Seismic data interpolation with generative adversarial networks: the impact of hyperparameters on the learning process and the interpolated solution

Daniel N. Pinheiro*, Gilberto Corso, Mesay G. Gebre, Jaime C. Gonzalez, Carlos A. N. da Costa,
Universidade Federal do Rio Grande do Norte, Natal, RN, 59078-970, Brazil.

This paper was prepared for presentation at the 18th International Congress of the Brazilian Geophysical Society, held in Rio de Janeiro, Brazil, 16-19 October, 2023.

Abstract

We performed a numerical study of the influence of the hyperparameters of the generative adversarial networks on the learning process in an interpolation seismic data application. This study was carried out using a decimated data set as input for the interpolation process and the results are compared with the original one, data set without decimation. From our numerical tests, we have observed that the batch size and the training/test ratio are not decisive hyperparameters, and the loss norm $L1$ produces sharper resolution when compared to the $L2$ norm. In addition, using the size kernel three produced the best interpolation result, and that large value for the hyperparameter λ works better than a low value. Finally, we have found that the more accurate interpolated data set is associated with small learning rates.

Introduction

Seismic data interpolation is an important step in seismic processing when traces are missing in the seismogram and/or when it requires a dense number of traces from a sparse seismic acquisition, where the node spacing is in the order of hundreds of meters. The missing data can be associated with several causes such as the presence of obstacles or no-permit areas during the acquisition and hardware problems with geophones/hydrophones, (Porsani, 1999; Gülünay, 2003). In addition, due to economic reasons, data acquisition can be more sparse than necessary and regular interpolation is needed. This can be the case of time-lapse seismic projects, in which successive acquisitions are done at the same location for monitoring oil reservoirs (Nguyen et al., 2015). In this situation, it is common that the baseline first acquisition is performed with a dense quantity of receivers and the successive acquisitions have fewer receivers, making a regular interpolation necessary (Johnston, 2013).

Seismic interpolation methods can be arranged into three basic groups: Wiener-like filter methods based on data linearity assumptions (Crawley et al., 2005; Porsani, 1999; Naghizadeh and Sacchi, 2009), wave-equation techniques based on seismic wave propagation (Ronen, 1987; Fomel, 2003) and frequency transform methods based on the sparsity of data in the Fourier domain (Sacchi and Ulrych, 1996; Gülünay, 2003). Recently, a new interpolation methodology based on Machine Learning (ML) tools has been introduced, see Fang et al. (2021) using Convolutional Neural Networks (CNN), and Chai et al. (2020); Kaur et al. (2021) employing the Generative Adversarial Network (GAN).

In the last decade, there has been an increasing amount of new studies with applications of ML to geophysical methods and particularly in the oil and gas industry, Dramsch (2020); Sircar et al. (2021). Besides, we call attention to the relevance of ML applied to 2D and 3D seismic inversion problems (Zhang et al., 2022; Wu and Lin, 2019). However, in general, applications of ML to physical problems have to be viewed with caution due to the large number of hidden free parameters inside the ML mathematical apparatus (Géron, 2022). In this way, it is necessary to carefully investigate the ML algorithm and make tests on its hyperparameters. Our study is posed inside this research line: we present a numerical study of the hyperparameters of a GAN-based interpolator, where we mainly have tested the interpolator over a large range of hyperparameters.

Seismic data interpolation can be performed for irregular or regular (or randomly) subsampled data sets. In this study, we are basically concerned with the second scenario, where the data set was decimated in a regular way. However, the main results of our work can be generalized to the random case. In fact, this paper is a methodological study, our focus is on ML hyperparameters employed in the interpolation task. The rest of this manuscript is organized as follows: the methodology is split into three parts, the seismic problem used to perform our numerical experiments, a presentation of the Neural Network used in the ML, and a concise introduction to the ML interpolator. The results concern on the numerical exploration around the tested hyperparameters of the ML interpolator. Finally, we concluded our work by discussing the outcomes of the study.

Methodology

Forward seismic modeling

In this work, to carry out our numerical simulation tests, we used a synthetic data set modeled using a realistic P-wave velocity model based on a typical Brazilian pre-salt reservoir (Mojica and Maciel, 2020) and a density model, built from this velocity model following a polynomial form of the Gardner law (Operto et al., 2015). The data set was modeled by a two-dimensional (2D) acoustic finite-difference modeling (Thorbecke and Draganov, 2011) using Ricker pulse as a wavelet with a 15 Hz peak

frequency. A total number of 361 sources were used with a shot spacing of 50 meters at 10 meters depth. We used 361 receivers placed at 2100 meters depth with a spacing of 50 meters, and hence we have 361 traces for each shot-gather. Figure 1 shows an example of a shot-gather from the modeled data set.

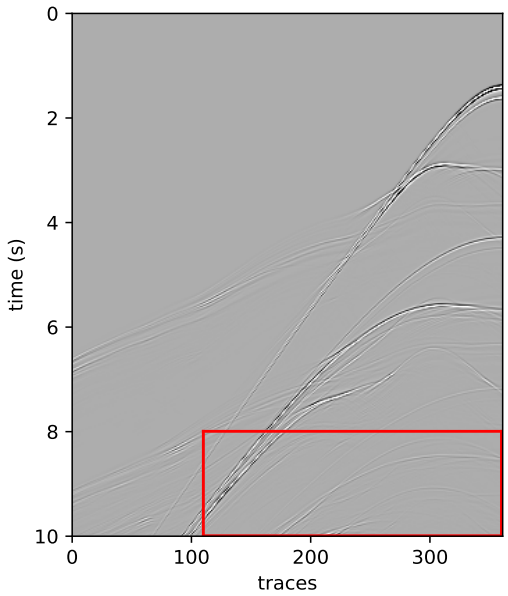


Figure 1: A 2D shot-gather simulated by an acoustic finite-difference modeling. The red rectangle is used to illustrate the interpolator in the numerical tests.

Generative Adversarial Network

The Generative Adversarial Networks are a class of Deep Learning Neural Networks mainly used in the image processing context (Goodfellow et al., 2014). The GAN structure is formed by two complementary neural networks: a generator G, usually a Convolutional Neural Network CNN, and a discriminator D. The generator G is trained to produce outputs that cannot be distinguished from "real" input images, however, the discriminator challenges the generator. The main concept behind GAN architecture comes from game theory, the generator produces images and the discriminator try to falsify the work of the generator, identifying the "fake" images. In this way, both Neural Networks, G and D learn and compete with each other during the training process.

In this study, we employ a convolutional GAN which is associated with the following mapping:

$$G : x, z \rightarrow y \tag{1}$$

where x and y are images and z is a random number. In our work, images x and y are same size seismograms. We start with the main objective of the GAN, that shows the interplay between G and D:

$$\begin{aligned}\mathcal{L}_{cGAN}(G,D) \quad = \quad & \mathbb{E}_{x,y}[\log D(x,y)] + \\ & \mathbb{E}_{x,z}[\log(1 - D(x,G(x,z))],\end{aligned} \tag{2}$$

where $\mathbb{E}$ is the expected value of the expression. In this expression, G tries to minimize its objective against an adversarial D that tries to maximize it. In this way, the balance between G and D is summarized in the expression $G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G,D)$.

The basic problem in the Machine Learning methodology consists in approximating the output y from the produced $G(x,z)$ by minimizing their difference. The choice of the best norm, $||.||_L$, to estimate the difference is a research topic. The L1 norm, that is mostly used in this paper, is done by:

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[||y - G(x,z)||_1]. \tag{3}$$

In the next section we test both L1 and L2 norms that in our model are considered hyperparameters.

An interesting quantifier to quantify the behavior of the discriminator during the training process is the loss function of the discriminator:

$$G_{loss}(G,D) = -\mathbb{E}_{x,z}[\log(D(x,G(x,z)))] \tag{4}$$

the objective of the discriminator is to maximize this function.

The final objective of the GAN methodology contemplates the minimization of both equations (2) and (3). That means, the minimization of the functional:

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G,D) + \lambda \mathcal{L}_{L1}(G). \tag{5}$$

Note that this equation introduces the extra hyperparameter λ which balances the pure minimization of the generator $\mathcal{L}_{L1}(G)$ and the discriminator generator interplay term $\mathcal{L}_{cGAN}$.

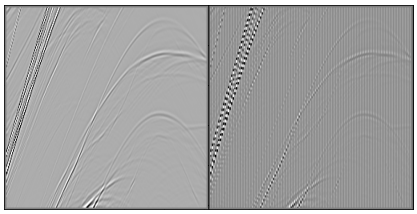
The GAN interpolator

The construction of the interpolator is based on GAN, the architecture used is the same from Isola et al. (2017). The concept of the method is the following, we take seismograms and decimated them to create a training set for the ML. After we use the trained Neural Network produced by the GAN, the GAN interpolator, to create full seismograms from interpolated ones. The decimation process consisted in deleting half of the traces. By methodological reasons, we keep the vector in the seismic matrix, to produce a decimated matrix with the same size of the original matrix, that means, with the same shape (250×250) of the training set.

The data used to train and test the neural network is in the shot-gather domain. The training data preprocessing consisted in normalizing the amplitudes in the range $\{-1,1\}$, furthermore the seismogram was split in 10 windows (250 samples per 250 traces). The splitting process is important to create size windows compatible with the GAN algorithm. Figure 2 shows an example of training set with the original seismogram and the corresponding decimated one with half of the traces. This seismogram window is the one inside the rectangular box of Figure 1.

Results

The GAN technique to interpolate seismic traces depends on hyperparameters that are intrinsic to the GAN. In this study, we numerically explore the main hyperparameters used in ML methodology. Our approach consists in testing the convergence of the loss function for the



(a) True (b) Decimated

Figure 2: Example of a 250×250 window from the original (a) and decimated (b) seismogram. In the decimated window, 50% of the traces have been set to zero.

analyzed hyperparameters and checking the quality of the interpolated seismograms.

We tested the following hyperparameters of the GAN:

- Batch size: 1 and 2;
- Training/test ratio: 0.85, 0.60 and 0.35;
- Factor λ: 10, 100 and 1000;
- Loss norm: L1 and L2;
- Kernel size: 3, 4, 5 and 6;
- Generator learning rate: 1×10^{-4}, 2×10^{-4} and 4×10^{-4};
- Discriminator learning rate: 1×10^{-4}, 2×10^{-4} and 4×10^{-4}.

The total number of tested configurations is $2 \times 3 \times 3 \times 2 \times 4 \times 3 \times 3 = 1296$. We trained the GAN with each combination of these hyperparameters. In the following, we present figures with the mean values and the 95% confidence intervals of the quantifiers $\mathcal{L}_{cGAN}$, G_{loss} and $\mathcal{L}_{Ln}$ depending on the hyperparameter. We analyze each hyperparameter in the next sections.

Batch size

The batch size specifies the amount of samples trained, at the same time, during the neural network learning process. In our case, we consider both neural networks, the generator and the discriminator. In principle, smaller batch sizes are preferred since it implies in smaller neural network sizes and smaller computational times. In Figure 3 we show the evolution of $\mathcal{L}_{cGAN}(G,D)$ during the learning process for two values of batch size. The quantifier $\mathcal{L}_{cGAN}(G,D)$ shows a similar behavior for all batch sizes. We conclude that the batch size is not an important factor in our analysis. In this way, our option was using a batch size equal to one because of computational time economy. Similar result was related by Isola et al. (2017).

Training/Test set ratio

The neural network inside the ML technique needs to be trained and tested. The most usual training test ratio is 70 to 30, that means, from the available data we employ 70% of the data for training and 30% for testing. If the testing set is not large, the confidence of the results weakens.

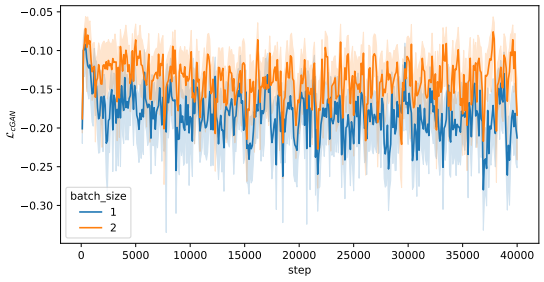


Figure 3: Comparing the evolution of $\mathcal{L}_{cGAN}(G,D)$ during the learning process for batch sizes 1 and 2.

Otherwise, if the training set is not large, the neural network may have not enough data to learn, impairing the ML performance. Figure 4 shows a similar pattern of Figure 3 for three distinct training/test ratios. Our results suggest that the training/test ratio is not decisive in our problem, which means that the GAN can perform well even when the training set is not large.

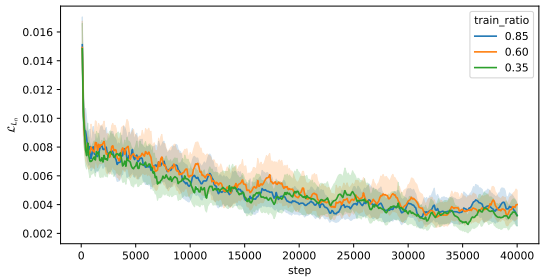


Figure 4: Comparing the evolution of $\mathcal{L}_{Ln}(G)$ during the learning process for the training/test ratios 0.85, 0.60 and 0.35.

λ Factor

The factor λ is a regularizer parameter that weights the trade-off between $\mathcal{L}_{cGAN}$ and $\mathcal{L}_{Ln}$ in the objective function. If $\lambda = 0$, then the generator just try to fool the discriminator by minimizing $\mathcal{L}_{cGAN}$, regardless of the generator output being similar to the target output or not. But if λ tends to infinity, then the neural network behave like a CNN, ignoring the GAN discriminator. In Figure 5 we show the evolution of $\mathcal{L}_{Ln}(G)$ during the learning process for three values of λ, the smaller value of loss function which corresponds to best λ factor is $\lambda = 1000$.

In Figure 6 we present the output of the generator using the factors $\lambda = 10$ (b) and $\lambda = 1000$ (c) for the same decimated input seismogram (a). We observe that $\lambda = 10$ generated noisy artifacts that are absent in $\lambda = 1000$, see the red rectangle in the figure. Considering that $\lambda = 1000$ leads to smaller values of $\mathcal{L}_{Ln}$ and that it generates less noisy artifacts, then we suggest using the largest value.

Loss norm

The choice of the best norm ($\|.\|_1$ or $\|.\|_2$) influences how the generator tries to find the minimum (5) by adjusting its neural network weights. In Figure 7 we show the evolution

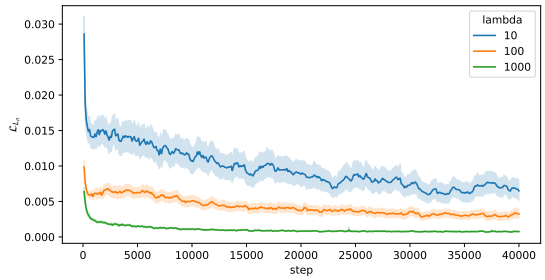
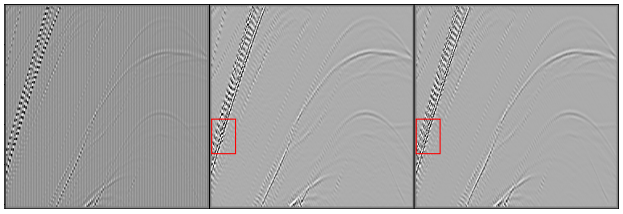


Figure 5: Comparing the evolution of $\mathcal{L}_{L_n}(G)$ during the learning process for the λ factors 10, 100 and 1000.



(a) Input (b) $\lambda = 10$ (c) $\lambda = 1000$

Figure 6: Comparison of the generator output for different λ values. In (a) we present the input decimated seismogram window, in (b) the output for $\lambda = 10$ and in (c) the output for $\lambda = 1000$. The red rectangle emphasizes a noisy output in (b) that is not present in (c).

of $G_{loss}(G,D)$ during the learning process for the norms $L1$ and $L2$. Note that the generator tries to minimize this function, while the discriminator tries to maximize it. The graph shows that the generator is better at fooling the discriminator with norm $L2$ because the G_{loss} values are smaller when we use this norm.

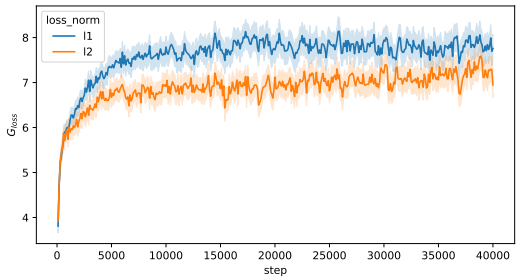
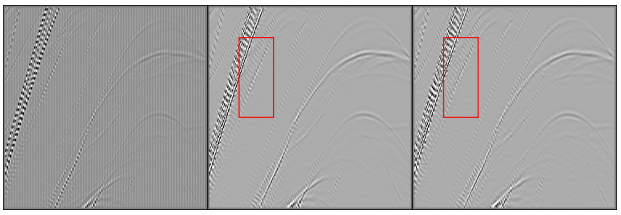


Figure 7: Comparing the evolution of $G_{loss}(G,D)$ during the learning process for $L1$ and $L2$ norms.

In Figure 8 we present the output of the generator using the norms $L1$ (b) and $L2$ (c) given the decimated input seismogram (a). In the rectangular box, we observe that the reflected waves are a little more blurry in $L2$ than in $L1$ case. Indeed, this is an expected behavior, since $L1$ encourages less blurring than $L2$ (Isola et al., 2017). Thus, $L1$ is preferred because it allows the observation of higher frequency signals.



(a) Input (b) $L1$ (c) $L2$

Figure 8: Comparison of the generator output for $L1$ or $L2$ norms. In (a) we present the input decimated seismogram window, in (b) the output for $L1$ and in (c) the output for $L2$. The red rectangle emphasizes a crop in which the reflected wave is more blurry in (b) than in (c).

Kernel size

Another important hyperparameter to test is the kernel size of the convolution of the CNN generator. In fact, the kernel size determines the image area that is compressed during the convolutional process of the CNN. In Figure 9 we show the G_{loss} versus iteration process for kernel size from 3 up to 6. The graph shows that the generator is better at fooling the discriminator with smaller kernel sizes.

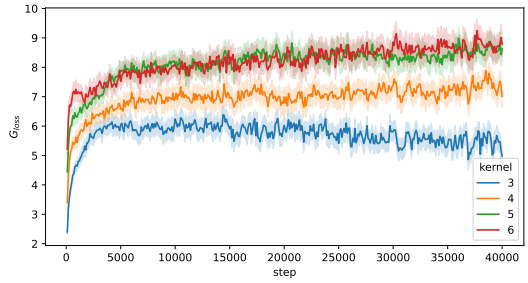
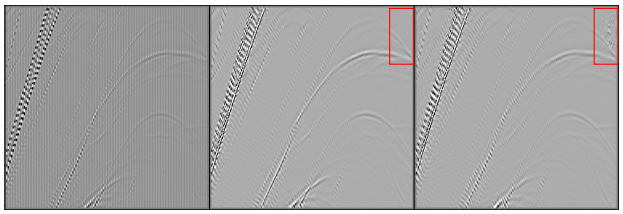


Figure 9: Comparing the evolution of $G_{loss}(G,D)$ during the learning process for kernel sizes from 3 to 6.

In Figure 10 we present the output of the generator using the kernel sizes 3 (b) and 6 (c) given a decimated window of a seismogram as input (a). In the rectangular box, we observe some noisy artifacts when the kernel size is 6, which is not present when the kernel size is 3. Thus, we opt to use smaller kernel sizes.

Generator learning rate vs Discriminator learning rate

The learning rate is a hyperparameter used to set the rate at which the neural network updates its parameters. We have compared the generator and discriminator learning rates together evaluating how the difference between these two parameters influences the performance of the neural network. In Figure 11 we present the generator's output using nine pairs of learning rates, varying the values 1×10^{-4}, 2×10^{-4} and 4×10^{-4}. Each column refer to a single value of discriminator learning rate (D_{LR}) while each row refer to a single value of generator learning rate (G_{LR}). Inspecting the rectangular box area, we observe that the wave reflections have better quality when both learning rates are 1×10^{-4}, leading to less noisy image. Thus, we conclude that it is better to use lower learning rates for both



(a) Input (b) Kernel = 3 (c) Kernel = 6

Figure 10: Comparison of the generator output for different kernel sizes. In (a) we present the input decimated seismogram window, in (b) the output for kernel size 3 and in (c) the output for kernel size 6. The red rectangle emphasizes a noisy output in (c) that is not present in (b).

networks.

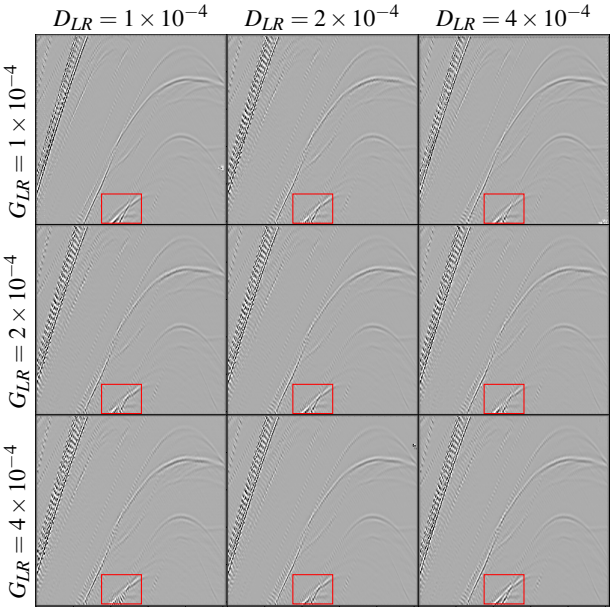


Figure 11: Comparison of the generator output for different learning rates of the generator and the discriminator. Each column refer to the outputs using the discriminator learning rates of 1×10^{-4}, 2×10^{-4} and 4×10^{-4}, respectively, and each row refer to the outputs for generator learning rates of 1×10^{-4}, 2×10^{-4} and 4×10^{-4}, respectively. The red rectangle emphasizes a crop that is difficult to interpolate due to the overlapping wave reflections.

Conclusions

In this work we explore the ML behind the seismic data GAN interpolator. We illustrated the GAN interpolator using a geometry in which the receivers are at high depth. We decimated seismograms to produce the training set of the GAN interpolator. The testing was performed reconstructing the decimated seismograms. The core of the work consists in testing the hyperparameters of the GAN interpolator, seven hyperparameters were examined: batch size, training/test ratio, loss norm, kernel size, λ factor, generator and discriminator learning rates.

The summary of results of the numerical exploration of the hyperparameters is the following. The batch size is not a decisive hyperparameter, we recommend using batch size equal to one to save computational time. We call attention that a similar result was related by Isola et al. (2017). The training/test ratio is also a non determinant quantity, in this case employing the standard pattern 70% training and 30% testing is a good choice. The loss norm $L1$ has a sharper resolution when compared to the $L2$ norm. Similar result was related by Isola et al. (2017). The best kernel size was three, larger kernel sizes produce clear artifacts in the interpolated seismograms. The λ factor is, indeed, an important hyperparameter, the best choice is the largest λ, indicating that for a good interpolated seismogram it is important to give more weight to the generator than to the discriminator. Both, the generator and the discriminator learning rates are relevant quantities, we have obtained the best interpolated seismograms for the smallest learning rates suggesting that slow neural network learning is associated to better learning.

Before finishing the study we notice that we have employed 40,000 steps in our simulations. This time corresponds to the total learning time of the GAN interpolator. However, for the maximal number of steps studied the simulations reveal that the neural network has reached a stationary behavior. To conclude, this is a work in progress and we are still exploring the best ML parameters to create optimal interpolated seismograms.

ACKNOWLEDGMENTS

The authors gratefully acknowledge support from Shell Brazil through the "New Computationally Scalable Methodologies for Target-Oriented 4D Seismic in Pre-Salt Reservoirs" project at Federal University of Rio Grande do Norte and the strategic importance of the support given by ANP through the R&D levy regulation. This research was supported by NPAD/UFRN.

References

Chai, X., Gu, H., Li, F., Duan, H., Hu, X., and Lin, K., 2020, Deep learning for irregularly and regularly missing data reconstruction: Sci Rep, **10**, no. 1, 3302.

Crawley, S., Clapp, R., and Claerbout, J., 2005, Interpolation with smoothly nonstationary prediction-error filters:, *in* SEG Technical Program Expanded Abstracts 1999, 1154–1157.

Dramsch, J. S., 2020, Chapter one - 70 years of machine learning in geoscience in review *in* Moseley, B., and Krischer, L., Eds., Machine Learning in Geosciences:: Elsevier, 1–55.

Fang, W., Fu, L., Zhang, M., and Li, Z., 2021, Seismic data interpolation based on u-net with texture loss: GEOPHYSICS, **86**, no. 1, V41–V54.

Fomel, S., 2003, Seismic reflection data interpolation with differential offset and shot continuation: GEOPHYSICS, **68**, no. 2, 733–744.

Géron, A., 2022, Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow: O'Reilly Media, 3 edition.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and

Bengio, Y. Generative Adversarial Networks:, 2014. arXiv:1406.2661 [cs, stat].

Gülünay, N., 2003, Seismic trace interpolation in the fourier transform domain: GEOPHYSICS, **68**, no. 1, 355–369.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A., 2017, Image-to-image translation with conditional adversarial networks: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 5967–5976.

Johnston, D. H., 2013, Practical applications of time-lapse seismic data: Society of Exploration Geophysicists.

Kaur, H., Pham, N., and Fomel, S., 2021, Seismic data interpolation using deep learning with generative adversarial networks: Geophysical Prospecting, **69**, no. 2, 307–326.

Mojica, O. F., and Maciel, J. S., 2020, Seismic modeling from scratch using devito: a demonstration with a typical brazilian pre-salt model: , 90th SEG Annual International Meeting, Expanded abstracts, 2714–2718.

Naghizadeh, M., and Sacchi, M. D., 2009, f-x adaptive seismic-trace interpolation: GEOPHYSICS, **74**, no. 1, V9–V16.

Nguyen, P. K. T., Nam, M. J., and Park, C., 2015, A review on time-lapse seismic data processing and interpretation: Geosciences Journal, **19**, no. 2, 375–392.

Operto, S., Miniussi, A., Brossier, R., Combe, L., Métivier, L., Monteiller, V., Ribodetti, A., and Virieux, J., 2015, Efficient 3-d frequency-domain mono-parameter full-waveform inversion of ocean-bottom cable data: application to valhall in the visco-acoustic vertical transverse isotropic approximation: Geophysical Journal International, **202**, 1362–1391.

Porsani, M. J., 1999, Seismic trace interpolation using half-step prediction filters: GEOPHYSICS, **64**, no. 5, 1461–1467.

Ronen, J., 1987, Wave−equation trace interpolation: GEOPHYSICS, **52**, no. 7, 973–984.

Sacchi, M. D., and Ulrych, T. J., 1996, Estimation of the discrete fourier transform, a linear inversion approach: GEOPHYSICS, **61**, no. 4, 1128–1136.

Sircar, A., Yadav, K., Rayavarapu, K., Bist, N., and Oza, H., 2021, Application of machine learning and artificial intelligence in oil and gas industry: Petroleum Research, **6**, no. 4, 379–391.

Thorbecke, J., and Draganov, D., 2011, Finite-difference modeling experiments for seismic interferometry: Geophysics, **76**, no. 6, H1–H18.

Wu, Y., and Lin, Y., 2019, Inversionnet: An efficient and accurate data-driven full waveform inversion: IEEE Transactions on Computational Imaging, **PP**, 1–1.

Zhang, S.-B., Si, H.-J., Wu, X.-M., and Yan, S.-S., 2022, A comparison of deep learning methods for seismic impedance inversion: Petroleum Science, **19**, no. 3, 1019–1030.