OXFORD

Journal of Geophysics and Engineering

Journal of Geophysics and Engineering (2019) **16**, 175–189 doi:10.1093/jge/gxy015

Seismic fault detection using an encoder–decoder convolutional neural network with a small training set

Shengrong Li [1,2,3], **Changchun Yang** [1,2,*], **Hui Sun** [1,2,3] **and Hao Zhang** [1,2]

[1] Key Laboratory of Petroleum Resources Research, Institute of Geology and Geophysics, Chinese Academy of Sciences, Beijing 100029, China
[2] Institutions of Earth Science, Chinese Academy of Sciences, Beijing 100029, China
[3] University of Chinese Academy of Sciences, Beijing 100049, China

[*] Corresponding author: Changchun Yang. E-mail: ccy@mail.iggcas.ac.cn

Received 13 August 2018, revised 27 October 2018
Accepted for publication 30 November 2018

Abstract

In seismic interpretation, fault detection is a crucial step that often requires considerable manual labor and time. The convolutional neural network (CNN) is state-of-the-art deep learning technology that can perform even better than humans at image recognition. However, traditional methods of using CNNs for prediction require a very large dataset to train the network, which is impractical for common researchers and interpreters in geophysics who have difficulty obtaining sufficient quantities of labeled real data. In this paper, we propose a method for seismic fault detection using a CNN that requires only a very small training set. We treat the fault detection process as a semantic segmentation task and train an encoder–decoder CNN, namely, a U-Net, to perform a pixel-by-pixel prediction on the seismic section to determine whether each pixel is a fault or non-fault. Using this type of CNN in the experiments, we obtain good prediction results on real data. When interpreting a new seismic volume with the proposed method, interpreters need only to pick and label several 2D sections; subsequently, the model can predict faults in any other section of the same volume, greatly improving the interpretation efficiency. To evaluate the performance of the proposed method, we introduce a fault detection accuracy index that describes the accuracy of the prediction results. In this paper, we show that using only seven seismic sections of a seismic volume to train a CNN can allow us to predict faults successfully in any other section of the same volume.

Keywords: fault detection, interpretation, CNN, deep learning, real data

1. Introduction

Because faults may indicate the locations of petroleum reservoirs, seismic fault detection is an important task in seismic interpretation. However, traditional methods require interpreters to trace and pick faults manually. This process consumes substantial amounts of manual work and time; as a result, the interpretation efficiency is very low. For example, for an experienced interpreter, it may take anywhere from weeks to months to label faults within a typically sized seismic volume. To improve the interpretation efficiency, numerous researchers have proposed various methods. Because faults indicate discontinuities in seismic sections, several methods use seismic attributes, such as the coherence (Marfurt *et al.* 1999), variance (Van Bemmel & Pepper 2000) and curvature (Boe & Daber 2010), to detect faults. However, fault detection by only a few seismic attributes is limited.

Therefore, to achieve better results, researchers have proposed other methods, such as the ant tracking algorithm (Silva *et al.* 2005) and many other automated and semi-automated methods (Cohen *et al.* 2006; Hale 2013; Wu & Hale 2016a, 2016b; Wang & AlRegib 2017; Wu & Zhu 2017; Wu & Fomel 2018).

In recent years, the rapid development of machine learning (Bishop 2006), especially deep learning (LeCun *et al.* 2015; Schmidhuber 2015), has made it possible to process big data in a structured manner and has also greatly increased the efficiency of data processing. Inspired by the outstanding performances of machine learning and deep learning models in various fields, such as computer vision, many researchers have attempted to apply these newly developed technologies to geophysics, such as in full-waveform inversion (FWI) (Lewis & Vigh 2017) and wave field separation (Serfaty *et al.* 2017). Accordingly, machine learning and deep learning technologies are showing great potential in geophysics.

In some recent studies, researchers have introduced these state-of-the-art technologies into seismic fault detection tasks. As an alternative to traditional machine-learning methods (Guitton *et al.* 2017; Hami-Eddine & de Ribet 2017), some methods were proposed that use deep neural networks (DNNs) to map raw input seismic data directly to faults in 2D (Zhang *et al.* 2014; Dahlke *et al.* 2016). In addition, the Wasserstein distance was introduced to the loss function of the network to address space-dependent fault-detection tasks (Frogner *et al.* 2015; Araya-Polo *et al.* 2017). The convolutional neural network (CNN) (LeCun *et al.* 1998) is one of the most popular DNNs, and it has achieved substantial success in many tasks. It has been proven that CNNs can perform even better than humans at image classification (Krizhevsky *et al.* 2012), speech recognition and so on. Moreover, some recent work has shown promising results regarding the application of the CNN to seismic interpretation (Wu *et al.* 2018; Xiong *et al.* 2018).

Universal approximation theorem (Hornik *et al.* 1989) states that a neural network with a sufficient number of parameters can approximate any continuous function. However, training a large number of network parameters requires a large number of samples; thousands or even millions. For common researchers and interpreters in geophysics, a very large amount of manually labeled real data is usually beyond reach. Therefore, in work involving seismic fault detection using deep learning, researchers always use generated synthetic data rather than real data as training sets to train DNNs. However, such an approach has notable disadvantages. For example, DNNs trained by synthetic data were used in most studies to predict marine data, but the performances of DNNs in predicting land data, which have a much lower signal-to-noise ratio, are still unknown. Accordingly, in this paper, we propose a method that requires only a small dataset to train an encoder–decoder CNN based on the U-Net architecture (Ronneberger *et al.* 2015) for seismic fault detection. Therefore, we can use real data as the training set without any concerns about insufficient data. In practical applications, interpreters need only to pick and label several 2D seismic sections to train the network; then the well-trained network can accurately predict faults in any other section of the same seismic volume.

In this paper, we begin by explaining our methodology consisting of the encoder–decoder CNN based on the U-Net architecture. Next, we introduce the workflow of the proposed seismic fault detection method, after which we discuss our method and directions for future work. Finally, we conclude our findings at the end of this paper.

2. Methodology of the encoder–decoder CNN

In this paper, we use an encoder–decoder CNN to detect faults. Traditional methods detect faults by carefully designing algorithms to compute seismic attributes. In our workflow, we convert a 2D seismic section into a 2D image consisting of pixels and use a CNN to learn to automatically extract the proper image features for fault detection. In addition, we treat fault detection as a semantic segmentation task in computer vision and make a prediction for each pixel in the image. We use U-Net, which exhibits a good performance in semantic segmentation tasks, as the encoder–decoder CNN architecture of the proposed method.

2.1. *CNN model and semantic segmentation*

In computer vision, CNNs have become one of the most popular methods for processing images. Filters can be used to convolve images and extract features from them; for example, a Sobel filter can be employed to extract edge information from the image (Aqrawi & Boe 2011). In simple terms, a CNN uses a variety of different-scale, parameter-trainable filters to convolve the input image and intermediate results to extract different-scale features from the image. If we define $z_i \in \mathbb{R}^{m_i \times n_i \times c_i}$ as the output of layer i of the CNN with dimensions of $m_i \times n_i$ and c_i channels, the relationship between the output z_i^j of a single channel j of layer i (called a feature map) and the output of the previous layer can be given by

$$z_i^j = \sigma\left(\sum_{k=1}^{c_{i-1}} h_{ijk} * z_{i-1}^k + b_i^j\right), \tag{1}$$

where $\boldsymbol{h} * \boldsymbol{z}$ is a 2D convolution operation of an image $\boldsymbol{z}$ with a filter $\boldsymbol{h}$, $\boldsymbol{b}$ is a bias and σ is a nonlinear activation function such as rectified linear unit (ReLU) (Nair & Hinton 2010).

Currently, following the development of CNNs, the component units of a typical CNN now consist of convolution (CONV), nonlinear operation (ReLU), batch normalization

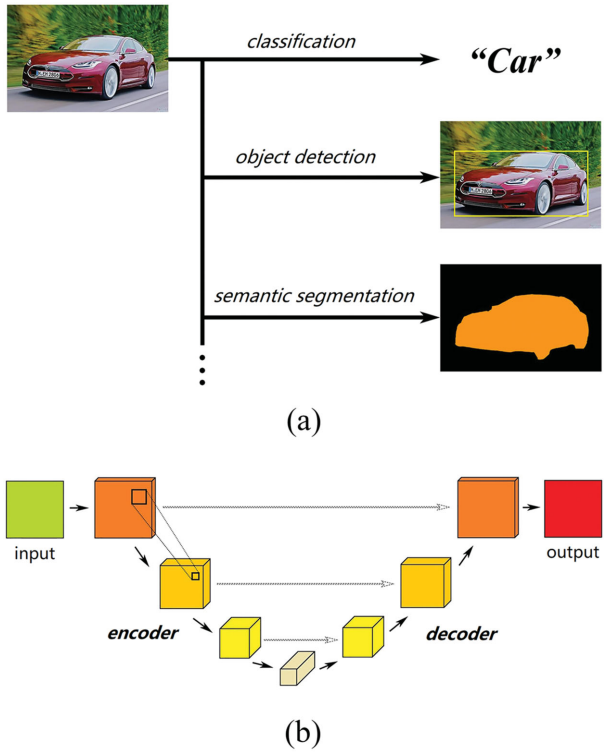

Figure 1. (a) A schematic illustrating the typical uses of the CNN. The figure shows examples of classification, object detection, and semantic segmentation of a car image. (b) A schematic of a typical encoder–decoder CNN architecture that consists of downsampling and upsampling parts with skip connections between them.

(BN) (Ioffe & Szegedy 2015), and pooling (POOL). The output of each layer, which constitutes the input of the next layer, is convolved with new filters. Repeating these operations can allow us to achieve a very deep network structure, which can be called a deep convolutional neural network (DCNN). Hyper-parameters such as $\boldsymbol{h}$ and $\boldsymbol{b}$ in the CNN are trained by supervised learning and back-propagation algorithms (Rumelhart *et al.* 1985). By using common optimization methods such as stochastic gradient descent (SGD), we can gradually reduce the error between the network outputs and labels, and finally, we can model the mapping of the training set to the tag set.

One typical application of the CNN is for classification tasks, where the output of each image is a one-hot vector indicating the likelihood that the image belongs to each class. However, in some tasks, such as cell segmentation in biomedicine, we need to classify not only an image but also each pixel in the image. This kind of task is called semantic segmentation (or image segmentation), as shown in figure 1a. Here, we treat seismic fault detection as a semantic segmentation task, i.e. we can classify each point in a seismic section into one of two classes, namely, faults and non-faults. Researchers investigating semantic segmentation proposed a method that trains a CNN to predict the class of each pixel by providing a patch around that pixel as input (Ciresan *et al.* 2012). However, this method has obvious drawbacks, such as redundancy. Consequently, researchers then proposed the fully convolutional network (FCN) that changes the last few fully connected layers in the CNN, which are generally used for classification tasks, into convolutional layers to implement dense pixel prediction (Long *et al.* 2015). Using this method can generate segmented images of any size, and this approach is much faster than previous methods. Subsequently, almost all advanced methods in semantic segmentation adopted the CNN model, and thus, this state-of-the-art technology has become increasingly popular. In recent years, more CNN models, such as U-Net (Ronneberger *et al.* 2015), SegNet (Badrinarayanan *et al.* 2015) and DeconvNet (Noh *et al.* 2015), all of which perform well on semantic segmentation tasks, have been proposed; most of them are encoder–decoder CNN architectures. A schematic of a typical encoder–decoder CNN architecture is shown in figure 1b.

2.2. *U-Net architecture*

For fault detection tasks in seismic interpretation, it is difficult to obtain a large amount of labeled data in most cases; thus, if we want to use real data to train the network, the training set will be small. With the development of CNNs, researchers have proposed some networks, such as DenseNet (Huang *et al.* 2017), which can work with a small amount of training data and achieve a very good performance. There are also networks, such as U-Net and MS-D CNN (Pelt & Sethian 2017), which have such features in semantic segmentation. To take the advantage of this kind of network, in our proposed method, we choose U-Net as the basic architecture of CNN model.

The U-Net architecture was originally designed for biomedical segmentation tasks such as cell segmentation. One of the U-Net architectures is shown in figure 2, which illustrates a typical encoder–decoder CNN model consisting of both a contracting path for downsampling the image and a symmetrical expansive path for upsampling. In the contracting path, the image size decreases while the number of channels becomes larger, and the image size reaches a minimum at the bottom. In contrast, in the expansive path, the image size becomes larger while the number of channels decreases, and finally, the output size is recovered to be the same as the original image.

Many studies have shown that U-Net performs well not only on biomedical segmentation but also on many kinds of semantic segmentation tasks, such as satellite image segmentation (Iglovikov *et al.* 2017). Therefore, we introduce U-Net into seismic fault detection. However, most segmentation tasks such as cell segmentation and satellite image segmentation are different from fault detection; the cell

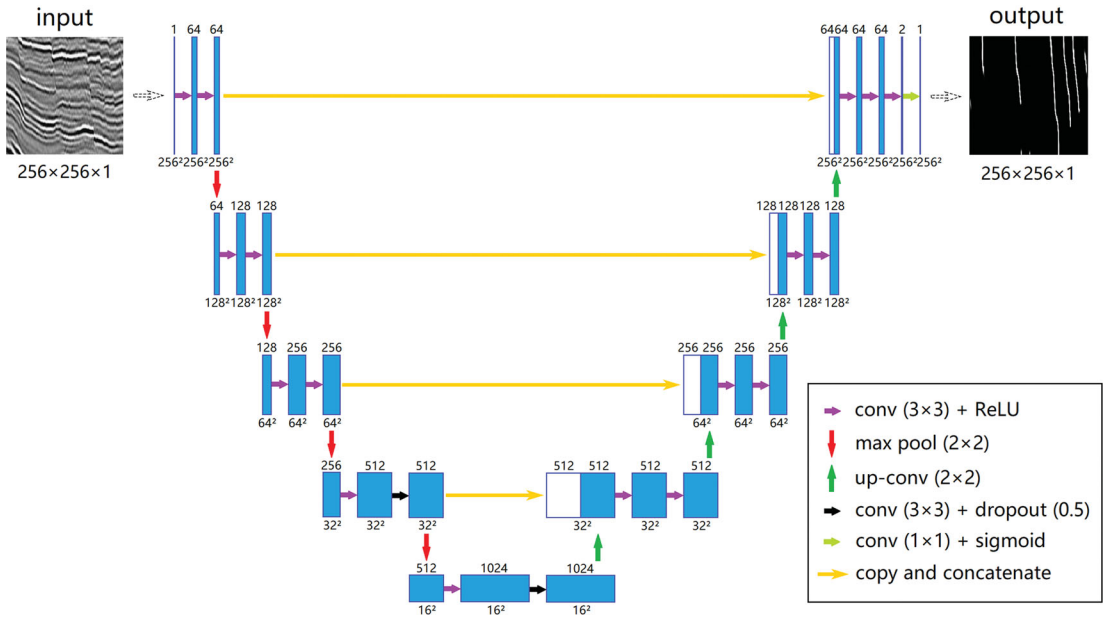

Figure 2. The encoder–decoder CNN model used in the proposed method based on the U-Net architecture consisting of four downsampling and four upsampling layers. The number of feature maps and the image dimensions of each layer are marked above and below each layer, respectively.

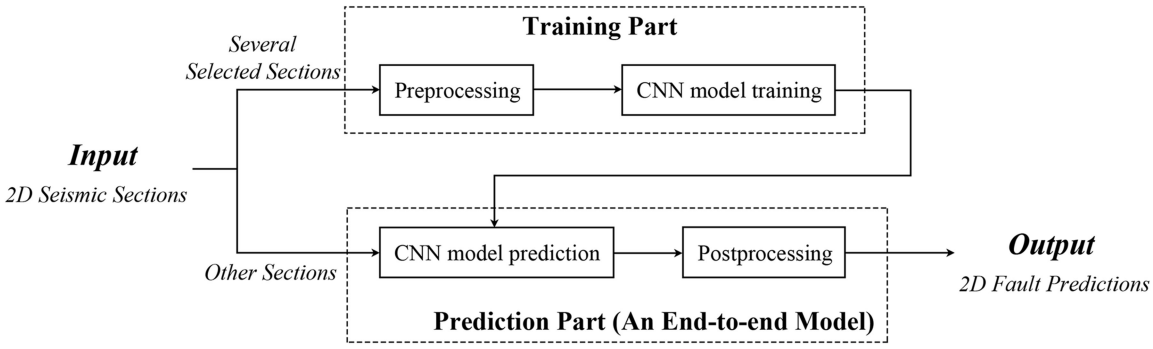

Figure 3. The workflow of the proposed method containing two main parts: training and prediction.

boundary in a biomedical image and the house boundary in a satellite image are closed curves that enclose an apparent area, whereas a fault in a seismic section is a line with a width of one or several pixels. Nevertheless, in essence, a 2D fault is still a narrow area enclosed by a closed curve. We demonstrate in the following experiments that U-Net can also exhibit a good performance in seismic fault detection.

3. Workflow and real data example

The workflow of the proposed method for seismic fault detection is shown in figure 3. Our proposed approach has two main part: training and prediction. The raw inputs of the workflow are 2D seismic sections obtained from a 3D volume, and the outputs are binary images of the same size, in which each pixel is classified into one of two classes: faults and non-faults. Once the CNN model is well trained, it can be reused to predict new images, thereby greatly improving the interpretation efficiency. After several parameters have been selected, the prediction part of the workflow constitutes an end-to-end model that can automatically predict faults.

3.1. *Training*

The training part has two steps: preprocessing and CNN model training. In this part, we choose N seismic sections with obvious faults from raw inputs as the training set of the CNN. Here, we choose not to set up a validation set. In the

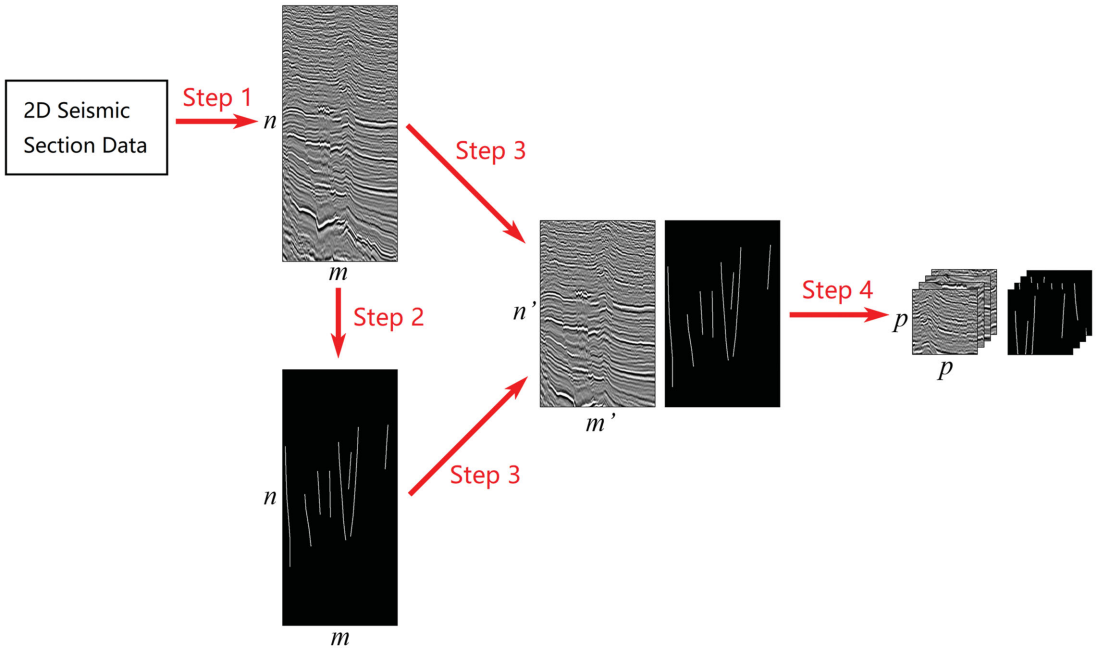

Figure 4. A schematic of the preprocessing scheme.

prediction part, the well-trained CNN model is used to predict faults in other sections.

3.1.1. Preprocessing. For data processing, especially in deep learning, preprocessing is crucial. We divide the preprocessing into four steps, as shown in figure 4.

Step 1: converting seismic sections into grayscale images. We convert the real-valued 2D matrices of N sections with dimensions of $m \times n$ into 2D grayscale images of the same size, i.e. the image size is $m \times n$, and each pixel value is an integer in the range of ∼0–255. Although doing so will lose some information, the result of this preprocessing step is more convenient for later processing.

Step 2: fault labeling. By referring to the grayscale images obtained above, we manually label the faults. Each label is a binary image that has a value of either one (faults) or zero (non-faults). When labeling faults, the pixel width of the fault label can be chosen in various ways, such as by trial-and-error. In the following experiments in this paper, we choose a width of 2 pixels.

Step 3: cropping the image to highlight faults. To highlight faults, we extract the region that contains the most faults from the N images and discard other regions, resulting in N images with dimensions of $m' \times n'$. For corresponding labels, we perform the same operations. Without this step, we may generate many samples that do not contain any positive examples in the next step, which will reduce the performance of the model.

Step 4: generating samples. Conventionally, the input images of a CNN are square. Therefore, we randomly generate N' $(N' > N)$ square samples with dimensions of $p \times p$ $[p \leq \min(m', n')]$ from N large images with dimensions of $m' \times n'$. For corresponding labels, we perform the same operations. It should be noted that the number of samples will affect the performance of the model. Too few samples may cause underfitting, while too many samples may cause overfitting.

When the number of training samples is limited, we can augment the data to increase the diversity of training samples and improve the robustness of the network. Traditional data augmentation methods include rotation, flipping, blurring, gamma transform and adding noise. Considering the symmetry of the seismic volume and the convenience of the calculation, we choose to only flip the image horizontally. Two choices can be made here. One is to flip a part of the images in the dataset to generate new images and preserve the old images, which makes the dataset larger; the other does not preserve the old images, and thus, the size of the dataset is not changed. Because we can obtain a sufficient number of images in the previous steps, we make the latter choice.

3.1.2. CNN model training. We use U-Net as the basic architecture of the CNN model in the proposed method, as shown in figure 2. An important reason for this architecture is that U-Net can show a good performance even if the training set is small. We use a U-Net architecture with four pooling layers, which means that it contains four downsampling operations and four upsampling operations. Each convolutional layer uses a 3 × 3 pixel filter. Each downsampling is

a 2×2 max-pooling operation, and the corresponding upsampling is a 2×2 up-convolution operation. Every step along the expansive path consists of a concatenation of feature maps after upsampling and the correspondingly cropped feature maps from the contracting path. The nonlinear activation function in the network is generally a ReLU, and the last layer uses a sigmoid function. In the corresponding two positions in figure 2, we use a dropout (0.5) operation, which means that half of the neurons in the previous layer do not work. Because the dimensions of the images in the experiments are 256×256, we use the same-padding convolution technique at each step in the network to prevent the pooling layer from encountering odd-sized feature maps. Unlike the valid-padding convolution technique used in the literature (Ronneberger *et al.* 2015), same-padding convolution ensures that the output feature map has the same dimensions as the input feature map. When training the network, we choose binary cross entropy, which is common for binary classification tasks, as the loss function. This loss function is defined as follows:

$$H(\hat{y}, y) = -\frac{1}{n}\sum_{i=1}^{n}\left(y_i \log \hat{y}_i + (1 - y_i)\log\left(1 - \hat{y}_i\right)\right), \tag{2}$$

where y is the ground truth and $\hat{y}$ is the prediction.

A U-Net architecture with different numbers of layers (e.g. three pooling layers) can be considered. Compared with the U-Net architecture with four pooling layers mentioned above, a U-Net architecture with three pooling layers lacks the bottom-most downsampling layer and the symmetric upsampling layer, meanwhile it has fewer hyper-parameters to be trained and thus has a shorter training time. Hence, this architecture with three pooling layers can be considered when the U-Net with four pooling layers is overfitted.

3.2. *Prediction*

The prediction part also has two steps: CNN model prediction and postprocessing.

3.2.1. CNN model prediction. When making predictions on the test set, images with the same dimensions as the training samples ($p \times p$) can be directly inputted to the network to obtain results. However, if we want to predict an image with any dimensions, such as $m' \times n'$, the image cannot be directly inputted to the network due to the size inconsistency. Instead, we use a sliding window with dimensions of $p \times p$ to predict different parts of the image with a certain stride. For computational convenience, we perform a p-size padding operation around the image with dimensions of $m' \times n'$ to obtain an image with dimensions of $(m' + 2p) \times (n' + 2p)$. The benefit of this approach is that when the sliding window with dimensions of $p \times p$ slides over the image with dimensions of $(m' + 2p) \times (n' + 2p)$, each pixel in the original image is framed by the sliding window and is predicted for the same number of times. Therefore, we can simply divide the summation of the overlapping predictions by the number of overlapping windows at each pixel to obtain a final normalized output probability map. For tasks such as cell segmentation, it is common to use mirroring as a padding method (Ronneberger *et al.* 2015). However, due to the nature of seismic data, mirroring is not appropriate. Instead, we use a simple method, namely, zero padding, which sets the pixel values of padding positions to 0, as shown in figure 5. Correspondingly, we need to crop the output of the CNN with dimensions of $(m' + 2p) \times (n' + 2p)$ to the original dimensions of $m' \times n'$, and remove the outliers at the edge of the image that exist as a consequence of the zero-padding operation. Through these operations, our model has the advantage of being able to predict sections of any size.

It should be noted that the distributions of data from the same seismic volume are similar, while the distributions of data from two different seismic volumes are usually dissimilar, as shown in figure 6. As a result, the proposed method will perform well when the test data and training data come from the same seismic volume, but the performance cannot be guaranteed when they are from different volumes. We will consider this in the discussion.

3.2.2. Postprocessing. The output of the CNN model is a probability map with values ranging from 0 to 1, reflecting the probability that the corresponding pixel in the input image belongs to a fault. The larger the pixel value is, the more likely the pixel in the input image belongs to a fault. Because the meaning of the probability map is sufficiently intuitive, we can directly provide it to assist interpreters in seismic interpretation, as shown in figure 7a. Another common approach is to convert the probability map into a binary image by selecting a threshold T, i.e. to map the values in the output image from $[0, 1]$ to $\{0, 1\}$. A value of 0 (black) represents the negative class (non-fault), and 1 (white) represents the positive class (fault). This approach indicates that a pixel with a probability value greater than a certain value is classified as a fault. However, it should be noted that fault lines in the resulting binary image may be discontinuous, as shown in figure 7b. This is understandable from the perspective of computer vision because the CNN model may not be able to classify the pixels correctly where faults are small or unclear. Therefore, we need postprocessing to connect fault lines that should belong to the same fault to form a complete fault.

To quantify this problem, we consider a fault line segment in a binary image as a curve in planar space. Because a fault in a seismic section has a small curvature, we approximate the curve as some straight lines. Various traditional line detection

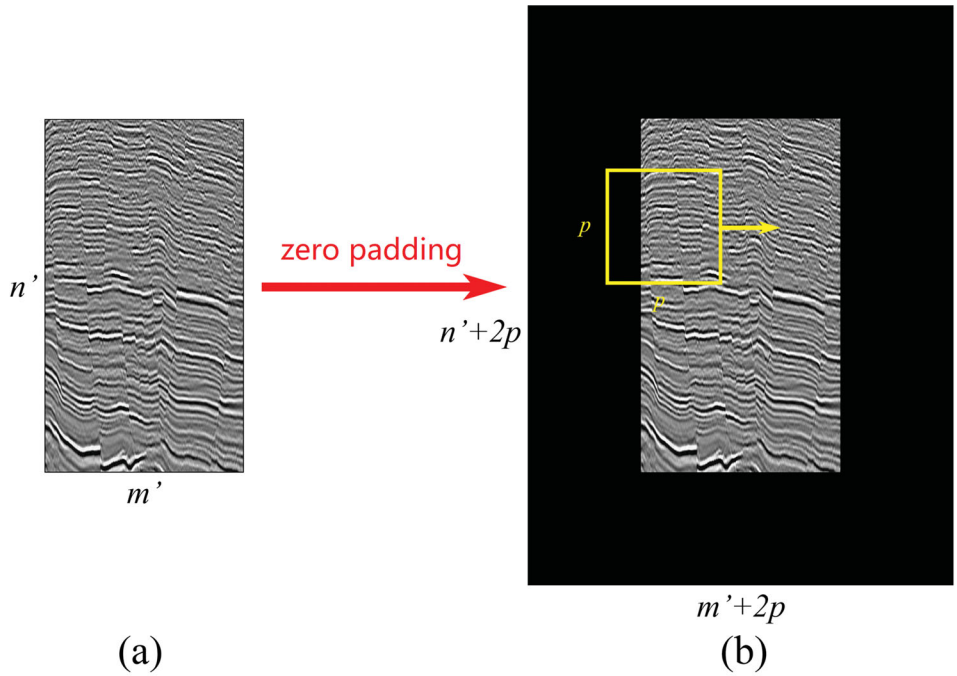

Figure 5. A schematic of the zero padding operation on an image: (a) the original image with dimensions of $m' \times n'$; (b) the padded image with dimensions of $(m' + 2p) \times (n' + 2p)$, where the yellow box is a sliding window with dimensions of $p \times p$ used to predict different parts of the image and finally superimpose them.

methods, including the Hough transform, can be used. The line segment detector (LSD) technique is another newly proposed method of line detection (Von Gioi *et al.* 2012) that performs better than the Hough transform in many cases. After performing some experiments, we choose LSD to detect lines in the image. Because many faults in seismic sections are nearly vertical, using the slope k to describe a line may lead to problems extending to infinity. Therefore, we use the angle θ to describe the line.

After detecting the lines in the image, we traverse them in the space from top to bottom. For the current processing line L_i, we remove the line if it is regarded as a small isolated outlier in the space by setting a threshold T_A. Then, fixing L_i, we traverse the remaining unprocessed lines in the space from top to bottom. For L_i and the current unprocessed line L_j, as figure 8 illustrates, we define the upper and lower endpoints of L_i as P_i and Q_i, respectively, and the upper and lower endpoints of L_j as P_j and Q_j, respectively. If Q_i and P_j do not coincide (i.e. if L_i and L_j are not connected) and θ_i (the angle between $\overline{Q_iP_j}$ and L_i) and θ_j (the angle between $\overline{Q_iP_j}$ and L_j) are both smaller than a threshold T_θ, and the length of $\overline{Q_iP_j}$ is less than a threshold T_B, then we connect the corresponding two curves in the binary image with a straight line to form a whole curve. The pixel width of the straight line is consistent with the fault line in the label. We use the pseudocode in Algorithm 1 to describe the postprocessing details: function $l(\cdot)$ calculates the lengths of the lines, and T_A, T_B, and T_θ are three pre-selected thresholds. Finally, the result comprising a better fault continuity is taken as the output of the entire model. A schematic of the postprocessing scheme is shown in figure 7.

3.3. *Fault detection accuracy index*

To evaluate the performance of a model, it is necessary to introduce some evaluation metrics. The FauSIM index metric was proposed based on the Fréchet distance to describe the similarity between detected faults and manually picked faults (Wang & AlRegib 2017). However, the definition of the FauSIM index is too complicated, and it requires the prior selection of several parameters. In this paper, we introduce a simple evaluation metric based on the Intersection over Union (IoU) to the fault detection task. IoU, also known as the Jaccard index, is a common evaluation metric utilized to measure the accuracy of model results (Jaccard 1901). In many object detection challenges, such as the PASCAL VOC challenge (Everingham *et al.* 2015), IoU is used as the metric, and it is defined as follows:

$$IoU\ (A, B) = \frac{|A \cap B|}{|A \cup B|}, \tag{3}$$

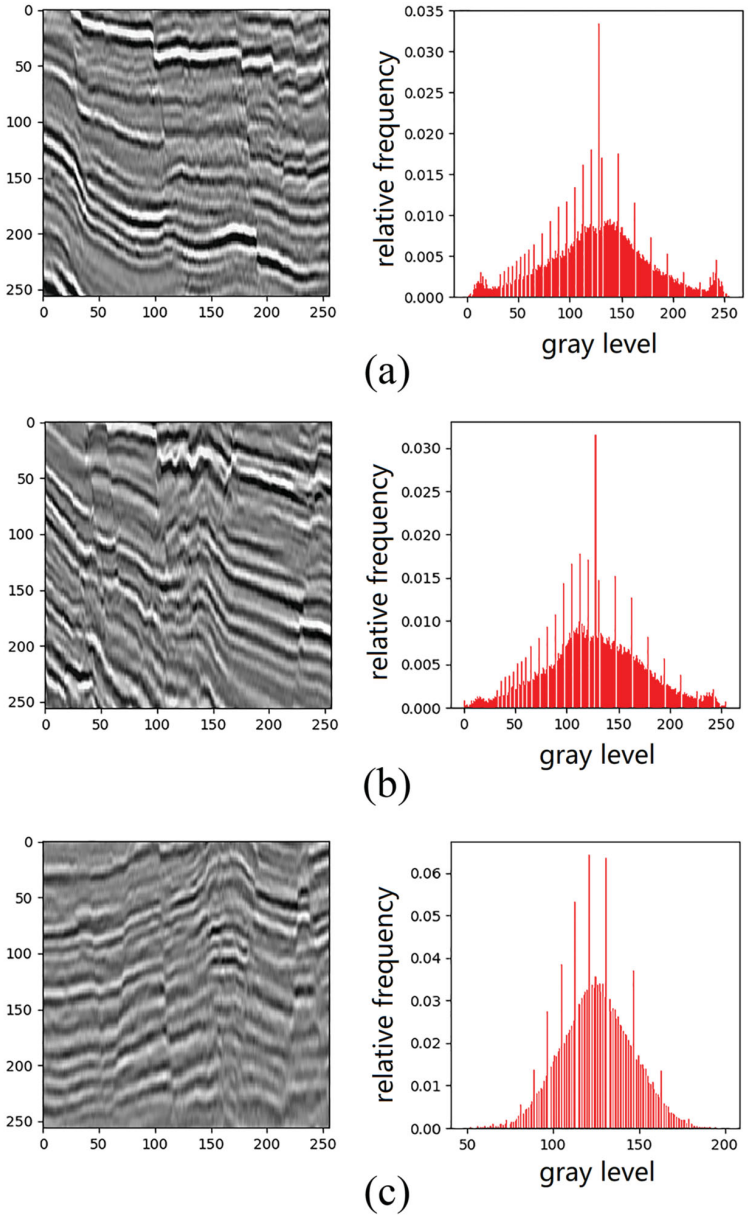

Figure 6. Three seismic sections and their normalized gray histograms: the data in (a) and (b) are from the same seismic volume, while the data in (c) are from another volume. The data distributions of the same seismic volume are similar, while the data distributions of the different seismic volumes are dissimilar. The section size is 256 × 256. The horizontal distance sampling interval is 25 meters while the vertical time sampling interval is 0.002 seconds.

where A is the area of ground truth and B is the area of prediction. The normalization condition is described as follows:

$$0 \leq IoU\ (A, B) \leq 1. \tag{4}$$

In simple terms, IoU reflects the correlation between the ground truth and prediction. A greater IoU value represents a higher correlation and vice versa, as shown in figure 9b.

For a complete seismic section, it is more likely that there are many faults rather than only one. For some subjective reasons, when manually labeling faults, interpreters generally do not label each of them, and some relatively small faults are excluded. However, the CNN may detect and predict small faults. This situation can be seen in the experiments presented later herein, thereby reflecting the effectiveness of the CNN model. Therefore, for fault detection, using IoU as an evaluation metric is not appropriate. Based on IoU, we introduce a fault detection accuracy (FDA) index that describes the accuracy of the fault detection results, and it is defined as follows:

$$FDA\ (A, B) = \frac{|A \cap B|}{|A|}, \tag{5}$$

where A is the area of ground truth and B is the area of the prediction. The normalization condition is defined as

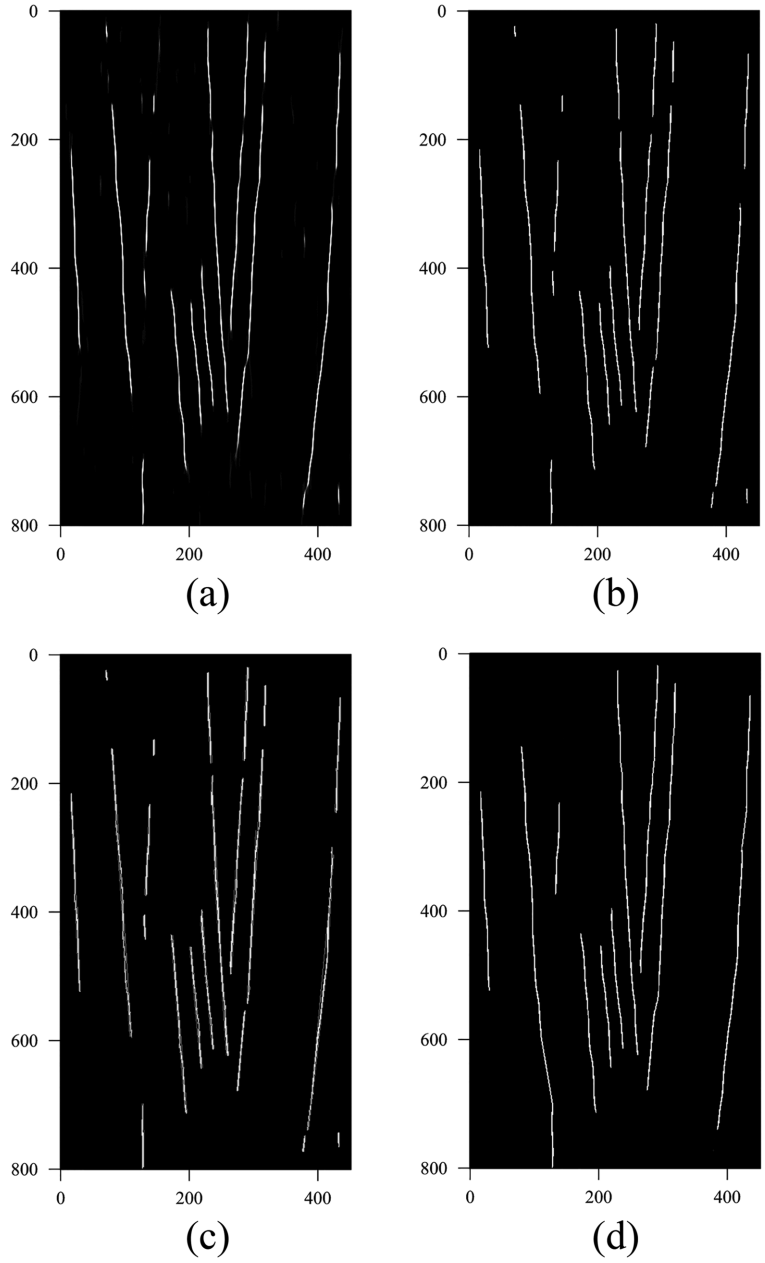

Figure 7. An example of postprocessing: (a) the probability map of the CNN prediction; (b) the binary image by the threshold $T = 0.5$, which indicates that a pixel with a probability value greater than 0.5 is classified as the positive class (fault); (c) the line detection result by LSD; and (d) the final result after postprocessing. The section size is 451×800. The horizontal distance sampling interval is 25 meters while the vertical time sampling interval is 0.002 seconds.

follows:

$$0 \leq FDA\,(A, B) \leq 1. \tag{6}$$

Compared with IoU, the denominator of the FDA index is the area of ground truth rather than the area of the union, as shown in figure 9c. When we evaluate the global fault detection performance of the model, we are mainly concerned with whether the model can accurately predict faults at manually labeled positions. Therefore, it is appropriate to consider only manually labeled faults in the denominator of the FDA index, and thus, the FDA index is particularly suitable for fault detection tasks using neural networks. In addition, the FDA index has an advantage when considering complex situations, e.g. when a seismic section contains many faults, for which calculating the FDA index is much easier than calculating the FauSIM index. It should be noted that evaluation metrics, including the FDA index and the FauSIM index, are affected by the subjective factors associated with manually labeling faults. Consequently, the performance of the model still needs to be judged by experienced interpreters and geologists.

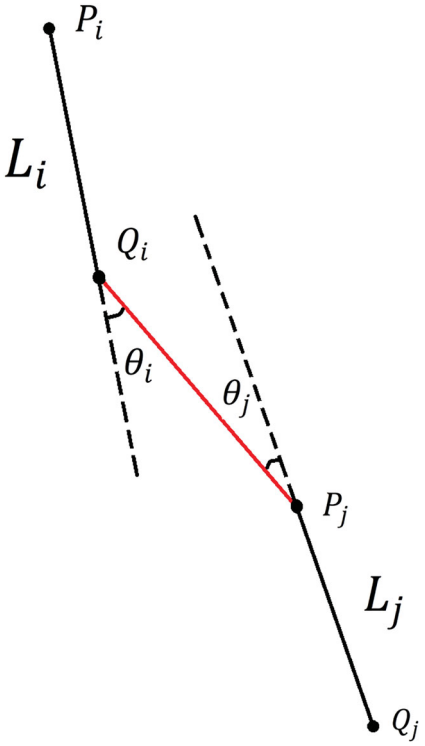

Figure 8. The illustration of two detected lines, namely, L_i and L_j. We connect the corresponding two fault curves if they belong to the same fault.

3.4. *Real data example*

We use Tensorflow (Abadi *et al.* 2016) as the deep learning framework. All models are trained on an NVIDIA GTX1070 GPU (8 GB).

The data we used are land data collected from a certain seismic volume in Bohai Bay, China. To highlight the fact that the proposed method requires only a very small amount of training data, we used only seven seismic sections as the training set of the network. In addition, when labeling faults, due to the complexity of faults in real data, we labeled only obvious and definite faults while ignoring those that are ambiguous. It is worth noting that the ability of the proposed method to extract fault features effectively from seismic sections will be validated if the predictions of the neural network can detect small faults.

In one of the experiments, we randomly generated 140 images with dimensions of 256×256 from seven raw input seismic sections with dimensions of 451×1001 during the preprocessing and generated 140 corresponding labels with dimensions of 256×256. The CNN model we used was a U-Net with four pooling layers, as shown in figure 2. We used an Adam Optimizer (Kingma & Ba 2014) and trained the network for 100 epochs with a learning rate of 1e-4. Each epoch was trained on batches, and each batch contained 10 images. The training time of the CNN model was approximately 30 minutes. Because the network could offer many predictions after being trained only once, the training time was very reasonable.

We then applied our network to the test set. When predicting an image with dimensions of 256×256, the prediction time was less than 1 second. The result was a probability map, as shown in figure 10b. Figure 11 shows its enlarged pixel-level details. During the postprocessing, to observe a typical situation, we classified the pixels with probability values greater than 0.5 as faults (i.e. with a threshold $T = 0.5$). The result after postprocessing is shown in figure 10c. To evaluate the performance of the model, we calculated the FDA index and IoU of the result. The FDA index reflecting the result accuracy at the manually labeled positions was 0.655, and the IoU value was 0.500. The IoU value was lower than the FDA index value because the CNN may detect small faults

Algorithm 1. The pseudocode describing the postprocessing details, where Function $1(\cdot)$ calculates the lengths of lines, and T_A, T_B, and T_θ are three pre-selected thresholds.

Algorithm 1: Postprocessing.

```
for i ← 1 to N_L do
    if l(L_i) < T_A then
        Remove (L_i)
    else
        for j ← 1 to N_r do
            if Q_i ≠ P_j and θ_i < T_θ and θ_j < T_θ and l(Q_iP_j) < T_B then
                Connect (Q_i, P_j)
                exit
            end if
        end for
    end if
end for
```

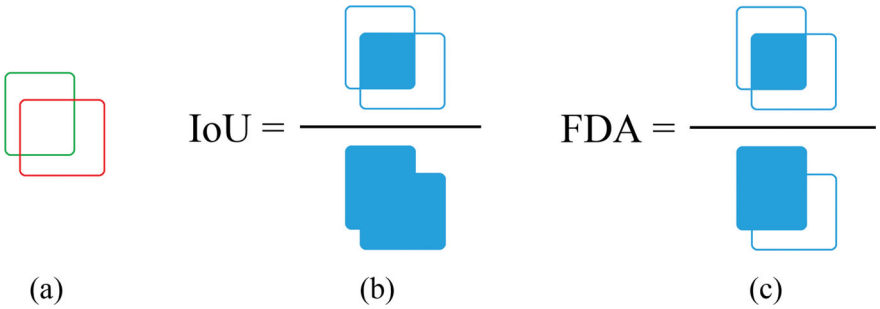

Figure 9. A schematic of IoU and the FDA index: (a) ground truth and prediction in the detection task, where the green box and the red box represent the ground truth bounding box and predicted bounding box, respectively; (b) IoU, defined as the area of overlap divided by the area of the union; and (c) the FDA index, defined as the area of overlap divided by the area of ground truth.

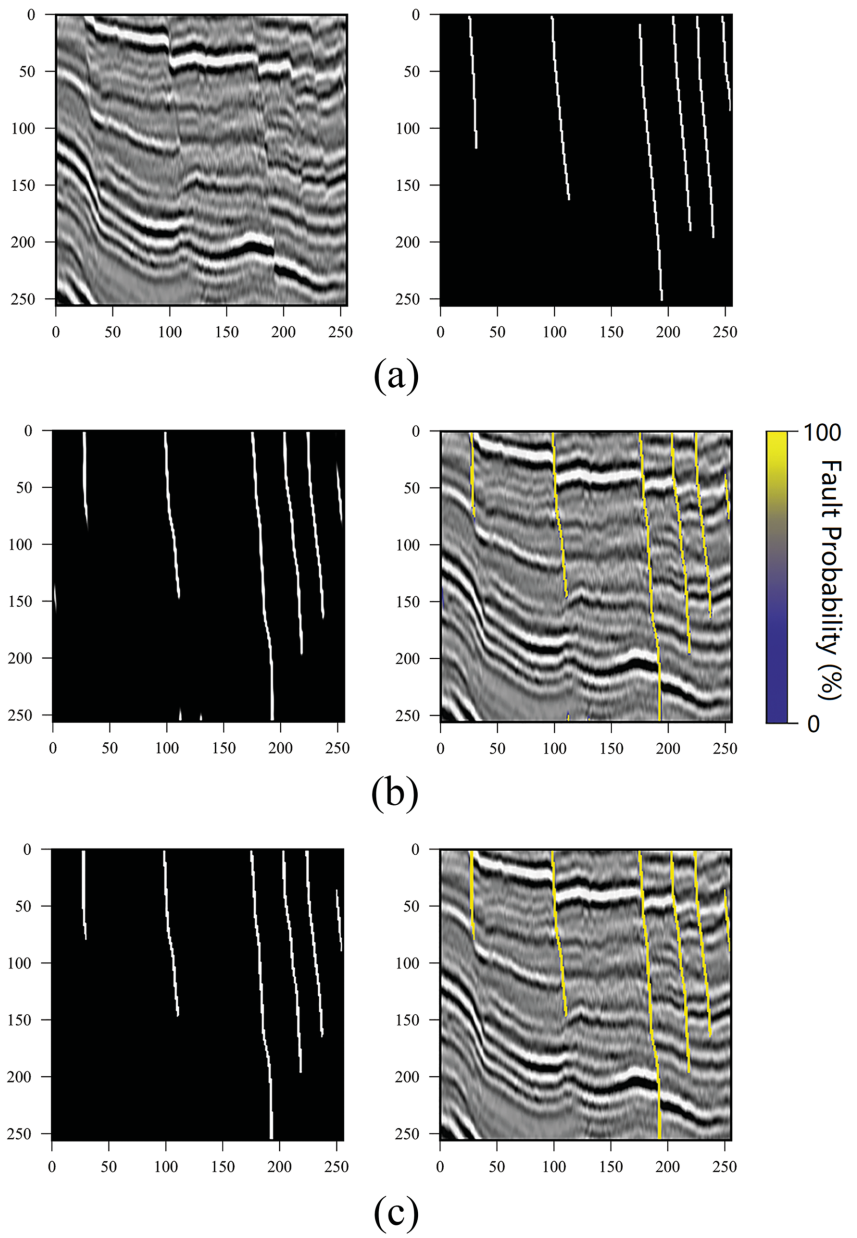

Figure 10. The prediction of a 256 × 256 image, which is a cross-line (xline) and time profile. The horizontal distance sampling interval is 25 meters while the vertical time sampling interval is 0.002 seconds. (a) the original image and its corresponding manual label; (b) the probability map of the CNN prediction; and (c) the result after postprocessing.

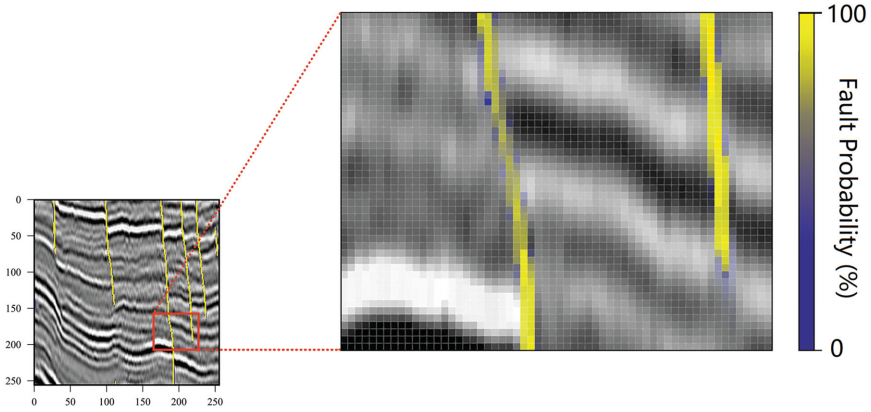

Figure 11. An enlarged pixel-level prediction probability map of a seismic section. Yellow areas show predictions with a high fault probability, whereas blue areas shows predictions with a low fault probability. The section size is 256 × 256. The horizontal distance sampling interval is 25 meters while the vertical time sampling interval is 0.002 seconds.

that have not been manually labeled. In addition, we show the results of the several representative training sets used in this study in Table 1. The author of a previous study on fault detection using deep learning (Araya-Polo *et al.* 2017) used DNNs to predict faults; the best result was a maximum IoU value of 0.395 when predicting a small synthetic geological model with one fault. We can therefore see the advantages of the proposed method.

When predicting a larger image with dimensions of 451 × 800, we set the stride of the sliding window to 16, and the prediction time was approximately 90 seconds. Because the decision made by superimposing more prediction results was theoretically better, the prediction result with a stride of 1 should be more accurate. However, the experiments showed that the results when the stride was 1, 4 and 16 were not significantly different. In addition, the prediction took several hours, half an hour and 90 seconds when the stride was 1, 4 and 16, respectively. Therefore, we chose a stride of 16. During the postprocessing, we also chose the threshold T to be 0.5. We show the result of the above model in figure 12, which proves that the proposed method can predict images of any size.

Table 1. Results obtained from several representative datasets. The first four columns describe the parameters of the experiments; the last two columns report the performance metrics, namely, the FDA index and IoU. All results are obtained on randomly generated training sets, except for the first row case, where the training set is only 19 manually selected images.

Training dataset					
Size	Method	Data augmentation	Pooling layers	FDA	IoU
19	Manually selected	No	4	0.564	0.404
70	Randomly generated	Yes	4	0.578	0.433
140	Randomly generated	Yes	4	0.655	0.500
140	Randomly generated	Yes	3	0.556	0.412
210	Randomly generated	Yes	4	0.571	0.427
700	Randomly generated	Yes	4	0.471	0.356

4. Discussion

Since we use a small amount of real data as the training set, the performance of the model is related to the selected samples. In fact, training a deeper network while using more data may improve the performance of the model to some extent. In addition, better results on various metrics can be obtained when using a pre-trained encoder (Iglovikov & Shvets 2018; Zhou *et al.* 2018). However, obtaining the most accurate result is not the intention of this paper. Instead, we mainly focus on the possibility of using a small training set to train neural networks for seismic fault detection.

Since the proposed method requires only a small training set, it has two obvious advantages, one of which is its high cost effectiveness. By simply labeling several seismic sections, we can obtain more accurate predictions than can some other methods with big data, which makes the cost-effectiveness of the proposed method very high. On the other hand, for common researchers and interpreters, it may be difficult to obtain a large amount of labeled real data to train good neural networks; as a result, synthetic data are used as training sets. However, the models trained by only synthetic data may obtain a poor performance when predicting land data with much lower signal-to-noise ratios, whereas the use of real data to train networks should result in more accurate predictions. Therefore, another advantage of the proposed method is that we can use real data as the training set without any concerns about insufficient data.

The use of a small training set could lead to network overfitting, and the generalization performance of the model could be sacrificed; therefore, in our proposed method, a network trained by a certain seismic volume may not make good predictions on other volumes. However, this does not affect

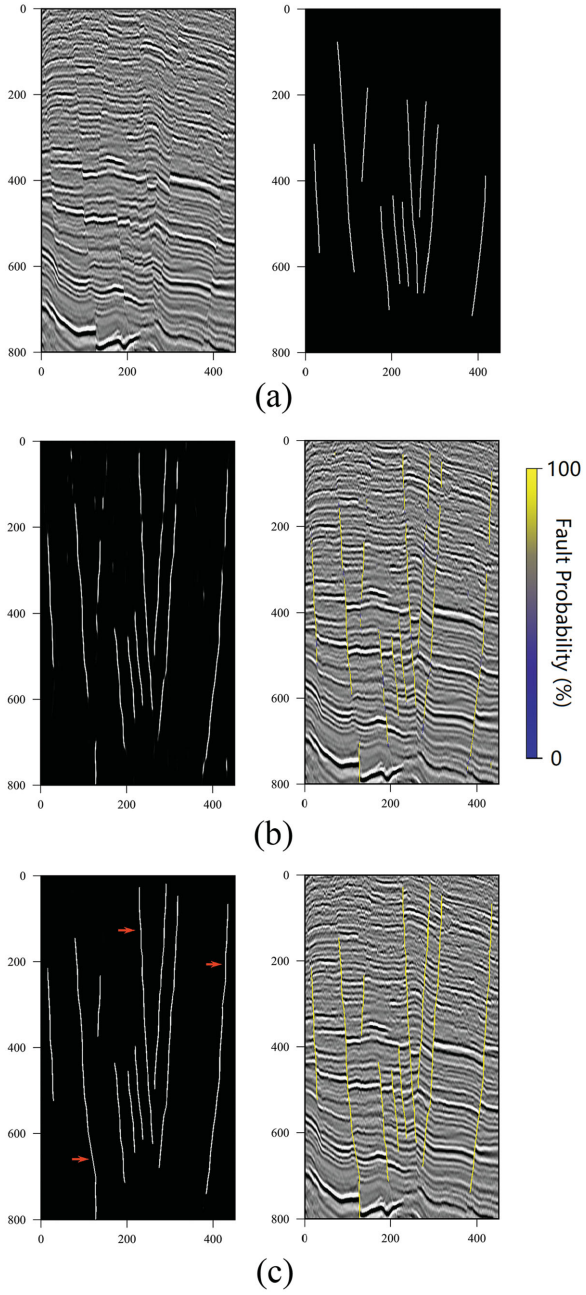

Figure 12. The prediction of a 451 × 800 image, which is a cross-line (xline) and time profile. The horizontal distance sampling interval is 25 meters while the vertical time sampling interval is 0.002 seconds. (a) the original image and its corresponding manual label; (b) the probability map of the CNN prediction; and (c) the result after postprocessing, where the red arrows show faults detected by the model but not annotated with a label.

the practical application of the proposed method. When interpreting a new seismic volume, interpreters need only to pick and label several 2D sections of the volume, which is very easy to accomplish. The strategy of transfer learning (Pan & Yang 2010) is often considered to enhance the generalization performance of a network; however, this approach requires interpreters to label some samples from the new volume, and thus, the costs of the transfer learning strategy and the proposed method to retrain a network for a new volume are similar. In addition, neural network overfitting problems are always being discussed (Recht *et al.* 2018). Accordingly, we should pay more attention to the performance and practicality of the proposed method.

Since fault systems are distributed in three dimensions, it is more natural to use 3D seismic cubes as inputs to the network than to use 2D seismic sections. Moreover, using 3D data should result in more accurate predictions. However, 3D convolution requires greater amounts of computational resources; consequently, in recent studies, researchers have used samples with three orthogonal slices of seismic images rather than 3D cubes when dealing with 3D data (Xiong *et al.* 2018). In addition, when using 3D data, interpreters may need to label a large number of 2D sections, which is contrary to the original intention of this paper to use a small training set. Thus, the proposed method addresses 2D data. Using a small 3D training set to train a CNN to predict seismic volumes directly constitutes our future work, which demonstrates promising results using 3D U-Net (Çiçek *et al.* 2016).

Some technical details can be improved in future work. The data augmentation used herein is flipping images horizontally. We can consider other advanced methods, such as generating additional labeled data through generative adversarial networks (GANs) (Goodfellow *et al.* 2014). Moreover, we currently use only U-Net as the CNN architecture in our model. In the future, we can try to use other popular architectures, such as SegNet, for semantic segmentation; then, we can potentially assemble them to improve the overall performance of the model. When training the CNN model, we can also try to use the parameters of pre-trained models that perform well on other datasets as the initial parameters. Preliminary results along these lines are promising.

5. Conclusion

Unlike traditional deep learning methods that use very large datasets to train neural networks, in this paper we propose a seismic fault detection method based on encoder–decoder CNN that needs only a small training set. Experimental results on real data show that the proposed model can perform better than some models trained by big data.

The proposed method is practical, especially for researchers and interpreters who have difficulty obtaining a large amount of labeled real data. Furthermore, the method has two obvious advantages. One is its high cost effectiveness, that is, several labeled sections can train a network that can make accurate predictions; another advantage is that it can allow us to use real data as the training set without any concerns about insufficient data. In addition, the proposed method can predict 2D seismic sections of any size, and

after several parameters have been selected, the entire prediction is automated. When interpreting a new seismic volume, interpreters need only to pick and label several sections, after which the model can predict faults in any other section of the volume. Due to the expensive labor and time costs required to label faults manually, the proposed method can greatly improve the interpretation efficiency. The advantages of the proposed method we have shown in this work demonstrate the potential of using small amounts of data for seismic interpretation.

Acknowledgements

The authors are grateful to the Strategic Priority Research Program of Chinese Academy of Sciences (under Grant No. XDA14040100). We also thank the National Natural Science Foundation of China (under Grant 41804129) and China Postdoctoral Science Foundation (under Grant 2018T110137) for supporting this work.

Conflict of interest statement. None declared

References

Abadi, M. *et al.*, 2016. TensorFlow: a system for large-scale machine learning, in *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, USENIX Association, Savannah, GA, pp. 265–283.

Aqrawi, A.A. & Boe, T.H., 2011. Improved fault segmentation using a dip guided and modified 3D Sobel filter, in *SEG Technical Program Expanded Abstracts*, pp. 999–1003, Society of Exploration Geophysicists.

Araya-Polo, M., Dahlke, T., Frogner, C., Zhang, C., Poggio, T. & Hohl, D., 2017. Automated fault detection without seismic processing, *The Leading Edge*, **36**, 208–214.

Badrinarayanan, V., Kendall, A. & Cipolla, R., 2015. Segnet: a deep convolutional encoder–decoder architecture for image segmentation, *arXiv:1511.00561*.

Bishop, C.M., 2006. *Pattern Recognition and Machine Learning*, Springer.

Boe, T.H. & Daber, R., 2010. Seismic features and the human eye: RGB blending of azimuthal curvatures for enhancement of fault and fracture interpretation, in *SEG Technical Program Expanded Abstracts,* pp. 1535–1539, Society of Exploration Geophysicists.

Çiçek, Ö., Abdulkadir, A., Lienkamp, S.S., Brox, T. & Ronneberger, O., 2016. 3D U-Net: learning dense volumetric segmentation from sparse annotation, in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2016*, pp. 424–432, Ourselin S., Joskowicz L., Sabuncu M.R., Unal G. & Wells W., (eds.), Springer, Cham.

Ciresan, D., Giusti, A., Gambardella, L.M. & Schmidhuber, J., 2012. Deep neural networks segment neuronal membranes in electron microscopy images, in *Advances in Neural Information Processing Systems (NIPS)*, **25**, pp. 2843–2851.

Cohen, I., Coult, N. & Vassiliou, A.A., 2006. Detection and extraction of fault surfaces in 3D seismic data, *Geophysics*, **71**, P21–P27.

Dahlke, T., Araya-Polo, M., Zhang, C. & Frogner, C., 2016. Predicting geological features in 3D seismic data, in *Advances in Neural Information Processing Systems (NIPS)*, **29**, 3D Deep Learning Workshop, Curran Associates, Inc.

Everingham, M., Eslami, S.A., Van Gool, L., Williams, C.K., Winn, J. & Zisserman, A., 2015. The pascal visual object classes challenge: a retrospective, *International Journal of Computer Vision*, **111**, 98–136.

Frogner, C., Zhang, C., Mobahi, H., Araya-Polo, M. & Poggio, T.A., 2015. Learning with a Wasserstein loss, in *Proceedings of the 28th International Conference on Neural Information Processing Systems*, MIT Press, Montreal, Canada, pp. 2053–2061.

Goodfellow, I. *et al.*, 2014. Generative adversarial nets, in *Advances in Neural Information Processing Systems*, pp. 2672–2680.

Guitton, A., Wang, H. & Trainor-Guitton, W., 2017. Statistical imaging of faults in 3D seismic volumes using a machine learning approach, in *SEG Technical Program Expanded Abstracts*, pp. 2045–2049, Society of Exploration Geophysicists.

Hale, D., 2013. Methods to compute fault images, extract fault surfaces, and estimate fault throws from 3D seismic images, *Geophysics*, **78**, O33–43.

Hami-Eddine, K. & de Ribet, B., 2017. A simple statistical approach to guide fault interpretation, in *SEG Technical Program Expanded Abstracts*, pp. 2336–2339, Society of Exploration Geophysicists.

Hornik, K., Stinchcombe, M. & White, H., 1989. Multilayer feedforward networks are universal approximators, *Neural Networks*, **2**, 359–366.

Huang, G., Liu, Z., Van Der Maaten, L. & Weinberger, K.Q., 2017. Densely connected convolutional networks, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4700–4708.

Iglovikov, V., Mushinskiy, S. & Osin, V., 2017. Satellite imagery feature detection using deep convolutional neural network: a kaggle competition, *arXiv:1706.06169*.

Iglovikov, V. & Shvets, A., 2018. TernausNet: U-net with VGG11 encoder pre-trained on imagenet for image segmentation, *arXiv:1801.05746*.

Ioffe, S. & Szegedy, C., 2015. Batch normalization: accelerating deep network training by reducing internal covariate shift, *arXiv:1502.03167*.

Jaccard, P., 1901. Étude comparative de la distribution florale dans une portion des Alpes et des Jura, *Bulletin de la Societ́e ́Vaudoise des Sciences Naturelles*, **37**, 547–579.

Kingma, D.P. & Ba, J., 2014. Adam: a method for stochastic optimization, *arXiv:1412.6980*.

Krizhevsky, A., Sutskever, I. & Hinton, G.E., 2012. ImageNet classification with deep convolutional neural networks, in *Proceedings of the 25th International Conference on Neural Information Processing Systems*, Curran Associates Inc., Lake Tahoe, Nevada, pp. 1097–1105.

LeCun, Y., Bengio, Y. & Hinton, G., 2015. Deep learning, *Nature*, **521**, 436–444.

LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P., 1998. Gradient-based learning applied to document recognition, *Proceedings of the IEEE*, **86**, 2278–2324.

Lewis, W. & Vigh, D., 2017. Deep learning prior models from seismic images for full-waveform inversion, in *SEG Technical Program Expanded Abstracts*, pp. 1512–1517, Society of Exploration Geophysicists.

Long, J., Shelhamer, E. & Darrell, T., 2015. Fully convolutional networks for semantic segmentation, in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Boston, MA, pp. 3431–3440.

Marfurt, K.J., Sudhaker, V., Gersztenkorn, A., Crawford, K.D. & Nissen, S.E., 1999. Coherency calculations in the presence of structural dip, *Geophysics*, **64**, 104–111.

Nair, V. & Hinton, G.E., 2010. Rectified linear units improve restricted Boltzmann machines, in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, Omnipress, Haifa, Israel, pp. 807–814.

Noh, H., Hong, S. & Han, B., 2015. Learning deconvolution network for semantic segmentation, in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, IEEE Computer Society, Washington, DC, pp. 1520–1528.

Pan, S.J. & Yang, Q., 2010. A survey on transfer learning, *IEEE Transactions on Knowledge and Data Engineering*, **22**, 1345–1359.

Pelt, D.M. & Sethian, J.A., 2017. A mixed-scale dense convolutional neural network for image analysis, *Proceedings of the National Academy of Sciences*, **115**, 254–259.

Recht, B., Roelofs, R., Schmidt, L. & Shankar, V., 2018. Do CIFAR-10 classifiers generalize to CIFAR-10?, *arXiv:1806.00451*.

Ronneberger, O., Fischer, P. & Brox, T., 2015. U-Net: convolutional networks for biomedical image segmentation, in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pp. 234–241, Nassir N., Joachim H., Wells W.M. & Frangi A.F., (eds.), Springer, Cham.

Rumelhart, D.E., Hinton, G.E. & Williams, R.J., 1985. Learning internal representations by error propagation, in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, pp. 318–362, MIT Press.

Schmidhuber, J., 2015. Deep learning in neural networks: an overview, *Neural Networks*, **61**, 85–117.

Serfaty, Y., Itan, L., Chase, D. & Koren, Z., 2017. Wavefield separation via principle component analysis and deep learning in the local angle domain, in *SEG Technical Program Expanded Abstracts*, pp. 991–995, Society of Exploration Geophysicists.

Silva, C.C., Marcolino, C.S. & Lima, F.D., 2005. Automatic fault extraction using ant tracking algorithm in the Marlim South Field, Campos Basin, in *SEG Technical Program Expanded Abstracts*, pp. 857–860, Society of Exploration Geophysicists.

Van Bemmel, P.P. & Pepper, R.E., 2000. Seismic signal processing method and apparatus for generating a cube of variance values, *U.S. Patent No. 6,151,555*, U.S. Patent and Trademark Office, Washington, DC.

Von Gioi, R.G., Jakubowicz, J., Morel, J.M. & Randall, G., 2012. LSD: a line segment detector, *Image Processing on Line*, **2**, 35–55.

Wang, Z. & AlRegib, G., 2017. Interactive fault extraction in 3-D seismic data using the hough transform and tracking vectors, *IEEE Transactions on Computational Imaging*, **3**, 99–109.

Wu, X. & Fomel, S., 2018. Automatic fault interpretation with optimal surface voting, *Geophysics*, **83**, O67–82.

Wu, X. & Hale, D., 2016a. 3D seismic image processing for faults, *Geophysics*, **81**, IM1–11.

Wu, X. & Hale, D., 2016b. Automatically interpreting all faults, unconformities, and horizons from 3D seismic images, *Interpretation*, **4**, T227–237.

Wu, X., Shi, Y., Fomel, S. & Liang, L., 2018. Convolutional neural networks for fault interpretation in seismic images, in *SEG Technical Program Expanded Abstracts*, pp. 1946–1950, Society of Exploration Geophysicists.

Wu, X. & Zhu, Z., 2017. Methods to enhance seismic faults and construct fault surfaces, *Computers & Geosciences*, **107**, 37–48.

Xiong, W., Ji, X., Ma, Y., Wang, Y., AlBinHassan, N.M., Ali, M.N. & Luo, Y., 2018. Seismic fault detection with convolutional neural network, *Geophysics*, **83**, O97–103.

Zhang, C., Frogner, C., Araya-Polo, M. & Hohl, D., 2014. Machine-learning based automated fault detection in seismic traces, in *Proceedings of 76th EAGE Conference and Exhibition*, EAGE, Amsterdam. DOI: 10.3997/2214-4609.20141500.

Zhou, L., Zhang, C. & Wu, M., 2018. D-LinkNet: linknet with pretrained encoder and dilated convolution for high resolution satellite imagery road extraction, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, IEEE, pp. 182–186.