Improving the resolution of migrated images by approximating the inverse Hessian using deep learning

Harpreet Kaur¹, Nam Pham¹, and Sergey Fomel¹

ABSTRACT

We have estimated migrated images with meaningful amplitudes matching least-squares migrated images by approximating the inverse Hessian using generative adversarial networks (GANs) in a conditional setting. We use the CycleGAN framework and extend it to the conditional CycleGAN such that the mapping from the migrated image to the true reflectivity is subjected to a velocity attribute condition. This algorithm is applied after migration and is computationally efficient. It produces results comparable to iterative inversion but at a significantly reduced cost. In numerical experiments with synthetic and field data sets, the adopted method improves image resolution, attenuates noise, reduces migration artifacts, and enhances reflection amplitudes. We train the network with three different data sets and test on three other data sets, which are not a part of training. Tests on validation data sets verify the effectiveness of the approach. In addition, the field-data example also highlights the effect of the bandwidth of the training data and the quality of the velocity model on the quality of the deep neural network output.

INTRODUCTION

The ultimate goal of seismic imaging is to retrieve a high-resolution image of the true reflectivity of the earth’s subsurface structures. Increasing the resolution of images becomes especially important for unconventional reservoir characterization to detect small-scale geologic features controlling production efficiency.

However, imaging algorithms often use the adjoint of the forward modeling operator rather than the inverse operator to estimate the subsurface model and therefore are unable to fully reverse seismic wave propagation effects (Claerbout and Black, 2001; Rickett, 2003; Clapp, 2005). Hence, standard depth migration often suffers from low resolution, uneven amplitude, limited bandwidth, and migration artifacts.


A systematic approach to preserve the amplitudes with wave equation migration is to formulate it as an inverse problem using iterative least-squares migration (Tarantola, 1987; Chavent and Plessix, 1999; Nemeth et al., 1999; Ronen and Liner, 2000; Kühl and Sacchi, 2003; Valenciano et al., 2006). Least-squares migration with regularization has proven effective with incomplete surface data (Nemeth et al., 1999) and irregular surface illumination due to complex structures (Prucha et al., 1999). Currently, least-squares migration is applied in either the data domain (Tarantola, 1987; Schuster, 1993; Chavent and Plessix, 1999; Nemeth et al., 1999; Duijndam et al., 2000; Plessix and Mulder, 2004; Symes, 2008)
or the model domain (Hu et al., 2001; Rickett, 2003; Guitton, 2004; Yu et al., 2006; Lecomte, 2008). The model-domain approach solves the Hessian matrix with few iterations but requires substantial memory, whereas the data-domain approach requires less memory with more migration iterations to solve the inversion problem (Wang et al., 2016).

In this paper, we propose to approximate the effect of the inverse Hessian using a deep neural network (DNN) and thus circumvent the need for iterative inversion. This adaptive nonlinear model makes the algorithm highly flexible and computationally efficient, and it is able to produce seismic images with correct amplitudes at a high resolution. We test the proposed method using a variety of synthetic examples and a field-data example.

**THEORY**

Wave equation imaging, particularly by reverse time migration (RTM), is currently the preferred method of seismic depth migration in complex areas (Liu et al., 2011). Conventionally, RTM makes use of the adjoint operator \( L^T \) to approximate the inverse of the forward modeling. Because \( L^T \) is not the exact inverse to the forward modeling operator, the migrated image is only a rough approximation of reflectivity (Hu et al., 2001). The accuracy of this approximation can be degraded further by the limited aperture, spatial aliasing, and nonuniform illumination due to the complex overburden (Claerbout, 1992; Wang et al., 2016). Because of these effects, the final image may appear blurred and exhibit uneven amplitudes (Gray, 1997; Guitton, 2004). A migration method capable of undoing such distortions is called a “true amplitude migration” (Gray, 1997); least-squares inversion is one such true amplitude migration approach (Ronen and Liner, 2000).

Least-squares migration can be implemented as an iterative inversion (Tarantola, 1987; Schuster, 1993; Chavent and Plessix, 1999; Nemeth et al., 1999; Duijndam et al., 2000; Plessix and Mulder, 2004; Symes, 2008; Wang et al., 2016) or approximated by a single operator (Hu et al., 2001; Rickett, 2003; Guitton, 2004; Yu et al., 2006; Lecomte, 2008; Wang et al., 2016). Instead of a data-domain approach in which each iteration requires at least one migration and one modeling, the Hessian of the least-squares objective function can be approximated and the problem can be solved in the image domain where each iteration requires relatively cheap matrix-vector multiplications (Hou and Symes, 2015).

Given a vector of seismic data \( \mathbf{d} \) and a linear modeling operator \( \mathbf{L} \), our aim is to find the model \( \mathbf{m} \) that best represents the data \( \mathbf{d} \). The model of the earth can be parameterized in terms of reflectivity or impedance as a function of depth or time (Guitton, 2004, 2017). The linear modeling operator \( \mathbf{L} \) is related mathematically to the migration operator by its adjoint (Claerbout, 1985; Guitton, 2004):

\[
\mathbf{m}_1 = \mathbf{L}^T \mathbf{d}, \quad (1)
\]

where \( \mathbf{m}_1 \) is the migrated image from the data vector \( \mathbf{d} \). Here the migration operator \( \mathbf{L}^T \) can be any operator such as the Kirchhoff migration operator, the downward continuation operator, or the RTM operator. In this work, we use RTM.

The objective function \( f(\mathbf{m}) \) needs to be minimized for estimating true reflectivity \( \mathbf{m} \) in the least-squares formulation:

\[
f(\mathbf{m}) = \| \mathbf{Lm} - \mathbf{d} \|^2. \quad (2)
\]

Then, the least-squares estimate \( \mathbf{m} \) of the model is given by

\[
\mathbf{m} = (\mathbf{L}^T \mathbf{L})^{-1} \mathbf{L}^T \mathbf{d}. \quad (3)
\]

\[
\mathbf{m} = (\mathbf{L}^T \mathbf{L})^{-1} \mathbf{m}_1. \quad (4)
\]

Here, \( (\mathbf{L}^T \mathbf{L})^{-1} \) is the inverse Hessian, which also can be described as a nonstationary deconvolution operator for the amplitude correction of migrated images (Hu et al., 2001). The task here is to approximate the inverse Hessian, which involves a large computational cost and is not feasible to compute directly. In this paper, we adopt generative adversarial networks (GANs) to train the network to approximate the inverse Hessian. Then, this computed inverse Hessian can be applied to the migrated image, equation 1, for test data sets to simulate the result of least-squares inversion, equation 3, at a significantly reduced cost.

To approximate the inverse Hessian, we start with the true reflectivity model \( \mathbf{m} \), which we compute by converting the acoustic impedance to the reflectivity. Next, we model data \( \mathbf{d} \) using \( \mathbf{m} \) as follows:

\[
\mathbf{d} = \mathbf{Lm}. \quad (5)
\]

Then, we migrate data \( \mathbf{d} \) with adjoint operator \( \mathbf{L}^T \) to obtain \( \mathbf{m}_1 \) in equation 1. The relationship between \( \mathbf{m} \) and \( \mathbf{m}_1 \) is given by equation 4. Knowing \( \mathbf{m} \) and \( \mathbf{m}_1 \), the inverse Hessian can be approximated using DNNs and then applied to \( \mathbf{m}_1 \) equation 4 to simulate the least-squares inverse solution for the test data sets. Once the neural network learns to approximate the inverse Hessian, validation data sets (not previously seen by the network during training) consisting of a migrated image and the corresponding velocity model can be given as an input to the neural network to compute the true reflectivity. In the absence of true reflectivity, an alternative to the proposed approach for training the neural network is to use the first migrated and second migrated images to approximate the inverse of the Hessian (Guitton, 2004; Greer et al., 2018).

**IMPLEMENTATION OF THE DNN: GAN**

The GANs are deep neural net architectures introduced by Goodfellow et al. (2014) to sidestep the difficulty of approximating many intractable probabilistic computations (Mirza and Osindero, 2014). In the GAN framework, a generator network and a discriminator network are trained jointly where a generative model (\( G \)) captures the data distribution to generate a sample, and a discriminative model (\( D \)) estimates the probability that a sample came from training data rather than \( G \) (Goodfellow et al., 2014). The network learns how to capture the characteristics of the input images to translate these characteristics into the output images. Assuming that an underlying relationship exists between the source and the target domain, the network seeks to learn that relationship. The network implicitly learns a latent, low-dimensional representation of arbitrary high-dimensional data.

Adversarial networks have the advantage that Markov chains are not needed; only backpropagation is used to obtain gradients (Mirza and Osindero, 2014). These networks require some kind of regularization, which can guarantee that the learned function can map an individual input \( x_i \) to the desired output \( y_i \). This regularization is introduced by Zhu et al. (2017) in CycleGAN by incorporating
cyclic loss, which ensures that the transformation from source distribution to target distribution and then back to source distribution should produce samples from the source distribution. However, in an unconditional generative model, there is no control on modes of data being generated. By conditioning the model with additional information, it is possible to direct the data-generation process (Mirza and Osindero, 2014).

We extend CycleGAN to conditional CycleGAN by requiring that mapping from \( x_A \) to \( x_B \) is subjected to an attribute condition of the corresponding velocity model (Lu et al., 2017), where \( x_A \) is migrated image (\( m_A \)) along with the corresponding velocity model and \( x_B \) is the true reflectivity (\( m \)) (Kaur et al., 2019). This makes it possible to engage the learned generative model in different “modes” by providing it with different contextual information (Gauthier, 2014). Further, the network uses adversarial loss \( L_{GAN} \) (Goodfellow et al., 2014; Benaim and Wolf, 2017; Zhu et al., 2017), cyclic loss \( L_{cycle} \) (Zhu et al., 2017), self-distance loss \( L_{self-distance} \) (Benaim and Wolf, 2017), and identity loss \( L_{Identity} \) (Zhu et al., 2017), which adds to give total loss function \( L_{total} \) to reduce the space of possible mapping functions and to regularize the model.

**Adversarial loss function**

The training data sets consist of two discrete distributions \( \hat{p}_A \) and \( \hat{p}_B \) sampled from the source and the target distributions \( p_A \) and \( p_B \). We denote the data distributions as \( x_A \sim \hat{p}_A \) and \( x_B \sim \hat{p}_B \). For the proposed algorithm, the source distribution is a migrated image with incorrect amplitudes computed using the adjoint of the forward modeling operator (equation 1) and the target distribution is the image with true amplitudes (equation 3). For the mapping function \( G: A \rightarrow B \) and its discriminator \( D_B \), the objective function is given as (Goodfellow et al., 2014; Benaim and Wolf, 2017; Zhu et al., 2017)

\[
L_{GAN}(G_{AB}, D_B, \hat{p}_A, \hat{p}_B) = E_{x_A \sim \hat{p}_A} \log D_B(x_B) + E_{x_B \sim \hat{p}_B} \log \left\{ 1 - D_B \left[ G_{AB}(x_A) \right] \right\}.
\] (6)

Here, \( E_{x_B \sim \hat{p}_B} \) is the expected value over all of the generated instances and \( E_{x_A \sim \hat{p}_A} \) is the expected value over all of the generated fake instances. This objective function is simultaneously maximized over \( D_B \) and minimized over \( G_{AB} \). The main goal of adversarial loss is to match the distribution of generated images to the data distribution of the target domain.

**Cycle consistency loss function**

Adversarial losses alone cannot guarantee that the learned function accurately maps input \( A \) to desired output \( B \). The addition of this constraint ensures that mapping of given data sample from domain \( A \) to domain \( B \) and then back to domain \( A \) results in an identical sample. It reduces the space of possible mapping functions. Its objective function has the following form (Zhu et al., 2017):

\[
L_{cycle}(G_{AB}, G_{BA}, D_B, \hat{p}_A) = E_{x_A \sim \hat{p}_A} \| G_{BA}(G_{AB}(x)) - x \|_1.
\] (7)

The main goal of the cycle consistency loss function is that, if a sample from domain \( A \) is translated to domain \( B \) using \( G_{AB} \), then the translation of this generated domain \( B \) sample back to domain \( A \) using generator \( G_{BA} \) should result in an identical sample.

**Identity loss function**

We apply an additional constraint on the generator such that \( G_{AB} \) applied to samples from domain \( B \) performs the identity mapping (Taigman et al., 2016; Benaim and Wolf, 2017). It regularizes the generator to be near an identity mapping when real samples of the target domain are provided as input to the generator (Zhu et al., 2017). Its objective function is given as

\[
L_{Identity}(G_{AB}, \hat{p}_B) = E_{x \sim \hat{p}_B} \| x - G_{AB}(x) \|_1. \] (8)

**Self-distance loss function**

To recover the alignment more effectively, we use the self-distance loss in addition to the cycle consistency loss. It uses one sample at a time and compares the distances between two parts of the input sample and two parts of the generated sample (Benaim and Wolf, 2017). It is defined as follows. Let \( L, R : R^{k \times w} \rightarrow R^{k \times w/2} \) be the operators that, given an input image, return the left and right part of it. Then, the self-distance loss equation is given as

\[
L_{self-distance}(G_{AB}, \hat{p}_A) = E_{x \sim \hat{p}_A} \left[ \frac{1}{\sigma_A} \left( \| L(x) - R(x) \|_1 - \mu_A \right) \right] - \frac{1}{\sigma_B} \left( \| L\left[ G_{AB}(x) \right] - R\left[ G_{AB}(x) \right] \|_1 - \mu_B \right), \] (9)

where \( \mu_A \) and \( \sigma_A \) are the mean and standard deviations, respectively, of the pairwise distances between the two halves of the image in the training set from domain \( A \); similarly, \( \mu_B \) and \( \sigma_B \) are the mean and standard deviations, respectively, of the pairwise distances between the two halves of the image in the training set from domain \( B \).

**NETWORK ARCHITECTURE AND TRAINING**

While training the network, the full objective function takes the following form:

\[
L_{total}(G, D) = \alpha_{A} L_{GAN}(G_{AB}, D_B, \hat{p}_A, \hat{p}_B) + \alpha_{2A} L_{cycle}(G_{AB}, G_{BA}, \hat{p}_A) + \alpha_{3A} L_{self-distance}(G_{AB}, \hat{p}_A) + \alpha_{4A} L_{Identity}(G_{AB}, \hat{p}_B). \] (10)

In equation 10, parameters \( \alpha_A \) and \( \alpha_B \) are trade-off parameters. After some experimentation, we have chosen them as follows:

\[
\alpha_A = \alpha_B = 1, \] (11)
\[
\alpha_{2A} = 10, \] (12)
\[
\alpha_{4A} = 5. \] (13)

We apply max-pooling with a stride in the generator and the discriminator network with nine residual blocks in the transformation.
layer of the generator. We use instance normalization to counteract the internal covariant shift and reduce the oscillations when approaching the minimum point (Johnson et al., 2016). In comparison to batch normalization, instance normalization removes instance-specific information from the data, which simplifies generation and leads to improved learning (Ulyanov et al., 2016). For GAN loss, the algorithm uses binary cross entropy error because it can find a better local optimum than the mean square error loss function (Golik et al., 2013). We use the history of images for the discriminator, rather than only the last image generated (Benaim and Wolf, 2017), which helps to improve the network performance by acting as a damping force to converge the model. To keep the output finite, we use leaky ReLU as the activation function for efficient gradient propagation to increase the performance of the network by allowing a small nonzero gradient when the unit is not active (Maas et al., 2013). The computations in this paper are done using the Madagascar software package (Fomel et al., 2013a) and Pytorch (Paszke et al., 2017).

NUMERICAL EXAMPLES

To train the algorithm to compute the inverse Hessian, we select three different synthetic velocity models: the Marmousi model (Bourgeois et al., 1991), the SEG/EAGE salt model (Aminzadeh et al., 1997), and the BP 2.5D model (Etgen and Regone, 1998). All of the models have significant structural complexity.

Training data sets

1997 BP 2.5D model

We demonstrate the training algorithm using the BP 2.5D model (Etgen and Regone, 1998), which is one of the three training models. Using the velocity model shown in Figure 1a, we first compute acoustic impedances and then convert them into corresponding reflectivities to obtain $m$ as shown in Figure 1b. Next, we generate poststack data sets as shown in Figure 1c from these reflectivity

Figure 1. (a) The BP 2.5D velocity model, (b) reflectivity for the 1994 BP 2.5D benchmark model (i.e., $m$), (c) poststack data created from the reflectivity model, (d) RTM image (i.e., $m_1$), and (e) image estimated after applying the inverse Hessian computed by the DNN to (d).

Figure 2. Image estimated after applying the inverse Hessian computed by the DNN to Figure 1d (a) with the velocity model as a conditioner and (b) without the velocity model as a conditioner.
models using low-rank wave extrapolation. Then, we migrate these data sets using low-rank zero-offset RTM (Fomel et al., 2013b) to compute the migrated image ($m_1$) shown in Figure 1d. This approach is similar to Guitton (2004), where $m$ is equivalent to the first migrated image and $m_1$ corresponds to the second migrated image, modeled and migrated from $m$. Using $m$ and $m_1$ conditioned with the respective velocity models as inputs, we train the neural network to compute the inverse Hessian. For training, we divide the inputs into $200 \times 200$ sample patches having 108 patches for the Marmousi data set, 16 patches for the SEG/EAGE salt model, and 42 patches for the BP 2.5D data set. Next, we apply the inverse Hessian computed by DNN to the migrated images $m_1$ to recover the true reflectivities for the respective data sets shown in Figure 2a. Considerable improvement is observable in the image approximated by the DNN shown in Figure 2a for the BP 2.5D data set. Figure 3 highlights various regions for a closer comparative analysis of the migrated image, true reflectivity, and approximated image (DNN output). The thrust zone separating the Carpathians on the left from

![Figure 3](image-url)

Figure 3. Regions of the migrated image used for comparative analysis, with the blue boxes indicating the magnified area shown in Figure 4.

Figure 4. Magnified sections, with the migrated image on the left, true reflectivity in the middle, and DNN output image on the right: (a) over the thrust zone, (b) over the beds to the right of the thrust zone, and (c) over the reflector at the bottom. Migration artifacts are suppressed, with the reflectors, thrust zone, and faults being clearly visible in the DNN output image as compared to the migrated image.
North Sea-type salt ridge sediments on the right (Etgen and Regone, 1998) is clearly delineated in the approximated image shown in Figure 4a (panel c) as compared to the migrated image shown in Figure 4a (panel a) and is very similar to the true reflectivity shown in Figure 4a (panel b). Apart from this, beds on either side of the thrust zone are clearly visible in the image output by DNN shown in Figure 4b (panel c) and the reflector at the bottom is evident shown in Figure 4c (panel c). Also, the spatial resolution of the image is increased, and migration artifacts are suppressed. Moreover, the amplitude behavior of the output DNN image is very similar to reflectivity, which shows that our algorithm corrects the amplitudes of migrated images.

Validation data sets

To test the efficiency of the training, we use two different models: the SEG/EAGE overthrust model (Aminzadeh et al., 1995) and the BP 2004 gas model (Billette and Brandsberg-Dahl, 2004). Similar to the training data sets, we divide test data sets into patches of 200 × 200 samples with total 60 patches having 42 patches for the SEG/EAGE overthrust model and 18 patches for the BP 2004 gas cloud model. It is important to stress that the network has not yet seen what true reflectivity looks like for these testing models because they have not been included in training. The conditional CycleGAN network applies the learned weights during the training phase to the migrated images of validation data sets.

SEG/EAGE overthrust model

For the first testing example, we use the SEG/EAGE overthrust model shown in Figure 5a. The 20 × 4.67 km model has spatial sampling of 25 m in the vertical and horizontal directions (Aminzadeh et al., 1995). We generate poststack data using low-rank extrapolation as shown in Figure 5c. We use low-rank RTM to...
obtain the migrated image shown in Figure 5d. Next, we input the migrated image conditioned with the corresponding velocity model to the trained neural network to apply the inverse Hessian learned during the training phase to obtain the output image shown in Figure 6a. We highlight certain areas of the image shown in Figure 7 to extract magnified sections for detailed analysis. In the approximated image by DNN, fault zones are clearly imaged in Figure 8a (panel c), along with considerable noise attenuation shown in Figure 8b (panel c) as compared with the original migrated images shown in Figures 8a (panel a) and 8b (panel a). Also, amplitudes of the approximated images are closer to the true reflectivities shown in Figure 8a (panel b) and 8b (panel b).

BP 2004 model

For the final example, we use a portion of the BP 2004 velocity model, which previously has been used by Sun et al. (2016) for least-squares reverse time migration (LSRTM). The model features a low-velocity, low-Q area, which is assumed to be caused by the presence of a gas chimney (Figure 9a). The model has 12.5 m sampling along the vertical and horizontal directions. In total, 31 shots with spacing of 162.5 m are modeled (Figure 9c) using a Ricker wavelet with 22.5 Hz peak frequency (Sun et al., 2016). Performing dispersion-only RTM leads to the results shown in Figure 9d, which suffer from poor illumination below the gas chimney. We input the RTM image conditioned with the velocity model to the neural network; using previous training, the network outputs the image shown in Figure 9e, which is better than the result obtained after 100 iterations of LSRTM shown in Figure 9f. Overall, there is considerable noise attenuation, suppression of migration artifacts, and an evident increase in the resolution of the approximated image.

Analyzing the effect of the velocity model on the DNN output

We analyze the effect of using the velocity model as a conditioner on the DNN output by examining three different scenarios for the BP 2.5D model and the SEG/EAGE overthrust model, which include the DNN output with the exact velocity model as a conditioner, without the velocity model as a conditioner, and with the smooth velocity model as a conditioner. We first compare the DNN output with the exact velocity model as a conditioner as shown in Figure 2a and the DNN output without the velocity model as a conditioner as shown in Figure 2b. Comparing the two results, we can...
see that, with the exact velocity model as a conditioner, the reflectors on the left of the thrust zone are imaged more clearly in the DNN output as shown in Figure 2a than the DNN output without using the velocity model as a conditioner, as shown in Figure 2b. Also, the bottom reflector is more continuous and noise is better attenuated when we use the velocity model as a conditioner. Next, we analyze the effect of using the smooth velocity model as a conditioner on the DNN output with different smoothing radii. We start with a smoothing radius of 20 grids, as shown in Figure 10a, with the corresponding DNN output shown in Figure 10b. We further

![Figure 9](image)

Figure 9. (a) The BP 2004 velocity model, (b) reflectivity model (i.e., \( m \)), (c) prestack data generated from reflectivity model, (d) RTM image (i.e., \( m_1 \)), (e) image estimated after applying the inverse Hessian computed by DNN to (d), and (f) image obtained after 100 iterations of least-squares RTM.

![Figure 10](image)

Figure 10. (a) The smoothed BP 2.5D velocity model with a smoothing radius of 20 grids, (b) image estimated after applying the inverse Hessian computed by DNN to Figure 1d using the smoothed velocity model in (a) as a conditioner, (c) the smoothed BP 2.5D velocity model with a smoothing radius of 40 grids, and (d) image estimated after applying the inverse Hessian computed by the DNN to Figure 1d using the smoothed velocity model in (c) as a conditioner.
increase the smoothing radius to 40 grids. The smooth velocity model is shown in Figure 10c, with the corresponding DNN output as shown in Figure 10d. Analyzing the effect of the smooth velocity model as a conditioner, we can see that, with the increase in the smoothing radii, some of the events on the left of the thrust zone are not recovered; however, the increase in the smoothing radii does not significantly affect the output of the DNN. Therefore, the use of the velocity model as a conditioner improves the image but is not the primary source of information for the DNN output. We repeat the same experiment on the SEG/EAGE overthrust model. Figure 6a and 6b shows a comparison of the DNN output with and without using the velocity model as a conditioner, respectively. With the velocity model as a conditioner, reflectors are imaged more clearly and noise is better attenuated. Further, we analyze the effect of using the velocity model with increasing smoothing radii. Figure 11a shows the velocity model with a smoothing radius of 10 grids, with the corresponding DNN output shown in Figure 11b. We further increase the smoothing radius to 20 grids. The smooth velocity model is shown in Figure 11c, with the corresponding DNN output as shown in Figure 11d. The use of the velocity model, whether smooth or exact, as a conditioner improves the resolution of the migrated images, reduces migration artifacts, and enhances the reflectors.

**Field-data example**

For the field-data example, we use the Viking Graben data set shown in Figure 12a. We use the velocity model shown in Figure 12b to obtain the migrated image in Figure 12c using low-rank RTM. We input the migrated image conditioned with the corresponding velocity model to the trained neural network to apply the weights associated with the inverse Hessian learned during the training phase to produce the output image shown in Figure 12d. There is an evident increase in resolution of the recovered image, along with the improved delineation of the fault zones in the lower part of the image. However, the quality of the output image in the shallow section is degraded, a consequence of the bandwidth of the training data sets. The deeper section has a bandwidth with a peak frequency of 20 Hz and is the same frequency range that the network has seen during the training phase. At the same time, the shallow section contains relatively high frequencies that have not been present during training, which prevents the high frequencies in the shallow section from being recovered effectively in the DNN output. If we focus on the cross-bedding structures present in the shallow part of the section (between \( x = 5 \) and 10 kms and \( t = 0.4 \)\( \times 10^3 \))...
and 0.6 s), these are present in the original migrated image with high-frequency content, but the absence of this high-frequency information in the DNN output causes these features to disappear. This shows that the bandwidth of the training data is important to obtain a high-quality output image with the required frequency content. In addition, the field-data example also highlights the importance of using the velocity model as a conditioner. It is evident from the previous examples that, although the velocity model is not the primary source of information, it does help to improve the quality of the output image by incorporating nonstationarity. An overly smoothed version of the true velocity model may not fully serve the purpose.

CONCLUSIONS

We present a robust method for increasing resolution of migrated images using DNNs. In the proposed approach, the relationship between the migrated image \( m \) and the true reflectivity \( m_0 \), the inverse Hessian operator, is captured by the neural networks during the training phase and then can be applied to the original migrated image conditioned with the velocity model in the testing phase to obtain an image analogous to a least-squares migrated image at a significantly reduced cost. Using a variety of synthetic and field-data examples, we demonstrate that this approach is capable of recovering true amplitudes, suppressing migration artifacts, improving resolution, and attenuating considerable noise in the recovered images. However, for real field-data application, a few additional aspects need to be taken into consideration. First, the bandwidth of the training data is important. If the network has not seen certain frequencies during the training phase, there is a lower probability of recovering these frequencies during the test phase. Second, the velocity model is an important component of the entire process. Although it is not the primary source of information during the feature vector extraction process, it plays an important role in defining the quality of the output image. If the input velocity model is an overly smoothed version of the actual velocity model, which is the case in our field-data example, then the velocity model might not be able to properly account for nonstationarity. For the proposed algorithm to work with noisy data, training labels with noise need to properly account for nonstationarity. For the proposed algorithm to work with noisy data, training labels with noise need to properly account for nonstationarity. An overly smoothed version of the true velocity model may not fully serve the purpose.

ACKNOWLEDGMENTS

We thank the sponsors of the Texas Consortium of Computational Seismology for the financial support. We acknowledge Nvidia for providing a Titan Xp GPU for running this algorithm.

DATA AND MATERIALS AVAILABILITY

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

REFERENCES


Aminzadeh, F., B. Jean, and T. Kunz, 1997, 3-d salt and overtrust models: SEG.


