GEOPHYSICS, VOL. 85, NO. 4 (JULY-AUGUST 2020); P. WA87–WA100, 17 FIGS., 4 TABLES.
10.1190/GEO2019-0252.1

DOI:10.1190/geo2019-0252.1

Deep learning for relative geologic time and seismic horizons

Zhicheng Geng[1], Xinming Wu[2], Yunzhi Shi[1], and Sergey Fomel[1]

ABSTRACT

Constructing a relative geologic time (RGT) image from a seismic image is crucial for seismic structural and stratigraphic interpretation. In conventional methods, automatic RGT estimation from a seismic image is typically based on only local image features, which makes it challenging to cope with discontinuous structures (e.g., faults and unconformities). We have considered the estimation of 2D RGT images as a regression problem, where we design a deep convolutional neural network (CNN) to directly and automatically compute an RGT image from a 2D seismic image. This CNN consists of three parts: an encoder, a decoder, and a refinement module. We train this CNN by using 2080 pairs of synthetic input seismic images and target RGT images, and then we test it on 960 testing seismic images. Although trained with only synthetic images, the network can generate accurate results on real seismic images. Multiple field examples show that our CNN-based method is significantly superior to conventional methods, especially in dealing with complex structures such as crossing faults and complicatedly folded horizons, without the need of any manual picking.

INTRODUCTION

The relative geologic time (RGT) volume (Stark, 2003, 2004) plays an important role in seismic interpretation. It has been successfully applied in horizon extraction (Lomask et al., 2006; Wu and Hale, 2015), sedimentologic interpretation (Zeng et al., 2012), stratigraphic interpretation (Karimi and Fomel, 2015; Zeng, 2016), and missing well-log data prediction (Bader et al., 2018). In an RGT volume, each data point is assigned an RGT value instead of the seismic reflection amplitude. A contour in an RGT volume represents a seismic horizon, which allows the RGT volume to store information about all seismic horizons. Some other geologic structural information, including faults and unconformities, can also be implicitly embedded as discontinuities in the RGT volume. An accurate RGT volume allows one to interpret all relevant geologic structures, including faults, horizons, and unconformities in the seismic image. Figure 1 shows a synthetic seismic image and the corresponding RGT image. Overlaying contours extracted from the RGT image on the seismic image, we generate Figure 1c. As shown in Figure 1, each contour consistently follows the structure of the seismic image, which makes seismic horizons easy to pick. It is also possible to directly detect faults from the shown RGT image. The calculation of other fault-related attributes such as fault displacement and dip can be straightforward.

Over the past decades, several methods have been proposed to generate RGT volumes. The most straightforward, but time-consuming, approach is to manually pick as many horizons as possible and interpolate them to obtain RGT volumes (Zeng et al., 1998a, 1998b; De Groot et al., 2006; de Bruin et al., 2007). The technique of unwrapping the seismic instantaneous phase (Stark, 2003; Wu and Zhong, 2012) can also be applied to generate RGT volumes because the time information is indicated in the phase of seismic data. Fomel (2010) uses slopes of seismic reflections estimated by the plane-wave-destruction method (Fomel, 2002) to generate an RGT volume, with good computational performance. Other authors (Bienati and Spagnolini, 2001; Lomask et al., 2006; Parks, 2010; Wu and Hale, 2013; Wu and Fomel, 2018) automatically extract horizons by fitting the horizon slopes with the seismic reflection slopes. However, obtaining a consistent horizon across faults is a challenge for all slope-based methods because they use local features and have no information about faults. Wu and Hale (2015) use several control points on both sides of the faults as constraints to generate a more accurate RGT volume, which requires some prior information. Luo and Hale (2013) and Wu et al. (2016) propose to first undo the faulting in a seismic image and then compute the horizons in the unfaulted space.

Manuscript received by the Editor 22 April 2019; revised manuscript received 25 December 2019; published ahead of production 16 March 2020; published online 30 April 2020.
[1]The University of Texas at Austin, John A. and Katherine G. Jackson School of Geosciences, Bureau of Economic Geology, University Station, Box X, Austin, Texas 78713-8972, USA. E-mail: zhichenggeng@utexas.edu; yzshi08@utexas.edu; sergey.fomel@beg.utexas.edu.
[2]University of Science and Technology of China, School of Earth and Space Sciences, Hefei, China. E-mail: xinmwu@ustc.edu.cn (corresponding author).

The recently developed deep-learning method (LeCun et al., 2015; Schmidhuber, 2015; Goodfellow et al., 2016) proved to be a powerful tool. Convolutional neural networks (CNN) form a class of deep neural networks that is efficient and effective in addressing computer vision problems, including image segmentation (Long et al., 2015; Ronneberger et al., 2015), image classification (Krizhevsky et al., 2012), and depth estimation (Eigen et al., 2014; Kuznietsov et al., 2017). Compared with conventional methods, deep learning extracts high-level information in a gradual manner without any input from domain expertise or manual feature extraction. In exploration geophysics, deep learning has already been successfully applied for automatic seismic interpretation including fault detection (Araya-Polo et al., 2017; Huang et al., 2017; Guo et al., 2018; Ma et al., 2018; Wu et al., 2019), salt detection (Waldeland and Solberg, 2017; Gramstad and Nickel, 2018; Waldeland et al., 2018; Shi et al., 2019), channel detection (Pham et al., 2019), facies classification (Zhao, 2018), and horizon picking (Lowell and Paton, 2018). However, all these applications try to extract only

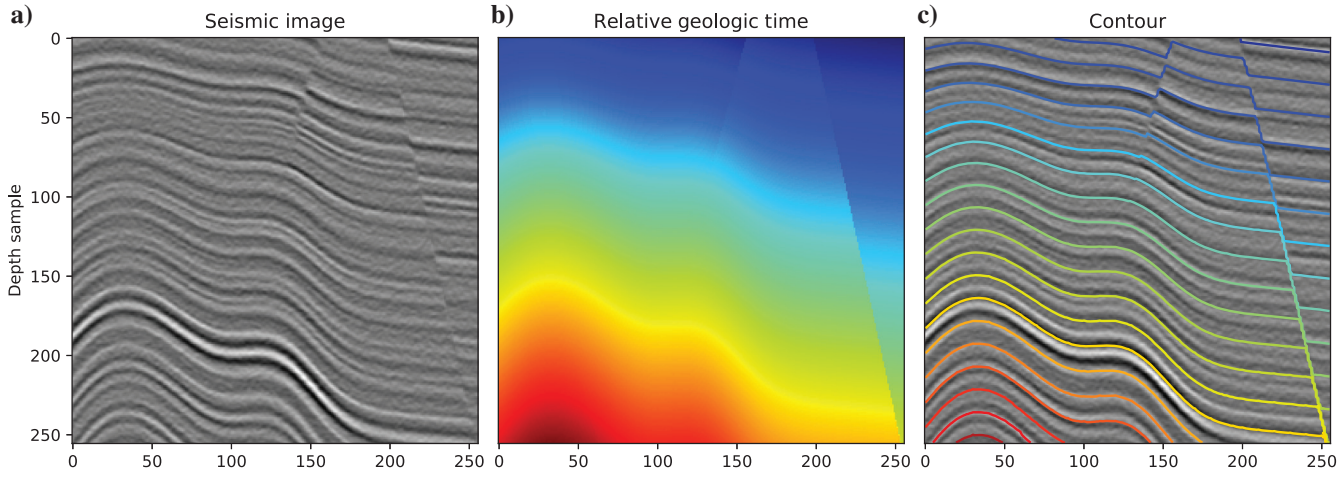

Figure 1. (a) A synthetic seismic image and (b) its corresponding RGT image from which (c) contours are extracted and overlaid on the seismic image.

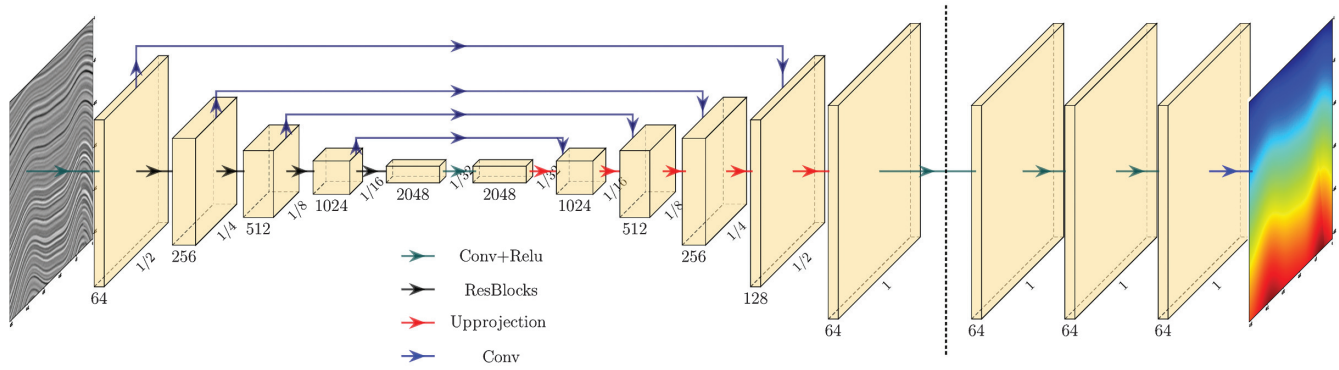

Figure 2. The proposed network architecture consists of an encoder, a decoder, and a refinement module.

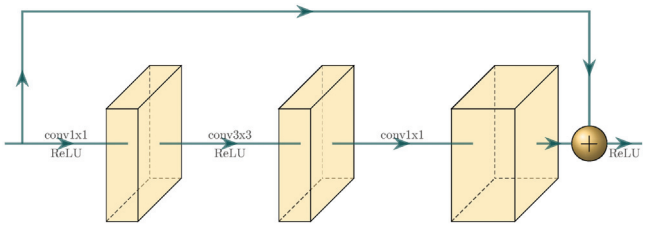

Figure 3. The structure of residual blocks.

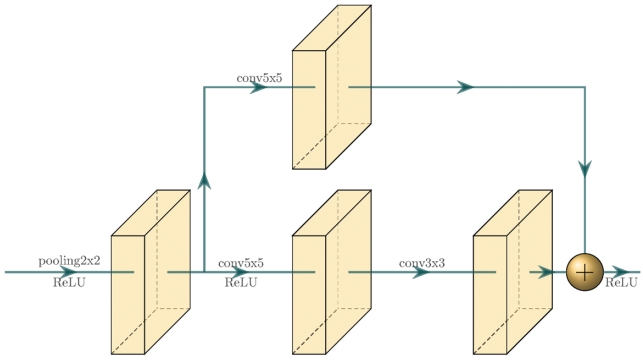

Figure 4. The structure of up-projections.

DOI:10.1190/geo2019-0252.1

one type of geologic structural information from seismic images whereas deep learning has the potential ability to provide more global structural information.

In this paper, we consider estimation of the 2D RGT image as a regression problem and propose to apply a U-net (Ronneberger et al., 2015) shape neural network architecture followed by a refinement network, to directly and automatically estimate an RGT image from a 2D input seismic image. The proposed network architecture is composed of three parts: an encoder, a decoder, and a refinement module. The encoder-decoder scheme extracts global features from input seismic image. The refinement module makes a prediction of an RGT image from the the encoder-decoder features. We use ResNet-50 (He et al., 2016) as the encoder module and load its pre-trained weights on ImageNet (Deng et al., 2009) as initialization weights for training, which allows us to construct a deep network. The network is implemented using PyTorch (Paszke et al., 2019). The workflow from Wu et al. (2019) generates synthetic seismic images and the corresponding RGT images to train the network. The mean squared error (MSE) is used as the loss function to help the network achieve better performance. We train the network by using only synthetic data sets and then apply it to estimating RGT images from real seismic images.

Table 1. Layers/blocks in our proposed neural network architecture.

Layer/block	Scale	Input/C[3]	Output/C
Conv1	1/2	3	64
Block1	1/4	64	256
Block2	1/8	256	512
Block3	1/16	512	1024
Block4	1/32	1024	2048
Conv2	1/32	2048	2048
Up1	1/16	2048	1024
Up2	1/8	1024	512
Up3	1/4	512	256
Up4	1/2	256	128
Up5	1	128	64
Conv3	1	64	64
Conv4	1	64	64
Conv5	1	64	1

[3]Input/C and Output/C columns show the number of channels of input and output at each layer/block.

METHOD

We consider RGT estimation to be a regression problem so that the RGT value of each data point is obtained directly from seismic images by using neural networks. In this section, we first introduce the proposed network architecture. We then describe the loss function and accuracy measurements for the RGT estimation problem.

Network architecture

In our proposed network architecture (Figure 2) the input of our network is a seismic image and the output is an RGT image, whose sizes are fixed as 256×256. The proposed network consists of an

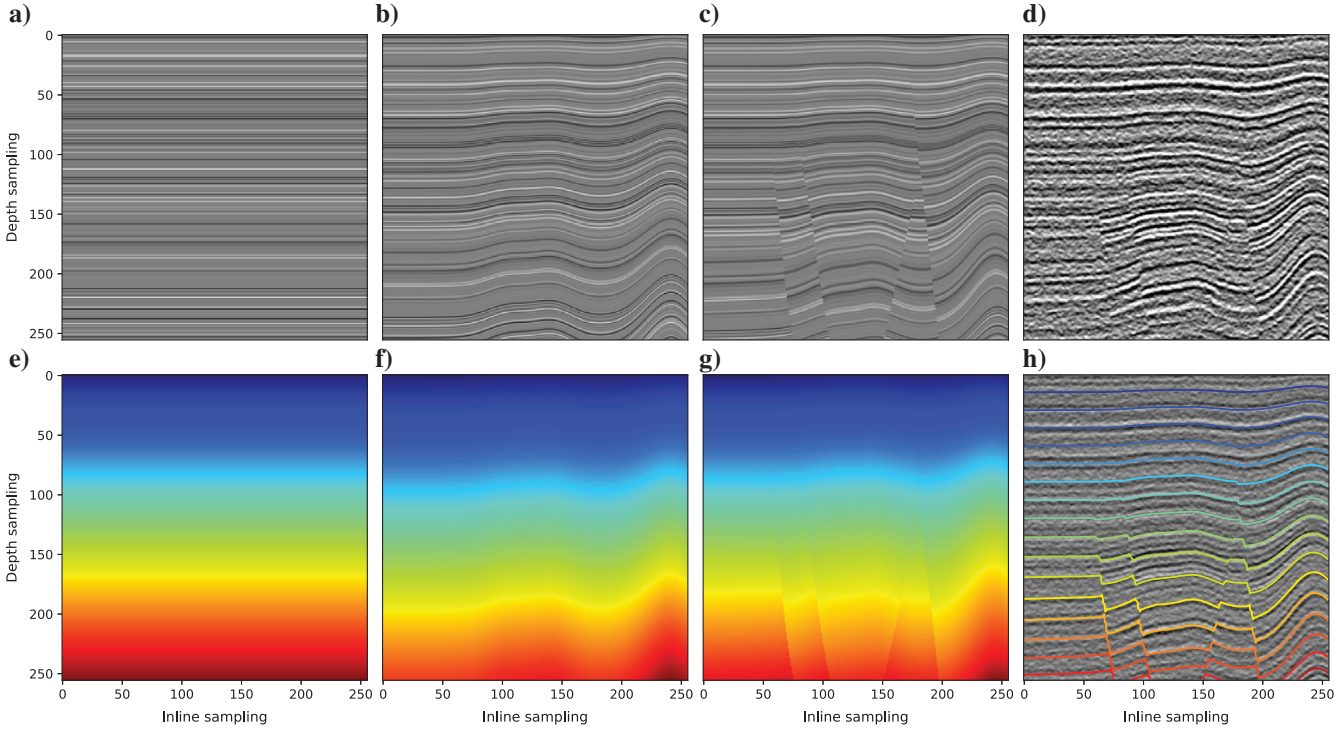

Figure 5. (a-d) A workflow of automatically generated synthetic seismic images and (e-g) the corresponding RGT images. (h) Contours/isovalue curves of the RGT image correspond to horizons that follow consistent seismic reflections.

encoder-decoder scheme followed by a refinement module. The encoder and decoder parts form a U-net (Ronneberger et al., 2015) shape network, and long skip connections are implemented to connect the encoder and decoder so that the spatial resolution of the output RGT image is improved.

ResNet-50 (He et al., 2016) is used as the encoder of our network. It is 50-layer residual nets (ResNet), which are designed to solve the degradation problem and increase the depth of neural networks. A stack of several residual blocks forms ResNet-50. Figure 3 shows the architecture of each residual block, which contains three successive convolution layers implemented by 1×1, 3×3, and 1×1 convolution filters, respectively. In the encoder part (Figure 2), the first 7×7 convolution layer `Conv1` with stride 2 downsamples the input to 1/2 scale. Then, four blocks (`Block1-Block4`), each consisting of a block of multiple residual blocks from ResNet-50, downsample the input to 1/4, 1/8, 1/16, and 1/32 scales, respectively. In this paper, we use the terms downsample or upsample to describe the process of reducing or increasing the sample rate of feature maps. Note that the fully connected layer in ResNet-50 is not used in our network.

For the decoder part, we use up-projection blocks (Laina et al., 2016). The building of up-projection blocks borrows the idea from He et al. (2016), and each up-projection block is an upsampling residual block, which passes the global information through the network more efficiently. The architecture of an up-projection block is shown in Figure 4. Five up-projection blocks (`Up1-Up5`) are implemented to upsample feature maps back to the same size as the input seismic image.

The encoder and the decoder are connected by a 1×1 convolutional layer `Conv2` and four long skip connections at different scales. In the refinement module, we use a stack of three convolutional layers (`Conv3-Conv5`) to give the final estimation result. The three layers are all 5×5 convolutions. Batch normalization and ReLU activation are applied to the output of all convolution layers except the last one (`Conv5`). Table 1 shows the details of each layer/block of our network.

Loss function and accuracy measurement

Loss function

In regression problems, one of the most commonly used loss function is the MSE, or $\mathcal{L}_2$ loss, which is the mean of the squared differences between the prediction and the ground truth:

$$\mathcal{L}_2 = \frac{1}{N}\sum_{i=1}^{N}(p_i - y_i)^2, \tag{1}$$

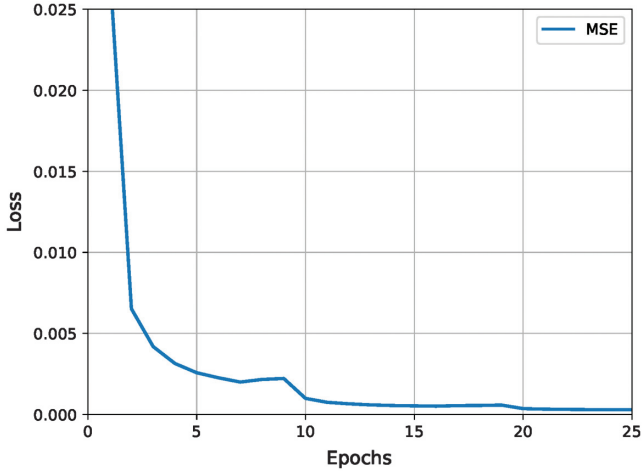

Figure 7. The history of MSE loss with respect to epochs.

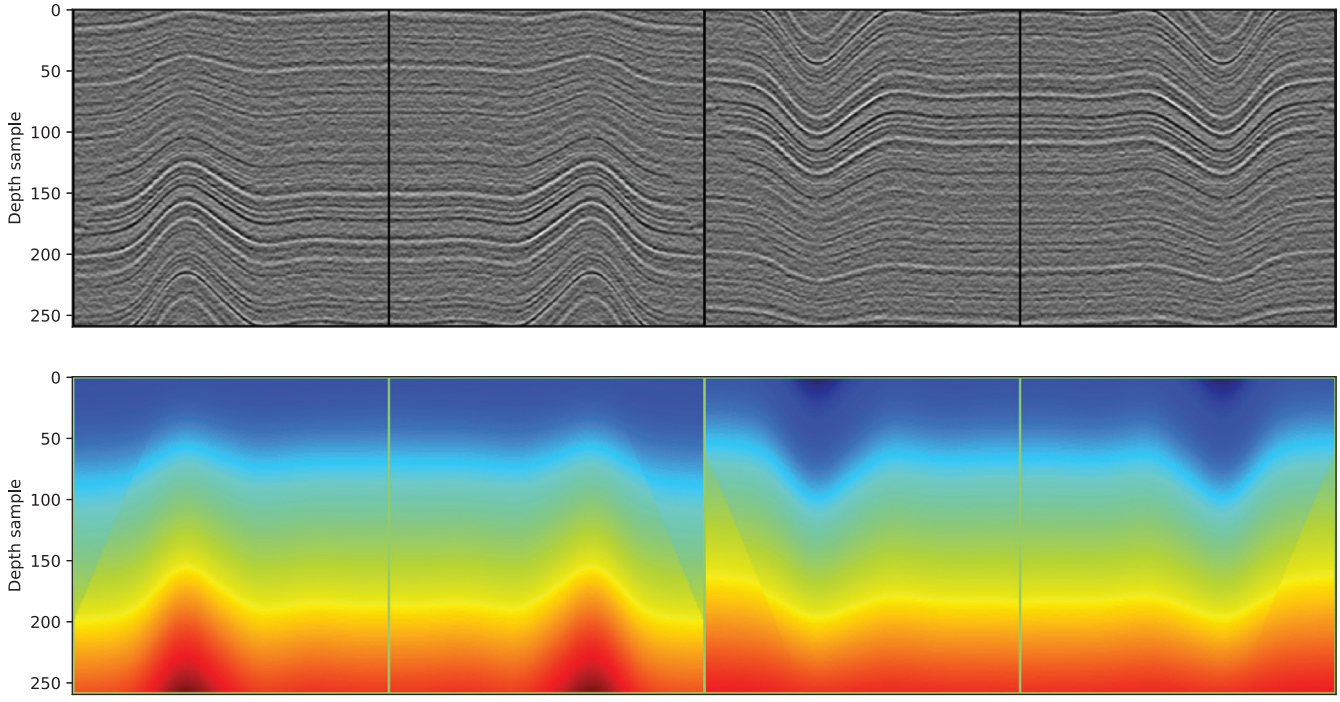

Figure 6. Example of data augmentation.

DOI:10.1190/geo2019-0252.1

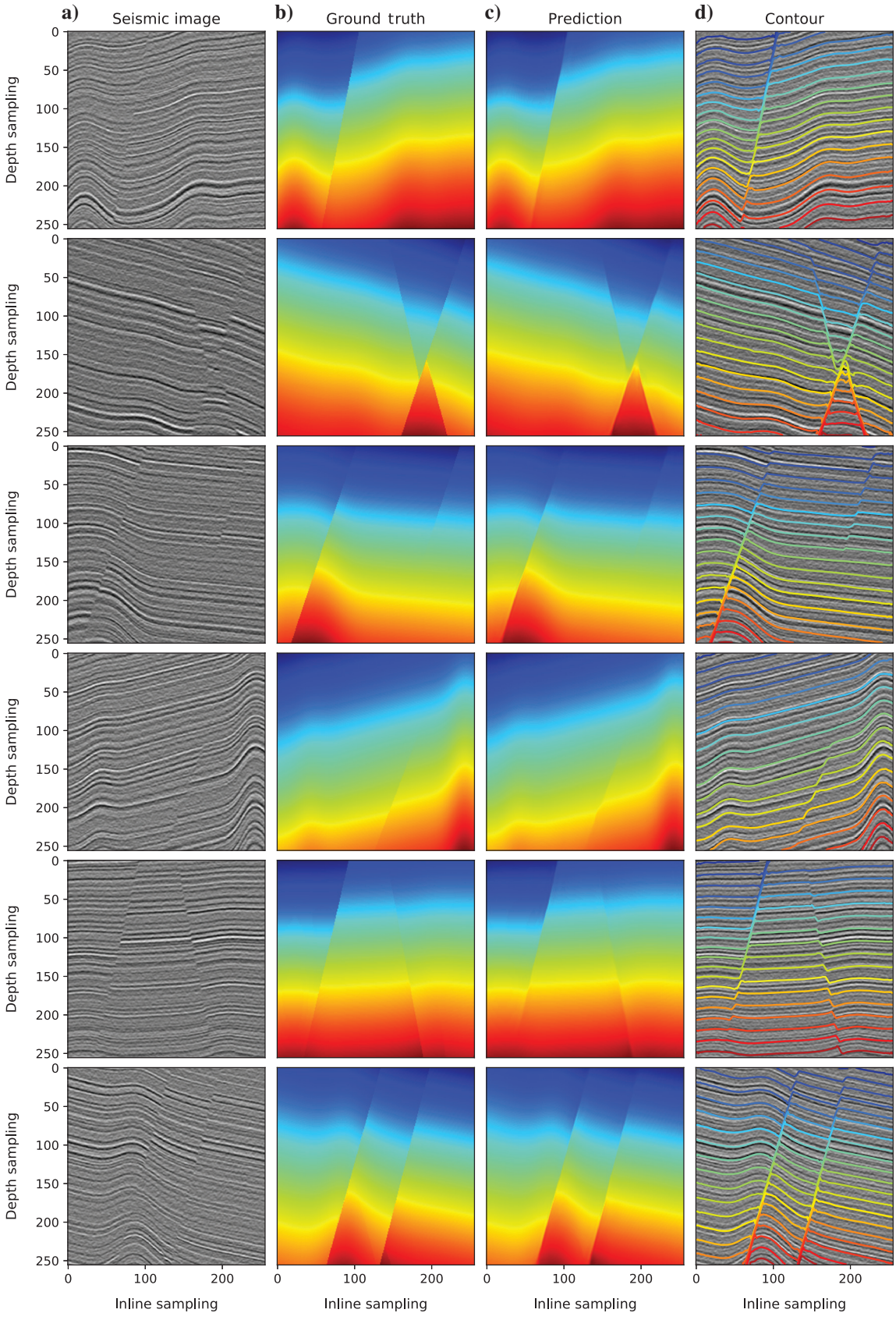

Figure 8. Panel (a): six selected seismic images as inputs to the network; (b): corresponding ground truth RGT images; (c): predictions from the left seismic images by using the trained network; (d): contours from the estimated RGT images.

where p_i denotes the prediction from the network and y_i denotes the ground truth. N represents the total number of pixels in the input seismic image. The term $\mathcal{L}_2$ loss is used as the loss function to train our network because it is easier to optimize and is able to produce a more stable solution.

Accuracy measurement

To evaluate the performance of our method, we use the following accuracy measurements:

1) root-mean-squared error (rms): $\sqrt{1/N\sum_{i=1}^{N}(p_i - y_i)^2}$
2) mean absolute error (MAE): $1/N\sum_{i=1}^{N}|p_i - y_i|$
3) mean relative percentage difference (MRPD): $2/N\sum_{i=1}^{N}|p_i - y_i|/|p_i| + |y_i|$

where N denotes the number of pixels in RGT images. Using these measurements, we are able to better quantitatively evaluate the performance of the network on the test data set.

EXPERIMENTS

In this section, we first illustrate how we generate the training data set. Then, we describe the setup of our experiment in detail. We also test the performance of the network on real data.

Training data sets

To train the network, we propose a workflow to automatically generate numerous synthetic seismic images and the corresponding ground truth of RGT images. The main limitation of applying CNN in geoscience problems is to prepare rich training data sets with reliable labels (Bergen et al., 2019). This is especially true in our problem of estimating RGT because it is hard to automatically or even manually obtain a reliable RGT image from a real seismic image.

In this workflow of generating a synthetic seismic image, we begin with an initially flat reflectivity model (Figure 5a), which is generated by horizontally extending a 1D trace of random reflectivity values. We then sequentially add some folding (Figure 5b) and faulting (Figure 5c) in this reflectivity model. We further convolve the reflectivity model with a wavelet (with a randomly chosen peak frequency), and we add some random noise to obtain the synthetic seismic image shown in Figure 5d. As discussed by Wu et al. (2019), applying convolution after the folding and faulting is helpful to simulate a more realistic fault in the seismic image (Figure 5d) because the convolution smears the sharp fault in Figure 5c. To generate a corresponding RGT image, we start with an initial RGT image (Figure 5e) with vertically monotonic values and horizontally constant values. We then add exactly the same folding (Figure 5f) and faulting (Figure 5g) as in the seismic image to this initial RGT image. The contours or isovalue curves (shown as the colorful curves in Figure 5h) of the folded and faulted RGT image will consistently follow the structures within the seismic image.

Table 2. Comparison of different network architecture (with or without refinement module) on the test data set.

	rms error	MAE	MRPD
CNN without refinement	0.070	0.047	0.127
CNN with refinement	0.061	0.041	0.116

Note: For all metrics, smaller is better.

After creating synthetic seismic images and the corresponding RGT images, a simple data augmentation is applied to enlarge the training data set and boost the performance of the network. The data augmentation has two steps: First, we horizontally flip an input seismic image; then the two horizontally symmetric images are flipped vertically. In this way, we obtain four seismic images from one input seismic image. For the corresponding target RGT image, the first process is the same and an RGT image is flipped horizontally to generate two RGT images. The second process, however, is different. Because the RGT value of each data point is supposed to be monotonically increasing with respect to the depth, if we just flip RGT images vertically, the RGT values would monotonically decrease, which is unrealistic in a geologic sense. Note that before data augmentation, we have already normalized the input seismic images and the target RGT images. Due to the fact that only the relationship between points is considered in an RGT image instead of the actual values, each target is subtracted by its mean and divided by its standard deviation in the normalization. Therefore, we negate the vertically flipped RGT image, which makes the RGT values monotonically increase without changing the mean and standard deviation of the RGT image.

Figure 6 is an example illustrating how we augment the input seismic image and the target RGT images. The top row shows four seismic images obtained from data augmentation, and the bottom row shows the corresponding target RGT image. The first column of Figure 6 is the input seismic image generated by the method described above and the corresponding target. The second column is the horizontally flipped seismic image and RGT. After vertically flipping seismic images and RGT images from the first two columns and negating RGT values, we obtain the last two columns. Then, all seismic images from the top row are fed to the network, and the RGT images serve as the target output, to calculate the loss function to train the network.

Implementation details

We implement our RGT estimation network using PyTorch (Paszke et al., 2019) version 1.0. The network is trained on four NVIDIA GTX 1080 Ti GPUs, each with 11 GB memory. Before training, we load the pretrained weights of ResNet-50 on the ImageNet data set to initialize the encoder module to improve the performance, whereas the weights of the decoder part are initialized randomly. In practice, the input seismic image is duplicated twice along the channel dimension to use the pretrained weights because images from ImageNet dataset all have three channels.

In total, 2080 pairs of synthetic seismic images and RGT images are generated for training the network and another 960 pairs are used for testing. As mentioned in the previous section, the input and target are normalized in an appropriate way. To relieve the effects caused by different ranges of amplitudes of synthetic and real data, each input seismic image is normalized by its minimum and maximum value, which means that the range of amplitudes becomes 0 to 1. The normalization method applied to RGT images is different. We normalize them by their mean and standard deviation to the range of -1 to 1. After normalization, data augmentation described in the last section is applied to seismic images and RGT images. We use these synthetic data to train the proposed neural network for 25 epochs with a batch

DOI:10.1190/geo2019-0252.1

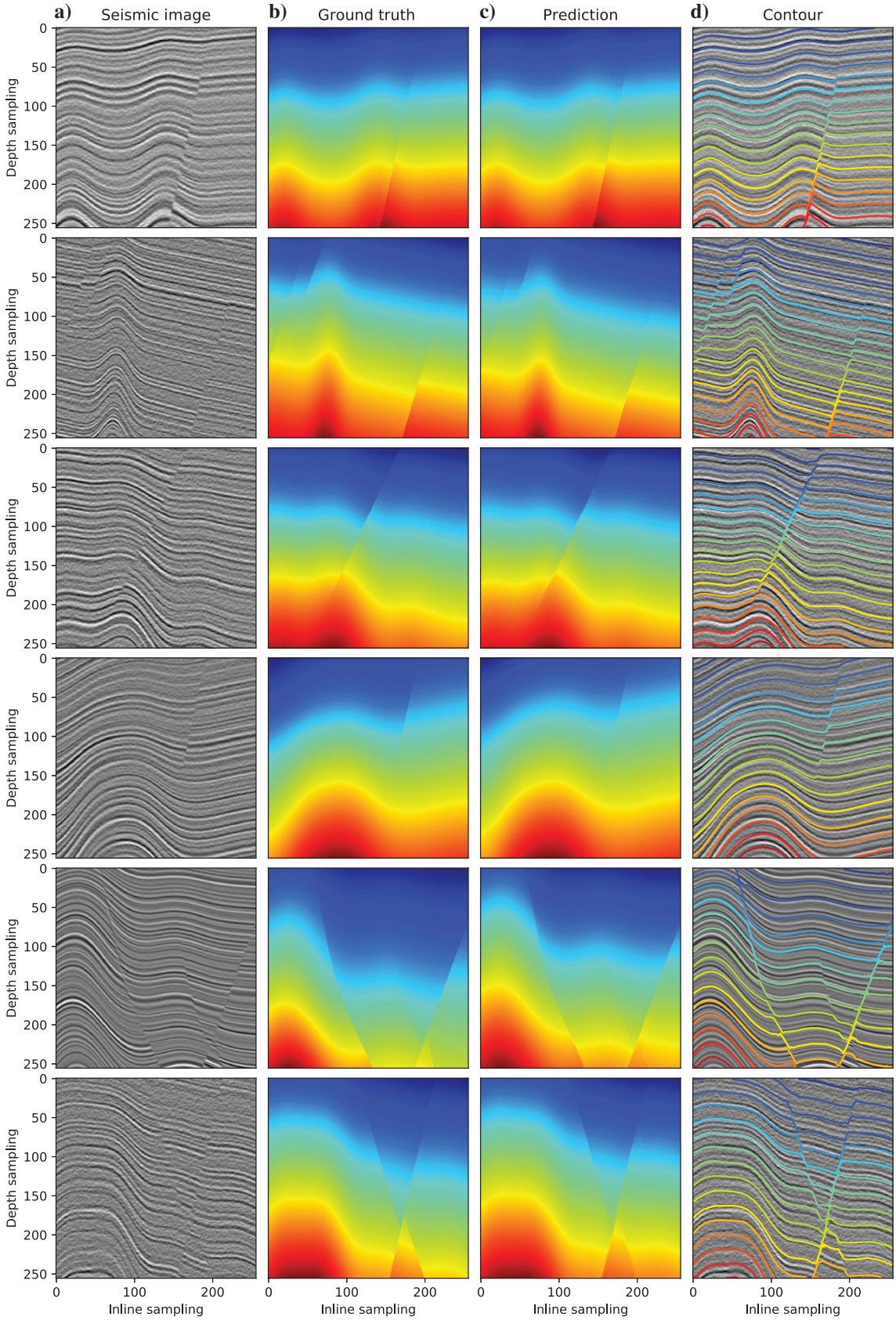

Figure 9. Panel (a): six selected seismic images as inputs to the network; (b): corresponding ground truth RGT images; (c): predictions from the left seismic images by using the trained network; (d): contours from the estimated RGT images.

size of four. In deep learning, one epoch means that each data in the training data set is passed forward and backward through the neural network only once. During training, Adam (Kingma and Ba, 2014) optimization is used to estimate the model parameters. The initial learning rate is 0.0001, and we divide it by 10 for every 10 epochs. We set $\beta_1 = 0.9$, $\beta_2 = 0.999$, and weight decay as 0.0001.

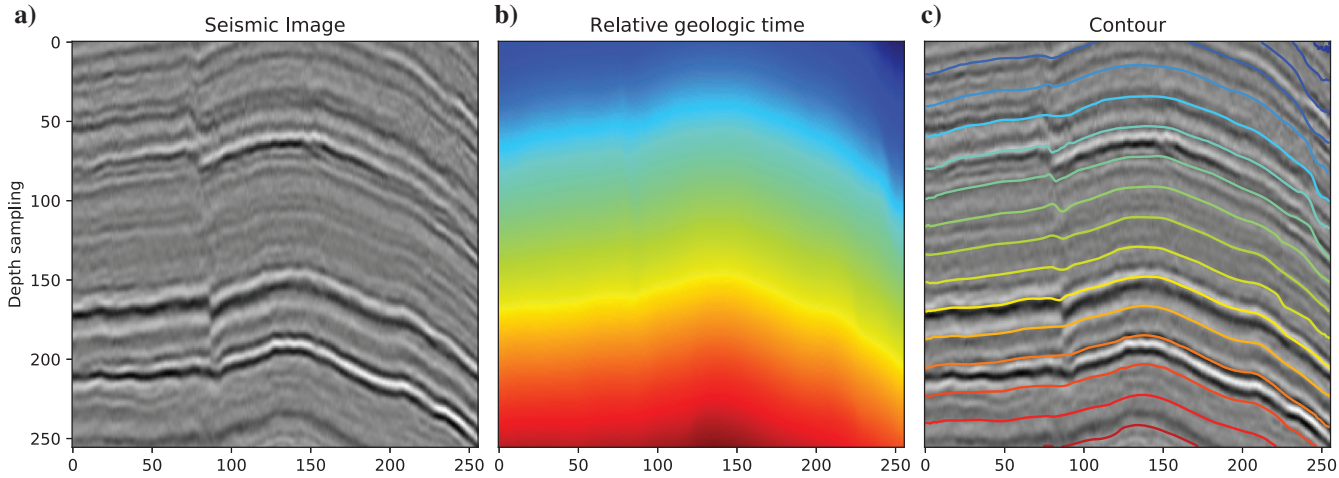

Figure 10. (a) A real seismic image, (b) the estimated RGT image from the network with (a) as the input, and (c) overlaying contours of RGT values on (a).

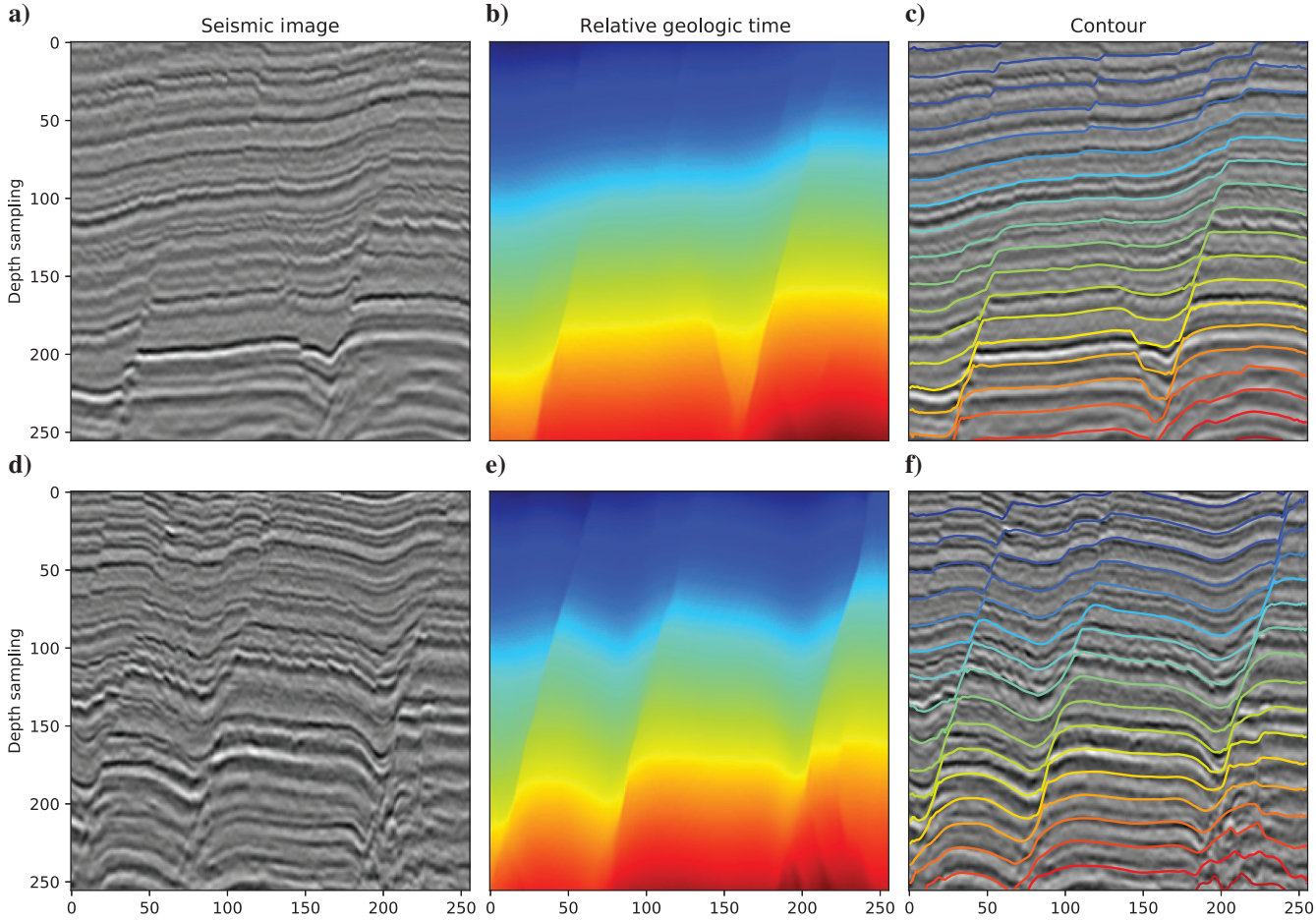

Figure 11. (a) and (d) Real seismic images, (b) and (e) estimated RGT images from the network with (a) and (d) as the input, and (c) and (f) overlaying contours of the RGT values on (a) and (d).

DOI:10.1190/geo2019-0252.1

As shown in Figure 7, the training loss converges to less than 0.0001 after 25 epochs, which shows an excellent training process. To further validate the performance of the trained network on training data set, we take six seismic images from the training data set as input to the network. Predictions from the network are shown in the third column of Figure 8. They are mostly the same as the ground truth shown in the second column of Figure 8. In the estimated RGT image, faults appear sharp, making them easy to detect. As shown in the right column of Figure 8, contours from the outputs of the network accurately follow the horizons of seismic images, even across faults. Comparisons of predictions to ground truth demonstrate that the trained network learns patterns from training data sets very well and it has remarkable performance on training data.

Table 2 shows the quantitative comparison of our method with and without the refinement module. Lower scores in all three metrics indicate that deeper network with refinement module has better

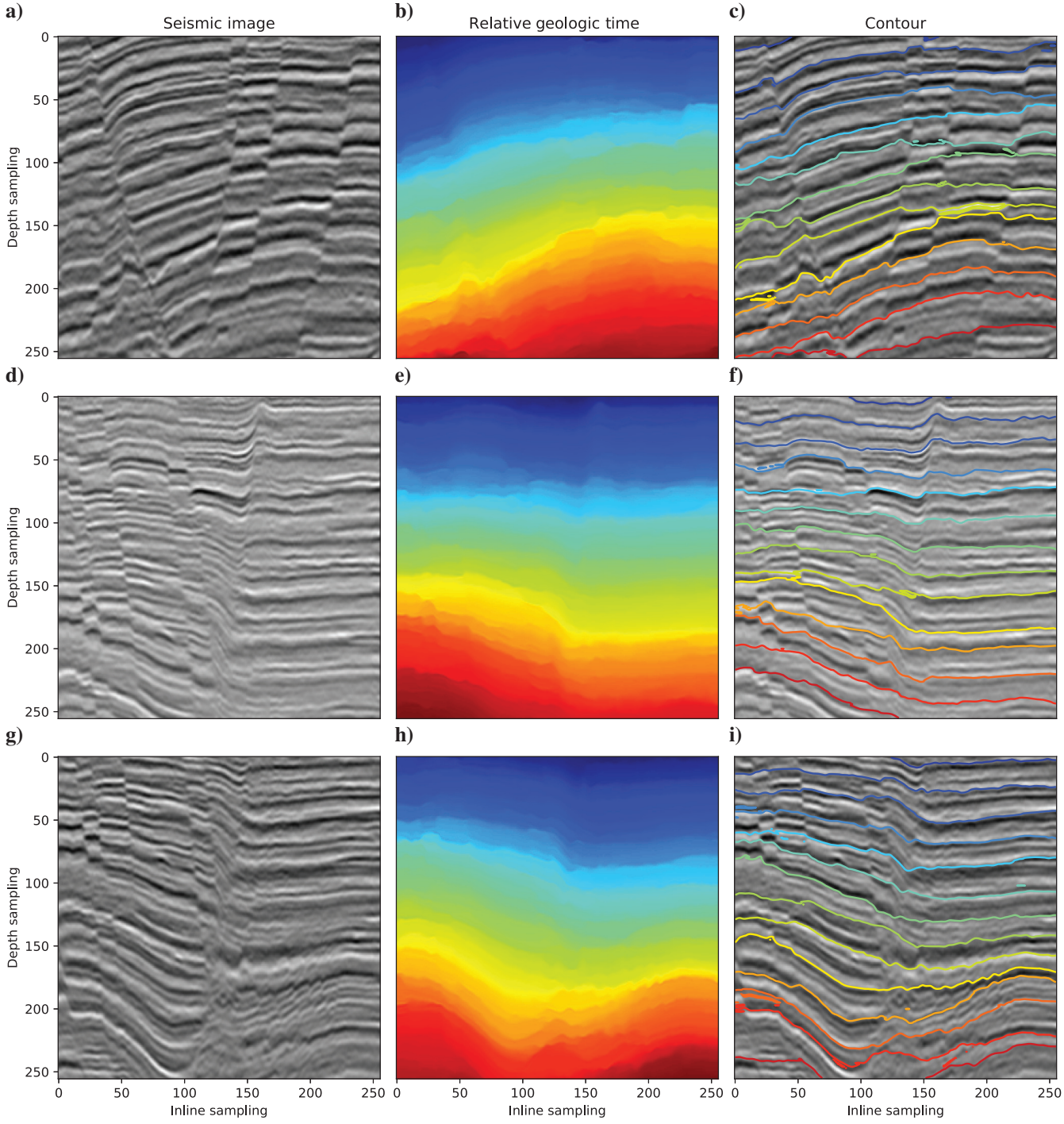

Figure 12. (a), (d), and (g) Real seismic images, (b), (e), and (h) estimated RGT images from the predictive painting with (a), (d), and (g) as the input, respectively, and (c), (f), and (i) overlaying contours of RGT values on (a), (d), and (g).

performance. Figure 9 shows examples of predictions on the testing data set. Consistent horizons crossing faults are accurately tracked and faults are correctly recovered, which shows the significant performance of our method.

Applications

Although the network is trained using only synthetic data set, it has excellent performance on field data. In this section, we test the trained network on several real seismic images from four field data sets to further demonstrate its powerful performance. Different from the processing for training data, these field seismic images are not augmented and are directly fed to the network after copying them twice along the channel dimension.

The first input field seismic image is a portion of Teapot Dome data set, as shown in Figure 10a. The size of this portion is consistent with the size of the training data, which is 256×256. Figure 10b shows the estimated RGT image from the network,

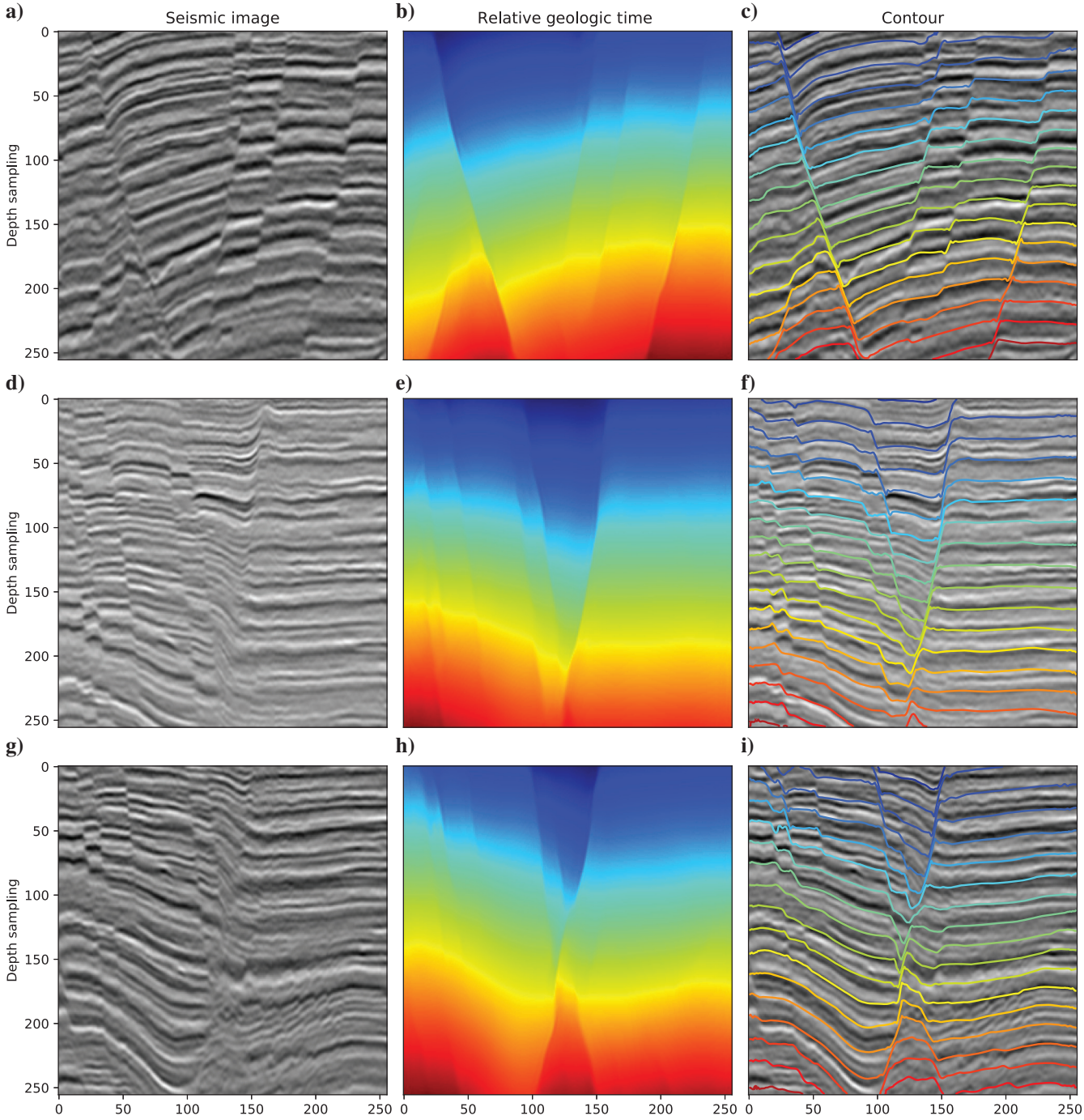

Figure 13. (a), (d), and (g) Real seismic images, (b), (e), and (h) estimated RGT images from the network with (a), (d), and (g) as the input, respectively, and (c), (f), and (i) overlaying contours of RGT values on (a), (d), and (g).

DOI:10.1190/geo2019-0252.1

and Figure 10c shows contours overlaying on the input seismic image. The trained network has excellent performance on this field data, even though the distribution of reflectivity and the length of the wavelet is totally different from the training data. From the estimated RGT image, it is easy to detect the fault with large dip directly. Information about consistent horizons is also contained in the estimated RGT image because contours perfectly follow the horizons across the fault (Figure 10c).

The second example is two real seismic images from Opunake-3D. As shown in Figure 11a and 11d, the first image is very clean and the second one has more noise with faults that have different patterns from the data used for training. The estimated RGT images and their contours are shown in Figure 11b, 11c, 11e, and 11f. Most of the horizons in Figure 11a are flat, so not surprisingly the network works well on these data. The proposed CNN model even detects two small faults at the top part of the image, which are obvious from contours Figure 11c. For the second noisy input seismic image, even though the fault patterns are much different from the training data, the network produces an accurate RGT image with sharp faults, which means the proposed method learns how to estimate an RGT image in a global sense instead of remembering patterns from the training data. Consistent horizons across faults are shown in Figure 11f.

The third example consists of three input seismic images (Figure 13a, 13d, and 13g), containing several closely spaced faults. Obtaining reasonable RGT images from this data set is a challenge for a conventional method due to the presence of faults and noise. Also, reflections of these three images are not clear, which poses problems for the calculation of RGT images. Results from the predictive painting (Fomel, 2010) are shown in Figure 12. Part of the estimated RGT images is accurate with contours exactly following the horizons. However, it fails to correctly track horizons across faults. Compared to predictive painting, the CNN model has much better performance on all of these input images, as shown in Figure 13b, 13e, and 13h. It detects all faults from seismic images and shows sharp edges in RGT images. Even some small spaced faults, for example in the middle of Figure 13a and 13d and on the left part of Figure 13g, are clearly detected. With the presence of multiple closely spaced faults, consistent horizons are still easy to pick from contours of RGT images, which shows the power of CNN.

The last example from the westCam data set (Figure 14a) is the most challenging one. This seismic image is highly contaminated by noise. Although most horizons are flat, there is one set of crossing faults and multiple sets of faults that are close to each other, increasing the difficulty of the problem. However, results from the network show the effectiveness of deep learning. As shown in Figure 14b, the contours of the estimated RGT accurately follow geologically consistent reflections even across the complex intersecting faults as shown in Figure 14c. The horizons are obviously dislocated across faults, where the dislocation amounts represent fault displacements. The estimated RGT image has sharp faults even for the cross faults.

Although the input size is fixed as 256×256, our method can be applied to field seismic images with various sizes. For a seismic image with a larger size, we can first split a the image into several 256×256 subimages with some overlap and then estimate an RGT image for each of the subimages by using the trained network. We further start extracting horizons from one of the estimated RGT images. We then extend the horizons from the current subimage to its overlapping subimages by using the corresponding RGT images. By doing this, we are able to extract full horizons that consistently follow reflections in the whole large seismic image. In this method, we do not need to merge the RGT images, which might not be trivial. We only need to extract contours from each RGT image.

Figure 15 shows an example of our strategy applied to field data (Figure 15a), whose size is 256×598. From this seismic image, we extract three overlapping subimages as shown in Figure 15b–15d, which correspond to the seismic sections inside the red, blue, and cyan rectangles (Figure 15a), respectively. The overlap width of the subimages in Figure 15b and 15c is 64 samples, whereas the width of the overlap for Figure 15c and 15d is 106 samples. Figure 15e–15g shows the RGT images estimated for the subimages (Figure 15b–15d) by using the trained neural network. To extract horizons going through the whole seismic image (Figure 15a), we start tracking the horizons in the left subimage (Figure 15h) that are extracted as contours of the corresponding RGT image (Figure 15e). The purple points in Figure 15h denote the locations where the horizons reach

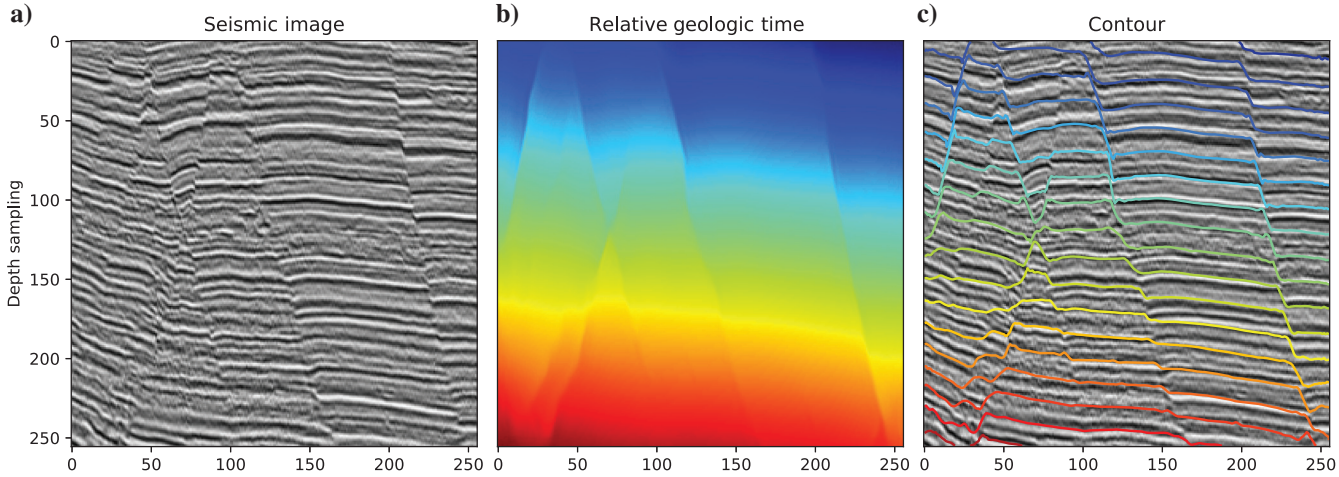

Figure 14. (a) A real seismic image, (b) the estimated RGT image from the network with (a) as the input, and (c) overlaying contours of RGT values on (a).

the middle trace of the overlap region. We choose to stop tracking at the middle traces to avoid boundary effects because usually we do not have enough confidence in the results at the boundary from the deep neural network. We continue tracking the horizons in the second overlapping subimage (Figure 15i) by extracting the contours passing through the purple points in the corresponding RGT image (Figure 15f). Similarly, we can consistently extend the horizons from the second subimage to the third subimage as shown in Figure 15g. Finally, we concatenate all horizons from the subimages (Figure 15h–15j) to obtain the full horizons that consistently follow reflections across faults in the whole seismic image (Figure 15k).

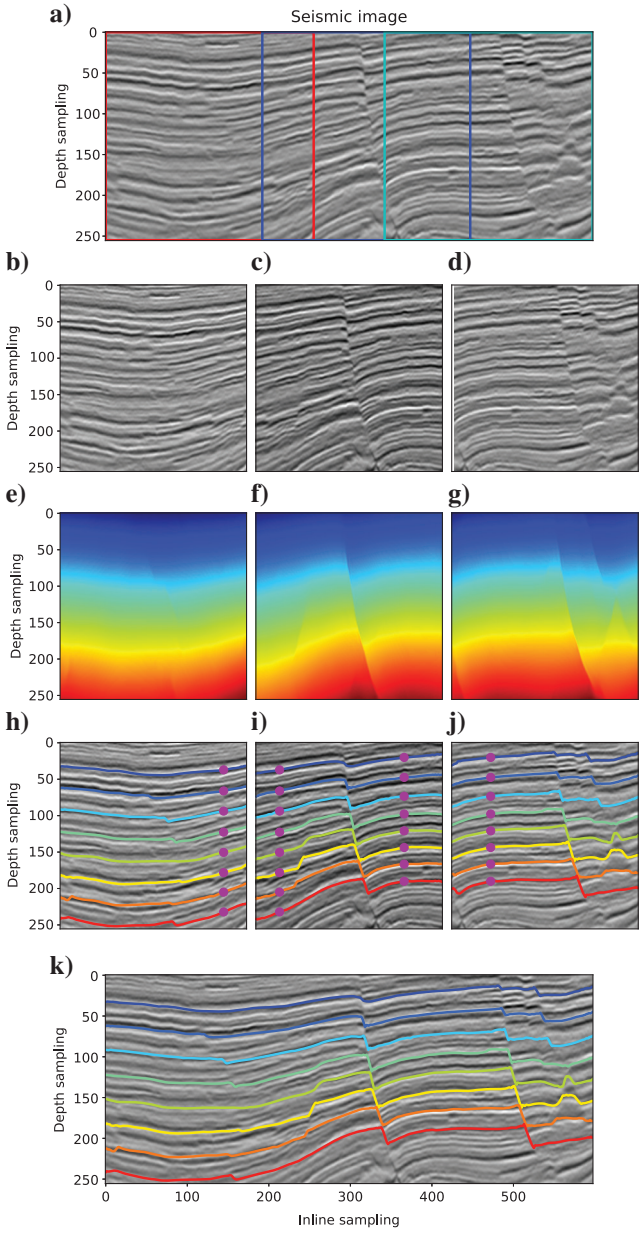

Figure 15. (a) A real seismic image, (b), (c), and (d) seismic images inside red, blue, and cyan rectangles in (a), respectively, and (e), (f), and (g) their estimated RGT images, (h), (i), and (j) extracted horizons from RGT images and (k) full horizons for the whole seismic images.

DISCUSSION

In this paper, we propose a neural network architecture to estimate an RGT volume from a seismic image. We have used a fixed input size for training the network, which, however, can be applied to a seismic image with a different size. Trained by the current data sets, the network works well on field images with complicated faulting and folding structures. To make the network more generalized, we will need to add more complex geologic structures (e.g., salt bodies, unconformities, and igneous intrusions) to the training data set.

Input size

In our method, the input size of the proposed neural network is fixed as 256×256. For an encoder-decoder architecture, the size of input training and testing data can vary, but this might work only for local problems such as fault detection and salt body detection. The reason why it does not work for global problems is mainly because of the concept of the receptive field. The receptive field of a neuron in CNN is the region in the original image that affects the output feature map. The receptive field in the higher layer is the combination of receptive fields from all of the previous layers. For example, considering two convolutional layers stacking together, each with a 3×3 kernel, the receptive field size for the first layer is 3×3, which means that each pixel in the output feature map of the first layer is affected by a 3×3 region of the input image, whereas the receptive field size is 5×5 for the second convolutional layer. Notice that the receptive field size increases by stacking several convolutional layers together. Therefore, a deeper network in principle would have larger receptive fields. This is how CNN gradually extracts global features from the original image. For global problems, the size of input to the network is supposed to be smaller than the receptive field of the whole network. Otherwise, the expected results might not be achieved by the neural network.

To estimate an RGT image, we need relationships between all points in the seismic image, which has to be considered as a global problem. The receptive field for the encoder part (ResNet-50) of our network is 483×483, which limits the input size of our proposed network. In our experiment, the input size is fixed as 256×256 for easier data preparation, which is not necessarily in practice.

Although the input size is limited, our method can be applied to seismic images with varying size by splitting images into multiple subimages with the fixed size that we use in our experiment. RGT images are then estimated from these subimages and are used to recursively extract horizon segments and finally concatenate them to form full horizons as discussed in the previous section. Another approach is to directly merge all of the estimated RGT images to obtain a full RGT image for the whole seismic image.

Generalization

Due to the limited training data sets with mostly faulting and folding structures, the trained model currently works well in complicated faulted and folded seismic images but may fail in examples with salt bodies, igneous intrusions, and unconformities not included in the training data sets. In other words, a trained network

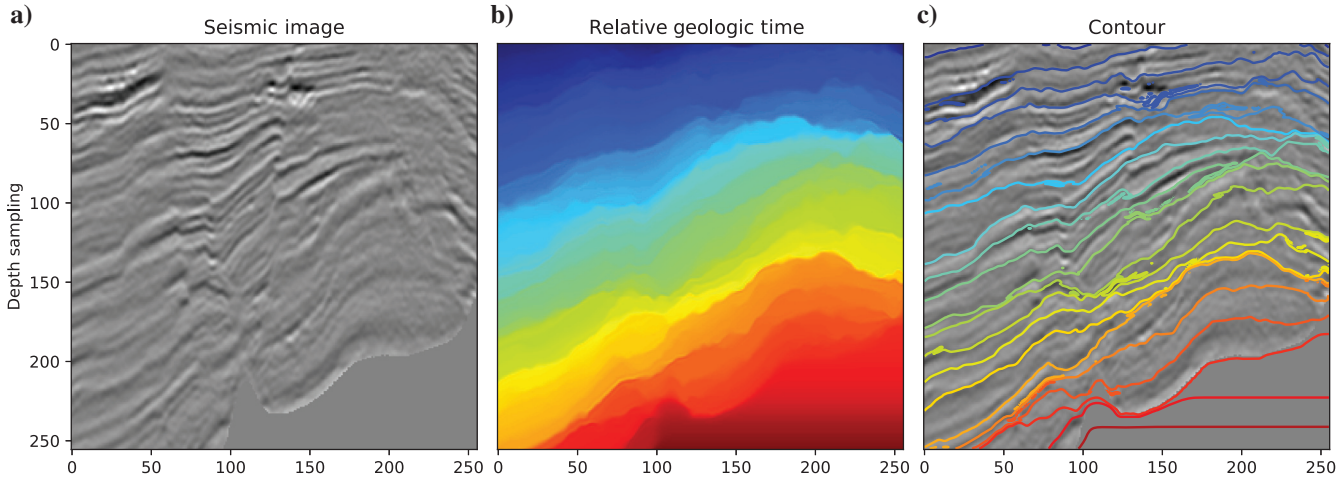

Figure 16. (a) A real seismic image, (b) the estimated RGT image from the predictive painting with (a) as the input, and (c) overlaying contours of RGT values on (a).

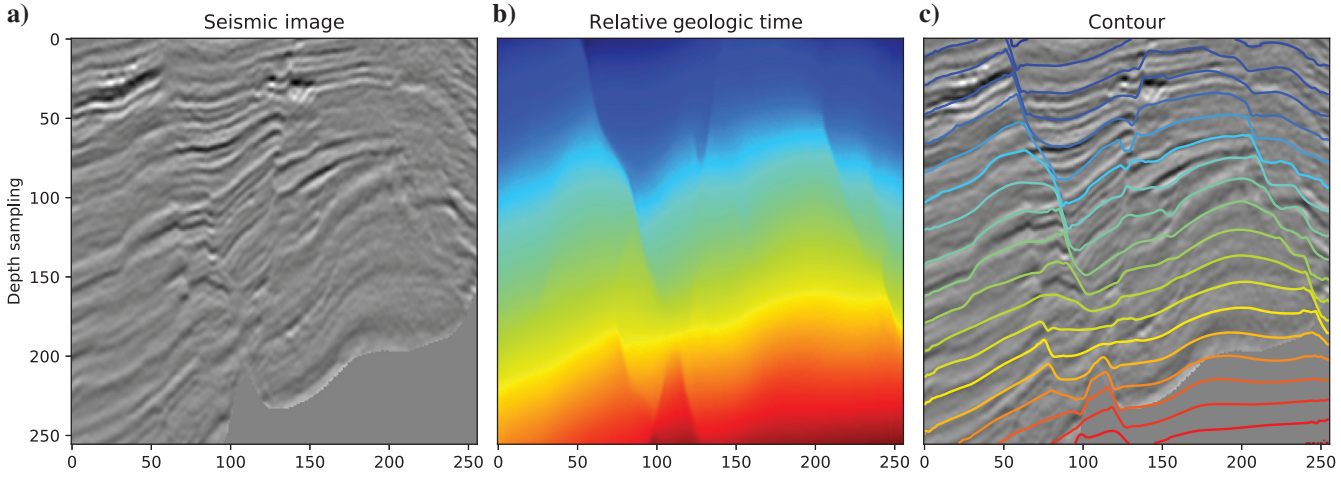

Figure 17. (a) A real seismic image, (b) the estimated RGT image from the network with (a) as the input, and (c) overlaying contours of RGT values on (a).

might only have good performance on some test data, which have similar patterns to the training data set.

Figure 17 shows a failed application of our method. The input seismic image (Figure 16a) contains faults and part of a salt body, which is masked out from this image. The major difference between fault and horizon patterns from this test data and the training data causes our method to fail. Additionally, our training data set consists of synthetic seismic images only with faulting and folding structures. Therefore, the deep neural network wouldn't be able to learn how to extract features from more complex geologic structures, including salt bodies and unconformities, as shown in Figure 17. Predictive painting, based on local filters, also fails to follow horizons crossing the middle and right faults on this image, as shown in Figure 16. Compared to Figure 16, the deep learning still achieves a relatively better result.

However, from our current promising results, the deep neural network shows the potential to generate accurate RGT images for a specific kind of input images, which can contain one certain complex geologic structure such as salt bodies, igneous intrusions, and unconformities. Considering the small amount of training data that we used, more realistic training data with specific complicated geologic structures or even some real seismic images with interpreted RGT images can be added to the training data set to improve the performance of the neural network, which is also what we are planning to do in the future.

CONCLUSION

We consider the RGT estimation as a regression problem and design a deep neural network to estimate an RGT image directly and automatically from an input seismic image. The proposed network has an encoder-decoder scheme, which helps to learn patterns from the training data in an incremental manner and an extra refinement module is added to the network. A workflow to build realistic models is used to generate synthetic seismic images and corresponding RGT images to train and test the network. Although trained by using only synthetic data, the proposed neural network produces accurate estimation of RGT images on real seismic images with complicated

faulting and folding structures. An extension of the proposed method to 3D is straightforward but requires additional computing resources.

ACKNOWLEDGMENTS

We thank the financial support by the National Science Foundation of China under grant no. 41974121 as well as the sponsors of the Texas Consortium for Computational Seismology (TCCS). The Texas Advanced Computing Center (TACC) provided the computational resources for this study. We thank the associate editor Wenyi Hu, reviewer Haibin Di, and two anonymous reviewers for providing valuable suggestions.

DATA AND MATERIALS AVAILABILITY

Data associated with this research are available and can be obtained by contacting the corresponding author.

REFERENCES

Araya-Polo, M., T. Dahlke, C. Frogner, C. Zhang, T. Poggio, and D. Hohl, 2017, Automated fault detection without seismic processing: The Leading Edge, **36**, 208–214, doi: 10.1190/tle36030208.1.

Bader, S., K. Spikes, and S. Fomel, 2018, Missing well-log data prediction using Bayesian approach in the relative-geologic time domain: 88th Annual International Meeting, SEG, Expanded Abstracts, 804–808, doi: 10.1190/segam2018-2997278.1.

Bergen, K. J., P. A. Johnson, V. Maarten, and G. C. Beroza, 2019, Machine learning for data-driven discovery in solid earth geoscience: Science, **363**, eaau0323, doi: 10.1126/science.aau0323.

Bienati, N., and U. Spagnolini, 2001, Multidimensional wavefront estimation from differential delays: IEEE Transactions on Geoscience and Remote Sensing, **39**, 655–664, doi: 10.1109/36.911122.

de Bruin, G., N. Hemstra, and A. Pouwel, 2007, Stratigraphic surfaces in the depositional and chronostratigraphic (Wheeler-transformed) domain: The Leading Edge, **26**, 883–886, doi: 10.1190/1.2756868.

De Groot, P., G. De Bruin, and K. McBeath, 2006, OpendTect SSIS-Sequence stratigraphic interpretation system: Drilling and Exploration World, **15**, 31–34.

Deng, J., W. Dong, R. Socher, L. Li, K. Li, and F. Li, 2009, ImageNet: A large-scale hierarchical image database: IEEE Conference on Computer Vision and Pattern Recognition, 248–255.

Eigen, D., C. Puhrsch, and R. Fergus, 2014, Depth map prediction from a single image using a multi-scale deep network: Advances in Neural Information Processing Systems, 2366–2374.

Fomel, S., 2002, Applications of plane-wave destruction filters: Geophysics, **67**, 1946–1960, doi: 10.1190/1.1527095.

Fomel, S., 2010, Predictive painting of 3D seismic volumes: Geophysics, **75**, no. 4, A25–A30, doi: 10.1190/1.3453847.

Goodfellow, I., Y. Bengio, and A. Courville, 2016, Deep learning: MIT press.

Gramstad, O., and M. Nickel, 2018, Automated interpretation of top and base salt using deep convolutional networks: 88th Annual International Meeting, SEG, Expanded Abstracts, 1956–1960, doi: 10.1190/segam2018-2996306.1.

Guo, B., L. Li, and Y. Luo, 2018, A new method for automatic seismic fault detection using convolutional neural network: 88th Annual International Meeting, SEG, Expanded Abstracts, 1951–1955, doi: 10.1190/segam2018-2995894.1.

He, K., X. Zhang, S. Ren, and J. Sun, 2016, Deep residual learning for image recognition: IEEE Conference on Computer Vision and Pattern Recognition, 770–778.

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.

Karimi, P., and S. Fomel, 2015, Stratigraphic coordinates: A coordinate system tailored to seismic interpretation: Geophysical Prospecting, **63**, 1246–1255, doi: 10.1111/1365-2478.12224.

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

Krizhevsky, A., I. Sutskever, and G. E. Hinton, 2012, Imagenet classification with deep convolutional neural networks: Advances in Neural Information Processing Systems, 1097–1105.

Kuznietsov, Y., J. Stiickler, and L. Bastian, 2017, Semi-supervised deep learning for monocular depth map prediction: IEEE Conference on Computer Vision and Pattern Recognition, 2215–2223.

Laina, I., C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, 2016, Deeper depth prediction with fully convolutional residual networks: IEEE International Conference on 3D Vision (3DV), 239–248.

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

Lomask, J., A. Guitton, S. Fomel, J. Claerbout, and A. A. Valenciano, 2006, Flattening without picking: Geophysics, **71**, no. 4, P13–P20, doi: 10.1190/1.2210848.

Long, J., E. Shelhamer, and T. Darrell, 2015, Fully convolutional networks for semantic segmentation: IEEE Conference on Computer Vision and Pattern Recognition, 3431–3440.

Lowell, J., and G. Paton, 2018, Application of deep learning for seismic horizon interpretation: 88th Annual International Meeting, SEG, Expanded Abstracts, 1976–1980, doi: 10.1190/segam2018-2998176.1.

Luo, S., and D. Hale, 2013, Unfaulting and unfolding 3D seismic images: Geophysics, **78**, no. 4, O45–O56, doi: 10.1190/segam2012-1356.1.

Ma, Y., X. Ji, N. M. BenHassan, and Y. Luo, 2018, A deep learning method for automatic fault detection: 88th Annual International Meeting, SEG, Expanded Abstracts, 1941–1945, doi: 10.1190/segam2018-2984932.1.

Parks, D., 2010, Seismic image flattening as a linear inverse problem: Master's thesis, Colorado School of Mines.

Paszke, A., S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, and A. Desmaison, 2019, PyTorch: An imperative style, high-performance deep learning library: Advances in Neural Information Processing Systems, 8024–8035.

Pham, N., S. Fomel, and D. Dunlap, 2019, Automatic channel detection using deep learning: Interpretation, **7**, no. 3, SE43–SE50, doi: 10.1190/int-2018-0202.1.

Ronneberger, O., P. Fischer, and T. Brox, 2015, U-net: Convolutional networks for biomedical image segmentation: International Conference on Medical Image Computing and Computer-assisted Intervention, Springer, 234–241.

Schmidhuber, J., 2015, Deep learning in neural networks: An overview: Neural Networks, **61**, 85–117, doi: 10.1016/j.neunet.2014.09.003.

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.

Stark, T. J., 2003, Unwrapping instantaneous phase to generate a relative geologic time volume: 73rd Annual International Meeting, SEG, Expanded Abstracts, 1707–1710.

Stark, T. J., 2004, Relative geologic time (age) volumes-relating every seismic sample to a geologically reasonable horizon: The Leading Edge, **23**, 928–932, doi: 10.1190/1.1803505.

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.

Waldeland, A. U., and A. H. S. Solberg, 2017, Salt classification using deep learning: 79th Annual International Conference and Exhibition, EAGE, Extended Abstracts, Tu-B4-12.

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., and D. Hale, 2013, Extracting horizons and sequence boundaries from 3D seismic images: 83rd Annual International Meeting, SEG, Expanded Abstracts, 1440–1445.

Wu, X., and D. Hale, 2015, Horizon volumes with interpreted constraints: Geophysics, **80**, no. 2, IM21–IM33, doi: 10.1190/geo2014-0212.1.

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.

Wu, X., S. Luo, and D. Hale, 2016, Moving faults while unfaulting 3D seismic images: Geophysics, **81**, no. 2, IM25–IM33, doi: 10.1190/geo2015-0381.1.

Wu, X., and G. Zhong, 2012, Generating a relative geologic time volume by 3D graph-cut phase unwrapping method with horizon and unconformity constraints: Geophysics, **77**, no. 4, O21–O34, doi: 10.1190/geo2011-0351.1.

Zeng, H., 2016, Phase unwrapping for thin-bed seismic chronostratigraphy and facies analysis: 86th Annual International Meeting, SEG, Expanded Abstracts, 1763–1767, doi: 10.1190/segam2016-13862007.1.

Zeng, H., M. M. Backus, K. T. Barrow, and N. Tyler, 1998a, Stratal slicing, Part I: Realistic 3-D seismic model: Geophysics, **63**, 502–513, doi: 10.1190/1.1444351.

Zeng, H., S. C. Henry, and J. P. Riola, 1998b, Stratal slicing, Part II: Real 3-D seismic data: Geophysics, **63**, 514–522, doi: 10.1190/1.1444352.

Zeng, H., X. Zhu, R. Zhu, and Q. Zhang, 2012, Guidelines for seismic sedimentologic study in non-marine postrift basins: Petroleum Exploration and Development, **39**, 295–304, doi: 10.1016/S1876-3804(12)60045-7.

Zhao, T., 2018, Seismic facies classification using different deep convolutional neural networks: 88th Annual International Meeting, SEG, Expanded Abstracts, 2046–2050, doi: 10.1190/segam2018-2997085.1.