Extracting horizon surfaces from 3D seismic data using deep learning

Valentin Tschannen, Matthias Delescluse, Norman Ettrich, Janis Keuper

Extracting horizon surfaces from 3D Seismic Data using Deep Learning

Article *in* Geophysics · March 2020

DOI: 10.1190/geo2019-0569.1

4 authors, including:

Matthias Delescluse
Ecole Normale Supérieure de Paris
52 PUBLICATIONS **577** CITATIONS
SEE PROFILE

Janis Keuper (Fehr)
Offenburg University of Applied Sciences
89 PUBLICATIONS **501** CITATIONS
SEE PROFILE

GEOPHYSICS, VOL. 85, NO. 3 (MAY-JUNE 2020); P. N17–N26, 14 FIGS., 2 TABLES.
10.1190/GEO2019-0569.1

Extracting horizon surfaces from 3D seismic data using deep learning

Valentin Tschannen[1], Matthias Delescluse[2], Norman Ettrich[3], and Janis Keuper[4]

ABSTRACT

Extracting horizon surfaces from key reflections in a seismic image is an important step of the interpretation process. Interpreting a reflection surface in a geologically complex area is a difficult and time-consuming task, and it requires an understanding of the 3D subsurface geometry. Common methods to help automate the process are based on tracking waveforms in a local window around manual picks. Those approaches often fail when the wavelet character lacks lateral continuity or when reflections are truncated by faults. We have formulated horizon picking as a multiclass segmentation problem and solved it by supervised training of a 3D convolutional neural network. We design an efficient architecture to analyze the data over multiple scales while keeping memory and computational needs to a practical level. To allow for uncertainties in the exact location of the reflections, we use a probabilistic formulation to express the horizons position. By using a masked loss function, we give interpreters flexibility when picking the training data. Our method allows experts to interactively improve the results of the picking by fine training the network in the more complex areas. We also determine how our algorithm can be used to extend horizons to the prestack domain by following reflections across offsets planes, even in the presence of residual moveout. We validate our approach on two field data sets and show that it yields accurate results on nontrivial reflectivity while being trained from a workable amount of manually picked data. Initial training of the network takes approximately 1 h, and the fine training and prediction on a large seismic volume take a minute at most.

INTRODUCTION

A key step in seismic interpretation is the mapping of the main horizons in the amplitude volume. Horizons are reflection surfaces visible in the data that present a similar character in terms of wavelet shape throughout the survey. Mapping the reflections enables us to analyze the amplitudes to scan for potential fluid anomalies. Highlighting the horizons is also essential to understand how and when the observed geologic structures were formed. At later interpretation stages, surfaces are used to tie the seismic to well logs to relate seismic reflections to actual geologic interfaces and to perform a depth conversion for building geomodels. Dorn (1998) explains that working with full 3D data, rather than with sparse 2D lines, is essential when dealing with complex geology. Typical surveys contain hundreds of inlines and crosslines, which makes the manual interpretation of these surfaces a time-consuming task. For this reason, autotracking tools were developed to help interpreters (Bacon et al., 2003). Working with an autopicker is usually an iterative process in which the interpreter starts by dropping seed points on the desired reflection and gives some key information such as the waveform phase or the expected maximal vertical deviation between two adjacent traces. The tracker uses those hints to extract a 2D surface from the 3D data by finding related waveforms between traces using similarity measures. More seeds are progressively added, and the

Manuscript received by the Editor 23 August 2019; revised manuscript received 8 February 2020; published ahead of production 13 March 2020; published online 08 April 2020.
[1]Fraunhofer Center for Machine Learning and Industrial Mathematics (ITWM), Competence Center for High Performance Computing, Kaiserslautern, Germany and PSL Research University, École Normale Supérieure, UMR 8538, Paris, France. E-mail: valentin.tschannen@itwm.fraunhofer.de (corresponding author).
[2]PSL Research University, École Normale Supérieure, UMR 8538, Paris, France. E-mail: delescluse@geologie.ens.fr.
[3]Fraunhofer Center for Machine Learning and Industrial Mathematics (ITWM), Competence Center for High Performance Computing, Kaiserslautern, Germany. E-mail: norman.ettrich@itwm.fraunhofer.de.
[4]Offenburg University, Institute for Machine Learning and Analytics, Offenburg, Germany. E-mail: janis.keuper@hs-offenburg.de.

completion usually requires the interpreter to finish the most difficult portions manually. Although autopickers work well on high-amplitude reflections with a consistent waveform across the data, in complex structural areas, they usually fail to track across faults with a large throw or to follow weak and chaotic reflections. Many authors have proposed tracking algorithms that try to alleviate those limitations. Gersztenkorn and Marfurt (1999) and Lomask and Guitton (2007) suggest computing 3D structural attributes to highlight faults and stratigraphic elements not readily apparent in the seismic data. Aurnhammer and Toennies (2002) show how one could further constrain the autotracking by incorporating information of the main faults in the area. Alternatively, Pauget et al. (2009) and Wu and Fomel (2018) propose global approaches in which they treat the mapping as an optimization problem and solve it for all the reflections at once. In practice, local approaches often result in misties when the tracks from independent seeds are merged, and global approaches suffer from the curse of dimensionality and might require a lot of manual interventions to constrain the problem in regions containing nonstratified objects such as salt or gas chimneys and chaotic depositions (Hoyes and Cheret, 2011). In addition, traditional similarity measures, such as crosscorrelation, are sensitive to coherent noise in the data (e.g., acquisition and migration artifacts or interference from multiples) and perform poorly in regions with a low signal-to-noise ratio.

In recent years, there has been a big resurgence of interest around the field of deep learning, and in particular, convolutional neural networks (CNNs), to tackle computer vision and waveform analysis problems. Those methods have proven to outperform traditional signal processing and other machine-learning techniques on a large panel of applications (LeCun et al., 2015). Several authors use neural networks for tracking horizons (Harrigan et al., 1992; Kusuma and Fish, 1993; Veezhinathan et al., 1993; Alberts et al., 2000; Leggett et al., 2003), but modern computing power and the accumulation of empirical findings have permitted the emergence of deep neural networks that possess greater potential. Recent applications of deep learning to seismic interpretation include salt classification (Waldeland et al., 2018; Shi et al., 2019), fault detection (Huang et al., 2017; Xiong et al., 2018; Wu et al., 2019; Zheng et al., 2019), diffraction picking (Tschannen et al., 2019), and seismic lithofacies classification (Liu et al., 2019).

Horizon picking is a typical pattern recognition problem, and deep learning is therefore a logical choice to approach it. Peters et al. (2019) use a 2D CNN to track horizons using only a few manual picks to train the algorithm and show that the network can accurately predict the position of the surfaces in field data. In this work, we use a similar approach, but we propose a more in-depth study of the problem and aim to provide a practical methodology for interpreters. We work with a 3D CNN and propose two detailed and challenging case studies, in which we identify the strengths and limitations associated with the use of neural networks to pick horizons. We present a practical and robust workflow to segment several horizons at once in a large seismic volume. Our method does not require any special assumption on the character of the seismic wavelet nor on the spatial continuity of the horizons. We design a 3D-CNN architecture inspired by Ronneberger et al. (2015), which processes data over multiple scales and yields a high-resolution prediction. Because we treat the spatial and temporal dimensions differently, our network provides a stable interpretation while keeping the memory and computational requirements to a workable level. We express the initial manual picks provided by the interpreter as probabilities, and we use the cross-entropy loss to train the network in a supervised manner. The probabilistic formulation allows for uncertainties in the exact location of the predicted horizons. Given a simple masking of the loss function, interpreters have the freedom to label the training data using either seed points or 1D- and 2D-line interpretations. By keeping the total number of free parameters of the CNN small and by using regularization and data augmentation, we show that our method requires only reasonable manual work to prepare the training data. The initial network training time is approximately 1 h, and the prediction on a large seismic survey takes only seconds. We also show how the interpreter can interactively fine train the network by picking a few additional examples in the most complex areas. Finally, we show that our method can be used to extend horizons to the prestack domain to perform higher-quality amplitude analysis. We verify the validity of our approach on two marine data sets that exhibit challenges because of the presence of faults and weak reflections.

METHODS

Semantic segmentation

After prestack time (depth) migration, seismic data are represented as a multidimensional array over a regular grid of inline, crossline, and time (or depth) coordinates. Every element of the array is called a voxel and holds the value of the wavefield amplitude at this position. In the image processing community, given a set of preestablished categories or labels, classification refers to the association of an image with one label. Segmentation goes beyond classification and refers to the association of every pixel of an image with a category. By analogy, with seismic data, we refer to one sample as a patch (i.e., a small cube) extracted from the global volume. Classification aims to associate one class to the entire sample. This is, for instance, the approach taken by Waldeland et al. (2018) to pick salt bodies. To obtain the final prediction on the entire seismic data, Waldeland et al. (2018) assume that the label corresponds to the central voxel of the patch and apply the network on overlapping patches extracted around every voxel of the volume. In segmentation, we also use patches for training, but the network associates one class to every voxel of each patch. This is similar to the approach taken in Wu et al. (2019) and Tschannen et al. (2019) to pick faults and diffracting objects.

Supervised deep learning is now established as the reference method to tackle such problems because it leads to the best results on a wide variety of applications (LeCun et al., 2015). The field of deep learning is almost entirely focused around neural networks, which are a set of algorithms expressed as a computational graph built from a sequence of layers performing simple operations. Rather than being manually engineered, the parameters of the transformations are initially chosen at random and given the freedom to adapt to the data and problem at hand. Chaining several layers is a key feature of these algorithms because this architectural design allows the algorithm to learn a hierarchical representation of the data. The deeper layers build upon the work of the previous layers and are sensitive to progressively more abstract and complex features, expressed as a composition of the simpler features learned by the shallower layers (LeCun et al., 2015). When the data exhibit a spatial structure and the surrounding information is relevant to understand the local context, a suitable choice for the linear transformations is convolutions, and the family of algorithms based on them is called CNNs (LeCun et al., 1998).

In this work, we aim to segment a set of K horizons in a 3D seismic stack. Let $\mathbf{h}(il, xl, t)$ be a seismic sample expressed in the 3D coordinate system (il, xl, t), where il and xl are the spatial coordinates in the inline and crossline directions and t is the vertical temporal coordinate. We express the labels as a 4D hypercube $\mathbf{p}(il, xl, t, K+1)$ representing the probable locations of the K target reflectors. For every voxel, $\mathbf{p}$ holds the discrete probability density function $(p_k \geq 0)_{k=1..K+1}$, such that $\sum_{k=1}^{K+1} p_k = 1$. The terms p_1 to p_K are the probabilities of each horizon, whereas p_{K+1} is the probability of not being any of the desired reflectors.

A neural network depending on the parameters $w \in \mathbb{R}^m$, where m is the number of free dimensions, takes as an input the seismic data $\mathbf{h}$ and outputs a pseudoprobability density function $\hat{\mathbf{p}}_w(il, xl, t, K+1)$ defined at every voxel. Training the network consists of optimizing the values of its parameters to increase its prediction performance so that $\hat{\mathbf{p}}_w$ approaches $\mathbf{p}$. A standard pseudodistance measure between two probability distributions is cross entropy, which measures how close the computed distribution is to represent the *true* distribution. In its discrete form, the cross entropy between $\mathbf{p}$ and $\hat{\mathbf{p}}_w$, summed over space and time, is

$$l(w; p, \hat{p}_w) = -\sum_{il,xl,t} \sum_{k=1}^{K+1} p(il, xl, t, k) \times \log[\hat{p}_w(il, xl, t, k)]. \quad (1)$$

For a training data set $\mathcal{D}$ composed of N pairs of samples $(\mathbf{h}^{(i)}, \mathbf{p}^{(i)})$, the training loss is defined as the average over all samples of the loss computed in equation 1:

$$\mathcal{L}(w; \mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} l(w; p^{(i)}, \hat{p}_w^{(i)}). \quad (2)$$

The most commonly chosen approach to minimize the loss function of equation 2 is to resort to an optimizer belonging to the minibatch stochastic gradient-descent family (Robbins and Monro, 1951; LeCun et al., 2015). In this iterative procedure, a random subset of the training data set (called a minibatch) is chosen at every iteration, and the local steepest-descent direction is found by computing the gradients of the loss function with respect to the network's parameters using the back-propagation algorithm (Werbos, 1974). The parameters are updated by taking a step toward this direction, and a new minibatch is selected for the next iteration. The learning rate is an important hyperparameter that defines the step size used for the update. When all minibatches have been seen once by the network, we say that it has been trained for one epoch.

Network architecture

Figure 1 and Table 1 present the architecture of our network. It is a traditional feed-forward convolutional network inspired by Ronneberger et al. (2015). The trainable parameters are contained in the convolutional layers, which transform the data by convolving the input with kernels and adding bias coefficients. Convolving with a single kernel yields a 3D feature map. Because every layer contains several kernels, the output of each layer is 4D with the fourth dimension corresponding to the number of channels (i.e., the number of kernels in the layer). We use rectified linear units $(\max(0, .))$ as activation functions for the output of the convolutional layers. To learn abstract concepts, the network should see the data at different scales.

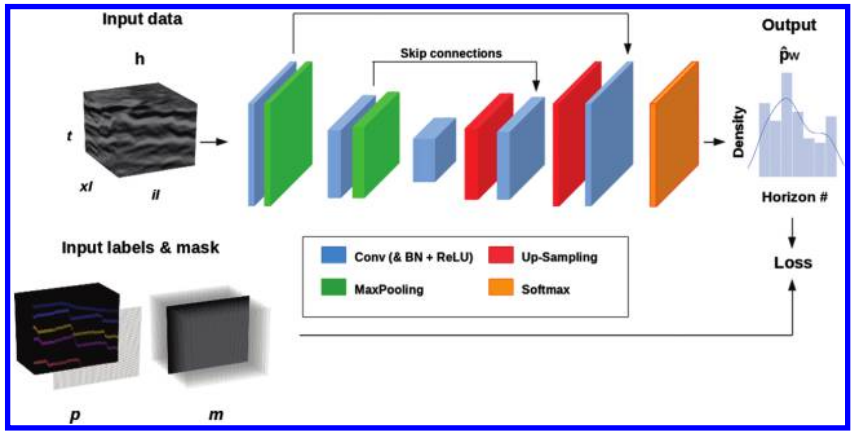

Figure 1. Simplified overview of the 3D CNN used in this work. The data flow is from left to right during the forward pass as indicated by the black arrows. The boxes represent multichannel 4D feature maps (here drawn in 3D for simplicity) color coded by layer type. In this example, the input data $\mathbf{h}$ are interpreted along their central crossline and the mask $\mathbf{m}$ contains ones at the location of the picks along the crossline slice and zeros elsewhere. For simplicity, the output hypercube is represented as a histogram counting the number of voxels associated with each horizon. The exact architecture of the CNN is described in Table 1.

Table 1. Architecture of our CNN, designed after Ronneberger et al. (2015)[5].

Input: Seismic Patch	
01	Conv (5×5×5, 24), BN, ReLU
02	MaxPool (1×1×2)
03	Conv (5×5×5, 24), BN, ReLU
04	MaxPool (1×1×2)
05	Conv (5×5×5, 48), BN, ReLU
06	MaxPool (2×2×2)
07	Conv (5×5×5, 96), BN, ReLU
08	Up-Sampling (2×2×2)
09	Concat; Conv (5×5×5, 48), BN, ReLU
10	Up-Sampling (1×1×2)
11	Concat; Conv (5×5×5, 24), BN, ReLU
12	Up-Sampling (1×1×2)
13	Concat; Conv (5×5×5, 24), BN, ReLU
18	Dropout (30%)
19	Conv (5×5×5, $K+1$), Softmax
Output: Predicted Patch	

[5]Conv stands for convolution, upsampling is performed by nearest neighbor interpolation, BN stands for batch normalization (Ioffe and Szegedy, 2015), ReLU for rectified linear unit, and Maxpool for max-pooling. The size and number of convolutional kernels in each layer are indicated in parentheses. The down and upsampling factors are indicated in parentheses next to the corresponding operations. The blue arrows indicate the skip connections that concatenate the activation maps, along the fourth dimension, which consists of a shallow convolution with its symmetric upsampled counterpart.

The receptive field refers to the effective window size that a convolutional layer is accessing from the original data. For the first layer, the receptive field is equal to the spatial size of the kernels. We use a constant kernel size of $5 \times 5 \times 5$ over the network, and we use max-pooling operations to progressively downsample the data. Max pooling is a sliding window operation that keeps the largest value inside the window and drops the others. A pooling window of $1 \times 1 \times 2$ keeps the inline and crossline dimensions unchanged and downsamples the time dimension by a factor of two. Accordingly, the effective receptive field of the next convolutional layer is twice as large in the vertical direction as in the horizontal ones. Because we aim to classify every voxel of the input data, the network must yield an output that has the same spatial shape as the input. The first half of the network is downsampling the data by a desired rate, and the second half is a symmetric counterpart that progressively transforms the data back to their original spatial shape (see Figure 1). We use nearest neighbor interpolation to perform the upsampling. To compensate for the loss of information caused by the successive pooling layers, we use skip connections (Ronneberger et al., 2015) to inject data from shallow layers to deeper ones (see Table 1).

To choose the number of layers, there is a trade-off between the largest data scale accessible to the network and the memory and computational costs. We consider that for the horizon picking problem, it is more important to access low-frequency information along time than along the inline and crossline dimensions. For this reason, as detailed in Table 1, we alternate between $1 \times 1 \times 2$ and $2 \times 2 \times 2$ downsampling factors. The three pooling layers yield to a total downsampling of $2 \times 2 \times 8$. To regularize the training, we apply batch normalization (Ioffe and Szegedy, 2015) and dropout (Srivastava et al., 2014) and learn the parameters with the Adam optimizer (Kingma and Ba, 2014) using the library TensorFlow (Abadi et al., 2016).

Training data and prediction

When picking horizons, the interpreter needs to build an understanding of the geometry of the area by establishing fault patterns and inferring the depositional and tectonic history of the site. Once the reflectors of interest have been identified, they need to be picked in a dense 3D grid. Most workstations allow this process to be partially automated by letting the interpreter provide information to the tracking algorithm of the reflections that should be followed and iteratively refining the results by making manual adjustments and adding constraints in the mispicked areas. The quality of the automatic tracking will have a big impact on the time needed to complete the task. For our supervised deep-learning approach, we use the picks of the interpreter to create the training labels. As explained in the previous section, the labels should be provided as a hypercube in which the fourth dimension holds the reflector probabilities. For every manually picked horizon, we set the probability of the corresponding voxels to one in the label's hypercube.

To introduce uncertainty in the exact position of the reflectors, we convolve the labels with a normalized 1D Gaussian kernel in the vertical direction (see Figure 2). A difficulty of this approach is that one needs to label every voxel of the training data, which is a tedious task when working with 3D seismic. However, a common solution is to use a masked loss that allows labeling only a subset of the voxels without affecting the training quality (Xu et al., 2015; Peters et al., 2019). In addition to providing the labels, one also defines a binary mask $\mathbf{m}(il, xl, t)$ containing ones for the voxels that are explicitly marked by the interpreter and zeros elsewhere. For instance, if one decides to label an entire inline section, the mask would be a cube of zeros for every inline except for the horizons interpreted on the single inline section in which the mask would contain ones for these voxels (looking at Figure 2, we set to one every voxel of the mask that corresponds to a nonzero probability for at least one of the horizons). If one wishes to highlight horizons with seed points, the mask would contain ones only at the seed locations. Because we incorporate uncertainties in the exact position of the picks by convolving with a Gaussian kernel, we also convolve the binary mask with the same kernel and set to one all voxels whose value is greater than 0.1. The loss function of equation 1 is then modified by masking the cross entropy by an element-wise multiplication:

$$l(w; p, \hat{p}_w) = -\sum_{il,xl,t} m(il, xl, t) \times \sum_{k=1}^{K+1} p(il, xl, t, k) \log[\hat{p}_w(il, xl, t, k)]. \quad (3)$$

In this way, during training, the gradients used to update the parameters will be nonzero only in the picked areas.

Once the network is trained, we run a prediction on the entire seismic stack to obtain a hypercube containing the probability density function for the presence of the horizons at every voxel. Because convolutions and other transformations of our network do not depend on the input data size, we can evaluate the network on data of arbitrary dimensions (Long et al., 2014). Because the entire 3D stack may not

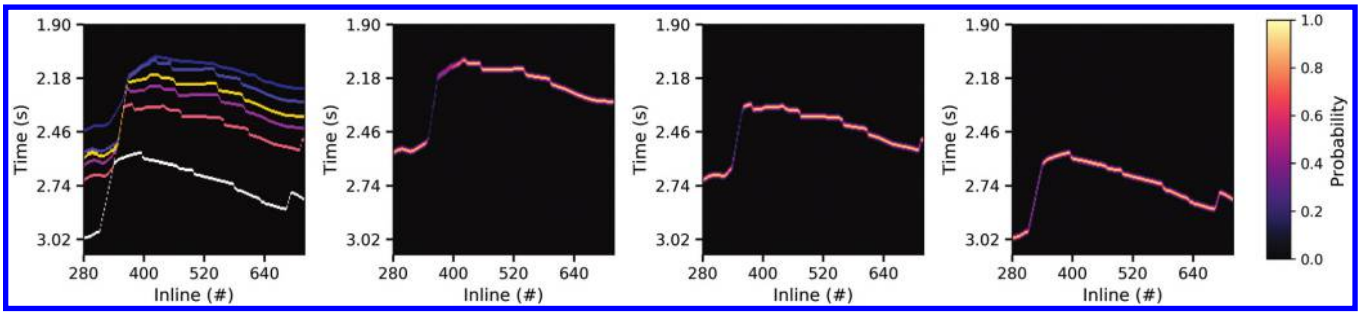

Figure 2. Preparation of the labels for training. The left image shows the picks performed on a 2D section for the different horizons that we want to map. The images on the right are three of the probability slices obtained by convolving in the vertical dimension the picks of individual horizons with a 1D Gaussian kernel. The final labels are created by concatenating every probability slices along an extra dimension. The last slice of the labels corresponds to the "other" class, and its values are chosen such that the total distribution sums to one.

fit in memory, we perform the evaluation in chunks. We split the volume into subcubes, run the segmentation iteratively, and merge the probabilities at the end. To extract an actual 2D horizon surface s_k from the hypercube, we take the position of the maximum probability along the vertical dimension as indicated in equation 4 (see Figure 3):

$$s_k(il, xl) = \underset{t}{\text{argmax}}[\hat{p}_w(il, xl, t, k)]. \quad (4)$$

Because storing the full 4D probability cube can occupy an excessive amount of disk space, one may instead extract the surfaces during the iterative segmentation and only store the horizons. The final horizons are obtained after despiking and smoothing postprocessing procedures.

RESULTS

We evaluate our method on two marine surveys and compare the results of the machine with the interpretations proposed by experts. We create the training data sets by selecting several interpreted lines. We could use seed points instead of full 2D interpretations, but in this way we get more training examples at a minimum extra of manual effort. Seed points may also not be enough to guide the algorithm for difficult reflections that have a low signal-to-noise ratio and that may have a fairly heterogeneous character over the survey. For such reflections, interpreters use their experience and intuition to draw the lines. We also select one interpreted line in a different region for the validation set of each survey. We normalize the seismic volume to the amplitude range $[-1, 1]$, and we extract samples around the training and validation lines with a shape of $[\Delta il, \Delta xl, \Delta t] = [32, 32, 96]$. For both experiments, we use a learning rate of 5×10^{-4} and reduce its value by 33% every 10 epochs. We train the network for 25 epochs with a batch size of 12 samples. To artificially increase the size of the training data sets, we perform a simple data augmentation (Simard et al., 2003) by flipping the inline and crossline directions and by adding white noise with a standard deviation of 0.025 to the input seismic image.

Faults and fine training

The first data set is a stack of 301 inlines × 201 crosslines × 301 time samples with a discretization of 25 m × 25 m × 4 ms. It contains six horizons of interest in a faulted area, and a manual interpretation serves as a reference. We use four inlines to create the training data set. The training takes 47 min on a TITAN X GPU. The final prediction takes less than a minute by splitting the stack in overlapping chunks of 32 inlines. Figure 3 shows the results on two cross sections (not used to prepare the training data) for the top reservoir. The probability attribute can be used to determine areas where the network's prediction is uncertain. The bottom section, for instance, shows that on the other side of the tectonic event (for crosslines less than 1020) is a region of low certainty, although the network is nevertheless able to find the reflection. Figure 4 displays all of the horizons on a crossline section. Although the extracted surfaces do not exactly match the original interpretation, we deem them to be of good quality for a

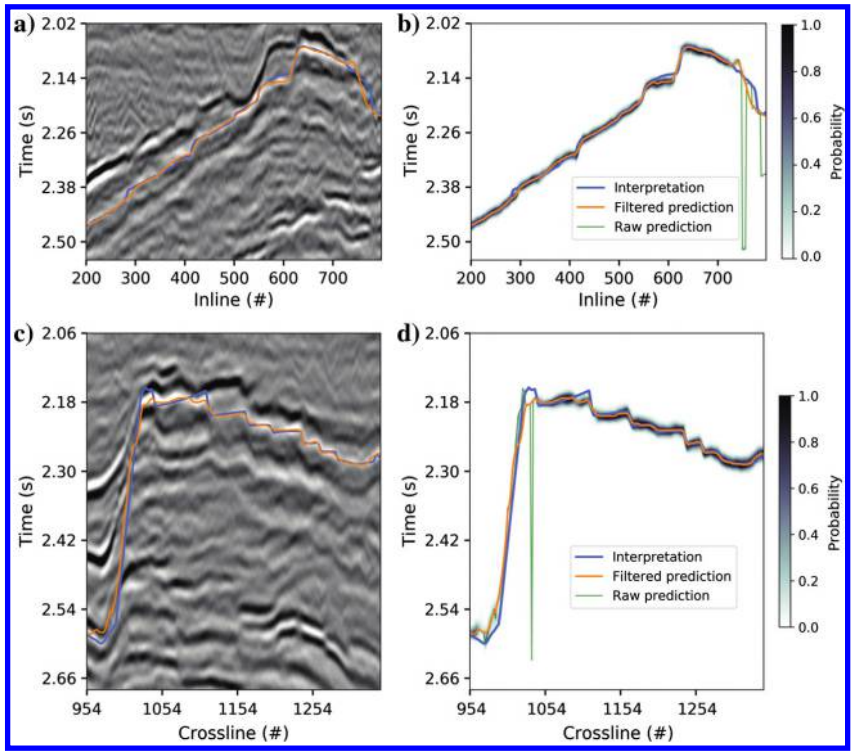

Figure 3. The 2D sections at (a) crossline 1200 and (c) inline 500 through the data and their corresponding probability cube for the top reservoir in (b and d). The interpretation and the machine's prediction are displayed, and we also show the prediction before despiking in semitransparency.

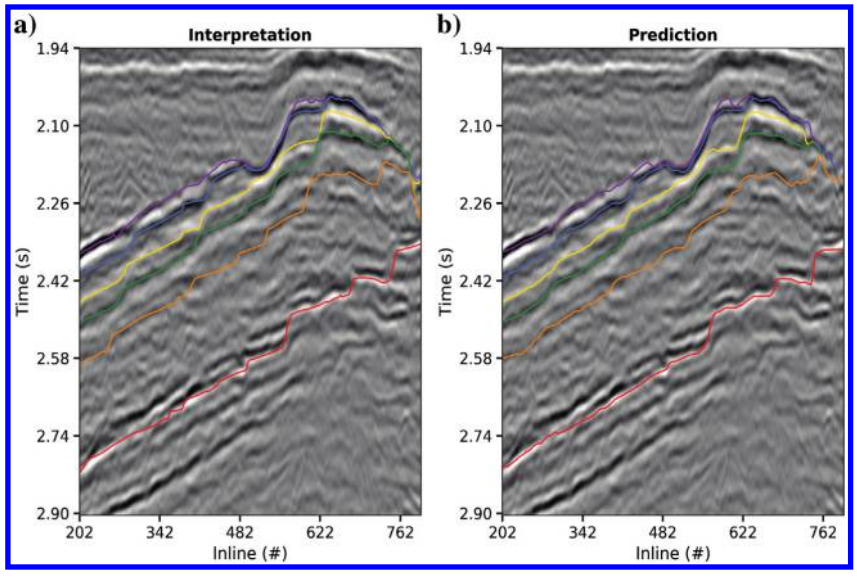

Figure 4. The 2D section at crossline 1200 shows (a) the interpretation and (b) the machine prediction for the six horizons.

first training pass. Even the relatively weak reflectors, such as the reflection highlighted by the orange horizon (the fifth deepest), are picked. Figure 5 shows the top reservoir. Regions of low values in the probability surface highlight the faults and the tectonic folding. By construction of the neural network, we do not impose special assumptions on the lateral continuity of the reflectors and, in particular, fault discontinuities are, in theory, not a problem for the prediction. We see that the faults are better defined on the expert's interpretation. This might be explained by the fact that it was done by hand and that the interpreter picked the reflection from both sides of the discontinuity and filled the gap by linear interpolation, leading to a sharp transition.

If one is not completely satisfied by the results of the first training pass, one can fine-tune the network training. Figures 6 and 7 show the effect of such fine training. In the area marked by the red rectangle in Figure 6a, we place 10 seed points to extract additional training examples. We create a new training data set composed of 10 samples extracted around the seed points, as well as 50 samples randomly chosen from the original training set. We further train the network for approximately 2 min with a learning rate 10 times smaller than the original one. We use a much smaller learning rate because the network is already trained, and we only want to fine tune its weights. We also incorporate samples from the original data set to limit overfitting. This phenomenon happens when the training set is too small and the network becomes overly specialized at recognizing the training data and performs poorly in other areas of the survey. For this reason, we also only reevaluate the fine-trained network in a small region around the seed points. After rerunning the segmentation, we observe that the prediction follows the interpretation more closely.

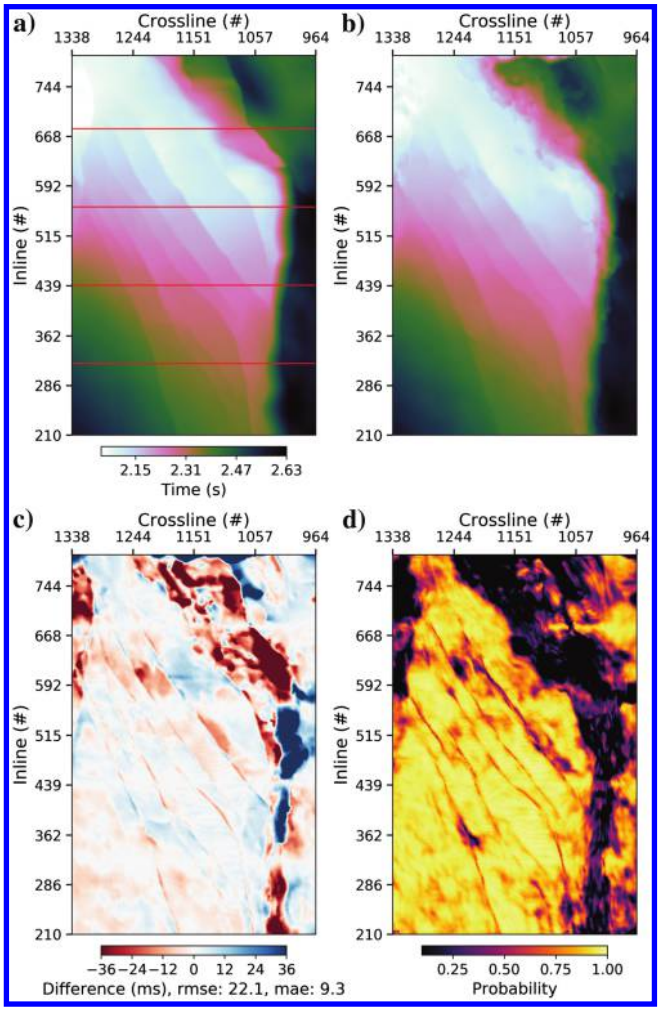

Figure 5. Map views of the top reservoir. (a) Solution provided by the interpreters; the training lines are shown in red. (b) Horizon predicted by our neural network. (c) Difference between the interpretation and the prediction. We also report the root-mean-square error (rms error) and mean absolute error (MAE). (d) Probability associated with the picks.

Extension to prestack seismic data

The second data set consists of prestack angle gathers of dimension 999 inlines × 699 crosslines × 30 angles × 241 time samples with a discretization of 12.5 m × 12.5 m × 2° × 4 ms. It contains six horizons of interest, around a reservoir, interpreted with a combination of handpicking and crosscorrelation-based autotracking. We use one inline and one crossline from a 2° to 12° near-angle stack to create the training data set, and we train the network for 1.03 h. The evaluation runs in less than a minute by splitting the volume in overlapping chunks of 32 inlines. Figures 8 and 9 show the results, obtained by predicting on the near stack, on a crossline section away from the training lines. The autopicking results are similar to the reference baseline, and differences observed on the weaker reflectors are subject to discussion with experts.

In addition, we study how sensitive the network is to changes in the wavelet in the prestack domain. We apply the trained network iteratively on all angle planes from 2° to 60°. Each angle plane is a 3D volume with the same dimensions as the near stack, and the final prediction yields 3D prestack horizons that depend on the incidence angle. We focus on the top of the reservoir shown in Figure 10 and display the corresponding horizon in Figures 11 and 12. We see that the network is correctly picking the horizon up to a certain angle before losing it once the waveform becomes too different from the one observed in the near stack. Extending horizons to the prestack domain is useful to perform an improved amplitude versus angle analysis because the gradient is strongly sensitive to moveout effects.

DISCUSSION

CNNs present several advantages to tackle the horizon picking problem. They do not require strong prior information to operate, and they can adapt to any seismic data. For instance because they work by scanning the entire volume and do not expect a reflection to be found within a certain vertical distance between two neighboring traces, they are appropriate to follow a signal in a faulted area or across a steeply dipping event. In Figure 5d, we observe that the confidence of the network is low at the faults because the reflection at these exact locations is not well-defined, but it nevertheless follows the horizon across the fault blocks. CNNs also have good scalability with respect to the number of reflections to be predicted. When increasing the number of horizons, one only needs to increase the number of channels in the last convolutional layer by the same amount, which results in a minor growth in computational cost.

The fact that our network performs a segmentation, instead of a classification, of the input data is also an advantage. Indeed, whereas segmentation networks need to see each voxel only once, classification networks associate one label to the center of a fixed-size input

patch, and obtaining a prediction over the entire volume requires us to evaluate the network on a few overlapping patches equal to the number of voxels in the data. For standard seismic stacks containing 10^6–10^9 voxels, this induces a considerable overhead that can lead to an evaluation time much greater than the training time itself. Because the evaluation time of our segmentation approach is small, it is suitable to run many successive segmentation on offset or angle planes to follow a reflection in the prestack domain, as shown in Figure 11.

The multidimensional and multiscale nature of CNNs also make them robust classifiers. In Figure 13, we experiment with a 1D network that only analyzes traces along time and observe that, given the same training data, the performance is worse than with the corresponding 3D version. The limited resolution of seismic data, and the various sources of noise, often lead to uncertainties in the interpretation. As such, we believe that exploiting deep learning in an interactive manner, by giving the interpreter the possibility to refine the results by progressively adding new examples to the training data set, is an important part of the presented workflow. By using a masked loss, we give flexibility to the interpreter to pick examples using either seed points or line interpretations, without worrying about the 3D aspects of the algorithm.

However, there are also challenges in applying deep learning to the horizon interpretation problem. Neural networks are dependent on the quantity and quality of the training data (LeCun et al., 2015). They can adapt to any data set and do not suffer from prior-induced limitations, but they need to be given enough examples before correctly generalizing. Although we show that our method only requires a reasonable amount of training labels, the approach still strongly relies on manual interpretation by an expert. For a seismic volume containing many horizons, providing the initial training samples is an obstacle to overcome. The workflow is also penalized by a slow start because the neural network needs to be trained to convergence before obtaining the firsts results. In Wu et al. (2019) and Tschannen et al. (2019), the authors use synthetic modeling to create large training data sets and train networks that can generalize to real data. However, in this application we aim to pick specific reflections

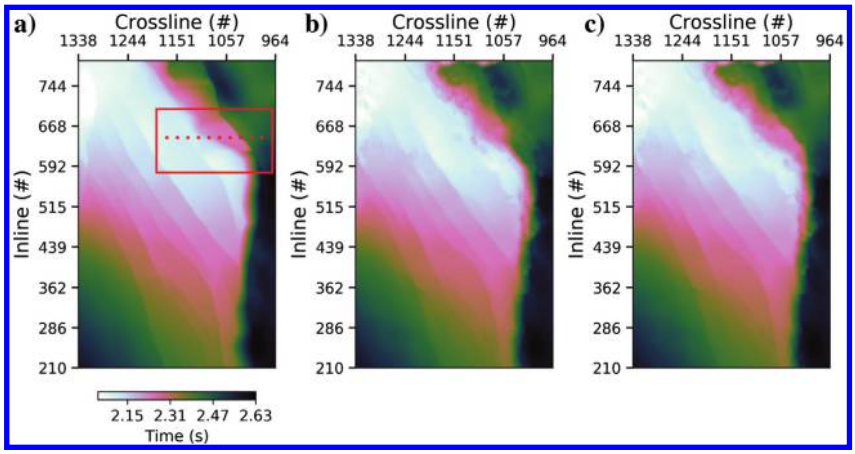

Figure 6. Map views of the top reservoir. (a) Ground truth provided by the interpreters. (b) Horizon picked by our neural network after the first pass of training. (c) Prediction, updated only inside of the red square, after fine-tuning the network. The rms error, computed inside the red square between the interpretation and the prediction, decreased from 36.9 to 15.5 and the MAE from 18.8 to 9.7. The 10 seed points used to fine-train the network are shown by the red dots.

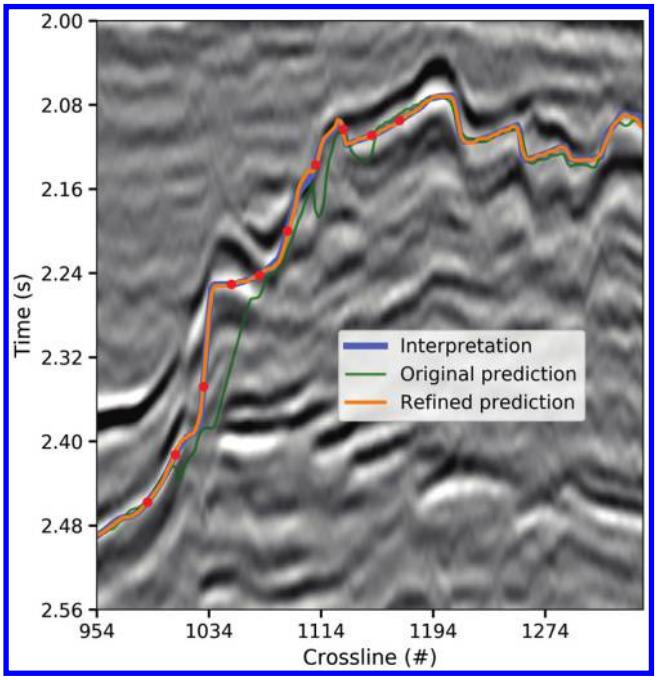

Figure 7. The 2D section at inline 646. We plot the interpretation, as well as the original and refined predictions are shown in Figure 6. The seed points used for fine-training are displayed as red dots.

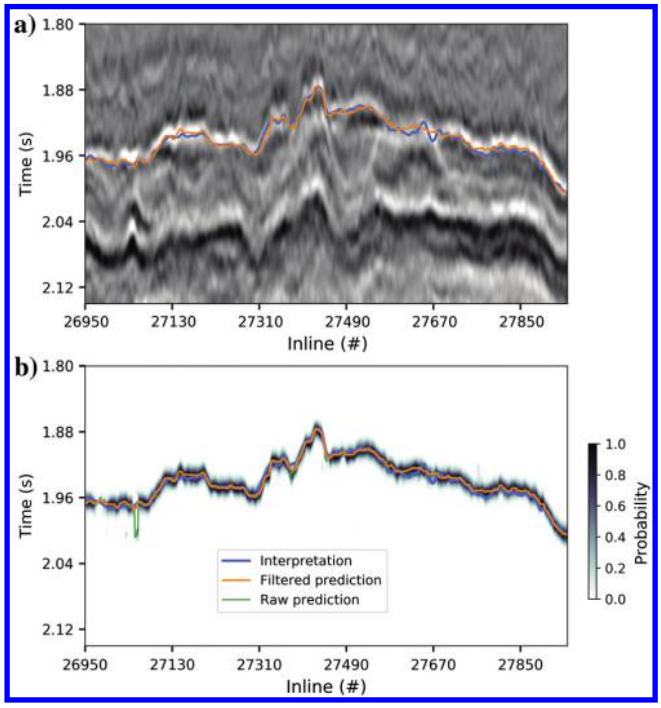

Figure 8. The 2D section at crossline 16301 of (a) the near-angle stack and (b) the probability cube for the top reservoir. The interpretation and the machine's prediction are displayed.

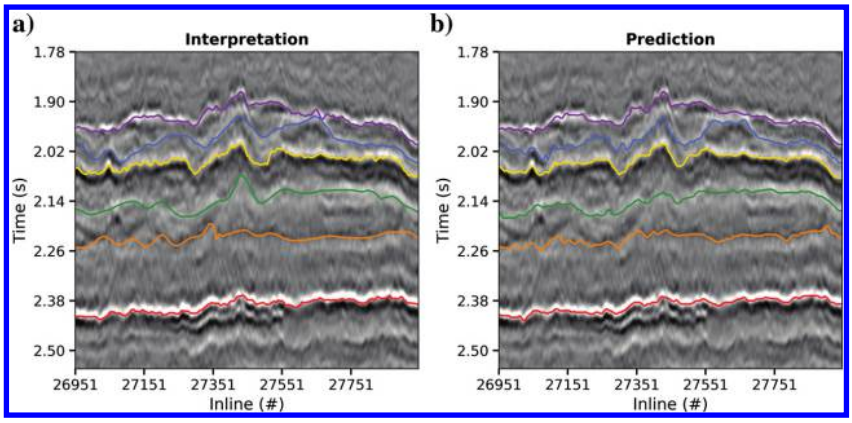

Figure 9. The 2D section at crossline 16301 of the near-angle stack shows (a) the interpretation and (b) the machine prediction for the six horizons.

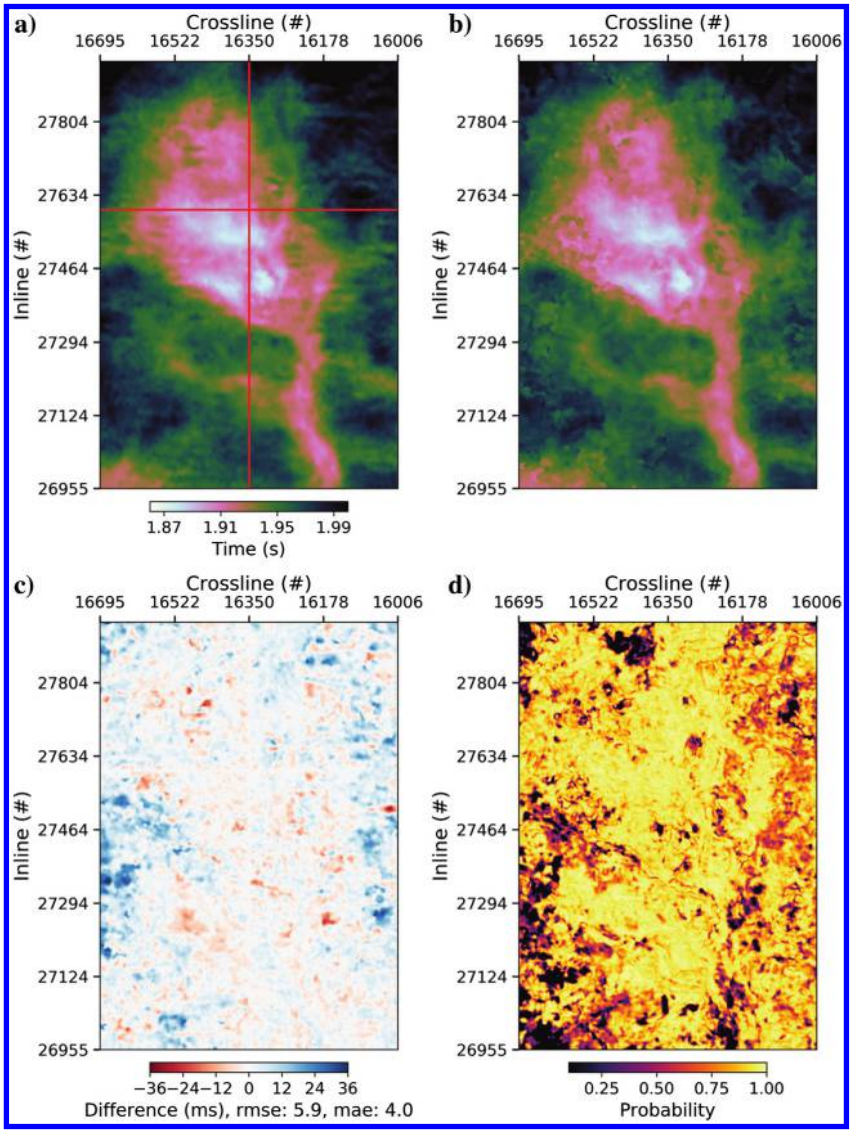

Figure 10. Map views of the top reservoir for the near stack. (a) Solution provided by the interpreters; the training lines are shown in red. (b) Horizon predicted by our neural network. (c) Difference between the interpretation and the prediction. (d) Probability associated with the picks.

and we cannot rely on the exploitation of more abstract and generalizable concepts such as faults or diffractions. By borrowing elements from state-of-the-art architectures (Ioffe and Szegedy, 2015; Ronneberger et al., 2015) and designing the network to have a smaller reach in the inline and crossline directions than in the time direction, we keep the computational and memory requirements low.

Another difficulty lies in the empirical nature of deep learning. Training a CNN requires the tuning of several hyperparameters. To find these parameters, practitioners usually rely on a performance metric evaluated on a validation data set. However in geosciences, labeled data are scarce and sometimes noisy. In our experiments, monitoring the training process using a single test line is not always very informative, and we also find it helpful to use a more qualitative assessment by carefully visualizing the predicted horizons together with the seismic data. We show in Figure 14 the evolution of the training and validation errors with the number of epochs for the first data set. We see that the validation error decreases with the number of epochs in a similar fashion as the training error. This behavior is a good sign that indicates low overfitting and is in accordance with the good quality, on average, of the prediction shown in Figure 5b. However, horizon interpretation is a task that requires precision, and the geology may rapidly change over a data set. The average value of the validation loss does not inform us directly on how the network is performing in the different areas of the data set. For example, in Figure 7, the prediction (in green) is very close to the interpretation except in the steeply dipping area, between crosslines 1000 and 1140. We also see that the validation error is more volatile and on average a bit larger than the training error, which indicates that the network does not perfectly generalize to the entire survey. Using larger training and validation sets would certainly improve the prediction's quality, but a trade-off must be found because increasing the amount of manual labeling is in contradiction to the automatization that is the purpose of the algorithm. Overall, we find that our architecture is not overly sensitive to key hyperparameters such as the learning rate, the number of kernels per layer, and the number of epochs, and we obtain good results on two different data sets with the same parameterization. As stated above, our solution depends on the training data, and we see for instance in Figure 12 that the prediction does not work when the waveform is too different from the one observed in the training data. In this case, to successfully recognize the reflection beyond the polarity reversal angle, one would need to repeat the training procedure using examples picked on a far stack.

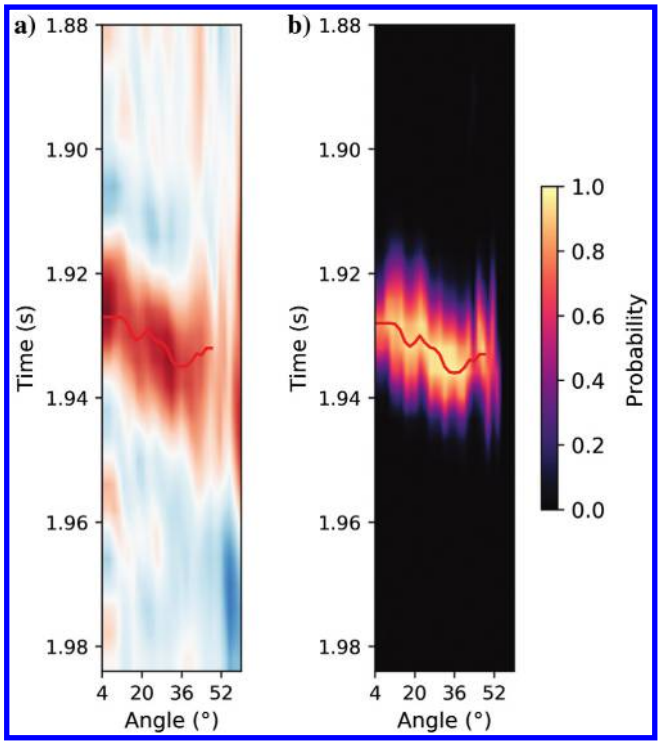

Figure 11. Prediction of the top reservoir across the prestack domain. (a) Angle gather at inline 27208 and crossline 16022; the predicted horizon is drawn in red. The prediction is following the moveout of the reflector up to 40°, and it fails to recognize the event once the waveform becomes too different from the reference near stack, in particular because of stretching. (b) Prediction probability.

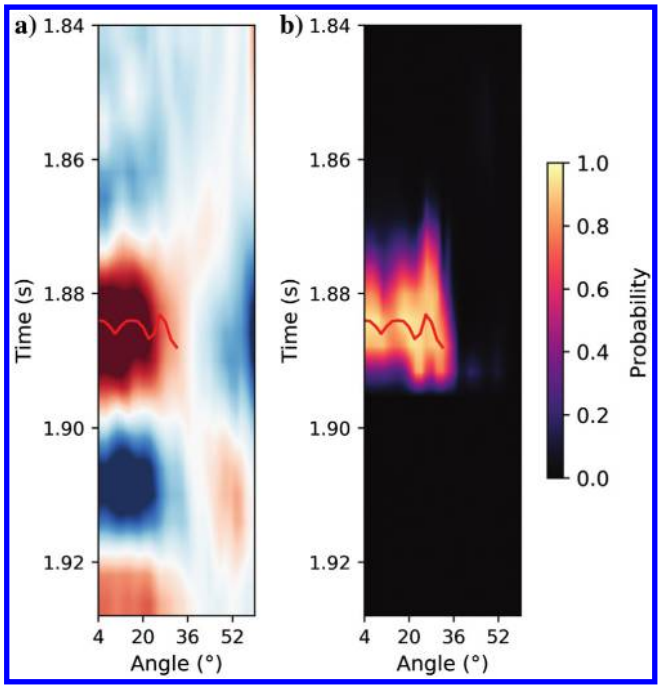

Figure 12. Prediction of the top reservoir across the prestack domain. The setting is similar to Figure 11, for a gather at inline 27600 and crossline 16402. The network correctly picks the reflection until 34°, and it fails to recognize the event once the waveform becomes too different from the reference near the stack because of a polarity reversal.

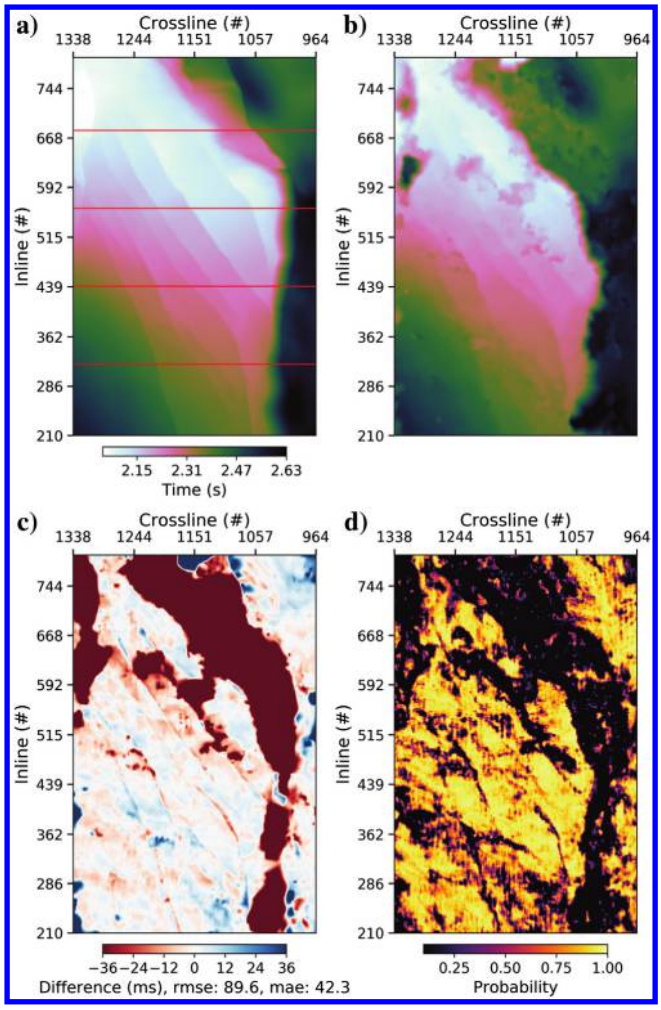

Figure 13. Results obtained for the same horizon as shown in Figure 5 but with a 1D CNN. The architecture of the CNN is the same as the one presented in Table 1, but 3D kernels are replaced by $1 \times 1 \times 5$ kernels. We use training samples of size $1 \times 1 \times 96$ in the inline, crossline, and time dimensions, and we train the network for 40 epochs with the same parameters as for Figure 5.

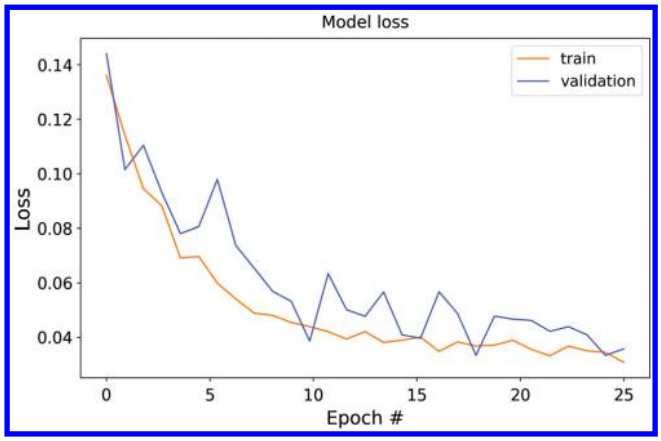

Figure 14. Training and validation errors, for the first data set, monitored over the training epochs.

In addition, CNNs may also suffer from boundary condition artifacts. In Figure 5b, some outliers are visible in the upper part of the predicted surface (at large inlines). Because the network is based on convolutions, performing a prediction on the edge of the volume requires artificially extending the data, using for instance a mirroring condition, which may affect the quality of the output. Finally, although here we treat the multihorizon picking problem, we do not explicitly enforce an ordering of the predictions, and crossings may occur. Because the network is trained using patches, it does not have knowledge of the global coordinate system, and it is nontrivial to make it aware of the relative position of the different samples. We do not address this issue here, and we enforce ordering of the horizons as a postprocessing operation.

CONCLUSION

We have discussed a practical approach to efficiently and simultaneously pick several horizons in a 3D seismic image. We formulate the problem as a segmentation task, in which the position of the reflectors is expressed as a probability distribution, and we use supervised deep learning to solve it. We design an efficient architecture for a 3D CNN that allows us to analyze the data over multiple scales while keeping the memory and computational requirements low. We use a masked loss function to give flexibility to the interpreters in the way they pick the training data. The method requires us to label only a few lines through the survey to yield good initial results, and we show how interpreters can progressively improve the predictions by fine-tuning the network training. Validation on field data shows the potential of the method, and in future work we plan to integrate, within the same end-to-end workflow, the interpretation of other structural features such as faults or salt bodies.

ACKNOWLEDGMENTS

We are grateful to Sharp Reflections for providing us with the data and horizons that we used in this work. We are also grateful for the helpful comments and suggestions provided by three anonymous reviewers.

DATA AND MATERIALS AVAILABILITY

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

REFERENCES

Abadi, M., P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, and M. Isard, 2016, Tensorflow: A system for large-scale machine learning: Proceedings of the 12th Symposium on Operating Systems Design and Implementation, 265–283.

Alberts, P., M. Warner, and D. Lister, 2000, Artificial neural networks for simultaneous multi horizon tracking across discontinuities: 70th Annual International Meeting, SEG, Expanded Abstracts, 651–653, doi: 10.1190/1.1816150.

Aurnhammer, M., and K. D. Toennies, 2002, Horizon correlation across faults guided by geological constraints: Proceedings of the SPIE, **4667**, 312–322, doi: 10.1117/12.467992.

Bacon, M., R. Simm, and T. Redshaw, 2003, 3-D seismic interpretation: Cambridge University Press.

Dorn, G. A., 1998, Modern 3-D seismic interpretation: The Leading Edge, **17**, 1262–1272, doi: 10.1190/1.1438121.

Gersztenkorn, A., and K. J. Marfurt, 1999, Eigenstructure-based coherence computations as an aid to 3-D structural and stratigraphic mapping: Geophysics, **64**, 1468–1479, doi: 10.1190/1.1444651.

Harrigan, E., J. Kroh, W. Sandham, and T. Durrani, 1992, Seismic horizon picking using an artificial neural network: IEEE International Conference on Acoustics, Speech, and Signal Processing, 105–108.

Hoyes, J., and T. Cheret, 2011, A review of global interpretation methods for automated 3D horizon picking: The Leading Edge, **30**, 38–47, doi: 10.1190/1.3535431.

Huang, L., X. Dong, and T. E. Clee, 2017, A scalable deep learning platform for identifying geologic features from seismic attributes: The Leading Edge, **36**, 249–256, doi: 10.1190/tle36030249.1.

Ioffe, S., and C. Szegedy, 2015, Batch normalization: Accelerating deep network training by reducing internal covariate shift: arXiv preprint, arXiv:1502.03167.

Kingma, D. P., and J. Ba, 2014, Adam: A method for stochastic optimization: arXiv preprint, arXiv:1412.6980.

Kusuma, T., and B. C. Fish, 1993, Toward more robust neural-network first break and horizon pickers: 63rd Annual International Meeting, SEG, Expanded Abstracts, 238–241, doi: 10.1190/1.1822449.

LeCun, Y., Y. Bengio, and G. Hinton, 2015, Deep learning: Nature, **521**, 436–444, doi: 10.1038/nature14539.

LeCun, Y., L. Bottou, Y. Bengio, and P. Haffner, 1998, Gradient-based learning applied to document recognition: Proceedings of the IEEE, **86**, 2278–2324, doi: 10.1109/5.726791.

Leggett, M., W. A. Sandham, and T. S. Durrani, 2003, Automated 3-D horizon tracking and seismic classification using artificial neural networks: Springer Netherlands, 31–44.

Liu, Z., J. Cao, Y. Lu, S. Chen, and J. Liu, 2019, A seismic facies classification method based on the convolutional neural network and the probabilistic framework for seismic attributes and spatial classification: Interpretation, **7**, no. 3, SE225–SE236, doi: 10.1190/INT-2018-0238.1.

Lomask, J., and A. Guitton, 2007, Volumetric flattening: An interpretation tool: The Leading Edge, **26**, 888–897, doi: 10.1190/1.2756869.

Long, J., E. Shelhamer, and T. Darrell, 2014, Fully convolutional networks for semantic segmentation: arXiv e-prints arXiv:1411.4038.

Pauget, F., S. Lacaze, and T. Valding, 2009, A global approach in seismic interpretation based on cost function minimization: 79th Annual International Meeting, SEG, Expanded Abstracts, 2592–2596, doi: 10.1190/1.3255384.

Peters, B., E. Haber, and J. Granek, 2019, Neural networks for geophysicists and their application to seismic data interpretation: The Leading Edge, **38**, 534–540, doi: 10.1190/tle38070534.1.

Robbins, H., and S. Monro, 1951, A stochastic approximation method: The Annals of Mathematical Statistics, **22**, 400–407, doi: 10.1214/aoms/1177729586.

Ronneberger, O., P. Fischer, and T. Brox, 2015, U-Net: Convolutional networks for biomedical image segmentation: arXiv e-prints arXiv:1505.04597.

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

Simard, P. Y., D. Steinkraus, and J. C. Platt, 2003, Best practices for convolutional neural networks applied to visual document analysis: 7th International Conference on Document Analysis and Recognition, 958–963.

Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, 2014, Dropout: A simple way to prevent neural networks from overfitting: The Journal of Machine Learning Research, **15**, 1929–1958.

Tschannen, V., N. Ettrich, M. Delescluse, and J. Keuper, 2019, Detection of point scatterers using diffraction imaging and deep learning: Geophysical Prospecting, **68**, 830–844, doi: 10.1111/gpr.v68.3.

Veezhinathan, J., F. Kemp, and J. Threet, 1993, A hybrid of neural net and branch and bound techniques for seismic horizon tracking: Proceedings of the ACM/SIGAPP Symposium on Applied Computing, ACM, 173–178.

Waldeland, A. U., A. C. Jensen, L.-J. Gelius, and A. H. S. Solberg, 2018, Convolutional neural networks for automated seismic interpretation: The Leading Edge, **37**, 529–537, doi: 10.1190/tle37070529.1.

Werbos, P., 1974, Beyond regression: New tools for prediction and analysis in the behavioral sciences: Ph.D. thesis, Harvard University.

Wu, X., and S. Fomel, 2018, Least-squares horizons with local slopes and multigrid correlations: Geophysics, **83**, no. 4, IM29–IM40, doi: 10.1190/geo2017-0830.1.

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

Xiong, W., X. Ji, Y. Ma, Y. Wang, N. M. AlBinHassan, M. N. Ali, and Y. Luo, 2018, Seismic fault detection with convolutional neural network: Geophysics, **83**, no. 5, O97–O103, doi: 10.1190/geo2017-0666.1.

Xu, J., A. G. Schwing, and R. Urtasun, 2015, Learning to segment under various forms of weak supervision: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 3781–3790.

Zheng, Y., Q. Zhang, A. Yusifov, and Y. Shi, 2019, Applications of supervised deep learning for seismic interpretation and inversion: The Leading Edge, **38**, 526–533, doi: 10.1190/tle38070526.1.