GEOPHYSICS, VOL. 85, NO. 1 (JANUARY-FEBRUARY 2020); P. U21–U29, 6 FIGS.
10.1190/GEO2018-0786.1

Seismic velocity estimation: A deep recurrent neural-network approach

Gabriel Fabien-Ouellet[1] and Rahul Sarkar[2]

ABSTRACT

Applying deep learning to 3D velocity model building remains a challenge due to the sheer volume of data required to train large-scale artificial neural networks. Moreover, little is known about what types of network architectures are appropriate for such a complex task. To ease the development of a deep-learning approach for seismic velocity estimation, we have evaluated a simplified surrogate problem — the estimation of the root-mean-square (rms) and interval velocity in time from common-midpoint gathers — for 1D layered velocity models. We have developed a deep neural network, whose design was inspired by the information flow found in semblance analysis. The network replaces semblance estimation by a representation built with a deep convolutional neural network, and then it performs velocity estimation automatically with recurrent neural networks. The network is trained with synthetic data to identify primary reflection events, rms velocity, and interval velocity. For a synthetic test set containing 1D layered models, we find that rms and interval velocity are accurately estimated, with an error of less than 44 m/s for the rms velocity. We apply the neural network to a real 2D marine survey and obtain accurate rms velocity predictions leading to a coherent stacked section, in addition to an estimation of the interval velocity that reproduces the main structures in the stacked section. Our results provide strong evidence that neural networks can estimate velocity from seismic data and that good performance can be achieved on real data even if the training is based on synthetics. The findings for the 1D problem suggest that deep convolutional encoders and recurrent neural networks are promising components of more complex networks that can perform 2D and 3D velocity model building.

INTRODUCTION

In recent years, deep-learning algorithms have become increasingly better at solving many real-world problems to the extent of being competitive and in some cases exceeding human-level performance. The application of deep learning to seismic imaging problems is getting more and more attention. So far, most work has focused on seismic interpretation or event detection. For example, Di et al. (2017) and Di and AlRegib (2017) use cluster analysis and classification techniques for salt-boundary detection, Alaudah et al. (2017) use machine-learning techniques for seismic structure labeling, and Lu et al. (2018) use a generative adversarial network to perform fault detection. A notable application to full-waveform inversion is by Lewis et al. (2017), who propose to use deep learning as a way to interpret migrated images and build prior velocity models for full-waveform inversion. The literature on the application of deep learning for seismic interpretation is already vast and expanding at a very fast pace.

Although the application of machine learning to seismic inversion is not new (Roth and Tarantola, 1994; Langer et al., 1996; Baronian et al., 2009), there exist very few studies on automating seismic inversion and processing with deep learning. These early studies were promising, but the use of shallow fully connected artificial neural networks (NNs) limited the number of parameters that could be estimated and did not scale to the size of real seismic data. Deep learning allows the application of NNs to much more complex tasks, with very large input and output spaces. Araya-Polo et al. (2018) show the potential of deep learning for seismic tomography — they directly predict gridded velocity models with deep NNs using semblance as input. Data-driven seismic inversion and imaging based on deep learning thus now seems a plausible scenario.

Building a general and efficient deep NN for robust seismic inversion is still challenging. One difficulty is the large quantity of data required for inversion, which leads to long training times, making the process of testing different architectures costly. In this

Manuscript received by the Editor 30 November 2018; revised manuscript received 26 September 2019; published ahead of production 06 November 2019; published online 19 December 2019.
[1]Polytechnique Montréal, 2900 Boulevard Edouard-Montpetit, Montréal, Quebec H3T 1J4, Canada. E-mail: gabriel.fabien-ouellet@polymtl.ca.
[2]Stanford University, 450 Serra Mall, Stanford, California 94305, USA. E-mail: rahul@sep.stanford.edu.

paper, we propose a benchmark problem that is simpler to solve, bearing enough resemblance to seismic inversion — so its study should bring relevant insights. The problem in question is the estimation of the root-mean-square (rms) and interval velocities in time for flat-layered velocity models. This problem is typically solved by semblance analysis, which is routinely used to obtain coherent stacked sections. In contrast, we aim to estimate the rms velocity directly from the seismic gathers, without computing semblance. Indeed, semblance is a lossy, non-invertible transform that removes the amplitude and phase information relevant for seismic inversion. This is why modern seismic inversion procedures, such as full waveform inversion (FWI), rely instead on the full recorded waveform, and why the full waveform should be the input to a NN-based approach.

In this study, we address two questions by studying the 1D velocity estimation problem. First, we want to know what kind of network architecture is suitable to estimate velocities directly from the seismic waveforms. Second, we want to determine if a NN can be trained on synthetic data and still perform well with real data. We will first discuss the problem statement and its link to semblance analysis. Then, we will present our network architecture and our training approach. Finally, we will evaluate the performance of the NN on different test data sets: (1) synthetics computed from 1D velocity models, (2) synthetics computed from gently dipping 2D velocity models, and (3) a real marine seismic 2D line. For specific details about the implementation or to reproduce the results, see our GitHub repository (Fabien-Ouellet and Sarkar, 2019).

METHODS

Problem statement

The problem we propose to solve is the problem of estimating the rms velocity in time when the earth is represented as a series of flat constant-velocity layers. This is the basic assumption behind the traditional velocity analysis based on semblance. In a sense, we can formulate our problem as automating semblance analysis with a NN.

For semblance analysis, we make the assumption of flat layers, which imply that a reflection will appear to follow a hyperbola on a common-midpoint (CMP) gather at short offsets. The hyperbola is given by the normal moveout (NMO) equation:

$$t^2 = t_0^2 + \frac{x^2}{v_{\mathrm{rms}}^2}, \tag{1}$$

where t is the two-way traveltime at offset x, $t_0 = 2z/v_{\mathrm{rms}}$ is the zero-offset two-way traveltime, z is the depth to the reflector, and v_{rms} is the rms velocity defined by

$$v_{\mathrm{rms}} = \sqrt{\frac{\sum_{i=1}^{N} v_i^2 \Delta t_i}{\sum_{i=1}^{N} \Delta t_i}}, \tag{2}$$

where v_i is the interval velocity and Δt_i is the one-way traveltime in the ith layer. The NMO correction consists of removing the dependency of the reflection traveltimes on offset, given by

$$d^{\mathrm{NMO}}(t, x) = d\left(\sqrt{t^2 + x^2/v_{\mathrm{rms}}^2}, x\right), \tag{3}$$

where d is the original CMP gather and d^{NMO} is the NMO-corrected CMP gather. After NMO correction with the correct v_{rms}, reflection events should ideally form a straight line. The right velocity can thus be found by searching for the velocity that aligns an event after NMO correction. This alignment can be measured by semblance defined as

$$S_t = \frac{\sum_{i=t-l}^{i=t+l}\left(\sum_{j=1}^{j=N_x} d_{ij}^{\mathrm{NMO}}\right)^2}{\sum_{i=t-l}^{i=t+l}\sum_{j=1}^{j=N_x}(d_{ij}^{\mathrm{NMO}})^2}, \tag{4}$$

for N_x traces, and a window length l, where d_{ij}^{NMO} now refers to the discretely sampled version of $d^{\mathrm{NMO}}(t, x)$.

Semblance-based processing builds the rms velocity profile by computing the semblance of a CMP gather on a range of test velocities. Viewing a CMP gather $d(t, x)$ as a tensor of size $N_t \times N_x \times 1$, with N_t time samples and N_x uniformly spaced traces, semblance analysis is a tensor transformation comprising of the following steps:

1) Representation building: Compute the NMO-corrected gather (equation 3) for each of the N_v test velocities, resulting in a tensor of size $N_t \times N_x \times N_v$.
2) Data reduction: Compute the semblance of the NMO-corrected gather (equation 4) for each test velocity and time sample, resulting in a tensor of size $N_t \times 1 \times N_v$.
3) Velocity decoding: Build the rms velocity profile from the semblance panel by picking the maximum at each reflection event, resulting in a tensor of size $N_t \times 1 \times 1$.

Two by-products also result from the semblance-analysis workflow — the identification of primary reflections and the possibility of estimating the interval velocity from the rms velocity. We thus formulate the following problem. Given as input a CMP gather, output:

1) the arrival time of the primary reflections
2) the rms velocity profile in time
3) the interval velocity profile in time.

Neural-network-based analysis

To perform this task, we need to design the architecture of our NN. To guide this design, we tried to incorporate the important physical intuitions that are contained in the traditional semblance-analysis process. In what follows, we will discuss the architecture of the NN, explaining the links with semblance analysis along the way. A schematic representation of the proposed NN can be found in Figure 1.

Step 1: Representation building

The input of semblance analysis is a CMP gather, a tensor of size $N_t \times N_x \times 1$, which will also be the input to our NN. In semblance-based velocity analysis, the semblance is used as the feature on which the velocity is chosen. On the other hand, NNs can build powerful representations of data and identify features commonly disregarded by the semblance. As deep convolutional neural networks (CNNs) have been found to be particularly efficient for automatic feature detection, for example in image analysis, we decided to build a representation of an input CMP gather through several layers of CNNs with rectified linear unit (RELU) activation functions. Mathematically, we can express a CNN layer as

$$y_{txj} = \max\left(0, \sum_{i=1}^{C_i}\sum_{p=1}^{L_t}\sum_{q=1}^{L_x} \mathbf{W}_{pqij} x_{t+p,x+q,i} + b_j\right), \qquad (5)$$

where x is the layer input, a tensor of size $N_t \times N_x \times C_i$, y is the layer output, a tensor of size $N_t \times N_x \times C_o$, $\mathbf{W}$ is a weight matrix (or filter) of size $L_t \times L_x \times C_i \times C_o$, and b is a tensor of size C_o. The RELU activation function takes the form of the max function, zeroing the convolution output if it is smaller than zero. The weight and the bias are the parameters that are optimized during training.

The encoder part of the NN found in Figure 1 is made up of four consecutive CNN layers, with filter shapes $15 \times 1 \times 1 \times 16$, $1 \times 9 \times 16 \times 16$, $15 \times 1 \times 16 \times 32$, and $1 \times 9 \times 32 \times 32$. The first CNNs gradually increase the number of feature maps (or channels) from 1 to 32. The encoder is completed by recursively applying seven times the same CNN filter of shape $15 \times 3 \times 32 \times 32$. This last step aims to increase the receptive field in time of the NN, as reflection hyperbolas can spread over a vast number of time samples across traces. The output from the encoder is a tensor of dimensions $N_t \times N_x \times 32$ and represents a set of automatically learned features from the data. Thus, the encoder is analogous to step 1 of the semblance-analysis workflow — 32 feature maps are created by the NN, whereas multiple NMO-corrected gathers (one per trial velocity) are created during semblance analysis.

Step 2: Data reduction

Step 2 of semblance analysis involves measuring the semblance of each NMO-corrected gather, at each time sample. In other words, we reduce the data size from $N_t \times N_x \times N_v$ to $N_t \times 1 \times N_v$. We mimic this data-reduction procedure in our NN with a recursive convolutional layer (RCNN), which has been reported to be quite successful for sequential data analysis (Socher et al., 2010; Kim et al., 2016). The RCNN is formed by applying recursively the same CNN filter of shape $1 \times 2 \times 32 \times 32$, with a stride of 2, that is, skipping every other element in the output. As a consequence, the offset dimension is reduced twofold after each convolution pass. The CNN is applied until the offset dimension is reduced to 1, giving a final tensor with a shape of $N_t \times 1 \times 32$. Note that this data-reduction strategy has the advantage of being applicable to a different number of offsets, in contrast to a matrix multiplication of fixed input and output sizes, as performed by fully connected layers.

Step 3: Velocity decoding

The final step of semblance analysis is picking the maxima to build the velocity profile in time; that is, we want to reduce the $N_t \times 1 \times N_v$ tensor to $N_t \times 1 \times 1$. This decoding phase implies that the interpreter has to choose the maxima corresponding to primary reflections and that the rms velocity profile should be realistic, so that the corresponding interval velocity profiles obey reasonable bounds (e.g., it should be positive). In the same fashion, the NN decodes the input from the RCNN into three different outputs — primary reflection detection, rms velocity estimation, and interval velocity estimation.

To detect primary reflections, we define a binary classification problem, with classes defined as

$$R_t = \begin{cases} 1 & \text{if } t_0^i - l < t < t_0^i + l, \text{ for } i \in P, \\ 0 & \text{otherwise,} \end{cases} \qquad (6)$$

where P is the set of primary reflections and l is the window width. To perform the classification, we apply a final CNN layer with a shape of $1 \times 1 \times 32 \times 2$ to the output of the RCNN. This results in a tensor of dimensions $N_t \times 1 \times 2$. The last dimension contains the classification scores for each class R_{pred}. Jointly building the capacity to differentiate primary reflections from multiples should make the NN robust to the presence of multiples, for the velocity-estimation process.

For predicting the rms and interval velocity profiles in time, we use recurrent NNs, or more precisely long short time memory (LSTM) units (Hochreiter and Schmidhuber, 1997; Gers et al., 2000). Recurrent NNs are commonly used for prediction tasks based on sequential data where the prediction for the current input depends on the history of inputs fed to the NN (Elman, 1990, 1991; Servan-Schreiber et al., 1991). Such a scenario is encountered for example in speech recognition and language translation (Graves and Jaitly, 2014), sequence labeling (Graves et al., 2006), and certain computer vision tasks. The capacity of recurrent NNs to build coherent outputs from causal sequences is the main property that motivates us to apply them to the estimation of the rms velocity. Indeed, the rms velocity at a particular time depends on the rms velocities at earlier times, as shown by equation 2. The output of the RCNN layer is fed to an LSTM cell, as a sequence in time of vectors with 32 elements, $\mathbf{x}_t$. The LSTM cell follows the formulation of Sak et al. (2014)

$$\mathbf{i}_t = \boldsymbol{\sigma}(\mathbf{W}_{ix}\mathbf{x}_t + \mathbf{W}_{im}\mathbf{m}_{t-1} + \mathbf{W}_{ic}\mathbf{c}_{t-1} + \mathbf{b}_i), \qquad (7)$$

$$\mathbf{f}_t = \boldsymbol{\sigma}(\mathbf{W}_{fx}\mathbf{x}_t + \mathbf{W}_{fm}\mathbf{m}_{t-1} + \mathbf{W}_{fc}\mathbf{c}_{t-1} + \mathbf{b}_f), \qquad (8)$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{h}(\mathbf{W}_{cx}\mathbf{x}_t + \mathbf{W}_{cm}\mathbf{m}_{t-1} + \mathbf{b}_c), \qquad (9)$$

$$\mathbf{o}_t = \boldsymbol{\sigma}(\mathbf{W}_{ox}\mathbf{x}_t + \mathbf{W}_{om}\mathbf{m}_{t-1} + \mathbf{W}_{oc}\mathbf{c}_t + \mathbf{b}_o), \qquad (10)$$

$$\mathbf{m}_t = \mathbf{o}_t \odot \mathbf{h}(\mathbf{c}_t). \qquad (11)$$

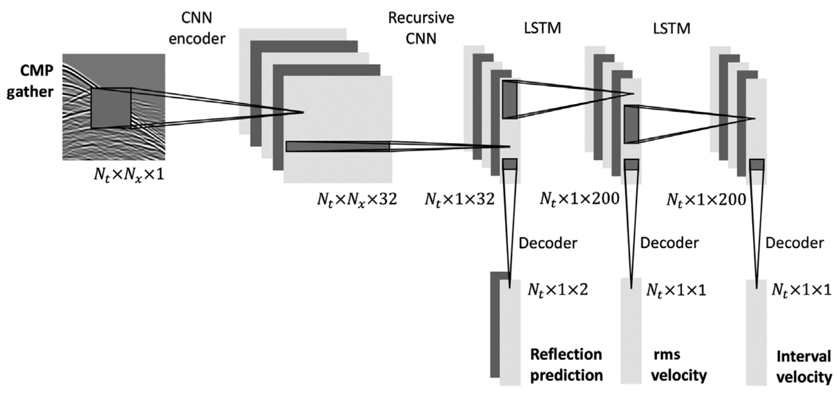

Figure 1. A schematic representation of the deep NN architecture used for velocity prediction. The architecture of the three main components of NN (the CNN encoder, the recursive CNN, and the LSTM decoder) are described in the text.

Here, the $\mathbf{W}$ terms are the weight matrices, $\mathbf{m}_t$ is the state or output of the cell at time t, and $\mathbf{i}_t$, $\mathbf{f}_t$, $\mathbf{o}_t$, $\mathbf{c}_t$, and $\mathbf{m}_t$ are, respectively, the input gate, forget gate, output gate, cell activation vectors, and cell output. All of these vectors have the same size, chosen to be 200 in this work. The operator $\odot$ stands for the element-wise vector product, $\boldsymbol{\sigma}$ is the logistic sigmoid function, and $\mathbf{h}$ is the hyperbolic tangent function. As can be seen, the output at time t of the cell $\mathbf{m}_t$ depends on the previous output $\mathbf{m}_{t-1}$ as well as the input $\mathbf{x}_t$, which allows to model complex, nonlinear behaviors. Also, the architecture of recurrent NNs allows inputs of different lengths, which means that CMPs of different acquisition time can be processed without the need to retrain the network.

The output of the LSTM unit is finally decoded using a matrix of trainable parameters of dimensions 200×1, resulting in a tensor of dimensions $N_t \times 1 \times 1$. Two sequential LSTM cells are used to obtain the predicted rms velocities $v^{\mathrm{rms}}_{\mathrm{pred}}$ and interval velocities $v^{\mathrm{int}}_{\mathrm{pred}}$.

Training the network

The training of a deep NN requires a large data set containing thousands to millions of examples in which each input is labeled with the appropriate output. The challenge for seismic processing is to obtain sufficient seismic data with accurate labeling, that is, CMP gathers for which we know the accurate velocity model. Because no such data set currently exists, we trained the NN on synthetic data, created to be similar to the real marine 2D survey processed later on. Synthetics allow us to generate a large quantity of accurately labeled data and allow a precise appraisal of the performance of the NN. Furthermore, as will be shown in the "Results" sections, the NN still performs well on real data despite being trained on synthetics.

Generation of labeled data

The synthetic data are generated by a finite-difference code (Fabien-Ouellet et al., 2017). We generate random 2D laterally homogeneous isotropic velocity models, with velocities between 1300 and 4000 m/s. We use absorbing boundary conditions on all sides of the model, preventing the occurrence of surface multiples in the resulting gathers. This is consistent with the industry practice of removing surface multiples before performing any velocity analysis. The data are modeled using first- and second-derivative Gaussian wavelets, randomly phase rotated with a peak frequency $f_{\max}$ chosen randomly between 21 and 31 Hz. We use an end-on spread acquisition geometry, with a minimum offset of 470 m and a maximum offset of 4075 m. The maximum time is chosen as 8 s. Time sampling (Δt) and trace spacing (Δx) are chosen to reproduce the acquisition parameters of the real seismic line presented later on. The dimensions of the resulting CMP gathers are $N_t = 2000$ and $N_x = 72$. The labels for the true rms velocities v_{true} are generated using equation 2, and the labels for primary reflection detection R_{true} are generated using equation 6 and $l = (1/4 f_{\max})/\Delta t$, on the basis of the randomly generated layered models.

The training set contains a total of 40,000 random models. To balance the complexity of those models, the training set is divided into four subsets. Each subset differs by the minimum number of layers contained in a model, chosen to be 5, 10, 30, and 50.

Loss function engineering

The total loss for learning the rms velocity contains several parts, needed to regularize the problem. For the classification task, we used the softmax cross-entropy loss function as a measure of the classification error

$$L_0 = -\sum_{n=1}^{N_t}\sum_{j=1}^{2} p_{nj} \log q_{nj}, \tag{12}$$

where p represents the true class labels R_{true} and q is the softmax normalized probability distribution of the predicted class labels R_{pred}. Denoted by $\|\cdot\|_2^2$ the sum of squares for all time samples, the second component of the loss function is given by the standard ℓ^2 error

$$L_1^{\mathrm{rms|int}} = \| v_{\mathrm{pred}}^{\mathrm{rms|int}} - v_{\mathrm{true}}^{\mathrm{rms|int}} \|_2^2, \tag{13}$$

where $v_{\mathrm{pred}}^{\mathrm{rms|int}}$ and $v_{\mathrm{true}}^{\mathrm{rms|int}}$ refer either to the rms or the interval velocities. To favor smooth velocity profiles, a contribution penalizing the velocity time-derivative misfits is also included with another ℓ^2 error term

$$L_2^{\mathrm{rms|int}} = \| \partial_t v_{\mathrm{pred}}^{\mathrm{rms|int}} - \partial_t v_{\mathrm{true}}^{\mathrm{rms|int}} \|_2^2, \tag{14}$$

where ∂_t denotes the time-derivative operator. The total loss function L is then given as a weighted sum of all these individual losses:

$$L = \alpha_0 L_0 + \alpha_1 L_1^{\mathrm{rms}} + \alpha_2 L_2^{\mathrm{rms}} + \alpha_3 L_1^{\mathrm{int}} + \alpha_4 L_2^{\mathrm{int}}, \tag{15}$$

where α_0, α_1, α_2, α_3, and α_4 are the scaling factors.

Training strategy

We adopted a hierarchical strategy divided into three phases with different choices of $\alpha_0, \alpha_1, \alpha_2, \alpha_3, \alpha_4$ — (1) reflection identification ($\alpha_0 = 0.95$, $\alpha_1 = 0.05$, $\alpha_2 = 0$, $\alpha_3 = 0$, and $\alpha_4 = 0$), (2) rms velocity training ($\alpha_0 = 0.1$, $\alpha_1 = 0.85$, $\alpha_2 = 0.05$, $\alpha_3 = 0$, and $\alpha_4 = 0$), and (3) interval and rms velocity training ($\alpha_0 = 0.1$, $\alpha_1 = 0.45$, $\alpha_2 = 0.05$, $\alpha_3 = 0.35$, and $\alpha_4 = 0.05$). We performed 1000 iterations for the first stage and 10,000 iterations for the later stages, using the Adam optimizer (Kingma and Ba, 2014), with a batch size of 40 CMPs. We developed this training strategy from the heuristic that the training should start from simple problems and gradually progress to more complex problems. The choice of the free parameters $\alpha_i (0 \le i \le 4)$ is performed by trial and error, until a reasonable convergence rate is achieved. The hierarchical strategy proved to be essential in achieving low prediction errors, in a reasonable amount of training time.

We also used ensemble learning (Opitz and Maclin, 1999), which has been shown to reduce the generalization error, leading to more robust predictions. We trained 16 different NNs, which differ by the random weight initialization and the order in which they see the training examples. The predictions are given by the mean of the ensemble. The standard error of the predictions is also computed, which provide a measure of the variability and stability of the estimated velocities.

RESULTS

1D synthetic data

For this first test, we built a test data set comprising of 1600 1D layered models. Note that the models in the test set were not seen by the NN during training. Because the network has been trained on a similar data distribution, these results represent an upper limit on the performance of the NN.

Figure 2a shows the loss as a function of the epoch during training, which goes up to 20. In other words, an example from the training set was seen 20 times on average during iterations. The three steps of the scheduling appear in blue, orange, and green, for phase 0, 1, and 2, respectively. As the cost function is different for each phase, their magnitude cannot be readily compared. Note, however, that in the three cases, the training error decreases rather slowly after the first few epochs. Figure 2b shows the prediction error for the rms velocity. As can be seen, the prediction error decreases for all epochs, which means that we did not overfit the training set. Finally, Figure 2c shows the prediction error for the interval velocity. The rms error does not decrease until phase 2 because no training on interval velocity occurs for phases 0 and 1.

The 10th, 50th, and 90th percentiles of the predicted rms velocity rms error are shown in Figure 3d–3f, respectively. Overall, the rms error for the rms velocity predictions is 44 m/s. As can be visually seen, predictions are quite accurate and follows the semblance trend. The predicted velocities vary smoothly in time and nicely follow the trend of the true rms velocity profiles. The standard deviation of the ensemble is 36 m/s, below the rms error, which means that the estimates provided by the ensemble are stable.

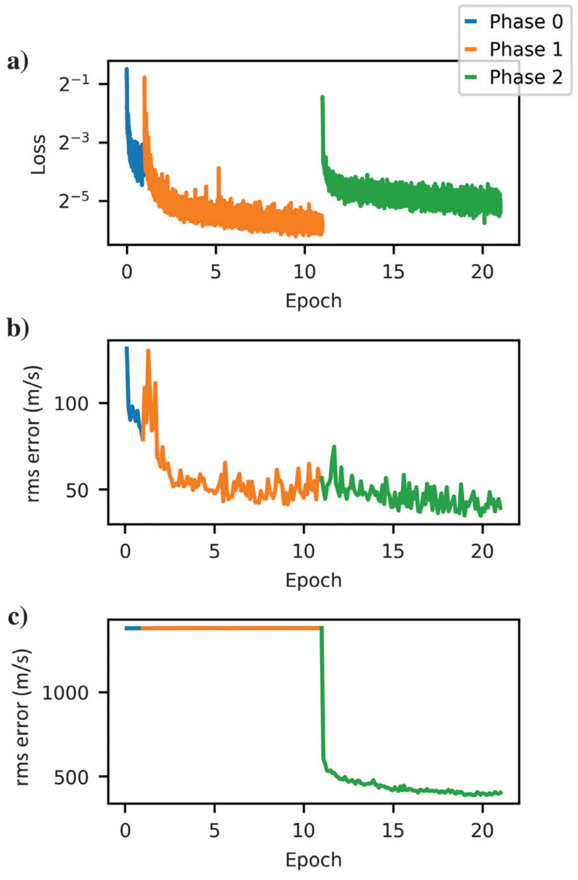

Figure 2. (a) Loss function during training as several epochs, (b) the rms error on the test set for the rms velocity, and (c) the rms error on the test set for the interval velocity.

The interval velocity, although noisier than the rms velocity, is still quite well-predicted with an rms error of 323 m/s. The interval velocity profiles contain sharp interfaces, and the NN is able to predict them even for very large velocity contrasts. The standard deviation of the interval velocity predicted by the ensemble is 91 m/s, well below the rms error. Hence, the standard deviation of the ensemble

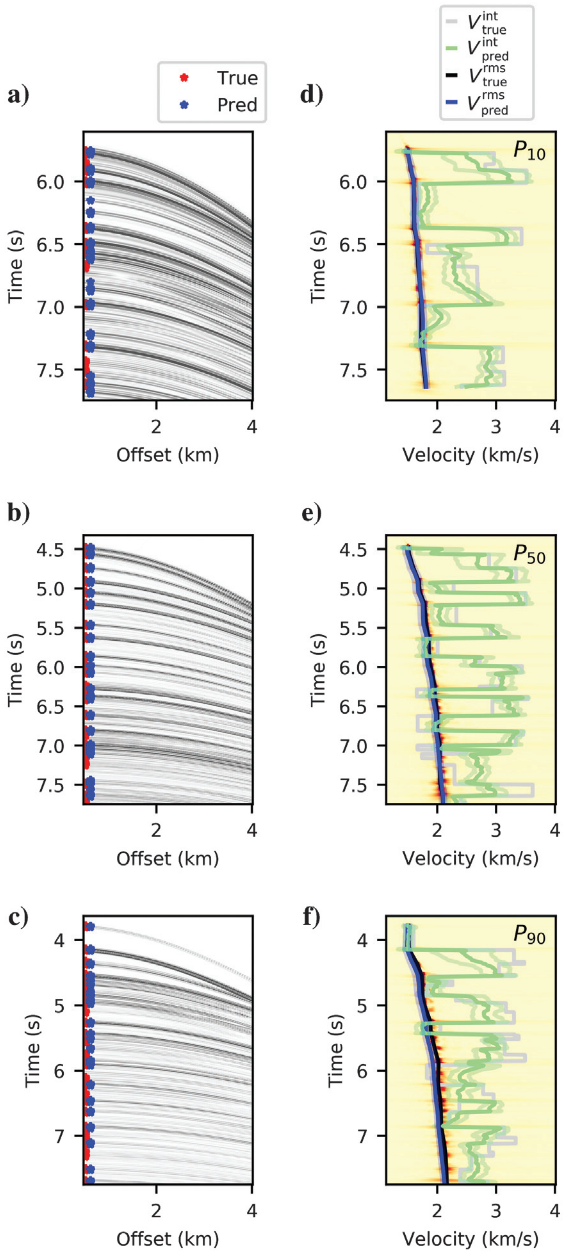

Figure 3. Examples of the prediction of the NN for three different input CMP gathers. Gathers are shown in (a-c), corresponding, respectively, to the 10th, 50th, and 90th percentiles of the predicted rms velocity rms error. The corresponding true and predicted rms velocities (v^{rms}_{true} and v^{rms}_{pred}), and the true and predicted interval velocities (v^{int}_{true} and v^{int}_{pred}) are shown overlain on the semblance panels in (d-f), respectively, along with their standard deviation, in light green and light blue, respectively. Models are clipped at the time of the last reflection.

cannot be viewed as an uncertainty of the predictions, although it tends to be larger in areas of larger prediction errors.

Finally, the CMP gathers corresponding to the velocity profiles in Figure 3d–3f are shown in Figure 3a–3c, respectively. The rate of detection of primary reflection events is very good — 7.4% of true positives, 89.2% of true negatives, 2.4% of false negatives, and 1.0% of false positives. As Figure 3 shows, the NN does a pretty good job of detecting primary reflections and ignoring multiples. Although far from perfect, these results show that some robustness to multiples can be learned by deep NNs.

We interpret the good performance of our NN as evidence that it has built a representation of the CMP gather that is linked to the rms and interval velocities for a 1D flat-layered earth medium (equation 2). It can be argued that most of this knowledge is specific to 1D velocity models, with synthetic data. However, 1D analysis is the basis of more complete processing, and our results can be generalized to more complex models, as shown by the next set of results.

2D synthetic data

To show the potential for generalization of our approach, we tested the capacity of the trained NN to reconstruct 2D velocity models. We generated a testing set of 100 2D models containing slightly dipping layers (maximum dip of 6°) and random velocity perturbations (maximum of 8% of the layer velocity). We used the same acquisition configuration as for the 1D training.

The 2D interval-velocity models predicted by the NN are shown in Figure 4. The models shown correspond to the 10th, 50th, and 90th percentiles of the predicted interval velocity rms error. As can be seen, the 2D structure is well-recovered, albeit with less accuracy than for 1D flat-layered earth models. The total rms error for the interval velocity increased from 322 to 424 m/s, and the rms velocity rms error increased from 42 to 44 m/s. Still, the NN manages to generalize to simple 2D models. Although our simple NN is not appropriate for more complex cases, the representation built by our CNN encoder could be used as pretrained layers for deeper NNs.

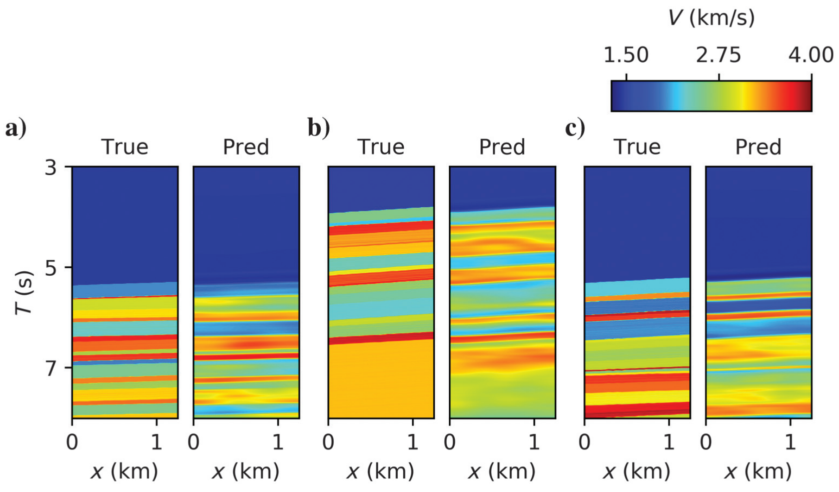

Figure 4. Generalization of 2D-layered models of the 1D-trained NN. The right panels in (a-c) correspond to the predicted interval velocity models for 10th, 50th, and 90th percentiles of predicted interval velocity rms error on the 2D data set. The corresponding true models are shown in the left panel of each figure. Models are clipped at the time of the last reflection.

Application to real data

As a final test, we present the application of the NN to a marine data set. The USGS has made public 21 seismic lines acquired in the 1970s along the U.S. Atlantic Continental Margin (Hutchinson et al., 1997). The shot gathers as well as the stacked section are available for download through the USGS website (USGS, 1978).

We applied our NN on the deepest section of line 32. The preprocessing of the CMP gathers is kept minimal. We first interpolated the traces to a constant interval of 50 m and applied a bandpass Butterworth filter between 10 and 35 Hz. Because the acquisition parameters of the training set and the real data set are the same, the NN can readily be applied to the real seismic line without further training.

Figure 5 shows CMPs 250, 1000, and 1750, along with the NN average outputs and standard deviation. No sonic well-log data exist for this seismic line. Assessment of the quality of the results will thus remain qualitative. Reflection identification is noisier for the real data set than for the synthetic case, due to overlapping multiples at early times. Because this output is only an accessory to train the NN for velocity estimation, it is a minor setback. On the other hand, the estimated rms velocity in all cases follows the trend shown by the semblance maximas. It does not follow all maximas, skipping what may be interbed multiples. The results are, however, clearly affected by free surface multiples, which causes the semblance maximas to decrease rapidly to an rms velocity close to the water velocity of 1500 m/s. Free surface multiples can be seen from 9.5 s in Figure 5d, 8.3 s in Figure 5e, and 6 s in Figure 5f. Although the predicted rms velocity does not fall back to 1500 m/s, it decreases substantially. The same behavior can be seen for the interval velocity. This is expected because no free surface multiples are present in the training set, so the NN could not learn how to process those arrivals. One way to address this issue is to apply standard surface multiple removal techniques to process the CMPs before feeding them to the NN. Alternatively, a NN trained to be robust to free surface multiples could be designed.

The light-blue and light-green lines in Figure 5 show the standard deviation of the predicted rms and interval velocities, respectively. As can be seen, the standard deviation remains small when clear reflections are visible and becomes large when free surface multiples are present.

Figure 6 shows the predicted rms velocity, the predicted interval velocity, and the stacked section using the predicted velocities, for CMPs 1–2080. The stacked section is obtained by processing each CMP individually with the following sequence: (1) application of the NMO correction based on the estimated rms velocity, (2) summation of all the traces of the CMP, (3) application of gain proportional to t^2, and (4) rms trace normalization. The processing workflow is kept as simple as possible to better appraise the output of the NN. Note first that the rms velocity is consistent among all CMPs. The velocity starts increasing from the arrival time of the seafloor reflection, which decreases from CMP 0 to CMP 2080. It then increases up to the arrival time of the seafloor surface multiple, which causes a rapid decrease in the estimated velocity. For interval and rms velocities, the range of valid estimation is thus between times 0 and the time of the first seafloor multiple. The estimated rms velocity produced a coherent stacked section that is comparable to what Hutchinson et al. (1997) obtained, reproduced in Figure 6d. This shows that the NN is readily applicable for processing real surveys, even with the rudimentary training set that is used.

Comparing the stacked section and the interval velocity, most sharp velocity variations are linked to coherent reflectors. This is especially true for the deeper reflections below 8 s, between CMP 0 and 1250. The shallower section shows more heterogeneous velocity variations, which is linked to less coherent reflections and strong diffraction hyperbolas. As the 1D velocity assumption is less applicable in the shallow section, the estimated velocities may be less reliable than for the deeper section. The velocity range between 2000 and 3000 m/s is coherent with the geology of the Blake Plateau and Outer Ridge. In particular, the increase from 2200 m/s to more than 2800 m/s is linked to reflectors identified as the top of the Cretaceous by Hutchinson et al. (1997).

DISCUSSION

Our findings indicate that a deep NN has the potential to estimate the rms and interval velocities accurately for complex 1D-layered and gently dipping 2D-layered models directly from the seismic waveform. Recent studies point to the same direction. Araya-Polo et al. (2018) show that velocity in depth could be estimated with a combination of a deep convolutional network with fully connected layers. However, their estimation is based on semblance, which limits the potential of the NN to use the phase and amplitude information of the seismic waveform. Yang and Ma (2019) recover 2D velocity models with a deep convolutional NN model directly from the seismic waveform, but their training and testing models remain quite simple, with large layer thicknesses compared with the wavelength. Our results show the validity of the approach for complex 1D models with thin layers (the minimum layer thickness was 1/4 the maximum wavelength). The level of complexity of the training models required to train a NN to obtain good performance on real data remains, however, an open question.

Another major finding is that NNs can be trained purely with synthetic seismic data and still perform adequately well with real data. For a data-driven approach such as deep learning, accessibility to an extensive labeled database is paramount. This is problematic for seismic processing in two ways: data are often proprietary and are hence difficult to share among researchers, and, most importantly, obtaining correctly labeled data is difficult or even impossible as the true velocity models are never known exactly. Synthetics solve both of these issues because they can be generated relatively cheaply and allow us to fix all parameters needed to generate accurate labeling of the data set. The drawback is that some important features found in real data, such as noise, may be missing from synthetics, preventing the network from performing well on real data. This happened in this work in the case of free surface multiples — because they were absent from the synthetic training set, their presence greatly affected the estimated velocity for the real data set. Surface multiples can be

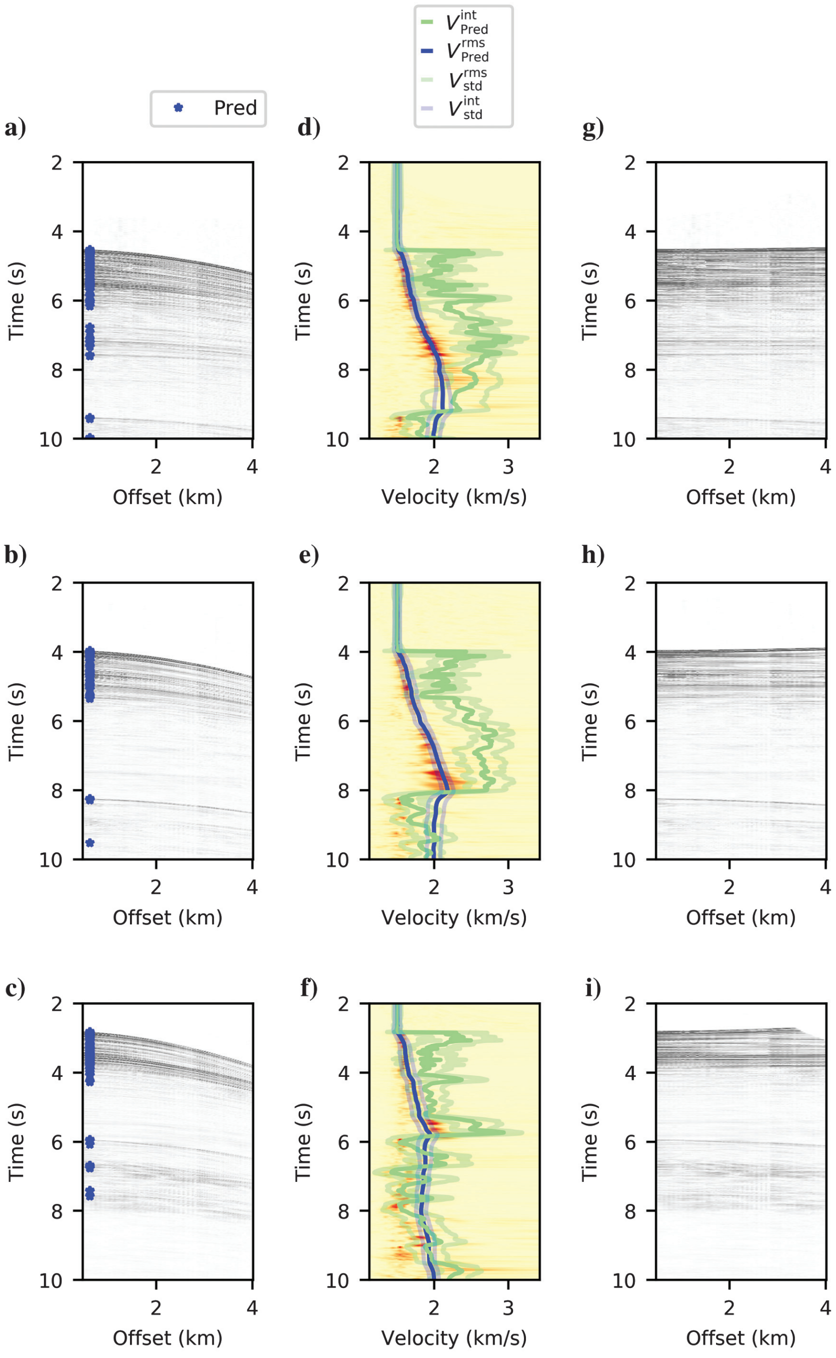

Figure 5. The NN predictions for three CMPs from seismic line 32. The CMPs 150, 1000, and 1750 are shown in (a-c), their semblance panels are shown in (d-f), and the NMO-corrected gathers are shown in (g-i), respectively. Note that the pale-blue and pale-green lines are the bounds given by the standard deviation of the predictions of an ensemble of 16 NNs.

modeled quite easily, so this is not a real shortcoming. However, it raises the question of the accuracy of the synthetic data required for training a network that generalizes well to real data. This issue should be the focus of further research.

For the moment, it is too early to say whether the NN we propose has real applicability for seismic processing. The real data results show that adequate estimation of the rms velocity can be obtained with the NN. However, the NN was neither designed to maximize the stacking power nor obtain a coherent stacked section. More rigorous testing on multiple surveys and comparison with expert seismic processors would be required to assess how accurate the NN is, compared with human experts. The main advantage of the NN is that rms estimation is automatic and cheap to compute — the 2000 CMPs of seismic line 32 took less than 5 min to complete on a single Nvidia Quadro 6000 GPU. The cost of training the network is much higher (approximately 24 h on a single GPU). Ideally, training should be performed once and lead to a network that can perform well with different survey acquisitions. This would render the training time irrelevant. In fact, our NN can be used on slightly different acquisition parameters — with different source signatures (as shown in our training set and test set) and different number of traces or sampling intervals (tested, but not shown herein). However, more work is required to develop a truly general NN, able to handle arbitrary geometries, sensors, or environments (land or marine).

The main aim of this work was to propose a benchmark problem, simple enough so that different NN designs can be tested at a small cost, but complex enough so that insights learned from it can be applicable to more realistic applications. As a matter of fact, due to the sheer volume of data required for 3D data processing, simple design options should be tested beforehand on smaller, simplified problems. The estimation of 1D velocity profile from a CMP gather is such a test. NNs trained on the 1D benchmark problems will generally have limited practical applicability. For instance, our NN is adequate only for gently dipping layered earth models and cannot handle more complex structures, which is a general limitation of simple NMO-based analysis. However, NNs trained on 1D-layered models such as ours can be used as a building block for more elaborate networks. For example, the encoder could be embedded in a larger network with pretrained weights to reduce the training time of a network designed to predict 3D velocity models.

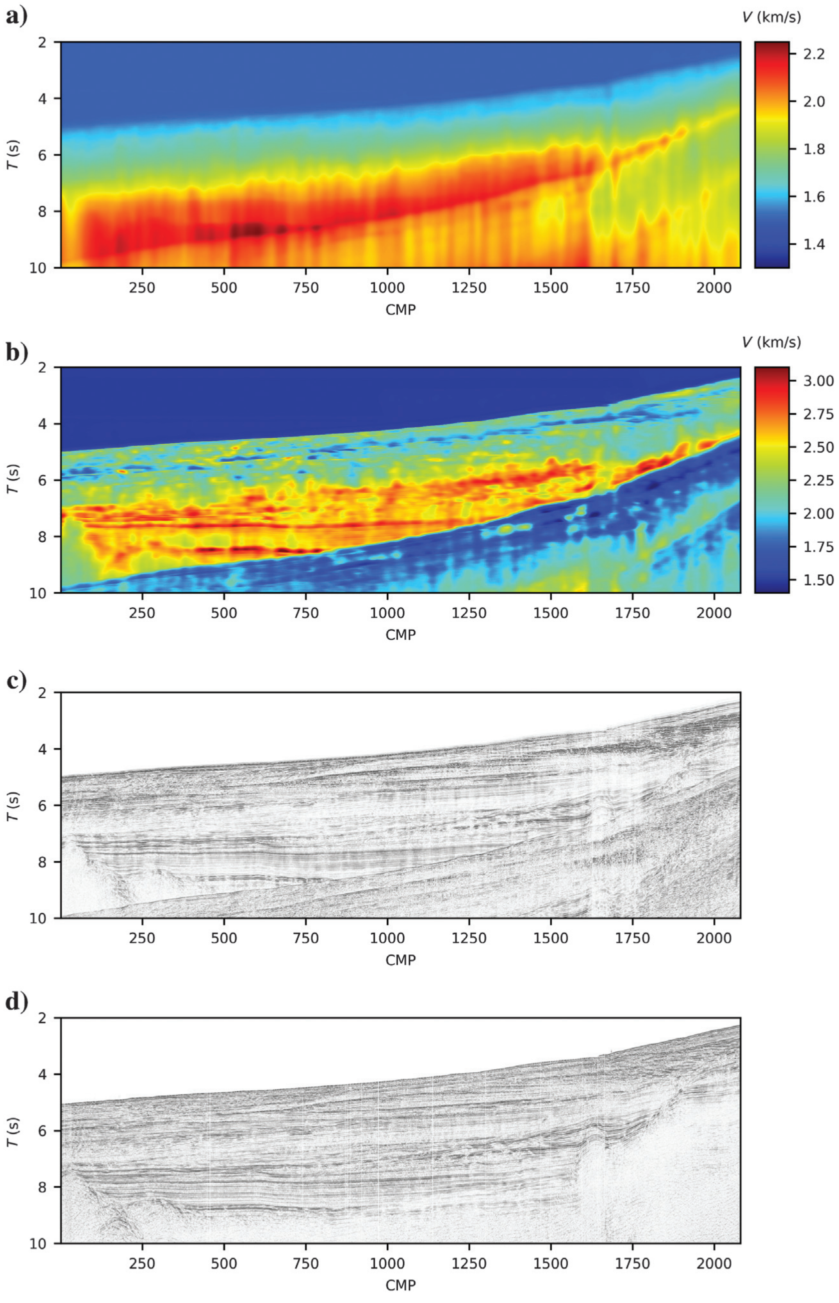

Figure 6. (a) Predicted rms velocity, (b) predicted interval velocity, (c) stacked section using the predicted velocity, and (d) the stacked section provided by USGS for seismic line 32.

CONCLUSION

In this work, we proposed to study the simplified problem of 1D velocity estimation to gain insights on the design of NN for general velocity inversion. We have shown that by using a deep convolutional NN, one can build a representation of seismic data that is suitable for automated velocity analysis. Based on this representation, the rms and interval velocities were estimated with recurrent NNs, for complex 1D velocity models with high velocity contrasts and thin layers. This provides evidence that a data-driven approach using NNs has the potential to automate velocity analysis in realistically complex environments and suggests the plausibility of automating modern velocity model-building workflows used for imaging the subsurface.

We further showed that training with synthetic data is a plausible option to develop a NN that performs reasonably well on real data. The estimated velocities correlate well with semblance and lead to a coherent stack section. However, further testing is required for estimating its true applicability for the purpose of stacking or migration. Rather, our work focused on proposing a benchmark problem simple enough to guide the development of more sophisticated NNs capable of predicting 3D velocity models in depth.

A major limitation of the NN architecture discussed in this paper is that it is only efficient for gently dipping layered earth models. It should be noted that the same limitation applies to conventional processing based on normal moveout analysis, which is still routinely used in time velocity analysis. However, our results showed that deep convolutional networks and recurrent NNs are promising architectures for larger networks, capable of handling 3D velocity structures.

ACKNOWLEDGMENTS

The training and testing of the deep-learning architecture were implemented using the TensorFlow framework (Abadi et al., 2015). We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), funding reference number RGPIN-2018-06163. For the duration of this research, R. Sarkar was partially supported by the Stanford Exploration Project.

DATA AND MATERIALS AVAILABILITY

Data associated with this research are available, and can be accessed via the following URLs: http://doi.org/10.5281/zenodo.3492114 and https://github.com/GeoCode-polymtl/Deep_1D_velocity.git.

REFERENCES

Abadi, M., A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, 2015, Tensorflow: Large-scale machine learning on heterogeneous distributed systems, arXiv preprint arXiv:1603.04467.

Alaudah, Y., H. Di, and G. AlRegib, 2017, Weakly supervised seismic structure labeling via orthogonal non-negative matrix factorization: 79th Annual International Conference and Exhibition, EAGE, Extended Abstracts, doi: 10.3997/2214-4609.201700921.

Araya-Polo, M., J. Jennings, A. Adler, and T. Dahlke, 2018, Deep-learning tomography: The Leading Edge, **37**, 58–66, doi: 10.1190/tle37010058.1.

Baronian, C., M. Riahi, and C. Lucas, 2009, Applicability of artificial neural networks for obtaining velocity models from synthetic seismic data: International Journal of Earth Sciences, **98**, 1173–1184, doi: 10.1007/s00531-008-0314-3.

Di, H., and G. AlRegib, 2017, Seismic multi-attribute classification for salt boundary detection — A comparison: 79th Annual International Conference and Exhibition, EAGE, Extended Abstracts, doi: 10.3997/2214-4609.201700919.

Di, H., M. Shafiq, and G. AlRegib, 2017, Multi-attribute k-means cluster analysis for salt boundary detection: 79th Annual International Conference and Exhibition, EAGE, Extended Abstracts, doi: 10.3997/2214-4609.201700915.

Elman, J. L., 1990, Finding structure in time: Cognitive Science, **14**, 179–211, doi: 10.1207/s15516709cog1402_1.

Elman, J. L., 1991, Distributed representations, simple recurrent networks, and grammatical structure: Machine Learning, **7**, 195–225.

Fabien-Ouellet, G., E. Gloaguen, and B. Giroux, 2017, Time-domain seismic modeling in viscoelastic media for full waveform inversion on heterogeneous computing platforms with OpenCL: Computers & Geosciences, **100**, 142–155, doi: 10.1016/j.cageo.2016.12.004.

Fabien-Ouellet, G., and R. Sarkar, 2019, Code for seismic velocity estimation: A deep recurrent neural-network approach, Version v1.0-alpha, Zenodo, doi: http://doi.org/10.5281/zenodo.3492114.

Gers, F. A., J. Schmidhuber, and F. Cummins, 2000, Learning to forget: Continual prediction with LSTM: Neural Computation, **12**, 2451–2471, doi: 10.1162/089976600300015015.

Graves, A., S. Fernández, F. Gomez, and J. Schmidhuber, 2006, Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks: Proceedings of the 23rd International Conference on Machine Learning, ACM, 369–376.

Graves, A., and N. Jaitly, 2014, Towards end-to-end speech recognition with recurrent neural networks: International Conference on Machine Learning, 1764–1772.

Hochreiter, S., and J. Schmidhuber, 1997, Long short-term memory: Neural Computation, **9**, 1735–1780, doi: 10.1162/neco.1997.9.8.1735.

Hutchinson, D., C. Poag, A. H. Johnson, P. Popenoe, and C. Wright, 1997, Geophysical database of the east coast of the United States; southern Atlantic margin, stratigraphy and velocity in map grids: Report 2331-1258, US Geological Survey.

Kim, J., J. K. Lee, and K. M. Lee, 2016, Deeply-recursive convolutional network for image super-resolution: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1637–1645.

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

Langer, H., G. Nunnari, and L. Occhipinti, 1996, Estimation of seismic waveform governing parameters with neural networks: Journal of Geophysical Research: Solid Earth, **101**, 20109–20118, doi: 10.1029/96JB00948.

Lewis, W., D. Vigh, A. M. Popovici, and S. Fomel, 2017, Deep learning prior models from seismic images for full-waveform inversion: 87th Annual International Meeting, SEG, Expanded Abstracts, 1512–1517, doi: 10.1190/segam2017-17627643.1.

Lu, P., M. Morris, S. Brazell, C. Comiskey, and Y. Xiao, 2018, Using generative adversarial networks to improve deep-learning fault interpretation networks: The Leading Edge, **37**, 578–583, doi: 10.1190/tle37080578.1.

Opitz, D., and R. Maclin, 1999, Popular ensemble methods: An empirical study: Journal of Artificial Intelligence Research, **11**, 169–198, doi: 10.1613/jair.614.

Roth, G., and A. Tarantola, 1994, Neural networks and inversion of seismic data: Journal of Geophysical Research: Solid Earth, **99**, 6753–6768, doi: 10.1029/93JB01563.

Sak, H., A. Senior, and F. Beaufays, 2014, Long short-term memory recurrent neural network architectures for large scale acoustic modeling: 15th Annual Conference of the International Speech Communication Association.

Servan-Schreiber, D., A. Cleeremans, and J. L. McClelland, 1991, Graded state machines: The representation of temporal contingencies in simple recurrent networks: Machine Learning, **7**, 161–193.

Socher, R., C. D. Manning, and A. Y. Ng, 2010, Learning continuous phrase representations and syntactic parsing with recursive neural networks: Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop, 1–9.

USGS, 1978, Field activity details for field activity 1978-015-fa, https://cmgds.marine.usgs.gov/fan_info.php?fan=1978-015-FA, accessed 28 July 2019.

Yang, F., and J. J. G. Ma, 2019, Deep-learning inversion: A next generation seismic velocity-model building method: Geophysics, **84**, no. 4, R583–R599, doi: 10.1190/geo2018-0851.1.