Elastic pre-stack seismic inversion through Discrete Cosine Transform reparameterization and Convolutional Neural Networks

ABSTRACT

We develop a pre-stack inversion algorithm that combines a Discrete Cosine Transform (DCT) reparameterization of data and model spaces with a Convolutional Neural Network (CNN). The CNN is trained to predict the mapping between the DCT-transformed seismic data and the DCT-transformed 2-D elastic model. A convolutional forward modeling based on the full Zoeppritz equations constitutes the link between the elastic properties and the seismic data. The direct sequential co-simulation algorithm with joint probability distribution is used to generate the training and validation datasets under the assumption of a stationary non-parametric prior and a Gaussian variogram model for the elastic properties. The DCT is an orthogonal transformation that is here used as an additional feature extraction technique that reduces the number of unknown parameters in the inversion and the dimensionality of the input and output of the network. The DCT reparameterization also acts as a regularization operator in the model space and allows for the preservation of the lateral and vertical continuity of the elastic properties in the recovered solution. We also implement a Monte Carlo simulation strategy that propagates onto the estimated elastic model the uncertainties related to both noise contamination and network approximation. We focus on synthetic inversions on a realistic subsurface model that mimics a real gas-saturated reservoir hosted in a turbiditic sequence. We compare the outcomes of the implemented algorithm with those provided by a popular linear inversion approach and we also assess the robustness of the CNN inversion to errors in the estimated source wavelet and to erroneous assumptions about the noise statistic. Our tests confirm the applicability of the proposed approach, opening the possibility to estimate the subsurface elastic parameters and the associated uncertainties in near real-time while satisfactorily preserving the assumed spatial variability and the statistical properties of the elastic parameters.

INTRODUCTION

The estimation of elastic properties from processed pre-stack seismic data can be solved through a deterministic or probabilistic inversion process. The former approach starts from an initial guess of the subsurface model and uses gradient-descent algorithms to find an optimal solution in the least-squares sense (e.g. a solution that minimizes the difference between observed and predicted seismic gathers). Probabilistic approaches combine prior assumptions with information brought by the recorded data to infer the so-called posterior probability density (PPD) function in the model space (Tarantola, 2005; Sen and Stoffa, 2013) that fully quantifies the ambiguities in the retrieved solution (Rimstad et al. 2012; Aleardi 2015; Grana et al. 2017; de Figueiredo et al. 2018; Aleardi et al. 2018). However, the PPD can be analytically computed only for Gaussian distributed data and model parameters and for linear forward operators. For non-Gaussian assumptions and non-linear forward operators, numerical Markov Chain Monte Carlo (MCMC) algorithms can be used to draw samples from the PPD (Sambridge and Mosegaard, 2002). In any case, the estimation of elastic properties from pre-stack data is an ill-posed inverse problem in which many models reproduce the observed data, and for this reason model regularization strategies and a-priori information are often infused into the inversion framework to stabilize the solution. However, the inclusion of 2-D or 3D spatial constraints severely increases the computational cost of both analytical and MCMC inversions, especially in high-dimensional parameter spaces. For this reason, this inverse problem is often solved for each seismic gather independently (e.g. for each pre-stack time-migrated gather) thus neglecting the lateral correlations of the sought elastic properties (Grana and Della Rossa, 2010; Aleardi and Salusti, 2020). Geostatistical inversion approaches (Doyen 2007; Bosch et al. 2009; Azevedo and Soares 2017) guarantee that the assumed spatial variability pattern is preserved in the retrieved solution, but their considerable computational cost often hampers the extensive sampling of the model space needed for accurate uncertainty appraisals.

Model reparameterization techniques are commonly used to reduce the dimensionality and the computational complexity of inverse problems and several methods have been proposed using different orthogonal basis functions (e.g. principal component analysis, wavelet transforms, Legendre polynomials, Discrete Cosine Transform). After such reparameterization, the unknown parameters become the numerical coefficients that multiply the basis functions. Some examples of applications of these methods to geophysical problems can be found in Fernández Martínez et al. (2011), Dejtrakulwong et al. (2012), Satija and Caers (2015), Fernández Martínez et al. (2017), Lochbühler et al. (2018), Aleardi (2019), Szabó and Dobróka (2019), Qin et al. (2019), Aleardi (2020). These compression techniques must be applied taking in mind that part of the information in the original (unreduced) parameter space could be lost in the reduced space and for this reason, the model parameterization must always constitute a compromise between model resolution and model uncertainty (Malinverno, 2000; Grana et al. 2019).

In addition to the previously described conventional inversion techniques, over the last years, the recent advent of high-speed multi-core CPUs and GPUs have promoted the applications of machine learning approaches (Monajemi et al., 2016; Goodfellow et al. 2016) to solve geophysical problems. In particular, Convolutional Neural Networks (CNNs) have recently gained attention (Krizhevsky et al. 2012) because overcome some limitation of artificial neural networks such as local minima, overfitting, vanishing gradient, significant computational cost (Schmidhuber, 2015). The hierarchy of local convolution filters and different convolutional layers used by CNN define an ensemble of non-orthogonal basis functions that are used to extract essential features from the training dataset (2-D or 3-D image-like data or 1-D vector signals) or to approximate the function that maps the input into the associated output response. Training a CNN is a supervised learning task that requires a sufficiently large training set to iteratively refine and update the internal network parameters. This learning is an optimization process that minimizes a difference criterion between predicted and desired output. Even though the training is often computationally intensive, once the network is fully trained it can convert an input dataset into the corresponding output response in real-time. In geophysics, CNNs have been initially applied to aid structural interpretation of geophysical data such as seismic horizon and fault interpretation, and seismic texture identification (Xiong et al. 2018;

Waldeland et al. 2018). Then, they have been extended to quantitatively solve geophysical problems such as velocity estimation (Araya-Polo et al. 2018), full-waveform inversion (Lewis and Vigh 2017; Richardson, 2018; Wu and McMechan, 2019; Yang and Ma, 2019), impedance inversion (Das et al. 2019), electromagnetic inversion (Puzyrev, 2019), lithology prediction (Raeesi et al. 2012; Hall, 2016) seismic deblending (Richardson and Feller, 2019; Sun et al. 2020), missing trace interpolation and noise attenuation (Liu et al. 2018; Mandelli et al. 2019; Wang et al. 2018; Kaur et al. 2019), multiple attenuation (Ma, 2018), pre-stack seismic waveform classification and first-break picking (Yuan et al. 2018), automatic velocity analysis (Park and Sacchi, 2020), elastic inversion and reservoir characterization studies (Zhang et al. 2018; Zhong et al. 2019; Biswas et al. 2019). One common issue of machine-learning methods is that their performances sensibly worsens if the target and training data are significantly different. To overcome this issue the transfer learning can be used (Pan and Yang, 2010; Yosinski et al., 2014; Li et al. 2020; Yuan et al. 2020), which consists in an additional training process with a small portion of target data.

In this work, we train a CNN to solve the amplitude versus angle (AVA) inverse problem, in which the subsurface elastic properties of P-wave velocity (*Vp*), S-wave velocity (*Vs*), and density are inferred from partially stacked seismic data at different incidence angles. We also combine the CNN with a Discrete Cosine Transform (DCT; Lochbühler et al. 2014) reparameterization of both data and model spaces. The DCT expands a signal (e.g. expressing the subsurface *Vp* model) into a series of cosine functions oscillating at different frequencies. Usually, most of the variability of the original signal is expressed by the first DCT coefficients (low-order coefficients) and for this reason, this mathematical transformation can be used for model and/or data compression, which is accomplished by setting the coefficients of the basis functions beyond a certain threshold equal to zero. In our application, the DCT constitutes an additional feature extraction technique that reduces both the number of unknown parameters in the elastic inversion and the number of pixels in the images input and output of the CNN.

A crucial aspect of AVA inversion is the preservation of both the mutual and spatial/temporal relationships between the elastic parameters as inferred, for example, from available well log data. In this context, the use of the DCT parameterization in the model space also guarantees that the assumed spatial and temporal variabilities are satisfactorily preserved in the retrieved solution. Indeed, the order of the retained non-zero DCT coefficients determines the wavelength of the recovered elastic model along the spatial and temporal directions. The mutual and spatial/temporal dependencies of the elastic properties are also imposed on the training and validation datasets and to this end, we employ the Direct Sequential Co-Simulation method with joint probability distribution (Co-DSSj; Horta and Soares, 2010). This geostatistical method also allows us to draw random simulations from a non-parametric elastic prior distribution that properly models the non-Gaussian behavior of the elastic parameters. The exact, non-linear Zoeppritz equations constitute the forward modeling operator that computes the observed data associated with each generated model.

The CNN inversion is combined with a Monte Carlo simulation approach to estimate the uncertainties affecting the retrieved solution. To this end, we propagate onto the model space not only the uncertainties related to noise contamination but also the so-called modeling error introduced by the CNN. Indeed, the trained network learns an approximated function that maps the observed data into the associated model and this approximation introduces an additional source of uncertainty. We focus on synthetic inversion experiments in which the reference (true) model mimics a real gas-saturated reservoir hosted in a turbiditic sequence. Different inversion tests that realistically simulate errors in the estimated wavelet and in the assumed noise distribution are carried out to properly assess the robustness and applicability of the implemented approach. The CNN predictions are also compared with those yielded by the popular linear inversion algorithm proposed by Buland and Omre (2003). In the following, we illustrate that the proposed approach provides more accurate estimates of the elastic properties than the standard inversion approach and we illustrate that the CNN inversion is quite robust to erroneous assumptions about the noise statistics and errors in the estimated source wavelet. We will also illustrate that transfer learning can be successfully employed when the statistical properties of the target differ from those assumed during the learning process.

METHODS

Discrete Cosine Transform

The DCT is a linear and orthogonal transformation that projects an *N*-length signal (e.g. vector of model parameters) to an *N*-length vector containing the coefficients of *N* different cosine (base) functions. This approach concentrates most of the information of the original signal into the low-order DCT-coefficients so that only $q<N$ coefficients can be used to accurately approximate the input signal. In the context of geophysical inversion, this means that the numerical values of these *q* DCT coefficients become the unknowns to be inferred from the data. Estimating the retained DCT-coefficients reduces the parameter space dimensionality and can significantly improve the computational efficiency of the inversion procedure.

Several variants of DCT exist with slightly modified definitions, but in this work, we use the so-called DCT-2 formulation that is the most common. Hereafter we simply refer to the DCT-2 transformation as the DCT. The DCT is a Fourier-related transform that uses only real numbers to express a finite signal in terms of the sum of cosine functions oscillating at different frequencies. The DCT transformation of a 1-D signal *x* of length *N*, can be written as follows:

$$y(k) = \sqrt{\frac{2}{N}} \sum_{n=1}^{N} x(n) \frac{1}{\sqrt{1+\delta_{k1}}} \cos\left(\frac{\pi}{2N}(2n-1)(k-1)\right), \quad (1)$$

where δ_{k1} represents the Kronecker delta, *y* are the *N* coefficients of the transformation that fully describe the original signal *x* in the transformed DCT space, and *k* represents the order of each DCT coefficient. In matrix form, equation 1 becomes:

$$\mathbf{y} = \mathbf{Bx}, \quad (2)$$

where the vectors **x** and **y** represent the original and the transformed signal, respectively, and **B** is a *N*-by-*N* matrix containing the cosine functions (basis function) spanning the DCT space. Some examples of 1-D DCT basis functions of different orders k are represented in Figure 1. The DCT is a linear transformation expressed by the orthonormal matrix $\mathbf{B}$ (so that $\mathbf{B}\mathbf{B}^T = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix). An approximation of the signal $\mathbf{x}$ can be obtained by considering only the first q DCT basis functions:

$$\tilde{\mathbf{x}} = \mathbf{B}_q^T \mathbf{y}_q, \quad (3)$$

where $\tilde{\mathbf{x}}$ is the approximated signal, $\mathbf{B}_q^T$ is a partition of the matrix $\mathbf{B}$, with N rows and q columns representing the first q DCT basis functions, whereas the vector $\mathbf{y}_q$ contains the first q coefficients that multiply the basis functions. These coefficients become the unknown parameters to be determined in a DCT reparameterization of the inverse problem. Note that the approximated signal $\tilde{\mathbf{x}}$ tends toward the original signal $\mathbf{x}$ as the number of considered coefficients q tends to N.

The DCT can be also extended to multi-dimensional signals (i.e. 2-D matrices) and such multi-dimensional DCT transform follows straightforwardly from the one-dimensional definition because it is simply a separable product (equivalently, a composition) of DCTs along each dimension. For example Figure 2 shows some DCT basis functions of different orders in a 2-D space. Note that the variability of the solution along each dimension is directly determined to the orders of the retained DCT coefficients.

Convolutional neural networks

The relation between a CNN and its generated model is usually expressed as follows:

$$\mathbf{O} = F(\mathbf{W}, \mathbf{L}), \quad (4)$$

where F denotes the CNN as a function that maps the input $\mathbf{L}$ to the output $\mathbf{O}$ through the CNN internal parameters $\mathbf{W}$. CNNs use blocks of convolutional layers, subsampling layers, and fully connected layers, to extract features from 1-D, 2-D, or 3-D input maps treated as grids of pixels. The extracted features form the so-called feature maps. The core of CNNs is the convolutional layer, in which the features maps are convolved with convolution filters. This process can be written as:

$$O_j^p = f\left(b_j + \sum_{i=1}^{I} O_i^{p-1} * W_j\right), \quad j = 1, 2, \dots, J \qquad (5)$$

where I represents the number of the feature maps in the (p-1)-th layer, whereas J is the total number of feature maps in the p-th layer, which is equal to the number of filters considered in that layer; b_j is a scalar value representing the j-th bias of the p-th layer, $*$ represents the convolution operator, $f()$ is the so-called activation function used to include non-linearity in the mapping process, O_j^p is j-th feature map in the p-th layer, O_i^{p-1} represents the i-th feature map in the (p-1)-th layer, and W_j denotes the j-th filter of the p-th layer, which is the weight matrix connecting O_j^p with O_i^{p-1}. This filter has a user-specified size and slides over the input map with a specified stride. The internal CNN parameters to be updated are the values associated with the filters W_j and the biases b_j in each layer. More in detail, in this example we have I input maps in the (p-1)-th layer, J outputs maps in the p-th layer and if we also assume that $n\times m$ is the size of each convolution filter, it results that the number of internal parameters to be optimized is $(n\times m\times I+1)\times J$. Note that different filters extract different features from the input map, or in other terms they use non-orthogonal basis functions to project the input map onto an alternative space. The aim of the subsampling layers (also known as pooling) is to prevent overfitting by reducing the dimension of the feature map generated in the convolutional layer and the number of features. The most common pooling strategies are the max-pooling or the average pooling (Scherer et al. 2010).

After the features of the input image are extracted by the convolutional layers, they are usually fed into fully connected layers, which are appended to the end of the last convolutional blocks. The output of the fully connected layer depends on the specific research requirement and can be either continuous values (regression problems) or discrete values (classification problems). Differently, from the convolutional layers, in a fully-connected layer, each input data have a separable weight to each output, and a bias value is also associated with each output node. Therefore, an I-dimensional input and a J-dimensional output result in $(I+1)\times J$ parameters to be optimized.

At the first iteration, the internal CNN parameters are initialized and then updated during the iterative learning process. The learning process aims to minimize an error (loss) function measuring the difference between the desired and the computed output. A back-propagation algorithm usually drives the minimization process and the updating of the filter values. This updating process can be written as follows:

$$\mathbf{W}_i = \mathbf{W}_{i-1} - \gamma \frac{\partial \varepsilon}{\partial \mathbf{W}_{i-1}}, \qquad (6)$$

where i represents the iteration number, ε is the loss function value, and γ is the so-called learning rate.

To define the CNN architecture some hyperparameters must be set: number of hidden layers and number of filters, kernel width and stride of the convolution operators, activation function, a method for weight initialization, optimization algorithm to minimize the loss function and to update the filters weights, number of epochs. There are no rigid rules to set these hyperparameters and the final choice is often dictated by personal preference and experience. However, some very general rules exist such as the dimension of the convolution filters must not be greater than the dimension of the features map to which the filter is applied. Moreover, more filters are used, the more features can be extracted, but the number of features is limited and specific for the problem at hand. Therefore, too many extra filters make the model difficult to be well trained and increase the training time. On the contrary, a too low number of filters can produce underfitting. We found the optimal setting through a trial and error procedure in which different hyperparameters are changed (i.e. number of filters, filter size, learning rate, batch size, and the type of activation function) and the final net architecture has been determined based on the net performances on the validation dataset.

The implemented CNN inversion

For the lack of available real seismic data, we discuss synthetic inversion experiments in which the trained CNN is used to invert a seismic dataset generated on a reference model that simulates a real geological context in which a turbiditic sequence hosts a gas-saturated reservoir (see Figure 3a). This subsurface model has been derived by integrating the borehole information provided by five wells with an accurate geologic interpretation. The reference model we employ represents an in-line section constituted by 51 time samples and 71 cross-lines. The time sampling is 0.004 s, whereas 50 m is the distance between the cross-lines. In Figure 3a note that significant elastic contrasts at the interface separating the encasing shales from the reservoir sands. In our application each elastic model comprises $51 \times 71 \times 3 = 10863$ parameters to be estimated from $50 \times 71 \times 3 = 10650$ data points.

To generate the training and validation datasets we must define the expected statistical properties of the elastic parameters in the investigated area. The prior elastic model has been inferred from the information provided by the five wells that have also been used to build the reference model. We consider this prior information as stationary over the entire study area, and given the structural complexity of the reference model this assumption constitutes a simplification of the actual variability of the elastic properties. We assume a non-parametric distribution for the elastic parameters that has been derived by applying the kernel density estimation algorithm (KDE; Parzen 1962) to the available well log data (Figure 3b). For example, for a univariate random variable r, the KDE estimates of the probability distribution $p(r)$ can be computed as:

$$p(r) = \frac{1}{T}\sum_{n=1}^{T} H\left(\frac{r - r_{\mathrm{n}}}{h_r}\right) \quad (7)$$

where H is the kernel function, T is the total number of data points, and h_r is the kernel width that controls the smoothness of the distribution and should be set assessed on the available data. In this work, a simple Gaussian kernel is adopted. We also assume a stationary 2D Gaussian variogram model in which the vertical and lateral ranges have been inferred from the vertical variability of the upscaled well log data and from the lateral variability of the observed seismic data, respectively. The ranges of the variogram are equal to 0.008 ms and 160 m along the temporal (vertical) and spatial (lateral) directions, respectively. The direct-sequential co-simulation method with joint probability distribution has been used to draw 20000 2-D elastic models from the prior distribution. A forward modeling based on the full Zoeppritz equations computes the observed seismic gathers by convolving the angle-dependent reflectivity series with a 30-Hz, zero-phase Ricker wavelet. Then, the so obtained observed data are contaminated with Gaussian uncorrelated noise with a standard deviation of 0.02, which corresponds to the 35% percent of the standard deviation of the noise-free observed dataset. We consider three partial angle stacks corresponding to incidence angles of 0 (near stack), 15 (mid stack), and 30 (far stack) degrees. Note that the use of the full Zoeppritz equations makes it possible to extend the considered angular range. However, the linear inversion approach used as a benchmark is only applicable in a narrow-angle range where the linear approximation of the Zoeppritz equation holds. Therefore, for comparability reasons the seismic data are computed over this limited angle range. The information used to define the training and validation sets are summarized in Table 1.

After the generation of the training and validation examples, the next step involves the estimation of the optimal number of DCT coefficients needed to approximate the elastic models and the associated seismic response. The optimal number of DCT coefficients in the data and model spaces are determined by decomposing some of the 20000 *Vp*, *Vs,* and density models and seismic gathers previously computed. Figure 4 shows an example of a DCT transformation of a seismic dataset and the associated *Vp, Vs,* and density model, whereas Figure 5 illustrates the explained variability of the seismic data and elastic properties as the number of the considered DCT coefficients increases. We observe that 500 and 2000 coefficients explain almost the total variability of the original, un-compressed elastic models and seismic data, respectively. Therefore, to approximate the *Vp*, *Vs,* and density we retain the first 24 rows and columns of the associated DCT matrices, while in the seismic domain we consider the first 30 rows and 30 columns of the first two slices forming the DCT cube (see the green box and rectangles in Figures 4). Therefore, the use of DCT allows us to compress the 10863-D full elastic space into a 1728-D space, while the 10650-D data domain has been compressed into an 1800-D space. In the implemented approach the DCT transformation acts as an additional feature extraction layer that reduces both the number of unknown parameters to invert for and the dimensionality of the input and output of the network. Therefore, the use of the DCT allows for a reduction of the complexity of the CNN architecture (i.e. number of hidden layers) needed to accurately convert the input into the associated output response. This translates into an easier CNN hyperparameter setting and a faster training phase because less CNN parameters must be adjusted.

Figure 6 describes the entire workflow of the implemented CNN inversion that starting from the observed partially stacked data retrieves the elastic properties. However, one crucial step should be the assessment of the uncertainty affecting the estimated solution. In particular, we must project onto the model space both the noise affecting the seismic data and the so-called modeling error introduced by the CNN approximation. In the following discussion we describe our Monte Carlo approach to quantify and project such errors: Let **M** represent the ensemble subsurface elastic models forming the training dataset, while **N** is the associated ensemble of models predicted by the trained CNN. A sample of the modelings error can be obtained as $\mathbf{E} = \mathbf{M} - \mathbf{N}$ (Hansen and Cordua, 2017)**.** Assuming a Gaussian distribution, the modeling error can be defined as $\mathcal{N}(0, \mathbf{C}_e)$, where $\mathbf{C}_e$ is the covariance of **E.** This error together with the noise term $\mathcal{N}(0, \mathbf{C}_n)$ (also assumed Gaussian-distributed) are propagated onto the final prediction with an iterative MC approach. Now, let **d** be the vector expressing the observed data input to the CNN, whereas *n* represents the number of MC simulations. The implemented MC approach for uncertainty propagation comprises the following six steps:

1) Use the trained CNN to compute the predicted elastic model $\mathbf{m}_\mathrm{b}$ from the observed data vector **d**;
2) Run a forward modeling to compute the noise-free $\mathbf{d}_\mathrm{b}$ data associated to $\mathbf{m}_\mathrm{b}$;
3) Draw **n** from $\mathcal{N}(0, \mathbf{C}_n)$ and compute $\mathbf{d}_\mathrm{n} = \mathbf{d}_\mathrm{b} + \mathbf{n}$;
4) Use the trained CNN to compute the predicted model $\mathbf{m}_\mathrm{n}$ from $\mathbf{d}_\mathrm{n}$;
5) Draw **e** from $\mathcal{N}(0, \mathbf{C}_e)$ and compute $\mathbf{m}_\mathrm{e} = \mathbf{m}_\mathrm{n} + \mathbf{e}$;
6) Store $\mathbf{m}_\mathrm{e}$ and repeat from 3) to 6) for *n* times.

Each generated vector $\mathbf{m}_\mathrm{e}$ can be considered a possible subsurface elastic model that is in accordance with the observed data, the trained CNN, and the assumed distributions for the noise and modeling errors. The posterior statistic (e.g. uncertainty on the estimated elastic model) can be numerically derived from the ensemble of n MC simulations. For simplicity, we assume that both the error terms (noise and modeling errors) are Gaussian, but the implemented approach can be applied to whatever parametric or non-parametric error distributions. Note that the previous MC approach is extremely fast because the network predicts a model from an input data (steps 1 and 4) in real-time. The most time-consuming step of this procedure is represented by the fifth step that draws spatially correlated realizations from $\mathcal{N}(0, \mathbf{C}_e)$. To reduce the computational cost of this step we adopt the FFT-MA simulation algorithm (Ravalec and Noetinger, 2000).

The CNN architecture

We randomly select 18000 out of 20000 models and associated seismic datasets to train the CNN, whereas the remaining 2000 examples constitute the validation set. The CNNs usually consists of one input layer, one or more hidden convolutional layers, one or several fully connected layers, and one output layer. In our case, the training ensemble is constituted by a tensor of $30 \times 30 \times 2 \ \times 18000$ where 18000 is the number of training examples, whereas $30 \times 30 \times 2 = 1800$ is the number of the retained DCT basis functions in the data space. The corresponding output is a vector of $24 \ \times 24 \ \times 3 \times 18000$ DCT coefficients where $24 \ \times 24 \ \times 3 = 1728$ represents the number of DCT basis functions used to compress the elastic parameters (576 coefficients for elastic property). We perform different experiments to optimally set the main CNN hyperparameters, and the final CNN architecture (represented in Figure 7) has been chosen according to the best fit on the validation set. It consists of two convolutional blocks and a fully connected layer. The two convolutional blocks use 5 convolution filters of size 3 × 3, and a stride of 1. After each convolutional layer, we use the LeakyRelu activation function with a slope of 0.1 (Krizhevsky et al., 2012). Batch normalization is used within each convolution block because Santurkar et al. (2018) suggested that it guarantees a more predictive and stable behavior of the gradients and faster training. Note that applying batch normalization to a given layer associated with N features maps, adds $N \times 2$ trainable parameters.

After the convolutional blocks, max-pooling of size 2 × 2 and a stride 1 is applied for subsampling. In the fully connected layer, the feature maps are flattened to 1-D vectors, whereas a dropout of 0.1 is used to prevent overfitting. We adopt the RMSprop optimizer (i.e. an unpublished, adaptive learning rate method) running for 20 epochs to minimize the root-mean-square-error (RMSE) between the expected and the predicted outputs. The He method is used to initialize the network weights (He et al., 2015). We use a batch size of 64, and an initial learning rate of 0.001 that is multiplied by 0.9 every epoch. We chose this batch size and initial learning rate because they guaranteed the best performances in the experiments we carried out. In particular, the benefit of small batch sizes has been discussed in many studies (e.g. Masters and Luschi, 2018). The selected network configuration results in 4978713 learnable parameters. Only 325 of these parameters ($(3 \times 3 \times 2 + 1) \times 5 + (3 \times 3 \times 5 + 1) \times 5 = 325$) pertain to the convolution filters in the two convolutional blocks, 20 parameters are associated with the batch normalization within the two blocks ($(5 + 5) \times 2 = 20$), while the remaining 4978368 ($(2880 + 1) \times 1728 = 4978368$) pertain to the fully-connected layer. From the evolution of the RMSE values on the training and the validation sets, we observe that the network successfully converges after 15-16 epochs (Figure 8).

Figure 9 shows some comparisons between two elastic models extracted from the validation set and the corresponding CNN predictions. The good quality of the results seems to demonstrate the reliability of the implemented CNN inversion. Just for comparison, Figure 10 represents the evolution of the RMSE error for a CNN with the same architecture previously described (Figure 7) but now the DCT has not been applied to compress the input and output responses, while the training and the validation sets are the same previously used. In other terms, in this experiment, the CNN has been trained to directly predict the 10863 elastic properties form the 10650 data points. By comparing Figure 10 with Figure 8, we observe a much slower convergence rate and higher errors on the training and validation sets that indicate the poor generalization capability of the network. If we applied this trained network to the validation set we estimate the elastic models shown in Figure 11. These severely scattered and unrealistic predictions prove that a much more sophisticated network and a longer training are now needed to directly map the seismic response into the associated elastic properties. This example clearly illustrates the benefits provided by the DCT compression.

Figure 12 compares the lateral and vertical correlation functions used to generate the training and validation datasets with those computed on the predicted elastic models of Figure 9a. We observe that the number of retained DCT coefficients in the model space guarantees a satisfactory preservation of the assumed spatial/temporal continuity in the recovered solution. However, all the previous results refer to elastic models that perfectly honor the statistical characteristics of the prior used to generate the training dataset. For this reason, in the following section, we describe the application of the CNN inversion to the more realistic reference model (test set) previously described (see Figure 3a). The CNN results will be also compared with the predictions of the popular linear inversion approach proposed by Buland and Omre (2003). We will also assess the robustness of the CNN inversion to errors in the estimated source wavelet and erroneous assumptions about the statistical properties of the noise. Finally, we will show an application of transfer learning.

INVERSION EXAMPLES

In this section, we describe the application of the trained CNN to invert synthetic seismic data computed on the test model shown in Figure 3a. We assess the robustness and the applicability of the trained CNN inversion by performing several inversion tests that mimic realistic conditions than can be encountered in real data applications. The details of the seven inversion experiments are given in the following and also summarized in Table 2:

Test 1: The noise standard deviation and the source wavelet used to generate the observed data input to the CNN are the same previously used to generate the training and validation sets (see Table 1). This test is aimed at assessing the network performance in the most optimal case.

Test 2: The standard deviation of the Gaussian and uncorrelated noise that contaminates the data is higher than that assumed in the training phase. The numerical value of the standard deviation is 0.05, which corresponds to the 88%, approximately, of the standard deviation of the noise-free observed data. This test illustrates the network performances in case of an underestimation of the actual noise contamination in the data.

Test 3: In this case, we add to the observed data both uncorrelated and spatially-temporarily correlated Gaussian noise. Both types of noise have a standard deviation equal to 0.03. This test illustrates the network performances in case of correlated noise affecting the data. Indeed, the popular assumption of uncorrelated noise constitutes an oversimplification and correlated noise can be ascribed, for example, to residual of multiple reflections than have been not successfully removed during the processing phase.

Test 4: The source wavelet used to generate the data has a peak frequency of 25 Hz that is smaller than that used to generate the training set. The noise statistic and the source wavelet phase are the same used for the learning process.

Test 5: The source wavelet is derived by applying a phase rotation of 30 degrees to the wavelet used in the training process.

Test 6: The source wavelet is characterized by a peak frequency of 25 Hz and a phase rotation of 30 degrees with respect to the zero-phase wavelet used in the training phase. Examples 4, 5, and 6 investigates the stability of the network predictions in case of errors in the estimated source wavelet.

Test 7: The last example is a combination of Tests 3 and 6. We generate the observed data with a source wavelet with a peak frequency of 25 Hz and a 30°-phase rotation. The observed data are also contaminated with both uncorrelated and coherent noise with standard deviations of 0.03.

Figure 13 shows a comparison between the source wavelet used to generate the training examples and those employed in Tests 4, 5, 6, and 7 to generate the observed dataset. In all the 7 tests the CNN outcomes are compared with the maximum-a-posteriori solution provided by the linear inversion approach of Buland and Omre (2003) that inverts each seismic gather separately. For brevity in the following discussion, we refer to this inversion method as the standard approach (SA). Many methods can be used to quantitatively compare the true and predicted models (Wang et al. 2004; Zujovic et al.

2013; Yang and Duan, 2018). In this work, the normalized cross-correlation (Rao et al. 2014) is used to assess the quality of the CNN and SA outcomes. Note that the linear inversion assumes a log-Gaussian distribution for the elastic properties and imposes vertical constraints on the recovered model, while it neglects the lateral correlation of elastic parameters. In all the linear inversions the data covariance matrix has been computed assuming the noise statistic used to generate the training examples (i.e. Gaussian uncorrelated noise with a standard deviation of 0.02). The vertical correlation range included in the a-priori model covariance matrix is equal to that used to generate the training and validation sets (8 ms).

For the first inversion experiment, both the CNN and the linear inversion retrieve similar results that accurately reproduce the lateral and vertical variability of the true *Vp, Vs,* and density models (Figures 14a, 15a). However, the noise contamination generates some lateral scattering in the SA solution, while the DCT parameterization preserves the lateral continuity in the CNN estimated model. The lateral scattering affecting the SA results aggravates in the second test in which we underestimate the noise affecting the data (Figures 14b, and 15b). It is expected that this underprediction generates overfitting with the observed data that significantly deteriorates the overall quality of both the CNN and SA predictions. The inclusion of coherent noise results in biased predictions, which are particularly visible around the reservoir zone where the elastic contrasts between the encasing shales and the reservoir sands are mispredicted (Figures 14c and 15c). As expected, errors in the source wavelet yield biased predictions (Figures 14d-f and 15d-f) characterized by both overpredicted or underpredicted elastic contrasts (e.g. see the results of Test 6 in which we simulate errors in both the source wavelet peak frequency and phase). The combined presence of errors in the wavelet estimation and coherent noise in the data significantly deteriorates the quality of the retrieved elastic models (Figures 14g and 15g). However, also in this unfavorable scenario the CNN approach satisfactorily preserves the lateral continuity of the *Vp, Vs,* and density predictions, thereby facilitating the mapping of the lateral and vertical formation boundaries. In all the tests and for both the CNN and SA approaches, the erroneous assumptions about source wavelet and noise particularly affect the density estimates, which is the parameter least constrained by the observed data.

The direct comparison of the normalized cross-correlation coefficients associated with the CNN and SA inversions demonstrates that in all the considered examples the implemented approach guarantees final predictions closer to the true elastic model (Figure 16). As expected, for both inversions the quality of the final results usually decreases passing from *Vp*, *Vs*, and density in accordance with the different sensitivity of the observed data to the three elastic properties. These results confirm the applicability of the CNN inversion and its suitability to invert seismic data also in realistic scenarios of heavy noise contamination and errors in the assumed noise statics and estimated source wavelet.

In Figures 17-20 we compare for the Tests 2 and 3 the observed and predicted data computed on the final models yielded by the CNN and SA inversions. The sample-by-sample difference between the observed and predicted seismic responses appear slightly lower in the SA example. However, note that the solution provided by this approach corresponds to the minimum of an error function that is expressed as a linear combination of data misfit and a-priori model constraints. Differently, the CNN solution is not driven by a minimization process but it is simply obtained by applying the previously trained network to the observed data. It is also important to mention that the CNN inversion imposes both vertical and lateral constraints to the solution, while only vertical constraints are imposed by the SA algorithm. Therefore, we expect that the inclusion of the additional lateral constraint promotes the stability of the CNN solution but at the expense of a decreased data matching with respect to the laterally unconstrained inversion.

Figure 21 compares the marginal *Vp*, *Vs*, and density distributions directly computed on the true model and those derived from the CNN predictions for Tests 1 and 2. When the noise is correctly estimated (Figure 21a) the marginal distributions computed on the final predictions show good agreement with those derived on the true model. Otherwise, when the noise is underestimated (Figure 21b) the predicted distributions exhibit longer tails than the true distributions, and particularly the retrieved density model shows more disperse values. Again, this degradation of the density prediction is expected since this is the least resolvable elastic property. However, also in this scenario, important statistical properties of the true and predicted model are very similar (i.e. the mode). In this regard, note that the statistical properties of the a-priori model used to generate the training examples are not directly derived from the true model but they are realistically inferred from well log data recorded in the study area. In our opinion, this fact can explain the minor deviations between the true and predicted distributions, for example, the different variability ranges of the true and recovered models. This comparison illustrates that the implemented inversion approach can satisfactorily preserve the actual distribution of the elastic parameters, provided that the statistical properties of the prior distribution used to generate the training examples accurately capture the actual distribution of the *Vp*, *Vs*, and density values in the investigated area.

To better analyze this crucial aspect we directly apply the previously trained network to invert seismic data computed on a completely different elastic model (i.e. a portion of the elastic Marmousi model). Figure 22 compares the new reference model, the CNN predictions, and the SA result. For comparability reason, the SA uses the same prior model previously employed in the inversion tests of Figure 15, while for both algorithms the noise variance is assumed perfectly known. The SA and CNN give congruent predictions that satisfactorily preserve the geological structure of the model, but both approaches significantly underestimate the true elastic contrasts at the reflecting interfaces. In particular, in Figure 22c note that the actual distributions of the elastic properties in this Marmousi example are completely different from the a-priori elastic model employed in the learning process (Figure 3b). This example highlights that accurate prior model assumptions, which exhaustively capture the actual distribution of the elastic properties in the study area, are a crucial ingredient for both standard and the machine-learning inversion approaches.

However, this does not mean that the network must be retrained from scratch when applied to different datasets. Indeed, the transfer learning can be employed when we have an unsatisfying result with the previously trained CNN model (Pan and Yang, 2010, Yang and Ma, 2019; Sun et al. 2020;

Park and Sacchi 2020). Transfer learning is commonly used in deep learning applications and it takes a pre-trained network and uses it as a starting point to learn a new task. Fine-tuning a network with transfer learning is usually much faster and easier than training a network with randomly initialized weights. This allows for a quick transfer of the learned features to a new task using a smaller number of training examples. When using transfer learning, it is important to decide which part of the already trained CNN model must be updated and this usually depends on the difference between the target and the training data. For example, only the fully connected layers are updated if this difference is small. In the opposite case, the entire pre-trained CNN model should be adjusted (Huot et al., 2018; Park and Sacchi 2020). Now we repeat the Marmousi example but before the CNN inversion, we use transfer learning to update part of the internal parameters of the previously trained network, according to new 500 training examples with 50 examples forming the validation set. After some tests (not shown here for brevity) we decided to update only the parameters associated with the second convolutional block and the fully-connected layer. The prior elastic model used to generate the new training and validation sets is numerically defined with the KDE algorithm now applied to the first, central, and last columns of the true elastic model shown in Figure 22a, which are now considered as available well log data. For simplicity, the assumed variogram model is the same previously employed. Figures 23a and 23b shows that transfer learning significantly improves the CNN predictions especially for those parameters better constrained by the data (*Vp* and secondarily, *Vs*). Figure 23c illustrates that transfer learning ensures a final error values on both the validation and test sets similar to those achieved in the previous application (Figure 8). In this case, re-training the CNN with transfer learning takes less than a minute. However, even though transfer learning can be applied to adjust the network weights when the target properties differ from those of the training examples, an accurate estimation of the prior is always crucial to obtain a new training set that exhaustively captures the target properties.

Figures 24-25 illustrate nine MC *Vp, Vs,* and density realizations generated for Test 1. All these realizations are in accordance with the assumed noise static and with the CNN modeling error. As expected the differences between the nine realizations increase moving from *Vp*, to *Vs*, and density. Finally, Figure 26 shows for Test 1 the CNN predictions, some MC realizations, and the true *Vp*, *Vs*, and density profiles extracted at the horizontal coordinate of 1000 m. We observe that the estimated elastic properties always lie within the 95 % confidence intervals that have been numerically estimated from the MC simulations. This proves the reliability of the CNN results.

DISCUSSION

Often, the main practical limitations of spatially-constrained linear inversions is the high computational effort to invert matrices with considerable dimensions. Similarly, the applicability of geostatistical inversions is discouraged by their computational cost. Finally, the application of numerical MCMC inversions with 2-D or 3-D model constraints is hampered by the considerable number of forward evaluations needed to attain stable posterior estimations. The popularity of machine learning approaches has motivated us to develop an alternative inversion approach characterized by a modest computational demand while being robust and capable to also provide a quantification of the uncertainties affecting the final solution. Our efforts were devoted to combining the regression capability of CNN and the compression ability of DCT. In our application, the use of DCT makes it possible for a substantial dimensionality reduction of the input and output of the network, thereby reducing the complexity of the CNN architecture and the computational cost of the training phase. Other compression techniques can be used but here we apply the DCT because it has been proven to exhibit superior compression power over other similar methods (see Lochbühler et al. 2014; Kotsi et al. 2020). The choice of the number of DCT coefficients to approximate the data and the elastic model should always constitute a compromise between the desired temporal and spatial resolution, and the dimensionality reduction of the parameter space. However, our examples showed that such a threshold level can be accurately determined from the elastic models and associated seismic responses extracted from the training set.

The implemented method does not require the regularization in its common-sense meaning (i.e. inclusion of model constraints into the error function), but the network is trained on a data set containing realistic subsurface scenarios and thus learns how to reproduce a similar model that fits the input data. In machine learning applications, regularization aims at reducing the generalization error of a learning algorithm and at making the network performances more stable during the learning phase, often by sacrificing the training error. The most popular regularization approaches are dropout, early stopping, and batch normalization. Following the recent trend in modern convolutional architectures (Puzyrev, 2019) we used dropout only on the fully-connected layer, while batch normalization was applied in the convolutional blocks.

The main advantage of the CNN-based inversion is its high computational efficiency. Indeed, all the examples discussed in this paper used Matlab codes running on a common notebook equipped with a quad-core intel(R) Core(TM) i-7 7700HQ CPU@2.80 GHz with 16 Gb RAM. The implemented inversion consists of four stages: data generation, learning process, model prediction from a given data input, and Monte Carlo simulation for posterior model generation. The first stage can be performed in various ways and here we use the Co-DSSj geostatistical simulation algorithm that can handle non-parametric prior models while preserving the joint distribution of the elastic properties to be simulated. In this work we use a non-parametric and stationary elastic prior model and a Gaussian variogram to generate the training and validation datasets, but if needed the method can be easily extended to whatever stationary or non-stationary prior distribution and spatial correlation pattern provided that appropriate geostatistical simulation codes are available to generate the training and validation examples. This first stage can be very time consuming although it is perfectly parallelizable. The computing time for generating the ensemble of 20000 training models with the Co-DSSj method was 6 hours approximately, whereas the training phase took only 10 minutes. It is to mention that the computing time for the ensemble generation can be drastically reduced if more scalable codes and a computer cluster are employed. The CNN inversion gives predictions in real-time, while the MC simulation of the nine elastic models shown in Figures 24-26 runs in less than 2 seconds. As a side note, we point out that the computing time for training the CNN without the DCT compression (Figures 10 and 11) was 70 minutes, approximately. This value is more than 10 times longer than that needed for training the CNN model after the DCT compression. This highlights another outstanding benefit provided by the DCT. Indeed, in this case, many networks with different hyperparameter settings can be evaluated rapidly, thereby reducing the human and computational efforts required for the network configuration.

In view of 3-D CNN inversions, much effort must be devoted to decreasing the computational effort of the data generation process, although there is still room for a substantial decrease of its computational cost, for example by adopting more efficient geostatistical simulation codes. To this end, we are also exploring the possibility to train a Generative Adversarial Neural Network to quickly generate 2-D and 3-D elastic models (Laloy et al. 2018). Additional tests we carried out (not shown here for brevity) demonstrated that similar results to those shown here can be obtained by employing only 10000 models for the training phase. This possibility halves the computing time of the Co-DSSj simulation process.

Our experiments also highlighted that the quality of the predictions significantly decreases if the actual subsurface parameter distributions differ from those used to generate the training examples. However, transfer learning can be employed to update the internal network weights when the statistical properties of the target differ from those of the training examples. Anyway, as for any standard Bayesian inversion approach, a high-quality and rich suite of well log data and/or borehole information is needed for an accurate and robust estimation of the a-priori statistical properties of the target. In view of field data applications, such well log information and blind well tests are of primary importance to also compare the quality of the CNN predictions with those provided by other inversion approaches.

In this work, we did not perform an extensive and quantitative analysis of how errors in the assumed prior model properties (e.g. error on the a-priori mean and variance, erroneous variogram model, incomplete well log information) affect the CNN predictions. A comprehensive analysis of these aspects would have required many additional figures and tables and certainly deserves a paper of its own. Preliminary tests we are carrying out demonstrate that, as expected, the quality of the CNN predictions is much more affected by errors in the prior model than by erroneous assumptions on the noise statistic. However, we also found that the implemented CNN inversion provides satisfactory predictions, comparable to those yielded by the SA method, if the error, for example, on the a-priori mean and variance is less than the 15%-20%.

CONCLUSIONS

We proposed a CNN inversion method that presents an alternative to the conventional AVA inversions approaches. Instead of minimizing an error function or sampling from a posterior probability distribution, the proposed approach employs a trained CNN to reconstruct the elastic parameters from the observed pre-stack seismic gathers. We use a Discrete Cosine Transform (DCT) reparameterization of data and model spaces to reduce the computational effort of the training phase. In our case, the DCT constitutes an additional feature extraction tool that uses orthogonal basis functions to compress the dimensionality of the input and output of the CNN. On the other hand, the DCT reparameterization also acts as a model regularization strategy that reduces the number of unknown parameters to be estimated and satisfactorily preserves the assumed lateral continuity in the recovered solution. The CNN inversion is combined with a subsequent Monte Carlo simulation process for uncertainty appraisals, which takes into account both the noise affecting the observed data and the modeling error associated with the network approximation.

Our synthetic experiments showed very promising results and demonstrated that a convolutional neural network can effectively approximate the inverse of a nonlinear operator that is very difficult and expensive to compute analytically especially when vertical and lateral constraints are imposed on the recovered solution. The learned network achieved satisfactory predictions when realistic conditions were simulated (e.g. errors in the estimated source wavelet or erroneous assumptions about the noise statistic). In all the tests we carried out the CNN provided more accurate predictions than the standard, laterally-unconstrained AVA inversion approach (SA). Notably, the quality of the CNN predictions seemed less affected by erroneous assumptions on the noise statistic and errors in the estimated source wavelet than the outcomes provided by the SA method. We illustrated that transfer learning can be used to update the network weights when the properties of the target model differ from those of the training examples. In any case, reliable prior model assumptions, and high-quality seismic data are essential for a successful application of the implemented inversion approach to field datasets.

REFERENCES

Aleardi, M., 2020, Discrete cosine transform for parameter space reduction in linear and non-linear AVA inversions: Journal of Applied Geophysics, 104106. In print.

Aleardi, M., and A. Salusti, 2020, Markov chain Monte Carlo algorithms for target-oriented and interval-oriented amplitude versus angle inversions with non-parametric priors and non-linear forward modellings: Geophysical Prospecting, **68**, no. 3, 735-760.

Aleardi, M., 2019, Using orthogonal Legendre polynomials to parameterize global geophysical optimizations: Applications to seismic-petrophysical inversion and 1D elastic full-waveform inversion: Geophysical Prospecting, **67**, no. 2, 331-348.

Aleardi, M., F. Ciabarri, T. and Gukov, 2018, A two-step inversion approach for seismic-reservoir characterization and a comparison with a single-loop Markov-chain Monte Carlo algorithm: Geophysics, **83**, no. 3, R227-R244.

Aleardi, M., 2015, The importance of the Vp/Vs ratio in determining the error propagation, the stability and the resolution of linear AVA inversion: a theoretical demonstration: Bollettino di Geofisica Teorica ed Applicata, **56**, no. 3, 357-366.

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

Azevedo, L., and A. Soares, 2017, Geostatistical methods for reservoir geophysics: Springer.

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

Bosch, M., C. Carvajal, J. Rodrigues, A. Torres, M. Aldana, and J. Sierra, 2009, Petrophysical seismic inversion conditioned to well-log data: Methods and application to a gas reservoir: Geophysics, **74**, no. 2, O1-O15.

Buland, A., and H. Omre, 2003, Bayesian linearized AVO inversion: Geophysics, **68**, no. 1, 185-198.

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

De Figueiredo, L. P., D. Grana, F. L. Bordignon, M. Santos, M. Roisenberg, and B. B. Rodrigues, 2018, Joint Bayesian inversion based on rock-physics prior modeling for the estimation of spatially correlated reservoir properties. Geophysics, **83**, no. 5, M49-M61.

Dejtrakulwong, P., T. Mukerji, and G. Mavko, 2012, Using kernel principal component analysis to interpret seismic signatures of thin shaly-sand reservoirs: 82nd Annual International Meeting, SEG, Expanded Abstracts, 1-5.

Doyen, P. 2007, Seismic reservoir characterization: An earth modelling perspective. EAGE publications.

Fernández-Martínez, J. L., T. Mukerji, E. García-Gonzalo, and Z. Fernández-Muñiz, 2011, Uncertainty assessment for inverse problems in high dimensional spaces using particle swarm optimization and model reduction techniques: Mathematical and Computer Modelling, **54**, no. 11-12, 2889-2899.

Fernández-Martínez, J. L., Z. Fernández-Muñiz, J. L. G. Pallero, and S. Bonvalot, 2017, Linear geophysical inversion via the discrete cosine pseudo-inverse: application to potential fields: Geophysical Prospecting, **65**, 94-111.

Grana, D., and E. Della Rossa, 2010, Probabilistic petrophysical-properties estimation integrating statistical rock physics with seismic inversion: Geophysics, **75**, no, 3, O21-O37.

Grana, D., T. Fjeldstad, and H. Omre, 2017, Bayesian Gaussian mixture linear inversion for geophysical inverse problems: Mathematical Geosciences, **49**, no. 4, 493-515.

Grana, D., L. Passos de Figueiredo, and L. Azevedo, 2019, Uncertainty quantification in Bayesian inverse problems with model and data dimension reduction: Geophysics, **84**, no. 6, M15-M24.

Goodfellow, I., Y. Bengio, and A. Courville, 2016, Deep learning: Cambridge, MIT Press.

Hall, B., 2016, Facies classification using machine learning: The Leading Edge, **35**, no. 10, 906-909.

Hansen, T. M., and K. S. Cordua, 2017, Efficient Monte Carlo sampling of inverse problems using a neural network-based forward—applied to GPR crosshole traveltime inversion: Geophysical Journal International, **211**, no. 3, 1524-1533.

He, K., X. Zhang, S. Ren, and J. Sun, 2015, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification: In Proceedings of the IEEE international conference on computer vision, 1026-1034.

Horta, A., and A. Soares, 2010, Direct sequential co-simulation with joint probability distributions: Mathematical Geosciences, **42**, no. 3, 269-292.

Huot, F., B. Biondi, and G. Beroza, 2018, Jumpstarting neural network training for seismic problems: 88th Annual International Meeting, SEG, Expanded Abstracts, 2191–2195.

Kaur, H., N. Pham, and S. Fomel, 2019, Seismic data interpolation using CycleGAN: 89th Annual International Meeting, SEG, Expanded Abstracts, 2202-2206.

Kotsi, M., A. Malcolm, and G. Ely, 2020, Uncertainty quantification in time-lapse seismic imaging: a full-waveform approach: Geophysical Journal International, **222**, no. 2, 1245-1263.

Krizhevsky, A., L. Sutskever, and G. E. Hinton, 2012, Imagenet classification with deep convolutional neural networks: In Advances in neural information processing systems, 1097-1105.

Laloy, E., R. Hérault, D. Jacques, and N. Linde, 2018, Training-image based geostatistical inversion using a spatial generative adversarial neural network: Water Resources Research, **54**, no. 1, 381-406.

Le Ravalec, M., B. Noetinger, and L. Y. Hu, 2000, The FFT moving average (FFT-MA) generator: An efficient numerical method for generating and conditioning Gaussian simulations: Mathematical Geology, **32**, no. 6, 701-723.

Lewis, W., and D. Vigh, 2017, Deep learning prior models from seismic images for full-waveform inversion: 87th Annual International Meeting, SEG, Expanded Abstracts, 1512–1517.

Li, S., B. Liu, Y. Ren, Y. Chen, S. Yang, Y. Wang, and P. Jiang, 2020, Deep-learning inversion of seismic data: IEEE Transactions on Geoscience and Remote Sensing, **58**, no. 3, 2135–2149.

Liu, D., W. Wang, W. Chen, X. Wang, Y. Zhou, and Z. Shi, 2018, Random noise suppression in seismic data: What can deep learning do?: 88th Annual International Meeting, SEG, Expanded Abstracts, 2016-2020.

Lochbühler, T., S. J. Breen, R. L. Detwiler, J. A. Vrugt, and N. Linde, 2014, Probabilistic electrical resistivity tomography of a CO2 sequestration analog: Journal of Applied Geophysics, **107**, 80-92.

Ma, J., 2018, Deep learning for attenuating random and coherence noise simultaneously: 80th Conference and Exhibition, EAGE, Extended Abstracts, TuP7.

Malinverno, A., 2000, A Bayesian criterion for simplicity in inverse problem parametrization: Geophysical Journal International, **140**, no. 2, 267-285.

Mandelli, S., V. Lipari, P. Bestagini, and S. Tubaro, 2019, Interpolation and denoising of seismic data using convolutional neural networks: arXiv preprint: 1901.07927.

Masters, D., and C. Luschi, 2018, Revisiting small batch training for deep neural networks: arXiv preprint: 1804.07612.

Monajemi, H., D. L. Donoho, and V. Stodden, 2016, Making massive computational experiments painless: 2016 IEEE International Conference on Big Data, 2368–2373.

Pan, S. J., and Q. Yang, 2010, A survey on transfer learning: IEEE Transactions on Knowledge and Data Engineering, **22**, 1345–1359.

Park, M. J., and M. D. Sacchi, 2020, Automatic velocity analysis using Convolutional Neural Network and Transfer learning: Geophysics, **85**, no. 1, V33-V43.

Parzen, E., 1962, On estimation of a probability density function and mode: The annals of mathematical statistics, **33**, no. 3, 1065-1076.

Puzyrev, V., 2019, Deep learning electromagnetic inversion with convolutional neural networks: Geophysical Journal International, **218**, no. 2, 817-832.

Qin, H., X. Xie, and Y. Tang, 2019, Evaluation of a Straight-Ray Forward Model for Bayesian Inversion of Crosshole Ground Penetrating Radar Data: Electronics, **8**, no. 6, 630.

Raeesi, M., A. Moradzadeh, F. D. Ardejani, and M. Rahimi, 2012, Classification and identification of hydrocarbon reservoir lithofacies and their heterogeneity using seismic attributes, logs data and artificial neural networks: Journal of Petroleum Science and engineering, **82**, 151-165.

Rao, Y. R., N. Prathapani, and E. Nagabhooshanam, 2014, Application of normalized cross correlation to image registration: International Journal of Research in Engineering and Technology, **3**, no. 5, 12-16.

Richardson, A., and C. Feller, 2019, Seismic data denoising and deblending using deep learning. arXiv preprint: 1907.01497.

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

Rimstad, K., P. Avseth, and H. Omre, 2012, Hierarchical Bayesian lithology/fluid prediction: A North Sea case study: Geophysics, **77**, no. 2, B69-B85.

Sambridge, M., and K. Mosegaard, 2002, Monte Carlo methods in geophysical inverse problems: Reviews of Geophysics, **40**, no. 3, 3-1.

Santurkar, S., D. Tsipras, A. Ilyas, and A. Madry, 2018, How does batch normalization help optimization?: In Advances in Neural Information Processing Systems, 2483-2493.

Satija, A., and J. Caers, 2015, Direct forecasting of subsurface flow response from non-linear dynamic data by linear least-squares in canonical functional principal component space: Advances in Water Resources, **77**, 69-81.

Scherer, D., A. Müller, and S. Behnke, 2010, Evaluation of pooling operations in convolutional architectures for object recognition: International Conference on Artificial Neural Networks, Springer, **6354**, 92–101.

Schmidhuber, J., 2015, Deep learning in neural networks: An overview: Neural networks, **61**, 85-117.

Sen, M. K., and P. L. Stoffa, 2013, Global optimization methods in geophysical inversion: Cambridge University Press.

Sun, J., S. Slang, T. Elboth, T. Larsen Greiner, S. McDonald, and L. J. Gelius, 2020, A convolutional neural network approach to deblending seismic data: Geophysics, **85**, no. 4, WA13-WA26.

Szabó, N. P., and M. Dobróka, 2019, Series expansion-based genetic inversion of wireline logging data: Mathematical Geosciences, **51**, no. 6, 811-835.

Tarantola, A., 2005, Inverse problem theory and methods for model parameter estimation: siam.

Waldeland, A. U., A. C. Jensen, L. J. Gelius, and A. H. S. Solberg, 2018, Convolutional neural networks for automated seismic interpretation: The Leading Edge, **37**, no. 7, 529-537.

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

Wang, B., N. Zhang, W. Lu, and J. Wang, 2019, Deep-learning-based seismic data interpolation: A preliminary result: Geophysics, **84**, no. 1, V11-V20.

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

Xiong, W., X. Ji, Y. Ma, Y. Wang, N. M. AlBinHassan, M. N. Ali, and Y. Luo, 2018, Seismic fault detection with convolutional neural network: Geophysics, **83**, no. 5, O97-O103.

Yang, N., and Y. Duan, 2018, Human vision system-based structural similarity model for evaluating seismic image quality: Geophysics, **83**, no. 5, F49-F54.

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

Yosinski, J., J. Clune, Y. Bengio, and H. Lipson, 2014, How transferable are features in deep neural networks?: Advances in Neural Information Processing Systems, 3320–3328.

Yuan, S., J. Liu, S. Wang, T. Wang, and P. Shi, 2018, Seismic waveform classification and first-break picking using convolution neural networks: IEEE Geoscience and Remote Sensing Letters, **15**, no. 2, 272-276.

Yuan, C., X. Zhang, X. Jia, and J. Zhang, 2020, Time-lapse velocity imaging via deep learning: Geophysical Journal International, **220**, no. 2, 1228-1241.

Zhang, G., Z. Wang, and Y. Chen, 2018, Deep learning for seismic lithology prediction: Geophysical Journal International, **215**, no. 2, 1368-1387.

Zhong, Z., T. R. Carr, X. Wu, and G. Wang, 2019, Application of a convolutional neural network in permeability prediction: A case study in the Jacksonburg-Stringtown oil field, West Virginia, USA: Geophysics, **84**, no. 6, B363-B373.

Zujovic, J., T. N. Pappas, and D. L. Neuhoff, 2013, Structural texture similarity metrics for image analysis and retrieval: IEEE Transactions on Image Processing, **22**, no. 7, 2545-2558.

FIGURES AND CAPTIONS

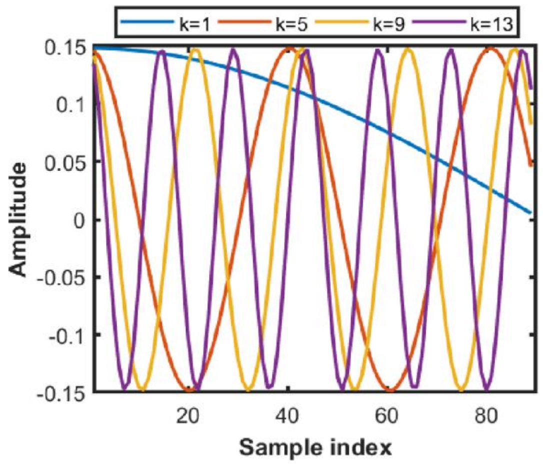


Figure 1: Some examples of 1-D DCT basis functions of different orders *k*.

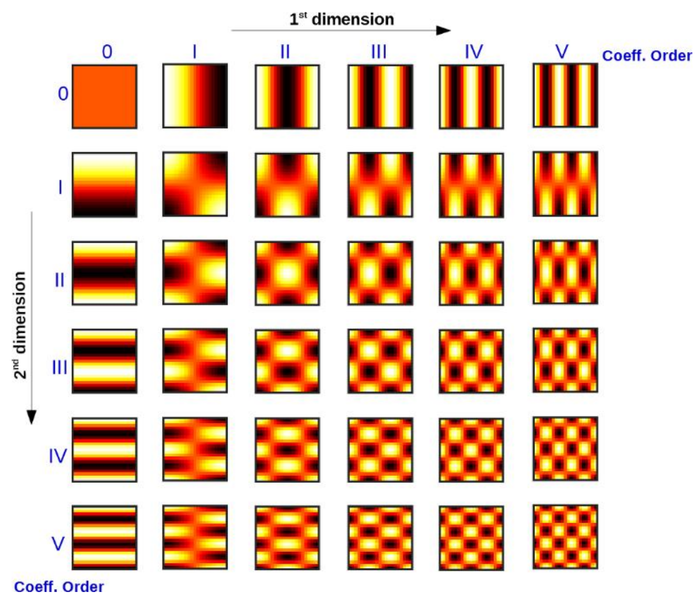


Figure 2: 2-D DCT basis functions of different orders. Dark and light colors code low and high numerical values, respectively.

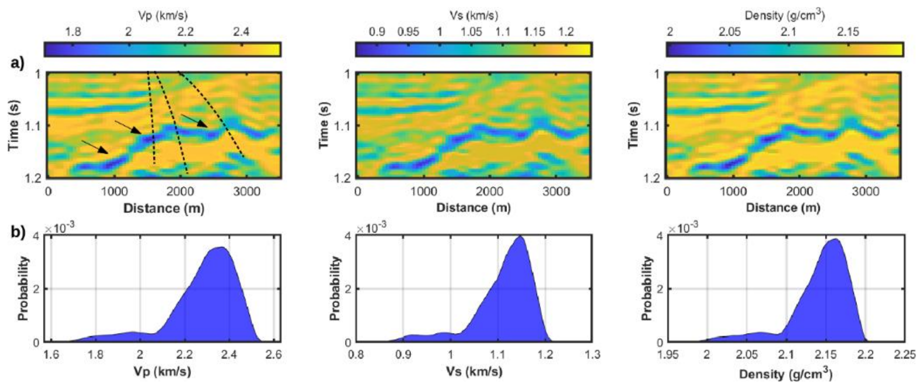


Figure 3: a) The elastic properties of *Vp*, *Vs,* and density of the reference model. In a) the black arrows point toward the main sand reservoir body, whereas the dotted black lines depict the trajectory of 3 out 5 wells that have used to build the model and to define the a-priori elastic distribution. b) The marginal non-parametric prior distributions for the three elastic properties derived from the five available wells.

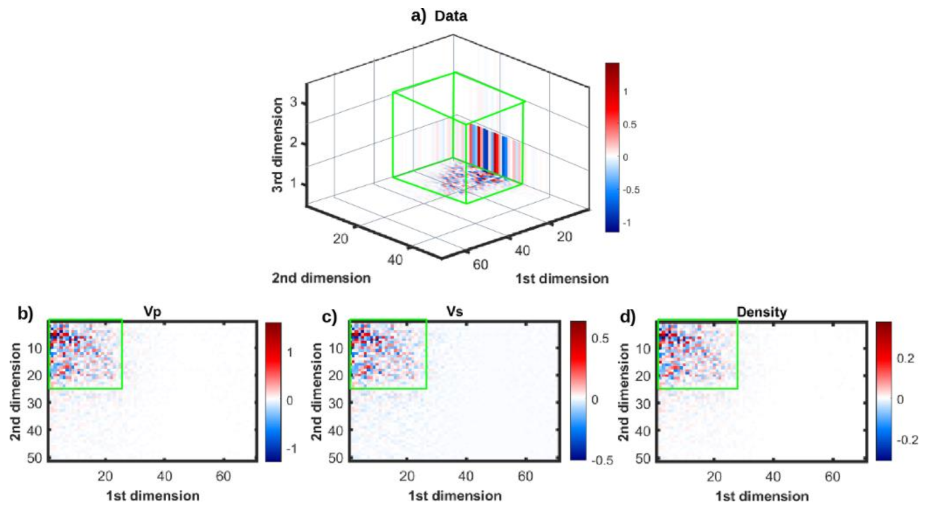


Figure 4: Some examples showing three sections from a DCT-transformed seismic data (a) and the associated DCT-transformed *Vp* (b), *Vs* (c), and density (d) models. The green boxes and green rectangles enclose the retained DCT coefficients in the data and model space, respectively (see the text for additional explanations). In a) the 1st, 2nd, and 3rd dimensions refer to the cross-line, time, and incidence angle directions, respectively. In b)-d) the 1st and 2nd dimensions refer to the cross-line and time axes, respectively.

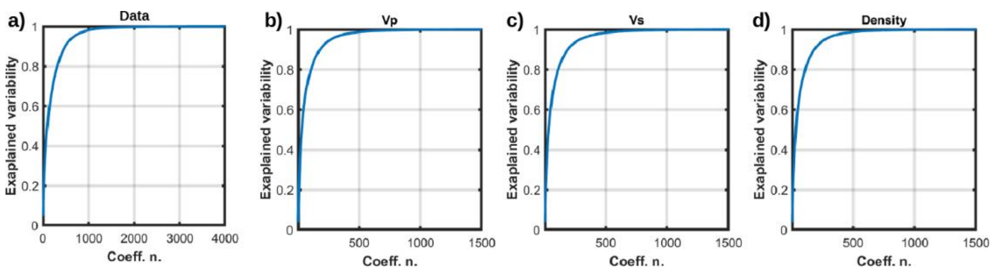


Figure 5: Close-ups showing the explained variability versus the number of DCT coefficients for a seismic dataset (a) and the associated *Vp* (b), *Vs* (c), and density (d) models extracted from the a-priori distribution.

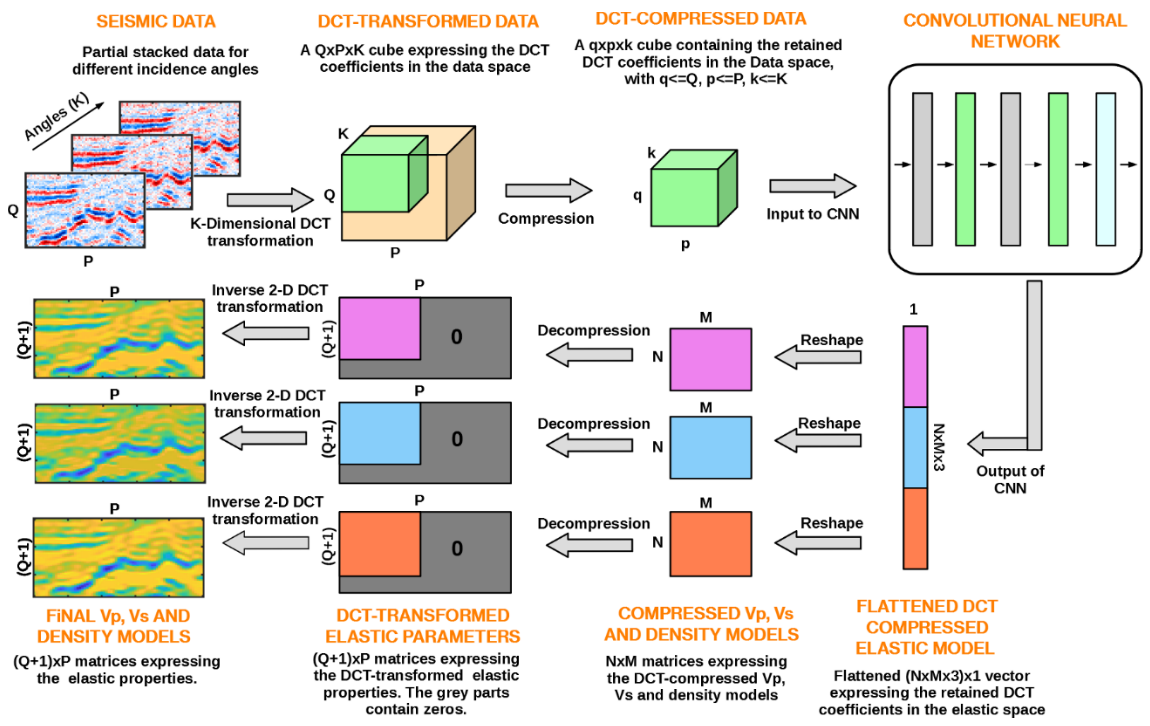
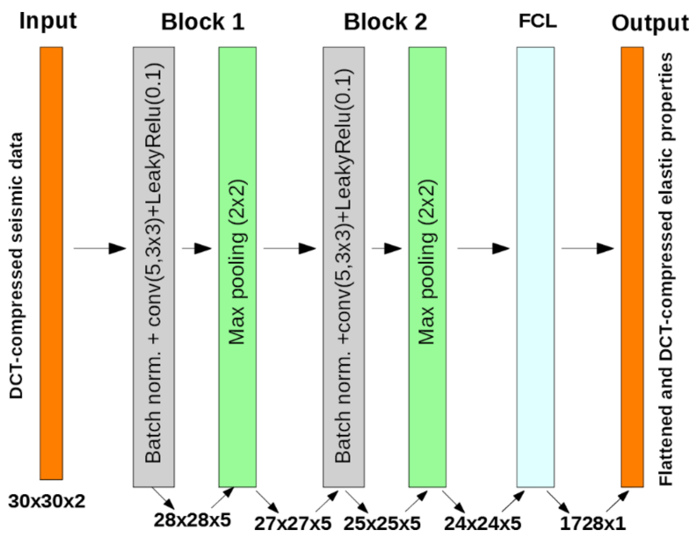

SEISMIC DATA
Partial stacked data for different incidence angles
Angles (K)
Q
P
K-Dimensional DCT transformation
DCT-TRANSFORMED DATA
A QxPxK cube expressing the DCT coefficients in the data space
K
Q
P
Compression
DCT-COMPRESSED DATA
A qxpxk cube containing the retained DCT coefficients in the Data space, with q<=Q, p<=P, k<=K
k
q
p
Input to CNN
CONVOLUTIONAL NEURAL NETWORK
Output of CNN
1
NxMx3
Reshape
Reshape
Reshape
M
N
Decompression
Decompression
Decompression
P
(Q+1)
0
Inverse 2-D DCT transformation
Inverse 2-D DCT transformation
Inverse 2-D DCT transformation
FiNAL Vp, Vs AND DENSITY MODELS
(Q+1)xP matrices expressing the elastic properties.
DCT-TRANSFORMED ELASTIC PARAMETERS
(Q+1)xP matrices expressing the DCT-transformed elastic properties. The grey parts contain zeros.
COMPRESSED Vp, Vs AND DENSITY MODELS
NxM matrices expressing the DCT-compressed Vp, Vs and density models
FLATTENED DCT COMPRESSED ELASTIC MODEL
Flattened (NxMx3)x1 vector expressing the retained DCT coefficients in the elastic space

Figure 6: Workflow of the implemented CNN inversion approach. The Monte Carlo method for uncertainty propagation is not represented in this figure. The CNN architecture is detailed in the next Figure 7.



Input
Block 1
Block 2
FCL
Output
DCT-compressed seismic data
Batch norm. + conv(5,3x3)+LeakyRelu(0.1)
Max pooling (2x2)
Batch norm. +conv(5,3x3)+LeakyRelu(0.1)
Max pooling (2x2)
Flattened and DCT-compressed elastic properties
30x30x2
28x28x5
27x27x5
25x25x5
24x24x5
1728x1

Figure 7: Schematic representation of the adopted CNN architecture. The image on the far left is the input of the network that is a 3-D cube expressing the 1800 DCT coefficients used for data compression. The image on the far right is the output of the network that is a vector containing the 1728 DCT coefficients expressing the elastic properties (576 coefficient for each elastic property). The intermediate rectangles represent convolutional layers and are annotated with key parameters. In the grey rectangles the initial value in brackets (e.g. 5) indicates the number of filters. This is followed by the filter size (i.e. 3 × 3). In the green rectangles it is also indicated the dimension of the max-pooling filter (2×2). The cyan rectangle represents the fully connected layer (FCL). The number on the bottom of each rectangle indicates the dimension of the input and output to each layer.

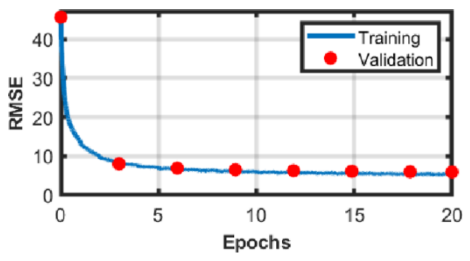


Figure 8: Evolution of the RMSE error on the training and validation sets during the learning process.

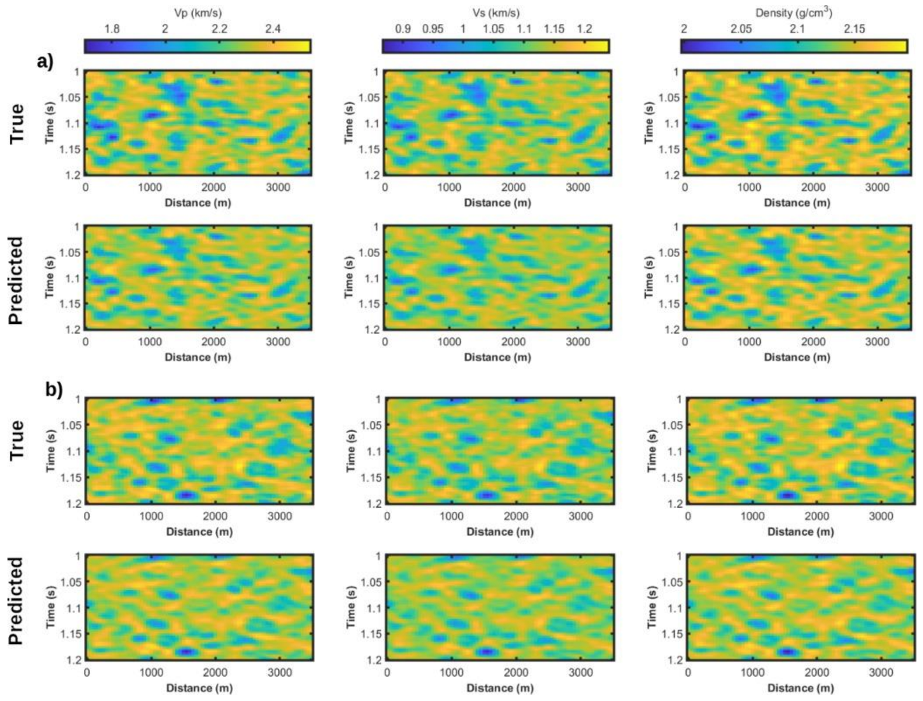


Figure 9: a), and b) show a comparison between two elastic models extracted from the validation set and the corresponding CNN prediction. In a) and b) from left to right we represent *Vp*, *Vs*, and density.

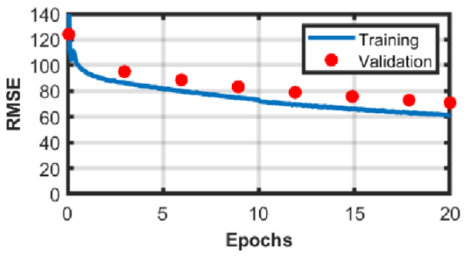


Figure 10: Evolution of the RMSE error for a CNN without DCT compression.

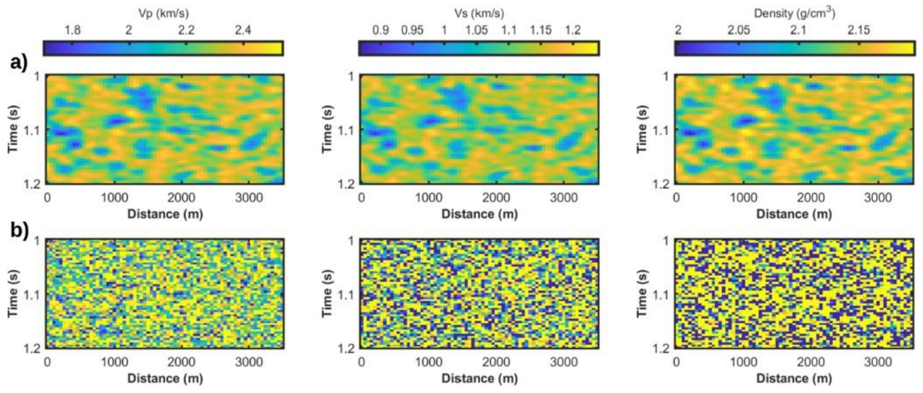


Figure 11: a) An elastic model extracted from the validation set (also shown in Figure 9a). b) The elastic model predicted by a trained CNN with the same architecture previously described (Figure 7) but in this case the DCT has not been used to compress the input and output responses.

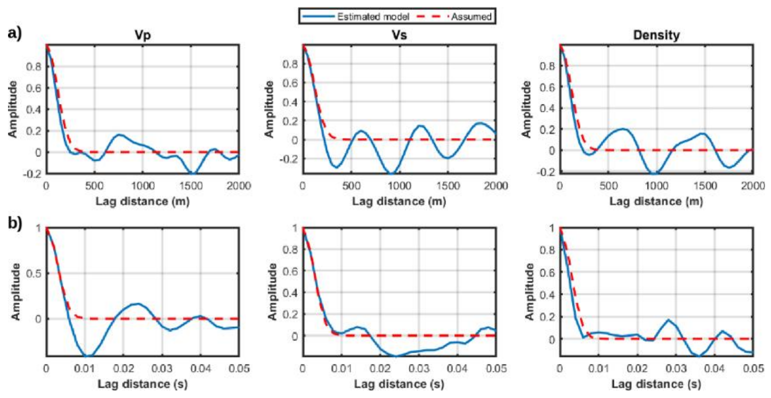


Figure 12: Examples of comparison between the normalized lateral (a) and temporal (b) correlation functions used to generate the training examples (dashed red lines) and those computed on the CNN estimated model (continuous blue lines).

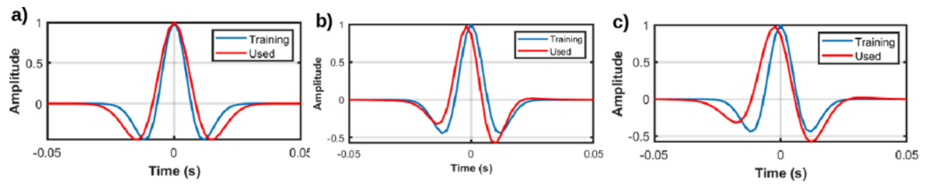


Figure 13: a) Comparison between the source wavelet used to generate the training examples and that used to compute the observed data in Test 4. b) As in a) but for Test 5. c) As in a) but for Tests 6 and 7.

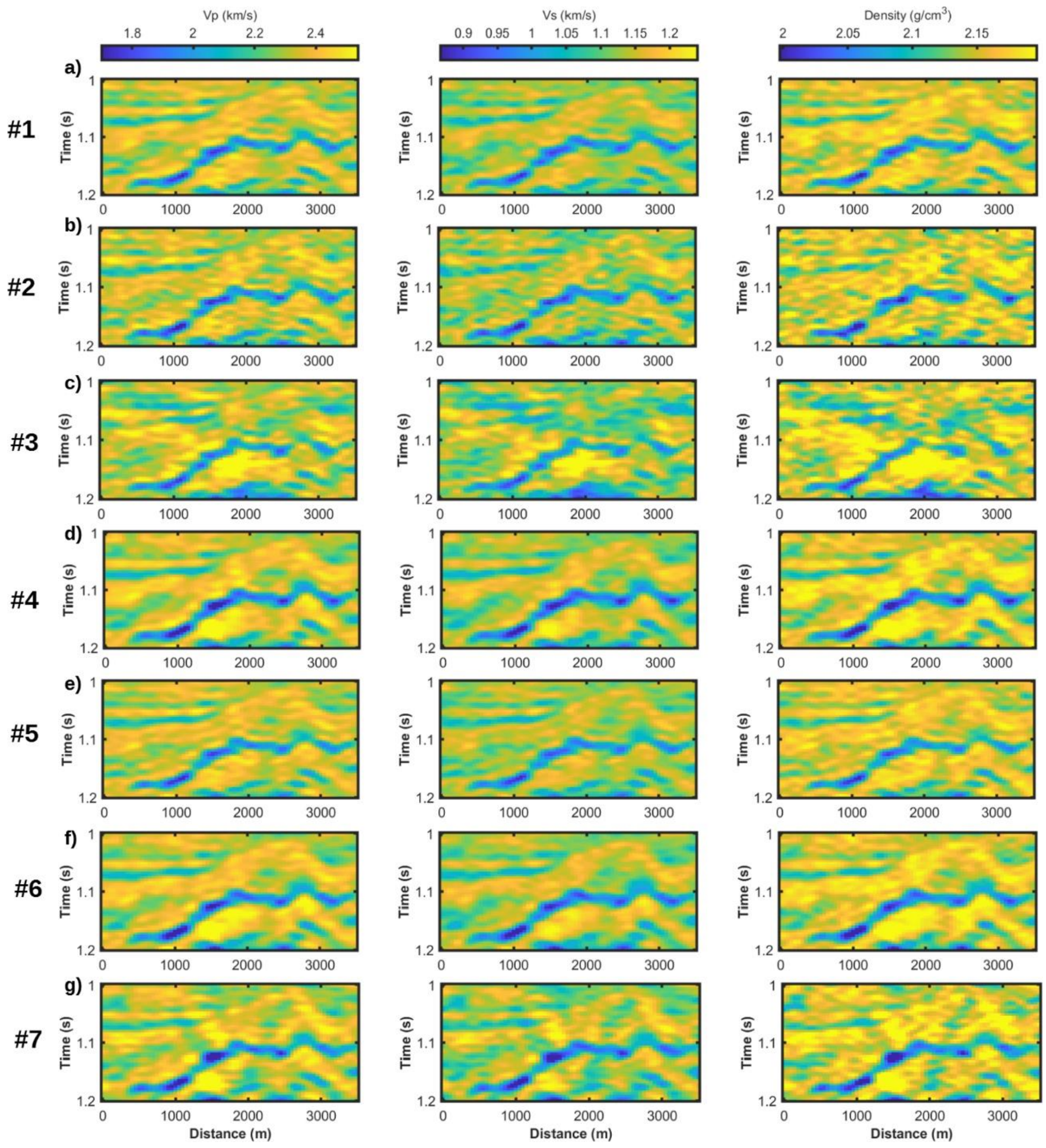


Figure 14: *Vp* (left), *Vs* (center), and density (right) models estimated by the CNN approach in the seven inversion tests. The true model is shown in Figure 3a.

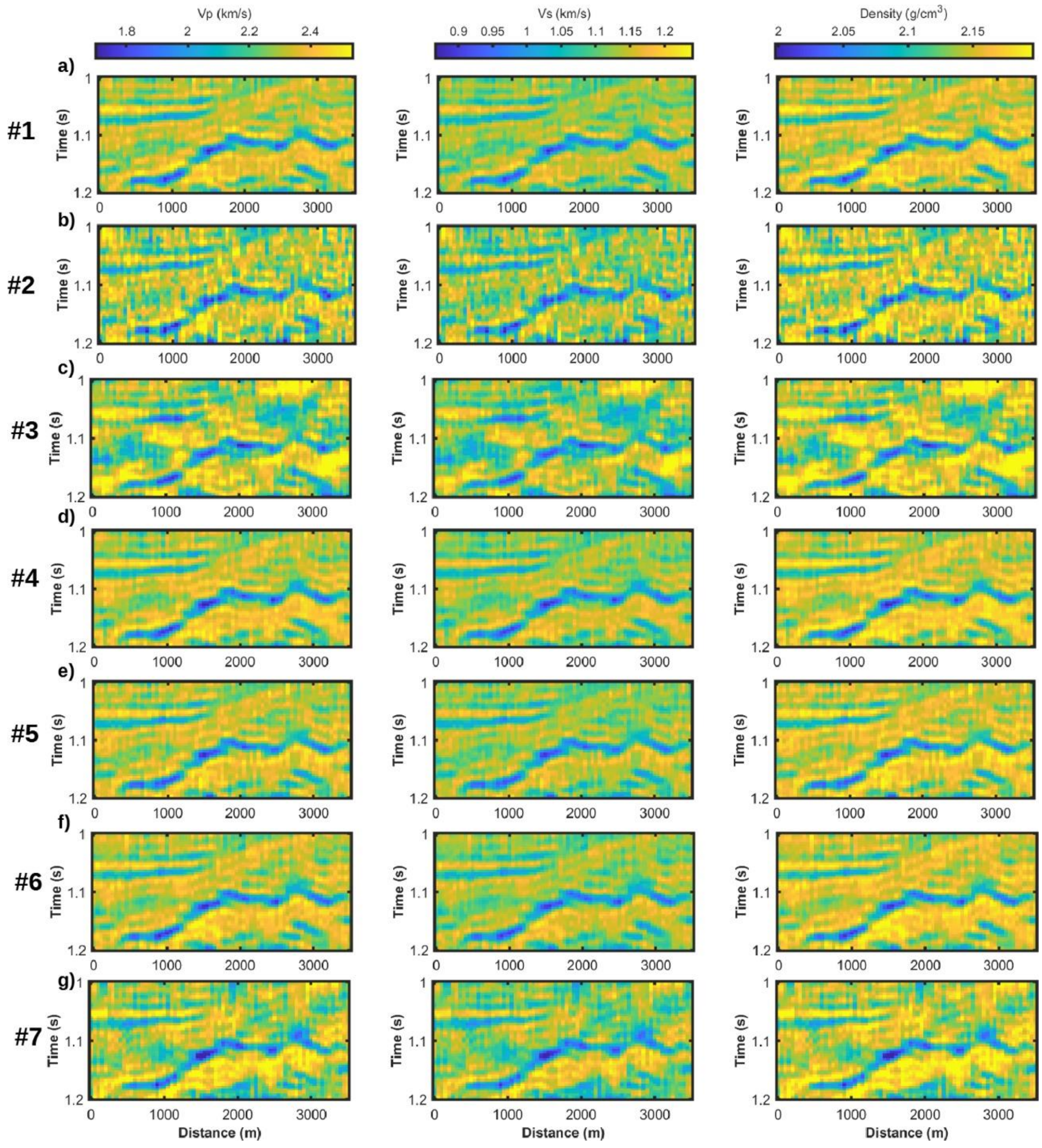


Figure 15: *Vp* (left), *Vs* (center), and density (right) estimated by the linear inversion for the seven tests. The true model is shown in Figure 3a.

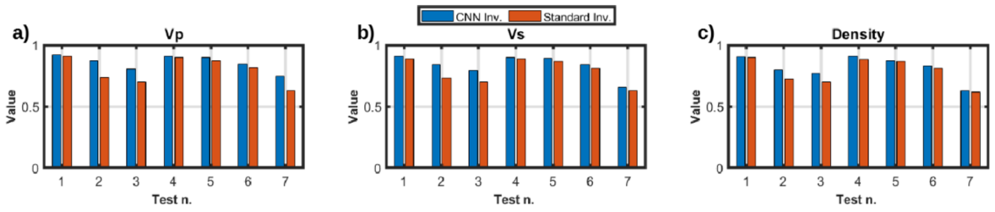


Figure 16: Comparison of normalized correlation coefficients associated with the CNN and SA results for the seven inversion tests.

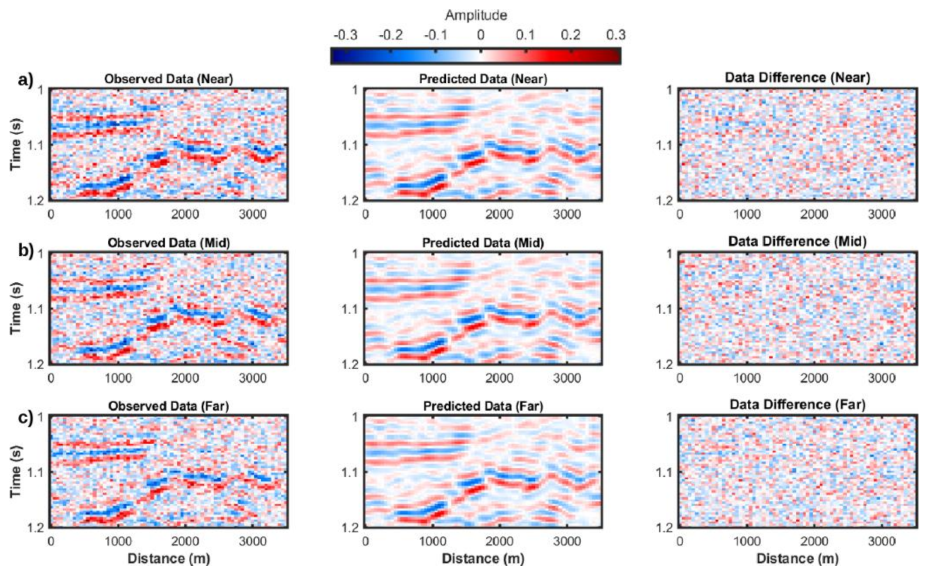


Figure 17: Comparison between observed data (left column), predicted data (central column), and their sample-by-sample difference (right column) in Test 2 and for the CNN inversion.

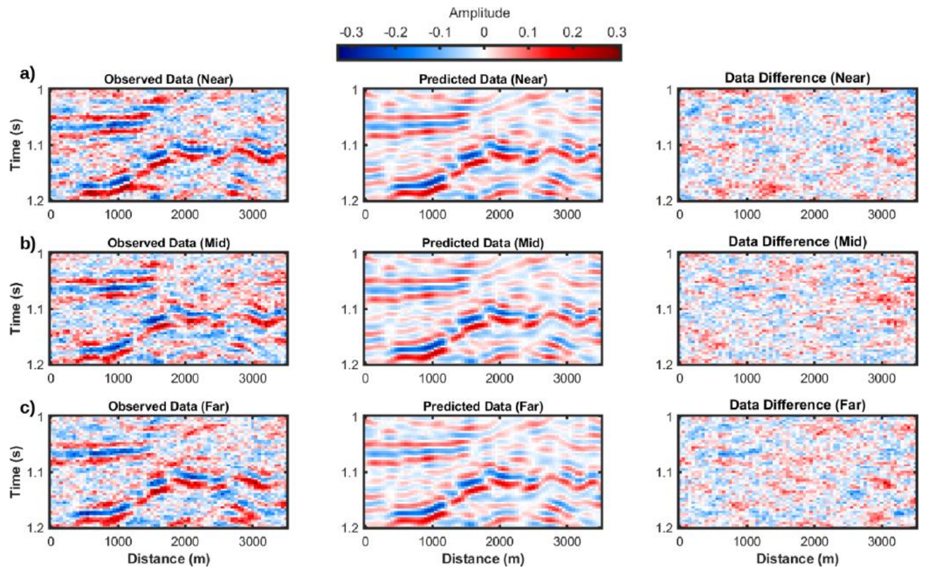


Figure 18: As in Figure 17 but for Test 3.

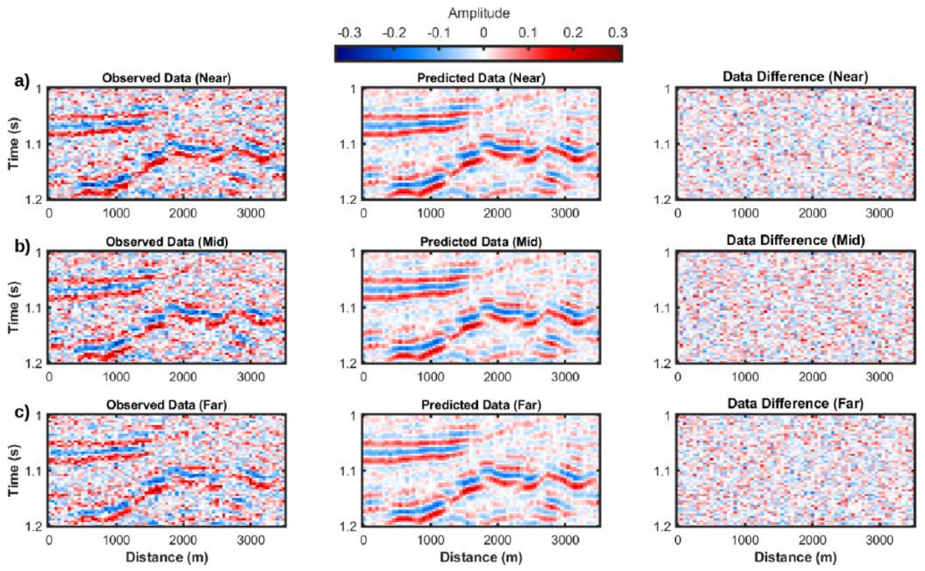


Figure 19: Comparison between observed data (left column), predicted data (central column), and their sample-by-sample difference (right column) in Test 2 and for the SA inversion.

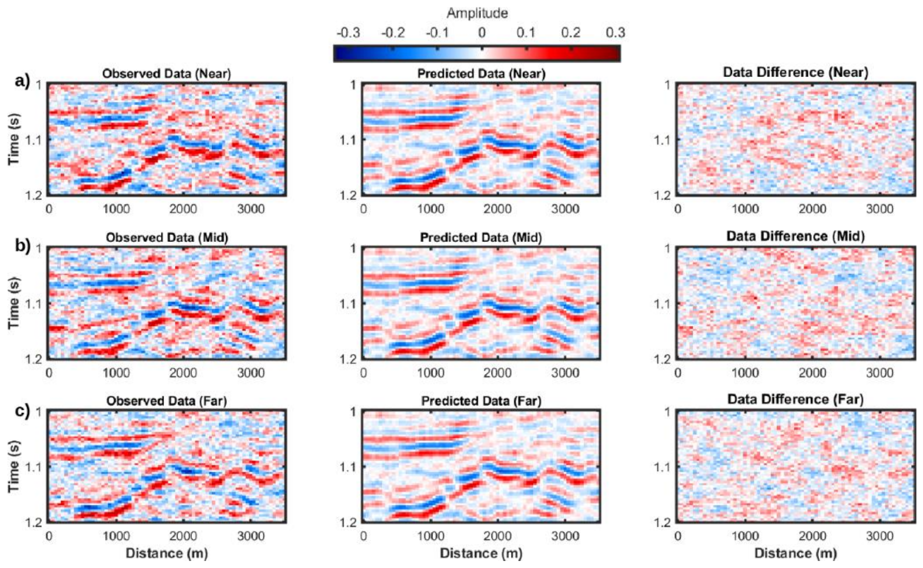


Figure 20: As in Figure 19 but for Test 3.

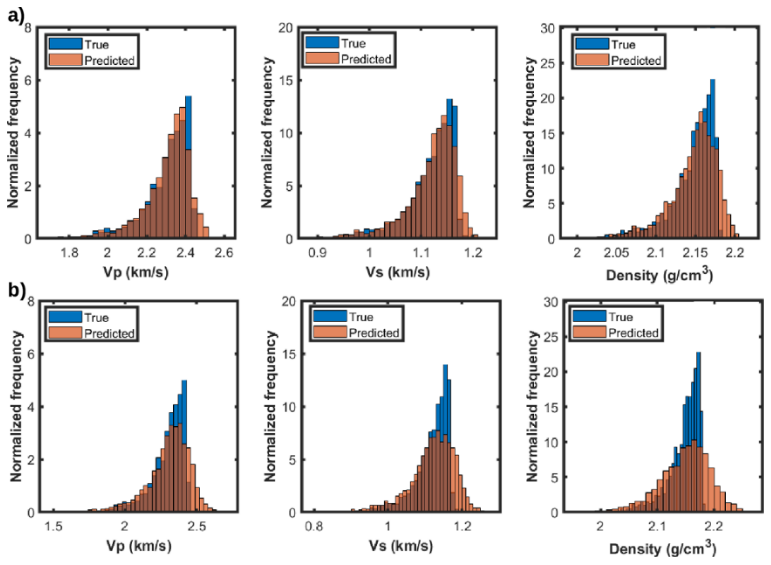


Figure 21: Comparison between the marginal distributions computed on the true *Vp*, *Vs* and density models, and those associated with the CNN predictions. a) for Test 1. b) for Test 2.

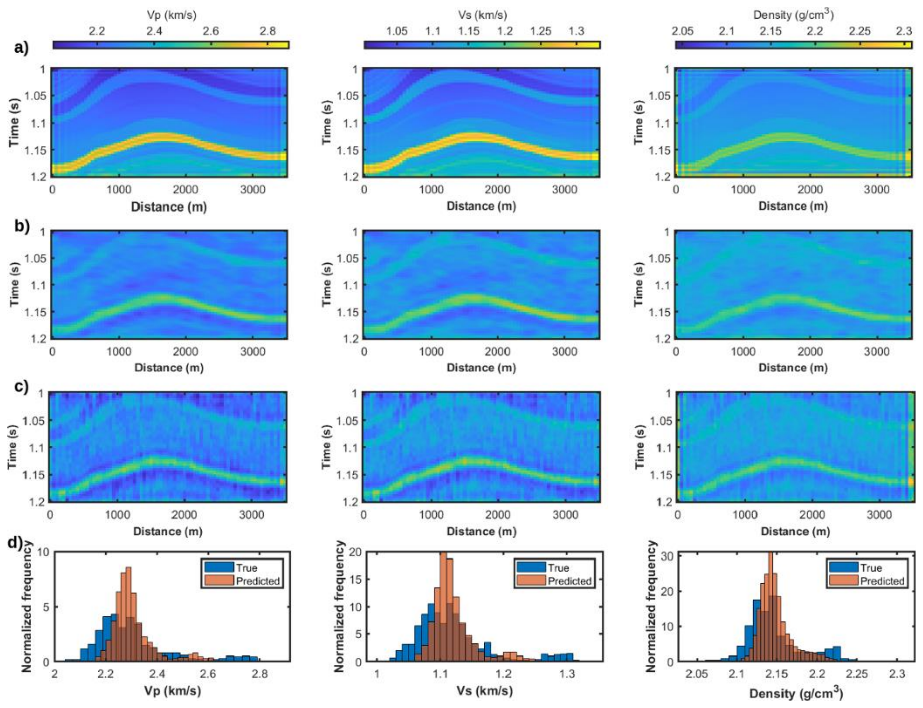


Figure 22: a) Portion of the elastic Marmousi model interpolated and converted from depth to time and used as the reference model in this inversion test. b) Elastic properties predicted by the previously trained CNN and assuming a correct estimation of the source wavelet and noise. c) SA results. d) Comparison between the marginal distributions computed on the true and the CNN-predicted models for each elastic parameter. In c) note that the true parameter distributions in this example significantly differ from that used to generate the training examples (see Figure 3b).

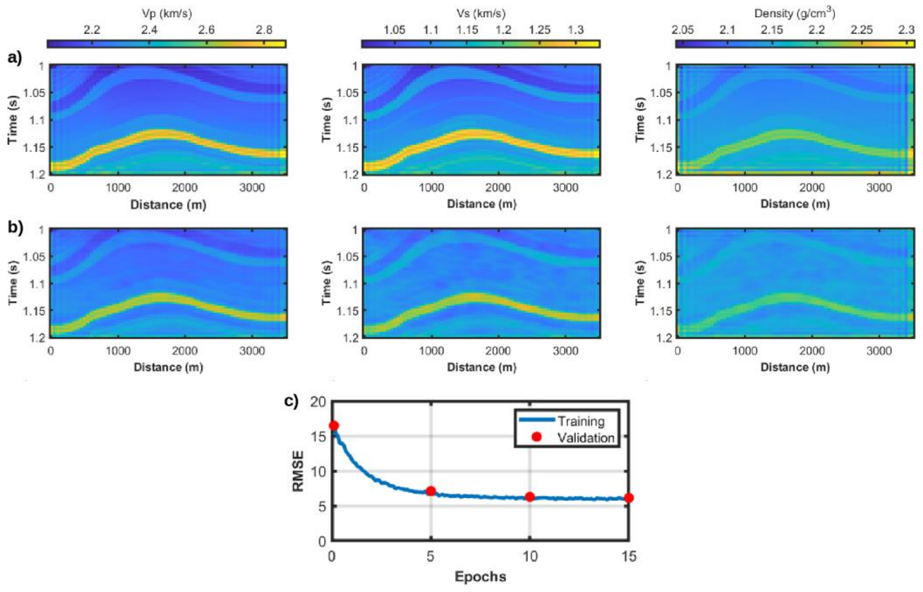


Figure 23: a) Portion of the elastic Marmousi model used as the reference model. b) Elastic properties predicted by the CNN after transfer learning and assuming a correct estimation of the source wavelet and noise. c) Evolution of the RMSE error during the transfer learning.

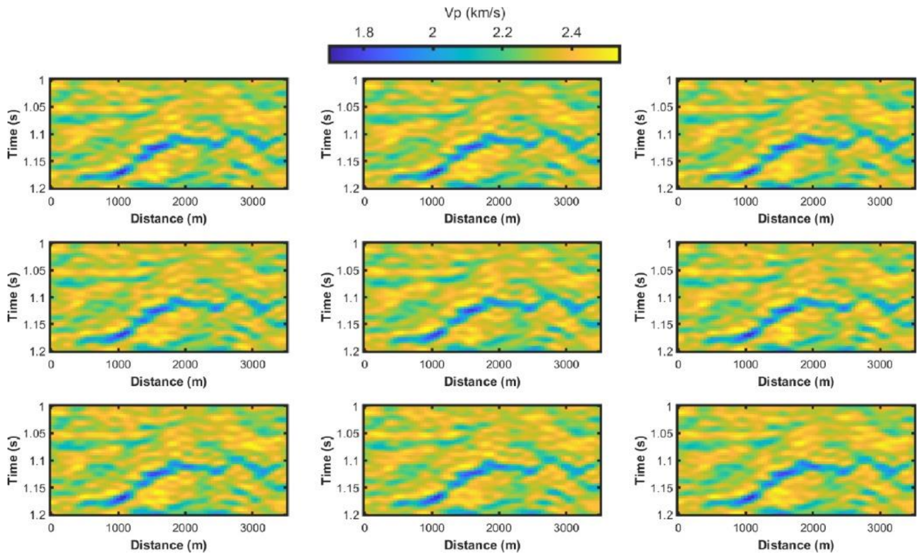


Figure 24: Nine MC realizations of *Vp* models associated with Test 1.

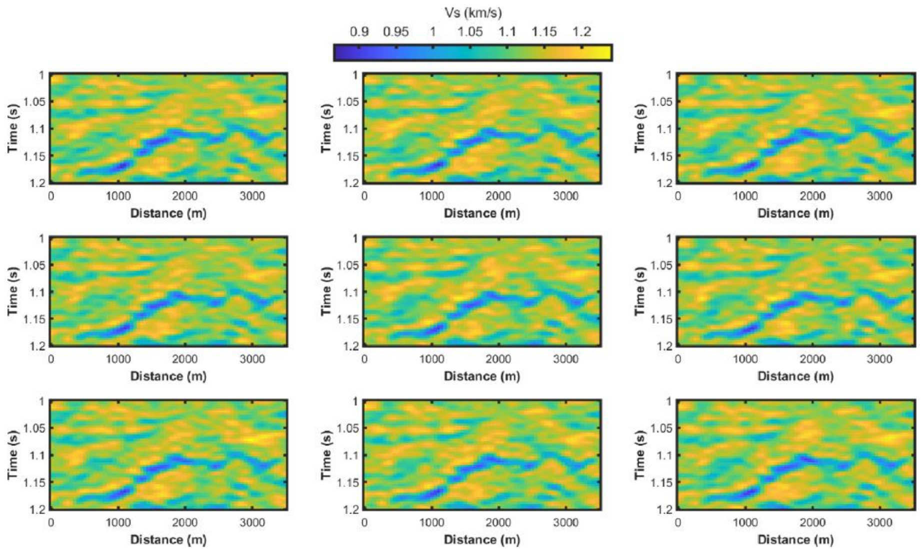


Figure 25: Nine MC realizations of *Vs* models associated with Test 1.

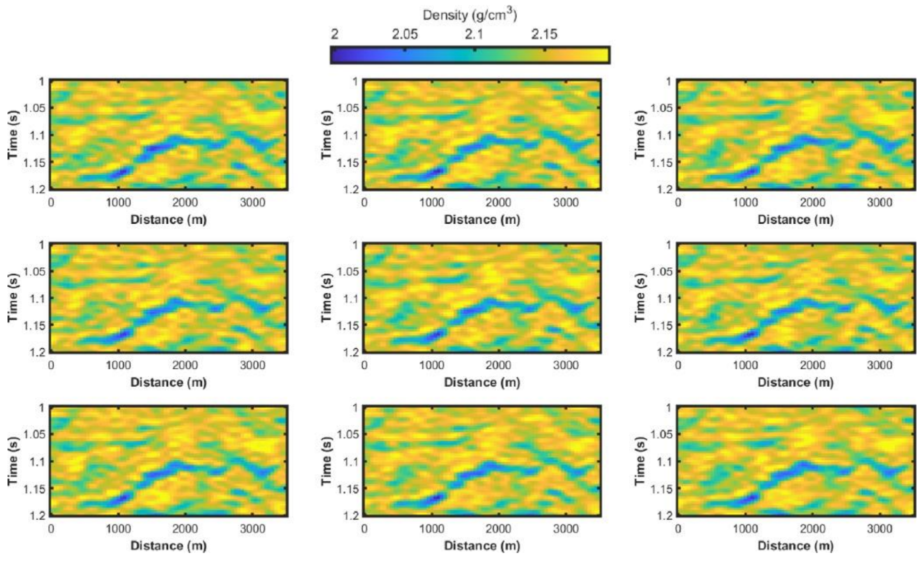


Figure 26: Nine MC realizations of density models associated with Test 1.

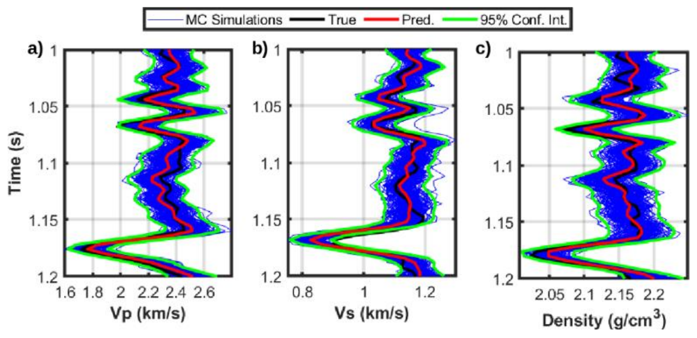


Figure 27: For the spatial location equal to 1000 m and for Test 1 we represent some comparisons between the true properties, the predicted properties, some MC simulations, and the 95 % confidence interval as estimated from the MC realizations. a) *Vp*, b) *Vs*, c) Density.

TABLES

Seismic data generation		**Elastic model generation**	
Uncorrelated noise standard deviation	0.02	*Type of prior distribution*	Non-parametric
Correlated noise standard deviation	0	*Vertical range of the Gaussian variogram model*	0.008 s
Source wavelet phase	0	*Lateral range of the Gaussian variogram model*	160 m
Source wavelet peak frequency	30		

Table 1: Principal characteristics of the observed seismic data and of the elastic prior model that has been used to generate the training and validation datasets.

Parameters	**Training**	**#1**	**#2**	**#3**	**#4**	**#5**	**#6**	**#7**
Uncorrelated noise standard deviation	0.02	0.02	0.05	0.03	0.02	0.02	0.02	0.03
Correlated noise standard deviation	0	0	0	0.03	0	0	0	0.03
Source wavelet phase	0°	0°	0°	0°	0°	30°	30°	30°
Source wavelet peak frequency	30 Hz	30 Hz	30 Hz	30 Hz	25 Hz	30 Hz	25 Hz	25 Hz
Vertical range of correlated noise	/	/	/	12 ms	/	/	/	12 ms
Lateral range of correlated noise	/	/	/	150 m	/	/	/	150 m

Table 2: Characteristics of the noise distribution and assumed source wavelet in the training datasets and in the seven different inversion experiments.