly missing in accordance with a certain percentage (e.g., 40% → 95%) in the space direction. Please note that no pre-interpolation process is involved. The model output $\mathbf{Y}_{\text{ref}}$ (at the bottom) is the corresponding complete data.

The model architecture (Fig. 1) is an encoder-decoder-style NN solving the missing data reconstruction task end-to-end, which is logically composed of a contracting path (upper-side, interpreted as an encoder) and a more or less symmetric expanding path (lower-side, interpreted as a decoder). The encoder takes an incomplete data sample as input and gradually calculates feature maps at multiple scales and abstraction levels resulting in a multi-level, multi-resolution feature representation. Layers in the decoder successively synthesize the complete data starting at low-resolution feature maps (denoting large scale structures) up to high-resolution feature maps (representing fine scale structures)[4]. Please see Fig. 2 for a better understanding.

The contracting path/encoder follows a typical CNN, consisting of the repeated application of two padded convolutions (padding avoids the loss of border pixels in every convolution), each followed by an activation operation (black-arrow) and a pooling operation (red-arrow) with stride two halving the resolution of the resulting feature map. Convolutions directly following down-sampling steps double the number of feature maps[4]. Each step in the expansive path/decoder consists of a bed-of-nails up-sampling of the feature maps by a factor of two

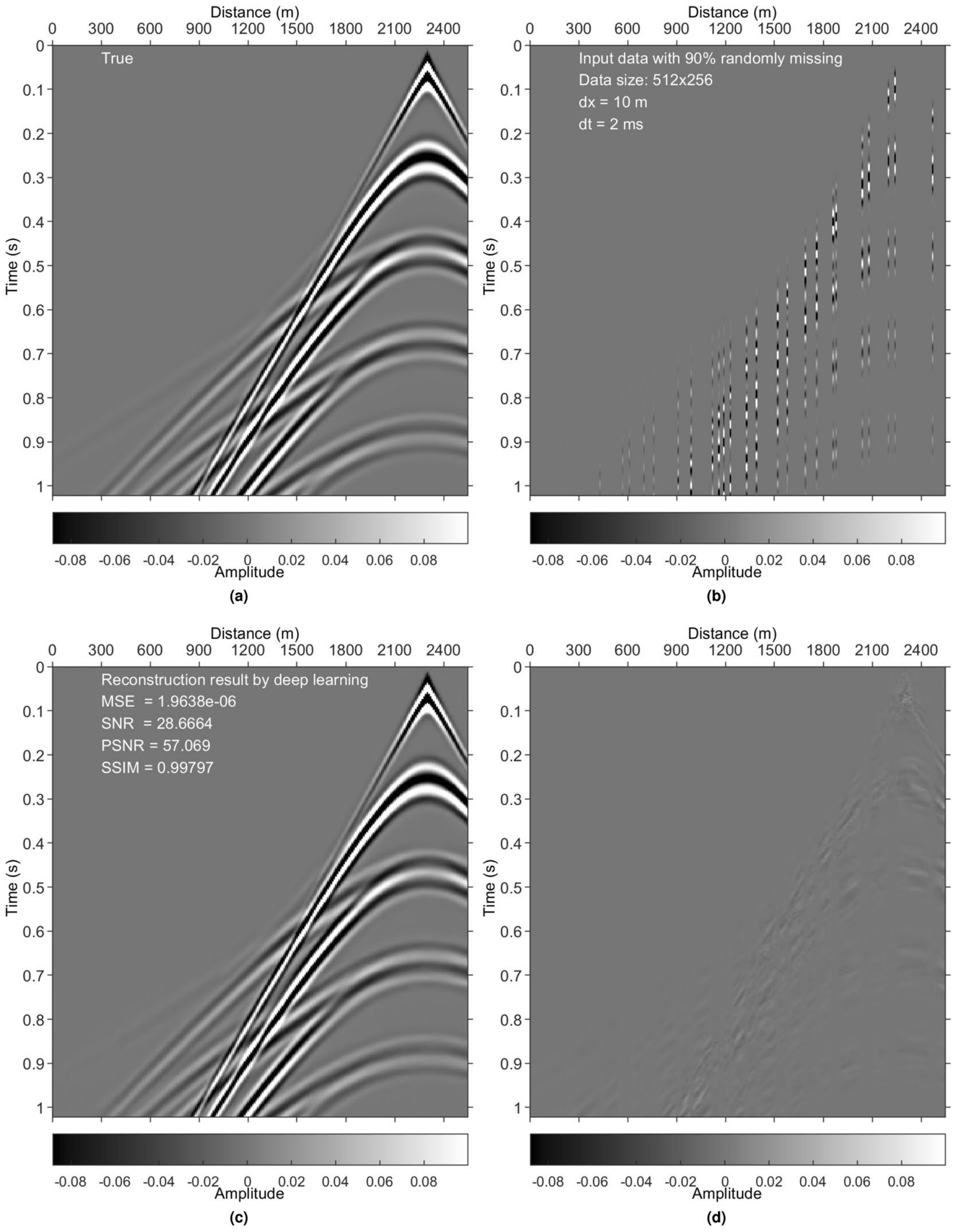

Figure 4. Results on a synthetic training data set. (**a**) True data (a common-shot-point, CSP, gather). (**b**) Irregularly sampled data with 90% missing. (**c**) DL reconstruction result. (**d**) Difference between (**a**,**c**).

(blue-arrow) followed by a concatenation (right curly brace) with the copied encoder feature maps at the corresponding resolution (purple-arrow), and two convolutions, each followed by an activation. The feature maps from the contracting path are combined with the up-sampled output. Successive convolution layers then learn to assemble a more precise output based on this. Skip connections have been shown to help train deeper networks by preventing vanishing gradients[25]. At the final layer, a 1×1 convolution (green-arrow) is utilized to map multiple feature maps to the desired output $\mathbf{Y}_{\text{ref}}$.

A widely used non-linear activation function is the rectified linear unit (ReLU), which returns element-wise $\max(\mathbf{x},\ 0)$ with $\mathbf{x}$ being an input tensor. The ReLU activation was adopted in the constructed model. For the pooling operation, we employed the max-pooling layer. Note that the number of output filters in the convolution (i.e., $F_{i, i \in [1,\, 5]}$) increases (e.g., from 64 to 128, 256, 512, and 1024) as we go deep in the model. At each down-sampling step, we generally double the number of feature maps.

Table 1 provides a specified model summary. There are fifty layers (including 19 convolution layers). The trainable parameters focus on convolution layers. No trainable parameters exist in the input, ReLU activation, max-pooling, up-sampling, and concatenate layers. The trainable parameters in a convolution layer are computed by the following:

$$(K_h \times K_w \times F_{i-1} + 1) \times F_i, \tag{2}$$

where $K_h \times K_w$ is the convolution kernel size, F_{i-1} and F_i denote the number of feature maps in the previous and current layers, respectively. $+1$ means a bias is added. Figure 2 shows the feature maps for a randomly missing data sample going through a trained model. Once trained, it is capable of filling in the gaps in corrupted data by going through encoding and decoding steps.

Loss function. For our problem, we used a mean-squared-error (MSE) training loss function, which measures the average of the squares of the errors. Given the reference solution $\mathbf{Y}_{\text{ref}}$ and the model prediction $\mathbf{Y}_{\text{pre}}$, the MSE is computed as follows:

$$\text{MSE} = \frac{1}{L}||\mathbf{Y}_{\text{ref}} - \mathbf{Y}_{\text{pre}}||^2_{\text{Fro}} = \frac{1}{L}||\mathbf{Y}_{\text{ref}} - f(\mathbf{X};\boldsymbol{\theta})||^2_{\text{Fro}}, \tag{3}$$

where L is the number of elements in $\mathbf{Y}_{\text{ref}}$, $||\cdot||_{\text{Fro}}$ being the Frobenius norm. The MSE loss function is widely used in statistics. Because we will work in a batch-wise fashion in the training process, the loss function is composed of a sum of subfunctions evaluated at different mini-batches of data. The loss function is accordingly stochastic.

Optimization algorithm. We employed an Adam (derived from adaptive moment estimation) algorithm to optimize our stochastic loss function. Adam is a simple and computationally efficient algorithm for first-order gradient-based optimization[26]. Please see algorithm 1 for the pseudo-code of the employed Adam algorithm. $\phi(\boldsymbol{\theta})$ denotes the stochastic scalar loss function, which is differentiable with respect to parameters θ. $\mathbf{g}_t = \nabla_\theta \phi_t(\boldsymbol{\theta})$ represents the gradient, i.e., the vector of partial derivatives of ϕ_t with respect to θ evaluated at time step t. Adam updates exponential moving averages of the gradient $\mathbf{m}_t$ and the squared gradient $\mathbf{v}_t$ where the hyper-parameters $\beta_1, \beta_2 \in [0, 1)$ control the exponential decay rates of these moving averages. The moving averages themselves are estimates of the first moment (the mean) and the second moment (the uncentred variance) of the gradient[26].

Algorithm 1 shows that Adam only requires first-order gradients, which is straightforward to implement with little memory requirement. Adam is designed to combine the advantages of two previously popular optimization algorithms: AdaGrad[27], which deals with sparse gradients well, and RMSProp, which works well in on-line and non-stationary objectives[26]. Adam is based on adaptive estimates of lower-order moments. Kingma and Ba[26] analysed the theoretical convergence properties of the Adam algorithm. Adam is a versatile algorithm for DL problems with big data and/or high-dimensional parameter spaces. Using large ANNs and data sets, Kingma and Ba[26] found Adam to be efficient, robust and well-suited to a wide range of practical non-convex optimization problems in the DL field.

Evaluation metrics. As mentioned above, MSE is a measure of the quality of a predictor, which is always non-negative. With our experience in the seismic data reconstruction field, we also utilized the signal-to-noise ratio (SNR) as follows:

$$\text{SNR(dB)} = -20\log_{10}\frac{||\mathbf{Y}_{\text{ref}} - \mathbf{Y}_{\text{pre}}||_2}{||\mathbf{Y}_{\text{ref}}||_2}, \tag{4}$$

to assess the result's quality. Moreover, we used two other metrics that are widely used in the computer vision super-resolution field: peak signal-to-noise ratio (PSNR) and structural similarity index method (SSIM)[28]. PSNR is computed in the following way:

Algorithm 1. Adam, employed optimization algorithm for stochastic loss function.

```
1. Requirements: $\alpha$: Step size, aka learning rate. $\beta_1, \beta_2 \in [0,1)$: Exponential decay rates for the moment estimates. $\phi(\theta)$: Stochastic loss function with parameters $\theta$.
2. Initialization: All the following operations on vectors are element-wise. $\theta_0$: Initial parameter vector. Initialize first moment vector $\mathbf{m}_0 = 0$, second moment vector $\mathbf{v}_0 = 0$, time step $t = 0$.
3. while $\theta_t$ not converged do
4.     $t = t + 1$
5.     $\mathbf{g}_t = \nabla_\theta \phi_t(\theta_{t-1})$. Get gradients with respect to stochastic loss at time step $t$.
6.     $\mathbf{m}_t = \beta_1 \times \mathbf{m}_{t-1} + (1-\beta_1) \times \mathbf{g}_t$. Update biased first moment estimate.
7.     $\mathbf{v}_t = \beta_2 \times \mathbf{v}_{t-1} + (1-\beta_2) \times \mathbf{g}_t^2$. Update biased second moment estimate. $\mathbf{g}_t^2$ indicates the element-wise square $\mathbf{g}_t \odot \mathbf{g}_t$.
8.     $\hat{\mathbf{m}}_t = \frac{\mathbf{m}_t}{1-\beta_1^t}$. Compute bias-corrected first moment estimate. $\beta_1^t$ denotes $\beta_1$ to the power $t$.
9.     $\hat{\mathbf{v}}_t = \frac{\mathbf{v}_t}{1-\beta_2^t}$. Compute bias-corrected second moment estimate. $\beta_2^t$ denotes $\beta_2$ to the power $t$.
10.    $\theta_t = \theta_{t-1} - \alpha\frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t}+\varepsilon}$. Update parameters.
11. end while
12. Output: Resulting parameters $\theta_t$.
```

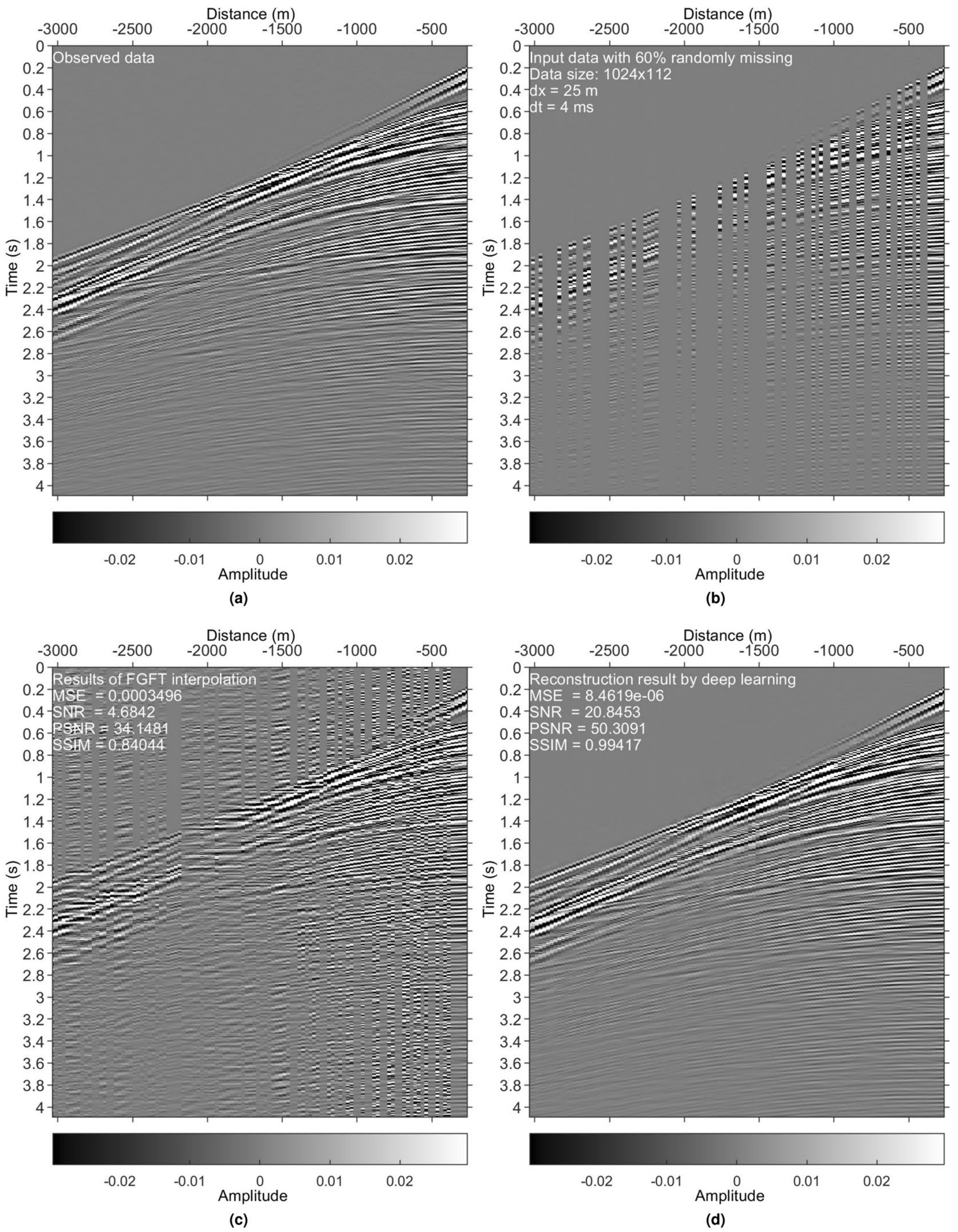

Figure 5. Results on the Mobil Viking graben line 12 data set. (**a**) Reference data. (**b**) Irregularly sampled data with 60% missing. (**c**) Interpolated data using the FGFT interpolation method[23]. (**d**) DL reconstruction result.

$$\mathrm{PSNR(dB)} = 10\log_{10}\left(\frac{M^2}{\mathrm{MSE}}\right), \tag{5}$$

where M is the maximum value of elements in $\mathbf{Y}_{\mathrm{ref}}$. The SSIM is defined as follows:

$$\mathrm{SSIM}\left(\mathbf{Y}_{\mathrm{ref}}, \mathbf{Y}_{\mathrm{pre}}\right) = \frac{\left(2\mu_{\mathbf{Y}_{\mathrm{ref}}}\mu_{\mathbf{Y}_{\mathrm{pre}}} + C_1\right)}{\left(\mu^2_{\mathbf{Y}_{\mathrm{ref}}} + \mu^2_{\mathbf{Y}_{\mathrm{pre}}} + C_1\right)} \times \frac{\left(2\sigma_{\mathbf{Y}_{\mathrm{ref}}\mathbf{Y}_{\mathrm{pre}}} + C_2\right)}{\left(\sigma^2_{\mathbf{Y}_{\mathrm{ref}}} + \sigma^2_{\mathbf{Y}_{\mathrm{pre}}} + C_2\right)}, \tag{6}$$

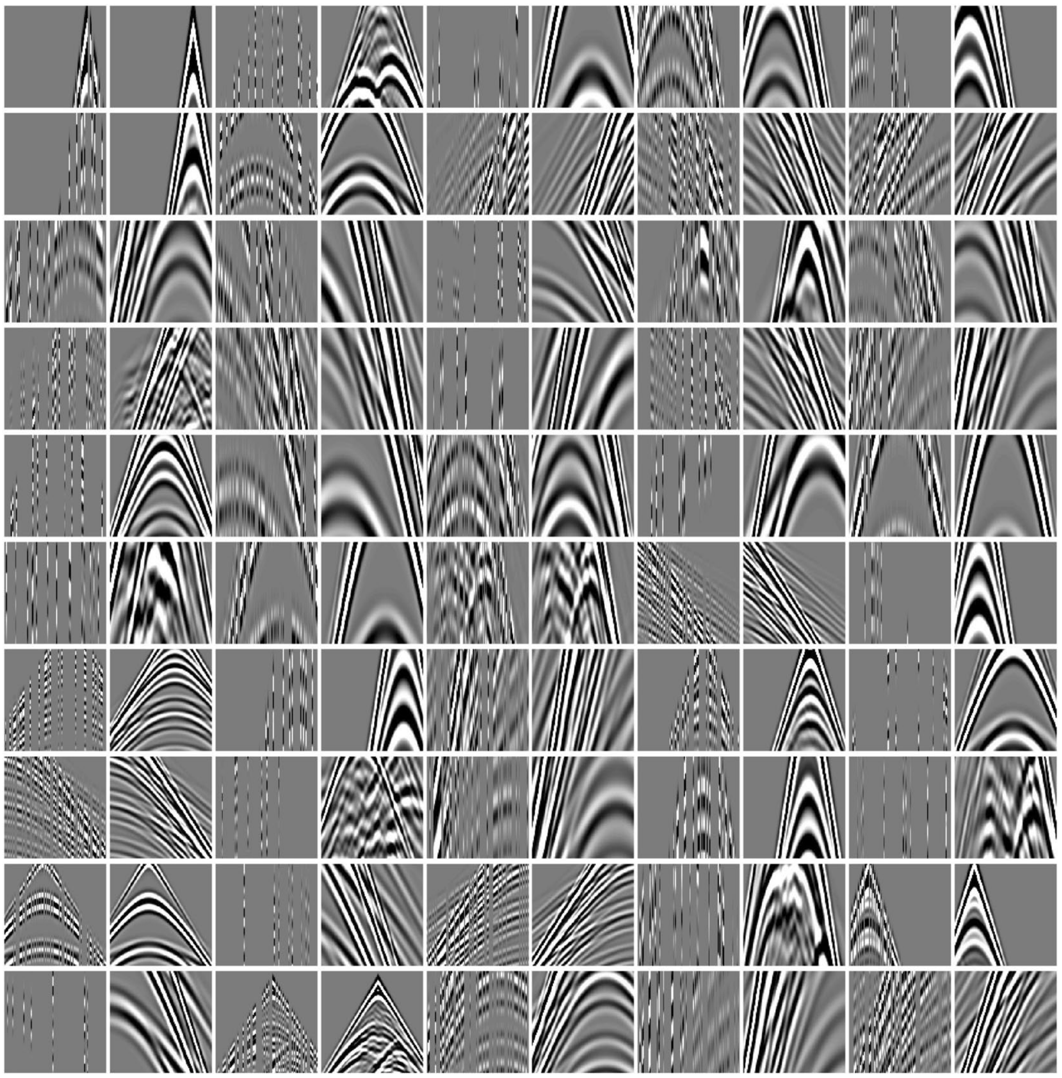

Figure 6. Model input-output pairs, which are randomly selected from the training data set. Each pair is composed of irregularly missing incomplete data (in the odd column, i.e., the model input **X**) and the corresponding complete data (in the even column, i.e., the model output $\mathbf{Y}_{ref}$). The size of each panel is 112×112.

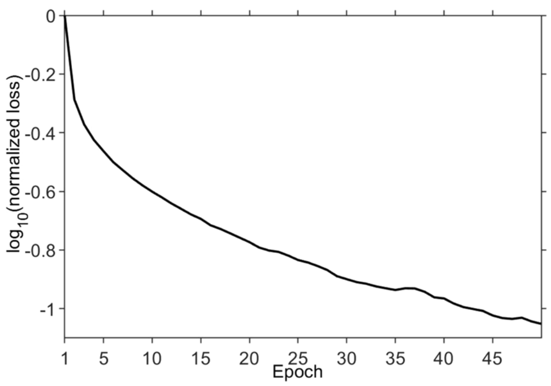

Figure 7. During training, variation of $\log_{10}$(normalized loss) with epoch.

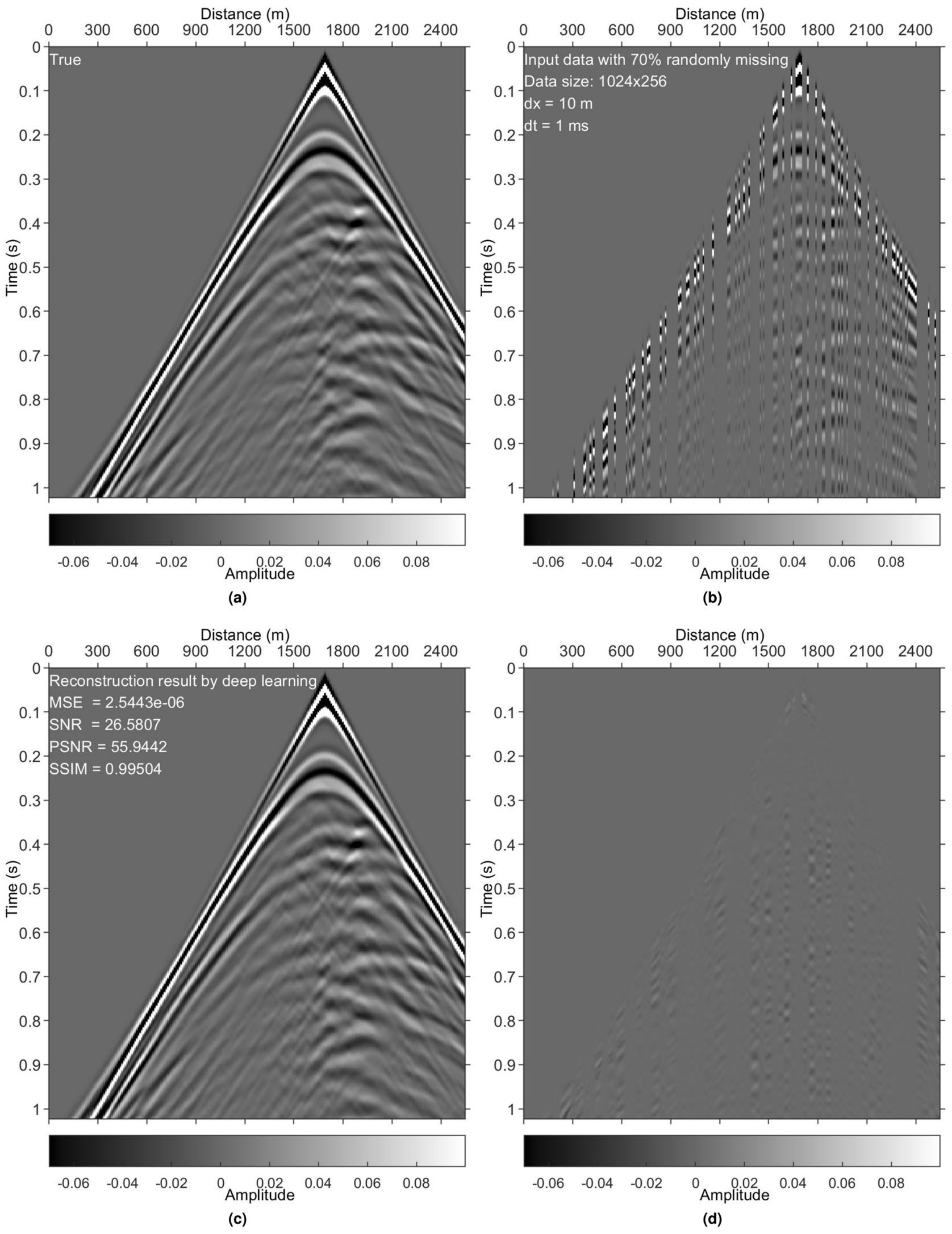

Figure 8. Results on a test data set generated with the Marmousi2 model. (**a**) True data. (**b**) Irregularly sampled data with 70% missing. (**c**) DL reconstruction result. (**d**) Difference between (**a**,**c**).

where $\mu_{\mathbf{Y}_{\mathrm{ref}}}$ and $\mu_{\mathbf{Y}_{\mathrm{pre}}}$ denote the mean of $\mathbf{Y}_{\mathrm{ref}}$ and $\mathbf{Y}_{\mathrm{pre}}$, respectively; $\sigma_{\mathbf{Y}_{\mathrm{ref}}}$ and $\sigma_{\mathbf{Y}_{\mathrm{pre}}}$ represent the variance of $\mathbf{Y}_{\mathrm{ref}}$ and $\mathbf{Y}_{\mathrm{pre}}$, respectively; $\sigma_{\mathbf{Y}_{\mathrm{ref}}\mathbf{Y}_{\mathrm{pre}}}$ is covariance between $\mathbf{Y}_{\mathrm{ref}}$ and $\mathbf{Y}_{\mathrm{pre}}$. The constant C_1 is included to avoid instability when $\mu^2_{\mathbf{Y}_{\mathrm{ref}}} + \mu^2_{\mathbf{Y}_{\mathrm{pre}}}$ is very close to zero. Similarly, the constant C_2 is developed to avoid instability if $\sigma^2_{\mathbf{Y}_{\mathrm{ref}}} + \sigma^2_{\mathbf{Y}_{\mathrm{pre}}}$ is very close to zero.

The metrics MSE, SNR, PSNR, and SSIM can be used in the training and test processes to quantify the performance of the result. The MSE values closer to zero are better. Generally, the higher the SNR, PSNR, and SSIM values are, the better the result.

Training preparation and setup.

Training data. The training data are vital for DL. We prepared the training data utilizing not only the synthetic data but also the field seismic data, to let the model learn the features of seismic data by being given as many instances as possible.

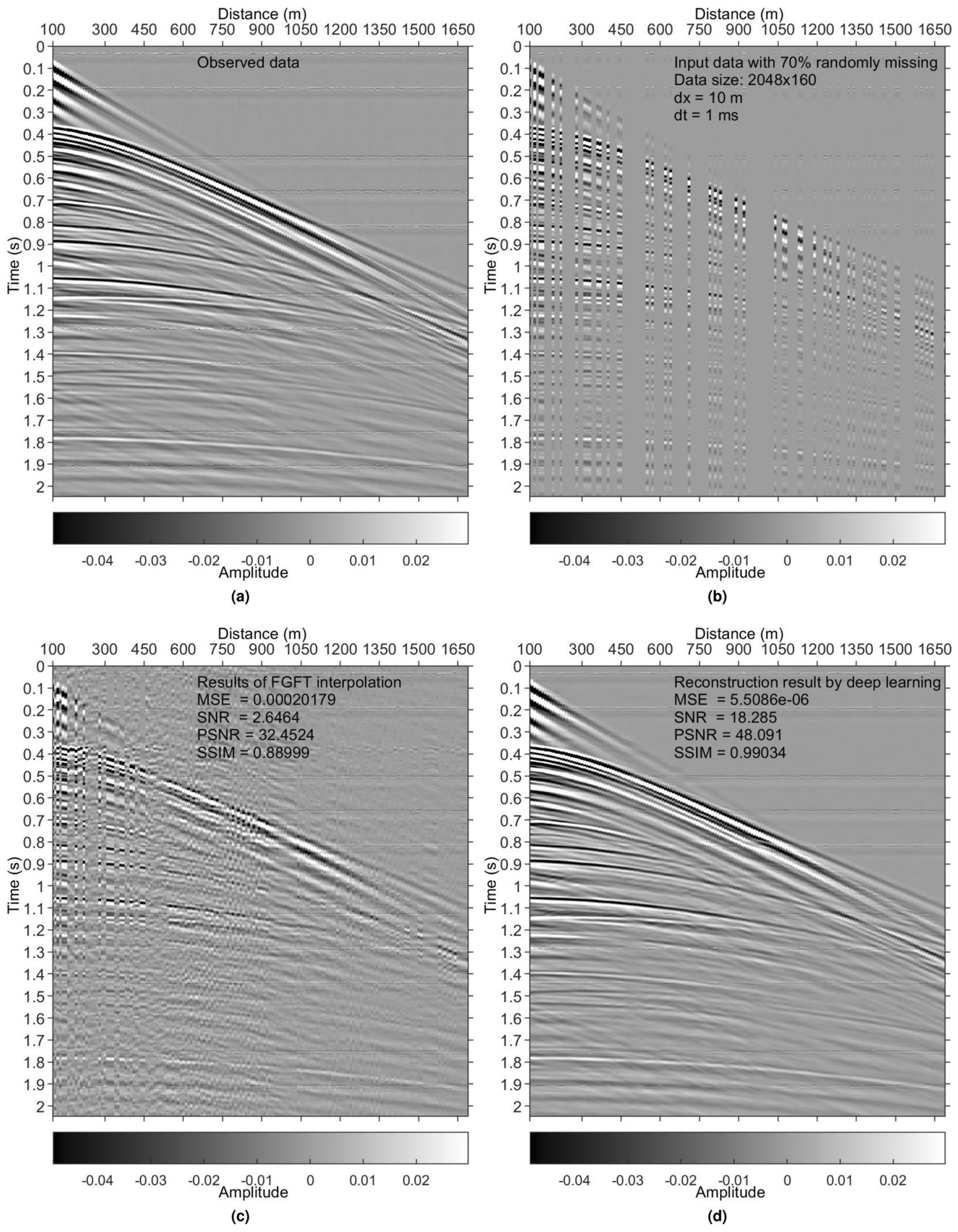

Figure 9. Results on a physical modelling data set[30]. (**a**) Reference data. (**b**) Irregularly sampled data with 70% missing. (**c**) Interpolated data using a fast-generalized Fourier transform (FGFT) method[23]. (**d**) DL reconstruction result.

The synthetic training data are modelled with a forward-modelling code[29] based on some well-designed earth models (e.g., Fig. 3a). The model in Fig. 3b is used to generate the test data that are completely unseen in the training process. The sources and receivers are equally distributed from 0 to 2550 m with 10 m spacing. The source is shifted from the location of the first shot to the last one. The source and receiver depths are changed for different earth models and experiments. The simulated data include 2048 shots (8 simulations, 256 shots per simulation), and 256 traces per shot. There are 2048 samples along the time axis with a time interval dt of 0.5 ms. Because dt = 0.5 ms is barely used in industry, we revised the data of size 2048 × 256 × 2048 (arranged in the order of time axis, receiver axis, and shot axis) to three data sets: 1024 × 256 × 2048 (dt = 1 ms), 512 × 256 × 2048 (dt = 2 ms), and 256 × 256 × 2048 (dt = 4 ms), composing the synthetic training data. Figure 4 shows a sample.

We exploited the Mobil Viking graben line 12 data set to generate the field training data. This data set is composed of 1001 shot gathers. Each gather is of size 1024 × 120: 1024 rows represent the time domain, sampled

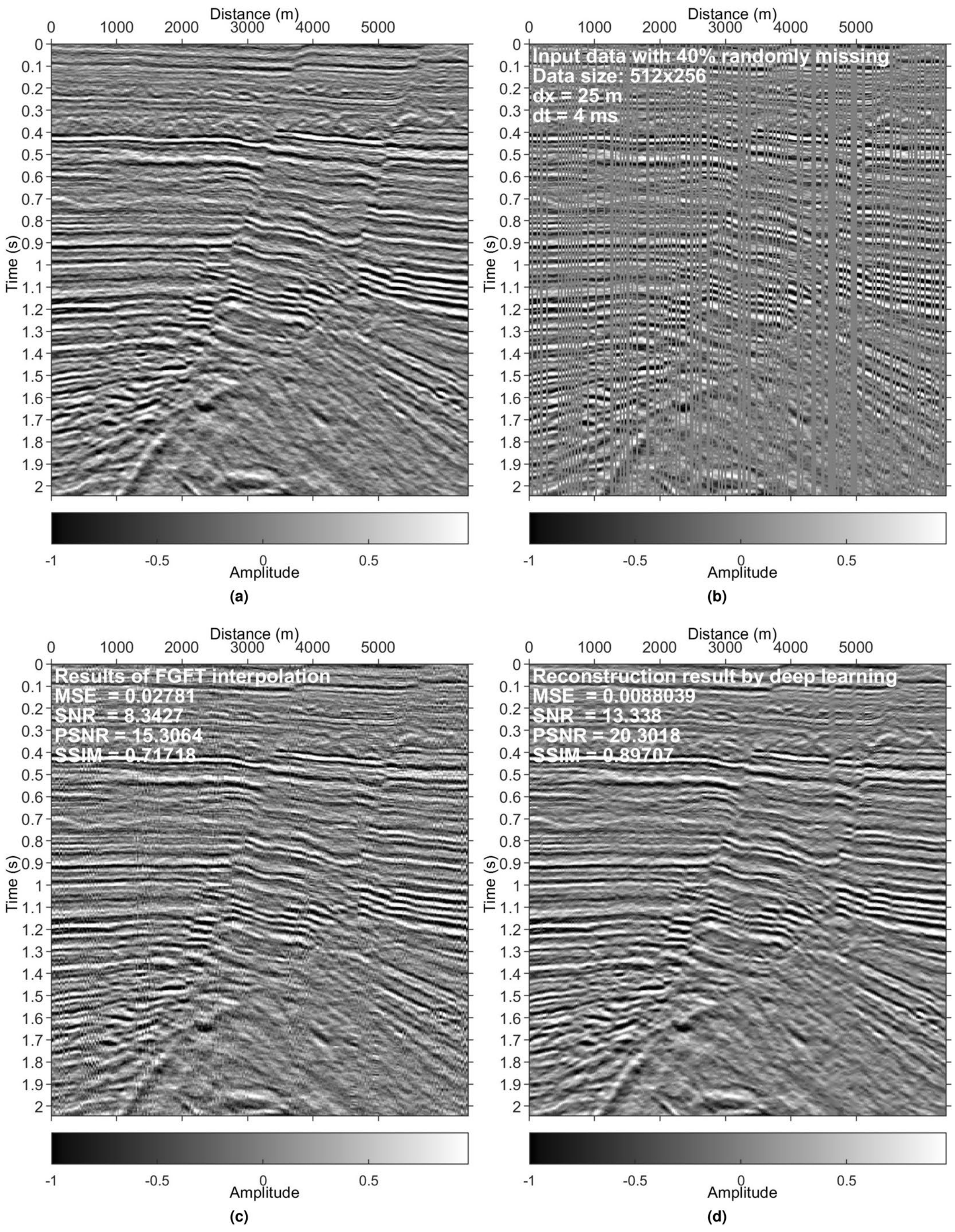

Figure 10. Results on a data set from the GeoFrame software. (**a**) Reference data. (**b**) Irregularly sampled data with 40% missing. (**c**) Interpolated data using the FGFT interpolation method[23]. (**d**) DL reconstruction result.

every 4 ms; 120 columns are in the spatial domain with 25 m of sampling. We randomly choose 200 shots as the field training data set (see Fig. 5 for a sample).

Before being fed into the model, each shot gather is normalized by dividing the maximum value of the absolute value of the corresponding shot gathers. Consequently, the amplitudes of the model input and output are finally in the range $[-1, 1]$. To ensure a sufficiently large number of data samples for learning, we work in a patch-wise fashion. The shot gathers are divided into small patches with a specified size. The training data in terms of patches is much larger than the number of training shot gathers.

Training setup. There is a trade-off between the patch size (determining the receptive field) and the model depth. A larger patch size demands more down- and up-sampling layers, while small patches allow the model to see only local features. In addition, we should select the patch size such that all 2×2 down-sampling operations can be applied to a layer with an even height and width size. The patch size and the batch size are primarily limited by the

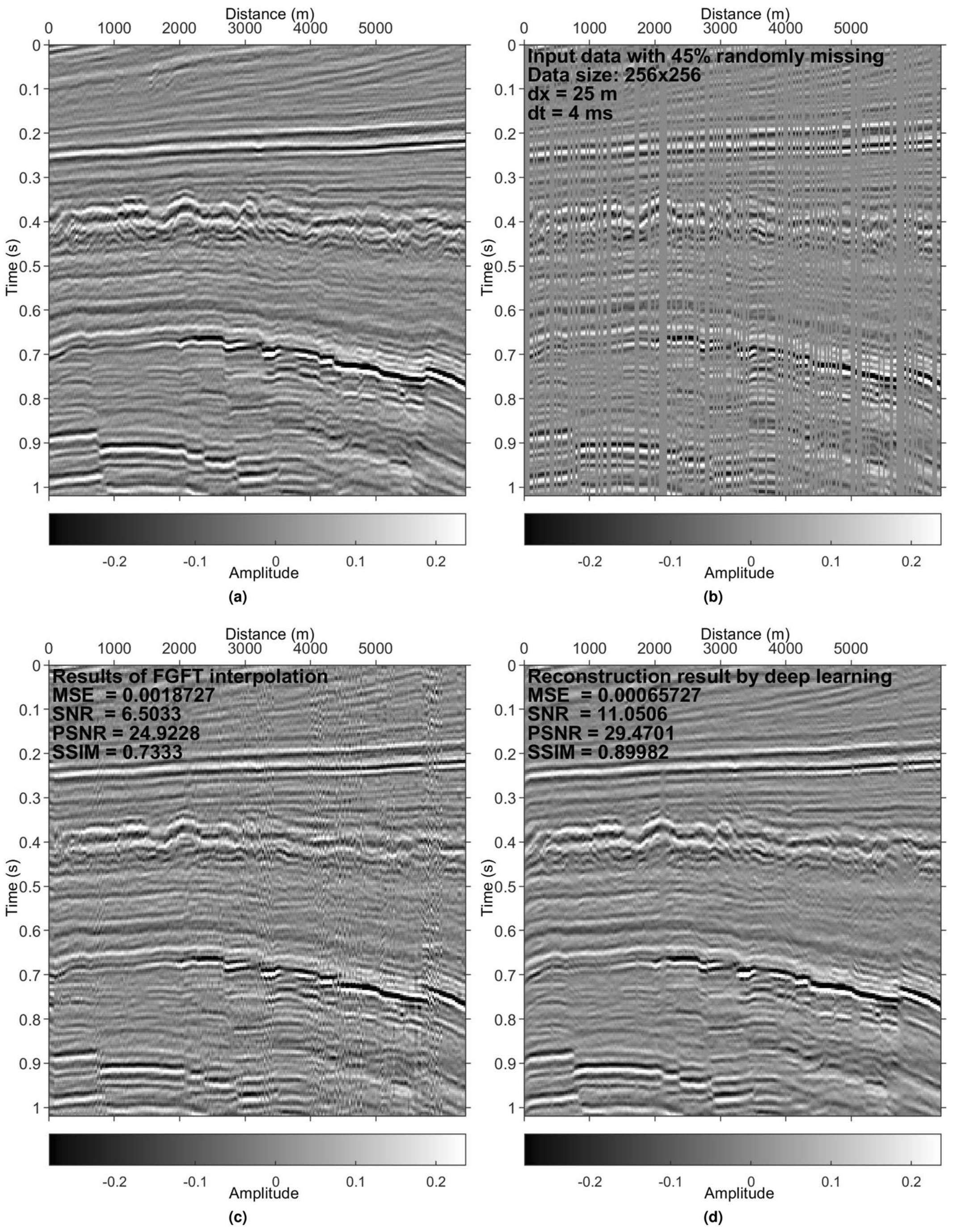

Figure 11. Results on the F3 data set. (**a**) Reference data. (**b**) Irregularly sampled data with 45% missing. (**c**) Interpolated data using the FGFT interpolation method[23]. (**d**) DL reconstruction result.

graphics processing unit (GPU) memory. To minimize the overhead and make full use of the GPU memory, we prefer a large patch size over a large batch size.

Our available computing resources are summarized as follows: a workstation with Windows 7, two Intel Xeon E5-2620 processors, 2.10 GHz CPU, 176 GB of RAM, and an NVIDIA GeForce RTX 2080 Ti GPU (11 GB). Our codes are written in Python based on Keras (a Python DL library). After trial and error, the patch size is set as 112×112, which allows four times of down-sampling operations for the field training shot gather with 120 traces. To overlap adjacent patches, the patch-stride is 23 pixels for the synthetic training data sets, and 10 pixels for the field training data set. Patches with a smaller mean absolute value (e.g., ≤ 0.001) indicate that there are few events located within the patch or these amplitude values are nearly zeros. The patches below a threshold value are removed from the training data. Then, we paid special attention to the first-arrival areas as their samples are fewer in comparison to other areas. As a result, we augment the proportion of the samples belonging to the shallow first-arrival areas to some degree. The training process finally involves 1,132,800 (more than 1 million)

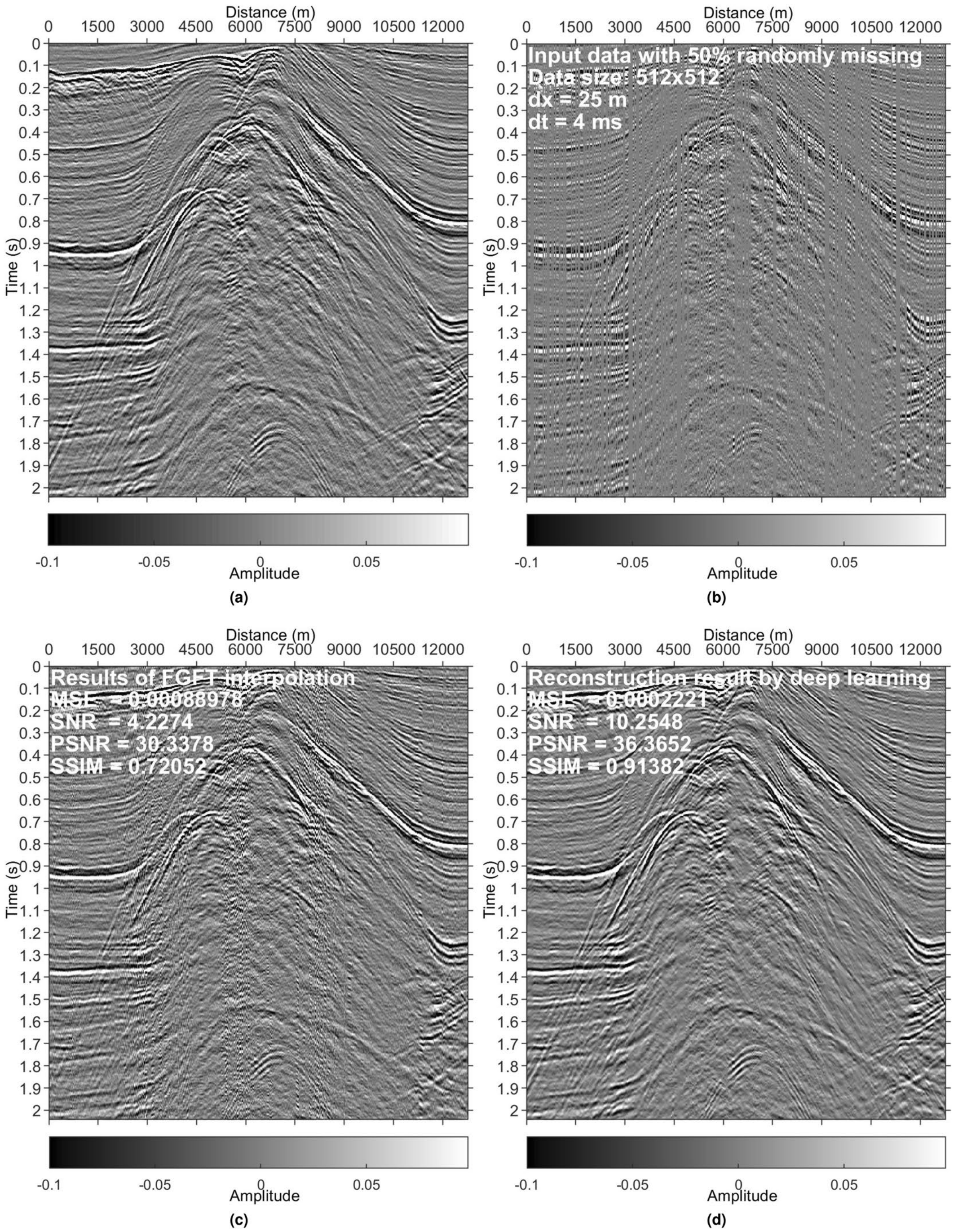

Figure 12. Results on a data set from the North Sea. (**a**) Reference data. (**b**) Irregularly sampled data with 50% missing. (**c**) Interpolated data using the FGFT interpolation method[23]. (**d**) DL reconstruction result.

patches. We set the batch size as 128. The steps-per-epoch is set as 8850, which denotes the total number of steps (batches of samples) before declaring one epoch finished and starting the next epoch. The steps-per-epoch is typically equal to the number of samples of the training data set (1,132,800) divided by the batch size (128). Figure 6 demonstrates fifty model input-output training pairs.

The speed provided by computation of the updates on small batches of data, in parallel, on specialized hardware (e.g., GPU) allows one to fit networks with millions of parameters on data sets with millions of observations. Good default settings of Adam for the tested DL problems are $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\varepsilon = 10^{-8}$. The learning rate α is initialized at 0.0001, which is a critical parameter. Nevertheless, determining how to obtain optimal values of the learning rate is still an open issue. The number of epochs should be specified to train the model. Too few epochs generate a poor under-fitting DL result, and too many epochs waste running time and possibly produce over-fitting results. With our experience, 50 epochs obtain sufficiently good results. Moreover, the indexes of the

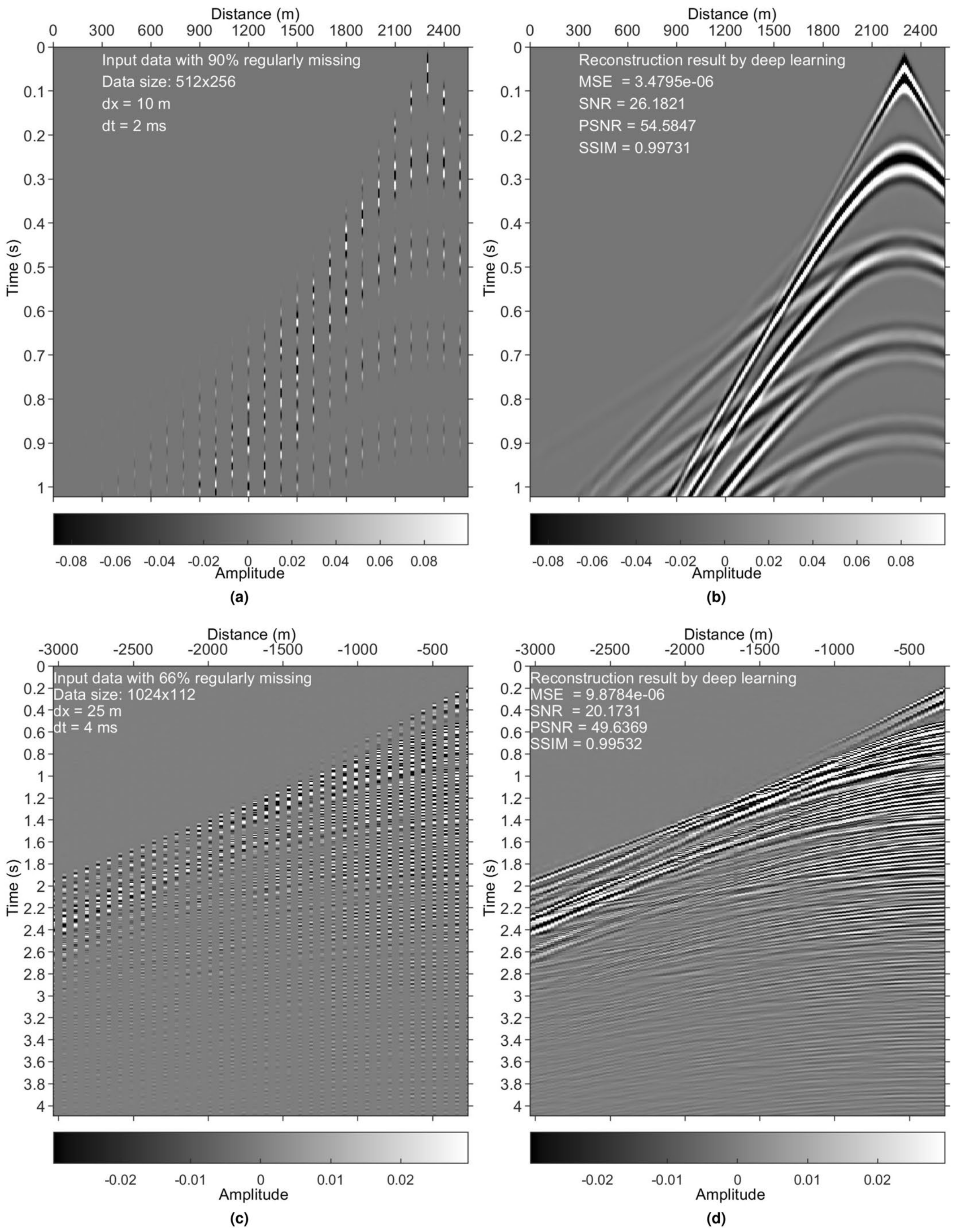

Figure 13. Regularly missing data reconstruction. (**a**) Regularly sampled data from the synthetic training data set with a decimation factor of 10 in the space direction (90% missing). (**b**) DL result for (**a**). (**c**) Regularly sampled data from the Mobil Viking graben line 12 data set with a decimation factor of 3 in the space direction (66% missing). (**d**) DL result for (**c**).

input-output pairs are shuffled before starting the next epoch. In the end, to ensure the original available data remain unaltered, the live data from the original input are reinserted into their original positions in the DL result.

Results

In the training process, the model input is restricted in a specified patch size (112 × 112). However, for the model test, the input data need not to be divided into small patches. That is, a profile can be directly fed into the model. One of the most important issues is convergence of the training process. The training log shown in Fig. 7 indicates convergence, which is successively going down with the increasing epoch numbers. For a well-trained model, it should produce reasonable output for new input that is never seen in the training process (aka the model's generalization capability). We exploit several typical data sets (e.g., those shown in Figures 5, 8–13) to test the generalization capacity. Because we work in a local patch-wise fashion in the training process, the complete feature of a shot gather belonging to the training data (e.g., Fig. 4) is not completely seen. Application of the trained model to irregularly missing data reconstruction is first on the agenda; then, regularly missing data reconstruction follows.

Irregularly missing data reconstruction. Examples in Figs. 4, 5, 8 and 9 validate that the trained model can reconstruct irregularly missing data with high accuracy. The model output, seen in Figs. 4 and 8, shows negligible difference between the true data and the DL result. To check the superiority of the new method over conventional methods, we compare it with a peer-reviewed FGFT interpolation method[23]. The code of the FGFT method is open-source. We made no change to the code of the FGFT method. There are few adjustable parameters of the FGFT method. The results of the FGFT method are correspondingly shown in Figs. 5, 9–12. Though the parameter may be not the optimal one, Figs. 5, 9–12 can show the drawbacks of the FGFT method to some degree. By comparing the FGFT interpolation results with those of DL, DL achieves smaller MSE values and higher SNR, PSNR, SSIM values. These results verify the feasibility, effectiveness, superiority, and generalization capacity of the evaluated method.

Comparing Figs. 4, 5, 8 and 9 (pre-stack data applications) with Figs. 10–12 (post-stack data applications) reveals that reduced precision emerges (reflected in the relatively lower SNR values in Figs. 10–12) if the features of the test data are significantly different from those of the training data. Note that the model is trained with pre-stack data only. The bias increases as the differences increase. Even though the performance decreases with the increasing feature difference between training and test data, the model still generates acceptable results in comparison to the FGFT results (see Figs. 10–12). We think different types of "reliable" data should be added to the training data to further improve the model's generalization capability.

Regularly missing data reconstruction. Our primitive goal is to accomplish the irregularly missing data reconstruction. We did not realize that the trained model is suitable for the regularly missing case. Excited by successful tests (two of them are shown in Fig. 13), we found that the evaluated framework is also competent in regularly missing data reconstruction. The reason is that regularly missing can be seen as a special case of irregularly missing. The model trained with irregularly sampled data can be applied to regularly missing data reconstruction. However, the model trained with regularly sampled data cannot be applied to irregularly missing data reconstruction.

Discussion

DL is a promising data-driven approach for solving inverse problems and, by extension, data reconstruction tasks. The model as established in this work may have tens to hundreds of millions of trainable parameters (see Table 1, approximately 87 million), giving rise to a large GPU memory requirement. The key computational cost of DL rests in the training process. However, it occurs once up front. The computational cost of model prediction is inexpensive. For example, the prediction of a 1024×112 shot gather costs less than 2 s on a computer without using the GPU. Hence, the overall computational cost is efficient.

Although we have concentrated on 2D, our method can be generalized to 3D/5D cases. A generalization to 3D demands substituting the 2D convolution/pooling/up-sampling layers with 3D versions, which is supported by numerous DL frameworks (e.g., Keras, TensorFlow, and PyTorch). We are moving towards 3D/5D reconstruction with the hope of obtaining superior results by using more spatial constraints. In this work, we have focused on missing data reconstruction, but the framework presented here also suggests similar potentials of DL in other fields (e.g., super-resolution reconstruction of photos and maps, signal processing, and imaging). Once a general model architecture is ready, the same idea can be applied to many problems.

Conclusions

We assessed a deep-learning-based framework for both irregularly and regularly missing data reconstruction, which is aimed at transforming incomplete data into their corresponding complete data. For achieving this goal, we first build a network architecture with the randomly sampled incomplete data as the model input and the corresponding complete data as the model output, which is based on an encoder-decoder-style end-to-end U-Net CNN. Then, we use a mean-squared-error loss function and an Adam optimization algorithm to train the model. Next, we prepare the training data utilizing both synthetic and field seismic data. We describe the established model architecture, the used loss function, the employed Adam optimization algorithm, the training data and the training setups in detail. We demonstrate the feature maps for a randomly sampled data set going through the trained model, with the aim of trying to explain how the missing data are reconstructed. We test the trained model with several typical data sets for irregularly missing data reconstruction, which achieves better performances compared with the FGFT interpolation method, verifying the feasibility, effectiveness, superiority, and generalization capability of the evaluated framework. Because regularly missing data can be considered as one special case of irregularly missing data, the trained model is also successfully applied to regularly missing data reconstruction. This work supports that DL can avoid some assumptions limiting conventional interpolation methods (e.g., assumptions of linear events, sparseness, and low-rank) and possesses great potential in advanced intelligent applications over traditional techniques.

Received: 10 September 2019; Accepted: 4 February 2020;
Published online: 24 February 2020

References

1. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436–444, https://doi.org/10.1038/nature14539 (2015).
2. Jordan, M. I. & Mitchell, T. M. Machine learning: Trends, perspectives, and prospects. *Science* **349**, 255–260, https://doi.org/10.1126/science.aaa8415 https://science.sciencemag.org/content/349/6245/255.full.pdf (2015).
3. Sarah, W. Deep learning for biology. *Nature* **554**, 555–557, https://doi.org/10.1038/d41586-018-02174-z (2018).
4. Falk, T. *et al.* U-Net: deep learning for cell counting, detection, and morphometry. *Nature Methods* **16**, 67–70, https://doi.org/10.1038/s41592-018-0261-2 (2019).
5. Zhu, B., Liu, J. Z., Cauley, S. F., Rosen, B. & Rosen, M. S. Image reconstruction by domain-transform manifold learning. *Nature* **555**, 487–492, https://doi.org/10.1038/nature25988 (2018).
6. Belthangady, C. & Royer, L. A. Applications, promises, and pitfalls of deep learning for fluorescence image reconstruction. *Nature Methods* **16**, 1215–1225, https://doi.org/10.1038/s41592-019-0458-z (2019).
7. Bergen, K. J., Johnson, P. A., de Hoop, M. V. & Beroza, G. C. Machine learning for data-driven discovery in solid Earth geoscience. *Science* **363**, https://doi.org/10.1126/science.aau0323 (2019).
8. Goodfellow, I., Bengio, Y. & Courville, A. *Deep Learning* http://www.deeplearningbook.org (MIT Press, 2016).
9. Schmidhuber, J. Deep learning in neural networks: An overview. *Neural Networks* **61**, 85–117, 1016/j.neunet.2014.09.003 (2015).
10. Wu, X., Liang, L., Shi, Y. & Fomel, S. FaultSeg3D: Using synthetic data sets to train an end-to-end convolutional neural network for 3D seismic fault segmentation. *Geophysics* **84**, IM35–IM45, https://doi.org/10.1190/geo2018-0646.1 (2019).
11. Wang, Z., Di, H., Shafiq, M. A., Alaudah, Y. & AlRegib, G. Successful leveraging of image processing and machine learning in seismic structural interpretation: A review. *The Leading Edge* **37**, 451–461, https://doi.org/10.1190/tle37060451.1 (2018).
12. Röth, G. & Tarantola, A. Neural networks and inversion of seismic data. *Journal of Geophysical Research: Solid Earth* **99**, 6753–6768, https://doi.org/10.1029/93JB01563, https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/93JB01563 (1994).
13. Jia, Y. & Ma, J. What can machine learning do for seismic data processing? An interpolation application. *Geophysics* **82**, V163–V177, https://doi.org/10.1190/geo2016-0300.1 (2017).
14. Jia, Y., Yu, S. & Ma, J. Intelligent interpolation by Monte Carlomachine learning. *Geophysics* **83**, V83–V97, https://doi.org/10.1190/geo2017-0294.1 (2018).
15. Wang, B., Zhang, N., Lu, W. & Wang, J. Deep-learning-based seismic data interpolation: A preliminary result. *Geophysics* **84**, V11–V20, https://doi.org/10.1190/geo2017-0495.1 (2019).
16. Isola, P., Zhu, J., Zhou, T. & Efros, A. A. Image-to-image translation with conditional adversarial networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 5967–5976, https://doi.org/10.1109/CVPR.2017.632 (2017).
17. Abma, R. & Kabir, N. 3D interpolation of irregular data with a POCS algorithm. *Geophysics* **71**, E91–E97, https://doi.org/10.1190/1.2356088 (2006).
18. Trad, D. Five-dimensional interpolation: Recovering from acquisition constraints. *Geophysics* **74**, V123–V132 (2009).
19. Ma, J. Three-dimensional irregular seismic data reconstruction via low-rank matrix completion. *Geophysics* **78**, V181–V192, https://doi.org/10.1190/geo2012-0465.1 (2013).
20. Naghizadeh, M. & Sacchi, M. Multidimensional de-aliased Cadzow reconstruction of seismic records. *Geophysics* **78**, A1–A5, https://doi.org/10.1190/geo2012-0200.1 (2013).
21. Wang, L. & Wang, Y. A joint matrix minimization approach for seismic wavefield recovery. *Scientific Reports* **8**, 2188 (2018).
22. Sacchi, M. D., Ulrych, T. J. & Walker, C. J. Interpolation and extrapolation using a high-resolution discrete Fourier transform. *IEEE Transactions on Signal Processing* **46**, 31–38 (1998).
23. Naghizadeh, M. & Innanen, K. A. Seismic data interpolation using a fast generalized Fourier transform. *Geophysics* **76**, V1–V10, https://doi.org/10.1190/1.3511525 (2011).
24. David, E., Rumelhart, R. J. W. & Geoffrey, E. H. Learning representations by back-propagating errors. *Nature* **323**, 533–536, https://doi.org/10.1038/323533a0 (1986).
25. He, K., Zhang, X., Ren, S. & Jian, S. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778 (2016).
26. Kingma, D. & Ba, J. Adam: A method for stochastic optimization. *Computer Science* (2014).
27. Duchi, J. C., Hazan, E. & Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* **12**, 2121–2159 (2011).
28. Wang, Z., Bovik, A. C., Sheikh, H. R. & Simoncelli, E. P. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing* **13**, 600–612, https://doi.org/10.1109/TIP.2003.819861 (2004).
29. Chen, H., Zhou, H., Zhang, Q., Xia, M. & Li, Q. A *k*-space operator-based least-squares staggered-grid finite-difference method for modeling scalar wave propagation. *Geophysics* **81**, T45–T61, https://doi.org/10.1190/geo2015-0090.1 (2016).
30. Chai, X., Wang, S., Wei, J., Li, J. & Yin, H. Reflectivity inversion for attenuated seismic data: Physical modeling and field data experiments. *Geophysics* **81**, T11–T24, https://doi.org/10.1190/geo2015-0250.1 (2016).

Acknowledgements

This work was supported by the National Natural Science Foundation of China Program (Grant nos. 41774143, 11805166, 41974154) and the Great and Special Project (Grant no. 2016ZX05026001-01). We thank Prof. Jianxin Wei at the China University of Petroleum (Beijing) for providing the physical modelling data. We thank the data contributors of the Mobil Viking graben line 12 data set and the F3 data set, which are available at an open data website (https://wiki.seg.org/wiki/Open_data). We thank Prof. Jianwei Ma for providing the data set from the GeoFrame software and a section of the North Sea data set. We thank the authors of the FGFT interpolation method for making their codes open-access (https://www.crewes.org/ResearchLinks/FreeSoftware/). We thank SMAART JV for the used Sigsbee2B and Pluto 1.5 earth models. We thank the authors of the Marmousi2 model. We acknowledge Dr. Hanming Chen at the China University of Petroleum (Beijing) for providing the used seismic forward modelling codes. We thank Junyong Yu, Jiankun Jing, and Jinhan Zhang at the China University of Geosciences (Wuhan) for help with proofreading.

Author contributions

X.C., K.L., H.G. and F.L. designed the research. X.C., K.L., H.D. and X.H. implemented the algorithm. F.L., H.D. and X.H. collected and processed the data. X.C., K.L., H.G. and F.L. designed, conducted and analysed the experiments. All authors participated in preparing the manuscript and provided significant input to the final manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to H.G.

Reprints and permissions information is available at www.nature.com/reprints.