Artificial Intelligence in Geosciences 3 (2022) 101–114

Contents lists available at ScienceDirect

Artificial Intelligence in Geosciences

journal homepage: www.keaipublishing.com/en/journals/artificial-intelligence-in-geosciences

Original research articles

MLReal: Bridging the gap between training on synthetic data and real data applications in machine learning

Tariq Alkhalifah *, Hanchen Wang, Oleg Ovcharenko

Physical Sciences and Engineering, King Abdullah University of Science and Technology, Mail box # 1280, Thuwal 23955-6900, Saudi Arabia

ARTICLE INFO

Keywords:
Neural networks
Induced seismicity
Image processing
Computational seismology
Waveform inversion

ABSTRACT

Among the biggest challenges we face in utilizing neural networks trained on waveform (i.e., seismic, electromagnetic, or ultrasound) data is its application to real data. The requirement for accurate labels often forces us to train our networks using synthetic data, where labels are readily available. However, synthetic data often fail to capture the reality of the field/real experiment, and we end up with poor performance of the trained neural networks (NNs) at the inference stage. This is because synthetic data lack many of the realistic features embedded in real data, including an accurate waveform source signature, realistic noise, and accurate reflectivity. In other words, the real data set is far from being a sample from the distribution of the synthetic training set. Thus, we describe a novel approach to enhance our supervised neural network (NN) training on synthetic data with real data features (domain adaptation). Specifically, for tasks in which the absolute values of the vertical axis (time or depth) of the input section are not crucial to the prediction, like classification, or can be corrected after the prediction, like velocity model building using a well, we suggest a series of linear operations on the input to the network data so that the training and application data have similar distributions. This is accomplished by applying two operations on the input data to the NN, whether the input is from the synthetic or real data subset domain: (1) The crosscorrelation of the input data section (i.e., shot gather, seismic image, etc.) with a fixed-location reference trace from the input data section. (2) The convolution of the resulting data with the mean (or a random sample) of the autocorrelated sections from the other subset domain. In the training stage, the input data are from the synthetic subset domain and the auto-corrected (we crosscorrelate each trace with itself) sections are from the real subset domain, and the random selection of sections from the real data is implemented at every epoch of the training. In the inference/application stage, the input data are from the real subset domain and the mean of the autocorrelated sections are from the synthetic data subset domain. Example applications on passive seismic data for microseismic event source location determination and on active seismic data for predicting low frequencies are used to demonstrate the power of this approach in improving the applicability of our trained NNs to real data.

1. Introduction

In our attempt to discover the Earth's subsurface, by inverting measurements of its physical properties, we have developed hand-crafted (fixed, in which their elements are not optimized or learned) algorithms to perform various tasks, like denoising, imaging, inversion, etc. (Bourgeois et al., 1989). These hand-crafted (fixed) algorithms are independent of the specific features in the data, and usually, new research is required to update these algorithms to adapt to any evolution (or limitation) in the data. Recently, data-driven approaches have taken center stage in replacing fixed algorithms with neural networks (NNs) trained to perform specific tasks. Machine learning (ML) has gained a lot of traction as a tool to help us solve outstanding problems in seismic waveform data processing and interpretation. Most of the applications in our field have relied on supervised training of NNs, where the labels (answers) are available (Wrona et al., 2018; Araya-Polo et al., 2018; Ovcharenko et al., 2019; Di and AlRegib, 2020; Araya-Polo et al., 2019; Vinard et al., 2022). These answers are often available for synthetic data as we numerically control the experiment, or they are determined using human interpretation or human hand crafted algorithms applied to real data (like those used for first arrival picking, Molyneux and Schmitt, 1999). The challenge in training our NNs on synthetic data is the generalization of the trained NNs to real data, as that process requires careful identification of the training set and the inclusion of realistic noise and other variables between synthetic and real data. In other words, the synthetic and real data are usually far from being

* Corresponding author.
E-mail address: tariq.alkhalifah@kaust.edu.sa (T. Alkhalifah).

https://doi.org/10.1016/j.aiig.2022.09.002
Received 31 May 2022; Received in revised form 24 September 2022; Accepted 24 September 2022
Available online 7 November 2022
2666-5441/© 2022 The Authors. Publishing services by Elsevier B.V. on behalf of KeAi Communications Co. Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

drawn from the same distribution, which is essential for the success of a trained NN (Kouw, 2018). Thus, many synthetically trained NNs have performed poorly on real data. On the other hand, training on real data provides NNs that are often, at best, as good as the accuracy of the labels that were determined by humans or human-crafted algorithms (weak supervision, due to limited labels or labels prone to errors Karamanolakis et al., 2021). So, the data-driven feature of machine learning, in this case, will be highly compromised (Zhou, 2017; Zheng et al., 2021). As a result, we should synthetically generate data for training our NN that are as realistic as possible.

Modeling is a general term encompassing any forward operation in which we have a model and we seek to obtain the corresponding data. The process can be as easy as a matrix–vector multiplication (linear) that can be used, for example, to mute potentially missing traces from common shot gathers or image data for interpolation objectives, where the full shot gathers or images represent the model (our objective). Modeling can also include more complex operations like solving the wave equation using any numerical method (Aki and Richards, 2009). For seismic inversion, modeling, in its simplest form, is often given by the convolution of a source wavelet with a reflectivity model (Wang, 2016). Modeling is a deterministic process (Tarantola, 1987a), and thus, the input–output combination is unique and can often be determined numerically. As such, using modeling and simulation, we can generate ML training input data with their corresponding labels in a straightforward manner. Usually, in this case, the synthetically generated data (like shot gathers with missing traces) represent the input to the NN in an attempt to have it learn to output the model (which is the corresponding full shot gather) (He et al., 2021). For the trained NNs on such synthetic data to generalize to real (application) data, the synthetic training set has to include the features embedded in the real data as much as possible (Kouw, 2018). For one, the training dataset (inputs and labels) should be represented by distributions that include the input and expected labels lead the training to non-sharp (flat) local minima for the real data (Mulayoff and Michaeli, 2020). However, this requirement, especially with respect to the input data to the network, is hard to achieve considering the simplified assumptions we use in modeling and simulation. It requires that we accurately reproduce the correlated and uncorrelated noise ($n(t)$) present in the real data, where t here is the time (or any vertical axis like depth). More importantly, it also requires that we properly represent the source wavelet ($s(t)$) and the reflectivity ($r(t)$), present in the real data. In other words, for the synthetic data $d_s = r_s(t) * s_s(t) + n_s(t)$ in our simplified Earth model, the distributions of these three functions should cover those of the real data, and this is very hard to accomplish (Birnie, 2018). Here $*$ stands for the convolution process. So, we need to find a way to bridge this gap between training NNs on synthetic data and applying them on real ones (see Fig. 10).

The concept of trying to bridge the gap between the training and application data in machine learning is referred to as domain adaptation (Kouw, 2018; Lemberger and Panico, 2020). In this case, the training dataset is assumed to belong to the source domain and the application/inference data are assumed to belong to the target domain, the target of our training. The classic theory of machine learning assumes that the application (target) data of a trained model should come from the same general population (sampled from the same distribution) as the training (source) set. So we need the probability distribution of the synthetic (source) dataset, $P_s(\mathbf{x_s}, \mathbf{y_s})$, where $\mathbf{x_s}$ are the inputs (i.e. synthetic waveform data), and $\mathbf{y_s}$ are the labels (i.e. traveltime picks or horizons) for the source set, to equal the probability distribution of the real (target) dataset, $P_t(\mathbf{x_t}, \mathbf{y_t})$, where $\mathbf{x_t}$ are the inputs (i.e. real seismic data), and $\mathbf{y_t}$ are the labels for the target set that we often seek to predict. One category of data adaptation is referred to as subspace mapping (or more generally, alignment) in which we find a transformation, T, that results in the distribution of the training (source) input data to equal that of the application (target) input data (Fernando et al., 2013). Specifically, $P_s(T(\mathbf{x})) = P_r(\mathbf{x})$. This can be accomplished by projecting the source and target data to the eigenvectors of the two subspaces, then finding a transformation between these projected spaces. Such projections can be achieved by Neural Network embeddings (which are low-dimensional, learned continuous vector representations of discrete variables, Koehrsen, 2018) aimed to find the weights of the embedding layers that minimizes the distance between the distribution of the source samples and the target ones. There are many ways to constrain the transformation or weights to make the distributions similar including the use of optimal transport (Villani, 2008). In this category, even cycle Generative Adversarial Networks (GANs) are used for the purpose of learning a generator to map the target data distribution to the source data distribution (Palladino et al., 2020). However, these methods become more difficult to apply when the dimensions of the data are large, as is the case with waveform (including seismic and ultrasound) data. The method proposed in this paper shares the general framework of subspace alignment implemented in an empirical fashion, and specifically tailored for waveform (like seismic) data.

For seismic applications, as well as many other applications, we often acquire waveform data with sensors placed on the surface of the investigated body Earth. In imaging applications, such data often loosely constrain the vertical (depth) axis (mostly described by the behavior of features along recording channels), and thus, we face issues in imaging events accurately in depth. In this case, wells are often used to correct the depth misties as wells are considered ground truth markers (Luo et al., 2019). Nevertheless, the structural information in these images is provided by the seismic (waveform) data. In applications, not requiring precise vertical dimension labeling (scale-wise), crosscorrelation of the input data with a reference trace from the same data has recently been suggested to help reduce the variance in the distributions of the training and application data, which ultimately helped with the application of ML for direct waveform microseismic event location (Wang and Alkhalifah, 2021). Specifically, the crosscorrelation process converts the vertical axis to lag dimensions with the range of both training and application data becoming much closer. Prior to that, Choi and Alkhalifah (2011) used the process of convolving a reference trace from the observed data with the synthetic data, and conversely convolving a reference trace from the synthetic data with the observed data in a waveform optimization problem to invert for the velocity model. This process helped reduce the difference between the terms of the objective function, and especially in mitigating the dependency of the optimization on the source signature, which is often unknown. In the spirit of these two developments, the correlation with a reference trace and convolution, we propose similar transformations to help us adapt the NN trained on synthetic data to work on real data.

An objective of a neural network model is to provide us with an output for a given input. The output that a trained model will give is based on the training it experienced, and that depends mostly on the source training set and its distribution. A trained neural network generalizes well when the target data are represented, as much as possible, in the source data set. To help accomplish that when the application (target) data are real (field) data, we propose, here, to inject as much of the real data features into the synthetic data training as possible. This can be accomplished by utilizing a combination of linear operations including crosscorrelation, autocorrelation, and convolution between the synthetic and field data. These operations will bring the distributions of the training (source) synthetic dataset, and the (target) real dataset closer to each other, which will help the trained model generalize better on real data. One real data generalization example we use here to test the approach is an NN dedicated to locating microseismic sources directly from recorded waveform data. We also test the approach on an NN trained to predict low-frequency data from high-frequency ones to ultimately help full waveform inversion (FWI) converge better Ovcharenko et al. (2022).

The rest of the paper is organized as follows. We introduce our empirical data projection approach, and explain why it is able to map

information between the training (synthetic) and testing (real) data. We then describe the transformation steps in details in a form of an Algorithm. Applications on two real data examples including passive seismic data to locate microseismic events and active seismic data to predict missing low frequencies follows. Finally, we wrap up the paper with a detailed discussion of the strength and weaknesses of the approach, and some concluding remarks.

2. An empirical data projection

The method proposed here is applicable mainly to supervised learning. Also, we assume that the vertical axis of the input sections (images or shot gathers) are not crucial in absolute values to the task at hand. Only the relative relation between events matters along that dimension. The reason for this assumption will be clear later.

2.1. Setup

We assume we do not have labels for the application data, and thus, we cannot perform transfer learning (a form of domain adaptation) (Wu et al., 2020). In our case, the source synthetic data are labeled, but the target real data are not. This form of domain adaptation is often addressed with unsupervised ML methods (Fernando et al., 2013). In such domain adaptation, another important assumption is implied, and that is the target labels are drawn from the same distribution as the source labels ($P(\mathbf{y_s}) = P(\mathbf{y_t})$). This is an important assumption for the application of synthetically trained NN models on real data. Of course, this requires that the task we plan to perform on the field data is represented by the synthetic training data. Specifically, $P_s(\mathbf{y_s}|\mathbf{x_s}) = P_t(\mathbf{y_t}|\mathbf{x_t})$. In physical terms, this implies that the modeling we do for our synthetic data generation represents the actual physical behavior. In other words, the assumptions used to model the synthetic training set represent the object of application (like the Earth). This assumption is used in most optimization applications including FWI (Tarantola, 1987b). Thus, the issue we are addressing here is the case when the input distributions for the source and target data are not the same, specifically $P_s(\mathbf{x_s}) \neq P_t(\mathbf{x_t})$.

This form of difference between the training and application dataset distributions in machine learning terms is referred to as a covariate shift (Fernando et al., 2013). There are many ways to measure such a shift, including using the Kullback–Leibler (KL) divergence metric. In domain adaptation, and similar to error bounds defined for machine learning in general, we can define an error bound on the application of a trained network model. This error bound is given by the combination of the error in the training and a term related to the complexity of the NN model (like its size). This, however, assumes that the training and application data come from the same distribution. For the case of a covariate shift, we have a similar, more complicated bound, given by Ben-David et al. (2007), Lemberger and Panico (2020):

$$\varepsilon_t(\mathcal{NN}) \leq \varepsilon_s(\mathcal{NN}) + d(P_s(\mathbf{x_s}), P_t(\mathbf{x_t})) + \lambda, \tag{1}$$

where ε_s is the bound on the training error, and $d(.,.)$ is the distance between the marginal distributions of source and target datasets. Here, λ represents the optimal joint errors of a neural network model between the source and target datasets. So the upper bound of the application error is guided by these three terms.

So our objective is to devise a transformation that minimizes the difference measure d in Eq. (1) between the distribution of the training (source) and application (target) datasets. Specifically, we aim to find the transformation $\hat{\mathbf{x}}_s = T_s(\mathbf{x_s})$ on the source data set and $\hat{\mathbf{x}}_t = T_r(\mathbf{x_t})$ on the target data set so that the probability distributions $P_s(\hat{\mathbf{x}}_s) \approx P_t(\hat{\mathbf{x}}_t)$. Fig. 1 summarizes this goal within the framework of the training process. So the input for the training of the Neural network ($\mathcal{NN}$) model is $\hat{\mathbf{x}}_s$, in which the model parameters are optimized to match the labels $\mathbf{y_s}$ using a loss function ($\mathcal{L}$). On the other hand, the input during inference is $\hat{\mathbf{x}}_t$. We will discuss the transformations T_s and T_r in the next section, where the input are waveform (i.e. seismic) data, d.

2.2. Conditioning synthetic data for training

A trace in the seismic data can be represented by a combination of reflectively, source wavelet and noise, as follows:

$$d^{ij}(t) = r^{ij}(t) * s^{ij}(t) + n^{ij}(t), \tag{2}$$

where i is the index of the trace, and j is the index of the section in which the trace belongs to whether the section corresponds to a shot gather or a seismic image. Depending on the data, all three components $(r(t), s(t), n(t))$ can vary over traces and sections. Here, t may represent time or depth depending on whether the inputs are shot gathers or depth images, respectively. For a shot gather for example, often $r(t)$ changes with moveout, and of course, $n(t)$ changes from one trace to another. In training a neural network model to work properly on d(t), we often use synthetic data, d_s, in the training that hopefully includes a proper representation of all these components.

To do that, we use transformations applied on the vertical axis (time or depth), and thus, as mentioned earlier, we assume that the scale of the vertical axis is not crucial to the task. This assumption, though limiting the application of this approach, will allow us to apply the necessary transformation for domain adaptation. We will elaborate in the discussion section on the effect that this assumption may have on the application of this approach. To migrate the components described in Eq. (2) from the real data to the synthetic ones, we use linear operations, and thus, we define new training data (source data transformed by the proposed approach) as follows:

$$\hat{d}_s^i(t) = d_s^i(t) \otimes d_s^k(t) * d^{ij}(t) \otimes d^{ij}(t), \tag{3}$$

where k is the index of a reference trace from the synthetic input data (fixed for both synthetic and real data), and j is the index of a section from the real data whether the section corresponds to a shot gather or a seismic image. The operator $\otimes$ represents crosscorrelation, and in this equation, we have a crosscorrelation between the input synthetic section $d_s^i(t)$ and a reference trace from that section $d_s^k(t)$ convolved with a randomly drawn (j) autocorrelated section from the real data $d^{ij}(t)$ per epoch of training. As a result, the synthetic data will be exposed to a distribution of field data. The reference trace can be a near offset trace in the case of a shot gather input, or it can be any trace from the input for a seismic image. The index of the reference trace should not change between sections to maintain the relative relation between sections. The randomly picked j index for autocorrelated real data varies in the training per epoch to allow for proper representation of the real data imprint on the training set. Assuming we only have noise in the real data, the autocorrelation of random noise yields a quasi delta function at zero lag proportional to the energy of the noise. A convolution with such a function will incorporate that energy into the synthetic data so that the signal-to-noise ratio (SNR) in the transformed synthetic data would be comparable to that of the autocorrelated real data.

To allow for a transformed source data to have a similar distribution to the application data, we also transform the target (application) data in a similar fashion in which the features of the source synthetic data are incorporated in the target data. To achieve that, we apply the following transformation, T, to the real data during the inference stage:

$$\hat{d}^i(t) = T(d^i) = d^i(t) \otimes d^k(t) * \frac{1}{N_s} \sum_j d_s^{ij}(t) \otimes d_s^{ij}(t), \tag{4}$$

which includes the same operations as in Eq. (3) with the role of the real and synthetic data reversed. In this case, N_s represents the number of synthetic sections in the training set, and we actually convolve with the mean of the autocorrelated synthetic data, instead of a randomly drawn sample, like in the training. We will refer to the transformations in Eqs. (3) and (4) as {*MLReal transformations*}. The idea of having two instances of each data (synthetic and real) in Eqs. (3) and (4) is to balance their contribution to the new training and application data

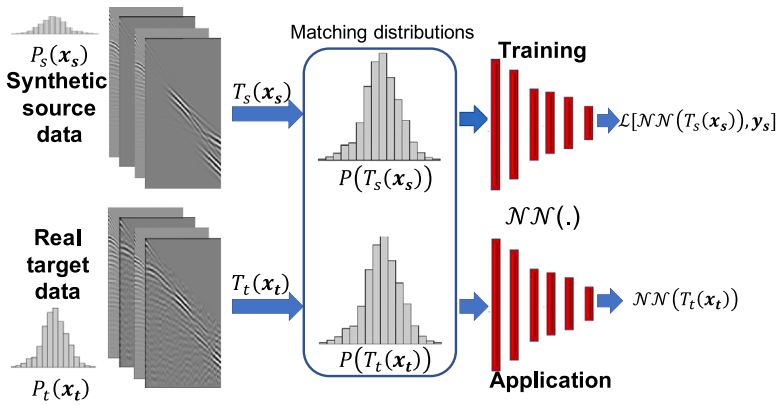


Fig. 1. The workflow of the proposed data adaptation in which the training (source) dataset might have a different distribution than the application (target) dataset, where transformations Ts and Tr will help reduce such differences and provide new data as input to the neural network function (N N) to train the network to reduce loss L, and then apply to real data. Here, P are the probability distributions and we show schematic versions of them for the source and target datasets.

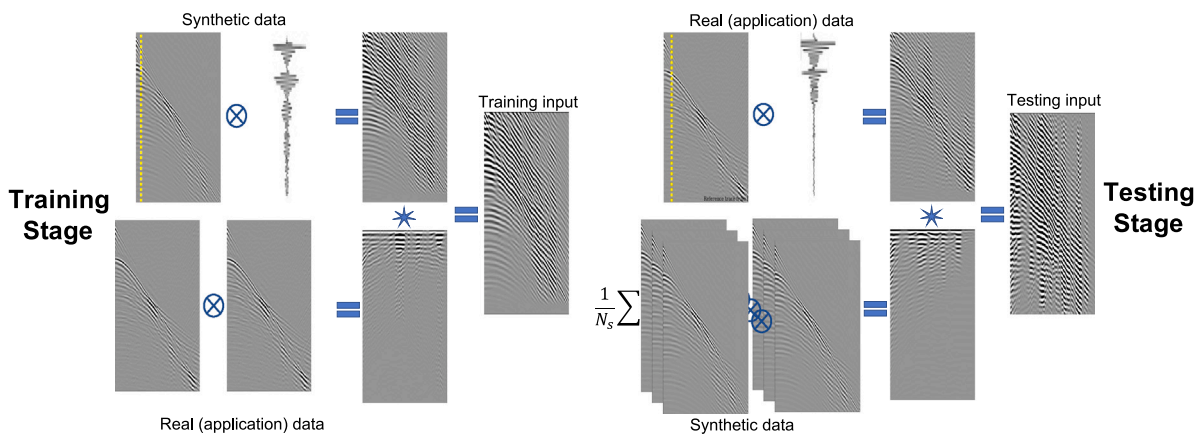


Fig. 2. The workflow chart for the MLReal method applied to marine data. Left side: The proposed process used for producing the training data; Right side: the proposed process used for producing the application data. The circled cross symbol denotes a crosscorrelation operation, and the star symbol denotes a convolution operation. The dashed vertical yellow line in the input traces indicates the location of the reference trace.

sets. This way, we match the properties of the synthetic and real data used for training and inference, respectively. Note that the convolution operation that connects equal amounts of the synthetic and real data contributions is commutative. We will see the value of this operation more in the next section.

Meanwhile, Fig. 2 (left) demonstrates the process of applying equation (3), where the synthetic data were generated for the training of an NN model to predict low frequencies (Ovcharenko et al., 2019). The objective was to improve the FWI convergence for real marine data by adding low-frequency content into the data. The importance of such low frequencies for FWI convergence is demonstrated in Ovcharenko et al. (2022). On the other hand, Fig. 2 (right) demonstrates the process of applying equation (4) on real data for an input to the trained model. Note that the resulting shot gathers look similar for the two processes, and they are similar, especially, in the distribution of energy along the channels. They, however, still maintain the characteristics (the moveout) of the original data (synthetic or real). However, they contain more energy and more features in which the NN model can utilize. We will show the results of applying the proposed process for this task in the examples.

2.3. A frequency domain analysis

If we transform the real data to the frequency (or vertical wavenumber) domain, considering the basic laws of the Fourier representation of crosscorrelation and convolution, equation (2) can be written as

$$D^{ij}(\omega) = A^{ij}(\omega)e^{i\phi^{ij}(\omega)} = R^{ij}(\omega)S^{ij}(\omega) + N^{ij}(\omega), \tag{5}$$

where ω is the angular frequency, A and ϕ are the amplitude and phase, respectively, of the complex-valued data. All capital letters represent the frequency domain form of the reflectivity, source, and noise functions in Eq. (5), given respectively. As a result, we can write equation (3) in the frequency domain as

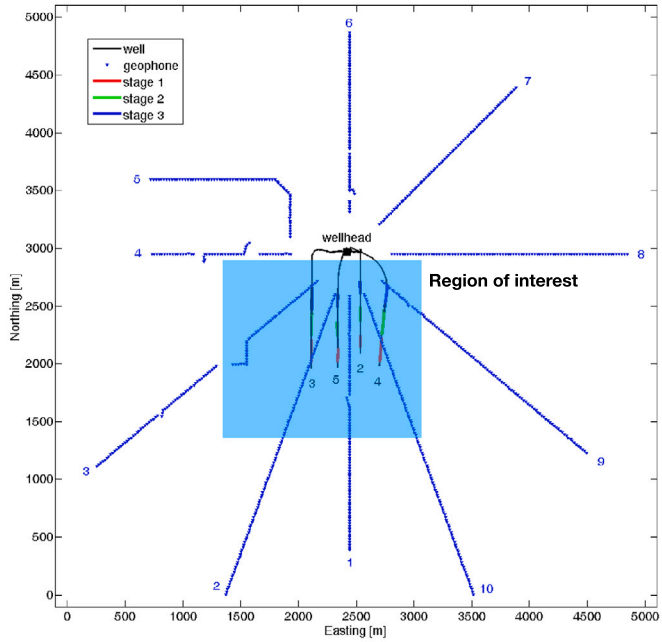
$$\begin{aligned} D_t^i(\omega) &= \bar{D}_s^{\,i}(\omega)D_s^k(\omega)\bar{D}^{ij}(\omega)D^{ij}(\omega) = \bar{D}_s^{\,i}(\omega)D_s^k(\omega)\left(A^{ij}(\omega)\right)^2 \\ &= \bar{D}_s^{\,i}(\omega)D_s^k(\omega)\left(R^{ij}(\omega)S^{ij}(\omega) + N^{ij}(\omega)\right) \\ &\quad \times \left(\bar{R}^{ij}(\omega)\bar{S}^{ij}(\omega) + \bar{N}^{ij}(\omega)\right), \end{aligned} \tag{6}$$

where the overstrike, $\bar{.}$, symbol stands for the complex conjugate.

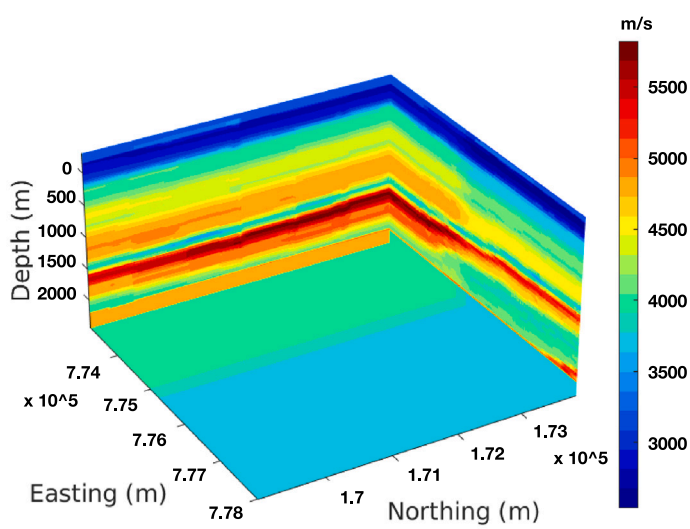
The application of the model on real data will involve an input to the model given by Eq. (4), which can be represented in the frequency domain by

$$\begin{aligned} D_r^i(\omega) &= \bar{D}^i(\omega)D^k(\omega)\frac{1}{N_s}\sum_j \bar{D}_s^{\,ij}(\omega)D_s^{ij}(\omega) \\ &= A^i(\omega)A^k(\omega)e^{i\left(\phi^i(\omega)-\phi^k(\omega)\right)}\frac{1}{N_s}\sum_j \bar{D}_s^{\,ij}(\omega)D_s^{ij}(\omega) \\ &= \left(R^i(\omega)S^i(\omega) + N^i(\omega)\right)\left(\bar{R}^k(\omega)\bar{S}^k(\omega) + \bar{N}^k(\omega)\right) \\ &\quad \times \frac{1}{N_s}\sum_j \bar{D}_s^{\,ij}(\omega)D_s^{ij}(\omega). \end{aligned} \tag{7}$$

Note that the frequency content of the input to the training and the application on real data will include a squared amplitude (A) of the real data in an even manner. Also, the key here is that Eqs. (6) and (7) share a similar magnitude of noise, reflectivity, and source



(a)



(b)

Fig. 3. (a) The passive seismic acquisition lines. (b) The velocity model estimated in the region and used here to generate the synthetic data.

signature, or in more general terms, similar energy. Since the two data sets, after transformation, share the same elements sampled from their corresponding distributions, then the distributions of the two data sets should be close, which allows us to satisfy the requirement $P_s(T_s(\mathbf{x})) \approx P_s(T_s(\mathbf{r}))$ for the generalization of the NN model. In other words, the second term in Eq. (1) will be small.

3. The algorithm

Considering we have unlabeled real data in which we want to perform a particular task through training a neural network to do so, we first generate synthetic samples consisting of input features and labels that represent the task at hand, in which we assume $P_s(y_s|x_s) =$

$P_t(y_t|x_t)$. Next, we expose the source (synthetic) input features and the real data ones to the proposed transformations.

Thus, the steps involved in the transformations, suggested here for domain adaptation, can be summarized using Algorithm 1 during training stage. After training, the application (inference) on the real data can be summarized using Algorithm 2. Considering the linear nature of the transforms involved, all the transformation steps can be performed efficiently in the Fourier domain. We only need to inverse Fourier transform of the data at the end of the process.

Algorithm 1 The training stage

Input: Real seismic input data, $\mathbf{d}$; Number of input real data sections N_r; Synthetic input seismic data, $\mathbf{d_s}$; The corresponding synthetic data labels for the task at hand, $\mathbf{y}$; Number of synthetic samples N_s; The neural network model $\mathcal{NN}$; The Loss function $\mathcal{L}$; The number of epochs M; Reference correlation trace index, j.

Output: The trained model $\mathcal{NN}_t$.

Initialize: $\mathcal{NN}$ with parameters $\theta = \theta_0$;
for $i = 1 \dots N_r$ **do**
  $a^i(t) = d^i(t) \otimes d^i(t)$, the autocorrelation of the real data over time, t;
end for
for $epochs = 1 \dots M$ **do**
  for $i = 1 \dots N_s$ **do**
    $c^i(t) = d_s^i(t) \otimes d_s^{ij}(t)$, the crosscorrelation of the input synthetic data with a reference trace, j;
    $T_s(d_s^i) = c^i(t) * a^k(t)$, The convolution of crosscorrelated input with a randomly drawn, indexed $k \in \{1, N_r\}$, auto-correlated real data;
  end for
  Using loss, $\mathcal{L}(\mathcal{NN}(T_s(\mathbf{d_s})), \mathbf{y})$, we update θ
end for
$\mathcal{NN}_t = \mathcal{NN}$

Algorithm 2 The application stage for a single input

Input: Real seismic input data section, d^i, for $i \in \{1, N_r\}$; Synthetic input seismic data, $\mathbf{d_s}$; Number of synthetic samples N_s; The trained neural network model $\mathcal{NN}_t$;

Output: The prediction, y_t^i.

Initialize: $a(t) = 0$;
for $i = 1 \dots N_s$ **do**
  $a(t) = a(t) + d_s^i(t) \otimes d_s^i(t)$, the sum of the autocorrelation of the synthetic data;
end for
$a(t) = \frac{1}{N_s} a(t)$, the mean
$c^i(t) = d^i(t) \otimes d^{ij}(t)$, the crosscorrelation of the input real data with a reference trace, j;
$T_r(d^i) = c^i(t) * a(t)$, The convolution of crosscorrelated input with the mean of the autocorrelated synthetic data;
$y_t^i = \mathcal{NN}_t(T_r(d^i))$, the prediction.

4. A microseismic data example

We will first show the impact of the above operations on real data acquired as part of monitoring microseismic events. The passive seismic acquisition was performed using a star configuration of sensors, as shown in Fig. 3(a), to monitor a hydraulic fracturing stimulation of a shale gas reservoir in the Arkoma Basin in the United States. Fig. 4(a)a shows the real data for one microseismic event. The shot gather section includes ten segments from the various lines (azimuths), shown in Fig. 4(a), plotted here side by side. We were provided a total of 75 of these microseismic event recordings and the corresponding locations of the events, which were determined using conventional methods (we use here only 10 of them). These labels (event locations) will help us evaluate the accuracy of our trained NN model. For more details on the data, we refer you to Staněk and Eisner (2017). We were also given a velocity model for the area, which is shown in Fig. 3(b). Using this velocity model, we employ a second-order in time and fourth-order in space, finite difference approximation to solve the acoustic wave equation and simulate wavefields from 5000 randomly placed seismic point sources within the region of interest (the region we expect the real events to be located). The resulting 5000 synthetically recorded sections, using the layout in Fig. 3(a), and the corresponding event location (labels) are split in a random manner into a training set (4000 samples) and a validation set (1000 samples). An example synthetic data section, for an event near the one estimated for the real data section in Fig. 4(a), is shown in Fig. 4(b). If we compare this synthetically generated section with the field one, we can appreciate the large difference between the two data despite them sharing similar general shapes as they originate from nearby sources. We do not have the source time information, which explains the shift between the events in the two sections. We trained the neural network model on such synthetic data (used the crosscorrelation operation with a reference trace to mitigate the shift (Wang and Alkhalifah, 2021), and because of the large differences in features between training and testing data, the accuracy of the location of the 10 events, as we will see later, is low.

4.1. Data transformations

We first calculate the crosscorrelation of the input section with a reference (fixed location for all data synthetic and real) trace from the section. The reference trace in this application is given by the first trace of each input section, or in other words, the first trace of the first line in the section. An example result of this operation applied to the sections shown in Figs. 4(a) and 4(b), is given by Fig. 5(a) and Fig. 5(b), respectively. This operation has largely mitigated the time shift between the field and synthetic data, as the vertical axes are now given by the time lag. We then calculate the autocorrelation of the field and synthetic data, a sample of which are shown in Fig. 6(a) and Fig. 6(b) corresponding to the sections shown in Fig. 4(a) and Fig. 4(b), respectively. We can appreciate how much the autocorrelation of the field section carries information representing the source wavelet and the noise, in a zero-phase fashion. Applying the operations involved in the MLReal transformations (Eqs. (3) and (4)) on the synthetic and real data, we obtain the sections shown in Fig. 7(a) and Fig. 7(b), respectively. We window the part around zero lag to reduce the size of the input-to-the-network data. The two sections look much more alike than those in Figs. 4(a) and 4(b).

4.2. The training

Using sections like that shown in Fig. 7(b) in which the location of the source (as the label) is known from modeling, we train a 14-layer convolutional neural network to predict the location of the microseismic source (x, y, and z). The training included 4000 input sections (samples, modeled synthetically) and their corresponding labels (the training set), and the training was executed over 5000 epochs, single batch, using an Adam optimizer (Kingma and Ba, 2014). The convolution with randomly selected auto correlated sections from the real data is done at every one of the 5000 epochs. The random selection allows for more variance in the information extracted from the real data to be present in the training samples. The loss function for the training and validation, shown in Figs. 8(a) and 8(b), respectively, show good convergence. As we may expect due to the data being recorded at the surface, the error in the horizontal location of an event is far less than that for the vertical location. Thus, the lateral resolution in locating the event is expected to be higher.

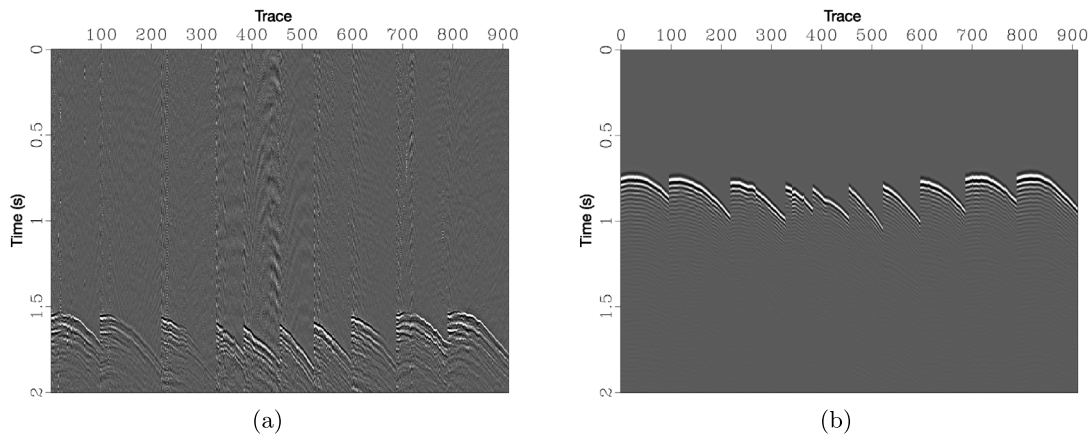


Fig. 4. (a) The field recorded data for a single microseismic event along the 10 lines plotted side by side. (b) The synthetic data along the same lines from a source near the field data one, which was provided for this event. The time of the source is unknown, which explains the shift.

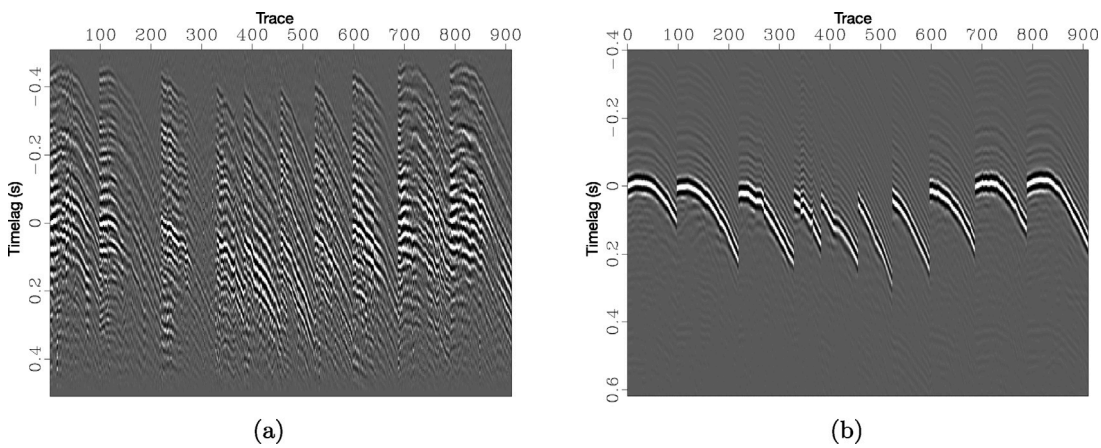


Fig. 5. (a) The crosscorrelation of the field recorded data shown in Fig. 3(a) with the first trace in the section. (b) The crosscorrelation of the synthetic data shown in Fig. 3(b) with the first trace in the section.

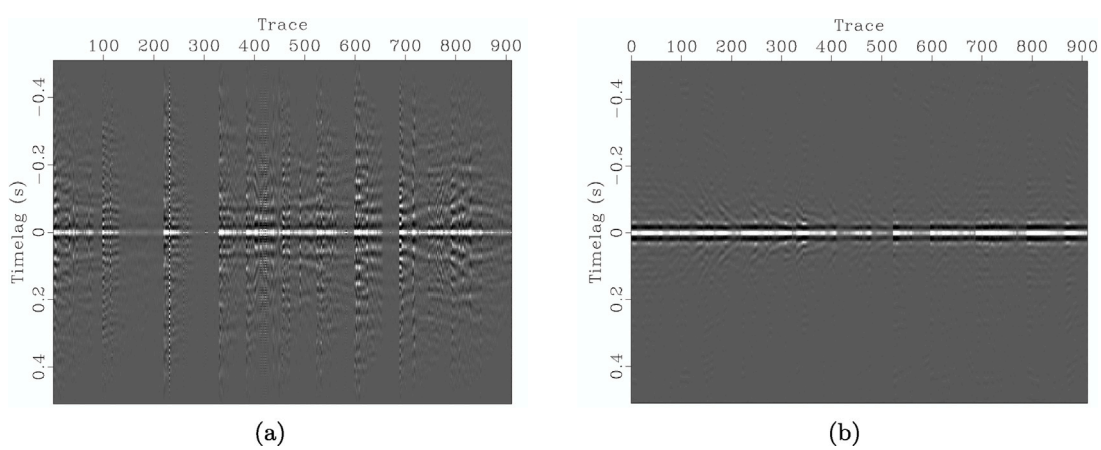


Fig. 6. (a) The autocorrelation of the field recorded data shown in Fig. 6(a). (b) The autocorrelation of the synthetic data shown in Fig. 6(b).

4.3. Results

Then we input 10 real data sections, after applying the operations in Eq. (4), into the trained model to evaluate the accuracy of the prediction. Fig. 9 shows the predicted locations (in blue) and the provided ones (in red) in the region of investigation. We also show the predicted without convolving with the autocorrelation, just the crosscorrelation with a reference trace (Wang and Alkhalifah, 2021). A projection of the reference and predicted locations onto the 2D $x-y$, $x-z$, and $y-z$ planes highlights the accuracy along the various axes (Fig. 10) in support of the loss curves shown in Figs. 8(a) and 8(b). The differences are generally small and can be caused by many factors. For one, the NN model is known to have a bias toward smoothing the output (Rahaman et al., 2019). A very small network (Wang et al., 2020) will levitate towards the mean of the training labels. Of course, a cure for that is to increase the network size. Another reason for

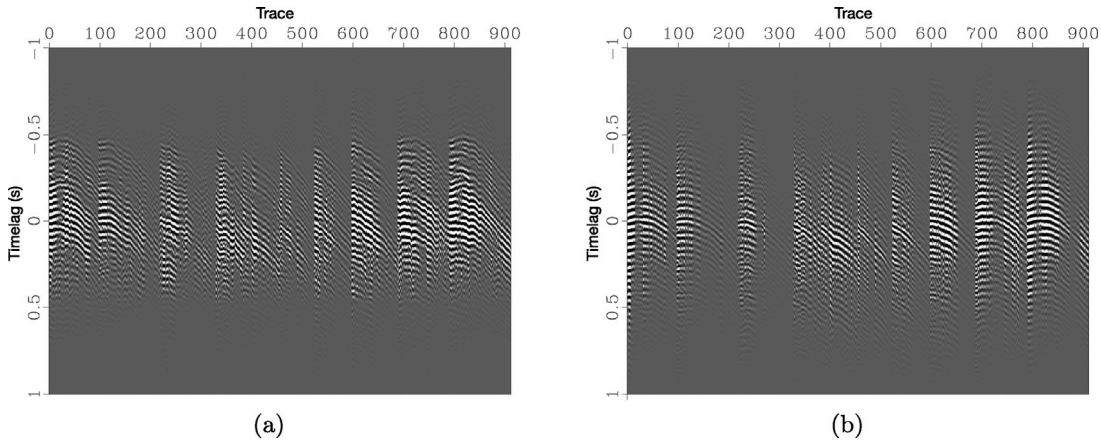


Fig. 7. (a) An example section testing application data after applying Eq. (4) (the proposed transformation) for the section shown in Fig. 3(a). (b) The training input data corresponding to the section shown in Fig. 3(b) after injecting it with real data information using the proposed method, and specifically Eq. (3).

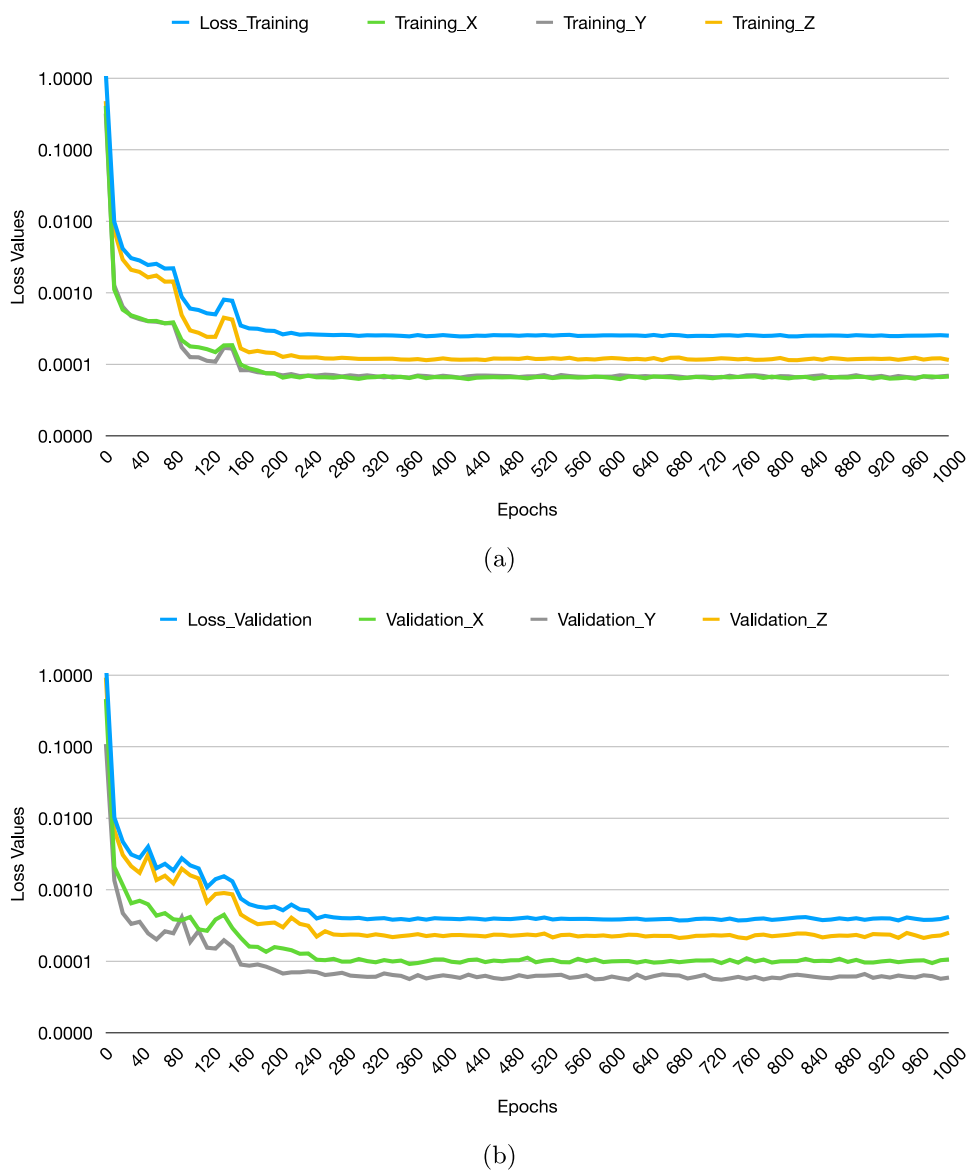


Fig. 8. (a) The training loss for the separate coordinate components as well as the total loss (distance). (b) The validation losses.

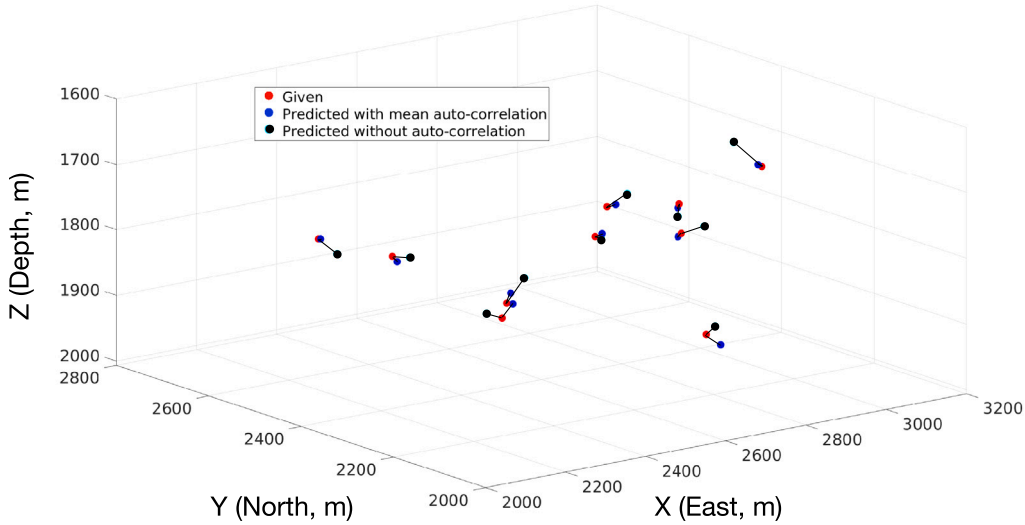


Fig. 9. (a) A 3D plot of the locations of the predicted (blue dots) and provided (red dots) source locations for 10 field data events, as well as the trained and predicted without the convolution with the autocorrelation of the other data (gray dots). The thin lines between the dots connect both predictions to the provided locations from both predictions.

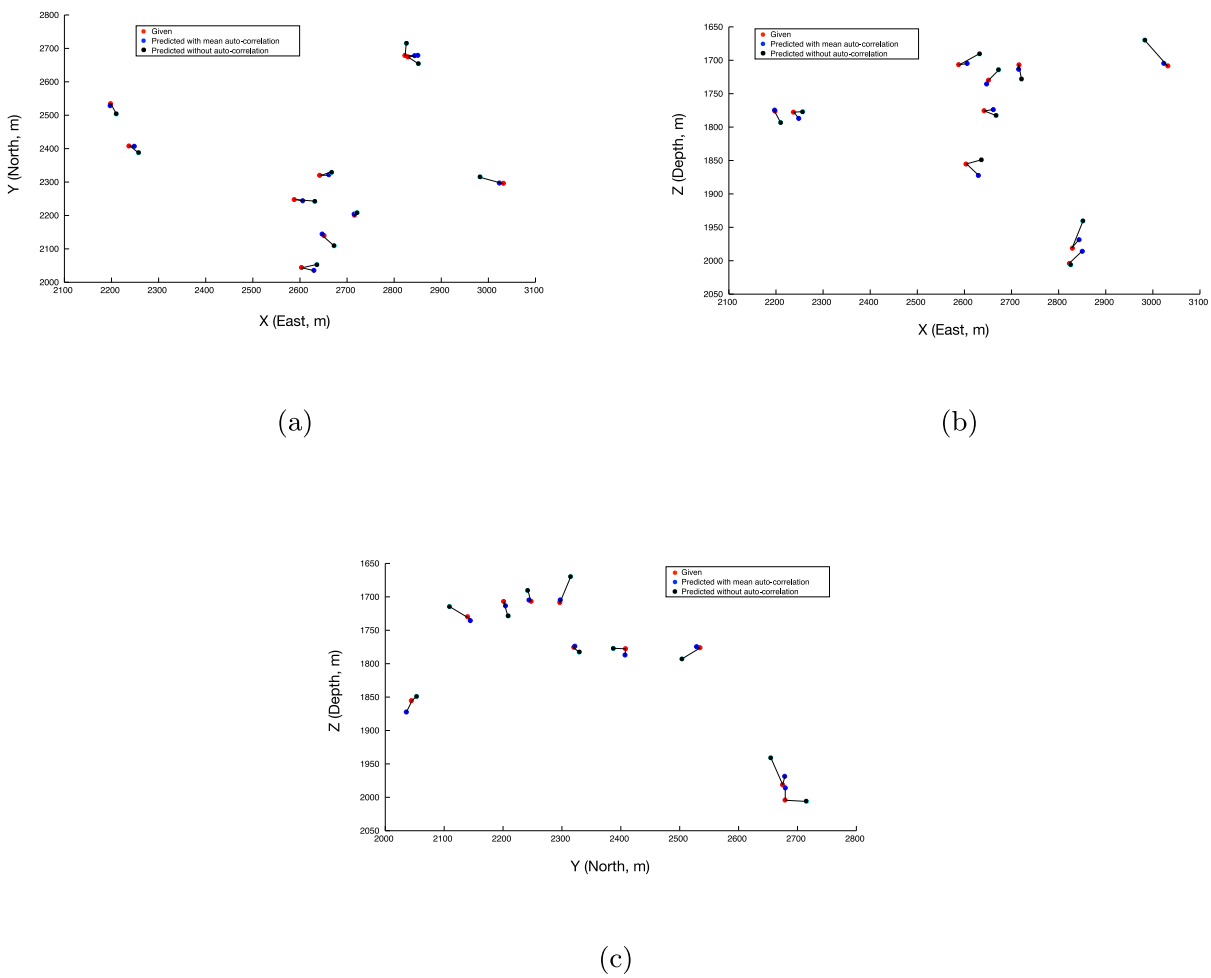


Fig. 10. The locations of the predicted (blue dots) and provided (red dots) source locations for 10 field data events, as well as the trained and predicted without the convolution with the autocorrelation of the other data (black dots), all of which are projected on to (a) the $x-y$ plane, (b) the $x-z$ plane, and (c) the $y-z$ plane. The thin lines between the dots connects both predictions to the provided locations.

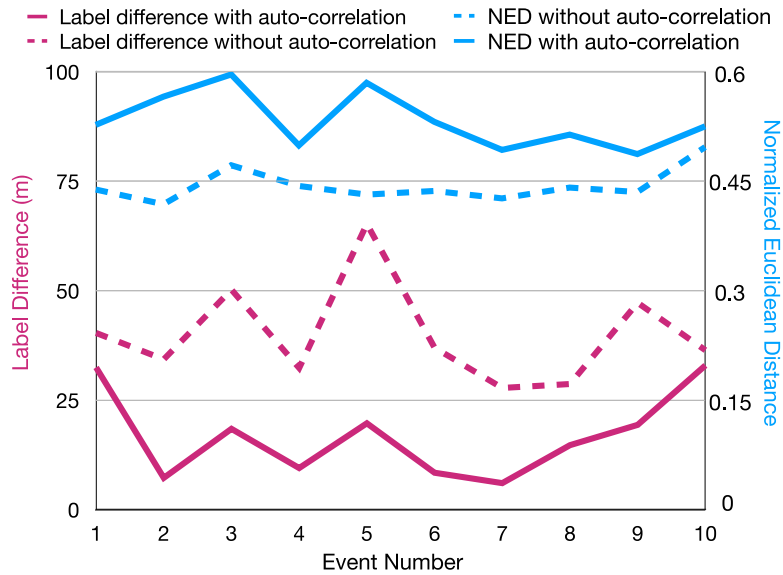


Fig. 11. The normalized euclidean distance (blue) between the field data and modeled data from the predicted source locations (blue), as well as the actual source location (the label) difference between the provided locations and the predicted locations using the proposed method (solid lines) and using the same network trained without the convolution with the auto-correlation (dashed lines).

the difference could be the simplified assumptions used in our data simulation for the training of our NN model compared to the modeling approach potentially used in determining the location of the events by the data providers. To further evaluate the improvements in data coherency between training and application using the proposed MLReal transformations, we compute the normalized Euclidean distance (NED) between the new testing (real) sections and the corresponding synthetic ones. Fig. 11 shows NED for sections in which only the crosscorrelation with a reference trace is used to mitigate the shift (dashed blue), and those for sections in which MLReal transformations were used (solid blue) in the training. Though NED values can range between 0 to 1, the values are reasonably higher for our proposed method, compared to just cross-correlating with a reference. We also plot in Fig. 11 the distances between the predicted sources and provided ones for the same 10 events. For our proposed method, these values average around 15 m. Considering the quality of the data and the layout of the sensors on the surface, the differences for the proposed method are very reasonable. Meanwhile, with only the crosscorrelation with a reference trace, which was developed earlier (Wang and Alkhalifah, 2021), the location difference averaged around 45 meters (Fig. 11, in black). So the transformations (our approach for domain adaptation) helped a lot in this example.

5. Low-frequency prediction example

In this example, we apply the approach to the task of low-frequency extrapolation. In particular, we reconstruct low-frequency components, < 5 Hz, for a complete shot gather from the available high-frequency representation, > 4 Hz, of this shot gather. The predicted low-frequency data aim to help full-waveform inversion (FWI) converge to the global minimum by mitigating the cycle-skipping problem (Ovcharenko et al., 2019). Since low frequencies are often not available in real-world field data or are contaminated by noise, we train a deep neural network on synthetic data and run inference on a band-limited marine dataset acquired offshore northwest Australia. An example shot gather is shown in Fig. 2.

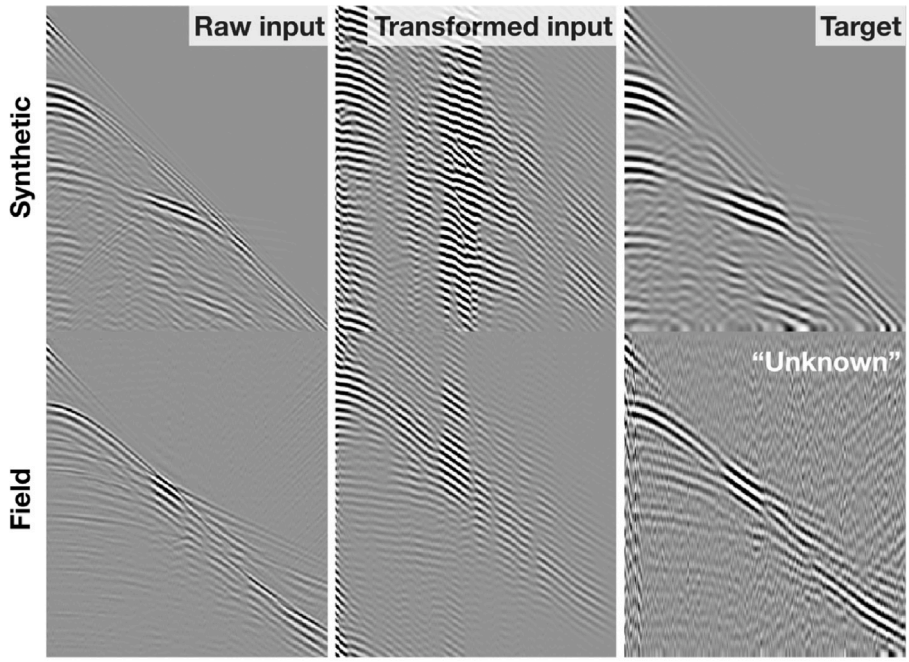
Similar to an inverse problem, where the sought-after model are often unknown, available low-frequencies in the field dataset are insufficient to enable unsupervised training for bandwidth extrapolation. For this reason, we train the deep learning model on synthetic waveforms simulated in a variable range of elastic media initializations. Specifically, we extract the acquisition parameters and source signature from the field data and use these for numerical modeling in a set of synthetic subsurface initializations. The source wavelet should have a broad spectrum to allow us to bandpass the modeled shot gather to a high-frequency shot gather (similar band to the field data) and a low-frequency band shot gather (our objective). Common shot gathers, generated using such a setup, provide inputs and labels for the training of the deep learning model.

5.1. Data transformations

For training on synthetic data, we only apply the MLReal transformations on the inputs to the network assuming that we seek targets similar to those from the synthetic data distribution. In other words, we assume that the velocities used in generating the synthetic data are close to the true Earth ones, which will mitigate the limitation we mentioned with regard to the vertical dimension of the input. To set up the adaptation workflow for the input high-frequency data, we select an arbitrary trace from the high-frequency partition of the synthetic dataset and use this trace as a constant amplitude reference throughout the application. During training, for a given sample of input high-frequency data, we first cross-correlate it trace-wise with a reference trace from the same shot gather (Fig. 2). Then we draw a random high-frequency data sample from the field dataset and convolve its auto-correlation with the result of the previous operation, as demonstrated in Fig. 2. The effect of such transformations is shown in Fig. 12. The transformed synthetic and real data look-alike to the point they share similar energy distribution as a function of offset. The synthetic transformed data are used to train the network without the need to apply any alterations to the output synthetic low-frequency shot gather, as the NN needs to learn such mapping. Of course, we impose the assumption that the synthetic elastic modeling here represent the physics of wave propagation inside the Earth.

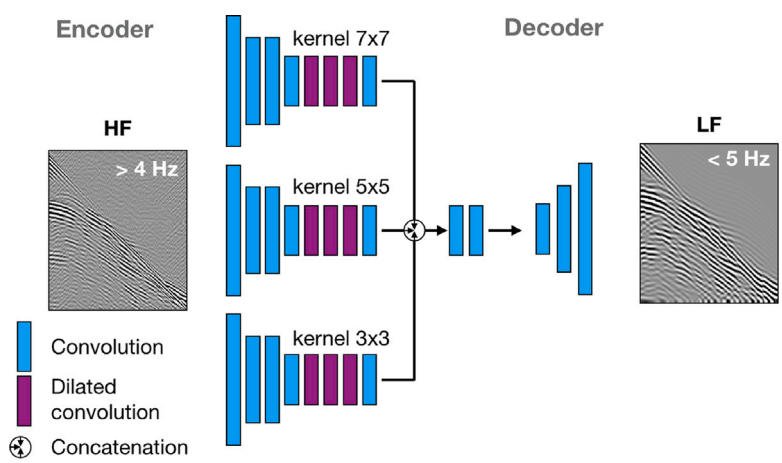
5.2. Deep learning framework

We approach the frequency bandwidth extrapolation for the entire shot gather as an image-to-image translation task. The design of a neural network should account for the need to capture long-wavelength patterns in the data since these are the data-domain projection of low-frequencies. For this reason, we follow (Wang et al., 2018) to build a



(a)

Fig. 12. The original input and target data from synthetic and field datasets (first and last columns; the input data transformed by the proposed MLReal transformations (central column).



(a)

Fig. 13. The multi-column architecture for low-frequency extrapolation neural network model. This translates the high-frequency data, HF, into its low-frequency counterpart, LF.
.

network that extracts multi-scale representations from the input volume by using dilated convolutions. A similar note about the need for wide-span convolutional kernels was made by Sun and Demanet (2019). In particular, the proposed architecture (Fig. 13) includes a three-column encoder featuring dilated convolutions with kernel sizes of 3, 5, and 7, followed by a convolutional decoder. In total, there are around 900k trainable parameters.

The training strategy implements the super-convergence concept by scheduling the learning rate according to Smith and Topin (2019), with the minimum and maximum learning rates of $1e-5$ and $1e-3$, respectively. We also find it helpful to initialize the network following (He et al., 2016), as well as to average predictions within an ensemble (Chollet, 2017) of 5 network initializations to reduce noise in the output data. One more note about training is that we setthe set the batch size to 4 since smaller batch sizes are prone to lead the inversion to non-sharp local minima (Keskar et al., 2016).

5.3. Results

Each shot gather used as input and output of the network is a single-channel image of 324 × 376, measuring the number of receivers in the marine streamer and the number of time samples, respectively. Receivers are spaced 25 m apart while the temporal sampling is coarsened to 8 ms. We create a training dataset of 3072 shot gathers by modeling 3 shots in each of the 1024 random subsurface realizations. These shots are then split into partitions of 2765, 154, and 153 samples for training, validation, and testing, respectively. The set of random subsurface models is derived by distorting the layered models using

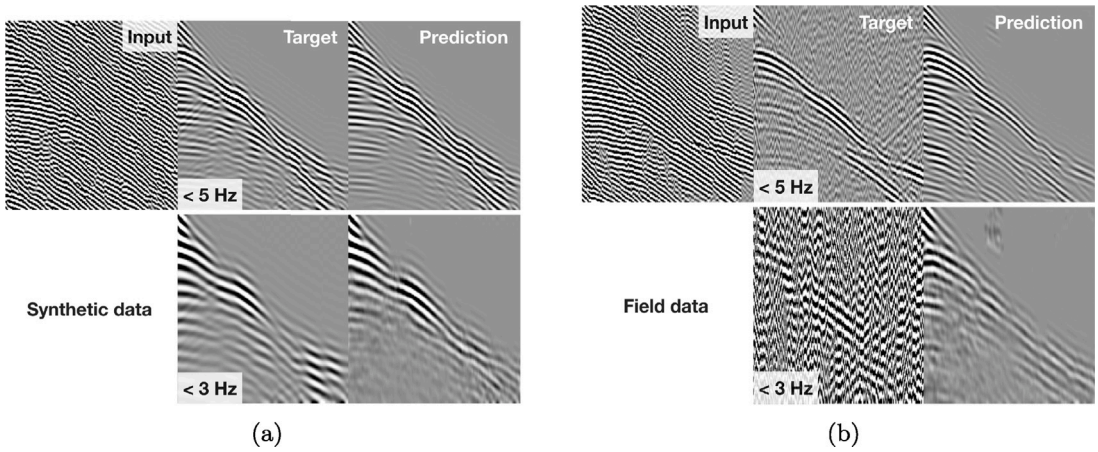


Fig. 14. Prediction results for synthetic (a) and field data samples (b). The top row contains the input data processed by the proposed transformations and the corresponding predictions of data below 5 Hz. The bottom row contains the same predicted data lowpassed below 3 Hz.

elastic transforms, similar to Kazei et al. (2021). The optimization by Adam (Kingma and Ba, 2014) stagnates after 80 epochs, delivering sufficient convergence of the proposed training.

The predicted low-frequencies < 5 Hz indicate a good match for both synthetic (Fig. 14(a)) and field datasets (Fig. 14(b)). This is expected due to the intentional overlap from 4 to 5 Hz between input and target frequency bands used to recover the amplitude of the signal when needed. We also low-pass the predicted data below 3 Hz to explore the regression capability of the trained network in the part of frequency spectra where data were not present. For the synthetic test, while weak reflections in the low-passed data are missing in the predictions, the general shape of the wavelet, as well as the early arrivals, are fairly well reconstructed. On the other hand, since real data suffer from a low signal to noise ratio at low frequency, like below 3 Hz, the predictions admitted higher signal-to-noise ratio, and reasonable smooth wavefields that can ultimately benefit waveform inversion. The high signal-to-noise ratio is courtesy of the synthetic data training where the output is free of noise and includes very low frequency.

In this example, we mainly conduct a proof-of-concept study of a geophysical application of our domain adaptation approach given by MLReal transformations (Eqs. (3) and (4)), rather than seek the state-of-the-art bandwidth extrapolation accuracy. Realistic inference results by the same trained network on synthetic and field datasets suggest the viability of the approach for supervised regression tasks where only input data from the field dataset is available.

6. Discussions

We are proposing, here, a framework for the use of machine learning models trained, in a supervised fashion, on synthetic data to adapt to a specific real data set. We assume a covariate shift between the input (to the network) synthetic and field data, in which the distribution of the generated synthetic data is different from the label-less field data we aim to address. Like all domain adaptation approaches (mostly based on neural networks) that address the shift in distribution between the input source and target data, the adaptation process is geared to a specific target data (Kouw, 2018; Palladino et al., 2020). So, the proposed approach assumes the availability of the field dataset. We prepare the training synthetic data through modeling for an particular task. Like in all such applications, including the examples we have shared, the synthetic data should reflect the dimensions and the characteristics of the field data as much as possible. We often do that in FWI applications (Tarantola, 1984) and it is common in machine learning applications on seismic data (Yu and Ma, 2021; Iturrarán-Viveros et al., 2021).

One of the reasons for the crosscorrelation with a reference trace is to balance the contributions from synthetic and real data in the input data to the network, without affecting the general features of the input data. In the microseismic example, this operation also helped reduce the input data size. The fact that the input-to-the-network is effectively different from the original data is not an issue for NN models, as they adapt to any data we deem as input. What matters is whether the input is consistent between the training (source) and application (target) data. Actually, the cross-correlation operation can enhance the data with features (from the cross-talk between unrelated events) that will further enrich the training data with information that can help in identifying the corresponding labels. For example, if a shot gather includes two reflections, the crosscorrelation of that shot gather with a reference trace from it will result in two additional events from the crosstalk of the two events, and these added events will be different from one shot gather to another depending on the distance between the two events and their moveouts. Such additional events will add to the information embedded in the input data that the NN model can use to learn to identify the corresponding labels.

However, the application of the MLReal transforms in Eqs. (3) and (4), and specifically, the crosscorrelation with a reference trace will alter the vertical axis of the input data, moving the energy closer to zero lag, or in other words, retaining only information on the relative location of events in the vertical dimension (whether the vertical dimension is time or depth), not their absolute locations. If the task is a classification of features not related to the absolute value of the vertical axis, like in the microseismic source location task we shared, this process will not affect the ability of trained models to classify the real data, since the real data are exposed to the same operations. This holds for any task that does not depend on the actual scale of the vertical axis (time or depth) and relies more on the relative location of events vertically, and on lateral behavior of features. This limitation should not be a critical obstacle for seismic data recorded on the Earth's surface as our resolution of depth (the vertical axis) has always been limited. This limitation often manifests itself in the misties we face between seismic and well depths. In other words, the vertical scale, in many seismic tasks, is poorly represented in the data.

Nevertheless, there is a relatively straightforward remedy for such a limitation that would allow us to apply the proposed approach to a wider range of tasks. If the task involves the same size data (in the vertical dimension) for the input and output to the network, like in denoising, super-resolution, and even interpolation of missing traces, the crosscorrelation operation can be applied to both the input and output data. The reference trace should be taken from the same channel location from the input data (as the input data are available for training and application). For prediction, in this case, we will need to apply

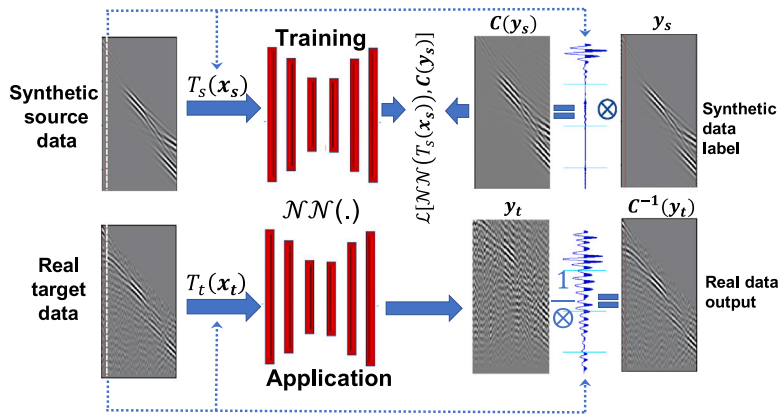


Fig. 15. A diagram describing the steps applied to the output of the NN network to allow the proposed approach to work for applications like denoising and missing trace recovery when the training is performed on synthetic data. For training (top row), the original label ys is correlated with a reference trace from a fixed location from the input data, as the dashed line indicates, to provide the output for the training C(ys). This same reference trace is used in the transform Ts given by Eq. (3). For inference (bottom row), the output of the NN model,yt, is inversely correlated with a fixed location reference trace from the input, as the dashed line indicates, to provide the final output. .

an inverse crosscorrelation (decorrelation) on the predicted data using the same location reference trace used in the training, but taken from the input to the prediction. The inverse crosscorrelation will transform the output of the prediction back to the form of the section we desire. Fig. 15 outlines these steps. The correlation and decorrelation steps could be done in the Fourier domain, as the decorrelation, in this case, implies dividing each output trace by the reference trace. Kinematically, this implies that in the training, we subtract the phase of the reference trace from the input data (crosscorrelation) and then we add it back again to the output of the NN model in the inference stage (inverse crosscorrelation). If the amplitudes of events are not critical, the inverse crosscorrelation can be replaced with convolution, which has exactly the same effect on the phase.

Finally, there is nothing in the proposed MLReal transformations that can prevent their application to waveforms (seismic or electromagnetic) at any scale. The proposed transformations are applied to the vertical dimension of a multi dimensional input to the network, and in the shared examples the vertical dimension was the time axis. The microseismic example can be considered as a miniature test of a potential Earthquake location task. So if the objective is to study and monitor earthquake locations in a certain region using a set of seismic stations (Boschi et al., 2018; Theunissen et al., 2017), we can use a global model (Lei et al., 2020) to simulate synthetic data at the seismic stations, albeit elastic (multi component), from random sources (with random source moment tensors) at the monitored area of interest. In this case, one of the stations are picked as the reference station for crosscorrelation and location of the reference trace is fixed for both synthetic and real data. Then the MLReal transformations will migrate the real data features (events not reproduced by the global model) to the synthetic training set, and vice versa, as shown for the microseismic example in Figs. 7(a) and 7(b).

7. Conclusions

We proposed a novel technique to precondition the synthetic training data set for a supervised neural network optimization so that the trained model works better on real data. The concept is based on incorporating as much information from the real data into the training without harming the synthetic data features crucial for the prediction. Considering the two data domains (synthetic and real), we specifically cross-correlate an input section from one domain of data with a reference trace from that data followed by convolution with an autocorrelated section from the other domain. For training the NN model, the input section is from the synthetic data domain, and for the application (inference) of the NN model, the input section is from the real data domain. A test of this approach on a microseismic source location task using input waveforms helped us improve the application of the NN model on real data. Another application, in which we train an NN to predict low frequencies, the preconditioning helped improve the prediction on real data.

Declaration of competing interest

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

Data availability

The microseismic dataset underlying this article was provided by Microseismic Inc. and Newfield Exploration Mid-Continent Inc. to be used in this study. This dataset will be shared on request to the corresponding author with permission of Microseismic Inc. and Newfield Exploration Mid-Continent Inc. The dataset is detailed in Staněk and Eisner (2017). The marine dataset underlying this article was provided by CGG to be used by our group. This dataset will be shared on request to the corresponding author and with permission of CGG.

Acknowledgments

We thank Umair bin Waheed from KFUPM, Frantisek Stanek from Seismik, and Claire Birnie and Yuanyuan Li from KAUST for helpful discussions. We thank Microseismic, Inc. and Newfield Exploration Mid-Continent, Inc. for graciously supplying the data. We thank CGG for the marine dataset. We also thank KAUST for its support and the seismic wave analysis group (SWAG) for constructive discussions.

References

Aki, K., Richards, P.G., 2009. Quantitative Seismology, second ed. University Science Books, (March 25, 2009).

Araya-Polo, M., Farris, S., Florez, M., 2019. Deep learning-driven velocity model building workflow. Leading Edge 38 (11), 872a1–872a9.

Araya-Polo, M., Jennings, J., Adler, A., Dahlke, T., 2018. Deep-learning tomography. Leading Edge 37 (1), 58–66.

Ben-David, S., Blitzer, J., Crammer, K., Pereira, F., 2007. Analysis of representations for domain adaptation. In: Schölkopf, B., Platt, J., Hoffman, T. (Eds.), Advances in Neural Information Processing Systems, vol. 19. MIT Press.

Birnie, C.E., 2018. Statistical methods for ambient noise characterisation, modelling and suppression: Theory and applications for surface microseismic monitoring. University of Leeds.

Boschi, L., Molinari, I., Reinwald, M., 2018. A simple method for earthquake location by surface-wave time reversal. Geophys. J. Int. 215 (1), 1–21. http://dx.doi.org/10.1093/gji/ggy261.

Bourgeois, A., Jiang, B., Lailly, P., 1989. Linearized inversion: A significant step beyond pre-stack migration. Geophys. J. Int. 99, 435–445.

Choi, Y., Alkhalifah, T., 2011. Source-independent time-domain waveform inversion using convolved wavefields: Application to the encoded multisource waveform inversion. Geophysics 76 (5), R125–R134.

Chollet, F., 2017. Deep Learning with Python. Simon and Schuster.

Di, H., AlRegib, G., 2020. A comparison of seismic saltbody interpretation via neural networks at sample and pattern levels. Geophys. Prospect. 68 (2), 521–535.

Fernando, B., Habrard, A., Sebban, M., Tuytelaars, T., 2013. Unsupervised visual domain adaptation using subspace alignment. In: 2013 IEEE International Conference on Computer Vision. pp. 2960–2967.

He, T., Wu, B., Zhu, X., 2021. Seismic data consecutively missing trace interpolation based on multistage neural network training process. IEEE Geosci. Remote Sens. Lett. 1–5. http://dx.doi.org/10.1109/LGRS.2021.3089585.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition. CVPR, pp. 770–778. http://dx.doi.org/10.1109/CVPR.2016.90.

Iturrarán-Viveros, U., Muñoz-Garcia, A., Castillo-Reyes, O., 2021. Machine learning as a seismic prior velocity model building method for full-waveform inversion: A case study from Colombia. Pure Appl. Geophys. 178, 423–448.

Karamanolakis, G., Mukherjee, S., Zheng, G., Awadallah, A.H., 2021. Self-training with weak supervision. arXiv preprint arXiv:2104.05514.

Kazei, V., Ovcharenko, O., Plotnitskii, P., Peter, D., Zhang, X., Alkhalifah, T., 2021. Mapping full seismic waveforms to vertical velocity profiles by deep learning. Geophysics 86 (5), 1–50.

Keskar, N.S., Mudigere, D., Nocedal, J., Smelyanskiy, M., Tang, P.T.P., 2016. On large-batch training for deep learning: Generalization gap and sharp minima. arXiv preprint arXiv:1609.04836.

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

Koehrsen, W., 2018. Neural network embeddings explained. https://Towardsdatascience.Com/Neural-Network-Embeddings-Explained-4d028e6f0526.

Kouw, W.M., 2018. An introduction to domain adaptation and transfer learning. arXiv abs/1812.11806, arXiv:1812.11806.

Lei, W., Ruan, Y., Bozdağ, E., Peter, D., Lefebvre, M., Komatitsch, D., Tromp, J., Hill, J., Podhorszki, N., Pugmire, D., 2020. Global adjoint tomography—model GLAD-M25. Geophys. J. Int. 223 (1), 1–21. http://dx.doi.org/10.1093/gji/ggaa253.

Lemberger, P., Panico, I., 2020. A primer on domain adaptation. arXiv preprint arXiv:2001.09994.

Luo, H., Xing, Y., Wang, C., Yang, P., 2019. A well-to-seismic calibration method for seismic data in depth domain. In: SEG Technical Program Expanded Abstracts 2019. pp. 1958–1962. http://dx.doi.org/10.1190/segam2019-3211214.1.

Molyneux, J.B., Schmitt, D.R., 1999. First-break timing: Arrival onset times by direct correlation. Geophysics 64 (5), 1492–1501. http://dx.doi.org/10.1190/1.1444653.

Mulayoff, R., Michaeli, T., 2020. Unique properties of flat minima in deep networks. arXiv preprint arXiv:2002.04710.

Ovcharenko, O., Kazei, V., Alkhalifah, T.A., Peter, D.B., 2022. Multi-task learning for low-frequency extrapolation and elastic model building from seismic data. IEEE Trans. Geosci. Remote Sens. 60, 1–17. http://dx.doi.org/10.1109/TGRS.2022.3185794.

Ovcharenko, O., Kazei, V., Kalita, M., Peter, D., Alkhalifah, T., 2019. Deep learning for low-frequency extrapolation from multioffset seismic data. Geophysics 84 (6), R989–R1001.

Palladino, J.A., Slezak, D.F., Ferrante, E., 2020. Unsupervised domain adaptation via CycleGAN for white matter hyperintensity segmentation in multicenter MR images. arXiv preprint arXiv:2009.04985.

Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F.A., Bengio, Y., Courville, A., 2019. On the spectral bias of neural networks.

Smith, L.N., Topin, N., 2019. Super-convergence: Very fast training of neural networks using large learning rates. In: Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications, Vol. 11006. International Society for Optics and Photonics, 1100612.

Staněk, F., Eisner, L., 2017. Seismicity induced by hydraulic fracturing in shales: A bedding plane slip model. J. Geophys. Res.: Solid Earth 122 (10), 7912–7926.

Sun, H., Demanet, L., 2019. Extrapolated full waveform inversion with deep learning. arXiv preprint arXiv:1909.11536.

Tarantola, A., 1984. Inversion of seismic reflection data in the acoustic approximation. Geophysics 49, 1259–1266.

Tarantola, A., 1987a. Inverse Problem Theory. Elsevier.

Tarantola, A., 1987b. Inverse Problem Theory. Elsevier Science Publ. Co..

Theunissen, T., Chevrot, S., Sylvander, M., Monteiller, V., Calvet, M., Villaseñor, A., Benahmed, S., Pauchet, H., Grimaud, F., 2017. Absolute earthquake locations using 3-D versus 1-D velocity models below a local seismic network: Example from the Pyrenees. Geophys. J. Int. 212 (3), 1806–1828. http://dx.doi.org/10.1093/gji/ggx472.

Villani, C., 2008. Optimal Transport: Old and New. In: Grundlehren der mathematischen Wissenschaften, Springer Berlin Heidelberg.

Vinard, N.A., Drijkoningen, G.G., Verschuur, D.J., 2022. Localizing microseismic events on field data using a U-Net-based convolutional neural network trained on synthetic data. Geophysics 87 (2), KS33–KS43. http://dx.doi.org/10.1190/geo2020-0868.1.

Wang, Y., 2016. Seismic Inversion: Theory and Applications. John Wiley & Sons.

Wang, H., Alkhalifah, T., 2021. Direct micro-seismic event location and characterization from passive seismic data using convolutional neural networks. Geophysics 86.

Wang, H., Alkhalifah, T., Hao, Q., 2020. Predict Passive Seismic Events with a Convolutional Neural Network. SEG Technical Program Expanded Abstracts 2020, pp. 2140–2145.

Wang, Y., Tao, X., Qi, X., Shen, X., Jia, J., 2018. Image inpainting via generative multi-column convolutional neural networks. In: Advances in Neural Information Processing Systems. pp. 331–340.

Wrona, T., Pan, I., Gawthorpe, R.L., Fossen, H., 2018. Seismic facies analysis using machine learning. Geophysics 83 (5), O83–O95.

Wu, B., Meng, D., Wang, L., Liu, N., Wang, Y., 2020. Seismic impedance inversion using fully convolutional residual network and transfer learning. IEEE Geosci. Remote Sens. Lett. 17 (12), 2140–2144. http://dx.doi.org/10.1109/LGRS.2019.2963106.

Yu, S., Ma, J., 2021. Deep learning for geophysics: Current and future trends. Rev. Geophys. 59 (3), e2021RG000742. http://dx.doi.org/10.1029/2021RG000742, e2021RG000742 2021RG000742.

Zheng, G., Karamanolakis, G., Shu, K., Awadallah, A.H., 2021. WALNUT: A benchmark on weakly supervised learning for natural language understanding. arXiv preprint arXiv:2108.12603.

Zhou, Z.-H., 2017. A brief introduction to weakly supervised learning. Natl. Sci. Rev. 5 (1), 44–53.