A MULTIDIRECTIONAL DEEP NEURAL NETWORK FOR SELF-SUPERVISED RECONSTRUCTION OF SEISMIC DATA

Mohammad Mahdi Abedi[1], David Pardo[2,1,3]
[1]Basque Center for Applied Mathematics, Bilbao, Spain
[2]University of the Basque Country, Department of Mathematics, Spain
[3]Ikerbasque, Basque Foundation for Science, Bilbao, Spain
Emails: mabedi@bcamath.org, david.pardo@ehu.es

ABSTRACT

Seismic studies exhibit gaps in the recorded data due to surface obstacles. To fill in the gaps with self-supervised deep learning, the network learns to predict different events from the recorded parts of data and then applies it to reconstruct the missing parts of the same dataset. We propose two improvements to the task: a rearrangement of the data, and a new deep-learning approach. We rearrange the traces of a 2D acquisition line as 3D data cubes, sorting the traces by the source and receiver coordinates. This 3D representation offers more information about the structure of the seismic events and allows a coherent reconstruction of them. However, learning the structure of events in 3D cubes is more complicated than in 2D images while the size of the training dataset is limited. Thus, we propose a specific architecture and training strategy to take advantage of 3D data samples, while benefiting from the simplicity of 2D reconstructions. Our proposed multidirectional convolutional neural network has two parallel branches trained to perform 2D reconstructions along the vertical and horizontal directions and a small 3D part that combines their results. We use our method to reconstruct data gaps resulting from several missing shots in a benchmark synthetic and a real land dataset. Compared to a conventional 3D U-net, our network learns to reconstruct the events more accurately. Compared to 2D U-nets, our method avoids the discontinuities that arise from the 2D reconstruction of each trace of the missing shot gathers.

Key words: Interpolation, Seismic, Deep learning.

1 Introduction

The presence of surface obstacles or other technical issues can result in gaps in the recorded seismic data. An accurate reconstruction of the missing data is essential for having uniform data available for key seismic applications such as imaging.

Reconstruction of the missing data parts from the existing data is based on assuming a form of continuity for specific seismic events, using simplifying assumptions about the events' shape, or finding such continuity from the existing data itself. Classical data reconstruction methods use prediction filters in the frequency domain (e.g., [1–3]), curvelet transform (e.g., [4, 5]), seislet transform (e.g., [6, 7]), Radon transform (e.g., [8–11]), and inversion (e.g., [12, 13]). Recent data reconstruction methods that use data-driven approaches include dictionary learning (e.g., [14,15]), support vector regression (e.g., [16, 17]), and deep learning (e.g., [18–24]).

Most deep learning approaches are trained in a supervised fashion. In seismic studies, a labeled training dataset is usually generated synthetically (e.g., [25–27]). The shape of the events and the relation between the neighboring traces are dependent on the earth model and acquisition design, with highly varying parameters from one acquisition site to another. Therefore, for the missing data reconstruction task, such synthetic training dataset should be generated

[0]This is the final version. This paper is pubished in IEEE Transactions on Geoscience and Remote Sensing, doi: 10.1109/TGRS.2022.3227212

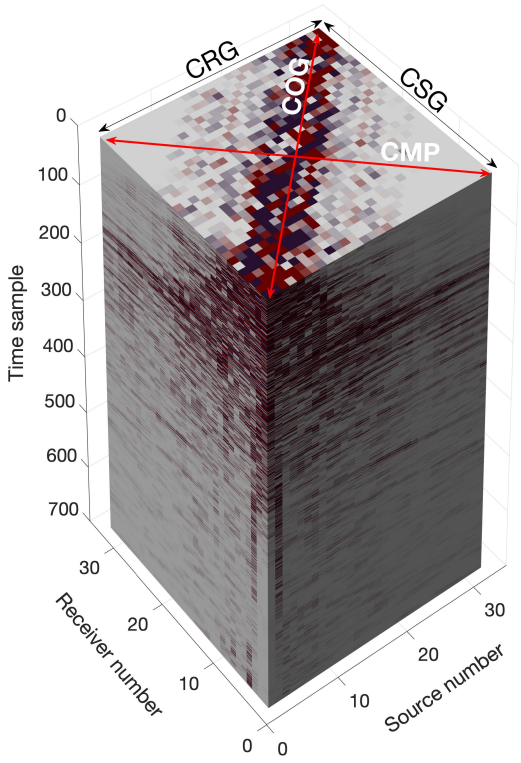

Figure 1: Arranging the raw recorded traces of a 2D acquisition line based on the source and receiver location, a 3D representation is obtained. In different directions, we have common shot gathers (CSGs), common receiver gathers (CRGs), common offset gathers (COGs), and common midpoint gathers (CMPs).

considering the parameters of the target real data. Considering the target-oriented nature of the problem and the difficulties of generating representative synthetic data, recently unsupervised or self-supervised deep learning has gained attention in different seismic applications (e.g., [22, 28–32]). In self-supervised deep learning for data reconstruction, a neural network is trained to learn the structure of events from available data, and then use it to predict the missing data.

Recorded seismic traces can be arranged into different gatherings, including common shot gathers (CSGs), common receiver gathers (CRGs), common offset gathers (COGs), and common midpoint gathers (CMPs). For each gathering type, the recorded events in a trace can have a different relationship with the neighboring traces. The shape and the continuity of such events can vary in each data gathering type. For example, recorded noise from an external source that appears as a coherent event in a CSG can become random in a CRG [33]. Therefore, for reconstructing the missing or corrupted parts of data, which rely on the predictability of events, it is beneficial to use all these arrangements of the data for learning the continuity of different events. Ordering the recorded traces from a 2D acquisition according to the shot and receiver coordinates, we obtain a 3D cube of data that comprises the aforementioned data gatherings in specific directions. Fig. 1 shows an example of such 3D arrangements of a 2D field dataset. In this example, the source and receiver intervals are identical, therefore, the COG and CMP appear in diagonal directions. For unidentical source and receiver intervals, the COG and CMP still exist in the data cube but are not necessarily diagonal. Using a 3D arrangement of traces similar to Fig. 1 in a data-driven approach, the shapes and continuity of events in different directions can be learned as 3D surfaces, resulting in a more robust data reconstruction.

We rearrange the recorded traces of a 2D dataset in 3D, and use self-supervised deep learning to reconstruct the missing shot gathers in the data. First, we try a conventional deep neural network (DNN) and show its limitations in the training and inference stages. Next, we propose a new architecture and an effective training strategy. We explain

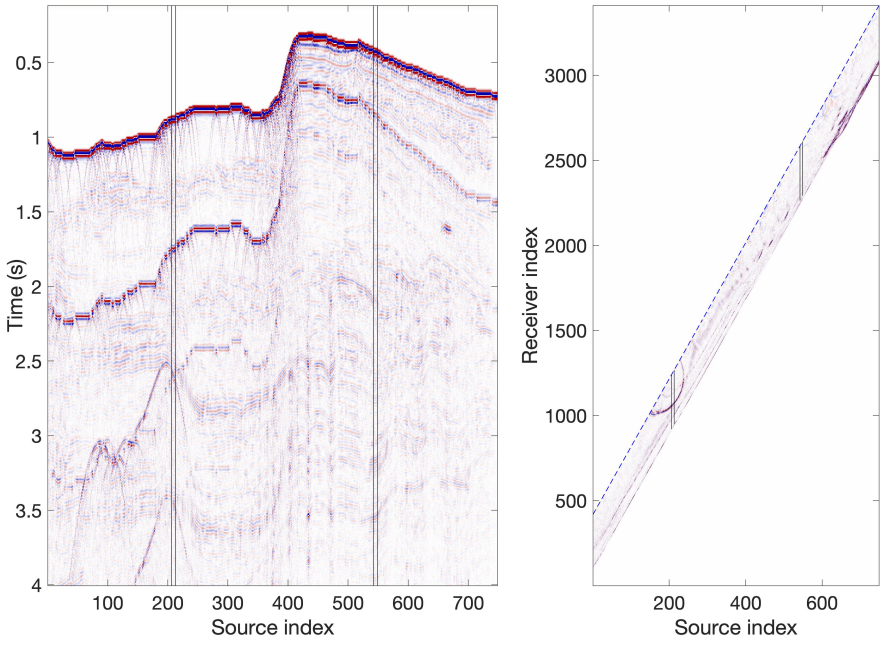

Figure 2: (a) A zero-offset section of the synthetic data. (b) A time-slice of rearranged data based on the source and receiver indices. The vertical lines show the location of two eight-shots gaps in the data. The dashed line in (b) shows the location of the section presented in (a).

our method and compare it with conventional methods, first, using a benchmark synthetic dataset, and then using a real land dataset.

2 Self-supervised data reconstruction

For self-supervised data reconstruction, we train a DNN on fully recorded parts of a dataset and then use the trained DNN to reconstruct the missing gaps in the same dataset.

The first dataset that we use is a part of the 2004 BP benchmark 2D synthetic data. Our data includes 748 shot gathers, each having 336 recorded traces for about 4s. The shot interval is 50m, the receiver interval is 12.5 m, and the time sampling interval is 6 ms. A section of the data is shown in Fig. 2a. For more information about the data, refer to [34]. We rearrange the data traces into a large cube by sorting them based on the source index on one axis and the receiver index on the other axis. Fig. 2b shows a horizontal time slice of the rearranged data. The dataset is normalized to the maximum absolute value in the entire data. Initially, we create data recording gaps at two randomly selected locations by removing eight consecutive shot gathers in each location, as shown in Fig. 2. Next, the dataset is decomposed into about 1800 overlapping segments with a size of 32 by 32 by 640, similar to data cubes in Fig. 3. Approximately 5% of the total data samples are randomly selected for validation. Data cubes that are centered over the induced gaps are separated as the test dataset. In the selection of the data segment size, there is a trade-off between the number of resulting data samples and the useful information in each sample that should be considered.

2.1 3D U-net

As a first attempt, we design the 3D U-net described in Fig. 4a. The general features of the network are selected following [35] and [36]. The network consists of convolutional layers that progressively downsample the data until reaching a latent space and then reverse the process until reproducing the input size in the output. Shortcut (skip) connections that connect layers of the same size allow the flow of low-level information to the latest layers. In our application, these shortcuts help to pass the known parts of the data to the output. The input of the network consists of the data cubes with zeroed shot gathers in the center of the cubes (Fig. 3a) and the output ground truth is the complete data cube (Fig. 3b). We progressively train the network to reconstruct missing shots, located at the center of the data

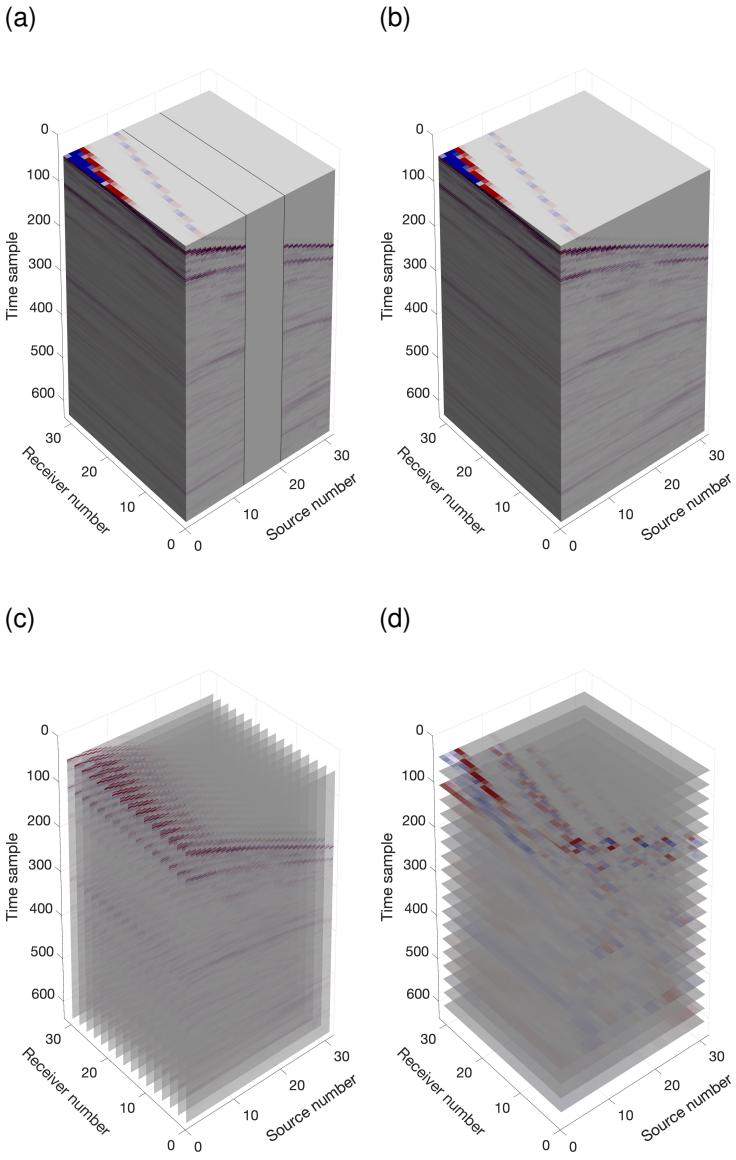

Figure 3: Examples of training data samples. (a) An arranged data cube with an induced gap by zeroing eight CSGs, which serves as the input of a DNN. (b) The complete data cube serves as the ground truth for the data in (a). (c) Representative vertical slices used to train the vertical 2D network. (d) Representative horizontal slices used to train the horizontal 2D network. The events appear flatter in CSGs due to the smaller receivers' interval.

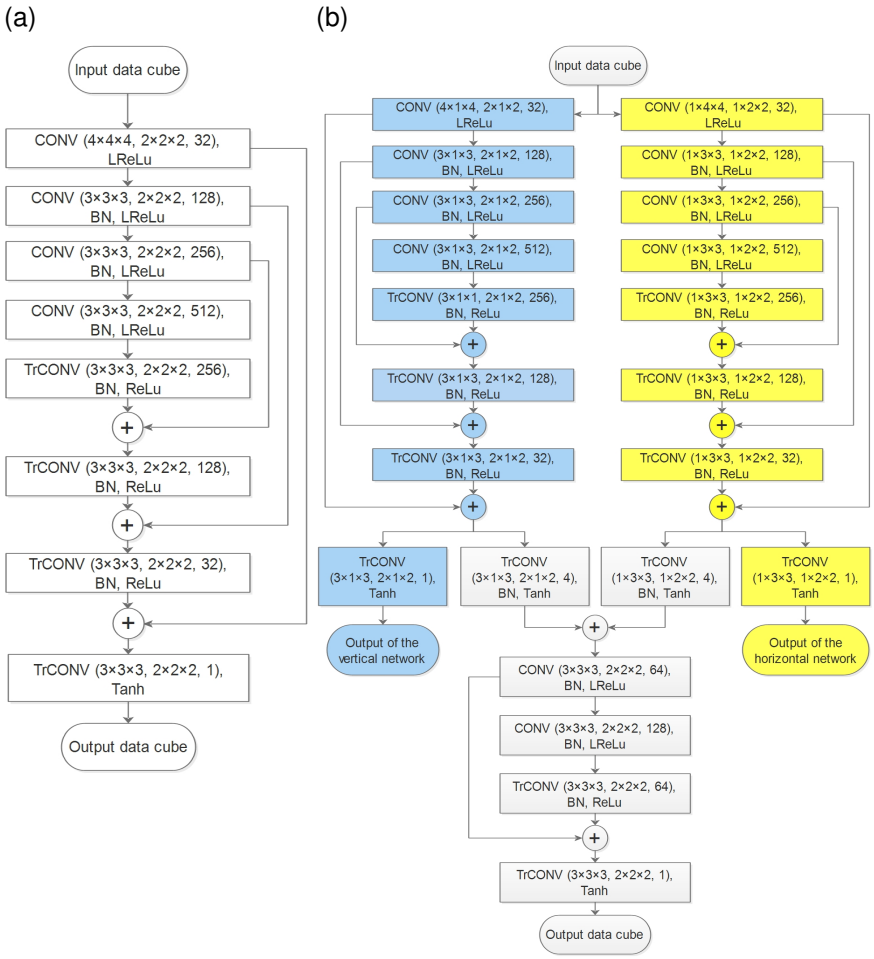

Figure 4: Architectures of (a) the 3D U-net and (b) the proposed multidirectional network. The networks are composed of convolutional (CONV), transposed convolution (TrCONV), batch normalization (BN), and concatenation (+) layers, plus the rectifier (ReLU), leaky ReLU (LReLU), and hyperbolic tangent (Tanh) activation functions. In parenthesis, we show the kernel size, stride size, and the number of filters, respectively.

sample, starting from one missing shot up to eight. The loss function we use is the sum of the mean absolute error (MAE) of the known and missing data, which are weighted separately by the number of data samples:

$$L = \frac{1}{N_m} \sum_{N_m} \left| d_{missing} - S(m)_{missing} \right| + \frac{1}{N - N_m} \sum_{N - N_m} \left| d_{known} - S(m)_{known} \right|. \quad (1)$$

where d is the complete data cube (ground truth) and $S(m)$ is the network output for the input cube m that has N_m missing data points, and N is the total number of data points in each training sample ($N = 640 \times 32 \times 32$ in 3D samples). The known data refer to the part of the input that is repeated in the output, and the missing data refer to the zeroed part of the data that the network reconstructs (see Fig. 3a and 3b). By separately normalizing the reconstruction errors in the known and missing data parts, the importance of errors in the missing part increases because of its smaller size compared to the known part. For the reconstruction tasks, the L1-norm is preferred over adversarial losses that can produce fake events [37].

We train the 3D U-net presented in Fig. 4a for different numbers of missing shots (from one to eight), using the ADAM optimizer with default parameters. The evolution of the reconstruction error is depicted in Fig. 5 in red. Increasing the number of missing shots in the center of data samples creates jumps in the reconstruction errors of the missing data (Fig. 5a). For a specific number of missing shots in the data, the loss decreases as the training progresses. However, as the number of missing shots increases the accuracy of the reconstructed missing data decreases. This is expected

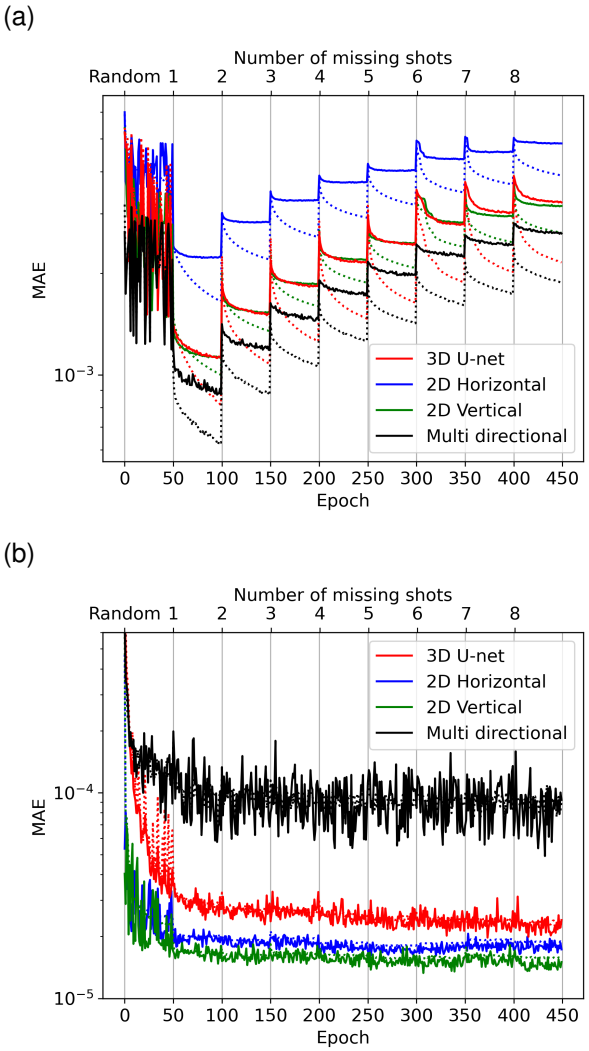

Figure 5: Mean absolute error of the reconstructed (a) missing data, and (b) known data, at the end of each epoch. The number of missing shots at the center of the data samples increases from one to eight resulting in jumps in the errors. The solid and dotted lines show the validation and training reconstruction loss, respectively. The vertical axis is logarithmic.

because in larger gaps only the events that are predictable over a larger span can be reconstructed. This can also be observed in the MAE of the reconstructed full shot gathers from the test data (the location of these shot gathers is shown in Fig. 2), as presented in Table 1. The results in Table 1 reflect the best performance of the networks, since continuing to train for more epochs (we tested for another 50) does not improve the test results. The reconstruction error of the known parts (Fig. 5b) is orders of magnitude smaller than that of the missing parts, therefore, it has an insignificant effect on the total loss.

We found that if we start the training with no missing shots and then add one or two missing shots, the network often minimizes only the error of the known data, performing poorly on the data reconstruction task. To avoid that, initially, we train the network for a random number of missing shots before starting to train for the reconstruction of one missing shot. As Fig. 5 shows, the networks mainly learn to reconstruct the known part in this stage.

We applied a data augmentation technique by randomly shifting the data up to eight time-samples in each training step. Other augmentation techniques that change the relationship between the neighboring traces, like rotating the data cubes, adversely affected the training.

Learning the structure of events as 3D surfaces is a complicated task. As Fig. 6 shows, the 3D U-net is unable to reconstruct some events (e.g., the diffractions), which leads to larger errors in Table 1. Besides, there is a large gap

between the training and validation losses (presented in Fig. 5a in red). We checked several modifications of the hyperparameters of the U-net, including: decreasing the trainable parameters by decreasing the width or depth of the network, removing the short-cut connections in the network, increasing the number of features in the deepest latent space, and using higher norms of the misfit in the loss function. However, none of these modifications improved the reconstruction accuracy in the test dataset. The network in Fig. 4a has about 10 million parameters. Decreasing the number of parameters by decreasing the number of kernels in each layer reduces the learnable features, resulting in a smaller gap between training and validation but also in a higher validation loss. In the following, we present a method to simultaneously reduce the number of model parameters and the reconstruction loss by converting a complicated 3D reconstruction task into simpler 2D reconstructions merged in a single network.

2.2 Multidirectional deep neural network

Our arrangement of the data traces in a cube based on the source and receiver positions (Fig. 1) provides a data sample with different predictability of missing events in 2D slices along different directions. Learning the structure of events in 2D is simpler than in 3D. 2D slices of the data cubes provide a greater number of samples, and a 2D network has about one-third of the trainable parameters of the equivalent 3D network. However, using 2D slices along one specific direction only allows the reconstruction of missing data for events that are predictable along that direction. To use the predictability of events in 3D and benefit from the better balance between the number of model parameters and data samples in 2D, we propose the network shown in Fig. 4b. This network has two parallel branches that are trained to perform 2D reconstructions along vertical and horizontal directions, connected with a small 3D part that combines the results of the 2D branches. The 3D U-net in Fig. 4a has 43% more trainable parameters than the multidirectional network in Fig. 4b.

First, we decompose the training and validation data cubes into vertical 2D slices as shown in Fig. 3c. Eliminating the repetitive data slices (due to the overlapping of data cubes), we obtain about 30,000 data samples of size 640 by 32. We obtain a 2D version of the 3D U-net using 2D vertical kernels (the part of Fig. 4b colored in blue). We then, follow the training strategy explained in the previous section, but in 2D. In 2D training, the data slices in each mini-batch are randomly selected from the entire dataset rather than from one data cube. The shuffling of data samples for a random selection of mini-batches is important for robust training.

Fig. 5 shows in green the evolution of reconstruction loss for the vertical 2D network. Table 1 presents the MAE of the reconstructed shot gathers from the test data. An example of the reconstruction of missing parts from the test data is shown in Fig. 6. The 2D vertical network has reconstructed the events that are predictable in vertical slices, but it has missed others. Examples of successful and unsuccessful reconstructions are marked in Fig. 6 with green and red arrows, respectively. Moreover, the 2D vertical network has produced horizontal discontinuities in the reconstructed shot gathers. This occurs because this network employs vertical 2D slices orthogonal to the shot gathers, and independently reconstructs each trace in the common shot gathers (see Fig. 6a).

Next, we decompose the training and validation data cubes into horizontal 2D slices of the data similar to Fig. 3d. Horizontal slices located before the first arrivals have no data and we eliminate them. After that, we obtain approximately 940,000 data samples of size 32 by 32. We obtain a 2D version of the 3D U-net using 2D horizontal kernels (the yellow part of Fig. 4b) and perform the 2D training strategy with shuffled mini-batches.

Fig. 5 shows in blue the evolution of reconstruction loss for the horizontal 2D network. Note that since we removed the zero-valued slices from the horizontal 2D dataset, the losses should not be compared with those of the vertical 2D. As shown in Table 1, the MAEs of the reconstructed common shot gathers from the test data are within the same level as those of the vertical 2D network. An example of the reconstruction of missing shot gathers is shown in Fig. 6, with marked examples of successful and unsuccessful reconstruction. The horizontal 2D network has reconstructed the events that are predictable in horizontal slices but has missed others. The 2D horizontal network has produced vertical discontinuities in the reconstructed shot gathers because it works with horizontal 2D slices orthogonal to the shot gathers.

Finally, we train the full multidirectional network (Fig. 4b). For each number of missing shots, we embed the corresponding trained vertical and horizontal branches in the network, remove their last layer, and add the small 3D network to their outputs. Highly accurate reconstruction of the known parts of the data by both branches (Fig. 5b) supplies the known data to the 3D part of the network. We train the new model using the same data cubes and training strategy explained in the previous section.

Fig. 5a shows the evolution of reconstruction loss of the missing data for the multidirectional network (in black). Both the training and validation errors and the gap between the training and validation are lower than in the 3D U-net graphs (note the logarithmic scale). Looking at the examples of the reconstruction of missing parts shown in Fig. 6, the events that were successfully reconstructed by either of the 2D networks are also present in the results produced by

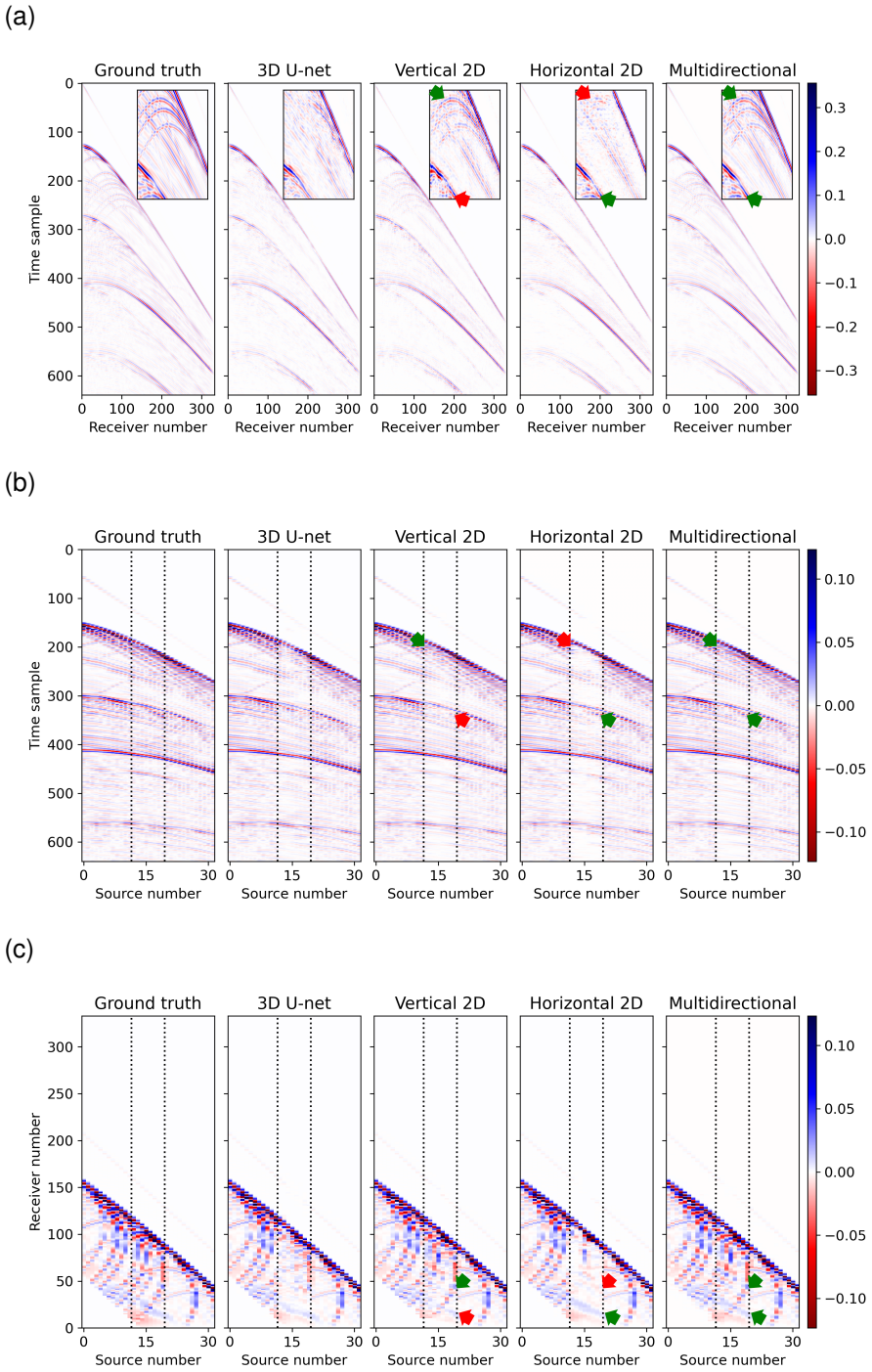

Figure 6: Comparison of the reconstructed test data (the left gap shown in Fig. 2) by different DNN approaches. a) Reconstructed common shot gathers at the middle of eight missing shots, and a zoomed part in overlapping panels. b) Vertical slices of the DNN outputs along the CRG direction. c) Horizontal slices of the DNN outputs. The green and red arrows indicate successful and unsuccessful reconstructions, respectively. In (b) and (c), the reconstructed missing data are between the vertical dotted lines.

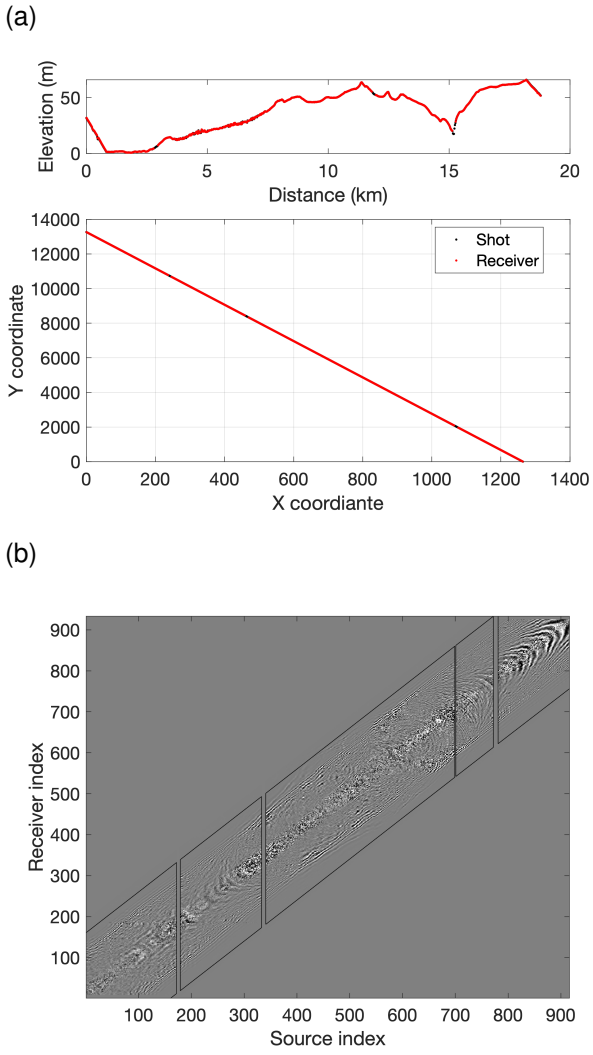

Figure 7: (a) Acquisition setting of the field data. (b) A time slice of the rearranged traces based on the shot and receiver coordinates. The location of missing shots can be seen.

the multidirectional network. It shows the ability of the network in combining the previously learned features along horizontal and vertical directions. The multidirectional network does not have the vertical or horizontal discontinuity problem of the 2D networks because it employs 3D samples. The MAE of the reconstructed shot gathers from the test data (Table 1) shows the higher accuracy of the proposed multidirectional network compared to the 3D and 2D U-nets.

The self-supervised training is devoted to a particular dataset. The network should ideally learn the detailed structures of events from the available data and extend it to the missing parts that are expected to exhibit similar structures. For simpler or smaller datasets, we suggest decreasing the number of kernels in all layers by a specific factor. Reducing the number of kernels by a factor of two reduces the number of model parameters by approximately a factor of four. The objective is to minimize the validation loss and reconstruction error of all the objective events. For self-supervised training, the training cost is also an important factor. Unfortunately, the training cost of our proposed multidirectional network is approximately three times higher than that of the 3D U-net. It is also affected by the size of the data and mini-batch.

3 Application to field data

The field data example we use is a 2D line of land seismic data recorded using vibroseis as source and geophones as receivers. The survey line extends for about 18 km, and includes 900 shots each recorded by 352 receivers in a split-spread pattern. The shot and receiver intervals equal 20 m, and the time sampling interval equals 4 ms. The

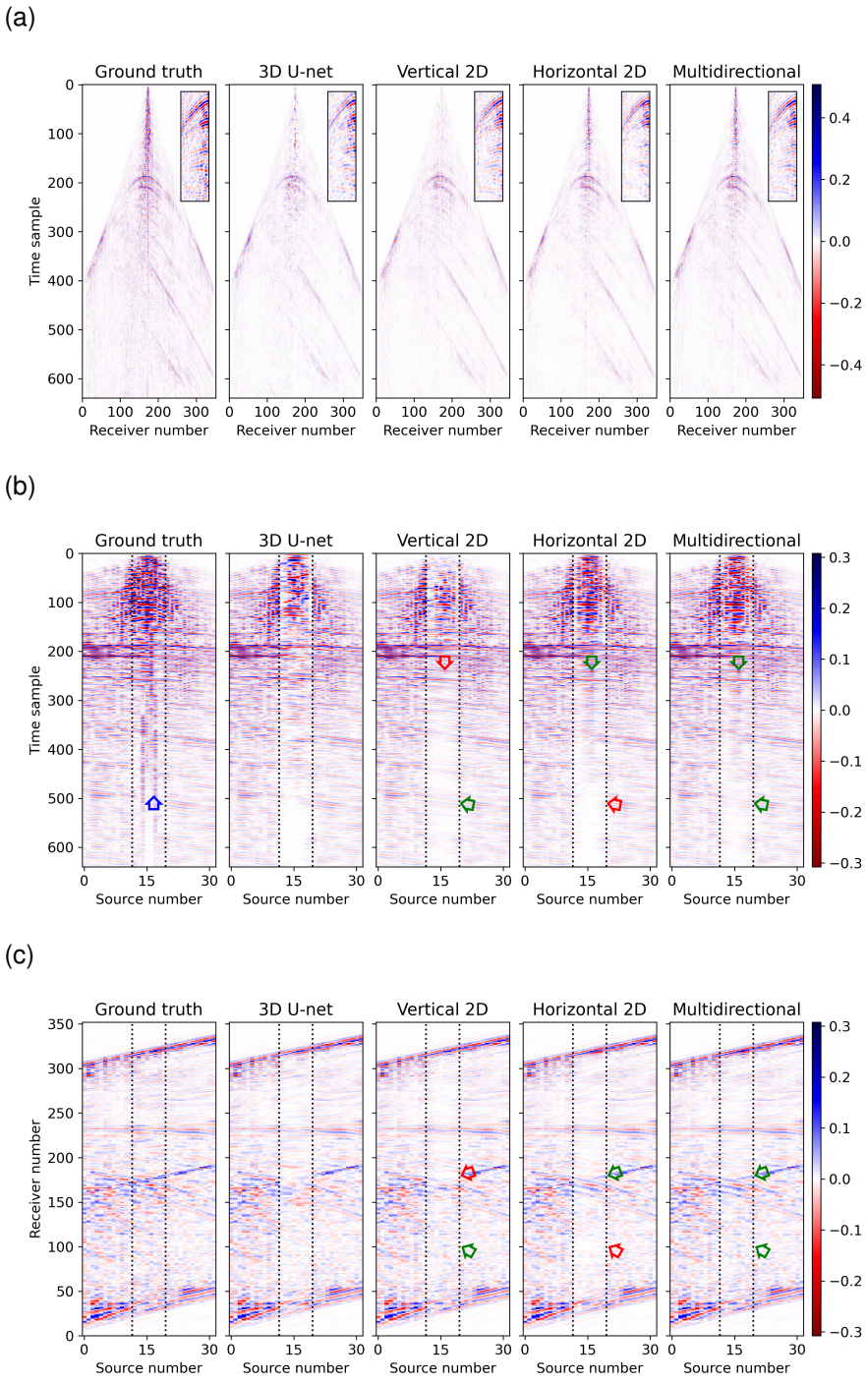

Figure 8: Comparison of the reconstructed validation data example by different DNN approaches. (a) Representative examples of the reconstructed common shot gathers from eight consecutive missing shots. (b) Vertical slices of the DNN outputs along the CRG direction. (c) Horizontal slices of the DNN outputs. The green, red, and blue arrows indicate successful reconstruction, unsuccessful reconstruction, and isolated events, respectively.

Table 1: Mean absolute error of reconstructed data gaps in Figure 2 (from the test synthetic data), after gathering data segments into common shot gathers. The values should be multiplied by 10^{-3}.

Missing shots	1	2	3	4	5	6	7	8
3D U-net	1.1	1.5	1.8	2.1	2.3	2.6	2.9	3.1
2D vertical	1.1	1.5	1.8	2.0	2.3	2.5	2.7	2.9
2D horizontal	1.2	1.5	1.8	2.0	2.3	2.5	2.6	2.8
Multidirectional	0.9	1.1	1.3	1.6	1.7	1.9	2.0	2.2

Table 2: Mean absolute error of reconstructed missing parts of the validation real example, after gathering data segments into common shot gathers. The values should be multiplied by 10^{-3}.

Missing shots	1	2	3	4	5	6	7	8
3D U-net	6.9	8.0	8.7	9.1	9.1	9.6	9.7	10.0
2D vertical	5.1	6.1	6.7	7.3	7.3	7.7	7.8	8.2
2D horizontal	4.8	6.0	6.5	7.1	7.2	7.7	7.7	8.2
Multidirectional	3.9	5.0	5.6	6.2	6.3	6.8	6.8	7.2

elevation and scaled coordinates of the shot and receiver points are plotted in Fig. 7a. Due to the presence of a few natural obstacles, there are four gaps in the data where between one to eight consecutive shots were not deployed. To reconstruct the missing data, we employ the proposed self-supervised deep learning using our multidirectional neural network.

We rearrange the recorded traces by sorting them based on the source and receiver coordinates. Fig. 7b shows a time slice of the resulting 3D arrangement of the data where the parts with missing shots can be seen. We prepare the training, validation, and test data samples by separating overlapping segments of the data in a similar way as in the aforementioned synthetic example. The validation data contains approximately 5% of the total data samples. Part of the validation is selected so that the segments build complete common shot-gathers when assembled together.

We follow the training strategy proposed in the previous section. We first train the vertical and horizontal branches of the proposed multidirectional network using 2D slices of the training data. Then, we embed the pretrained branches in the full network and train the entire network. Fig. 8 shows the reconstruction results of different deep-learning methods for part of the validation data. The MAE corresponding to different numbers of reconstructed common shot gathers from the validation data are presented in Table 2. The real data contains different types of noise that increase the reconstruction errors compared to the synthetic data example. Nonetheless, the proposed multidirectional method outperforms the 2D and 3D U-nets. Examples of events that are reconstructed by either of the 2D networks and successfully combined in the multidirectional network are marked with green arrows in Fig. 8. An event that falls entirely in the missing data gap in one direction is not reconstructed by the network that works in that direction. Isolated events that have no predictability in any direction are not reconstructed by any of the networks. An example of an isolated event (a processing artifact) is marked with a blue arrow in Fig. 8b.

Finally, we use the trained network to reconstruct the data gaps in the field data. We pass the test data cubes with actual missing parts to the network and fill the gaps with the reconstructed data. There are gaps with 1, 6, 7, and 8 missing shots; therefore, we use the networks trained to reconstruct the matching number of missing shots for each gap. Fig. 9a shows representatives of the reconstructed shot gather. The diversity of data at different locations of the acquisition line is noticeable. Two slices of the full data with zoomed portions in Fig. 9b show that the reconstructed data gaps properly blend in with the known data without any visible discontinuity.

4 Conclusion

Using a conventional 2D seismic gather (e.g. CRG) to reconstruct missing shot gathers restricts the learnable predictability and creates discontinuity when collecting independently reconstructed traces into a shot gather. These can be seen in the presented 2D reconstruction results. Rearranging a 2D dataset based on the source and receiver coordinates in orthogonal axes creates 3D data samples with robustly predictable seismic events in 3D. However, learning the structure of events in 3D cubes is a more complicated task than learning it in 2D images, therefore, the number of needed samples may be beyond what it is available to us. The best results that we obtained from a conventional 3D U-net still show inaccurate reconstructions in the test data and a large gap between training and validation losses.

We thus proposed a multidirectional neural network that combines the abilities of 2D and 3D learning. First, we train two networks to perform simpler 2D reconstructions in horizontal and vertical directions. There is also a better bal-

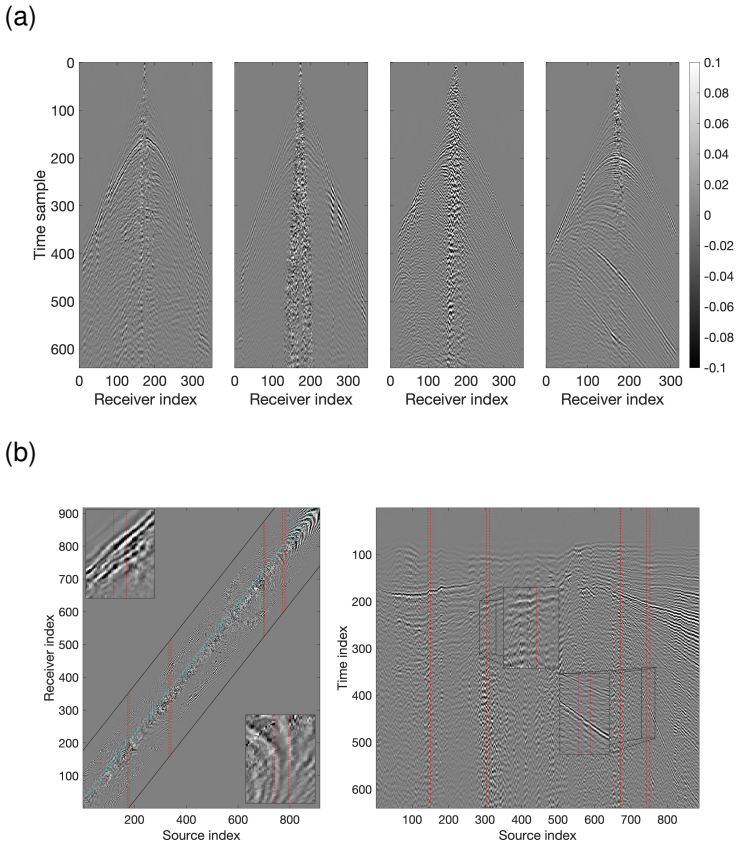

Figure 9: Application of the trained multidirectional network to reconstruct the missing parts of the data (shown in Fig. 7). a) Representatives of the reconstructed missing shots in the four missing gaps. b) A time slice and an offset slice of the full data, with zoomed regions. The dashed red lines show the filled gaps, and the cyan line in the time slice shows the location of the offset slice.

ance between the number of data samples and model parameters in 2D. Next, we use these networks as two parallel branches of one network to perform the reconstruction of 3D data samples. The resulting multidirectional network has fewer model parameters than the 3D U-net. It decreases the training and validation losses and the gap between them. It produces the lowest mean absolute error in the reconstructed shot gathers, as the presented tests on a benchmark synthetic data and a real field dataset demonstrate. It also avoids the induced discontinuities of the 2D U-nets because it together reconstructs patches of shot gathers in 3D segments. We train the networks progressively increasing the number of missing shot gathers. As the size of the data gap increases, the accuracy of all networks decreases because only the events that are predictable over a larger span can be reconstructed. A downside of our proposed multidirectional network is its training cost, which is approximately three times that of the 3D U-net.

Acknowledgments

This work has received funding from: the European Union's Horizon 2020 research and innovation program under the Marie Sklodowska-Curie grant agreement No 777778 (MATHROCKS); the Spanish Ministry of Science and Innovation projects with references TED2021-132783B-I00, PID2019-108111RB-I00 (FEDER/AEI) and PDC2021-121093-I00 (AEI/Next Generation EU), the BCAM Severo Ochoa accreditation of excellence (SEV-2017-0718); and the Basque Government through the BERC 2022-2025 program, the three Elkartek projects 3KIA (KK-2020/00049), EXPERTIA (KK-2021/00048), and SIGZE (KK-2021/00095), and the Consolidated Research Group MATHMODE (IT1456-22) given by the Department of Education. We acknowledge BP for making the 2D synthetic data available.

References

[1] Simon Spitz. Seismic trace interpolation in the fx domain. *Geophysics*, 56(6):785–794, 1991.

[2] Mostafa Naghizadeh and Mauricio D Sacchi. Multistep autoregressive reconstruction of seismic records. *Geophysics*, 72(6):V111–V118, 2007.

[3] Mostafa Naghizadeh and Kristopher A Innanen. Two-dimensional fast generalized fourier interpolation of seismic records. *Geophysical Prospecting*, 61:62–76, 2013.

[4] Felix J Herrmann, Deli Wang, Gilles Hennenfent, and Peyman P Moghaddam. Curvelet-based seismic data processing: A multiscale and nonlinear approach. *Geophysics*, 73(1):A1–A5, 2008.

[5] Gilles Hennenfent, Lloyd Fenelon, and Felix J Herrmann. Nonequispaced curvelet transform for seismic data reconstruction: A sparsity-promoting approach. *Geophysics*, 75(6):WB203–WB210, 2010.

[6] Shuwei Gan, Shoudong Wang, Yangkang Chen, Yizhuo Zhang, and Zhaoyu Jin. Dealiased seismic data interpolation using seislet transform with low-frequency constraint. *IEEE Geoscience and remote sensing letters*, 12(10):2150–2154, 2015.

[7] Wei Liu, Siyuan Cao, Shuwei Gan, Yangkang Chen, Shaohuan Zu, and Zhaoyu Jin. One-step slope estimation for dealiased seismic data reconstruction via iterative seislet thresholding. *IEEE Geoscience and Remote Sensing Letters*, 13(10):1462–1466, 2016.

[8] MM Nurul Kabir and DJ Verschuur. Restoration of missing offsets by parabolic radon transform 1. *Geophysical Prospecting*, 43(3):347–368, 1995.

[9] Juefu Wang, Mark Ng, and Mike Perz. Seismic data interpolation by greedy local radon transform. *Geophysics*, 75(6):WB225–WB234, 2010.

[10] Ali Gholami and Mauricio D Sacchi. Time-invariant radon transform by generalized fourier slice theorem. *Inverse Problems and Imaging*, 11(3):501, 2017.

[11] Mohammad Mahdi Abedi, Mohammad Ali Riahi, and Alexey Stovas. Three-parameter radon transform in layered transversely isotropic media. *Geophysical Prospecting*, 67(2):395–407, 2019.

[12] Shaohuan Zu, Hui Zhou, Yangkang Chen, Xiao Pan, Shuwei Gan, and Dong Zhang. Interpolating big gaps using inversion with slope constraint. *IEEE Geoscience and Remote Sensing Letters*, 13(9):1369–1373, 2016.

[13] Jing-Jie Cao, Gang Yao, and Nuno V da Silva. Interpolation of irregularly sampled noisy seismic data with the nonconvex regularization and proximal method. *Pure and Applied Geophysics*, 179(2):663–678, 2022.

[14] Mohammad Amir Nazari Siahsar, Saman Gholtashi, Vahid Abolghasemi, and Yangkang Chen. Simultaneous denoising and interpolation of 2d seismic data using data-driven non-negative dictionary learning. *Signal Processing*, 141:309–321, 2017.

[15] Hang Wang, Wei Chen, Quan Zhang, Xingye Liu, Shaohuan Zu, and Yangkang Chen. Fast dictionary learning for high-dimensional seismic reconstruction. *IEEE Transactions on Geoscience and Remote Sensing*, 59(8):7098–7108, 2020.

[16] Yongna Jia and Jianwei Ma. What can machine learning do for seismic data processing? an interpolation application. *Geophysics*, 82(3):V163–V177, 2017.

[17] Yongna Jia, Siwei Yu, and Jianwei Ma. Intelligent interpolation by monte carlo machine learning. *Geophysics*, 83(2):V83–V97, 2018.

[18] Benfeng Wang, Ning Zhang, Wenkai Lu, and Jialin Wang. Deep-learning-based seismic data interpolation: A preliminary result. *Geophysics*, 84(1):V11–V20, 2019.

[19] Xintao Chai, Hanming Gu, Feng Li, Hongyou Duan, Xiaobo Hu, and Kai Lin. Deep learning for irregularly and regularly missing data reconstruction. *Scientific reports*, 10(1):1–18, 2020.

[20] Shuhang Tang, Yinshuai Ding, Hua-Wei Zhou, and Heng Zhou. Reconstruction of sparsely sampled seismic data via residual u-net. *IEEE Geoscience and Remote Sensing Letters*, 19:1–5, 2020.

[21] Yingying Wang, Benfeng Wang, Ning Tu, and Jianhua Geng. Seismic trace interpolation for irregularly spatial sampled data using convolutional autoencoder. *Geophysics*, 85(2):V119–V130, 2020.

[22] Thomas André Larsen Greiner, Jan Erik Lie, Odd Kolbjørnsen, Andreas Kjelsrud Evensen, Espen Harris Nilsen, Hao Zhao, Vasily Demyanov, and Leiv-J Gelius. Unsupervised deep learning with higher-order total-variation regularization for multidimensional seismic data reconstruction. *Geophysics*, 87(2):V59–V73, 2022.

[23] Naihao Liu, Lukun Wu, Jiale Wang, Hao Wu, Jinghuai Gao, and Dehua Wang. Seismic data reconstruction via wavelet-based residual deep learning. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–13, 2022.

[24] Francesco Brandolin, Matteo Ravasi, and Tariq Alkhalifah. Pwd-pinn: slope-assisted seismic interpolation with physics-informed neural networks. In *Second International Meeting for Applied Geoscience & Energy*, pages 2646–2650. Society of Exploration Geophysicists and American Association of Petroleum , 2022.

[25] Shan Qu, Eric Verschuur, Dong Zhang, and Yangkang Chen. Training deep networks with only synthetic data: Deep-learning-based near-offset reconstruction for (closed-loop) surface-related multiple estimation on shallow-water field data. *Geophysics*, 86(3):A39–A43, 2021.

[26] H Hashemi, M Saadat, M Nabi Bidhendi, and P De Groot. Incorporating acquisition geometry in deep learning-based full waveform inversion. In *82nd EAGE Annual Conference & Exhibition*, volume 2021, pages 1–5. European Association of Geoscientists & Engineers, 2021.

[27] Mohammad Mahdi Abedi and David Pardo. Nonhyperbolic normal moveout stretch correction with deep learning automation. *Geophysics*, 87(2):U57–U66, 2022.

[28] Hosein Hashemi. Fuzzy clustering of seismic sequences: Segmentation of time-frequency representations. *IEEE Signal Processing Magazine*, 29(3):82–87, 2012.

[29] Claire Birnie, Matteo Ravasi, Sixiu Liu, and Tariq Alkhalifah. The potential of self-supervised networks for random noise suppression in seismic data. *Artificial Intelligence in Geosciences*, 2:47–59, 2021.

[30] Claire Birnie and Tariq Alkhalifah. Transfer learning for self-supervised, blind-spot seismic denoising. *arXiv preprint arXiv:2209.12210*, 2022.

[31] Sixiu Liu, Claire Birnie, and Tariq Alkhalifah. Coherent noise suppression via a self-supervised blind-trace deep learning scheme. *arXiv preprint arXiv:2206.00301*, 2022.

[32] Randy Harsuko and Tariq Alkhalifah. Storseismic: A new paradigm in deep learning for seismic processing. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–15, 2022.

[33] Claudio Strobbia, Tim Dean, Simone Re, Enrico Ceragioli, Denis Sweeney, and Megan Nightingale. Fundamental noisethe key to recovering your signal: an integrated workflow for seismic survey design. *First Break*, 40(1):87–95, 2022.

[34] FJ Billette and Sverre Brandsberg-Dahl. The 2004 bp velocity benchmark. In *67th EAGE Conference & Exhibition*, pages cp–1. European Association of Geoscientists & Engineers, 2005.

[35] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[36] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[37] Oleg Ovcharenko and S Hou. Deep learning for seismic data reconstruction: opportunities and challenges. In *First EAGE Digitalization Conference and Exhibition*, volume 2020, pages 1–5. European Association of Geoscientists & Engineers, 2020.