Synthetic seismic data generation for automated AI-based procedures with an example application to high-resolution interpretation

Fernando Vizeu[1], Joao Zambrini[1], Anne-Laure Tertois[2], Bruno de Albuquerque da Graça e Costa[3], André Queiroz Fernandes[3], and Anat Canning[4]

https://doi.org/10.1190/tle41060392.1

Abstract

This paper discusses the generation of synthetic 3D seismic data for training neural networks to solve a variety of seismic processing, interpretation, and inversion tasks. Using synthetic data is a way to address the shortage of seismic data, which are required for solving problems with machine learning techniques. Synthetic data are built via a simulation process that is based on a mathematical representation of the physics of the problem. In other words, using synthetic data is an indirect way to teach neural networks about the physics of the problem. An important incentive for using synthetic data to solve problems with artificial intelligence methods is that with real seismic data the ground truth is always unknown. When generating synthetic seismic data, we first build the model and then calculate the data, so the answer (model) is always known and always exact. We describe a methodology for generating on-the-fly simulated postmigration (1D modeling) synthetic data in 3D, which are high resolution and look similar to real data. A wide range of models is covered by generating an unlimited number of data examples. The synthetic data are built from impedance models that are constructed through geostatistical simulation of real well logs. With geostatistical simulation, we can describe various geologic variance models in 3D and obtain realistic images. To cover a broad range of scenarios, we need to generalize the seismic data story by randomly perturbing many parameters including structures, conformity styles, dip-strike directions, variograms, measured input logs, frequencies, phase spectra, etc.

Introduction

In recent years, there has been growing interest in the use of machine learning (ML) technologies for processing and interpreting seismic data. Many procedures that traditionally have been performed using deterministic methods and algorithms can be effectively replaced by neural networks and other artificial intelligence (AI) methodologies, improving simplicity, efficiency, and automation. Examples include seismic horizon and fault interpretation using convolutional neural networks (CNNs) (Huang et al., 2017; Di et al., 2018; Guo et al., 2018; Lowell and Paton, 2018; Wu and Zhang, 2018; Zhao and Mukhopadhyay, 2018; Wu et al., 2018, 2019; Bi et al., 2021), salt classification (Waldeland et al., 2018; Shi et al., 2019), seismic migration (Freitas et al., 2020; Zhang and Gao, 2022), and others. This data training approach replaces the algorithmic representation of physical procedures as well as other traditional signal processing methods. Instead of formulating a mathematical method to solve specific data problems that can be described theoretically, we use many data examples to teach a system about the nature of the problem and to provide a solution. The advantage is huge. Problems that are too complex to be described theoretically can be solved through learning from data examples, and several steps can be combined into one. Procedures that require substantial human resources can be automated with this approach. Seismic interpretation is a good example of a task that requires many highly skilled work hours. It can be replaced, to a large extent, by ML technologies.

The main challenge with ML technologies is data examples. With seismic data, there is a limit to how much data exist. The data sets are large and cumbersome, and many data sets are proprietary and therefore not available to most researchers. In addition, large amounts of data are required to solve any problem with ML. The data need to cover a wide range of geophysical problems, and the correct answer to the problem at hand needs to be available for each example data set. In other words, we need a large set of input/true-output pairs. However, with seismic data, the true answer is in the subsurface and is mostly unknown. The alternative to real data is synthetic data, and many in the geophysical community use this solution (Wu and Hale, 2016; Geng et al., 2019; Wu et al., 2019, 2020; Bi et al., 2021). Using synthetic data is also a common practice in other disciplines that apply ML technologies. The idea that real geophysical problems can be solved by training neural networks with synthetic data has already been demonstrated by Canning et al. (2017), Geng et al. (2019), Wu et al. (2019, 2020), and Bi et al. (2021).

Using synthetic data is not only a way to address a shortage of data, it is also a way to introduce physics into the solution. When generating synthetic data, the physics of the problem at hand is introduced because synthetic data are always generated by some simulation process that mimics a physical process or mathematical idea. Consequently, the algorithm used to generate the synthetic data controls what the AI system learns, and in this indirect way, teaches it the physics/mathematics of the problem. Furthermore, using synthetic data where the ground truth is known removes the human bias that is too often associated with input/interpreted output pairs created from real data.

The need for a good method for generating realistic synthetic data emerged when we began developing an automated high-resolution interpretation technology in order to automatically interpret internal horizons within a given layer (Vizeu et al., 2021). Here, we mainly discuss a method for generating the synthetic

[1]Emerson Automation Solutions, Rio de Janeiro, Brazil. E-mail: fernando.vizeu@emerson.com; joao.rodrigueszambrini@emerson.com.
[2]Emerson Automation Solutions, Paris, France. E-mail: anne-laure.tertois@emerson.com.
[3]Petrobras, Rio de Janeiro, Brazil. E-mail: brunoagc@petrobras.com.br; andreqf@petrobras.com.br.
[4]Emerson Automation Solutions, Herzelia, Israel. E-mail: anat.canning@gmail.com.

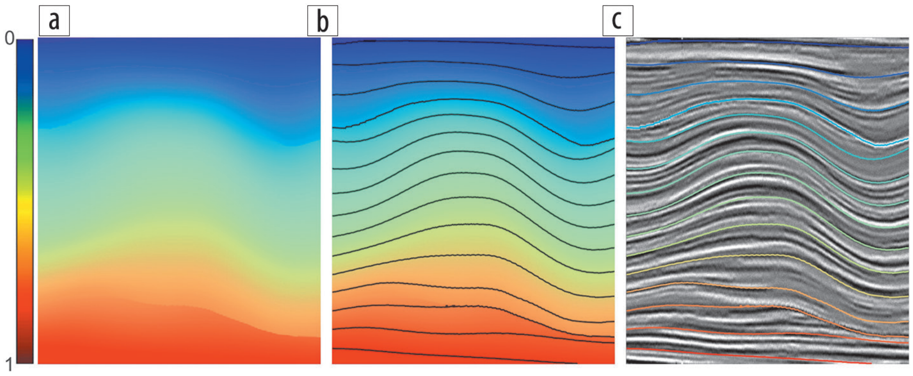

Figure 1. Example of an RGT cross section. (a) RGT. The colors represent RGT values. (b) Horizons extracted from RGT at constant RGT intervals. (c) Extracted horizons displayed over the seismic section.

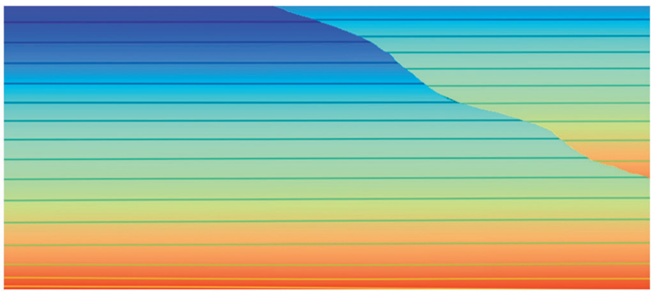

Figure 2. Example of RGT for a reverse fault. The complexity of this scenario makes the RGT not increase monotonically with depth.

seismic data for training a neural network. The methodology can be used to solve a wide range of seismic-related problems in seismic data processing, interpretation, and inversion. Note that the synthetic data generation is based on 1D convolution modeling and does not involve 3D wave equation simulations. It is therefore relevant to postmigration challenges when all seismic events are positioned correctly in 3D space and the amplitude is proportional to the reflection coefficient. One-dimensional convolution modeling is a good approximation for this state. Seismic interpretation is one example of a postmigration problem. However, a similar rationale can be used to generate synthetic pseudomigrated gathers involving Zoeppritz equations for the prestack amplitudes. Such synthetic gathers can be used to train an AI system to solve local gather processing tasks including prestack amplitude inversion, noise removal, etc.

Relative geologic time as a structural framework

A structural framework is often required before generating synthetic data. Relative geologic time (RGT) is a fantastic way to represent structures on a regular 3D grid because it can provide detailed vertical resolution. RGT refers to the geologic assertion that a reflector or layer is related to a specific geologic event that occurred at a specific geologic time. The layer is deformed in depth through geologic events such as folding and faulting that occurred over geologic time. Figure 1 shows an example of an RGT section where colors represent the geologic time (RGT value). The values are relative and are often normalized between 0 and 1. Tracking a constant value on an RGT grid will yield the layer geometry (Figure 1).

The concept of RGT was introduced by Wheeler (1958), more recently developed by Stark (2003, 2004), and built into a full mathematical framework by Mallet (2004) and Moyen et al. (2004). It is important to note that RGT is a simplified representation of the geology, and it works well in relatively simple situations. Complex geology involving extensive faulting with large throws and heavy layer deformation may need a more comprehensive description, as formulated by Mallet (2004) in the UVT framework. The UVT transform is a 3D mapping of the present-day geology to a pseudodepositional space, where the effects of faulting and folding have been removed. The U and V components represent the paleocoordinates where each particle of sediment was deposited, and the T component is the RGT. The UVT transform is more general. It enables precise modeling of geologic properties in the depositional space and their consistent transfer to the present-day geologic space with a minimum amount of deformation, regardless of the complexity of the geologic structure. A reverse fault is an example of a complication because RGT does not increase monotonously with depth (Figure 2). Therefore, a full UVT parameterization is required to describe complex structural geometry. Still, for many seismic problems, RGT is sufficient to provide the required structural framework.

In the context of this paper, RGT is used to build synthetic data sets for training neural networks. To generate nontrivial RGT models, we first build the geometric framework. A variety of software packages are available that can build a 3D geometric structure from a set of horizons and faults. We use SKUA-GOCAD (Gringarten et al., 2008) for this because it can deal with complex geologies. The basic set of structural frameworks is built from real interpreted horizons and then filled with RGT values using a variety of conformity styles per layer (e.g., proportional, toplap, baselap, etc.) (Figure 3) to generate the basic set of RGT data sets. The conformities are controlled by the geometry of the top and base of each layer, while the internal conformity style varies between the layers in a random fashion.

Synthetic data for training neural networks

To train a neural network with synthetic data, there is a need to generate data that look like real data and have real data characteristics. One of the main drawbacks in using synthetic data is that they look artificial. They are often very smooth and too good for solving real data problems with neural networks. The most common solution (Geng et al., 2019; Bi et al., 2021) is to add noise to the data, which is normally some kind of random noise. However, random noise has limited effect on CNNs because it acts somewhat like a filter. Moreover, random noise is added to the seismic data and not included in the model.

We have developed a method for generating synthetic seismic data for training neural networks that provides a realistic data set on one hand and great variability on the other. A central concept is the use of real impedance well logs to obtain realistic subsurface variability and geostatistical simulation methods to control the overall variance. Randomization on all parameters is another key aspect of this methodology. Thus, most perturbations are done for the impedance data, and seismic traces are calculated from that, instead of perturbing the seismic traces themselves. This enables the generation of realistic seismic data as well as variability in seismic parameters such as wavelet, frequency, phase, etc.

The following steps describe the method for building the set of synthetic data sets. It begins in 2D:

1) Generate a basic set of RGT data sets as described earlier.
2) Interpolate real impedance well logs using sequential Gaussian simulation (SGS) (Goovaerts, 1997) with a variety of randomly selected variogram models (Figure 4).
3) Deform the impedance data to match the selected basic RGT (Figure 5).
4) Add small structural deformations. Apply the deformations to both the impedance section and the corresponding RGT section (Figure 6).
   a) Small faults with randomly selected fault density, throw, and direction
   b) Small undulations with randomly selected parameters such as amplitude, curvature, etc.
   c) Possibly other features such as channels
   d) Standard deformations such as rotation, stretching, squeezing, etc.
5) Perform RGT normalization.
6) Calculate reflectivity from the impedance section.
7) Build realistic wavelets with randomly selected parameters (frequency and phase spectra).
8) Convolve the wavelet with the reflectivity traces to generate 2D synthetic data sets.
9) Add random noise.

This methodology enables great variability in the data by varying structures, wavelets, and stratigraphic variance, as well as realistic-looking data sets. Some examples are displayed in Figure 7, showing how realistic the data look and the variability that can be easily achieved in data character. In fact, an infinite number of different data sets that capture much of the seismic data characteristics can be generated this way.

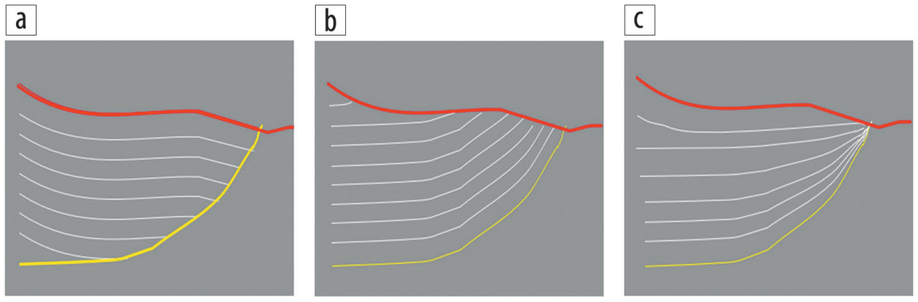

Figure 3. Conformity styles used in the synthetic data set. (a) Baselap. (b) Toplap. (c) Proportional.

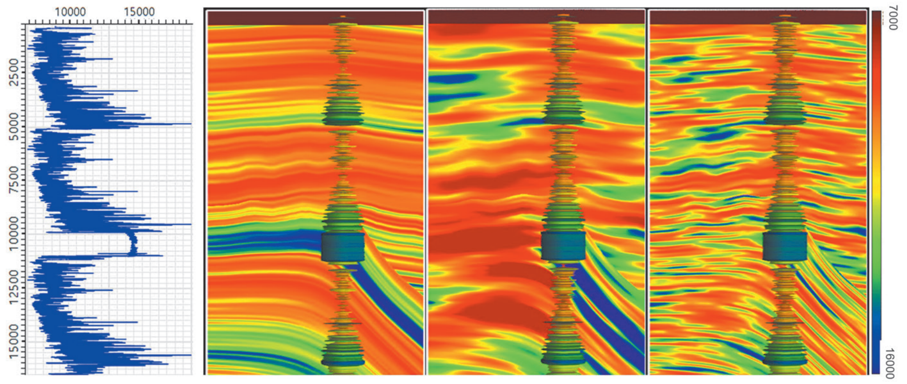

Figure 4. Examples of impedance data sets generated from well logs using geostatistical simulations (SGS). The input log is displayed on the left and is also overlaid on the impedance sections. Three different impedance sections generated by SGS are displayed, and all are deformed to the same structural framework (controlled by RGT) (see Figure 5) but produced using different 3D variogram models.

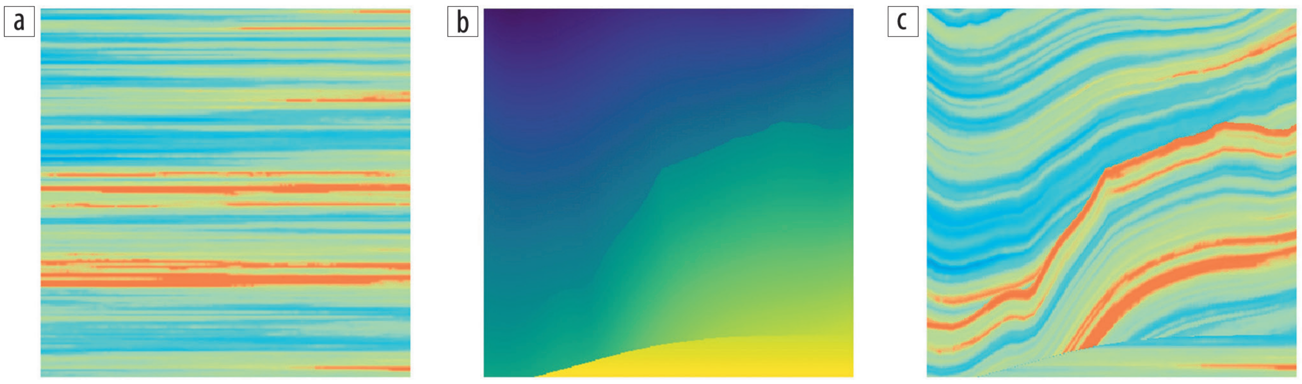

Figure 5. Deforming impedance section according to RGT. (a) An initial impedance section generated by SGS. (b) A random RGT section. (c) Structurally deformed impedance according to the RGT section.

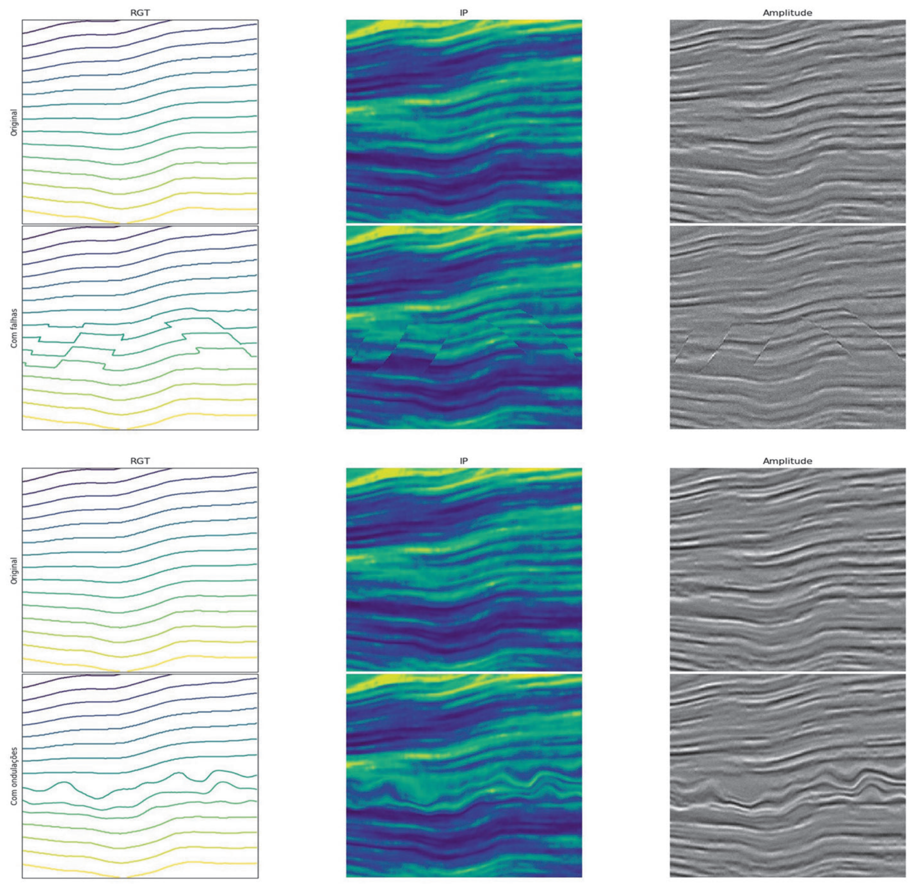

Figure 6. Two examples of additional secondary deformations. The top example is of small faults and the bottom example is of small random undulations. In each example, the top row is the original state and the bottom row is the deformed state. RGT contours are on the left. Impedance is in the center. Resulting seismic is on the right.

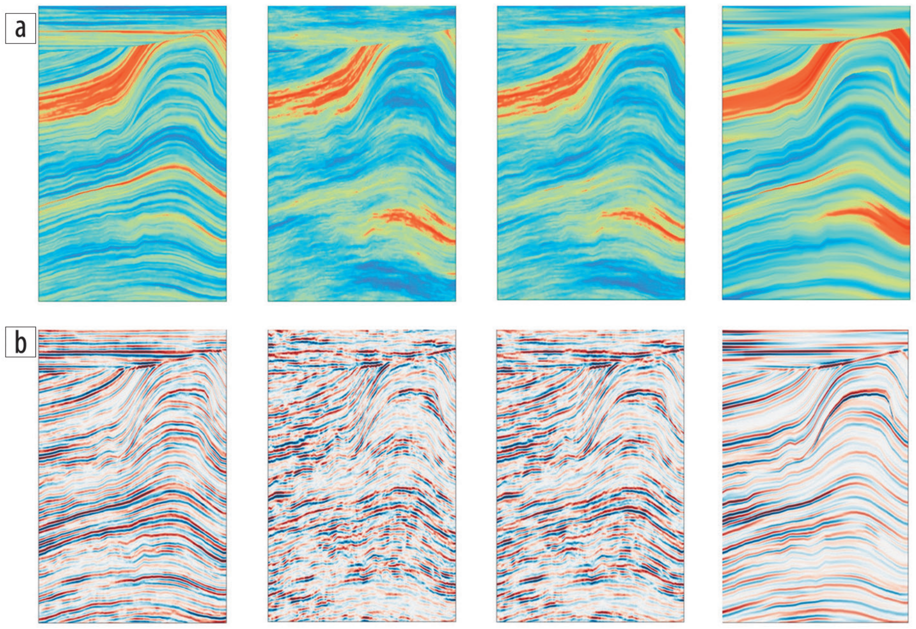

Figure 7. Examples of data variability (within a single RGT section). (a) Impedance. (b) Corresponding seismic amplitude data. The difference between these examples is in the variograms. Each example is created with the same structural model but with a different variogram.

In this workflow, the variability is in the impedance model, and the seismic data mainly reflect that. This is in contradiction to workflows that perturb the seismic data. This provides more realistic data because it follows the natural process. For example, if we want to introduce a fault into the synthetic data, we can do it in two ways: (1) generate the fault in the impedance data and create synthetic data from the faulted input or (2) reverse the order and create synthetic data from the unfaulted impedance and then introduce the fault into the seismic data. The first route may produce reflections from the fault plane, while the second route will result in an artificial-looking fault because the seismic data will be torn in an artificial way. This is illustrated in Figure 8. As explained, our training procedure controls the training by balancing the ratio of examples with specific features. In this case, we used 10% of the examples with faults applied to impedance and 10% of examples with faults applied to seismic.So, 80% of our examples do not include faults. This is another way to solve specific problems by balancing the training set.

Next, the 2D data sets are extended to 3D. The process begins with the 2D impedance section (the first line). We add more lines to build a 3D cube. The second line is a deformed copy of the first line. The deformation is very small between lines and is done by setting a random path at the center of the first line and shifting the data up, down, or sideways according to the random path (one step for each line). Figure 9 illustrates this concept. A similar method can be used for other types of deformation (e.g., stretching and squeezing) to build the 3D data. Finally, we randomly rotate the 3D volume to also provide variability in the dip-strike directions.

This methodology can provide an infinite number of seismic data examples. We build the synthetic data on the fly in parallel to the CNN training procedure, randomly selecting patches of data from the larger original set of RGT data sets. The synchronization between data generation and the training process optimizes the computation time. In addition, we statistically control the process by controlling the percentage of examples with a specific characteristic. For example, we can prefer a larger horizontal variogram by ensuring more data sets with large horizontal variograms will be generated, or we can provide higher fault density by adding a higher percentage of faulted examples.

The synthetic data generation process is very efficient, and there is no need to store or manage training data. Another important advantage is the flexibility to add new types of geologic scenarios to the process without the need to start over. We continue to train the network with new examples and continuously see improvements in performance.

This method for generating synthetic data to train neural networks is quite general. We demonstrate the method here by using it to train a network that provides high-resolution interpretation of internal horizons. It does so by building input-output example pairs in the form of seismic-stack/RGT pairs (similar to Geng et al. [2019], Wu et al. [2020], and Bi et al. [2021]). Similar processes can be used to generate data for other objectives. For example, one can use a similar methodology to generate synthetic gathers corrected for normal moveout by using simulated P- and S-impedance pairs combined with Zoeppritz equations for the amplitude. Such gathers can be used as training data for prestack inversion. The main factor that enables these data sets to mimic real data so well is the geostatistical simulations, which allow control over the variance of the data. Based on this methodology for generating realistic synthetic data for training, many seismic processing, interpretation, and inversion challenges can be solved with AI techniques.

Constructing a CNN that can be pretrained with synthetic data so it can be applied successfully to a wide range of real data sets requires generalization of the seismic character. Data character is partly represented with the variograms but is also associated with the wavelet. To obtain seismic amplitude from the acoustic impedance data, random wavelets are created for each impedance example and are convolved with it. The wavelets are generated by randomly perturbing the frequency content, the decay as a function of frequency, and the phase rotation (Figure 10).

QC: Network evolution

The described methodology was implemented to generate seismic data for automated AI-based interpretation of internal horizons on stacked seismic data volumes. We followed Geng et al. (2019) to build a U-Net, where input is a seismic patch

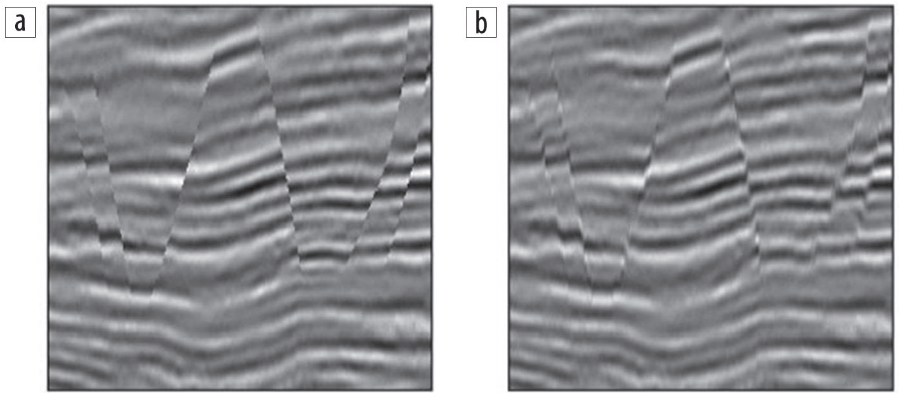

Figure 8. Introducing a fault into the data and comparing two workflows. (a) Introducing the faults into the impedance first. (b) Introducing the faults at the end of the workflow after convolution with a wavelet.

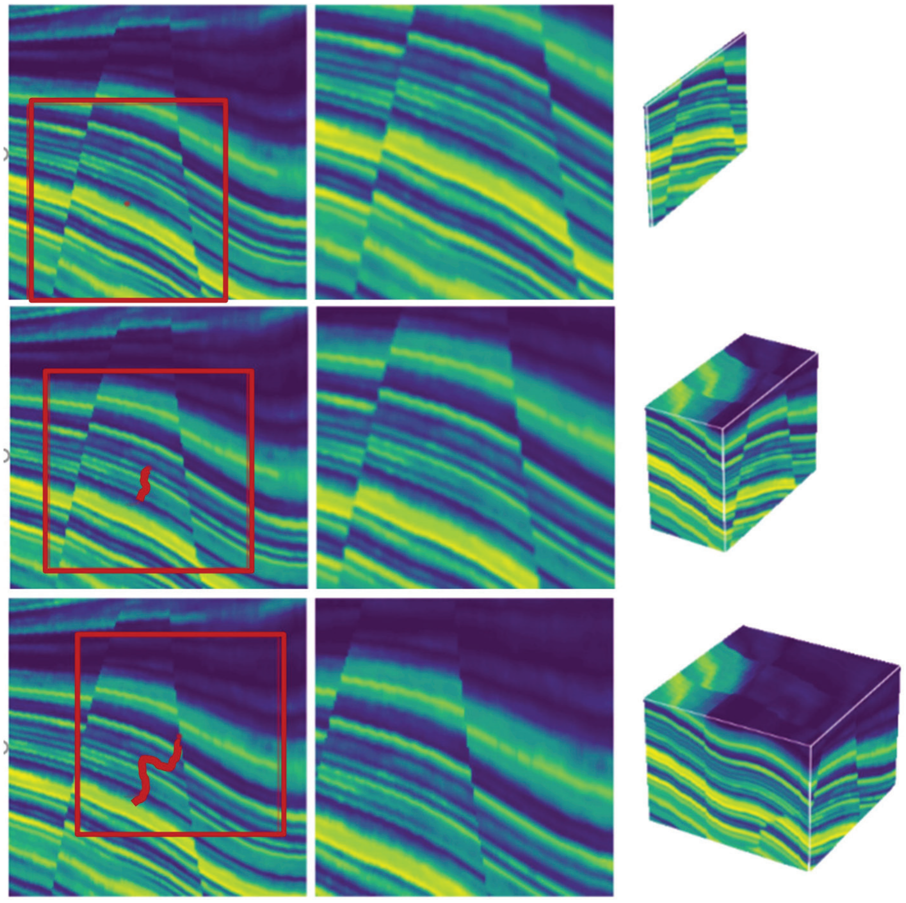
Figure 9. Building a 3D impedance data set from a 2D section using a random 3D path. The process starts at the top images and progresses downward. The random path is marked with the red curve.

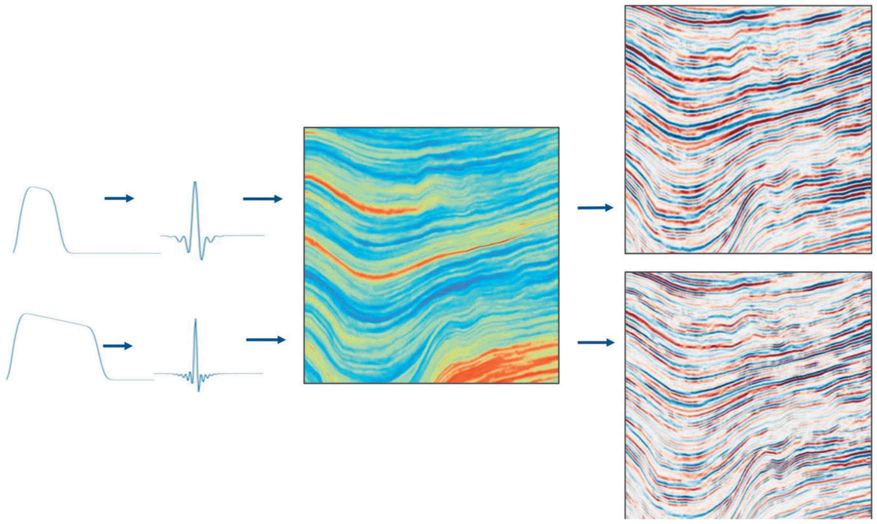

Figure 10. Wavelet generation and random selection of wavelet parameters. Example shows two seismic data sets with different wavelets calculated from the same impedance data set. Random frequency contents are generated in the frequency domain with varying bandwidth, decay, and phase.

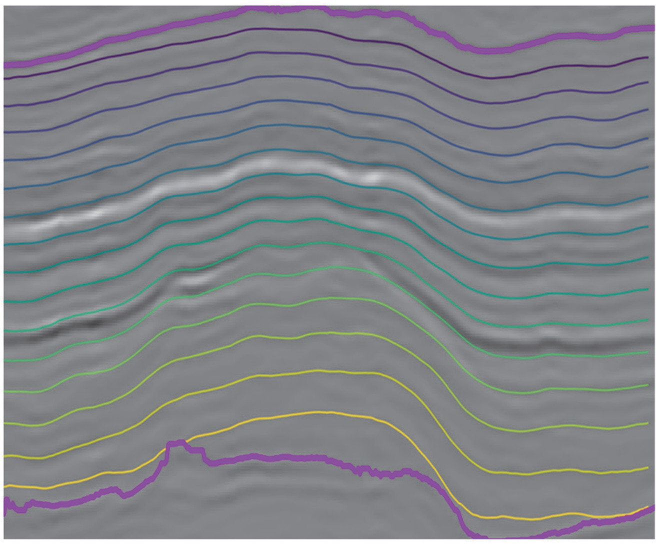

Figure 11. Example of automatic interpretation of internal horizons on real data. Contours of RGT isovalues are computed automatically and displayed over the seismic data. The network was pretrained with synthetic examples. Note that the pink lines mark the top and base of the layer to be interpreted and were provided by the interpreter. They are not computed automatically by the process.

and the desired output (known as tag in the popular terminology of AI) is RGT. We then extracted horizons by following constant RGT surfaces (isovalues) and displayed the resulting interpretation over the seismic data. Interpreters are accustomed to seeing the structural information this way, so it is a useful display, even though the RGT itself contains more information at a much higher resolution. Figure 11 shows a field data example. The network was trained with synthetic data sets, generated using the methodology described here, and applied to this real data. We concentrated on internal horizon interpretation within a given layer. Focusing the automated interpretation task this way leads to excellent results.

QC is an important part of the strategy. QC is done with validation data sets, which were generated with the earlier noted methodology but not used for training. The performance of the technique is monitored by following the evolution of the process as training goes on and more data sets enter the training cycle. Training starts with simple models, with complexity added over time. We could clearly see the effect of adding new training data to the network's ability to resolve complex scenarios. New model types were added when we noticed structures that did not resolve well. In other words, we continuously enriched the data model space and, in that way, improved the network's performance. Figure 12 shows three examples of network evolution and QC, monitoring the evolution of the network. The left column displays the input data sets, and the last column shows the true answer. The center columns show the network evolution from left to right, displaying the results as more training is done and additional data complications are introduced into the process. During the training process, we added two attributes to the architecture, replacing two of the three channels of the input image. We used instantaneous phase and envelope as the two attributes. The green line marks the time step where this change occurred, showing immediate improvements in the result. The next event in the evolution of the network is the introduction of faults into the data example generation process. This event is marked by the blue line, which again shows how it affects the ability of the process to interpret faults. It is easy to add complexities, and it does not complicate the program engineering. With this strategy, more training means that more data sets are used and the network continuously improves with time.

Conclusions

Using realistic synthetic seismic data can be a powerful mechanism for training ML algorithms to solve seismic-data-related problems, but they are not trivial to produce. The data need to be realistic, cover a broad range of scenarios, and reproduce the geologic features that characterize the problems we aim to solve. We have developed an innovative way to generate such data sets and demonstrated its effectiveness in one example implementation — automatically interpreting internal horizons in seismic data. We have shown a real data example to verify the concept that synthetic data can be used to train for real data problems and that pretraining with a broad set of seismic examples can be effective, even without training the specific data set.

The method produces large variability in the seismic character and mimics a wide range of geologic features and seismic scenarios. Variability is achieved by using geostatistical simulations and randomly selected variogram models, as well as randomization of many parameters that control the workflow.

The CNN can be fed by these data sets using on-the-fly data generation, enabling a very efficient parallel processing workflow. Because the data are created on the fly, there is no need to save them on disk, and there is no need for complex data management and tagging strategies. This method also enables continuous improvement of the results with time, as new information is added into the synthetic data creation system. A similar but more dedicated strategy can be applied to solve some specific geologic problems in specific data sets by statistically controlling the parameters of the synthetic data generation. TLE

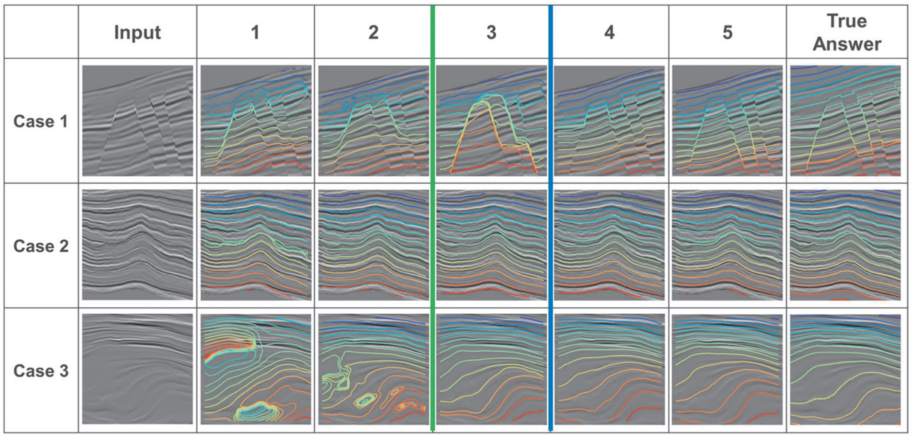

Figure 12. Three examples of network evolution and QC monitoring the evolution of the network. The input column displays the input data sets, and the last column displays the true answer. The center columns show the network evolution with time from left to right. Attributes were introduced after step 2 (green line), and faults were introduced after step 3 (blue line).

Acknowledgments

We would like to thank Petrobras for their contribution and advice. Petrobras sponsored this project and helped turn the idea into reality.

Data and materials availability

Data associated with this research are confidential and cannot be released.

Corresponding author: anat.canning@gmail.com

References

Bi, Z., X. Wu, Z. Geng, and H. Li, 2021, Deep relative geologic time: A deep learning method for simultaneously interpreting 3-D seismic horizons and faults: Journal of Geophysical Research: Solid Earth, **126**, no. 9, https://doi.org/10.1029/2021JB021882.

Canning, A., D. Moulière-Reiser, Y. Weiss, A. Malkin, E. Philip, N. Grinberg, A. Teitel, M. Reznikov, and V. Yehezkel, 2017, Neural networks approach to spectral enhancement: 87th Annual International Meeting, SEG, Expanded Abstracts, 4283–4286, https://doi.org/10.1190/segam2017-17751158.1.

Di, H., M. Shafiq, and G. AlRegib, 2018, Patch-level MLP classification for improved fault detection: 88th Annual International Meeting, SEG, Expanded Abstracts, 2211–2215, https://doi.org/10.1190/segam2018-2996921.1.

Freitas, R. S. M., C. H. S. Barbosa, G. M. Guerra, A. L. G. A. Coutinho, and F. A. Rochinha, 2020, An encoder-decoder deep surrogate for reverse time migration in seismic imaging under uncertainty: arXiv 2006.09550v1.

Geng, Z., X. Wu, Y. Shi, and S. Fomel, 2019, Relative geologic time estimation using a deep convolutional neural network: 89th Annual International Meeting, SEG, Expanded Abstracts, 2238–2242, https://doi.org/10.1190/segam2019-3214459.1.

Goovaerts, P., 1997, Geostatistics for natural resources evaluation: Oxford University Press.

Gringarten, E. J., G. B. Arpat, M. A. Haouesse, A. Dutranois, L. Deny, S. Jayr, A.-L. Tertois, J.-L. Mallet, A. Bernal, and L. X. Nghiem, 2008, New grids for robust reservoir modeling: Annual Technical Conference and Exhibition, SPE, Extended Abstracts, https://doi.org/10.2118/116649-MS.

Guo, B., L. Li, and Y. Luo, 2018, A new method for automatic seismic fault detection using convolutional neural network: 88th Annual International Meeting, SEG, Expanded Abstracts, 1951–1955, https://doi.org/10.1190/segam2018-2995894.1.

Huang, L., X. Dong, and T. E. Clee, 2017, A scalable deep learning platform for identifying geologic features from seismic attributes: The Leading Edge, **36**, no. 3, 249–256, https://doi.org/10.1190/tle36030249.1.

Lowell, J., and G. Paton, 2018, Application of deep learning for seismic horizon interpretation: 88th Annual International Meeting, SEG, Expanded Abstracts, 1976–1980, https://doi.org/10.1190/segam2018-2998176.1.

Mallet J.-L., 2004, Space-time mathematical framework for sedimentary geology: Mathematical Geology, **36**, https://doi.org/10.1023/B:MATG.0000016228.75495.7c.

Moyen, R., J.-L. Mallet, T. Frank, B. Leflon, and J.-J. Royer, 2004, 3D-parameterization of the 3D geological space — The GeoChron model: 9th Conference on the Mathematics of Oil Recovery, EAGE, Extended Abstracts, https://doi.org/10.3997/2214-4609-pdb.9.A004.

Shi, Y., X. Wu, and S. Fomel, 2019, SaltSeg: Automatic 3D salt segmentation using a deep convolutional neural network: Interpretation, **7**, no. 3, SE113–SE122, https://doi.org/10.1190/INT-2018-0235.1.

Stark, T. J., 2004, Relative geologic time (age) volumes — Relating every seismic sample to a geologically reasonable horizon: The Leading Edge, **23**, no. 9, 928–932, https://doi.org/10.1190/1.1803505.

Stark, T. J., 2003, Unwrapping instantaneous phase to generate a relative geologic time volume: 73rd Annual International Meeting, SEG, Expanded Abstracts, 1707–1710, https://doi.org/10.1190/1.1844072.

Vizeu, F., J. Zambrini, B. Costa, A. Fernandes, T. Dall'Oglio, and A. Canning, 2021, Automatic interpretation of internal horizons using CNN: Presented at the Advances in Seismic Interpretation Workshop, SEG.

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, https://doi.org/10.1190/tle37070529.1.

Wheeler, H. E., 1958, Time-stratigraphy: AAPG Bulletin, **42**, no. 5, 1047–1063, https://doi.org/10.1306/0BDA5AF2-16BD-11D7-8645000102C1865D.

Wu, H., and B. Zhang, 2018, A deep convolutional encoder-decoder neural network in assisting seismic horizon tracking: arXiv 1804.06814.

Wu, X., and D. Hale, 2016, 3D seismic image processing for faults: Geophysics, **81**, no. 2, IM1–IM11, https://doi.org/10.1190/geo2015-0380.1.

Wu, X., Z. Geng, Y. Shi, N. Pham, S. Fomel, and G. Caumon, 2020, Building realistic structure models to train convolutional neural networks for seismic structural interpretation: Geophysics, **85**, no. 4, WA27–WA39, https://doi.org/10.1190/geo2019-0375.1.

Wu, X., L. Liang, Y. Shi, and S. Fomel, 2019, FaultSeg3D: Using synthetic data sets to train an end-to-end convolutional neural network for 3D seismic fault segmentation: Geophysics, **84**, no. 3, IM35–IM45, https://doi.org/10.1190/geo2018-0646.1.

Wu, X., Y. Shi, S. Fomel, and L. Liang, 2018, Convolutional neural networks for fault interpretation in seismic images: 88th Annual International Meeting, SEG, Expanded Abstracts, 1946–1950, https://doi.org/10.1190/segam2018-2995341.1.

Zhang, W., and J. Gao, 2022, Deep-learning full-waveform inversion using seismic migration images: IEEE Transactions on Geoscience and Remote Sensing, **60**, https://doi.org/10.1109/TGRS.2021.3062688.

Zhao, T., and P. Mukhopadhyay, 2018, A fault-detection workflow using deep learning and image processing: 88th Annual International Meeting, SEG, Expanded Abstracts, 1966–1970, https://doi.org/10.1190/segam2018-2997005.1.