Neural Networks for Seismic Data Inversion

Cassio M. Oishi, Fabio Vinícius Goes Amaral, Hugo Leonardo França, William Hideki Nakata, Diego Alecsander de Aguiar, Gilmar Francisco de Oliveira Santos, Débora de Oliveira Medeiros, Gabriel Spadon, José Fernando Rodrigues Jr., José Mario Martínez, Lúcio Tunes Santos and Djalma Manoel Soares Filho

1 São Paulo State University, Brazil
2 University of São Paulo, Brazil
3 University of Campinas, Brazil
4 Petrobras, Brazil

(Communicated to MIIR on 7 July 2021)

Study Group: Sixth Brazilian Study Group with Industry, March 22-26, 2021

Communicated by: Cassio Oishi, José Fernando Rodrigues Jr.

Industrial Partner: Petrobras

Presenter: Lúcio Tunes Santos, José Mario Martínez, Djalma Manoel Soares Filho

Team Members: Cassio M. Oishi, Gabriel Spadon, José Fernando Rodrigues Jr., Giovanna Castello de Andrade, Fabricio Simeoni de Sousa, Hugo França, Debora Medeiro, Hugo de Oliveira, Amanda Figur, Pedro Peixoto, Felipe Silva, Fabio Amaral, William Hideki, Diego Aguiar, Gilmar Francisco, André Calcagniti Padilha, João Victor de Melo, Valério Nogueira, Paulo Roberto Mendes dos Santos, Filipe Cardoso de Oliveira, Anderson Ervino Schwertner.

Industrial Sector: Data Analysis

Tools: Optimization

Key Words: Seismic inversion, neural networks, GAN, ReGENN.

MSC2020 Codes: 86A22, 68T07
Summary

Building a velocity model is essential in seismic exploration and is used at all stages, including acquisition, processing and interpretation of seismic data. Reconstructing a subsurface image from seismic wavefields recorded at the surface (seismograms) requires accurate knowledge of the propagation velocities between the recording location and the image location at depth. Estimation of velocity models can also be used as initial models to recursively generate high-resolution velocity models through optimization algorithms. Machine learning is a field of artificial intelligence that uses computational techniques to give systems the ability to learn from a large volume of data. In particular, neural networks have been developed to reconstruct subsurface parameters, i.e., the acoustic (compressional) wave velocity model, directly from raw seismic data. Using this principle as a starting point we will use two neural network approaches to solve the problem, where a GAN neural network and a ReGENN network will be used.

1 Introduction

The problem proposed by Petrobras consists in building a neural network to construct a velocity model for the case of a two-dimensional acoustic medium, consisting of homogeneous layers separated by horizontal interfaces, from the supplied pairs (seismograms, models).

The problem is based on obtaining the velocity model from the seismograms that are obtained by the reflection seismic method, which is used to detect changes in impedance in the subsurface through an active seismic source (explosives, air gun, vibroseis). After generating these seismic pulses, the reflected, refracted, and diffracted waves from the impedance contrasts in the subsurface are recorded, generating seismograms.

Some methods used to extract velocity fields are transit-time tomography, migration velocity analysis, and full waveform inversion. All of these methods are local optimization and require a good initial model to converge to the target model, so we believe that neural networks can work to generate an initial model.

2 Problem Modeling

The problem proposes the use of neural networks so that given an input seismogram, the network will be able to tell which is the corresponding velocity model. To do this, a dataset with 2000 pairs (seismogram, model) was initially provided for training, and 50 were used for testing, where the seismogram was generated from the corresponding model.

To solve this problem the group adopted two neural network based approaches. The approaches chosen to solve the data inversion problem were: using an Generative Adversary Network (GAN), and also a Recurrent Graph Evolution Neural Network (ReGENN), where the performance of each one in the reconstruction of the velocity models will be measured.
2.1 Generative Adversarial Network (GAN)

A Generative Adversary Network (GAN) is a machine learning model that seeks to solve problems using two adversarial neural networks called generator and discriminator.

Given a data set, training is done by having the networks compete against each other. The generator aims to learn features of the set so that it can generate new false inputs, while the discriminator is a net that aims to tell whether a particular input in the set is true or created by the generator.

Let $x \in X$ be a matrix representing the real velocity model (from the training or test set) that generates the seismogram $s$ and $G(s) = \tilde{x}$ the model created by the generator network, we use the binary cross entropy function ($1 - \text{BCE}$) between the discriminator’s output and the expected result, i.e., 0 for $x$ and 1 for $\tilde{x}$, where

$$\text{BCE}(x, y) = \frac{1}{N} \sum_{i=1}^{N} l_i,$$

$$l_i = y_i \log(x_i) + (1 - y_i) \log(1 - x_i),$$

and $N$ is the batch size.

For the generator, we not only want it to deceive the discriminator, but we need to maintain the physical properties of the generated model. Thus, the objective function is given by the sum of the adversary function ($1 - \text{BCE}(D(\tilde{x}), 0)$) and the Mean Square Error (MSE) between the generated model and the model that generated the input seismogram $\text{MSE}(\tilde{x}, x)$, where

$$\text{MSE}(\tilde{x}, x) = \frac{1}{M} \sum_{i=1}^{M} (x_i - y_i)^2,$$

and $M$ represents the number of pixels in the model images.

As stated earlier, the network is divided into two modules. The discriminator is a network that takes as input a matrix $x \in M^{N \times 1 \times l \times c}$ and has as output a value $d \in [0, 1]$.

For the generator network, an autoencoder structure was used, in which an encoder structure generates, from the input, a latent vector with the main characteristics of the input, and a decoder structure uses this latent vector to generate the expected output image. For this the ReGENN framework was used.
2.2 Recurrent Graph Evolution Neural Network (ReGENN)

It is proposed to treat the problem with the ReGENN network, designed for multivariate and multi-task problems. Since the proposed problem is a univariate, single-task variation, the network could be applied by changing the input representation, and making adjustments culminated in a new architecture, as detailed below.

ReGENN is an automatic encoder based on linear and nonlinear components that work together to provide regressions from patterns learned in supervised training. Its architecture is described in Figure 2. The linear part is the autoregression, and the nonlinear component is composed of an autoencoder fed by a pair of Graph Soft Evolution (GSE) layers. The GSE stands for a graph theory-based learning representation layer that enhances the encoding and decoding processes by establishing a graph that describes the interactions between the different variables of the problem. In the case of the Petrobras problem, there is only one variable, so the graph became unnecessary, resulting in a new architecture - more details below.

The original ReGENN data representation is illustrated in Figure 2. Three-dimensional tensors were defined with axes describing \( \text{Sample} \times \text{Time} \times \text{Depth} \). In the case, the seismogram dimension refers to the individual slices defined by the \( \text{Time} \times \text{Depth} \) dimensions, i.e., one of the dimensions enumerates the number of samples per batch.

To handle the proposed task, a regression task modality, ReGENN was adjusted to a univariate configuration. This adjustment made it unnecessary to use the GSE com-
Figure 2. Original Recurrent Graph Evolution Neural Network (ReGENN) data representation, which predicts multiple variables over time and, consequently, multiple 3D tensors. For the proposed problem, the data representation was adapted to a single variable, defining a single 3D input tensor for processing. Figure reproduced from the work of Spadon et al.[2].

Figure 3. ReGENN data flow diagram; the network has a linear component working in parallel with a nonlinear one. The linear component has a single feed-forward layer, similar to a Multi-layer Perceptron network. The non-linear component has an autoencoder and two GSE layers. Figure reproduced from the work of Spadon et al.[2].

ponent, which requires multiple input variables - in the case of Petrobras, depth is the only variable. As a result, the final network has an autoencoder structure separated into two approaches, one linear represented by a monolayer autoregressor, and the other non-linear represented by the multilayer autoencoder. The adaptations allowed the problem to be successfully addressed as demonstrated in the experiments.
3 Results

In the intention of reducing the training time of the GAN, the generator network was preconditioned with a training for model generation without considering the discriminator. With this transfer learning strategy, based on [3], the initial outputs of the generator will not be completely random. As the goal of preconditioning is only to get a better initial state, the training was done with a few epochs, generating the results in Figure 4. After the training step, some results for the test set are presented in Figure 5.

![Figure 4](image1.png)  
**Figure 4.** Example of the generator output after preconditioning.

![Figure 5](image2.png)  
**Figure 5.** Samples from the test set after training.

During GAN training, the network was able to learn the behavior of the horizontal lines in the ensemble. Figure 6 shows an overview of the Mean Absolute Error (MAE) of the network on the entire ensemble.
Figure 6. Mean Absolute Error of the training samples.

Figure 7 shows the evolution of the errors of the GAN during training. It can be seen that the discriminating network has a very low error compared to the generating network. This means that the network is having no problem, most of the time, in distinguishing a real model from the generator model.

Figure 7. Error of the GAN modules during training.

Table 2 presents the results of the adapted and applied ReGENN on the four datasets provided for the proposed challenge. Dataset 1 consists of seismograms with increasing velocities, dataset 2 of seismograms with alternating velocities, dataset 3 of seismograms with noise, while dataset 4 presents varied multilayer seismograms with different ground layer structures.

| Type | Error | Dataset 1 | Dataset 2 | Dataset 3 | Dataset 4 |
|------|-------|-----------|-----------|-----------|-----------|
| Train | MAE   | 0.0003    | 0.0004    | 0.0008    | 0.0504    |
|       | RMSE  | 0.0006    | 0.0007    | 0.0009    | 0.0887    |
| Test  | MAE   | 0.0004    | 0.0005    | 0.0007    | 0.0669    |
|       | RMSE  | 0.0007    | 0.0008    | 0.0010    | 0.1167    |

Table 2: Autoencoder (ReGENN) results for the datasets

In the table, the datasets were ordered from easiest to hardest with respect to the complexity of the data and the consequent solving challenge. Similarly, the best results were observed with similar ordering, the best results observed in the least complex dataset. Still, in dataset 4, the most complex and significant, the results are promising - the mean absolute error was around 6.7% and the square error (RMSE) was around 12%.

In Figures 8 to 12, the results are plotted per dataset, from 1 to 4. In each figure the input seismogram, the known model, and the model produced by the neural network are shown. Visually it can be seen that in the first 3 datasets, the model produced is approximately identical to the known model, reflecting an error below 1% for these 3 datasets. Similarly, the images for dataset 4 illustrate the greatest underlying challenge and error in prediction.
Figure 8. Single sample from dataset 1.

Figure 9. Single sample from dataset 2.
Figure 10. Single sample from dataset 3.

Figure 11. Sample 1 from dataset 4.
4 Conclusion

Throughout the week, the Petrobras challenge proved to be interesting. Initially, the problem provided only dataset 1, which was solved successfully and quickly by several teams, showing the great power of the networks in solving the problem, with ReGENN getting a better result than GAN. Then the coordinators provided two new datasets, one with noise and another with alternating speeds, in which case only ReGENN proved to be more robust to solve the problem.

In the final stage of the workshop, the participants themselves generated new seismograms under the supervision of the coordinators, but with ripples between the layers using the Devito library [1], to test ReGENN in a situation closer to the real thing, and what was observed is that the network still has a certain difficulty in recognizing patterns that are not flat, and ends up presenting some noise and little definition.

At the end of the work, it can be verified that the ReGENN network, after being adapted in order to consolidate a new neural network architecture, produced promising results, producing models with high similarity to known models. Finally, considering that the challenge was solved in a very short period of time, it is assumed that there is potential to refine the proposed methodology, achieving even more accurate results. Some tests should be taken into consideration to try to improve the final result, such as hyperparameter adjustments, general module structure, better preconditioning and weighted objective functions.
References

[1] Divito project. https://www.devitoproject.org. Accessed: 24-03-2021.
[2] G. Spadonidesouza, S. Hong, B. Brandoli, S. Matwin, J. F. Rodrigues, and J. Sun. Pay attention to evolution: Time series forecasting with deep graph-evolution learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.
[3] Z. Zhang and Y. Lin. Data-driven seismic waveform inversion: A study on the robustness and generalization. *IEEE Transactions on Geoscience and Remote sensing*, 58(10):6900–6913, 2020.