Deep-learning tomography

Mauricio Araya-Polo[1], Joseph Jennings[1,2], Amir Adler[3], and Taylor Dahlke[2]

Abstract

Velocity-model building is a key step in hydrocarbon exploration. The main product of velocity-model building is an initial model of the subsurface that is subsequently used in seismic imaging and interpretation workflows. Reflection or refraction tomography and full-waveform inversion (FWI) are the most commonly used techniques in velocity-model building. On one hand, tomography is a time-consuming activity that relies on successive updates of highly human-curated analysis of gathers. On the other hand, FWI is very computationally demanding with no guarantees of global convergence. We propose and implement a novel concept that bypasses these demanding steps, directly producing an accurate gridding or layered velocity model from shot gathers. Our approach relies on training deep neural networks. The resulting predictive model maps relationships between the data space and the final output (particularly the presence of high-velocity segments that might indicate salt formations). The training task takes a few hours for 2D data, but the inference step (predicting a model from previously unseen data) takes only seconds. The promising results shown here for synthetic 2D data demonstrate a new way of using seismic data and suggest fast turnaround of workflows that now make use of machine-learning approaches to identify key structures in the subsurface.

Introduction

Exploration workflows are under great pressure due to such factors as the need to improve performance at lower costs and the ongoing avalanche of data coming from new generations of sensors and modern acquisition systems. Some of the key steps in exploration workflows depend on domain experts. Their time is precious and limited, but the amount of data that needs to be thoroughly analyzed is increasing. In addition, the complexity of some exploration areas requires extra attention. The problem can be summarized as an explosion of increasingly complex data.

Geoscientists need to be empowered with new tools that digest as much data as possible before a human expert intervenes. The high-performance-computing revolution (BizTech, 2014) shares the same purpose but essentially targets processing speed rather than any other specific step of the exploration workflow. Advanced data-oriented algorithms look to improve every step of the workflow through a deeper understanding of the data, from extracting the relevant information to having a better awareness of the rest of the steps in a more integrated fashion rather than through silos of knowledge.

What we propose in this work goes beyond what is becoming the new norm, which is machine-learning techniques being applied to specific well-known steps of the workflow. This same magazine carried a special section on how analytics and machine learning (*TLE*, March 2017) are paving inroads in different aspects of the exploration workflow, but most work is still focused on identifying features or attributes in migrated images (Hale, 2012; Hale, 2013; Guillen, 2015; Addison, 2016; Bougher and Herrmann, 2016), therefore helping to tackle the interpretation step. Very little has been proposed to help directly with processing or velocity-model building. In general, the literature is abundant with refinements to this workflow, but it remains largely untouched.

Our method produces velocity models directly from raw seismic data in a way that is alternative to classic tomography. It is also automatic and without the need for human intervention. The machine-learning technique employed follows recent work (Zhang et al., 2014; Frogner et al., 2015; Dahlke et al., 2016; Araya-Polo et al., 2017) that demonstrates this new approach, which uses a deep neural network (DNN) statistical model to transform raw-input seismic data directly to the final mapping in 2D or 3D. The computational costs come mostly from training, which happens only once up front. After training, velocity-model reconstruction costs are negligible, thus making the overall computing costs a fraction of that needed for traditional techniques, in particular those involving partial-differential-equation-based simulation. One key element of our method is the use of a feature based on semblance that predigests velocity information for the training process. This feature extraction step is automated and not subject to human bias.

In terms of deployment modes, we foresee models being trained with specific data belonging to different major formations, such as unconventional, presalt, or subsalt. The main concerns relate to the generalization error, which basically sets the limits on how much a predicting model can accurately predict for unseen data. Finally, regarding exploration workflows, one can imagine this technique being used just after data acquisition. Then, trained models can be loaded up to the cloud from which interpreters can pull realizations, thus performing online scenario testing when feeding back their model modifications to applications such as the one proposed in Araya-Polo et al. (2017). This imagined workflow is fully machine learning based, flexible, and with the domain experts at the center of the critical decision-making process. If it is accompanied by the proper resourcing, this workflow approaches a real-time ubiquitous experience.

In this paper, we start by explaining the basics of the problem followed by discussions on deep learning. Next, we introduce the general workflow used by our machine-learning system, which we termed GeoDNN. Then, we discuss our results and experiments with 2D synthetic data. Finally, conclusions with directions for future work are presented.

[1]Shell International Exploration & Production Inc.
[2]Stanford University.
[3]MIT.

https://doi.org/10.1190/tle37010058.1.

Figure 1. Tomography reconstruction of velocity models from recorded seismic data.

Problem formulation

Formally, the traditional tomography problem can be expressed as the minimization of the following objective function:

$$J(m) = \left\| d_m(m) - d_{obs} \right\|_2^2, \tag{1}$$

where m is the optimal velocity model that minimizes $J(m)$, d_m is a data vector modeled from a nonlinear modeling operator $f(m)$, and d_{obs} is the recorded data vector. While it is common to minimize the sum of the squares (represented here by the square of the L2 norm), other objective functions may be used. Note that in the case of traveltime tomography, the data vectors contain traveltimes that are modeled via the solution of the eikonal equation. Alternatively, in the full-waveform inversion (FWI) case, the data contain the seismic traces that are modeled via the numeric solution of the wave equation.

As is apparent by the nonlinear relationship between d_m and m, this inversion is nonlinear. Additionally, since for reflection seismic surveys d_{obs} contains surface seismic data, it does not contain all of the necessary information to define a velocity model that varies arbitrarily with depth and along the horizontal directions (Biondi, 2006). This means that, in general, minimizing the above equation is an ill-posed problem. While in using a deep-learning approach to tomography we do not rely on numerical solutions of the eikonal or wave equations, we still need to consider the nonlinearity and ill-posedness of this inverse problem.

The application of neural networks for velocity estimation and for geophysical applications in general is not new (van der Baan and Jutten, 2000). The first use of neural networks for velocity estimation was proposed by Röth and Tarantola (1994) in which neural networks are used to estimate 1D velocity functions from shot gathers. Nath et al. (1999) use neural networks for traveltime crosswell tomography. After training their network using traveltime maps and synthetic velocity models as training data, the network was then used to tomographically estimate velocities for crosswell data acquired in West Bengal, India. Although the problem we attempt to solve is similar, our work is novel in that it makes use of the recent development of more advanced DNN architectures; moreover, we use all of the data (not only selected traveltimes) to train our DNN and perform tomography.

Machine learning of tomography operators via DNNs

Using machine-learning algorithms is an appealing alternative to classic seismic processing, and among this class of algorithms, we have implemented the tomography operator using a DNN. The tomography operator is learned from seismic training data using statistical-learning (Hastie et al., 2001) principles. The tomography process is depicted in Figure 1. It performs reconstruction of the velocity model from raw seismic traces or from features computed from raw seismic traces (as part of the tomography operator). In a real-life application, the ground-truth model is unavailable, and the tomography operator is designed to minimize the difference between the reconstructed velocity model and the (unavailable) ground-truth one.

In the statistical learning framework, the tomography operator is learned using a collection of N training examples $\{X_i, V_i\}_{i=1}^N$, where X_i denotes the seismic traces (or features of seismic traces) generated from the i-th velocity model V_i. Specifically, the tomography operator is learned by solving the following optimization problem:

$$\hat{\alpha} = \arg\min_{\alpha} \frac{1}{N} \sum_{i=1}^{N} L\left(V_i, \underbrace{T(X_i, \alpha)}_{\hat{V}_i} \right), \tag{2}$$

where $T(X_i, \alpha)$ is the tomography operator parameterized by the coefficients vector α, and its output is the reconstructed velocity model $\hat{V}_i$. The loss function $L(V_i, \hat{V}_i)$ measures the difference between the ground-truth velocity model V_i and its reconstructed version $\hat{V}_i$. The loss function we employed is the squared error[4] $L(V_i, \hat{V}_i) = (V_i - \hat{V}_i)^2$, which is frequently used in regression problems, and leads to the following optimization problem:

$$\hat{\alpha} = \arg\min_{\alpha} \frac{1}{N} \sum_{i=1}^{N} \left(V_i - T(X_i, \alpha) \right)^2. \tag{3}$$

A frequently used minimization approach is the gradient descent, which iteratively updates the coefficients vector as follows:

$$\alpha_{t+1} = \alpha_t - \mu \frac{\partial L_E}{\partial \alpha}, \tag{4}$$

[4] Note that in the case of two images, the squared error loss is computed pixel-based; namely, it is the sum of all squared pixels differences.

where μ is a positive learning rate, L_E is the empirical loss

$$L_E = \frac{1}{N}\sum_{i=1}^{N} L\left(V_i, T\left(X_i, \alpha\right)\right), \tag{5}$$

and the gradient of L_E, with respect to α, is given by

$$\frac{\partial L_E}{\partial \alpha} = \frac{1}{N}\sum_{i=1}^{N} \frac{\partial L}{\partial \alpha}\left(V_i, T\left(X_i, \alpha\right)\right). \tag{6}$$

The tomography operator $T(X_i, \alpha)$ was implemented using a DNN, as detailed in the following.

DNNs

DNNs are powerful machine-learning algorithms (LeCun et al., 2015; Goodfellow et al., 2016) that provide state-of-the-art results in numerous computer vision, speech processing, and artificial intelligence problems. In particular, DNNs provide excellent results for imaging inverse problems such as denoising (Burger et al., 2012; Xie et al., 2012), super-resolution (Dong et al., 2016), compressed sensing (Adler et al., 2017), and X-ray computed tomography (Wang, 2016; Würfl et al., 2016). In addition, according to the universal approximation theorem (Hornik et al., 1989), DNNs can be used to approximate any arbitrary continuous function up to a specified accuracy. For these reasons, there is great promise in using this approach to approximate complex functions that are highly nonlinear.

DNNs are composed of "layers" of weighted nodes as depicted in Figure 2. The input to the network is connected to the input layer, which is followed by a varying number of hidden layers, and eventually the output of the network is computed at the output layer. Each hidden layer's inputs are activated by the outputs of the previous layer. These networks are trained with examples per the statistical-learning approach in which the correct output (label) is known for a given input, and the weight parameters in the nodes of the network evolve due to the minimization of the error between the prediction and true value. This causes the network to increasingly become a better predictor of the training examples and ultimately of any example (assuming proper training) of a class of data that is similar in nature to the training data.

The proposed tomography operator is therefore described as follows, assuming for example three hidden layers:

$$T(X, \alpha) = f_{out}(f_3(f_2(f_1(X, \alpha_1), \alpha_2), \alpha_3), \alpha_{out}), \tag{7}$$

where f_{out} is the output layer function, parameterized by α_{out}, and the hidden layer functions are f_1, f_2, and f_3, each parameterized by α_1, α_2, and α_3, respectively (the vector α is composed by α_1, α_2, α_3, and α_{out}). Our DNN has been highly tuned using hyperparametrical optimization. It also incorporates current techniques such as batch normalization and dropout. It is implemented in Google's Tensorflow open-source library.

Our ability to design effective neural networks is limited by constraints in computing resources. More complex networks are more computationally demanding to train, and generating accurate training examples can be computationally expensive due to large-scale forward modeling. Ultimately, our predictions are only as good as the complexity and refinement of our neural network coupled with the relevance and quality of the features we choose as inputs.

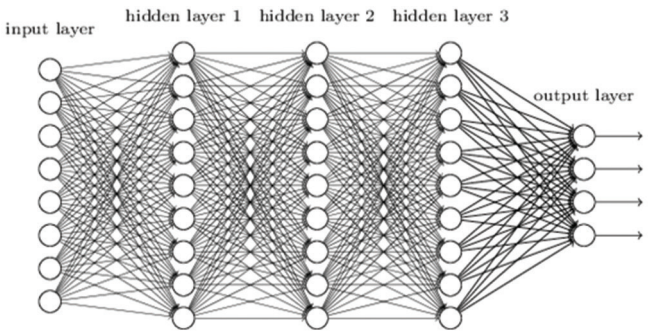

Figure 2. Topology of a DNN with three hidden layers.

Workflow

Since we lacked abundant labeled data, we risked the neural network's result being bound by the limited number of examples, which often leads to overfitting of the learning model to the training data. Also, because control of the main parameters involved (data generation) is key when proving a new concept, in this work we focus on results for 2D synthetic only, where our model generator produced large enough models for training and testing. Therefore, we introduce two workflows, one for training and one for inference (a.k.a. testing), as explained below.

In the training workflow (Figure 3), the first step is the pseudorandom generation of thousands of unbiased velocity models

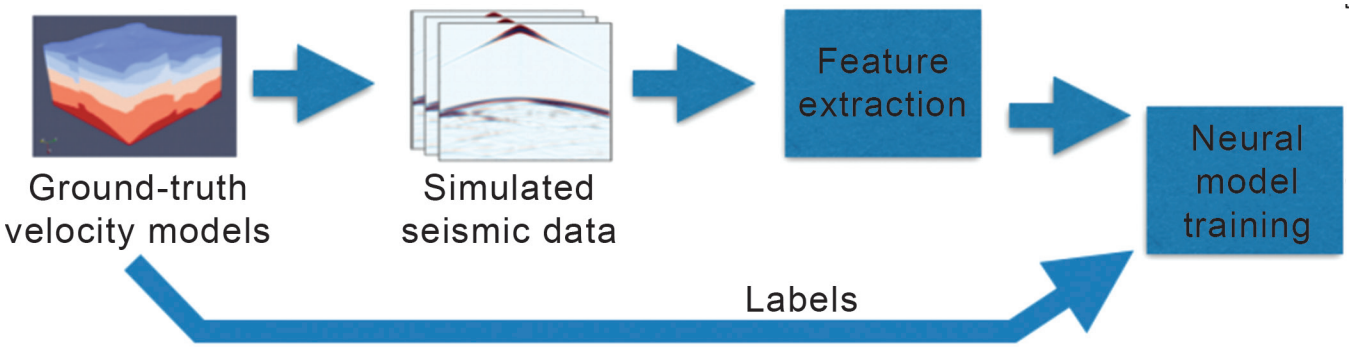

Figure 3. Training workflow.

and, from them, the labels that represent the experiment — for instance, models with faults or salt bodies. In the second step, a modeling step produces the seismic data. For the sake of simplicity and brevity, only acoustic approximation of the wave equation is used. The third step extracts features from the seismic data. The purpose of this step is twofold: it reduces the amount of data used for training, which therefore alleviates stress on the computing resources, and it helps the training to focus on certain aspects within the data that are relevant for the experiment. This also helps with the accuracy and convergence of the training task. Once we have extracted the features, the actual deep-learning process starts. Our workflow is fully parametrical, from the velocity generation to the feature extraction; therefore, the richness of the experiments is comprehensive in terms of variety of velocity models, acquisition geometries, etc.

The inference workflow (Figure 4) is where new models are predicted when exposed to unseen or new data. In our particular context of using synthetic data, it starts in the same fashion as the training workflow: models and data are generated, then those data — that have not been used for training — are presented to the predicting model that reconstructs a velocity model. Since we generate the testing data following the mentioned steps, calculations of accuracy of the model are straightforward.

Semblance as a feature for machine learning

Feature extraction is a key step in our workflow because it can greatly improve the training of the DNN by providing it with the most relevant data for learning. Our machine-learning platform, GeoDNN, is capable of handling diverse network architectures and data, but given that we desire to learn a velocity tomography operator from the data, we perform velocity analysis and provide semblance panels for different common-midpoint (CMP) locations as the input feature. To calculate the semblance panel for a given midpoint, we first apply a normal moveout (NMO) correction to a CMP gather using the second-order traveltime equation:

$$t_{NMO}^2 = t_0^2 + \frac{x^2}{V_{NMO}^2}, \tag{8}$$

where t_{NMO} is the calculated NMO traveltime, t_0 is the zero-offset travel time, x is the offset, and V_{NMO} is the NMO velocity. By choosing a trial V_{NMO}, we can then perform an NMO correction on the gather resulting in an NMO-corrected image $q[j, k]$, where j and k are the corrected NMO time and offset sample indices, respectively (following the notation of Luo and Hale, 2012). Semblance is then calculated by stacking along the offset index and smoothing along the time index of $q[j, k]$. This can be expressed mathematically as

$$s[i] = \frac{\sum_{j=i-M}^{i+M} \left(\sum_{k=0}^{N-1} q[j,k]\right)^2}{N \sum_{j=i-M}^{i+M} \sum_{k=0}^{N-1} q[j,k]^2}, \tag{9}$$

where $s[i]$ is the output semblance at the output time sample i, N is the total number of offset samples, and M is a parameter that defines the length of the time-smoothing window of length $2M + 1$ centered at i. Additionally, we calculate weighting functions that are applied to semblance panels that emphasize terms in the semblance calculation that are most sensitive to changes in velocity (Luo and Hale, 2012). While in the semblance calculation we assume for now only second-order moveout (i.e., the traditional NMO equation), we have the capability of using higher-order terms in the traveltime equation, allowing for greater accuracy at far offsets (Yilmaz, 2001).

Given that we provide semblance panels for multiple CMP locations, this input feature ends up having three dimensions, making a cube. Figures 5 and 6 show us two things about this feature space. First, for the particular model that the semblance cube represents, we have a high percentage of zero-entry and low-value parameters. This is true for many models on which we perform semblance cubes, which means there is an opportunity to sparsify the parameter space. Second, the events in the semblance cube space have patterns that relate to the reflector position and velocity. Humans can interpret some of these patterns (such as distinct energy spikes/clusters, which correspond to sharp unpolluted reflection events). Other patterns that are mixed or smeared across the semblance space can imply nonuniqueness, which is much more difficult to derive a model approximation from. The advantage of using machine learning is that we are able to leverage the DNN's ability to learn from a multitude of examples to discover complex patterns that would otherwise be very expensive and difficult to learn and utilize. Using these patterns, we can learn a mapping from the semblance space to the velocity-model space. Alternative methods, such as inversion, can be very expensive because the mapping between these spaces uses wave propagation (or some other approximation) as the forward operator. Other methods would try to linearize the forward operator or follow some more simplistic methodology, such as picking velocities from peak amplitudes in the semblance cube. All of these methods need to be repeated for each model of interest. Using a DNN methodology, we need to train only once, after

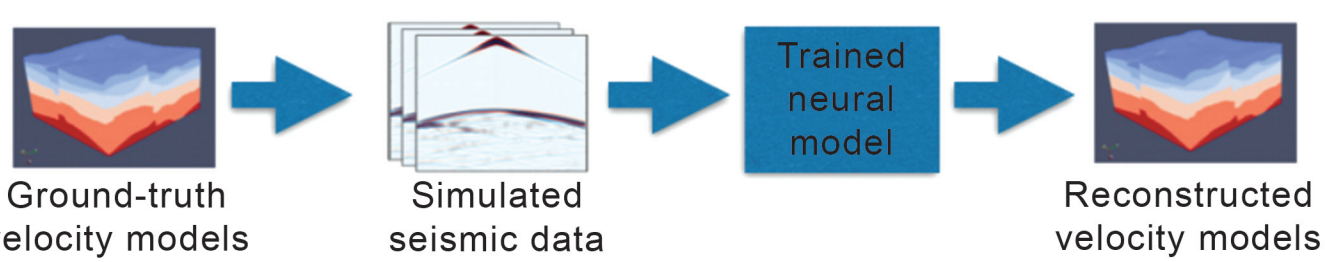

Figure 4. Inference (testing) workflow.

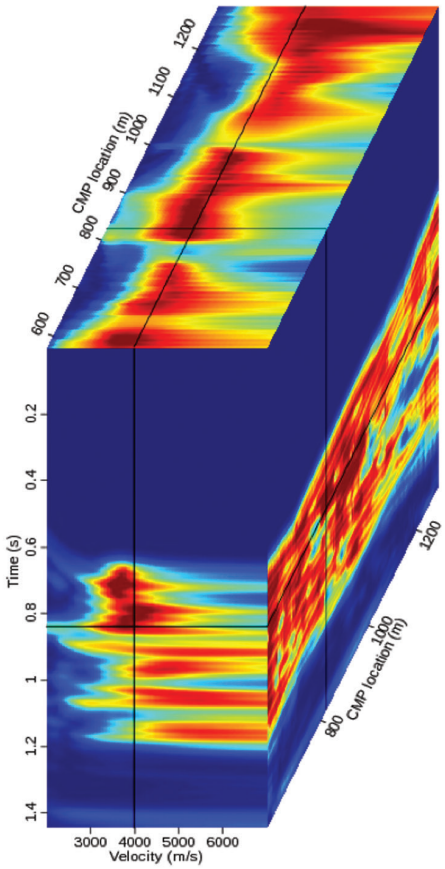

Figure 5. A calculated semblance cube used as an input feature for deep learning. The front face of the cube (with axes of zero-offset time and velocity) shows the semblance panel for a particular CMP location used in traditional velocity analysis. The side face of the cube (with axes of CMP location and zero-offset time) shows the calculated semblance for a particular velocity for all CMP locations and time. Note the spatially coherent structure of the semblance in the cube.

which subsequent model approximations can be found from their corresponding semblance plots at negligible cost.

As stated, the nature of using semblance as a feature input is that there are patterns that have meaning in relation to the velocity model, some of which are trivial to explain while others are more complex. DNN architectures fundamentally learn patterns in the feature space using stencils whose dimensions are predetermined. We believe that the geophysics-based transformation that the semblance cube represents makes it a good choice as an input feature for deep learning for tomographic velocity estimation, especially for DNN architectures that can leverage the patterns that are found in that space.

Implementation and results

We generated thousands of random 2D velocity models with up to four faults in them, dip angle, and position. Our models had between three and eight layers each, with velocities varying from 2000 to 4000 [m/s], with layer velocity increasing with depth. These models were 140 × 180 grid points at the sampling used for wave equation solving. The raw data collected was reduced to a semblance feature set that can fit in multiple NVIDIA K80 GPGPU memory.

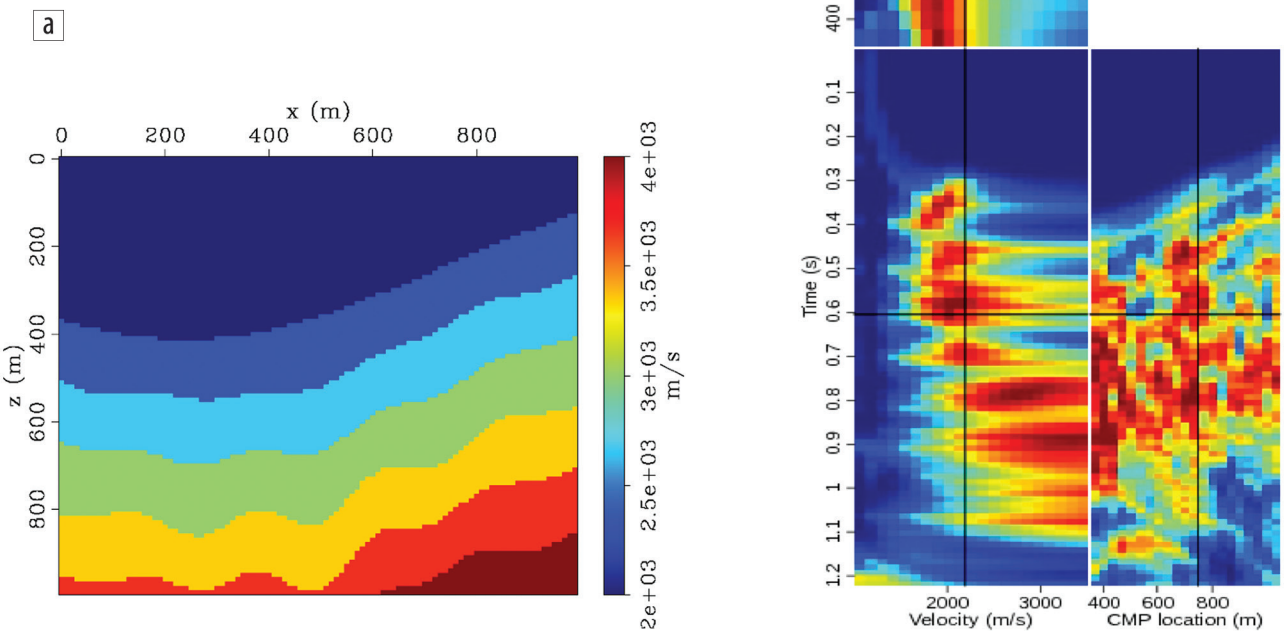

Figure 6. An example of a calculated semblance cube for a seven-layer model. Note that the traditional velocity analysis panel of the semblance cube shown in (b) captures each of the layer interfaces. Also, note that the right panel (axes of CMP location and zero-offset time) of the semblance cube qualitatively gives the approximate structure of the velocity model shown in (a).

We trained the proposed DNN using a training set composed of tens of thousands of velocity models and tested the tomography results with a testing set of thousands of velocity models.

Experiments type I. The output of the DNN is a continuous-valued image, whereas the ground-truth velocity-model images are composed from a discrete number of values, each corresponding to a unique velocity value. Therefore, we have applied a postprocessing image segmentation (Szeliski, 2010) stage to each reconstructed velocity model using two methods: (1) *k*-means segmentation, which uses the ground-truth number of layers to cluster all pixels into the correct number of segments, and (2) *k*-means segmentation with eight segments (layers) for all velocity models. (In a real application, the number of segments are unknown but can be controlled by the domain expert.) The visual quality of each segmented image was compared against the ground-truth velocity model (i.e., test example label) using the structural similarity image metric (SSIM) developed by Wang et al. (2004). The SSIM metric is computed using three image features that mostly influence the human visual system: structure, contrast, and luminance. Given two images, the SSIM formula computes a continuous number between 0.0 and 1.0, where 1.0 corresponds to identical images and 0.0 corresponds to complete visually dissimilar images. The SSIM metric is considered more coherent to human judgment than the mean squared error (MSE) metric for image comparisons. The averaged SSIM over thousands of test velocity models is 0.8717 for *k*-means with the correct number of segments and 0.8603 for *k*-means with eight segments for all images, which clearly indicates very high similarity to the ground-truth velocity models. In Figure 7, we provide examples of the reconstructed velocity models with varying numbers of layers, which demonstrates the high visual quality of the reconstructed images. We have observed that the reconstruction network tends to smooth faults (third row of Figure 7). Further improvements for accurate fault reconstruction are left for future research.

Experiments type II. In this set of experiments, the labels and reconstructed models are of a continuous value (not a binary or multiclass classification process) that represents velocity. Some

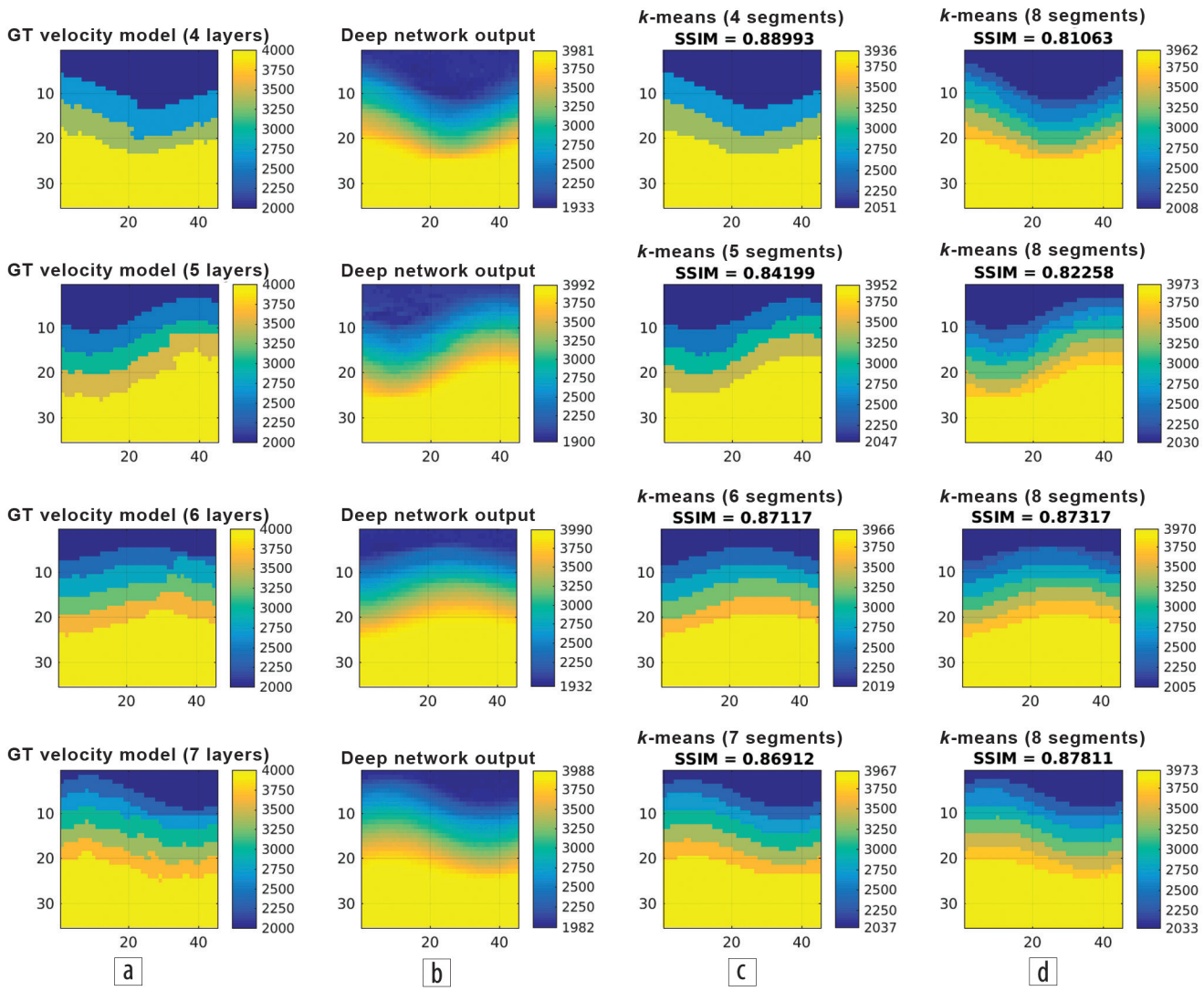

Figure 7. Tomography with four- to seven-layered velocity models. Each row represents a different experiment with unique number of layers. Column (a) is ground truth; column (b) is DNN output (prediction), column (c) us segmented prediction (from column b) image using *k*-means and the correct number of layers; and column (d) is segmented prediction (from column b) image using *k*-means and eight layers.

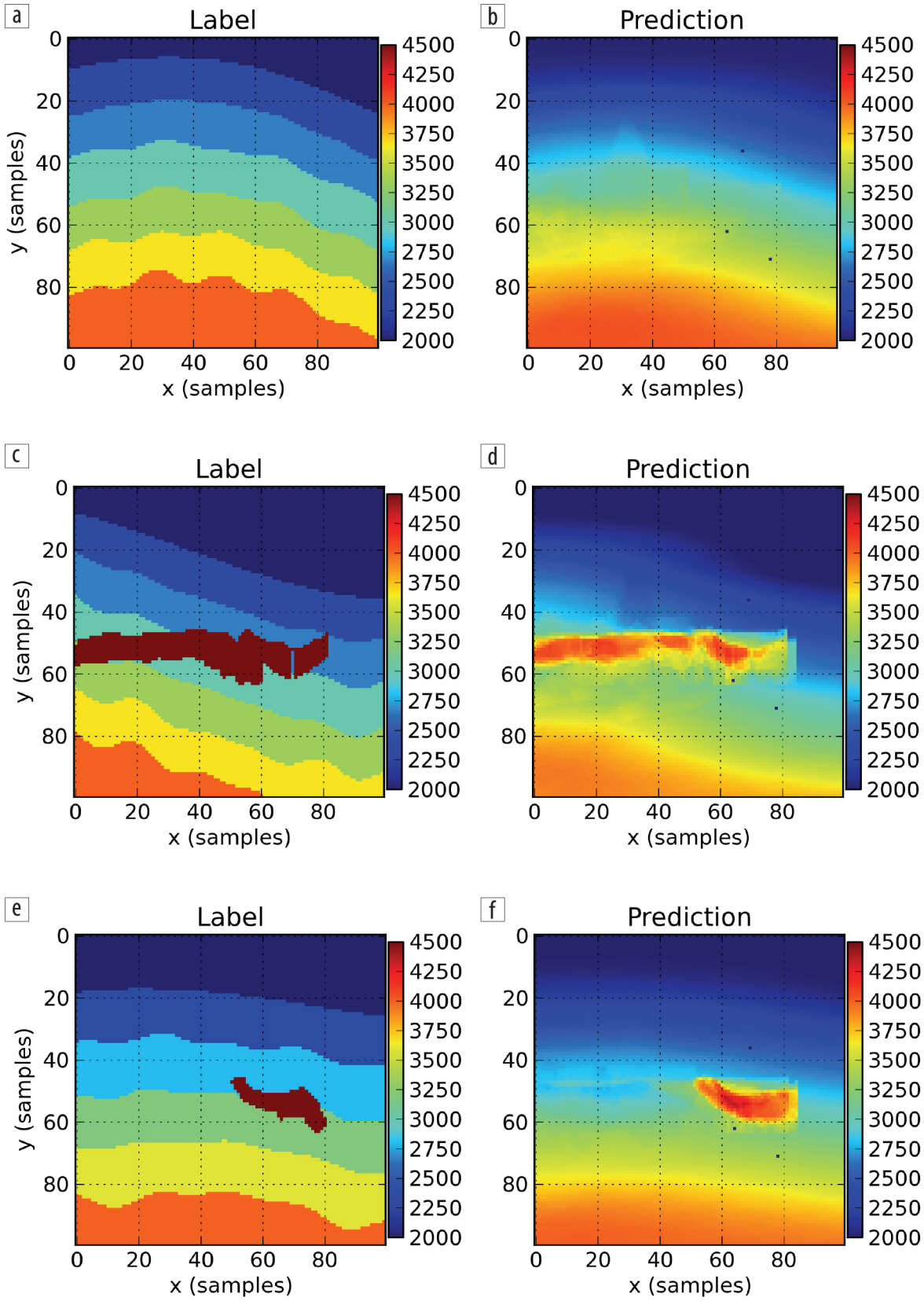

Figure 8. In panel (a), a layered model is presented as ground truth (label). Panel (b) shows the prediction generated with the trained model. That model was training with data that only contains layered models with different number of layers and velocity per layer. The predicted model closely resembles the label in structure and actual velocity. Panels (c–f) are results for a different trained model. This one has been trained with a data set that also contains salt bodies, which has been handcrafted.

of the labels and models include salt bodies along with three to seven layers. The evaluation metrics are R^2 score (coefficient of determination) and SSIM as described above. R^2 score measures the total variation of the outcomes provided by the model. It is interpreted as the goodness of the model fitting. The values can be negative, and the optimal value is 1.

In terms of prediction accuracy, for experiments that only contain layers (Figure 8 top), the R^2 score is 0.8124 and SSIM is 0.8939, which is comparable to the results obtained for the set of experiments of type I. For the experiments with and without salt bodies (Figure 8, mid and bottom), the R^2 score is 0.5536 and the SSIM is 0.8101. As expected, the task of predicting a model with salt bodies is more difficult, and therefore the performance is lower than the task of predicting plain velocity models. Also, the variability of the salt body shape and location is more difficult to learn with the size of training data set that was used. This explains why the R^2 score is more affected (Figure 9) than the SSIM for this case. In any case, the overall performance trend is positive; the salt bodies are located properly; and the surrounding structure resembles the labels in direction and velocity value.

Conclusions

The concept introduced here has enormous potential. Just from the computing perspective, the results were computed in a single computing node, not in a cluster. Further, the input to the machine-learning system has only one type of feature, and the number of training samples is tens of thousands, which barely falls in the big-data category. All these elements, when scaled up, can contribute to improve the accuracy and generalization of the predicting models.

In terms of performance, the results clearly show that most major elements of the expected models are identified and placed in their corresponding location. The results are stable and reliable, as the R^2 score informs. They are also accurate, as the SSIM metric computed for all the experiments is above 0.8 (out of 1.0).

Future work will take three main directions. The first is to extend the method to 3D data, which from the machine-learning perspective is mainly a scalability problem rather than a fundamental one. The second line of work must deal with testing our method with real data. To that end, the most sensitive part is the feature extraction step. Finally, our method can be combined with FWI, where our models can play the role of an accelerator. TLE

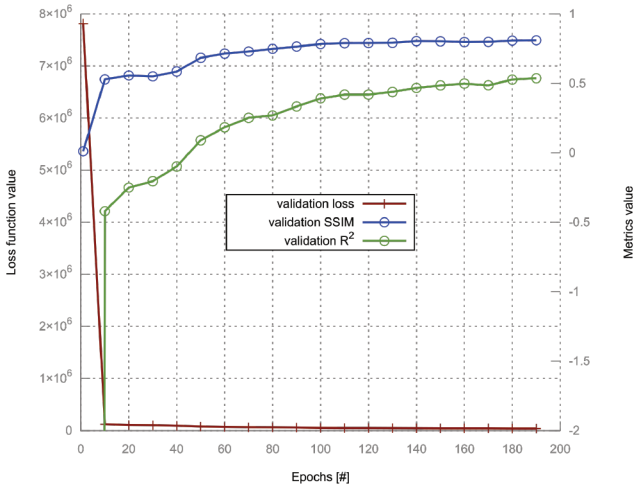

Figure 9. Validation loss function value (red curve related to left Y axis) informs about the effectiveness of the training process. Green and blue curves (related to the right Y axis) represent the evolution of the accuracy metrics during training with salt bodies. The horizontal axis represents training epochs.

Acknowledgments

The authors thank Shell International Exploration & Production Inc. and MIT for permission to publish this paper.

Corresponding author: mauricio.araya@shell.com

References

Addison, V., 2016, Artificial intelligence takes shape in oil and gas sector: EPmag, https://www.epmag.com/artificial-intelligence-takes-shape-oil-gas-sector-846041, accessed 5 December 2017.

Adler, A., D. Boublil, and M. Zibulevsky, 2017, Block-based compressed sensing of images via deep learning: Presented at the 19th IEEE International Workshop on Multimedia Signal Processing.

Araya-Polo, M., T. Dahlke, C. Frogner, C. Zhang, T. Poggio, and D. Hohl, 2017, Automated fault detection without seismic processing: The Leading Edge, **36**, no. 3, 208–214, https://doi.org/10.1190/tle36030208.1.

Biondi, B., 2006, 3D seismic imaging: SEG, https://doi.org/10.1190/1.9781560801689.

BizTech, 2014, High performance computing's role in energy exploration, https://biztechmagazine.com/article/2014/07/hpc%E2%80%99s-role-energy-exploration, accessed 5 December 2017.

Bougher, B., and F. Herrmann, 2016, AVA classification as an unsupervised machine-learning problem: 86th Annual International Meeting, SEG, Expanded Abstracts, 553–556, https://doi.org/10.1190/segam2016-13874419.1.

Burger, H. C., C. J. Schuler, and S. Harmeling, 2012, Image denoising: Can plain neural networks compete with BM3D?: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), https://doi.org/10.1109/CVPR.2012.6247952.

Dahlke, T., M. Araya-Polo, C. Zhang, and C. Frogner, 2016, Predicting geological features in 3D seismic data: Presented at 3D Deep Learning Workshop.

Dong, C., C. C. Loy, K. He, and X. Tang, 2016, Image super-resolution using deep convolutional networks: IEEE Transactions on Pattern Analysis and Machine Intelligence, **38**, no. 2, 295–307, https://doi.org/10.1109/TPAMI.2015.2439281.

Frogner, C., C. Zhang, H. Mobahi, M. Araya-Polo, and T. A. Poggio, 2015, Learning with a Wasserstein loss, in C. Cortes, D. D. Lee, M. Sugiyama, and R. Garnett, eds., Proceedings of the 28th International Conference on Neural Information Processing Systems — Volume 2: MIT Press.

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

Guillen, P., 2015, Supervised learning to detect salt body: 85th Annual International Meeting, SEG, 1826–1829, https://doi.org/10.1190/segam2015-5931401.1.

Hale, D., 2013, Methods to compute fault images, extract fault surfaces, and estimate fault throws from 3d seismic images: Geophysics, **78**, no. 2, O33–O43, https://doi.org/10.1190/geo2012-0331.1.
Hale, D., 2012, Fault surfaces and fault throws from 3d seismic images: 82nd Annual International Meeting, SEG, 1–6, https://doi.org/10.1190/segam2012-0734.1.
Hastie, T., R. Tibshirani, and J. Friedman, 2001, The elements of statistical learning: Springer New York Inc.
Hornik, K., M. Stinchcombe, and H. White, 1989, Multilayer feedforward networks are universal approximators: Neural Networks, **2**, no. 5, 359–366, https://doi.org/10.1016/0893-6080(89)90020-8.
LeCun, Y., Y. Bengio, and G. Hinton, 2015, Deep learning: Nature, **521**, no. 7553, 436–444, https://doi.org/10.1038/nature14539.
Luo, S., and D. Hale, 2012, Velocity analysis using weighted semblance: Geophysics, **77**, no. 2, U15–U22, https://doi.org/10.1190/geo2011-0034.1.
Nath, S. K., S. Chakroborty, S. K. Singh, and N. Ganguly, 1999, Velocity inversion in cross-hole seismic tomography by counterpropagation neural network, genetic algorithm and evolutionary programming techniques: Geophysical Journal International, **138**, no. 1, 108–124, https://doi.org/10.1046/j.1365-246x.1999.00835.x.
Röth, G., and A. Tarantola, 1994, Neural networks and inversion of seismic data: Journal of Geophysical Research, **99**, no. B4, 6753–6768, https://doi.org/10.1029/93JB01563.
Szeliski, R., 2010, Computer vision: Algorithms and applications: Springer Science & Business Media.
van der Baan, M., and C. Jutten, 2000, Neural networks in geophysical applications: Geophysics, **65**, no. 4, 1032–1047, https://doi.org/10.1190/1.1444797.
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, https://doi.org/10.1109/TIP.2003.819861.
Wang, G., 2016, A perspective on deep imaging: IEEE Access, **4**, 8914–8924, https://doi.org/10.1109/ACCESS.2016.2624938.
Würfl, T., F. C. Ghesu, V. Christlein, and A. Maier, 2016, Deep learning computed tomography: Medical Image Computing and Computer-Assisted Intervention, 432-440, https://doi.org/10.1007/978-3-319-46726-9_50.
Xie, J., L. Xu, and E. Chen, 2012, Image denoising and inpainting with deep neural networks: Proceedings of the 25th International Conference on Neural Information Processing Systems — Volume 1, 341–349.
Yilmaz, Ö., 2001, Seismic data analysis: SEG.
Zhang, C., C. Frogner, M. Araya-Polo, and D. Hohl, 2014, Machine-learning based automated fault detection in seismic traces: 76th Conference and Exhibition, EAGE, Extended Abstracts, https://doi.org/10.3997/2214-4609.20141500.