Computers & Geosciences 135 (2020) 104344

Contents lists available at ScienceDirect

Computers and Geosciences

journal homepage: www.elsevier.com/locate/cageo

# Seismic fault detection in real data using transfer learning from a convolutional neural network pre-trained with synthetic seismic data

Augusto Cunha[a,b,*], Axelle Pochet[a,b], Hélio Lopes[a,b], Marcelo Gattass[a,b]

[a] *Departamento de Informática, PUC-Rio, R. Marquês de São Vicente, 225, Gávea, Rio de Janeiro, RJ, 22451-900, Brazil*
[b] *Instituto Tecgraf, PUC-Rio, R. Marquês de São Vicente, 225, Gávea, Rio de Janeiro, RJ, 22451-900, Brazil*

ARTICLE INFO

*Keywords:*
Transfer learning
Convolutional neural network
Seismic fault

ABSTRACT

The challenging task of automatic seismic fault detection recently gained in quality with the emergence of deep learning techniques. Those methods successfully take advantage of a large amount of seismic data and have excellent potential for assisted fault interpretation. However, they are computationally expensive and require a considerable effort to build the dataset and tune the models. In this work, we propose to use Transfer Learning techniques to exploit an existing classifier and apply it to other seismic data with little effort. Our base model is a Convolutional Neural Network (CNN) trained and tuned on synthetic seismic data. We present results of Transfer Learning on the Netherland offshore F3 block in the North Sea. The method gives satisfying results using as input a single interpreted section, despite the naturally high imbalance of the labeled classes. The proposed networks are easily tuned and trained in a few minutes on CPU, making the technique suited for practical day-to-day use.

## 1. Introduction

Seismic faults interpretation is a crucial step for reservoir characterization. Depending on their geometry and behavior, faults can drain or block fluids in a reservoir, controlling the resources recovery potential and the leading exploitation strategies. The size and complexity of seismic surveys make manual interpretation a laborious task, highlighting the importance of developing efficient computer-aided tools to assist interpreters.

Faults can be detected in migrated data by looking at the local discontinuity of the seismic signal (coherence (Bahorich and Farmer, 1995) (Luo et al., 1996), semblance (Marfurt et al., 1998), variance (Van Bemmel and Pepper, 2000), chaos (Randen et al., 2001), edge detection (Di and Gao, 2014)) or at the geometry of the reflectors (curvature (Lisle, 1994), flexure (Roberts, 2001) (Al-Dossary and Marfurt, 2006) (Gao, 2013)). Information given by the geometry of horizons can also be useful to uncover faults location (horizon dip and azimuth maps (Rijks and Jauffred, 1991)).

The aforementioned seismic attributes gave rise to many fault detection tools in the past decades. For example, Pedersen et al. (2002) used one or more attribute cubes to extract fault surfaces using Artificial Ants to track discontinuity. Wang et al. (2014) performed color transformations on semblance maps, followed by skeletonization of the highlighted fault regions. Recently the same authors proposed the combination of the Hough Transform and tracking vector to extract faults from coherence maps (Wang and Alregib, 2017). Yan et al. (2019) performed principal component analysis (PCA) with forward and backward diffusion to create a fault attribute. Wu et al. (Wu and Zhu, 2017) used a matched filtering method to enhance a precomputed fault attribute volume, and simultaneously estimate fault strikes and dips that can be used in a fault surface generation strategy. Those methods generally fall into two categories: faults are well extracted, but many artifacts remain, or the result is clean, but not all faults are detected (Di et al., 2017a).

Another approach is to combine attributes using machine learning algorithms. Supervised methods extract meaningful information from a set of inputs (the features) and observations (here, the fault location), then apply acquired knowledge to predict new samples. For example, Tingdahl and de Rooij (Tingdahl and de Rooij, 2005) used a set of 12 attributes as input features of an Artificial Neural Network, generating fault probability maps. Support Vector Machine (SVM) also achieved promising results when used with a selection of seismic attributes (Di et al., 2017b), image textures (Guitton et al., 2017), seismic attributes on pre-stack data (Zhang et al., 2014), and with seismograph data to detect

* Corresponding author. Instituto Tecgraf, PUC-Rio, R. Marquês de São Vicente, 225, Gávea, Rio de Janeiro, RJ, 22451-900, Brazil.
*E-mail addresses:* acunha@tecgraf.puc-rio.br (A. Cunha), axelle.pochet@gmail.com (A. Pochet), lopes@inf.puc-rio.br (H. Lopes), mgattass@tecgraf.puc-rio.br (M. Gattass).

https://doi.org/10.1016/j.cageo.2019.104344
Received 25 July 2018; Received in revised form 24 October 2019; Accepted 1 November 2019
Available online 9 November 2019

seismic events (Kortström et al., 2016). The recent increase in computational power allowed the use of powerful deep learning techniques. Compared to standard machine learning methods, deep networks can learn new features dynamically during training, explaining their success in solving complex tasks. Araya-Polo et al. (2017) trained deep neural networks on synthetic pre-stacked data and detected faults using the innovative Wasserstein loss function. Image-oriented methods like Convolutional Neural Networks (CNNs) are especially promising as they recently achieved state-of-the-art results (Wang et al., 2018). While (Huang et al., 2017) used CNNs on seismic attribute maps (Di et al., 2018), and (Pochet et al., 2019) showed that such networks could work well with the seismic amplitude as the sole input, avoiding the problem of heavy computation for multi-attribute techniques. Also, CNNs were used to detect automatically geological landforms on Mars using high-resolution images acquired by the Mars Reconnaissance Orbiter (MRO) (Palafox et al., 2017) and other geological structures like hydrocarbon accumulations (Souza et al., 2019).

Despite very encouraging results, deep learning techniques present some drawbacks. As supervised learning methods, they require a large amount of marked seismic data as input; the generalizability of the proposed networks to other seismic cubes is hard to assess, as studies generally train and classify in the same field; building a CNN model on new data involves a long tuning step, due to the substantial amount of hyper-parameters to adjust.

In this work, we propose to mitigate those drawbacks: by applying Transfer Learning (TL) on an existing pre-trained model, we can work with small datasets, tune a few hyper-parameters and adapt to different kinds of data. Indeed, TL methods use the knowledge acquired from a trained model and adapt it to a similar learning task (Pan and Yang, 2010). Using as the base model a CNN trained on synthetic seismic data in previous work (Pochet et al., 2019), we test different TL strategies on the Netherland offshore F3 block (Opendtect). Fig. 1 sums up the process proposed in this article. The first step, training classifier 1 from synthetic data, it was detailed in our previous work (Pochet et al., 2019). We justify the use of this model in section 2. The second part, applying transfer learning on real data using classifier 1 as a pre-trained model, is detailed in this article. Section 3 details the preparation of the real dataset: we focus on making the real inputs as similar as possible to the base model inputs to increase TL effectiveness. Section 4 then presents TL strategies, and we draw conclusions on the whole process in Section 5.

## 2. CNNs and pre-trained model

CNNs perform image feature extraction and classification through a sequence of specific layers, as shown in Fig. 2. The first part of the network is composed of a series of convolution and pooling operations. Convolutions apply learnable filters throughout the input image, producing one feature map per filter. They are at the core of the feature

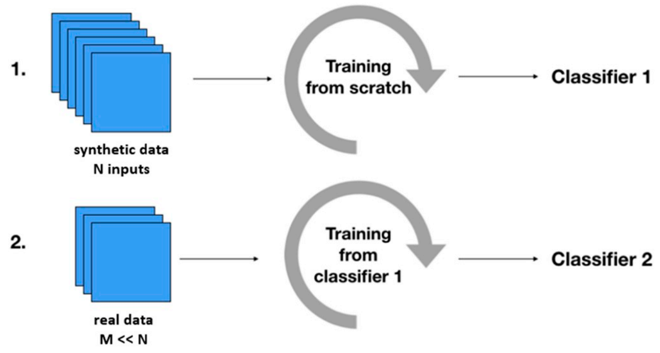

**Fig. 1.** Step 1 is the subject of our previous work (Pochet et al., 2019). Step 2 is the core of this article.

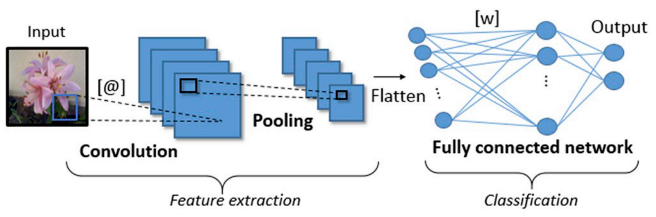

**Fig. 2.** Architecture of a CNN, with: [@] = trainable filters; [w] = trainable weights.

extraction process. Pooling layers perform non-linear downsampling of the feature maps to reduce the number of parameters and, consequently, the computational cost of training. Pooling also brings robustness to feature translation on the map. The last part of the network is a set of fully connected layers, which receive as input the flatten feature maps produced by the last pooling step. Each neuron connection has a trainable weight. Finally, optional layers can be added to this scheme to improve network performances. Rectified Linear Units (ReLU) avoid problems with vanishing gradient caused by other activation functions as a sigmoid or hyperbolic tangent, promote model sparsity (Glorot et al., 2011) and carry fast training (Krizhevsky et al., 2012). Dropout prevents overfitting the data, which is common in deep models. It is generally performed after layers learning a large set of parameters, which is when the overfitting risk is the highest.

CNNs are trained using an error back-propagation scheme, and the final classifier is the sequence of adjusted weights for each convolution filter and neuron connection. TL methods use all or part of those weights as the initial state for training a new dataset on a different classification task. While training CNNs from scratch requires a large amount of input data, TL allows working with small datasets, and usually performs well if the classification tasks are similar.

Consequently, we choose as base model a CNN trained for the task of seismic fault detection, presented in detail in previous work (Pochet et al., 2019). This way, the classification task is precisely the same, and the difference lay only in the datasets. Table .1 shows the architecture of the network.

This CNN was trained with synthetic amplitude patches generated with the IPF code from (Hale, 2014). Compared to real data, synthetic data present several advantages for deep network training. First, they allow total control of the ground truth and thus avoid marking errors. Second, the marking being automatic and fast, synthetic data provides easy scaling regarding the number of training examples. Finally, they avoid problems with data privacy, allowing free distribution. However, with the IPF code, we could not reproduce all types of real seismic objects, such as channels or salt domes. The geometry of the faults was also simplified, as only straight faults were generated.

The model input patches are of size $45 \times 45$, contain amplitude values between $-1$ and 1, and represent two classes: Fault and Non-Fault. Fault patches are centered on a single straight fault crossing the patch entirely, while Non-Fault patches are far from any fault. The model outputs a classification for each patch, which is considered as the classification of their central pixel.

This classifier performed very well on synthetic data. It also gave some promising visual results when applied to a section of the Netherland offshore F3 block, leading us to consider it as a good candidate for TL.

## 3. Real dataset preparation

In order to maximize TL effectiveness, we prepare the real dataset to make it as similar as possible to the base model training conditions. We thus extract patches from the real data following precise steps presented in this section.

**Table 1**
Pre-trained model architecture.

| Layers | Conv. | Conv. | ReLU | Max Pooling | ReLU | Conv. | Conv. | ReLU | Max Pooling | ReLU | Fully Connect. | ReLU | Dropout | Fully Connect | ReLU |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Filters | 20 | 20 | | | | 50 | 50 | | | | | | | | |
| Filter size | 3x3 | 3x3 | | 2x2 | | 3x3 | 3x3 | | 2x2 | | | | | | |
| Neurons | | | | | | | | | | | 16 | | | 32 | |

## 3.1. Patch size

The base model was trained with synthetic patches whose seismic signal frequency content does not match that of real data: while real seismic signal commonly ranges between 20 and 100 Hz although there are processes that can modify this range, the synthetic dataset that we made does not present patches with a higher frequency than 20 Hz. Consequently, the amplitude patterns are different in synthetic and real patches if the same patch size is used in both cases. However, proper scaling of real patches allows minimizing this difference: indeed, selecting smaller patches in the real data and resizing them to the proper input size of $45 \times 45$ naturally lowers the frequency of their seismic signal. We propose here a method to automatically select the proper patch size, based on patch texture comparison.

We begin by randomly selecting 500 patches in the synthetic dataset, and extracting 500 patches of constant size in the real dataset. We smooth the real patches to minimize noise, resize them to the expected size of $45 \times 45$ using bicubic interpolation, then for each patch we compute six of the Haralick textural features (Haralick et al., 1973), a set suggested by (Conners et al., 1984): Inertia, Cluster Shade, Cluster Prominence, Local Homogeneity, Energy, and Entropy. For efficiency, we calculate them in the gray-level domain: amplitude values are normalized between 0 and 255. Using these textures as coordinates, we build the *proximity matrix*, which contains the Euclidean distance between each patch pair. This distance represents here the texture dissimilarity between patches inside and between sets, in 6-dimension.

We then use the classical Multi-Dimensional Scaling (MDS) method (Kruskal, 1964) to represent computed distances in 2-dimension. Given the proximity matrix for dimension *n*, classical MDS allows to visualize the coordinates of the objects in a dimension $m \leq n$, by minimizing the Stress function:

$$Stress = \sqrt{\frac{\sum_{i<j}\left(d_{ij} - \widehat{d}_{ij}\right)^2}{\sum_{i<j} d_{ij}^2}}$$

where *dij* are the input distances between pairs of objects and $\widehat{d}_{ij}$ are the predicted distances in the output space, called disparities. Fig. 3 plots the synthetic patch set along with real patches of size $45 \times 45$, $20 \times 20$, and $8 \times 8$.

For each set, the texture is roughly homogeneous, and patches appear as clusters. The distance between clusters' centroids gives us the relative dissimilarity between patch sets. We found that the closest textures between synthetic and real data are obtained for real patches of size $20 \times 20$.

## 3.2. Patch extraction

In this step, we focus on minimizing the manual effort for real dataset patch extraction. First, we select a single, small section of the seismic cube where some faults are clearly visible: crossline number 900. We then format the section to enhance fault visualization and to match the base model training conditions: we clip the histogram between amplitude values of + - 6000, then smooth the section to minimize the effect of noise. Finally, we normalize amplitude values between $-1$ and 1. We manually pick some visible faults and generate the binary mask for the section.

As we know that marking all faults is difficult, we suppose that all faults of the section were not marked. Consequently, if we extract Non-Fault patches directly using this mask, the dataset will contain errors: many Non-Fault patches may indeed contain faults. To minimize this error, we thus select some regions where we are confident that no fault exists.

Finally, we extract all Fault patches using the binary mask and a set of Non-Fault patches using the defined non-faulted regions. The resulting dataset is highly imbalanced: extracting 1 Non-Fault patch every 7 pixels results in 3252 Non-Fault for 252 Fault patches. We thus perform data augmentation on the Fault class: one of the only relevant operations in the case of seismic data is horizontal flipping. Indeed, vertical flipping or rotation may lead to unrealistic configurations, while scaling would change the patch amplitude patterns.

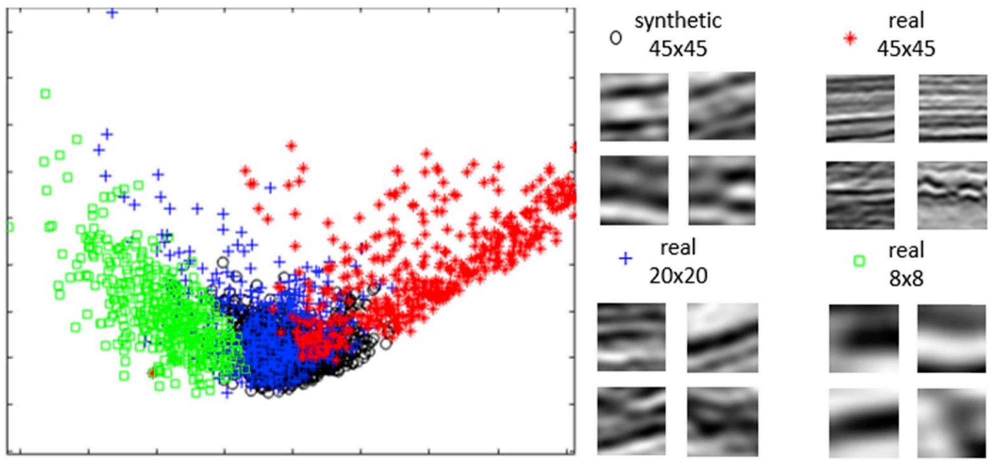

**Fig. 3.** Relative texture dissimilarity between patches, using MDS to plot the computed 6-dimensional Haralick texture distances in 2-dimension. Each point corresponds to a patch. The different symbols distinguish sets of same size patches, synthetic or real.

We then resize all patches from $20 \times 20$ to the expected input size of $45 \times 45$. The final dataset contains 504 Fault patches, resulting in a balance of 13.4% for the Fault class, which is still highly imbalanced.

## 4. Transfer learning strategies

TL strategies rely on the observation that the different parts of CNN learn different things. In particular, earlier convolutional layers learn general features such as edges or blobs, while deeper convolutional layers learn specific features related to the given classification task and dataset, as shown in Figs. 4 and 5. As we ensured similarity between tasks and datasets, we assume that the pre-trained model specific features should be of good quality. We thus investigate two common TL strategies: fine-tuning the whole network and using the CNN as a Feature Extractor with a tuning of the classification layers. In the latter strategy, we test two types of classifiers: the Multi-Layer Perceptron (MLP) and the Support Vector Machine (SVM).

### 4.1. *Full fine tuning (FFT)*

We train the entire network with initial weights equals to the pre-trained model weights instead of random ones. In this context, the learning rate should be low in order to avoid a sudden change in weights during back-propagation. The number of epochs should also be low to avoid overfitting.

### 4.2. *Feature extractor with MLP (FE-MLP)*

We freeze the weights of the whole convolutional part of the CNN, in order to extract fixed features that we feed in the fully connected layers. The only remaining trainable parameters are the neural network weights. The last layer is the Softmax function, which outputs a normalized probability for each class $i$ through the expression:

$$p_i = \frac{exp^{\left(w_i^T x\right)}}{\sum_{k=1}^{2} exp^{\left(w_k^T x\right)}}$$

with $x$ the features to predict and $w$ the vector of weights. We fix the number of hidden layers to 2 and the learning rate to 0.01. We tune the number of neurons in each layer and the number of epochs.

### 4.3. *Feature extractor with SVM (FE-SVM)*

In many deep learning applications, SVM proved better than MLP at predicting classes, given good input features (Tang, 2013). SVM minimizes the following squared hinge loss function:

$$\min_{w} \frac{1}{2} w^T w + C \sum_{n=1}^{N} \max\left(1 - \left(w^T x_n + b\right) y_n, 0\right)^2$$

with $x$ the vector of training features, $N$ the number of features, $w$ the vector of trainable weights, $b$ the bias term, and $y$ the vector of true labels. Constant $C$ controls the penalization of outliers. SVM performs new sample $\widehat{x}$ classification only by looking at the sign of the final optimized predictor function $\boldsymbol{w}^{\boldsymbol{T}} \widehat{\boldsymbol{x}} + \boldsymbol{b}$. We use SVM with kernels to explore non-linear solutions. To find a good SVM classifier, we tune the SVM kernel type and the value of constant C.

## 5. Results

We use as a benchmark the pre-trained model applied to real data without TL. Table 2 shows the parameters and numerical performance of each model. Benchmark metrics were calculated over the entire real dataset. For TL models, due to the small size and high imbalance of the dataset, metrics were averaged over 5-fold stratified cross-validation. We consider the Fault class as the positive class.

The benchmark model shows a high specificity, meaning it should output a few false positives: it is good at discarding faults. However, it has very low sensitivity, meaning that it will miss many faults in practice. TL models all manage to drastically increase sensitivity without losing in specificity, despite the under-representation of the Fault class in the dataset. Feature extractor strategies present overall better results than the Full Fine Tuning method. In particular, they have better sensitivity, meaning they should highlight most of the faults. All TL models were trained in a few minutes on CPU. Note that the 5-folds cross-validation increased the training time by 5 compared to a single training and validation step.

In order to better assess the models' quality, Figs. 6 and 7 show classification over two time-sections of the F3 cube. We classified all inlines of the regions and painted to white the Fault patches central pixels on the time slice. We provide a simple interpretation performed visually on the time section in order to help following the faults.

As expected, the benchmark model misses big parts of the faults. FFT model performs surprisingly well in the section of Fig. 6. However, it completely misses some of the faults of Fig. 7 section. Both FE methods give similar classification, but FE-SVM qualitatively seems to provide slightly cleaner results.

### 5.1. *Comparative results*

We compared our results with a similar technique, a deep learning approach using CNN to find fault regions proposed by Haibin Di et al., 2018. Due to the lack of quantitative results on the analyzed method, we will show a qualitative comparative analysis with our models'

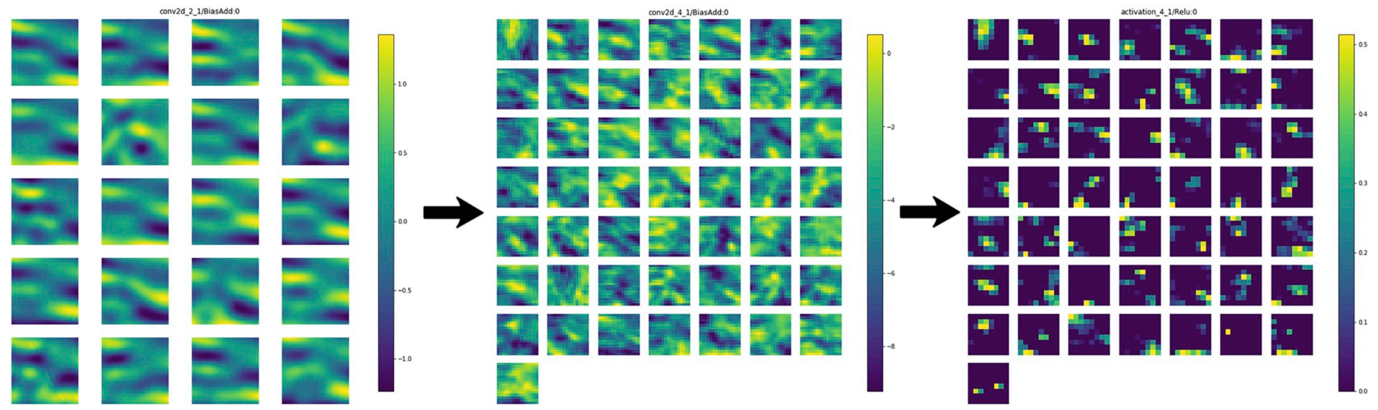

**Fig. 4.** CNN feature map of two convolution layers and activation layer highlighting learned features.

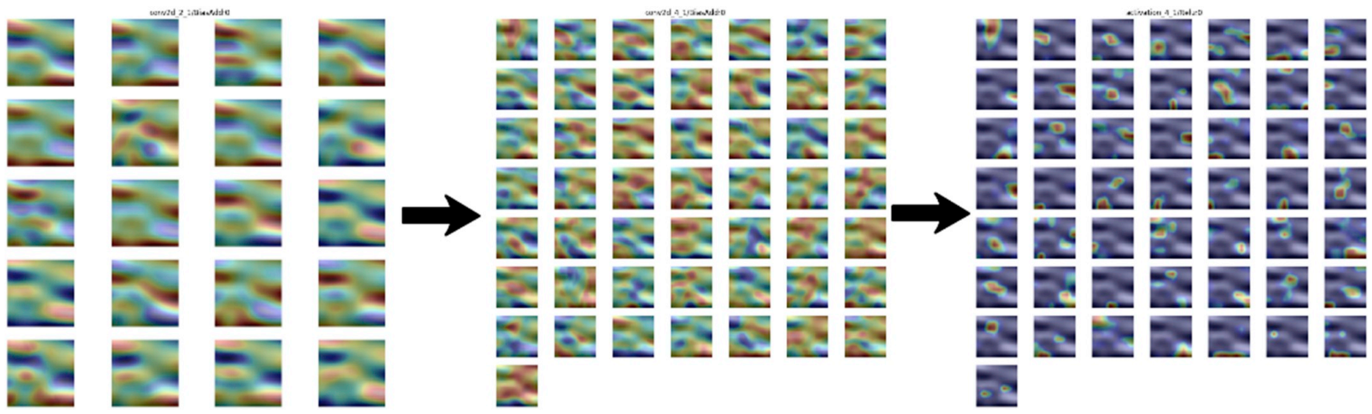

**Fig. 5.** CNN heatmap of two convolution layers and activation layer highlighting learned features.

**Table 2**
Quantitative evaluation of benchmark and TL models. Parameters were tuned manually. Training times were obtained with an Intel® Core™ i7-5960X Processor Extreme Edition, with 16 threads.

| Model | Parameters | Accuracy | Sensitivity | Specificity | F1-score | ROC AUC | Training time |
|---|---|---|---|---|---|---|---|
| Benchmark | – | 0.870 | 0.255 | 0.965 | 0.346 | 0.819 | — |
| FFT | Learning rate = **0.01**<br>Epochs = **20** | 0.957 | 0.720 | **0.994** | 0.818 | 0.992 | 653s |
| FE-MLP | Neurons in layer 1 = **100**<br>Neurons in layer 2 = **200**<br>Epochs = **20** | **0.977** | **0.908** | 0.988 | **0.915** | 0.992 | 83s |
| FE-SVM | Kernel type = **RBF**<br>C= **10** | 0.974 | 0.892 | 0.987 | 0.903 | **0.994** | 162s |

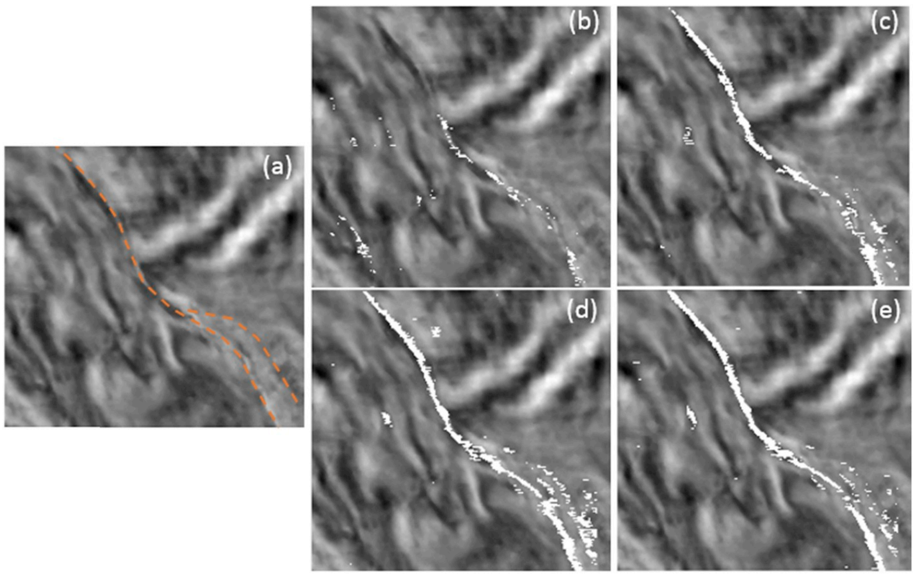

**Fig. 6.** (a) Time slice 924 (inlines 150 to 350, crosslines 840 to 1050) with simple interpretation. (b) Benchmark classification. (c) FFT. (d) FE-MLP. (e) FE-SVM.

classification and quantitative analysis with a manual interpretation for each slice provided by our expert using minimum similarity attribute (Barnes, 2016) with time gate [-28,28], 56 ms.

The CNN approach used a subvolume of New Zealand Great South Basin (GSB) at inline 1565 to 2531 with step 2, crossline 2568 to 3568 with step 2, and time −1000 ms to −1400 ms with step −4 ms. To generate a fair result, we used a thinner version manual interpretation of Haibin Di et al., 2018 because our network was trained with faults represented by lines with one-pixel thickness. We selected crossline 2800 as a slice for training with patch size 13 resized to 45 instead of crosslines 2700, 2800, and 2900 as the analyzed method did. We can see the comparative results in training crossline 2800 for each proposed model in Fig. 8 and Table 3.

Since the dataset is imbalanced, the specificity and accuracy score does not represent well these metrics meaning because 1% in specificity can be hundreds of false positive in the input section. In this scenario, where we have uneven class distribution, accuracy is not a good metric to measure the overall performance. Instead of using accuracy is better to use the F1 score because it is a weighted average of sensitivity and specificity that represents respectively how correctly we can classify fault patch and how correctly we can classify non-fault patches. Also, we can use AUC to evaluate how good the classifier was, where we expected

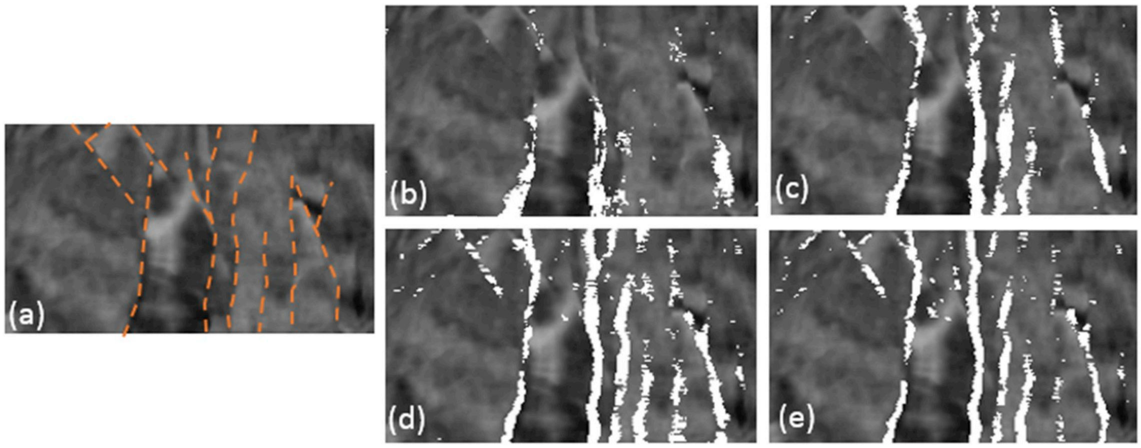

**Fig. 7.** (a) Time slice 1736 with simple interpretation (inlines 110 to 240, crosslines 390 to 590). (b) Benchmark classification. (c) FFT. (d) FE-MLP. (e) FE-SVM.

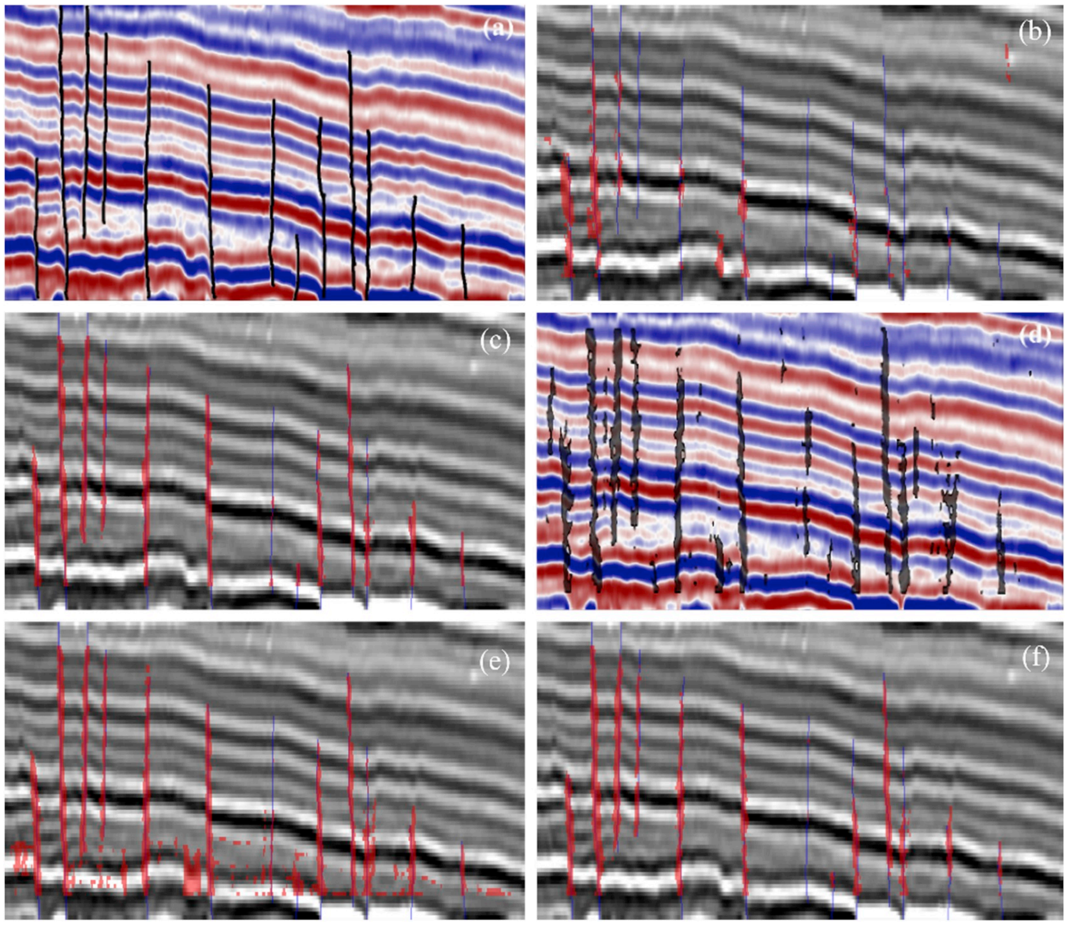

**Fig. 8.** GSB crossline 2800, the blue lines are a thinner version of manual interpretation provided by Haibin Di et al., 2018) (Di et al., 2018), the red regions are classified regions by our network. (a) Original provided manual interpretation. (b) Benchmark result. (c) FE-SVM. (d) Haibin Di et al., 2018 best result. (e) FE-MLP. (f) FFT. (For interpretation of the references to color in this figure legend, the reader is referred to the Web version of this article.)

**Table 3**
Quantitative evaluation of benchmark and TL models on GSB crossline 2800.

| Model | Accuracy | Sensitivity | Specificity | F1 Score | AUC |
|---|---|---|---|---|---|
| Benchmark | 90.0 | 19.4 | 99.5 | 31.6 | 70.2 |
| FE-SVM | 95.2 | 60.0 | 100 | 75.0 | 97.0 |
| FE-MLP | 93.9 | 68.3 | 97.4 | 72.8 | 93.1 |
| FFT | 94.8 | 56.4 | 100 | 72.1 | 97.8 |

value higher than 80%. That means that the classifier was classifying faults and non-fault patches well with a lower number of false positive.

In Fig. 8, we observed that TL helped to improve our results if compared with the benchmark, as reflected in training metrics in Table 3. FE-SVM in Fig. 8(c) and FFT in Fig. 8(f) shown the best results in the training section, where we aim to reach a higher sensitivity and specificity score. When compared with the analyzed method best result in Fig. 8(d), our results detected some full faults, few partial faults, and did not detect one fault, but we have fewer false positives.

We also generated comparative results with the test data using inlines 1791 and 2011 in Fig. 9, and crosslines 2600 and 3000 in Fig. 10.

As we can see in Tables 4–7, FFT model performs better in each test section and improve the benchmark results of our previous work. Although our models do not find all faults, we can find partial faults that can be a fault indicator that helps interpreter detect the full fault, and we have a lower number of false positives with FFT model when compared with the analyzed method results. However, for inline 2011, we had the worse results due to the false negatives on the zigzag like pattern, and MLP models in each tested section showed a higher number of false positives, as we can see in lower specificity metrics when compared with FE-SVM and FFT.

In terms of advantages, our proposed methodology is CPU based, fast, and only needs one manual interpreted section of real data to training the classifier in new seismic data instead use three as the analyzed method. Due to the TL approach, we usually need only a few epochs (lower than 20) to retrain synthetic weights with the real data section information to make a classifier that performs well, close to the compared one. Our methodology also works with GPU processing that will improve the processing time.

In terms of limitations, we identify that we cannot classify well zigzag like patterns due to spots that locally looks like a fault. The methodology is not an end-to-end network, having intermediate steps like preprocessing input data, find the best patch size, generate patches, and reconstruct the classified section. Although the analyzed method also uses patches strategy, they used fewer intermediate steps than us.

## 6. Conclusion

Detecting real faults using Transfer Learning from a model trained on synthetic seismic data showed powerful despite the naturally high imbalance of classes in real data. With a small dataset generated by marking only some of the faults on a single cross-section, we were able to build a satisfying fault classifier trained in a few minutes on CPU. The proposed strategies implied a tuning of a small number of parameters, the Feature Extractor with SVM model being the easiest to adjust. Using a CNN trained with synthetic data, this work shows that it is not necessary to have a large amount of well-marked real data to build a good fault detector with supervised techniques. However, results tend to mimic the interpretation, and their quality thus depends on the quality of the marking. One important step of the method is the preparation of the real dataset. The automatic patch size extraction method should allow adapting our work to other datasets with little effort.

FFT and FE-SVM are the models that had better performance in test data, where FFT has better metrics and higher training time, and FE-

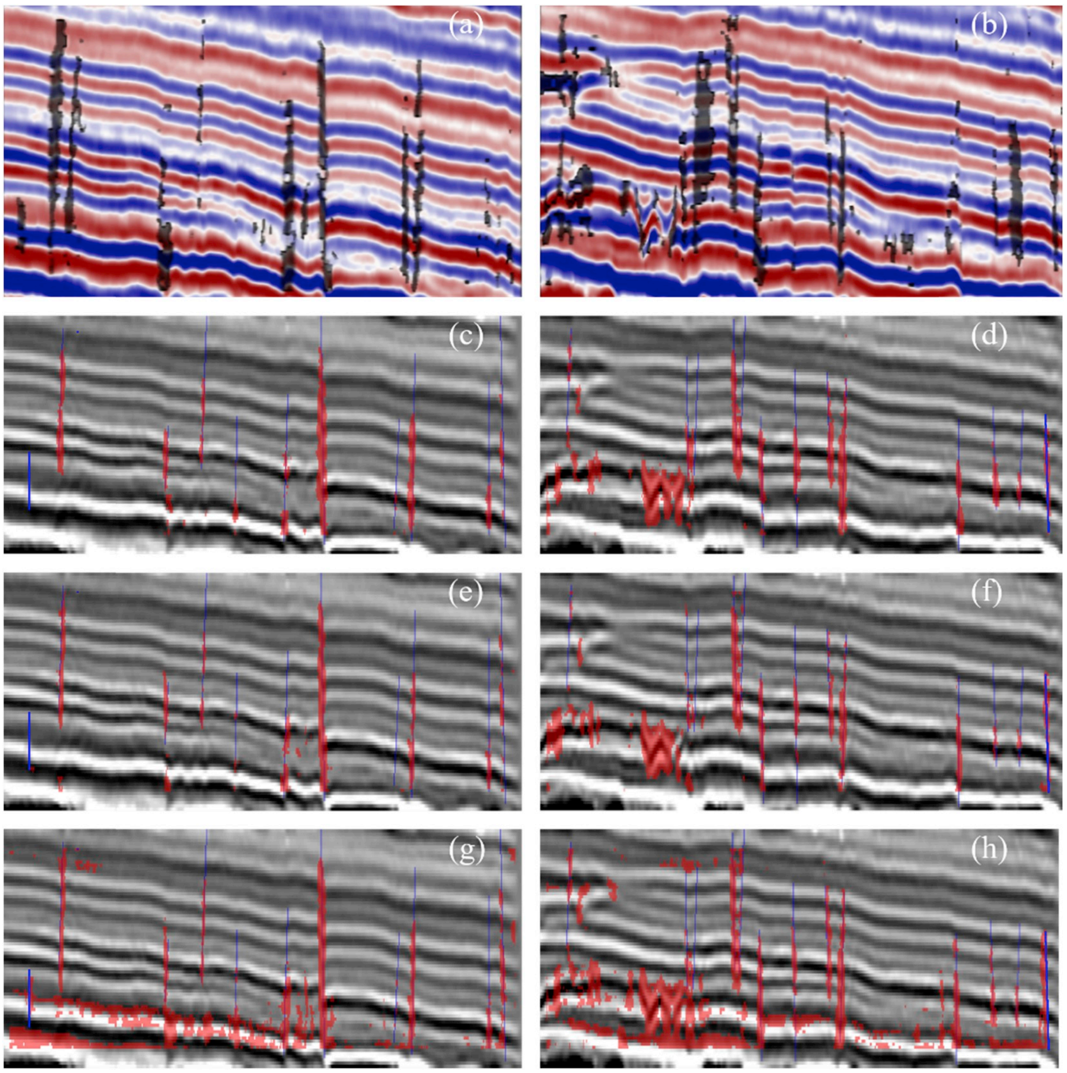

**Fig. 9.** Comparison in GSB inlines 1796 and 2011, (a) (b) are results of Haibin Di et. Al. 2018, (c) (d) are our results with FFT, (e) (f) are our results with SVM, and (g) (h) are our results with MLP. The blue lines are manual interpretation provided by the expert, and the red regions are classified regions by our network.

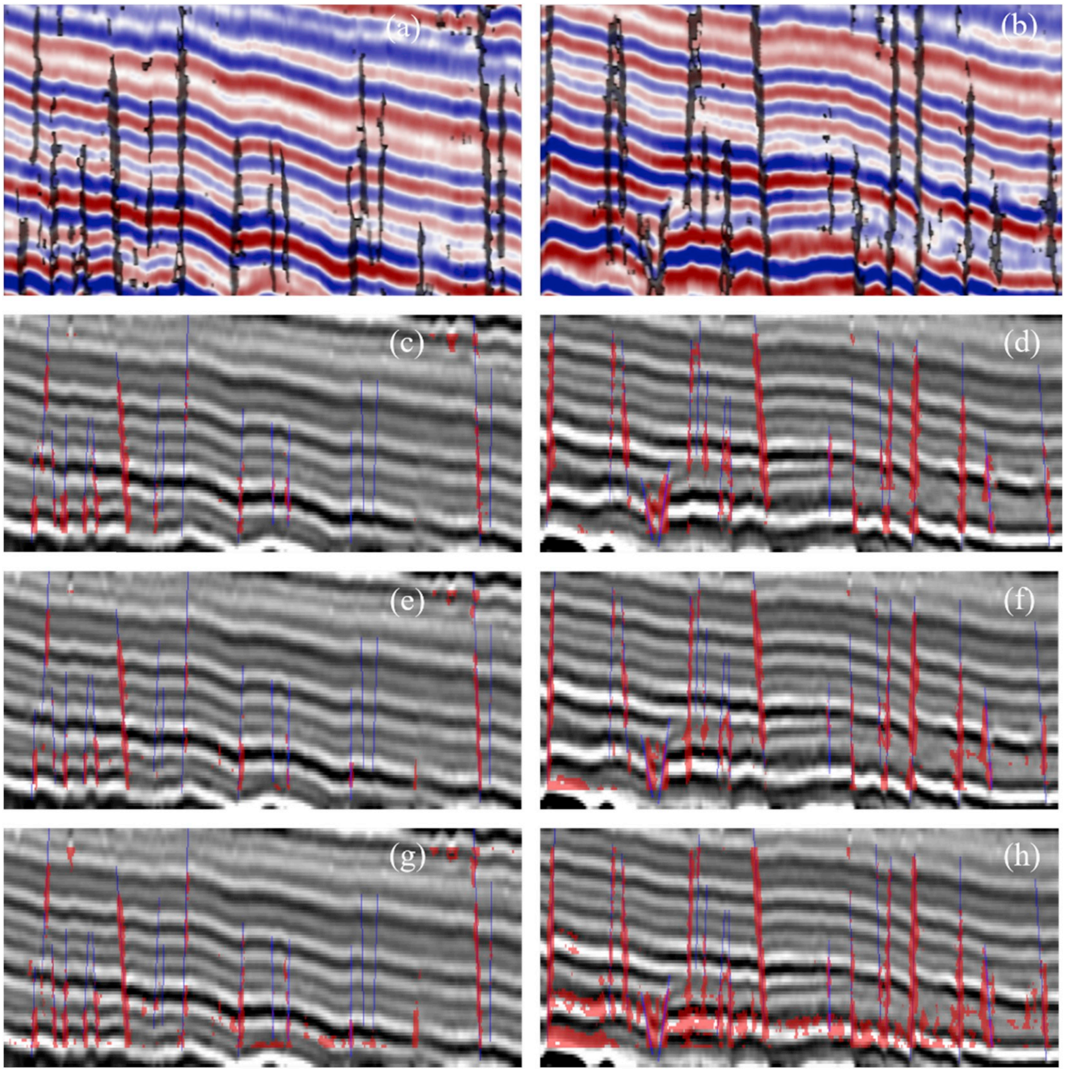

**Fig. 10.** Comparison in GSB crosslines 2600 and 3000, (a) (b) are results of Haibin Di et. Al. 2018, (c) (d) are our FFT results, (e) (f) are our SVM results, (g) (h) are our MLP results. The blue lines are manual interpretation provided by the expert, and the red regions are classified regions by our network.

**Table 4**
Quantitative evaluation of benchmark and TL models on GSB inline 1796.

| Model | Accuracy | Sensitivity | Specificity | F1 Score | AUC |
|---|---|---|---|---|---|
| Benchmark | 90.8 | 10.4 | 99.5 | 18.2 | 67.2 |
| FE-SVM | 93.2 | 33.4 | 99.6 | 49.0 | 84.2 |
| FE-MLP | 87.9 | 44.1 | 92.7 | 41.8 | 84.5 |
| FFT | 93.6 | 36.2 | 99.8 | 52.7 | 94.0 |

**Table 5**
Quantitative evaluation of benchmark and TL models on GSB inline 2011.

| Model | Accuracy | Sensitivity | Specificity | F1 Score | AUC |
|---|---|---|---|---|---|
| Benchmark | 86.7 | 3.0 | 96.7 | 4.6 | 5.9 |
| FE-SVM | 87.2 | 16.2 | 95.7 | 21.4 | 71.4 |
| FE-MLP | 82.2 | 27.5 | 88.8 | 24.9 | 65.8 |
| FFT | 88.9 | 20.5 | 96.1 | 26.9 | 76.7 |

**Table 6**
Quantitative evaluation of benchmark and TL models on GSB crossline 2600.

| Model | Accuracy | Sensitivity | Specificity | F1 Score | AUC |
|---|---|---|---|---|---|
| Benchmark | 82.6 | 4.3 | 99.5 | 8.1 | 54.2 |
| FE-SVM | 85.2 | 18.7 | 99.5 | 31.0 | 81.9 |
| FE-MLP | 85.7 | 27.3 | 98.3 | 40.4 | 81.7 |
| FFT | 85.8 | 21.4 | 99.7 | 34.9 | 88.7 |

**Table 7**
Quantitative evaluation of benchmark and TL models on GSB crossline 3000.

| Model | Accuracy | Sensitivity | Specificity | F1 Score | AUC |
|---|---|---|---|---|---|
| Benchmark | 82.6 | 20.5 | 99.8 | 33.9 | 72.8 |
| FE-SVM | 85.8 | 40.3 | 98.4 | 55.3 | 87.0 |
| FE-MLP | 83.3 | 55.1 | 91.2 | 58.9 | 82.7 |
| FFT | 87.9 | 45.3 | 99.7 | 61.9 | 92.2 |

SVM has slightly lower metrics and lower training time. The classification of an input section can be interpreted as a fault attribute to help the interpreter identify faults faster. Although the good results with sections (2D input), when we classify the entire seismic cube, as shown in Netherland offshore F3 block computed time slice, we can observe fault discontinuities due to the detection of partial faults or not detected faults. 3D fault discontinuities are a limitation if the goal is to generate the fault volume or the fault surfaces.

We could improve the method by investigating different solutions. First, we could, of course, increase the size of the real dataset by classifying another section. Increase the size of the synthetic dataset, their fault complexity, and their frequency range. Evaluate model with dice metric to create better metrics for unbalanced datasets. Other pre-trained models could also be studied. As shown in (Wang et al., 2018), there are many ways to build the main classifier, using synthetic or real data and even pre-stacked data. In addition, famous, powerful pre-trained classification models like VGG-Net (Simonyan and Zisserman, 2014) are easily available online, and it would be interesting to compare the TL performance using such models, trained on a tremendous amount of images from the ImageNet database (Deng et al., 2009), with TL from models specifically designed for fault detection as presented here. Also, other TL strategies exist and should be investigated. Finally, interpret both sections for training instead of only one should improve the detection results, or at least help discarding some false positives.

## Authors contributions

Augusto Cunha and Axelle Pochet worked on the idea development, implementation of all steps and results interpretation. Hélio Lopes and Marcelo Gattass supervised the project and provided critical reviewing.

## Acknowledgments

The authors gratefully acknowledge support from Shell Brazil through the “Coupled Geomechanics” project at Instituto Tecgraf/PUC-Rio and the strategic importance of the support given by ANP "Compromisso com Investimentos em Pesquisa e Desenvolvimento" through the R&D levy regulation. Also, NZP&M for provided Great South Basin seismic data, and the Exxon Mobil for acquired the data.

## Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.cageo.2019.104344.

## References

Al-Dossary, S., Marfurt, K., 2006. 3D volumetric multispectral estimates of reflector curvature and rotation. Geophysics 71 (5), P41–P51.
Araya-Polo, M., Dahlke, T., Frogner, C., Zhang, C., Poggio, T., Hohl, D., 2017. Automated fault detection without seismic processing. Lead. Edge 36 (3), 208–214.
Bahorich, M., Farmer, S., 1995. 3-D seismic discontinuity for faults and stratigraphic features: the coherence cube. Lead. Edge 14 (10), 1053–1058.
Barnes, A.E. (Ed.), 2016. Handbook of Poststack Seismic Attributes. Society of Exploration Geophysicists.
Conners, R.W., Trivedi, M.M., Harlow, C.A., 1984. Segmentation of a high-resolution urban scene using texture operators. Comput. Vis. Graph Image Process 25 (3), 273–310.
Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L., 2009. ImageNet: a large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition.
Di, H., Gao, D., 2014. Gray-level transformation and Canny edge detection for 3D seismic discontinuity enhancement. Comput. Geosci. 72, 192–200.
Di, H., Gao, D., 2017a. Seismic attribute-aided fault detection in petroleum industry: a review. In: Martin, D. (Ed.), Fault Detection: Methods, Applications and Technology. Nova Science Publicher Inc., pp. 53–80
Di, H., Shafiq, M., AlRegib, G., 2017b. Seismic-fault Detection Based on Multiattribute Support Vector Machine Analysis. SEG Technical Program Expanded Abstracts, 2017.
Di, H., Wang, Z., AlRegib, G., 2018. Seismic fault detection from post-stack amplitude by convolutional neural networks. In: 80th EAGE Conference and Exhibition.
Gao, D., 2013. Integrating 3D seismic curvature and curvature gradient attributes for fracture characterization: methodologies and interpretational implications. Geophysics 78 (2), O21–O31.
Glorot, X., Bordes, A., Bengio, Y., 2011. Deep sparse rectifier neural networks. In: Proceedings of the 14th International Conference on Artificial Intelligence and Statistics.
Guitton, A., Wang, H., Trainor-Guitton, W., 2017. Statistical Imaging of Faults in 3D Seismic Volumes Using a Machine Learning Approach. *SEG Technical Program Expanded Abstracts*, p. 2017.
Hale, D., 2014. Seismic image processing for geologic faults. https://github.com/dhale/ipf.
Haralick, R.M., Shanmugam, K., Dinstein, I., 1973. Textural features for image classification. In: IEEE Transactions on Systems, Man and Cybernetics, vol. 3. SMC, pp. 610–620, 6.
Huang, L., Dong, X., Clee, T., 2017. A scalable deep learning platform for identifying geologic features from seismic attributes. Lead. Edge 36 (3), 249–256.
Kortström, Jari, Uski, Marja, Tiira, Timo, 2016. Automatic classification of seismic events within a regional seismograph network. Comput. Geosci. 87, 22–30.
Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. Adv. Neural Inf. Process. Syst. 1097–1105.
Kruskal, J.B., 1964. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. Psychometrika 29 (1), 1–27.
Lisle, R.J., 1994. Detection of zones of abnormal strains in structures using Gaussian curvature analysis. AAPG (Am. Assoc. Pet. Geol.) Bull. 78.
Luo, Y., Higgs, W., Kowalik, W., 1996. Edge detection and stratigraphic analysis using 3D seismic data. SEG Tech. Progr. Expand. Abstr. 1996.
Marfurt, K., Kirlin, R., Farmer, S., Bahorich, M., 1998. 3-D seismic attributes using a semblance-based coherency algorithm. Geophysics 63 (4), 1150–1165.
Opendtect. Netherland offshore F3 block complete. https://www.opendtect.org/osr/Main/NetherlandsOffshoreF3BlockComplete4GB.
Palafox, Leon F., Hamilton, Christopher W., Scheidt, Stephen P., Alvarez, Alexander M., 2017. Automated detection of geological landforms on Mars using convolutional neural networks. Comput. Geosci. 101, 48–56.
Pan, S.J., Yang, Q., 2010. A survey on transfer learning. IEEE Trans. Knowl. Data Eng. 22 (10), 1345–1359.
Pedersen, S.I., Randen, T., Sonneland, L., Steen, Ø., 2002. Automatic Fault Extraction Using Artificial Ants. *SEG Technical Program Expanded Abstracts*, 2002.
Pochet, A., Diniz, P.H.B., Lopes, H., Gattass, M., 2019. Seismic Fault Detection Using Convolutional Neural Networks Trained on Synthetic Poststacked Amplitude Maps. IEEE Geosci. Remote Sens. Lett. 16 (3), 352–356. https://doi.org/10.1109/LGRS.2018.2875836. In this issue.
Randen, T., Pedersen, S., Sønneland, L., 2001. Automatic Extraction of Fault Surfaces from Three-dimensional Seismic Data. SEG Technical Program Expanded Abstracts, 2001.
Rijks, E., Jauffred, J., 1991. Attribute extraction: an important application in any detailed 3-D interpretation study. Lead. Edge 10 (9), 11–19.
Roberts, A., 2001. Curvature attributes and their application to 3D interpreted horizons. First Break 19 (2), 85–100.
Simonyan, K., Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. In: Proc. International Conference on Learning Representations.
Souza, J.F.L., Santos, M.D., Magalhães, R.M., Neto, E.M., Oliveira, G.P., Roque, W.L., 2019. “Automatic classification of hydrocarbon “leads” in seismic images through artificial and convolutional neural networks. Comput. Geosci. 132, 23–32.
Tang, Y., 2013. Deep learning using linear support vector machines. In: Workshop on Representational Learning. ICML.
Tingdahl, K., de Rooij, M., 2005. Semi-automatic detection of faults in 3D seismic data. Geophys. Prospect. 53 (4), 533–542.
Van Bemmel, P., Pepper, R., 2000. Seismic signal processing and apparatus for generating a cube of variance values. US Patent 6151555.
Wang, Z., Alregib, G., 2017. Interactive fault extraction in 3-D seismic data using the Hough Transform and tracking vectors. In: IEEE Transactions on Computational Imaging, vol. 3, pp. 99–109 no. 1.
Wang, Z., Temel, D., Alregib, G., 2014. Fault Detection Using Color Blending and Color Transformations. IEEE Global Conference on Signal and Information Processing (GlobalSIP), 2014.
Wang, Z., Di, H., Shafiq, M.A., Alaudah, Y., Alregib, G., 2018. Successful leveraging of image processing and machine learning in seismic structural interpretation: a review. Lead. Edge 37 (6), 451–461.
Wu, Xinming, Zhu, Zhihui, 2017. Methods to enhance seismic faults and construct fault surfaces. Comput. Geosci. 107, 37–48.
Yan, Zhe, Liu, Shaoyong, Gu, Hanming, 2019. Fault image enhancement using a forward and backward diffusion method. Comput. Geosci. 131, pp1–14.
Zhang, C., Frogner, C., Araya-Polo, M., Hohl, D., 2014. Machine-learning Based Automated Fault Detection in Seismic Traces, vol. 2014. EAGE Conference and Exhibition.