Seismic Fault Classification Using Multilayer Perceptron Neural Network Algorithm

João P. Gomes[1], Vinicius Carneiro[1,2], and Alexsandro G. Cerqueira[1,2] — [1]GAIA-UFBA [2]PPGEOF-UFBA

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

The problem related to the mapping of geological structures, such as faults, based on seismic data analysis is a matter of great interest and deserves attention since such structures can be related to oil and gas accumulation and, therefore, their exploration is of great economic interest. Thus, it is expected to encourage the development of new technologies that help in this task, with the goal of increasing the precision and speed of that process. Artificial Neural Networks (ANNs) are machine learning algorithms that propose to mimic the nervous system of living beings. More specifically, Multilayer Perceptron Neural Networks (MLP) were designed to behave similarly to the human retina, with the aim of recognizing geometric patterns (da Silva, 2016). Once we deal with a problem that can be treated as an image classification, we can assume that the development of such a network is suitable. In this sense, the current work proposed a methodology to detect seismic faults using an MLP. Every model of machine learning requires training and validation steps. So, for the work to be developed, it was necessary to build a database containing samples of seismic amplitude and coherence seismic attribute data from the public 3D seismic F3 Block dataset located at the North Sea, with its respective classifications. When it comes to a prediction problem, one of the main purposes is to increase the generalizability of the model so that it fits unknown data (from a different location, for example). Finally, after the training and testing, it was possible to observe acceptable results of predicted faults in both known and unknown data, indicating the developed model's helpful effectiveness.

Introduction

The drilling process for direct exploration of oil and gas is expensive, and it needs geological and physical data from the subsurface to be more accurate. Therefore, several indirect and cheaper studies of the physical properties of the subsurface are carried out before drilling. This mapping of subsurface properties contributes to understanding the regional geology, increasing the efficiency of an exploration process, improving accuracy, and reducing costs.

The seismic reflection method is one of the leading geophysical methods used in the oil industry. It consists of monitoring the time and amplitude with which acoustic waves artificially generated on the surface penetrate the soil, reflects on geological interfaces, and return to the surface where geophones record them. Afterward the data acquisition, some calculations are performed to process them so they can be better visualized. Finally, the filtered and processed seismic sections allow interpreters to understand the sedimentary basin's history and identify some essential geological structures for the context of oil and gas exploration.

The fault can be mentioned as one of the main structures in oil and gas exploration, and it is a geological formation that represents lithological discontinuity; in other words, it is a rupture in the rock followed by displacement and is generally caused by tectonic movements. These structures favor the movement of fluids between the rocks. Hence, their mapping can indicate zones of interest where the oil, for example, can be trapped. The task of mapping these structures is arduous and requires a significant amount of time since current datasets are enormous, and can reach scales of hundreds of gigabytes. Deep learning techniques can help identify these structures by learning from past knowledge. Numerous applications exist in language and image recognition, advertisements, financial fraud, medicine, etc. In geophysics, we can mention the application carried Santos (2021) that developed a neural network Multilayer Perceptron to estimate zero-phase wavelets from synthetic seismic data. Another successful implementation of artificial intelligence was in interpreting seismic data to detect seismic facies and faults (Wrona et al., 2018; Zheng et al., 2019).

This abstract aims to demonstrate how deep learning techniques can be helpful in fault detection and is divided into three sections: Methods (describes the techniques that were used and the workflow), Results (presents the results of the algorithm application in real data), and Conclusions (which brings an interpretation about the work and results carried out).

Theory

Identifying discontinuous seismic facies (faults) is a complex and highly nonlinear task, making it difficult to use machine learning algorithms based on just a few parameters. Therefore, deep neural networks could be feasible algorithms to automate this process.

Multilayer Perceptron Neural Network

The neural networks can differ from each other by the number of hidden layers, neurons, and interconnections. These settings are called the architecture of a network. The model used in this research was the Multilayer Perceptron (MLP) because this architecture can have one or more hidden layers, in which can be employed activation functions to let the model deal with nonlinear problems. Figure 1 shows an example of a fully connected MLP. For any artificial neural network to be trained, it must follow the *feedforward* and *backpropagation* processes so that

its weights can be updated. The following subsections will brief explain the feedforward and the backward process.

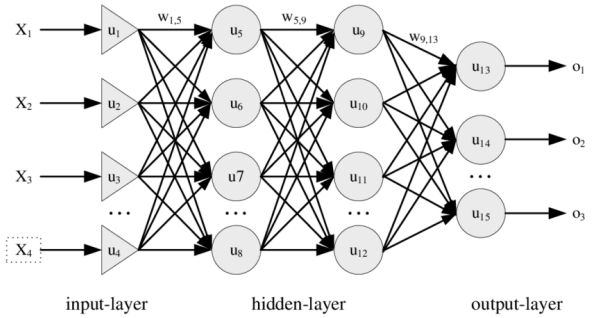

Figure 1: Multilayer Perceptron neural network architecture diagram.

Feedforward

The *feedforward* process contains the steps in which the input values of a single sample pass through the neural network from the input to the output layer in order to calculate the error associated with each of the input variables and their weights.

First, each neuron receives a certain input vector, applies an activation potential u, then an activation function g is applied to that potential, and finally, an error L is calculated to measure some difference between the predicted output and the correct one in such a way that the weights can be updated later on the *backpropagation* process. The general equations of the activation potencial, activation funcion and the cost function (error) is showed below:

$$u(\mathbf{w}) = w_0 + \sum_{i=1}^{n} x_i \cdot w_i \tag{1}$$

$$g(u(\mathbf{w})) = g(w_0 + \sum_{i=1}^{n} x_i \cdot w_i) \tag{2}$$

$$L(g(u(\mathbf{w}))) = L(g(w_0 + \sum_{i=1}^{n} x_i \cdot w_i)) \tag{3}$$

where $\mathbf{w} = [w_0,\ w_1,\ ...,\ w_n]$ is the weight vector, n is the size of the layer before that neuron, x_i is a element of the input vector and g and L are arbitrary functions.

For the first neural layer case, the input vector is the model input $\mathbf{x} = [x_1,\ x_2,\ ...,\ x_n]$, but for any layer after the input vector is the output of the previous layer.

Backpropagation

Most Machine Learning algorithms aim to minimize some loss function, and in order to achieve this task in deep learning models, the cost functions must be known and their derivatives. That is a crucial rule because the parameter optimization of the ANNs is based on a Gradient Descent (GD) algorithm, which consists of updating the parameters through a bunch of iterations by taking the partial derivatives of the loss function L concerning the weights $\mathbf{w}$. In order to accomplish it, the GD method is expressed by the chain rule as shown below in Equation 4.

$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial g_m} \cdot \frac{\partial g_m}{\partial g_{m-1}} ... \frac{\partial g_{i+2}}{\partial g_{i+1}} \cdot \frac{\partial g_{i+1}}{\partial w_i} \tag{4}$$

The initialization of the parameters is usually random and between [-1, 1], and they are updated interactively through the following equation (Eq.5):

$$w_{t+1} = w_t - \alpha \cdot \frac{\partial L}{\partial w_t} \tag{5}$$

where α is the *learning rate*.

Workflow

This research was implemented in Python 3 language, and we used the numpy, pandas, matplotlib, and Keras package to manipulate, visualize the dataset and train the MLP. Figure 2 demonstrate the workflow employed in this work.

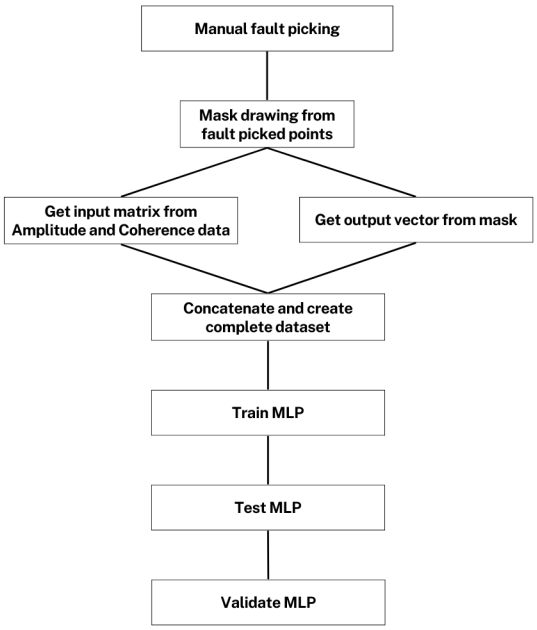

Figure 2: Research workflow.

Dataset

The first step of this research was to create the dataset containing the input and output values so we could train, test and validate the neural network model. The dataset was basically a matrix in which each row was a training/testing/validating sample, and each column represented a variable. The first step in building the dataset consisted of creating the columns corresponding to the input variables. Each input sample was represented by a vector with amplitude, and coherence values. In order, the seismic sections went through part of the process to create the MLP input vectors illustrated in Figure 3 so the vectors could contain information of a squared area of the two sections. For each of the four chosen lines (two inlines and two crosslines), this step was performed in such a way that the extracted red windows shown in Figure 3 slid through all the seismic sections, and the vectors were then concatenated as shown in Figure 4.

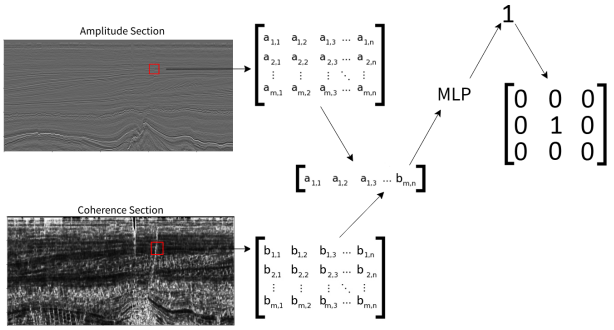

Figure 3: MLP prediction diagram.

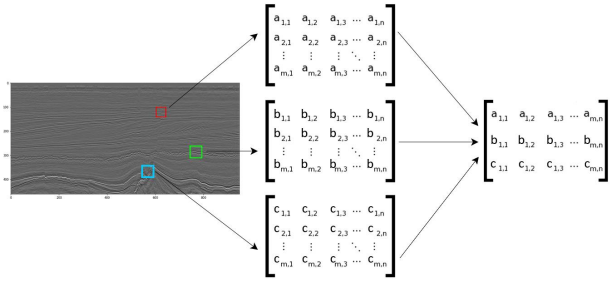

Figure 4: Input matrix building diagram.

As explained before, it was necessary to map some faults manually to obtain the known output values. The geophysicist and Ph.D. student in Geophysics at UFBA, Vinícius Carneiro, carried out this charge. As mentioned before, four lines of the F3 Block seismic cube were chosen. The lines used on this research to test and train the model was selected from different locations of the seismic cube:

- Inlines: 310 and 640; and
- Crosslines: 890 and 940.

As a result, we obtained the positions corresponding to the faults in four seismic sections. Then it was possible to draw lines, segment the faults and create sections only with these drawings with the help of the Python computer vision library OpenCV, as illustrated in Figure 5.

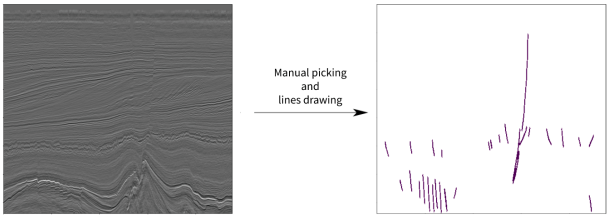

Figure 5: Mask sections building diagram.

Once we had the *mask* sections corresponding to the known segmented faults, we had to obtain a single vector with binary categorical values corresponding to every area of those sections, and a similar process shown in Figure 5 was performed to create the labels of the dataset. As we considered it a classification problem, not a segmentation, we associate each entire polygon with only one label, and these cropped sections must fulfill two requirements to be labeled as a fault:

1. The segmentation trace drawn must cross the center of the image; and
2. A minimum amount of pixels must be denoted as a fault in the image.

If these requirements are satisfied, a categorical value equal to 1 is used to indicate the presence of a fault at that selected zone, as can be displayed in Figure 6.

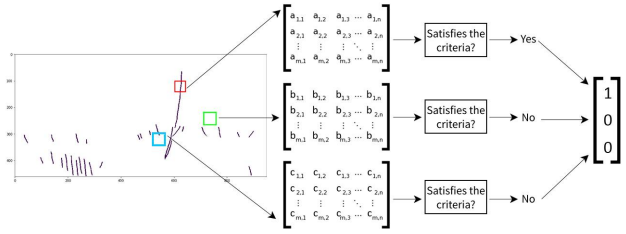

Figure 6: Target vector building diagram.

The width and length of the extracted windows (cropped section) are very important variables. If the window is too large, there may be several faults in different image regions, and it cannot be represented by a single central pixel. On the other hand, if the value is too low, the faults may be exaggeratedly enlarged, and their recognition would be harmed. Therefore, we tested several parameters, and the size of the windows extracted from the sections was 23x23. That is 529 samples of each attribute and 1058 samples in total.

Another criterion to be defined is the minimum number of samples belonging to the target class in the image extracted from the *mask* so that it can be classified as a fault or not. The thickness of the drawn mask was only one pixel, so the maximum possible number of detected pixels would be 23. Since the known faults were manually mapped, the ends of the discontinuities were not highly accurate. Therefore, it was seen that when that minimum number was very low, the discontinuity was often no longer at the associated coordinate. Thus, it was decided that the optimal number was 20 pixels, classified as a fault in the image. That is, the fault should cross the image almost completely.

Considering the established criteria, a dataset was obtained at the end of all the processes described above, as shown in Figure 7. It is possible to note in Figure 7 that the number of windows extracted from 4 lines was 1,349,228, each with 1,058 samples, and the output vectors, which are desirable to be predicted, have only 1 sample.

index	amp0	amp1	amp2	amp3	...	sim525	sim526	sim527	sim528	class
non_fault0	0.000000	0.000000	0.000000	0.000000	...	0.390314	0.330538	0.332001	0.419016	0.0
non_fault1	0.000000	0.000000	0.000000	0.000000	...	0.330538	0.332001	0.419016	0.608372	0.0
non_fault2	0.000000	0.000000	0.000000	0.000000	...	0.332001	0.419016	0.608372	0.592546	0.0
non_fault3	0.000000	0.000000	0.000000	0.000000	...	0.419016	0.608372	0.592546	0.592307	0.0
non_fault4	0.000000	0.000000	0.000000	0.000000	...	0.608372	0.592546	0.592307	0.503879	0.0
...	...	...	...	...	...	...	...	...	...	...
fault1899	-0.006813	0.121139	-0.044186	0.045354	...	0.645096	0.723402	0.764869	0.775791	1.0
fault1900	0.052556	-0.136257	0.065468	-0.013236	...	0.584714	0.399063	0.404265	0.536268	1.0
fault1901	-0.017843	-0.015962	-0.061446	-0.016870	...	0.737325	0.753601	0.762270	0.765323	1.0
fault1902	-0.190306	-0.464833	-0.194589	-0.069167	...	0.672157	0.506158	0.505086	0.601550	1.0
fault1903	-0.041980	-0.097521	-0.014988	-0.151311	...	0.863308	0.820581	0.779758	0.780796	1.0

1349228 rows × 1059 columns

Figure 7: Dataset where amp_i are the amplitude samples, sim_i are the similarity samples, and *class* is the true output with the categorical values (fault: 1; non-fault: 0).

Network Training

The Multilayer Perceptron neural network implemented has five layers: one for the input vector, three hidden layers, and the output one. The first layer (input layer) has 1,058 samples (529 samples of amplitude and 529 of coherence), and the second, third, fourth (hidden layers), and last (output layer) with 1,058, 529, 264, and 2 neurons, respectively, as illustrated on Figure 8. The two neurons at the last layer are the probabilities of the samples belonging to the *fault* and *non-fault* classes. Totaling, then, 2,941,505 parameters to be optimized.

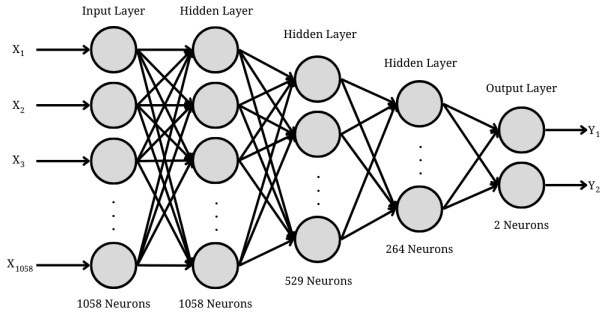

Figure 8: Trained Multilayer Perceptron neural network architecture.

With the architecture set, other parameters also should be configured:

- Activation function: hyperbolic tangent (limits the output value into the interval [-1, 1])

$$g(u) = \frac{1 - e^{-\beta \cdot u}}{1 + e^{-\beta \cdot u}} \tag{6}$$

where β is associated with the slope of the curve at its inflection point and u is the activation potential (Equation 1);

- Cost function (which is desired to be minimized): categorical cross entropy

$$CCE = -\sum_{i=1}^{n} y_i \cdot log(y_i') \tag{7}$$

where n is the number of samples, y is the known target value, and y' is the target value predicted by the model;

- Learning rate: is represented by the α controls the speed at which the parameters are optimized according to the cost function. If α is too large, the optimization can lead to non-convergence, and if it is very small, the optimization can lead to local convergence without reaching the global minimum. In this research, the value chosen for the training was 0.0005, and the optimizer was the Adaptive Moment Estimation (Adam); and

- Number of epochs: represents the number of times the data set will be used as input to the network. Again, the wrong choice of this number can lead to future problems. Once a very low number is chosen, the network may not be able to reach the solution in time. A very large number can determine an overfit. I.e. the network fits very well to the training data, but it is not able to generalize for different samples.

Results

The Holdout cross-validation method was used to verify the quality of the Multilayer Perceptron network developed. That is, the data set was split into a training subset and a test subset, and these were randomly rearranged. For this work, it was decided that 80% of the database would be used for training and the remainder (20%) for testing. From the execution of the model, it was possible to verify an accuracy of 97.69% and 91.73% when applied to the training and test subsets, respectively. However, beyond accuracy, it is common to use other metrics to validate classification models, such as precision, recall, and f1-score (Equations 8,9 and 10).

$$Precision = \frac{TP}{TP + FP} \tag{8}$$

$$Recall = \frac{TP}{TP + FN} \tag{9}$$

$$F1\ Score = \frac{2 \cdot precision \cdot recall}{precision + recall} \tag{10}$$

Where TP means "true positive" (amount of samples that were correctly predicted as the target class), FP means "false positive" (amount of samples that were classified as the target class but did not belong to it), and FN means false negative" (amount of samples that were not classified as the target class but belonged to it). Therefore, analyzing the widely used metric $F1\ Score$, a value of 91% was obtained.

To check the generalization capacity of the network, a seismic section which was not selected for the training step but also from the F3 Block was selected to be used as a blind test. As a result, it is possible to visualize the probability of a given sample to be or not a fault in Figure 9. Notably, the ANN enhanced the regions with seismic fault occurrences, such as the main fault, the faults associated with the carbonate platform, and chaotic seismic facies located between 280 and 350 pixel rows.

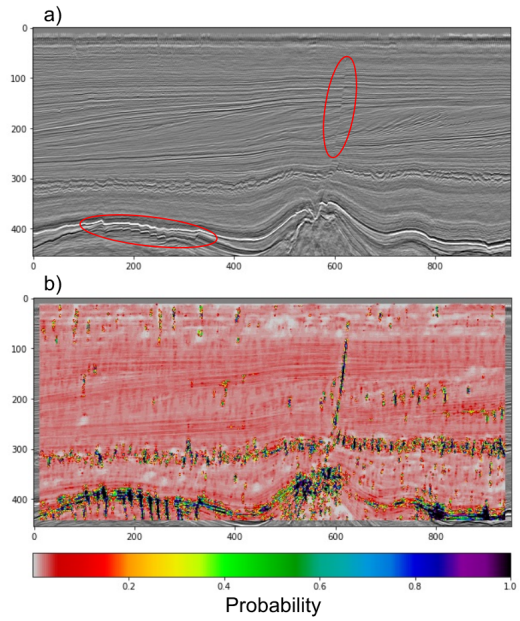

Figure 9: a) amplitude seismic section with faults highlighted by red elipses. b) probability map - result of the model applied to the section a).

Conclusions

The Multilayer Perceptron Neural Network trained in this work showed the capability to enhance regions in the seismic section with seismic fault occurrence without a high resolution in its delimitation. The main and secondary faults were pointed out as having a high probability of being a fault according to Figure 9 (a), which is visually coherent when compared with the regions with a low probability of being a non-fault. Although the methodology needs to be improved to identify faults more precisely, this method can assist interpreters with some faults interpreted in their dataset to have new insights into seismic interpretation.

Acknowledgments

The authors would like to thank National Council for Scientific and Technological Development (CNPQ) and the Instituto Nacional de Ciência e Tecnologia de Geofísica do Petróleo (INCT-GP) for João Pedro's Scientific Initiation scholarship. Vinicius Carneiro thanks the Fundacao de Amparo a Pesquisa do Estado da Bahia (FAPESB) for his Ph.D. scholarship.

References

ALAUDAH, Yazeed. & MICHALOWICZ, Patrycja. & ALFARRAJ, Motaz. & ALREGIB, Ghassan. (2019) A Machine Learning Benchmark for Facies Classification. Interpretation. 7. 1-51. 10.1190/int-2018-0249.1.

DA SILVA, Ivan. & SPATTI, Danilo & FLAUZINO, Rogério. (2016) Redes neurais artificiais para engenharia e ciências aplicadas, Artliber.

HAMZEH, Ali. (2021) Machine learning methods in fault detection. Thesis (Master's degree in Petroleum and Mining Engineering) - Politecnico di Torino, p. 110.

JAMES, G. & WITTEN, D. & HASTIE, T. & TIBSHIRANI, R. (2013) An introduction to statistical learning: with Applications in R, Springer International Publishing.

KOPILER, Alberto & DA SILVA, Victor & OLIVEIRA, Luiz & LINDEN, Ricardo & SILVA, Luis & FONSECA, Bruno. (2019) Redes neurais artificiais e suas aplicações no setor elétrico, Revista de Engenharias da Faculdade Salesiana, n. 9 p. 27-33.

MOREIRA, Sandro. Available at: https://medium.com/ensina-ai/rede-neural-perceptron-multicamadasf9de8471f1a9. Accessed: 03 May 2022.

SANTOS, Lorena. (2021) Estimativa de wavelets de fase zero a partir de traços sísmicos sintéticos utilizando a rede neural Perceptron multicamadas. Thesis (Undergraduate in Geophysics) - Instituto de Geociências. Universidade Federal da Bahia. Salvador, p. 82.

SCHROOT, B. & SCHÜTTENHELM, R. (2003) Expressions of shallow gas in the netherlands north sea, Netherlands Journal of Geosciences, 82(1):91–105.

SILVA JUNIOR, Antônio. (2013) Classificação de falhas geológicas em dados sísmicos usando correlograma e aprendizado de máquina. Thesis (Undergraduate in Computer Science) - Centro de Ciências Exatas e Tecnologia, Universidade Federal do Maranhão. p. 50.

SOARES, Pablo & DA SILVA, José. (2011) Aplicação de Redes Neurais Artificiais em Conjunto com o Método Vetorial da Propagação de Feixes na Análise de um Acoplador Direcional Baseado em Fibra Ótica. Revista Brasileira de Computação Aplicada. 3. 10.5335/rbca.2013.1803.

WEI, X. & ZHANG, C. & KIM, S. & JING, K. & WANG, Y. & XU, S. & Xie, Z. (2022) Seismic fault detection using convolutional neural networks with focal loss, Computers & Geosciences, 158.

WU, Xinming & LIANG, Luming & SHI, Yunzhi & FOMEL, Sergey. (2019) FaultSeg3D: Using synthetic data sets to train na end-to-end convolutional neural network for 3D seismic fault segmentation, Geophysics, 84(3): IM35-IM45.

WRONA, T. & PAN, I. & GAWTHORPE, R. & FOSSEN, H. (2018) Seismic facies analysis using machine-learning, Geophysics. GEO-2017-0595.R2.