Pure Appl. Geophys.
© 2020 Springer Nature Switzerland AG
https://doi.org/10.1007/s00024-019-02412-z

Pure and Applied Geophysics

Machine Learning Using U-Net Convolutional Neural Networks for the Imaging of Sparse Seismic Data

Jiayuan Huang[1] and Robert L. Nowack[1]

Abstract—Machine learning using convolutional neural networks (CNNs) is investigated for the imaging of sparsely sampled seismic reflection data. A limitation of traditional imaging methods is that they often require seismic data with sufficient spatial sampling. Using CNNs for imaging, even if the spatial sampling of the data is sparse, good imaging results can still be obtained. Therefore, CNNs applied to seismic imaging have the potential of producing improved imaging results when spatial sampling of the data is sparse. The imaged model can then be used to generate more densely sampled data and in this way be used to interpolate either regularly or irregularly sampled data. Although there are many approaches for the interpolation of seismic data, here seismic imaging is performed directly with sparse seismic data once the CNN model has been trained. The CNN model is found to be relatively robust to small variations from the training dataset. For greater deviations, a larger training dataset would likely be required. If the CNN is trained with a sufficient amount of data, it has the potential of imaging more complex seismic profiles.

Keywords: Seismic imaging, Machine learning, Convolutional neural networks, Interpolation of seismic data.

1. *Introduction*

In this study, machine learning using convolutional neural networks (CNNs) is applied for the imaging of seismic reflection data. CNNs have had a number of successful applications for image analysis in different fields (LeCun et al. 2015), and are modeled after the structure of visual systems (Hubel and Wiesel 1962). Fukushima and Miyake (1982) proposed a neural network with a multi-layer structure as a predecessor of CNNs (Bhandare et al. 2016). An early example of CNNs was given by LeCun et al. (1998) and used to classify handwritten letters from patterns of digital pixels. It was however limited by the speed of computing at that time. With the development of machine learning algorithms and the availability of sufficient computational resources in recent years, CNNs have become increasingly popular and more accessible for image analysis. Efficient ways to train CNNs using GPU computing has also been developed (Chellapilla et al. 2006; Hinton et al. 2006; Bengio et al. 2007). In 2015, the Google Brain Team implemented an open-source math library called TensorFlow for machine learning applications, including the use of CNNs (Abadi et al. 2016). Keras, a high-level open source neural network library written in Python, was also released in 2015 (Chollet 2015). The Keras library provides a user-friendly set of tools for the building and training of neural network and has been fully integrated into the TensorFlow framework. Here we apply CNNs with Keras using GPU computing.

There have been many successful applications of CNNs in the science and engineering. For example, a common application in computer vision is for facial recognition. CNNs can extract features at different locations of the face as an input image and then output a number of feature maps (Bhandare et al. 2016). In the medical fields, researchers have been successful in detecting skin cancer using CNNs (Esteva et al. 2017). CNNs have also been applied to the game of Go (Maddison et al. 2014; Clark et al. 2015) and has beaten many of the Go masters in the world since 2015.

Given the successful applications of CNNs in other scientific fields, they are becoming increasingly popular for solving problems in the geosciences. CNNs provide new ways for high-performance

[1] Department of Earth, Atmospheric, and Planetary Sciences, Purdue University, West Lafayette, IN 47907, USA. E-mail: yyorvictor@gmail.com; nowack@purdue.edu

Published online: 15 January 2020

automatic interpretation, complex relationship modeling and data-driven information extraction of geoscience data (Bergen et al. 2019) Convolutional neural network have been designed for earthquake identification which is faster and more sensitive than traditional methods for detecting induced seismicity (Perol et al. 2018). Neural networks have also been trained to automatically identify seismic waveforms and identify first breaks of seismic data (Yuan et al. 2018).

CNNs are also showing potential for solving inverse problems in imaging such as for image denoising, reconstruction and interpolation (McCann et al. 2017). However, CNNs have only recently been used in seismic data processing. Support vector regression (SVR) has been applied to reconstruct sparsely sampled seismic data by Jia et al. (2017). Li et al. (2019) successfully built deep neural networks (DNNs) seismic data inversion for time-series. Wang et al. (2018) interpolated seismic data for missing traces by developing a CNN-based residual learning network.

In the energy industry, an important problem is the identification of salt bodies in the subsurface. The TGS Salt Identification Challenge was a Kaggle competition to identify the boundaries of salt deposits based on selected seismic images (TGS Salt Identification Challenge 2018). The traditional interpretation of seismic data is an important and time-consuming part of the exploration workflow, but it is greatly dependent on experienced interpreters. It also relies on high-performance computational resources (Waldeland et al. 2018; Araya-Polo et al. 2018). In addition, manual interpretation is highly time-consuming and subject to human bias (Di et al. 2018).

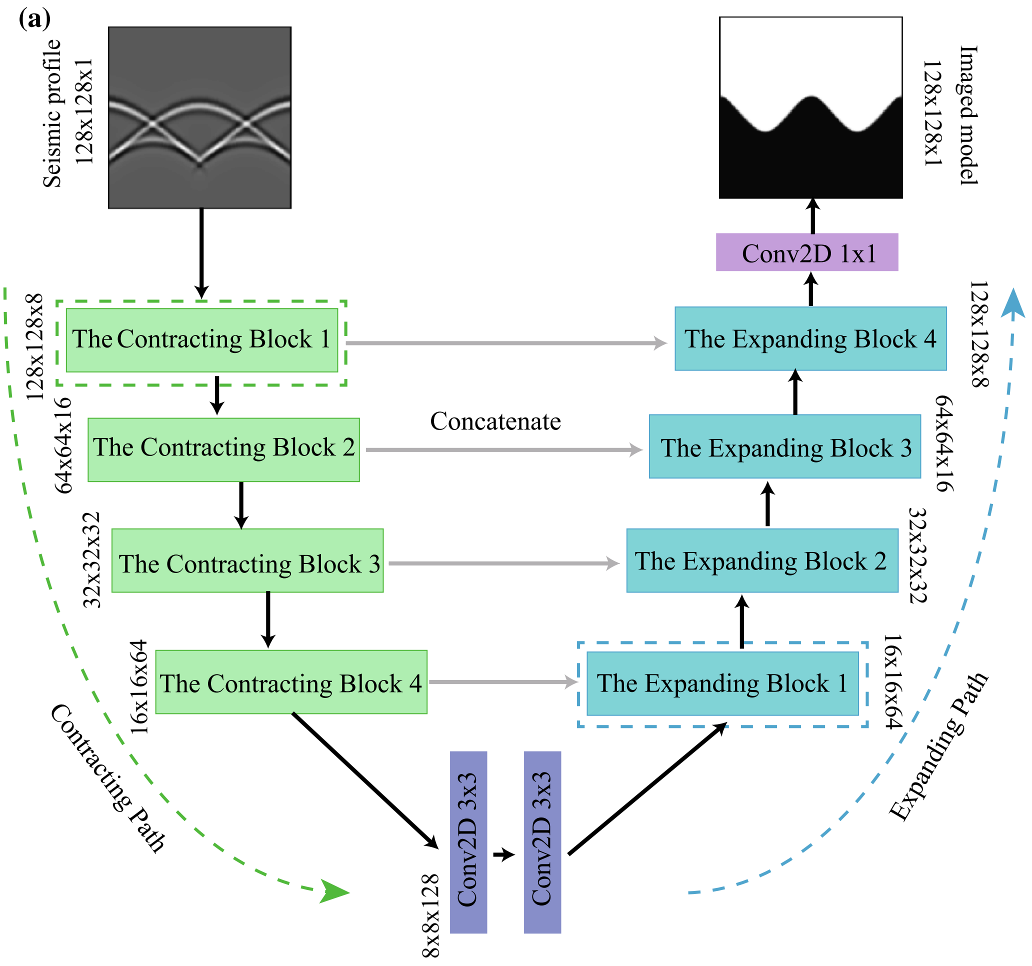
In this paper, we first generate synthetic zero-offset seismic reflection data from simple subsurface interface models. A CNN model is then built based on the U-net architecture to automatically image the seismic data. The input images are the seismic reflection data and the goal is to output the imaged subsurface models. The trained CNN is robust to small variations from the training dataset, but for larger deviations a larger training dataset would likely be required. If the CNN is trained with a sufficient amount of data, it should potentially be capable of imaging more complex data. Here we also use CNNs to image sparsely regularly and irregularly sampled seismic data. Although there are many approaches for the interpolation of seismic data, here seismic imaging is performed directly with sparse seismic data using a CNN model.

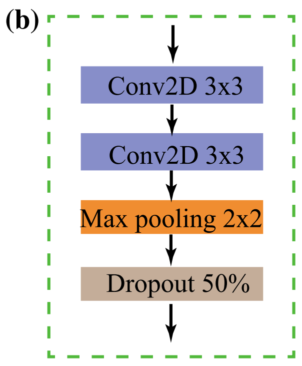
2. *The U-net Architecture*

The CNN architecture used in this study is the U-net which is a fully convolutional network developed earlier for biomedical image segmentation problems (Ronneberger et al. 2015a, b). This convolutional network can work with fewer training images but still produce accurate image segmentations (Ronneberger et al. 2015a, b). The U-net is an encoder-decoder neural network architecture consisting mainly of two paths, the contracting path (encoder) and expanding path (decoder). Each block in the two paths contains different sub-layers (Fig. 1). The contracting path consists of repeated application of convolutions, activation functions, max pooling and dropout operations which capture important features from the input images. The expanding path constructs the high-resolution feature maps by combining low-resolution feature maps and spatial information from the contracting path and includes several repeated layers of transposed convolution, concatenation, dropout and convolution operations.

2.1. *The Contracting Path*

The contracting path is typical of convolutional neural networks. Here we use four blocks in the contracting path where each block contains four layers (Fig. 1). The first two layers in each block are convolutional layers, where the kernels (filters) are 3×3. The stride (the steps to skip in the convolution operation) is 1. Each convolutional layer includes an activation function called a rectified linear unit (ReLU). This layer serves the purpose of extracting the features from the input images. For all the 3×3 convolutional layers in this architecture, each edge of the input images is zero-padded by one pixel so that the output feature map size is the same as the input size. The weights and the bias are two important sets





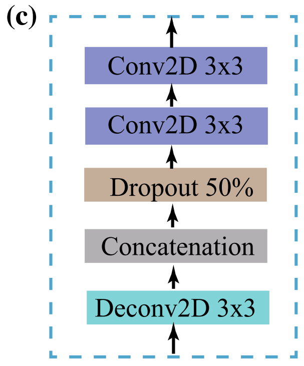


Figure 1
a The architecture of the U-Net Convolutional Neural network in this study. The architecture contains two paths, the contracting path and the expanding path. The contracting path has four blocks each with several layers. The size of the feature maps halves after each block, and the number of feature maps doubles. There are then two convolution layers each with 3 × 3 kernels (filters) between the contracting path and the expanding path. The expanding path also has four blocks. The size of the feature maps now double and the number of feature maps halves after each block. There is then a final 1 × 1 convolutional layer that maps the feature maps from 8 to 1. **b** Each contracting block has two convolutional layers with 3 × 3 kernels, a 2 × 2 max pooling layer and a dropout layer with 50% dropout rate. **c** An expanding block contains a transposed convolutional (deconvolutional) layer with 3 × 3 kernels, a concatenation layer that gets the spatial information from contracting blocks, a dropout layer with a 50% dropout rate and two convolutional layers with 3 × 3 kernels

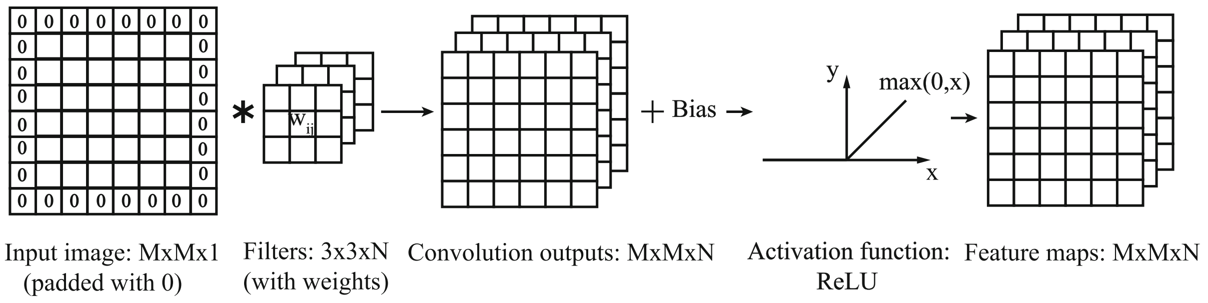

0
+ Bias
y
max(0,x)
x
w_{ij}
Input image: MxMx1 (padded with 0)
Filters: 3x3xN (with weights)
Convolution outputs: MxMxN
Activation function: ReLU
Feature maps: MxMxN

Figure 2
An example showing how a single convolutional layer with 3 × 3 kernels (filters) is configured. The MxMx1 size input image is zero padded by one pixel on each edge and is then convolved with N 3 × 3 kernels. These convolution operations result in N M × M convolution outputs. After the addition of a bias parameter, convolution outputs are input into an activation function (ReLU) resulting in N MxM feature maps for this layer

of learnable parameters from the convolutional layers. The weights are the values of the kernels. The bias is a parameter added after a convolution output before passing it to the nonlinear activation function (a ReLU). Figure 2 shows how each convolutional layer is configured. The third layer of each block is a 2 × 2 max pooling layer which halves the matrix size in order to reduce the number of parameters in the layer. For example, the original figure size is 128 × 128 and after the first max pooling layer is reduced to 64 × 64. The fourth layer of each block is a dropout layer. It is designed to randomly drop out nodes during the training process and serves the purpose of reducing overfitting for deep neural networks. The dropout rate that we use here is 0.5 which means it will randomly drop out 50% of the nodes for this step.

After the first contracting block extraction, the parameters are sent to the next contracting block and the process is repeated for each block. At the same time, the number of kernels after each block doubles so that the architecture can learn the complex image features effectively. After the contracting path, there are two padded convolutional layers with 3 × 3 kernels following the fourth contracting block. Each is followed by an activation function-ReLU and these mediate between the contracting path and expanding path (Fig. 1).

2.2. *The Expanding Path*

The expanding path has four blocks, and each expanding block has five layers (Fig. 1). For each block in the expanding block, the first layer is a transposed convolutional layer (a deconvolution layer) and is designed to up-sample the feature maps from low resolution to higher resolution. The kernel (filter) sizes are 3 × 3, and the stride is 2. The transposed convolutional layers with 3 × 3 kernels are zero-padded by one pixel. The input size is 8 × 8 in the first transposed convolutional layer and the output size is 16 × 16. The second layer of each expanding block is a concatenate layer which concatenates the feature maps from the contracting path to the expanding path at the same level. This action can get localization information from the contracting path and help to reconstruct high resolution feature maps in the expanding path. For example, the first block in the expanding path concatenates with the fourth block in the contracting path. The third layer of each expanding block is a dropout layer and the dropout rate is set to 0.5. The fourth and fifth layers of each expanding block are padded convolutional layers. The kernels are 3 × 3 and the stride is 1. The activation function for each convolutional layer is a ReLU. The number of kernels after each expanding block now halves in contrast to the contracting blocks. After the first block expansion, the parameters are sent to the next block and the process is repeated. After four expansion blocks, the figure size is 128 × 128 which is the same as the original image size (Fig. 1).

The last layer after the expanding path is the output layer which is a 1 × 1 padded convolutional layer with a 1 × 1 kernel and a stride of 1 and this maps the feature maps from 8 to 1. The activation

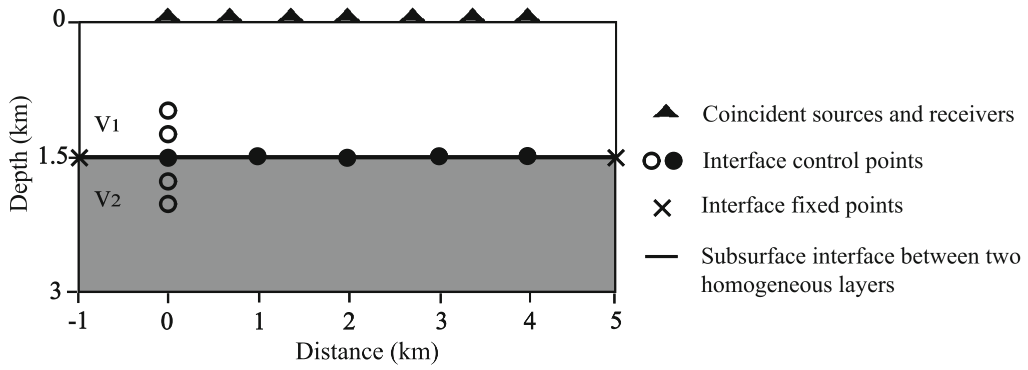


Figure 3
A diagram for the building of the subsurface interface models. The triangles show the zero-offset location of the sources and receivers at the surface. For each subsurface interface model, there are five interface control points and two fixed points at the beginning and the end distance. Each control point can move up and down with depths of 1.1, 1.3, 1.5, 1.7, 1.9 km and each fixed point is at a depth of 1.5 km. Therefore, there are 5^5 or 3125 interface models and 3125 corresponding computed seismic reflection profiles

function here is a sigmoid (Han and Moraga 1995), and the output of the sigmoid function is between 0 and 1 and can be interpreted as a probability.

The neural network then computes the loss function which measures the average difference between the predicted values and the true subsurface interface models. The lower the value of the loss function, the more correct the prediction is. An optimizer (optimization algorithm) is then applied to estimate the model parameters (kernel weights and biases) that minimizes the loss function. The parameters of the neural network are then iteratively updated using a backpropagation algorithm (Rumelhart et al. 1986). The iterations are repeated until the loss function reaches a global minimum.

3. *Synthetic Data*

We first build subsurface models and compute synthetic zero-offset synthetic seismic reflection data. The synthetic seismic reflection data are generated using the Gaussian beam modeling code Triseis in the Seismic Un*x package (Stockwell 1999).

3.1. *Subsurface Interface Models and Reflection Seismic Profiles*

For the simple interface models considered here, we set the horizontal distance of the subsurface interface model from − 1 to 5 km, and from 0 to 4 km to compute the corresponding reflection profiles. The depth range of the subsurface interface models is from 0 to 3 km. In order to create models with different interface shapes, there are 5 interface control points with horizontal distance intervals from 0 to 4 km and 2 fixed points at − 1 and 5 km. Each control point can move up and down at depths of 1.1, 1.3, 1.5, 1.7, 1.9 km and the two fixed points at the beginning and the end are at the depth of 1.5 km. Therefore, there are 5^5 or 3125 subsurface interface models and 3125 corresponding synthetic seismic reflection profiles (Fig. 3). The upper layer "sloth" (inverse of the velocity squared) is 0.25 s^2/km^2 (or a velocity equal to 2 km/s) and the lower layer sloth is 0.1 s^2/km^2 (or a velocity equal to 3.16 km/s).

When computing the zero-offset seismic reflection profiles, the horizontal range is from 0 to 4 km along the surface, including 101 traces with a horizontal distance interval of 0.04 km. Each trace has 101 time samples with a time sampling interval of 0.03 s. For the Gaussian beam modeling, the first ray takeoff angle is set at − 55° and the last ray takeoff angle is 55°. To avoid possible aliasing, the peak frequency of the Ricker wavelet is set to 4 Hz. For the 3125 subsurface interface models, corresponding seismic reflection profiles are generated in this way.

3.2. *Dataset Preprocessing*

The subsurface interface model used in the Gaussian beam modeling code is converted to a gridded model using the code tri2uni in Seismic Un*x and the gridded subsurface interface models are then stored as binary files. The seismic data generated from Seismic Un*x are also stored as binary files. The models and computed seismic reflection data are then randomly divided into 1875 (60%) for the training dataset, 625 (20%) for the validation dataset and 625 (20%) for the test dataset. The training dataset is initially used to estimate the parameters of the neural network model. The validation dataset is then used to evaluate the performance of the neural network model fit from the training dataset and can be used as an indicator if the neural network model is being overfit by the training dataset. The test dataset is then used for evaluating the final neural network model, where these data have not been used in the training process.

Since data normalization can accelerate neural network training and avoid local minima of the loss function (Ioffe and Szegedy 2015), the sloth values of the subsurface models are normalized to 0 (for 0.1 s^2/km^2) and 1 (for 0.25 s^2/km^2) and the amplitude data which contain positive and negative values are scaled to a range from -1 to 1 by dividing by the maximum absolute value of the seismogram datasets. For the imaging here, the velocities are assumed to be known from earlier processing steps of the data, and here we are only imaging the structure aspects of the model. This is similar to classical seismic migration imaging where the velocity model is given prior to imaging for the structure.

4. *Model Training*

4.1. *Loss Function and Metrics*

The performance of CNNs in this study is measured and correspondingly optimized by using a binary cross entropy loss function. For each pixel in the predicted model, the values are interpreted as a probability from 0 to 1. For this example, 1 represents the upper layer and 0 represents the lower layer. The binary cross entropy loss function is given by:

$$L = -[y\log(p) + (1 - y)\log(1 - p)]$$

where y is the true value (true distribution) and p is the predicted distribution. In our case, y is 1 and p is the predicted probability of upper layer. $(1 - y)$ is equal to 0 which is the true value of lower layer and $(1 - p)$ is its predicted probability. The binary cross entropy loss function as used in the image segmentation is the average evaluation of the class prediction for each pixel in the predicted models and is used to optimize the neural network.

Metric functions are used to evaluate the performance of the neural network. Although these can also be used as loss functions, the evaluation results of the metric functions are not used to train the neural network. Here we choose binary accuracy and the dice coefficient as metric functions (Dice 1945; Sørensen 1948).

The binary accuracy gives the percentage of correctly classified pixels in the images and is given by:

$$\text{Binary accuracy} = \frac{(\mathrm{TP} + \mathrm{TN})}{(\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN})}$$

where TP (true positive) is the number of pixels that successfully predict the upper layer, TN (true negative) is the number of pixels that correctly predict the lower layer. FP (false positive) is the number of pixels that wrongly recognize the upper layer as lower layer and FN (false negative) is the number of pixels which fail to predict the lower layer.

The dice coefficient, also known as the dice score or F1-score, measures the overlap of the true binary subsurface interface models and the predicted models. The dice coefficient is given by:

$$\text{Dice coefficient} = \frac{2\mathrm{TP}}{2\mathrm{TP} + \mathrm{FP} + \mathrm{FN}}$$

where the TP, FP, FN are the same as for the binary accuracy.

4.2. *Starting Kernels (Filters)*

Kernels work as feature extractors in the neural network. Their weights are initially random and then updated after each iteration of the training process. As mentioned previously, the number of kernels in the first contracting block need to be set first and the number of kernels after each block then doubles. Here we choose 2, 4, 8, 16, 32 for starting numbers of kernels. We then use an Adam optimizer to train the

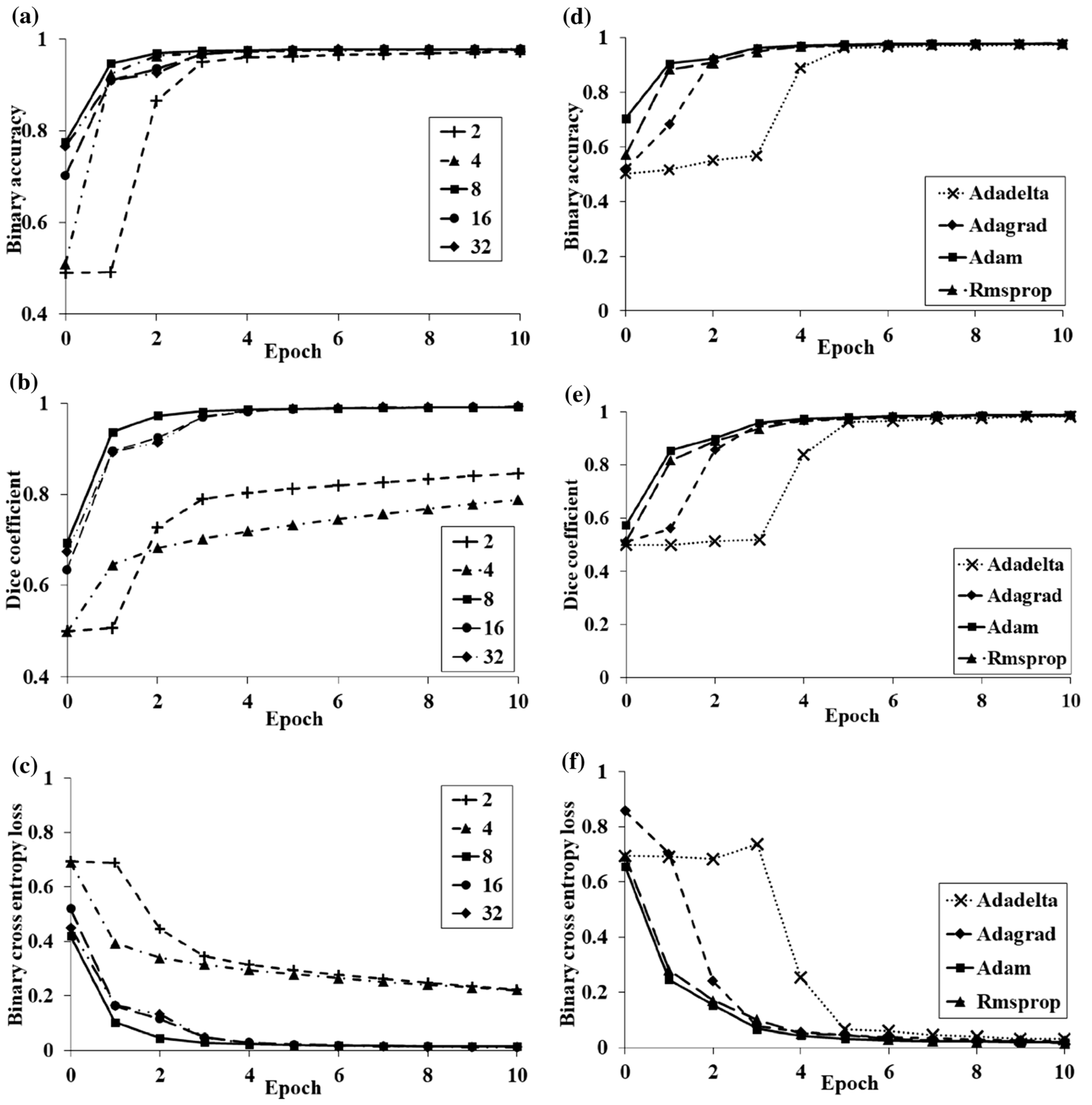


Figure 4
The experiment scores of the CNN models for 10 epochs. Subplots **a**, **b**, **c** are binary accuracy, dice coefficient and binary cross entropy loss function of the CNN models with a different number of starting kernels (filters) optimized using an Adam optimizer. The numbers of starting kernels are set in the first contracting block and double after each contracting block. They then gradually halve after each expanding block. Subplots **b**, **e**, **f** are the binary accuracy, dice coefficient and binary cross entropy loss function of the CNN models with eight starting kernels optimized using different optimizers

model (Kingma and Ba 2014). Figure 4a–c shows examples of the binary accuracy, dice coefficient and binary cross entropy loss function for different numbers of starting kernels. The results show that the convergence of binary accuracy, dice coefficient and binary cross entropy loss function are the fastest when the number of starting kernels is 8.

4.3. Optimizers

The optimizers are designed to minimize the loss function in the training process. We compare several optimization algorithms in the model training: Adadelta (Zeiler 2012), Adagrad (Duchi et al. 2011), Adam (Kingma and Ba 2014) and Rmsprop (Hinton et al. 2012). We use the default parameter values for each optimization algorithm in the Keras deep learning library. Here we set eight starting kernels (filters) for the first block. Figure 4d–f compare the binary accuracy, dice coefficient and binary cross entropy loss functions where the starting kernels are set to 8 for the first block. The results show that the Adam algorithm has the fastest convergence for the binary accuracy, dice coefficient and binary cross entropy loss function and is used for our study.

5. Results

In the model training process, we set eight starting kernels (filters) in the first block of the contracting path and choose an Adam optimizer to minimize the loss function since the accuracy of this combination converges faster than other combinations according to our tests. The binary accuracy, dice coefficient and binary cross entropy loss function for our model are shown in the Fig. 5. The triangles are for the training dataset and the squares are for the validation dataset.

The neural network model is implemented in Keras and is trained using a NVIDA RTX 2070 graphics card on a single workstation. The batch size (the number of training samples used to train the neural network in a single batch) is set to 32 and the epoch (the number of times that the entire training dataset is used to train the neural network) is set to 40. Since the neural network cannot pass through the entire training dataset all at once, the training dataset is randomly divided into several batches by the defined batch size, and all the batches pass through the neural network for one epoch. The training dataset is then randomly separated again, and the process is repeated in the next epoch. In Fig. 5, we see that the model only needs five epochs to get a high accuracy and after that the accuracy increases slowly. The training process stops at 35 epochs after applying an early stopping regularization with a patience (the number of epochs before stopping the training process if the model hasn't improved) of 10. This means that the binary accuracy and the binary cross entropy loss function do not improve after 25 epochs. The total training process takes approximately 2 min for a NVIDA RTX 2070 GPU on single workstation. We performed similar computations on

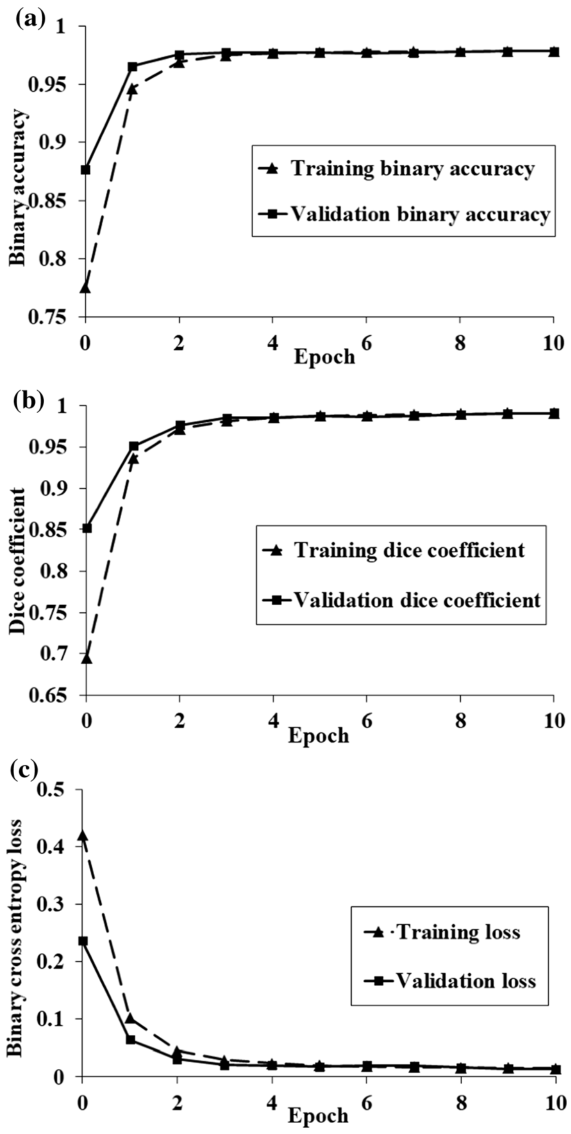


Figure 5
The binary accuracy, dice coefficient and binary cross entropy loss function of CNN models with 8 starting kernels (filters) optimized by an Adam optimizer

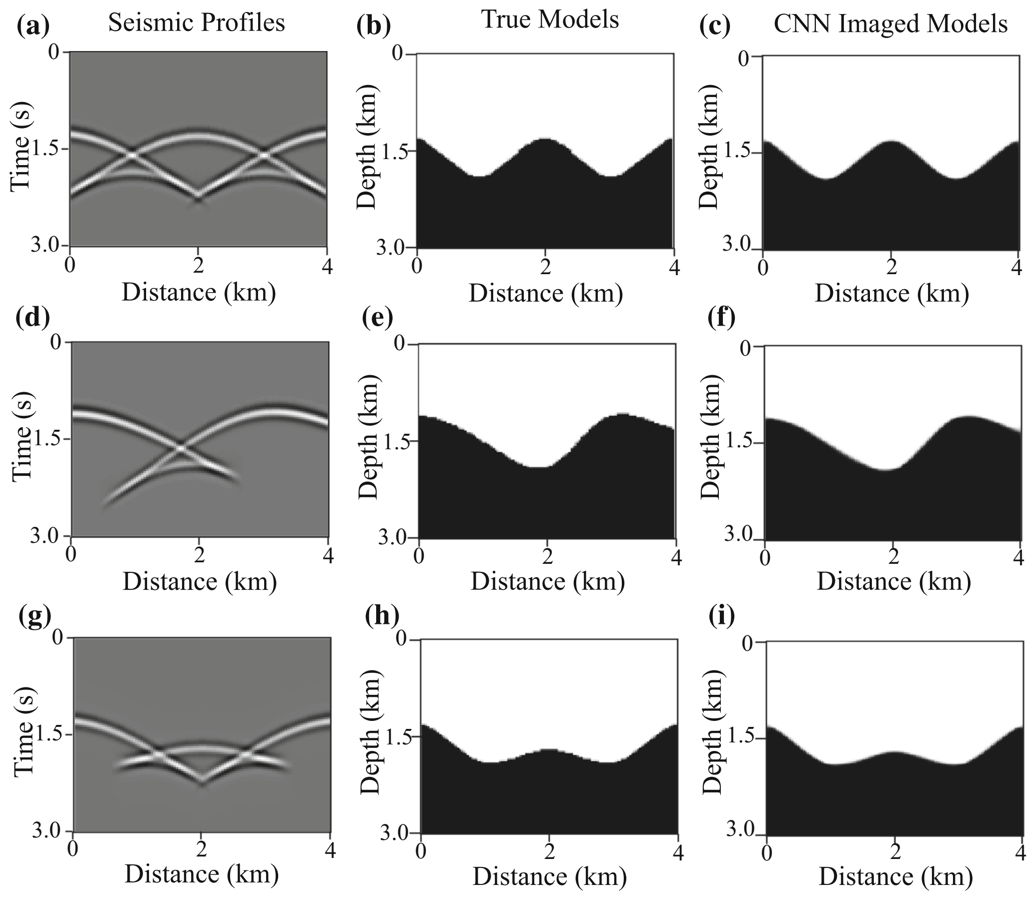


Figure 6
Imaged models from the trained neural network. Subplots **a**, **d**, **g** give seismic profiles with 101 seismic traces. Subplots **b**, **e**, **h** are true subsurface interface models and subplots **c**, **f**, **i** are the corresponding imaged models from the CNN imaging given the input seismic profiles

the larger Purdue Gilbreth GPU Cluster with NVIDA Tesla V100 GPUs. Although the results were faster, they were limited by the user allocation limits on the larger machine.

Figure 6 shows several examples for the imaged models using the trained neural network. For the given seismic profiles shown on the left, the true models are shown in the middle and the imaged models are shown on the right. As can be seen, the trained neural network does an excellent job of estimating the imaged models from the seismic profiles.

Figure 7a shows a seismic profile where the upper layer velocity of the model is 10% higher than the models used to generate the seismic profiles in the training dataset. Figure 7b shows the true subsurface interface model with the correct velocity model and Fig. 7c shows the CNN imaged model resulting in a slightly elevated interface from the true model.

Figure 7d shows a seismic profile with 10% Gaussian noise added which is not included in the training dataset. Figure 7e shows the true subsurface interface model and Fig. 7f shows the CNN imaged model from the noisy seismic data in Fig. 7d. In this case the model is well imaged by the CNN.

Figure 7g shows a seismic profile from a model not included in the training dataset by adding a depth of 0.7 km to the interface depths. The true model is shown in Fig. 7h) and the CNN imaged model is shown in Fig. 7i. In this case the average depth of the interface is correct, but the details of the imaged interface have some discrepancies with the true model in Fig. 7b. Nevertheless, the overall shape and

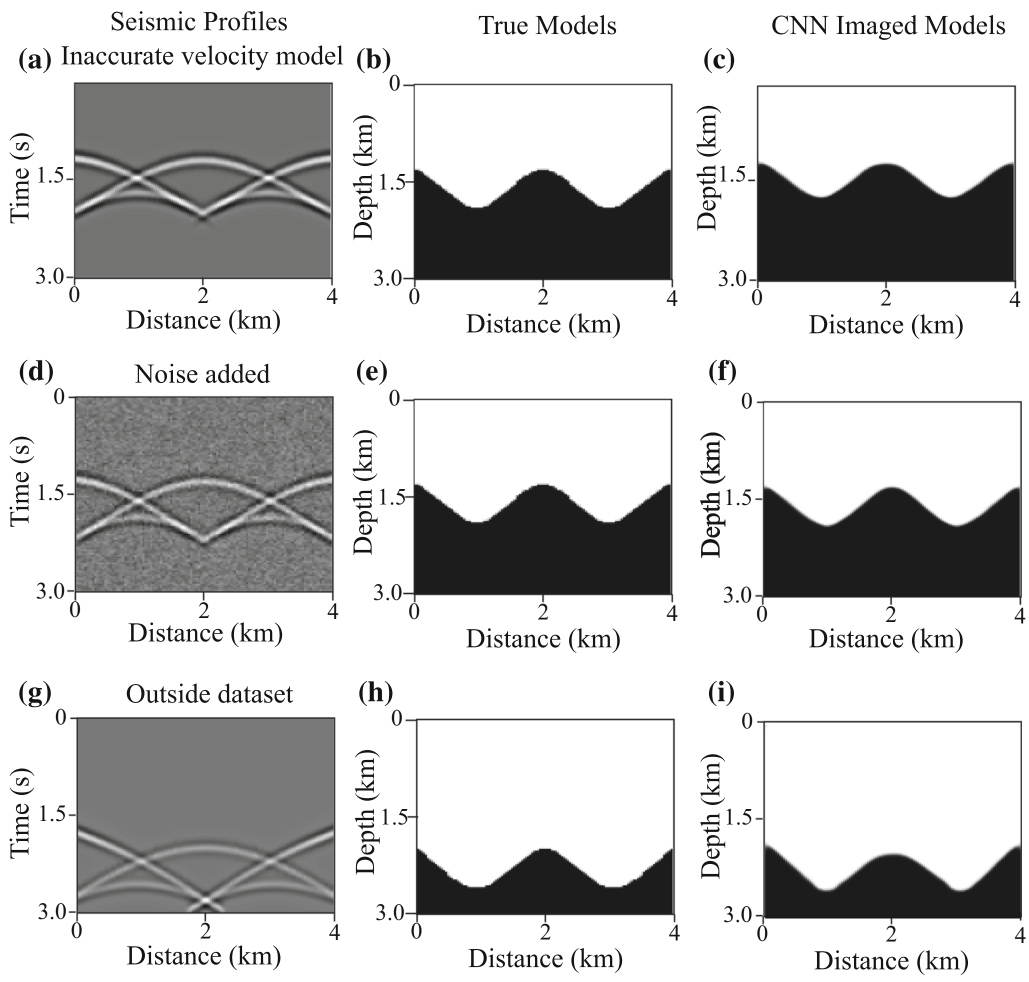


Figure 7

Imaged models from the trained neural network using seismic profiles with different effects not included in the training dataset. **a** The seismic profile when velocity model is inaccurately estimated, the upper layer velocity of the model is 10% higher than model used for the training dataset. **d** The seismic profile with 10% Gaussian noise to the data. **g** The seismic profile of a model that is outside the training dataset with an interface 0.7 km deeper than the training dataset. For each case **b**, **e**, **h** are true subsurface interface models. For each case **c**, **f**, **i** are the corresponding imaged models from the CNN given the input seismic profiles in **a**, **d** and **g**

depth of the CNN imaged model is similar to the true model.

From these examples it can been seen that the CNN model is relatively robust for small variations from the training dataset. However, larger deviations from the training dataset would likely require a larger training dataset.

Traditional migration methods are sensitive to the spatial sampling of the data. In order to get the best resolution of the subsurface images, the seismic data needs to be sufficiently sampled prior to migration imaging. Here we use the migration code Sustolt in Seismic Un*x and show several examples of the migration of seismic profiles with a different number of traces (Fig. 8). For these examples, when using 51 and 101 traces, accurate migration images can be obtained. However, for fewer traces, aliasing effects substantially degrade the images.

In order to see how the convolutional neural networks (CNNs) perform for a different number of traces, we input seismic profiles with 5, 9, 17, 26 and 51 traces with larger station spacings to the trained CNN model based on the seismic profiles with 101 traces. Figure 9 shows the seismic profiles with a

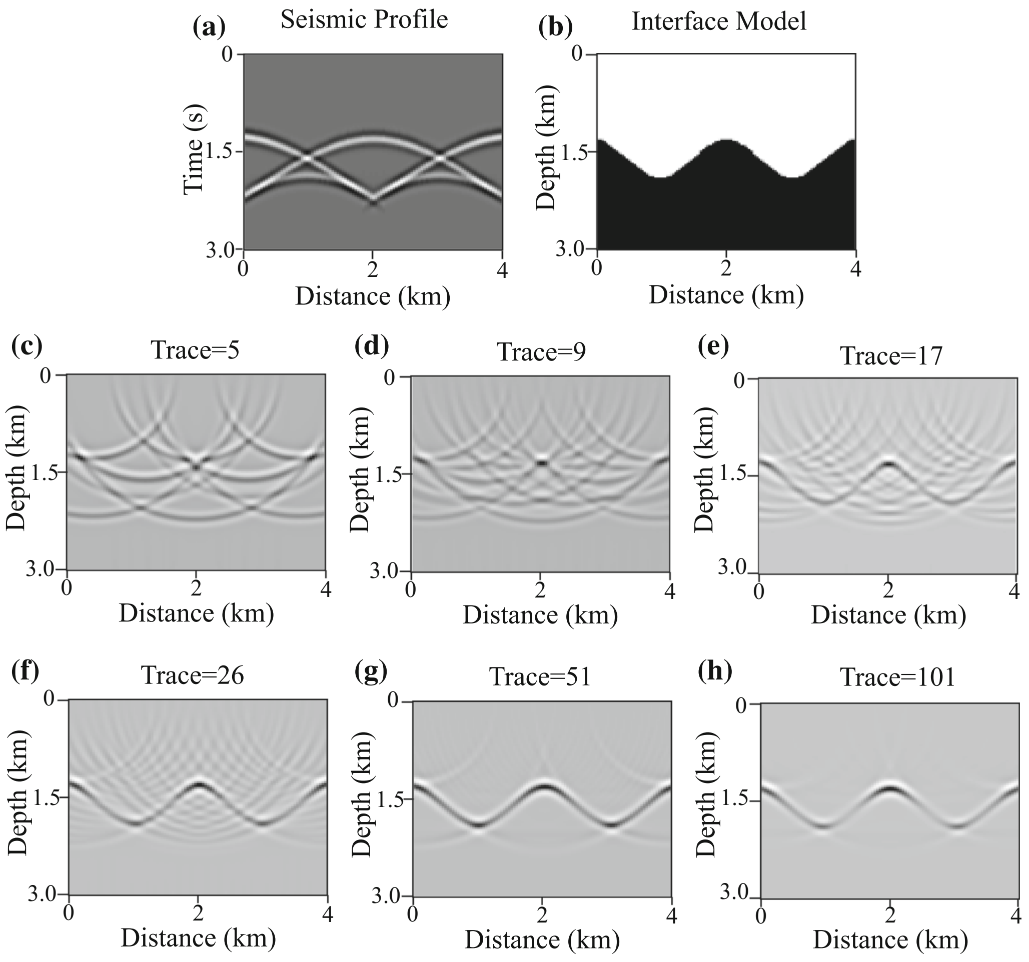


Figure 8
Migration imaging results for seismic profiles with a different number of traces. Subplot **a** is the seismic profile with 101 traces. Subplot **b** is the corresponding subsurface interface model. Subplots **c**, **d**, **e**, **f**, **g**, **h** are the migration imaging results where the numbers of seismic traces are 5, 9, 17, 26, 51 and 101, respectively

different number of seismic traces, 9, 17 and 26. The corresponding imaged interfaces comparing with true interfaces are given in the Fig. 10. The comparisons show that when there are just nine traces, the CNN results are fair but are still better than the results from seismic migration. When the number of traces is 17, the CNN results are improved but still have slight differences with the true interfaces. When the number of traces is 26, the CNN results are now very good and almost the same as the true models. The evaluation of the results of how well the CNN model trained using the 101 traces dataset performs on datasets with a different number of traces is given in Table 1.

The imaged models from the trained CNN model when the number of regularly or irregularly sampled seismic traces is 26 are then used to create new more densely sampled seismic reflection profiles in Fig. 11. Figure 11a, d, g show sparse seismic profiles where the number of regularly sampled seismic traces is 26. Figure 11j shows sparse seismic data when the traces are irregularly sampled. Figure 11b, e, h, k are the interpolated seismic data from the CNN imaged models and Fig. 11c, f, i, l are the true seismic data when the number of seismic traces is 101. As can be seen, the CNN provides interpolation capabilities which can prove useful in cases without a sufficient spatial sampling needed for traditional migration

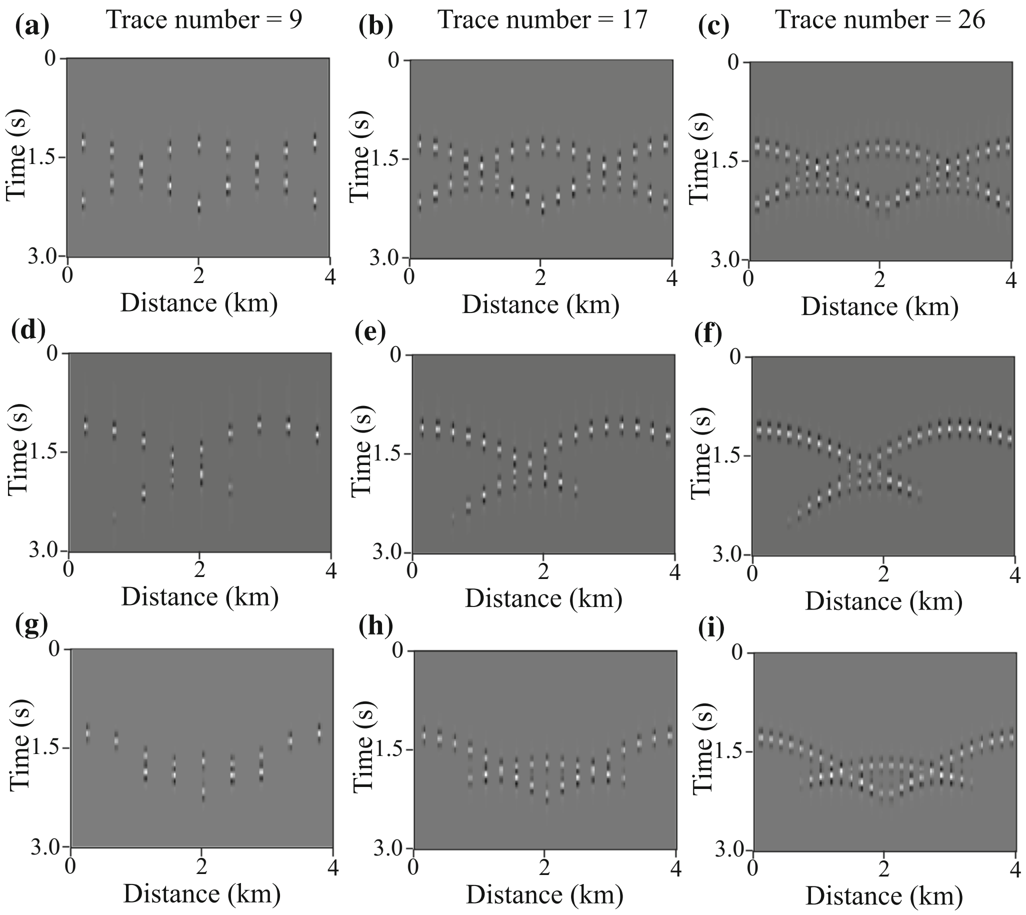


Figure 9
Several seismic profiles with a different number of seismic traces. Subplots **a**, **d**, **g** are seismic profiles when the number of seismic traces is 9. Subplots **b**, **e**, **h** are seismic profiles when the number of seismic traces is 17. Subplots **c**, **f**, **i** are seismic profiles when the number of seismic traces is 26

imaging methods. Although there are a number of approaches for the interpolation of seismic data (for a survey see Chen et al. 2019), for the approach followed here the CNN model is used to directly image the sparse seismic data. Although the training process requires some computational and data resources, once the CNN is trained the imaging and interpolation processes are very fast.

6. *Discussion and Conclusions*

We have shown that machine learning with a CNN U-net architecture works well for the seismic imaging test cases given here. A limitation of traditional migration imaging methods is that they often require sufficient spatial sampling of the data. Although there are a number of approaches for the interpolation of seismic data (Chen et al. 2019), a novelty of the approach used here is that imaging can be performed directly on sparse regularly or irregularly sampled data once the CNN has been trained. Even when the spatial sampling of the seismic profiles is large, the CNN can still obtain good imaging results. Convolutional neural networks therefore have the potential of providing improved imaging results compared with traditional migration imaging methods when the spatial sampling is sparse. The CNN model is robust to the small variations from the training dataset. However, for larger deviations a

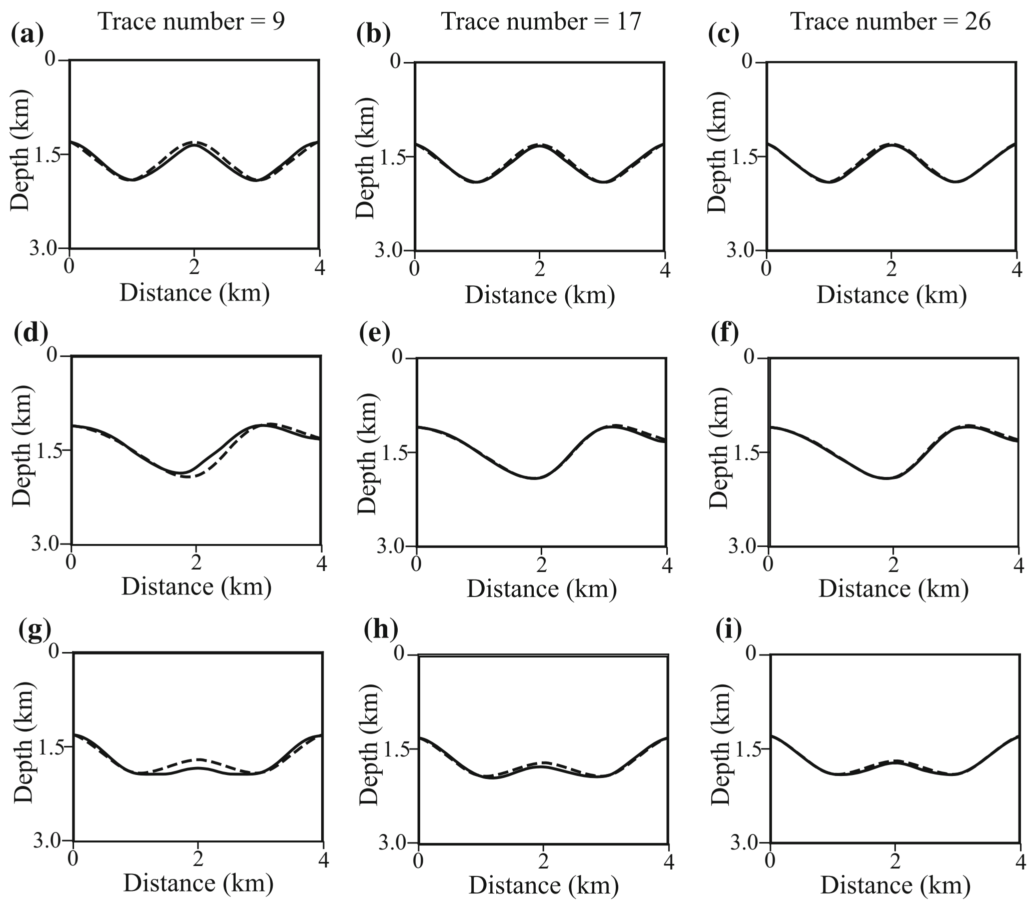


Figure 10
The CNN imaged interfaces when the number of seismic traces is 9, 17 and 26, respectively, comparing with the true interfaces. The solid lines are CNN imaged interfaces and the dashed lines are the true interfaces. Subplots **a**, **d**, **g** are comparisons of interfaces when the number of seismic traces is 9. Subplots **b**, **e**, **h** are comparisons of interfaces when the number of seismic traces is 17. Subplots **c**, **f**, **i** are comparisons of interfaces when the number of seismic traces is 26

Table 1

Performance evaluation results of the model trained using a 101 trace dataset and applied to datasets with a different number of traces

Traces	Score		
	Binary cross entropy	Binary accuracy	Dice coefficient
5	0.756	0.905	0.911
9	0.257	0.953	0.965
17	0.059	0.973	0.986
26	0.027	0.977	0.990
51	0.011	0.980	0.993
101 (from the test dataset)	0.009	0.981	0.994

For the binary cross entropy lower is better and for the binary accuracy and dice coefficient higher is better

larger training dataset would likely be required. Since CNN is a kind of supervised learning, the parameters of CNN model are trained based on the most important features from training dataset. If the CNN is trained with a sufficient amount of data, the CNN has the potential of imaging more complex seismic profiles.

There are several possible approaches in applying CNN to real seismic data. If enough data for a given region are available that have been previously imaged, then these data can be used for training and validation to obtain the parameters of the CNN model. The trained CNN can then be used for the imaging of new observed data. However, this assumes that the training data are obtained from

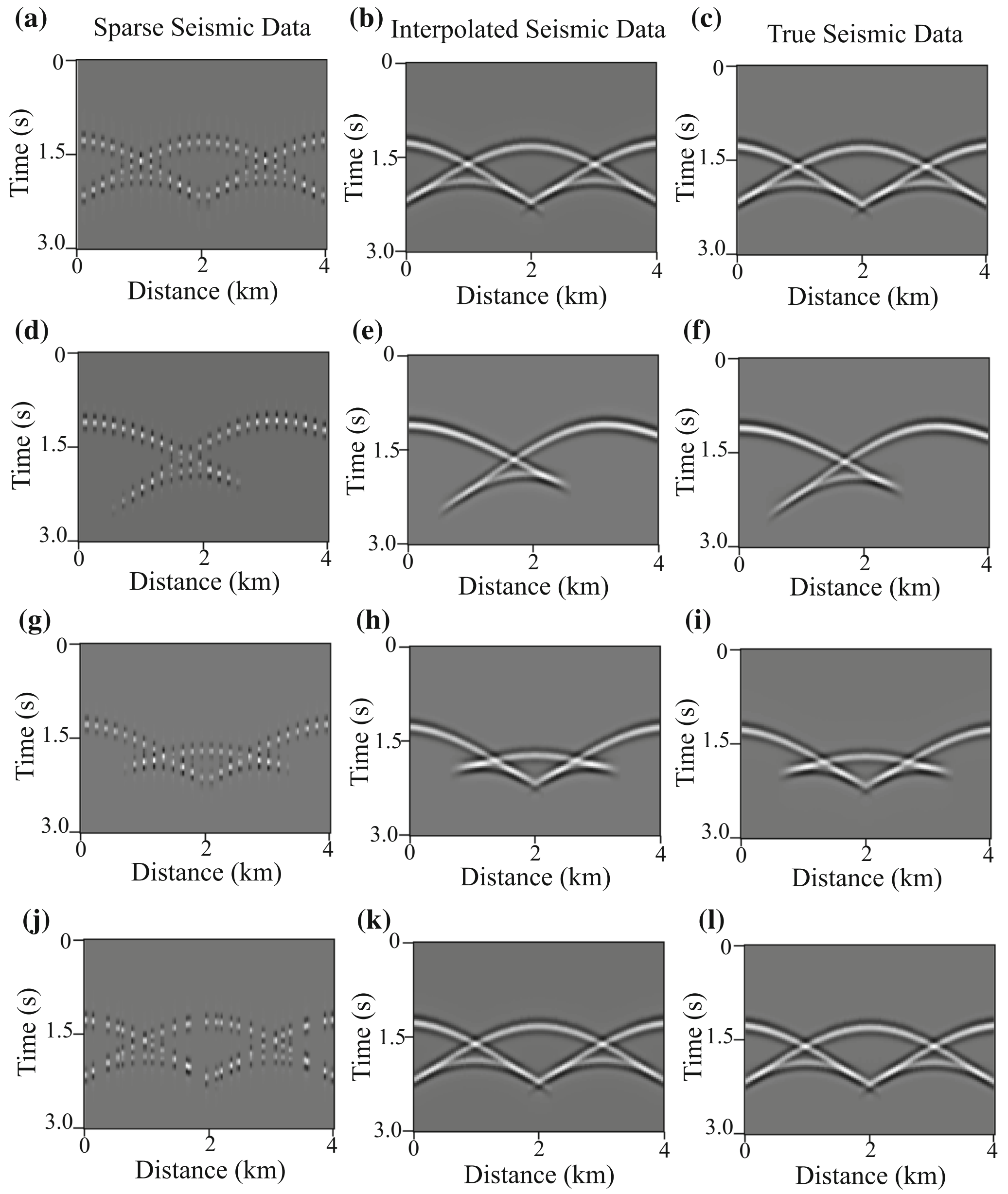


Figure 11
Interpolated seismic data obtained using the CNN imaged results from the trained CNN model when the number of regular and irregular seismic traces is 26. **a**, **d**, **g** Sparse seismic data when the number of regularly sampled seismic traces is 26. **j** The sparse seismic data when the number of irregularly sampled seismic traces is 26. **b**, **e**, **h**, **k** The interpolated seismic data from the CNN imaged models, and **c**, **f**, **i**, **l** True seismic data when the number of traces is 101

regions that are sufficiently similar to that of the new observed data.

A second alternative is to use synthetic data for the training and validation of the CNN. If fast forward modeling codes are used for the training and validation, as is done here using asymptotic beam codes, then this would result in an efficient approach for generating the parameters of the CNN for the imaging of the observed data. However, the synthetic data needs to be generated for models that are sufficiently similar to the real subsurface structures that result in the observed data.

For either alternative, if the CNN model can still perform well for real seismic data, which are much more complex, then once the training and validation have been performed, then very fast seismic imaging can be performed. It can also potentially reduce the amount of data required for seismic imaging and at the same time be used to interpolate sparse data to a finer grid.

Acknowledgements

The authors would like to thank the Editor and the reviewers for their constructive comments on the manuscript. The authors also thank Abdullah Khan Zehady for providing advice on CNN coding and model testing. This study was partially supported by NSF/EAR 1839322.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C. et al. (2016). TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. arXiv:1603.04467 [cs.DC]. Accessed 25 June 2019.

Araya-Polo, M., Jennings, J., Adler, A., & Dahlke, T. (2018). Deep-learning tomography. *The Leading. Edge, 37*(1), 58–66.

Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H. (2007). Greedy layer-wise training of deep networks. In J.D. Cowan, G. Tesauro, J. Alspector (Eds.), *Advances in Neural Information Processing Systems*, vol.19 (NIPS), (pp. 153–160). MIT Press.

Bergen, K.J., Johnson, P.A., Hoop, M.V. de, Beroza, G.C. (2019). Machine learning for data-driven discovery in solid Earth geoscience. *Science*, *363*(6433), eaau0323.

Bhandare, A., Bhide, M., Gokhale, P., & Chandavarkar, R. (2016). Applications of convolutional neural networks. *International Journal of Computer Science and Information Technologies, 7*(5), 2206–2215.

Chellapilla, K., Puri, S., Simard, P. (2006). High performance convolutional neural networks for document processing. In *Tenth International Workshop on Frontiers in Handwriting Recognition*, Université de Rennes 1, La Baule, France, October 23–26 2006 **(inria-00112631)**.

Chen, Y., Chen, X., Wang, Y., & Zu, S. (2019). The interpolation of sparse geophysical data. *Surveys in Geophysics, 40*(1), 73–105.

Chollet, F. (2015). Keras. https://github.com/fchollet/keras. Accessed 1 Jun 2019.

Clark, C., Storkey, A. (2015). Training deep convolutional neural networks to play go. In *32nd International Conference on Machine Learning*, vol. 37 (pp. 1766–1774). Lille **(06–11 Jul 2015)**.

Di, H., Wang, Z., AlRegib, G. (2018). Deep convolutional neural networks for seismic salt-body delineation. *AAPG 2018 Annual Convention and Exhibition*. Salt Lake City **(20–23 May 2018)**.

Dice, L. R. (1945). Measures of the amount of ecologic association between species. *Ecology, 26*(3), 297–302.

Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research, 12*, 2121–2159.

Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., et al. (2017). Dermatologist-level classification of skin cancer with deep neural networks. *Nature, 542*, 115–118.

Fukushima, K., & Miyake, S. (1982). Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Visual Pattern Recognition. In S. Amari & M. A. Arbib (Eds.), *Competition and Cooperation in Neural Nets. Lecture Notes in Biomathematics* (Vol. 45, pp. 267–285). Berlin: Springer.

Han, J., & Moraga, C. (1995). The influence of the sigmoid function parameters on the speed of backpropagation learning. In J. Mira & F. Sandoval (Eds.), *International Workshop on Artificial Neural Networks. Lecture Notes in Computer Science* (Vol. 930, pp. 195–201). Berlin: Springer.

Hinton, G. E., Osindero, S., & Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation, 18*(7), 1527–1554.

Hinton, G., Srivastava, N., Swersky, K. (2012). Neural networks for machine learning Lecture 6a Overview of mini-batch gradient descent. https://www.cs.toronto.edu/~hinton/coursera/lecture6/lec6.pdf. Accessed 3 May 2019.

Hubel, D. H., & Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology, 160*(1), 106–154.

Ioffe, S., Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv:1502.03167 [cs.LG]. Accessed 25 Jun 2019.

Jia, Y., & Ma, J. (2017). What can machine learning do for seismic data processing? An interpolation application. *Geophysics, 82*(3), V163–V177.

Kingma, D.P., Ba, J. (2014). Adam: A method for stochastic optimization. arXiv:1412.6980 [cs.LG]. Accessed 25 Jul 2019.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., & Others., (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86*(11), 2278–2324.

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

Li, S., Liu, B., Ren, Y., Chen, Y., Yang, S., Wang, Y., Jiang, P. (2019). Deep learning inversion of seismic data. arXiv:1901.07733 [cs.CV]. Accessed 25 Jul 2019.

Maddison, C.J., Huang, A., Sutskever, I., Silver, D. (2014). Move evaluation in go using deep convolutional neural networks. arXiv:1412.6564 [cs.LG]. Accessed 24 Jun 2019.

McCann, M. T., Jin, K. H., & Unser, M. (2017). Convolutional neural networks for inverse problems in imaging: A review. *IEEE Signal Processing Magazine, 34*(6), 85–95.

Perol, T., Gharbi, M., & Denolle, M. (2018). Convolutional neural network for earthquake detection and location. *Science Advances, 4*(2), e1700578.

Ronneberger, O., Fischer, P., & Brox, T. (2015a). Dental X-ray image segmentation using a U-shaped Deep Convolutional network. *International Symposium on Biomedical Imaging-ISBI 2015*.

Ronneberger, O., Fischer, P., & Brox, T. (2015b). U-Net: Convolutional Networks for Biomedical Image Segmentation. In N. Navab, J. Hornegger, W. Wells, & A. Frangi (Eds.), *International Conference on Medical image computing and computer-assisted intervention-MICCAI 2015. Lecture Notes in Computer Science* (Vol. 9351, pp. 234–241). Cham: Springer.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature, 323*, 533–536.

TGS Salt Identification Challenge. (2018). https://www.kaggle.com/c/tgs-salt-identification-challenge. Accessed 5 May 2019.

Sørensen, T. J. (1948). A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on Danish commons. *Det Kongelige Danske Videnskabers Selskab Biologiske Skrifter, 5*, 1–34.

Stockwell, J. W., Jr. (1999). The CWP/SU: seismic Un*x package. *Computers & Geosciences, 25*(4), 415–419.

Waldeland, A. U., Jensen, A. C., Gelius, L.-J., & Schistad Solberg, A. H. (2018). Convolutional neural networks for automated seismic interpretation. *The Leading Edge, 37*(7), 529–537.

Wang, B., Zhang, N., Lu, W., & Wang, J. (2018). Deep-learning-based seismic data interpolation: A preliminary result. *Geophysics, 84*(1), V11–V20.

Yuan, S., Liu, J., Wang, S., Wang, T., & Shi, P. (2018). Seismic waveform classification and first-break picking using convolution neural networks. *IEEE Geoscience and Remote Sensing Letters, 15*(2), 272–276.

Zeiler, M.D. (2012). ADADELTA: an adaptive learning rate method. arXiv:1212.5701 [cs.LG]. Accessed 4 Jun 2019.

(Received August 12, 2019, revised December 26, 2019, accepted December 27, 2019)