

Deep learning for velocity model building with common-image gather volumes

Zhicheng Geng¹, Zeyu Zhao,² Yunzhi Shi,^{3,*} Xinming Wu⁴, Sergey Fomel¹ and Mrinal Sen²

¹Bureau of Economic Geology, The University of Texas at Austin, Austin, TX 78713, USA. E-mail: zhichenggeng@utexas.edu

²Institute for Geophysics, The University of Texas at Austin, Austin, TX 78713, USA E-mail: zeyu@utexas.edu

³Formerly Bureau of Economic Geology, The University of Texas at Austin, Austin, TX 78713, USA

⁴School of Earth and Space Sciences, University of Science and Technology of China, Hefei 230026, China

Accepted 2021 September 20. Received 2021 September 7; in original form 2020 August 16

SUMMARY

Subsurface velocity model building is a crucial step for seismic imaging. It is a challenging problem for conventional methods such as full-waveform inversion (FWI) and wave equation migration velocity analysis (WEMVA), due to the highly nonlinear relationship between subsurface velocity values and seismic responses. In addition, traditional FWI and WEMVA methods are often computationally expensive. In this paper, we propose to apply a deep learning technique to construct subsurface velocity models automatically from common-image gather (CIG) volumes. In our method, pairs of synthetic velocity models and CIG volumes are generated to train a convolutional neural network. Our proposed network achieves promising results on different synthetic data sets. The training performance of several commonly used loss functions is also studied.

Key words: Image processing; Neural networks, fuzzy logic; Numerical solutions; Computational seismology.

1 INTRODUCTION

Subsurface velocity model building plays a critical role in understanding complex subsurface geological structures. An accurate velocity model is essential to most seismic imaging methods. Various velocity model building methods, including ray-based tomography (Dines & Lytle 1979), migration velocity analysis (MVA, Al-Yahya 1989) and full-waveform inversion (FWI, Tarantola 1984), have been proposed to recover subsurface structures in many studies. Conventional ray-based tomography methods (Dines & Lytle 1979; Bishop *et al.* 1985; Billette & Lambaré 1998; Bube & Langan 1999; Clapp *et al.* 2004) are based on the high-frequency approximation (Cerveny 2001). In these methods, ray tracing is usually used to compute the traveltimes, which makes ray-based tomography computationally efficient. The sensitivity kernel reflects the sensitivity of traveltime perturbations to velocity perturbations. The velocity models are updated by projecting the estimated traveltime perturbations back to raypaths. MVA (Al-Yahya 1989; Stork 1992; Liu & Bleistein 1995; Biondi & Sava 1999; Varela *et al.* 2002; Sava & Biondi 2004; Shen & Symes 2008) is a class of methods that utilizes migration results, usually common image gathers (CIGs), to analysis the accuracy of the seismic migration velocity and to update the velocity model. This category of method is based on the assumption that, given an accurate migration velocity model, seismic migration methods should produce flat or focused CIGs, which means that seismic responses, measured at the surface, are positioned to their correct subsurface locations. The subsurface image would be positioned at the wrong spatial location both vertically and horizontally due to the incorrect velocity model; additionally, inaccurate velocity models would lead to unflat or unfocused CIGs. MVA methods often only evaluate the flatness or focusing energy in CIGs to refine velocity models. Depending on the operator that MVA methods use for updating the velocity model, they can be split into two groups: ray-based MVA (Al-Yahya 1989; Liu & Bleistein 1995; Yilmaz 2001) and wave-equation MVA (WEMVA, Biondi & Sava 1999; Sava & Biondi 2004; Shen & Symes 2008). By iteratively updating the subsurface velocity model according to the flatness of seismic events or the focusing energy in CIGs, image quality can be improved with the updated subsurface model. FWI methods exploit full-waveform information from seismic gathers. The goal of FWI is to construct high-resolution subsurface velocity models directly from observed seismic data (Virieux & Operto 2009). Its theory

*Now at: Amazon Web Services, Austin, TX 78758, USA

was originally proposed in the 1980s (Lailly 1983; Tarantola 1984) and then it has been widely applied since the late 1990s (Pratt *et al.* 1998; Pratt 1999; Bozdağ *et al.* 2016). Conventional FWI is formulated using a least-squares misfit, and local optimization methods are typically used to update the model with the derivative information of the misfit function. Due to the nature of the highly nonlinear relationship between a velocity model and seismic data, multiple local minima present in the parameter space. Widely used local optimization based algorithms are very likely to be trapped at one of the local minima given an inaccurate starting model. Various misfit functions have been investigated to mitigate the local minima issue, such as dynamic warping misfit (Ma & Hale 2013), adaptive misfit (Warner & Guasch 2016), adaptive matching misfit (Zhu & Fomel 2016) and optimal transport misfit (Engquist & Froese 2013; Yang *et al.* 2018). Alternatively, one can tackle the local minima issue by employing ‘global search’ methods, which include global optimization (Tarantola *et al.* 1990; Sen & Stoffa 1991; Stoffa & Sen 1991), a hybrid of local and global optimization methods (Datta & Sen 2016; Zhao & Sen 2021b) or Markov Chain Monte Carlo nonlinear sampling methods (Biswas *et al.* 2020; Zhao & Sen 2021a).

Utilizing deep learning techniques for subsurface velocity model building and imaging is an emerging research field. Deep learning (LeCun *et al.* 2015; Schmidhuber 2015; Goodfellow *et al.* 2016) is a subfield of machine learning methods. It uses artificial neural networks with multiple layers, for example, convolutional neural networks (CNN) and recurrent neural networks, to learn data representations from the input data. In deep learning, each layer of the network learns to transform the input data into a representation at higher and more abstract level so that complex functions can be learned and higher level features can be gradually extracted from the input data. Since the enormous success of AlexNet (Krizhevsky *et al.* 2012) in ImageNet competition (Deng *et al.* 2009), deep learning has attracted extensive attention and is rapidly evolving. The strong ability of deep learning to automatically extract features from input data makes it achieve promising results on computer vision problems, such as image classification (Krizhevsky *et al.* 2012; Zhang *et al.* 2015; Szegedy *et al.* 2015, 2016), image segmentation (Long *et al.* 2015; Chen *et al.* 2017; Lin *et al.* 2017a), object detection (Ren *et al.* 2015; He *et al.* 2017; Lin *et al.* 2017b) and image generation (Isola *et al.* 2017; Zhu *et al.* 2017; Karras *et al.* 2019). Successful applications of deep learning have already been made in geophysics, including seismic phases picking (Zhu & Beroza 2019; Mousavi *et al.* 2020), seismic data processing (Wu *et al.* 2019b; Zhu *et al.* 2019) and seismic interpretation (Wu *et al.* 2019a; Shi *et al.* 2019; Wu *et al.* 2020; Geng *et al.* 2020). Attempts to incorporate deep learning to subsurface velocity model estimation workflow have been made by several authors. Araya-Polo *et al.* (2018) propose estimating subsurface P -wave velocities by feeding semblance panels computed from shot gathers into a neural network. Yang & Ma (2019) attempted to directly estimate P -wave velocity models from shot gathers without manual feature extraction. Similar work has been done by several other authors (Liu *et al.* 2019; Li *et al.* 2020; Wang & Ma 2020).

In this paper, we propose applying a CNN to directly and automatically construct subsurface velocity models from a CIG volume. After the migration workflow, a correct velocity model could position seismic responses back to their correct subsurface locations, which also produces flat seismic events and focused seismic energy in CIGs, while inaccurate velocity model would cause incorrect positioning for seismic energy both vertically and horizontally, which also leads to unflat or unfocused seismic energy in CIGs. The focuses of seismic events in the migrated section and the flatness or focuses of events in CIGs indicate the correctness of a given velocity model. Therefore, a CIG volume generated by migration methods contain subsurface velocity information. Hence, we can infer subsurface velocity information from a CIG volume. In 1-D cases, a CIG volume is a 2-D section, we only need to adjust the velocity according to seismic events along the angle direction. While in 2-D or 3-D cases, a CIG volume becomes a 3-D or 4-D cube, both the spatial locations of seismic events and the flatness or focuses of events along the angle direction need to be considered. Simultaneously relating the spatial shifts of imaging points and flatness or focuses in CIGs to a velocity update with an analytical formulation is extremely challenging, especially for 2-D and 3-D applications (Jiao *et al.* 2002). Here, we explore the possibility of training deep learning networks to predict subsurface velocity models from information estimated from CIG volumes. The key idea is to have the neural network learn the relation between the incorrect positioning of seismic energy present in a CIG volume and its corresponding velocity update. Pairs of input CIG volumes and output target velocity models are used to train the network. Since the input and output of the network share the same physical dimension, it is more reasonable and easier for the network to learn the relationship than directly mapping data to model parameters that have nonlinear relationship and also different physical dimensions. To create the training data sets, we first generate synthetic velocity models with complex folding and faulting structures, which are used as the ground truth during training. Then modelling methods and migration methods are used to generate CIG volumes. CIG volumes computed from the true model and a reference model are used to train the neural network so that the network can figure out the necessary velocity update from the reference model to flat the events in CIG volumes and position the seismic energy to the correct position both horizontally and vertically. The output of the network is expected to be close to the true velocity model. In this research, we use a constant velocity model as the reference model. To apply the trained network on unseen data, the reference velocity model is used to first perform seismic migration on the data to generate a CIG volume, then the CIG volume is inputted in to the trained network to predict the subsurface velocity model. Test results on other synthetic models demonstrate the potential of the proposed deep learning method in subsurface velocity modelling building.

2 METHODS

2.1 Network architectures

Fig. 1 shows the proposed network architecture in our method. The input to this network is a ray-parameter CIG volume obtained with a 2-D reference velocity model, and the output is the desired accurate velocity model. For both input and output, the sample size along the vertical

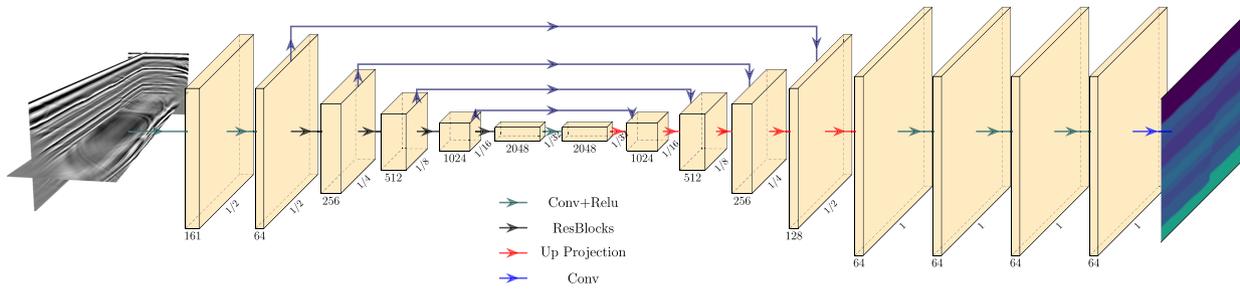


Figure 1. The proposed network architecture.

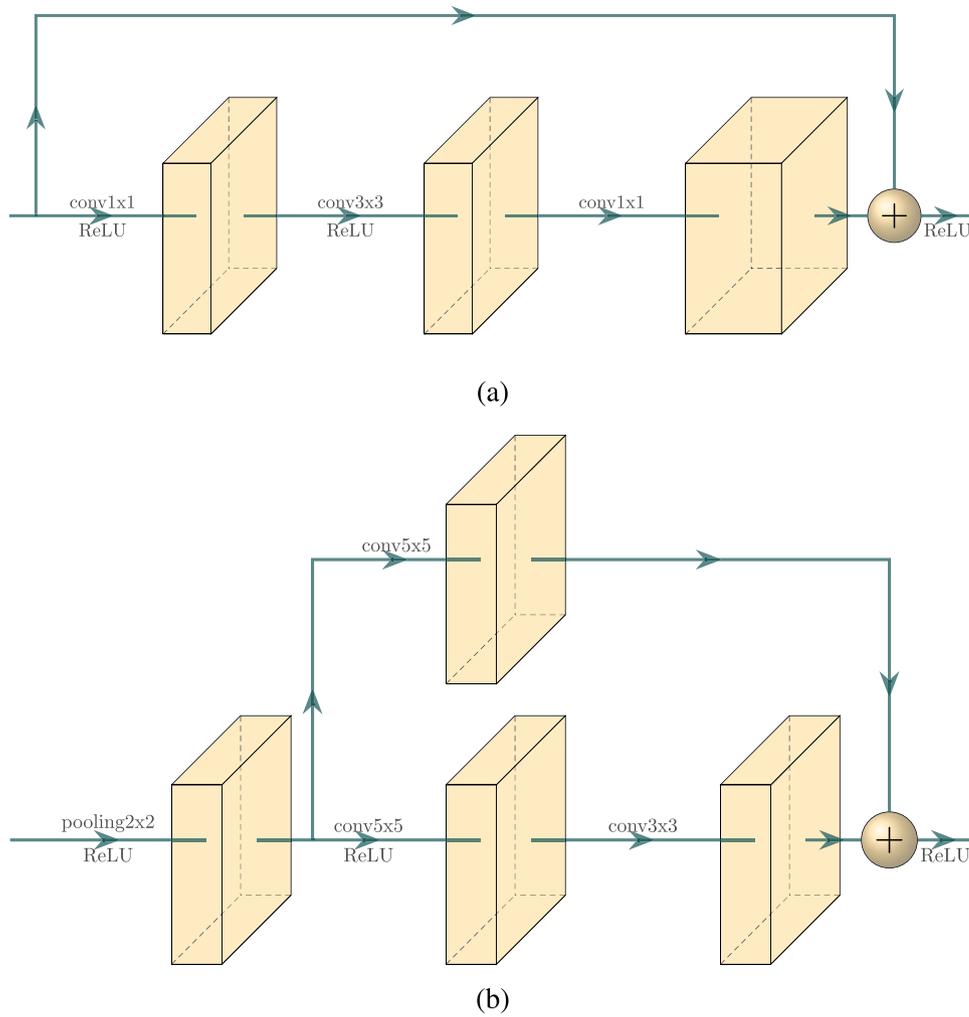


Figure 2. The structure of (a) a residual block and (b) an up-projection block.

and horizontal directions is 128 and 512, respectively. The spatial intervals along both directions are 20 m. Note that, in this work, we focus on building 2-D velocity models. The size of the ray-parameter dimension for the input CIGs is 161. Therefore, the input is a $128 \times 512 \times 161$ cube and the output is a 128×512 velocity model. In our method, the ray-parameter dimension in the CIGs is considered as the channel dimension, which means the input 3-D cube can be seen as a stack of several 2-D images. Therefore, although the input is 3-D, only 2-D convolutional layers are needed in the proposed network.

As shown in Fig. 1, the network uses the encoder–decoder scheme to capture multilevel information from the input. Each box in Fig. 1 represents the output feature maps of a convolutional layer or a block of multiple convolutional layers. The number at the bottom of each box denotes the number of features in the corresponding feature map, while the number at the right bottom represents the scale of feature maps relative to the original input. For example, the 6th box on the left represents the feature map with 2048 features and their sizes are 4×16 , which is $1/32$ of the original input CIGs. For the encoder part, ResNet-50 (He *et al.* 2016) is implemented, which is composed of several residual blocks (Fig. 2a). ResNet is an efficient feature extraction network architecture. It has been used as backbone of several state of the

Table 1. Details of layers/blocks in our proposed neural network. Input/C and Output/C columns show the number of channels of input and output at each layer/block.

Layer/block	Kernel	Scale	Input/C	Output/C
Conv1	5×5	1/2	161	161
Conv2	5×5	1/2	161	64
Block1		1/4	64	256
Block2		1/8	256	512
Block3		1/16	512	1024
Block4		1/32	1024	2048
Conv3	1×1	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
Conv4	5×5	1	64	64
Conv5	5×5	1	64	64
Conv6	5×5	1	64	1

art method in computer vision field (He *et al.* 2016; Chen *et al.* 2018). In the residual building blocks, the shortcut between two non-adjacent layers are introduced to solve the gradient vanishing and degradation problem. Degradation refers to the phenomenon that deeper networks gets higher error rate. Therefore, with the help of residual blocks, building a much deeper network is possible, which leads to the success of ResNet. To utilize ResNet-50 for our problem, a small modification is made on it. The first 7×7 convolutional layer of ResNet-50 is replaced by two convolutional layers (Conv1–Conv2) the kernel sizes of which are both 5×5 . The stride of Conv1 is 2 so that the input is downsampled to 1/2 scale, while the stride for the remaining layers is 1. The purpose of the first two convolutional layers is to gather information along the ray-parameter dimension and reduce the size of channel dimension to 64. In this way, a feature map with 64 features is fed to the successive blocks (Block1–Block4), each of which contains a block of multiple residual blocks from ResNet-50, so that the pre-trained weight of ResNet-50 on ImageNet data sets (Deng *et al.* 2009) can be used for these four blocks. The average pooling layer and the fully connected layer at the end of ResNet-50 are dropped in our network. Instead, we use a 1×1 convolutional layer (Conv2) to connect the encoder to the following decoder part for feature fusion in the channel dimension. For the decoder part, five up-projection blocks (Up1–Up5) (Laina *et al.* 2016) are implemented, the basic structure of which is shown in Fig. 2(b). The choice of parameters in the decoder part is based on constructing a symmetric encoder–decoder network, therefore, the output feature maps from each decoder block have the same size as those of the corresponding encoder block. After the decoder part, the feature map would be upscaled back to the same size as the input. Long skip connections between the encoder and the decoder at different scales are implemented to improve the spatial resolution. Three additional convolutional layers are added at the end of the network. The kernel size of the last three convolution layers is set as 5×5 to enlarge the receptive field so that a finer final result can be obtained. Detailed information about the proposed network, including the kernel size and the output feature size of each layer or block, is shown in Table 1.

2.2 Loss function

In image regression problems, the most commonly used loss functions are \mathcal{L}_1 losses, or mean absolute errors (MAE), and \mathcal{L}_2 losses, or mean squared errors (MSE). An \mathcal{L}_1 loss measures the absolute difference between the predicted and target values:

$$\mathcal{L}_1 = \frac{1}{N} \sum_{i=1}^N |v_i^p - v_i|, \quad (1)$$

while an \mathcal{L}_2 loss is computed using the squared differences between the predictions and true labels as follows:

$$\mathcal{L}_2 = \frac{1}{N} \sum_{i=1}^N (v_i^p - v_i)^2, \quad (2)$$

where N denotes the total number of pixels in one single image. v_i^p and v_i represent the predicted velocity and the true velocity, respectively.

Traditionally, \mathcal{L}_2 loss is easier to optimize, but the characteristic of penalizing large errors and tolerating small errors might cause artefacts in the results from the neural network (Zhao *et al.* 2016a). However, an \mathcal{L}_1 loss can tackle this problem by weighing errors the same, and it is also easy to optimize in deep learning problem with the help of standard deep learning packages such as TensorFlow (Abadi *et al.* 2016) and PyTorch (Paszke *et al.* 2019). Therefore, in our proposed method, we choose \mathcal{L}_1 losses over \mathcal{L}_2 losses as part of the difference measurement.

One disadvantage of \mathcal{L}_1 and \mathcal{L}_2 losses is that only pixel-by-pixel differences between predictions and targets are measured. Therefore, the local structural information, which is important for achieving high-resolution subsurface models, would not be retrieved by using \mathcal{L}_1 or \mathcal{L}_2 losses only. To better compare the structural difference between two images, Wang *et al.* (2004) proposed an image quality assessment called Structural Similarity (SSIM) index, which combines the comparison of luminance, contrast and structure. The SSIM index between two local image patches \mathbf{x} and \mathbf{y} can be formulated as follow:

$$\begin{aligned} \text{SSIM}(\mathbf{x}, \mathbf{y}) &= \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \cdot \frac{2\sigma_{xy} + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \\ &= l(\mathbf{x}, \mathbf{y}) \cdot cs(\mathbf{x}, \mathbf{y}), \end{aligned} \quad (3)$$

where μ_x and μ_y represent the mean values of \mathbf{x} and \mathbf{y} , respectively, and σ_x and σ_y are the corresponding standard deviation. σ_{xy} is the covariance of \mathbf{x} and \mathbf{y} . C_1 and C_2 are small constants to make the division stable. To compare two entire images, the sliding window method is applied, where the SSIM index for each local window patch is calculated and the mean of all SSIM indices is considered as the final output. If two images are exactly the same, then the SSIM index would be 1. Therefore, SSIM loss for our problem can be formulated by:

$$\mathcal{L}_{\text{SSIM}} = 1 - \frac{1}{W} \sum_{i=1}^W \text{SSIM}(\mathbf{v}_i^p, \mathbf{v}_i), \quad (4)$$

where \mathbf{v}_i^p and \mathbf{v}_i represent the predicted velocity and the true velocity model inside i th window, respectively. W denotes the number of total sliding windows. To better incorporate image details at different resolutions, Wang *et al.* (2003) further developed multiscale SSIM (MS-SSIM):

$$\text{MS-SSIM}(\mathbf{x}, \mathbf{y}) = [l_M(\mathbf{x}, \mathbf{y})]^{\alpha_M} \cdot \prod_{j=1}^M [cs_j(\mathbf{x}, \mathbf{y})]^{\beta_j}. \quad (5)$$

In this metric, different scales of images are obtained by iteratively downsampling two input images by a factor of 2. $cs(\mathbf{x}, \mathbf{y})$ defined in eq. (3) is evaluated for each scale and then multiplied together, while only images at the last scale M are used to estimate $l(\mathbf{x}, \mathbf{y})$ for the final MS-SSIM value. Similarly, the MS-SSIM loss can be defined as:

$$\mathcal{L}_{\text{MS-SSIM}} = 1 - \frac{1}{W} \sum_{i=1}^W \text{MS-SSIM}(\mathbf{v}_i^p, \mathbf{v}_i). \quad (6)$$

To measure both pixel-by-pixel differences and local structural differences, in our method, we use the combination of the \mathcal{L}_1 loss and the MS-SSIM loss as the loss function to train the network:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_1 + \alpha \mathcal{L}_{\text{MS-SSIM}}. \quad (7)$$

An optimized parameter α can be found by searching from a range of values. In our experiment, we simply set α as 1 and compare the $\mathcal{L}_{\text{total}}$ loss with the traditional \mathcal{L}_1 loss and \mathcal{L}_2 loss.

3 EXPERIMENTS

3.1 Training data sets

Due to the lack of field data and the corresponding labels, synthetic velocity models and their corresponding seismic data are used to train the network. First, we follow the workflow of a model building method (Wu *et al.* 2020) to obtain numerous velocity models for preparing our training data sets. In this workflow, we first generate an initial velocity model with multiple flat layers (Fig. 3a), where the velocity values of the layers are randomly chosen from a predefined range of 1.5–5.0 km s⁻¹. When randomly choosing the velocity value for each layer, we make sure the velocity values of each layer maintain a general increasing trend, while high velocity layers are allowed to be above slow velocity layers. After generating such an initial velocity model, we deform the model by randomly adding some folding (Fig. 3b) and faulting (Fig. 3c) patterns to obtain a final velocity model with realistic structures. With this workflow, we can build a large number of 3-D velocity models, from which we can extract 2-D velocity models along random directions to generate our training data sets. Examples of velocity models are shown in Fig. 4.

After obtaining synthetic velocity models, we generate shot gathers with the acoustic rapid expansion method (REM, Pestana & Stoffa 2010), where spatial derivatives are computed with pseudospectral method. For each model, we simulate 256 shots each with a shot spacing of 40 m. All surface points are treated as receivers for each shot to simulate a full coverage seismic survey. Therefore, the receiver spacing is 20 m and the maximum offset is 10.24 km. Recording length of each shot gather is 4 s, with 2 ms sampling interval. The final output shot gathers are clean data without noise.

One crucial and time-consuming step in our workflow is to generate CIG volumes from the simulated shot gathers for these synthetic velocity models. Thus, an efficient migration method that is able to generate CIGs with minimal effort is desirable. Here, we employ the frequency-domain double plane wave (DPW) reverse time migration (RTM) method (Zhao *et al.* 2016b) to obtain migration images and CIG

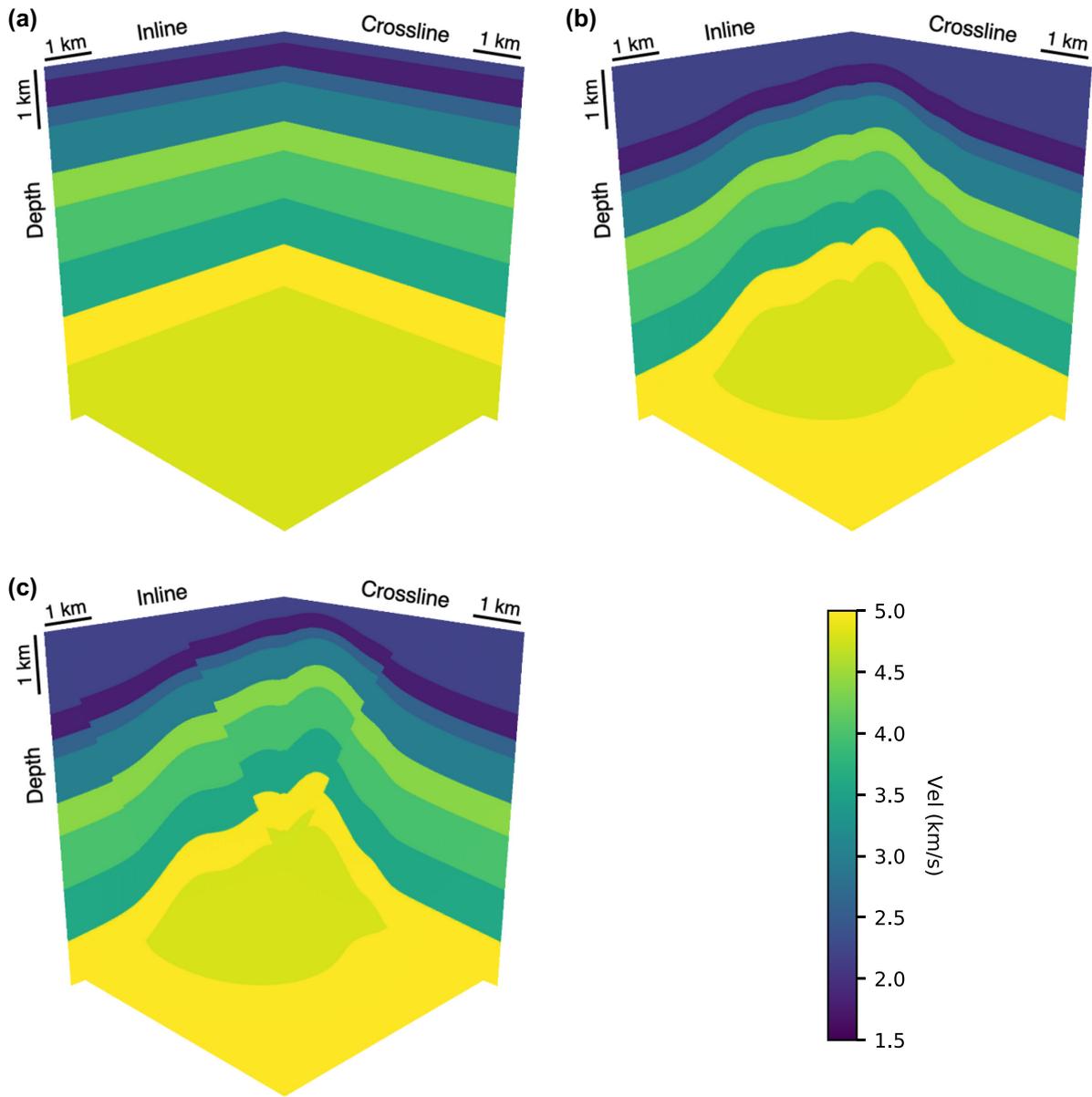


Figure 3. The workflow of automatically generating 3-D velocity models from an initial model (a) to the final velocity model (c) by adding folding (b) and faulting (c) structures.

volumes for the training and test data sets. We first transform time domain shot gathers for each velocity model into frequency domain DPW data set with DPW transform (Stoffa *et al.* 2006; Zhao *et al.* 2016b):

$$d(\mathbf{p}_s, \mathbf{p}_r, \omega) = \sum_s \sum_r d(s, \mathbf{r}, \omega) \exp(-i\omega [\mathbf{p}_s \cdot (\mathbf{s} - \mathbf{x}_{\text{ref}}) + \mathbf{p}_r \cdot (\mathbf{r} - \mathbf{x}_{\text{ref}})]), \quad (8)$$

where ω is angular frequency, and \mathbf{p}_s and \mathbf{p}_r are ray parameters for the source plane wave and the receiver plane wave, respectively. \mathbf{s} and \mathbf{r} denote source and receiver locations, respectively. \mathbf{x}_{ref} represents the reference point for the DPW transform. Then, migration is performed on each individual data set with DPW RTM method, in which the ray-parameter CIGs are generated via

$$I(\mathbf{x}, \mathbf{p}_r) = \Re \left[\sum_{\omega} \sum_{\mathbf{p}_s} \omega^2 f_s(\omega) G(\mathbf{p}_s, \mathbf{x}, \omega) G(\mathbf{p}_r, \mathbf{x}, \omega) d^*(\mathbf{p}_s, \mathbf{p}_r, \omega) \exp(i\omega (\mathbf{p}_s + \mathbf{p}_r) \cdot (\mathbf{x}_h - \mathbf{x}_{\text{ref}})) \right], \quad (9)$$

where $f_s(\omega)$ represents the source frequency signature, and $G(\mathbf{p}_s, \mathbf{x}, \omega)$ and $G(\mathbf{p}_r, \mathbf{x}, \omega)$ are plane wave Green's functions for a source plane wave \mathbf{p}_s and receiver plane wave \mathbf{p}_r , respectively. $d^*(\mathbf{p}_s, \mathbf{p}_r, \omega)$ represents the complex conjugate of the DPW data. \mathbf{x}_h is the horizontal

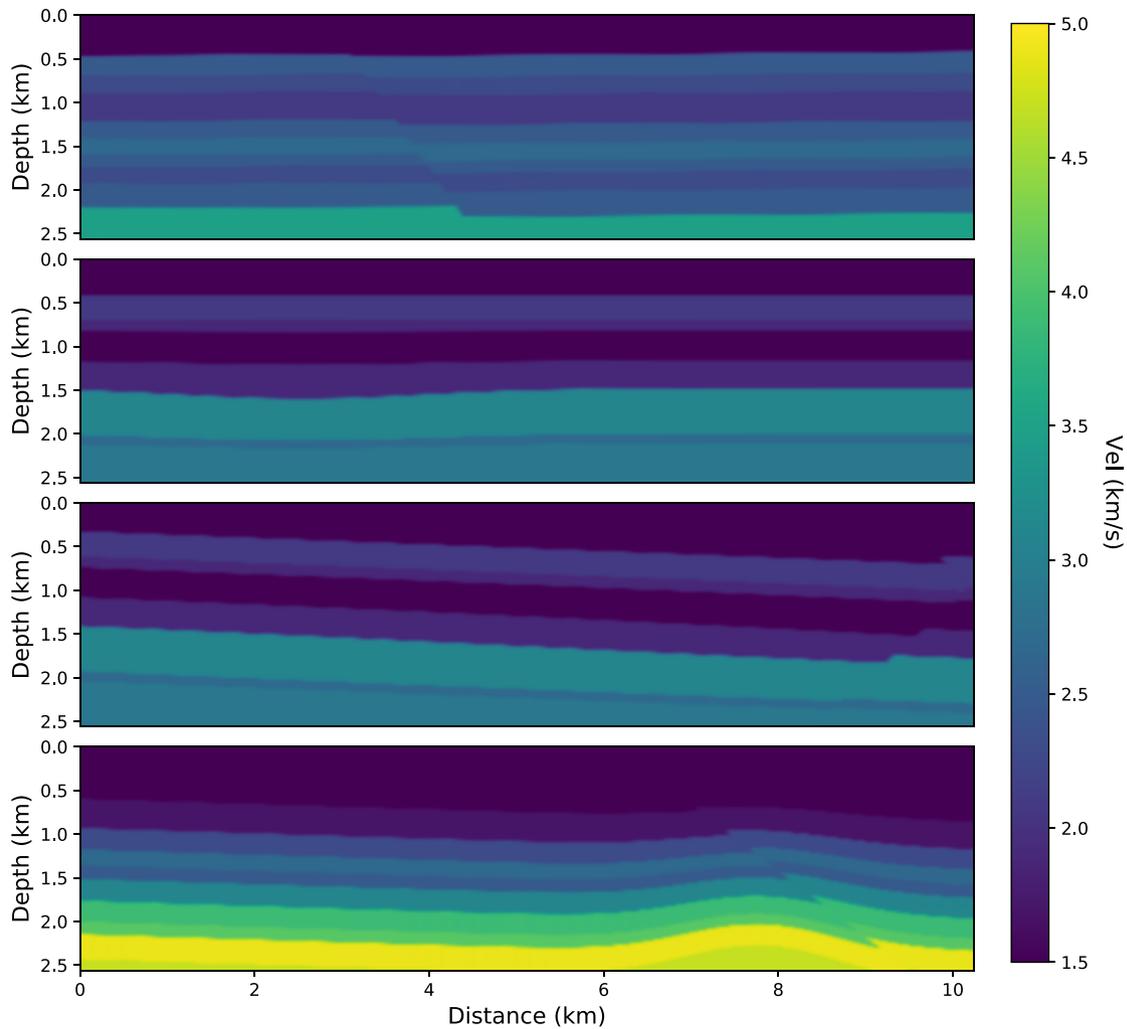


Figure 4. Selected synthetic velocity models with folding and faulting structures.

location of the image point \mathbf{x} . In the DPW RTM method, only a small number of plane-wave Green's functions need to be numerically calculated, which limits the number of the time-consuming wave field computation processes (Zhao *et al.* 2016b). In addition, ray-parameter CIGs can be readily generated by summing over migration profiles over p_s , which requires almost no additional computing cost. Therefore, the fast DPW RTM method greatly facilitates the data generation step. Note that the proposed workflow/architecture is not restricted by the method used to generate CIGs. One can also employ other methods, for example, methods proposed by Sava & Fomel (2003), or Xu *et al.* (2011) to generate CIGs for the training data set if a shot domain migration method and angle domain CIGs are preferred.

As previously mentioned, we use a reference velocity model as the migration velocity to generate input CIG volumes for the training data set. The reference velocity model is constant as 2 km s^{-1} . The reason why we use the same velocity for migration is that we want to provide a baseline for the network to learn the relationship. The migration velocity used here is just to generate CIGs and it can be any velocity values. Here, we use it as 2 km s^{-1} across different cases to simplify the data generation process. In addition, 2 km s^{-1} can be regarded as a relatively low velocity value. Therefore, all seismic events can be preserved in the CIGs. If a too fast velocity model is used, seismic events might be out of the deepest part for the model. Fig. 5 shows two sample CIG volumes generated from the same modelling data using the reference velocity model and the true velocity model, respectively. Using this incorrect migration velocity model, events in CIGs are not flat. In addition, imaging energy is positioned at incorrect spatial locations both horizontally and vertically. More examples are shown in Fig. 6. The left-hand column of Fig. 6 shows four CIGs in our training data set, the corresponding true velocity models are in Fig. 4. For better visualization, here, we only plot 15 gathers for each CIGs. For comparison, CIGs generated using the true velocity models are shown in the right-hand column of Fig. 6. As we can see, seismic images on the right-hand column have seismic energy positioned at the correct location and CIGs appear mostly flat, while the left-hand column contains images with wrong locations and curved seismic events. In our method, the network is asked to utilize these information contained in a CIG volume to estimate the subsurface velocity model. If the seismic events are under migrated, the network should be able to produce larger velocity values for the corresponding spatial location based on the baseline velocity. And the degrees of the curvatures guide the network how much larger velocity should be estimated. In addition, since the entire

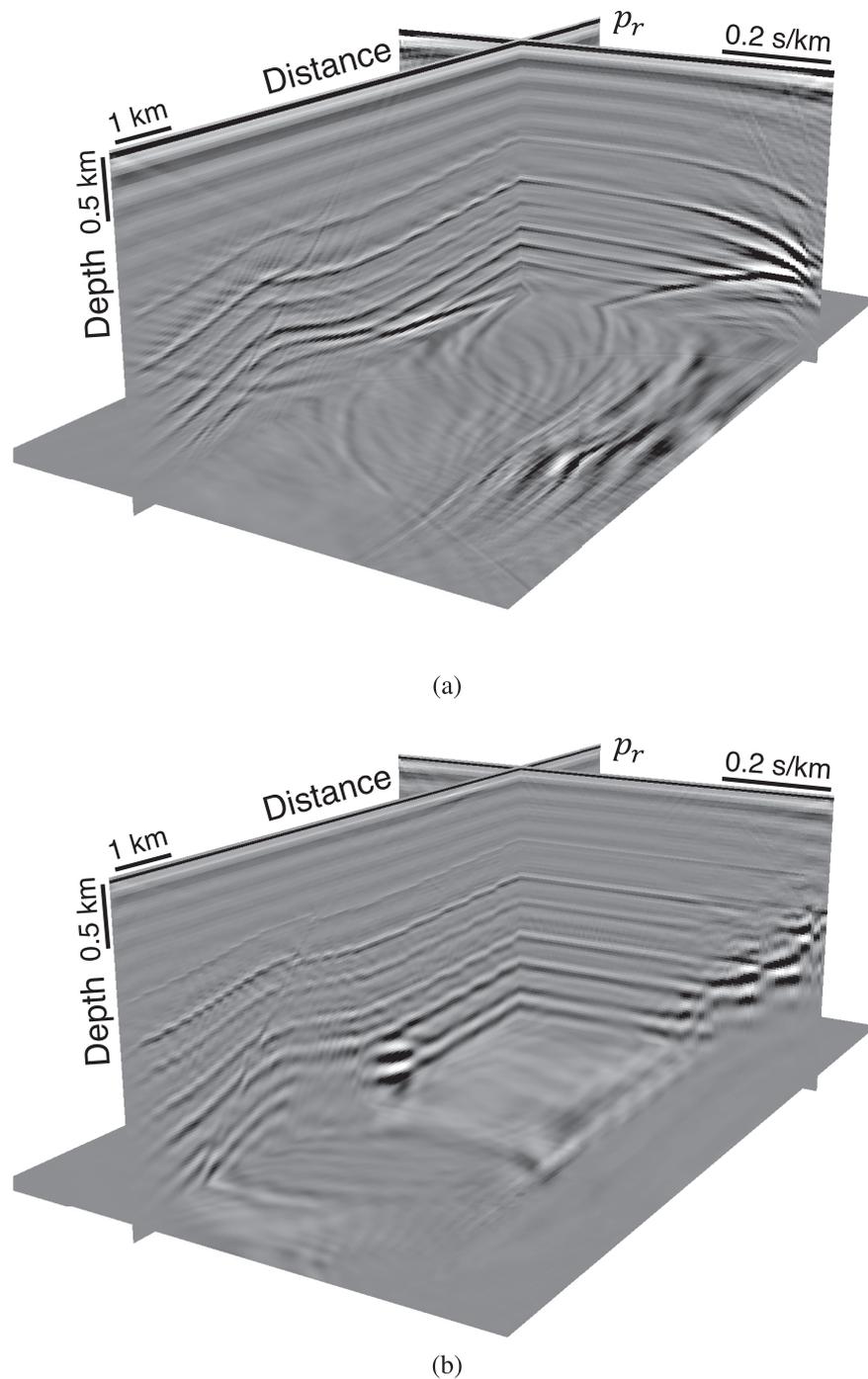


Figure 5. CIGs generated using (a) reference velocity and (b) true velocity.

CIG volume is used in the training, the network should be able to relate the seismic energy to its correct spatial location, both vertically and horizontally.

3.2 Implementation details

The neural network in our method is implemented using PyTorch (Paszke *et al.* 2019). To train the network, we generate 20 3-D seismic volumes in total, and extract 200 2-D velocity models from them. The previously described data generation workflow is applied on these velocity models to generate the training CIG volumes. We split the data set with 200 pairs of input CIG volumes and the corresponding true velocity models into three groups: training, validation and test data sets, in which there are 140, 30 and 30 pairs of data, respectively. Each CIG volumes is normalized by its mean and standard deviation when it is fed into the network. To improve the performance and generalization

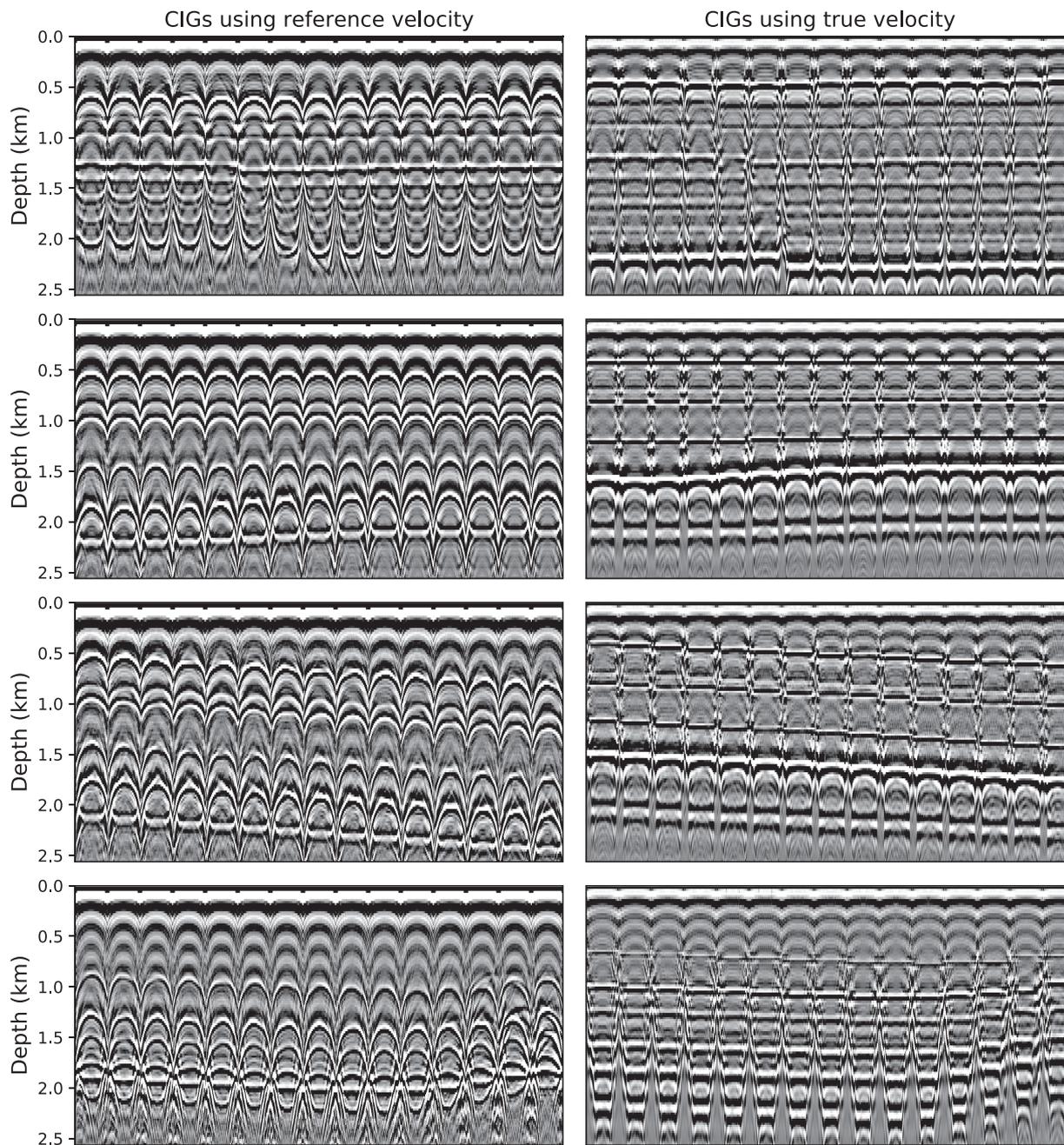


Figure 6. Left-hand column: CIGs generated using reference velocity. Right-hand column: CIGs generated using true velocity (Fig. 4).

ability of our network, we add random noise to CIGs, the level of which is randomly selected with signal-to-noise ratio in the range from 1 to 10.

The number of downsampling scales is set to 4 times and values for weight α_M and β_j in eq. (5) are borrowed from the original paper (Wang *et al.* 2003), which are $\beta_1 = 0.0448$, $\beta_2 = 0.2856$, $\beta_3 = 0.3001$, $\beta_4 = 0.2363$ and $\alpha_5 = \beta_5 = 0.1333$. After four times downsampling, the image size is 32×8 . Therefore, for computing the MS-SSIM loss, a window size of seven samples is chosen to compute the local mean and standard variance.

The velocity model building network is trained on two NVIDIA V100 GPUs, with a memory size of 16 GB. The Adam optimizer (Kingma & Ba 2014) is used to update the model parameters with the initial learning rate of 0.001 based on our experiments. We train the network for 200 epochs considering the training time and convergence and set the batch size as 20 due to the limited GPU memory size. The training time for 200 epochs under our training setup is about 4 hr. Although we train the network for 200 epochs, we only use the best model for comparison and the best model is chosen based on the metrics introduced in later sections using the validation data set. Empirically, we adopt

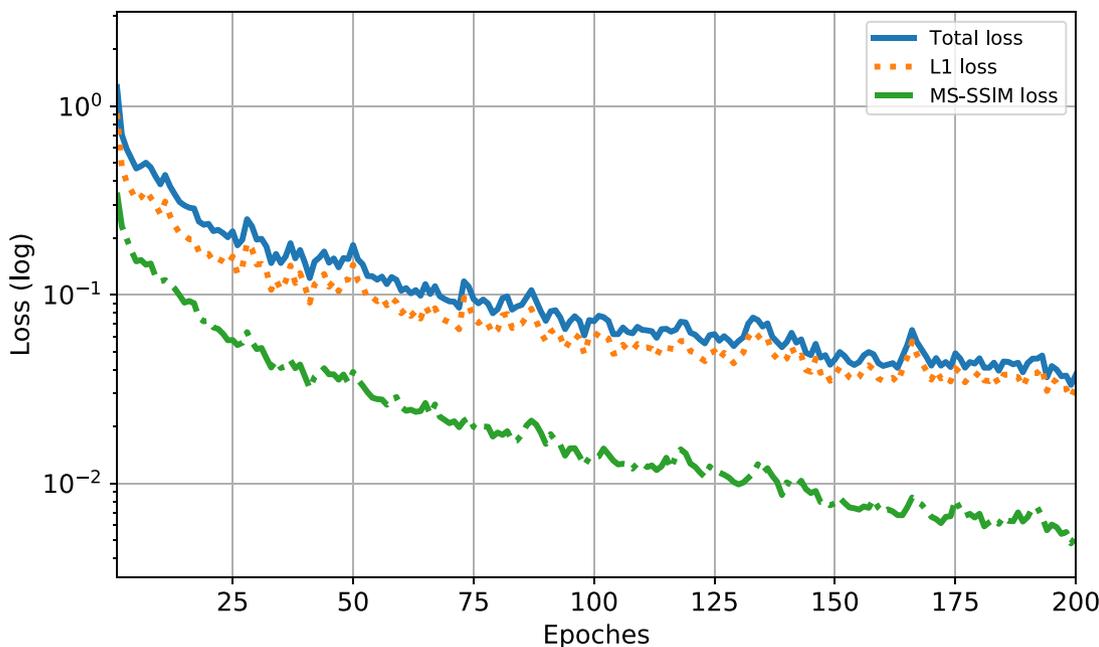


Figure 7. The history of total loss, \mathcal{L}_1 loss, and $\mathcal{L}_{\text{MS-SSIM}}$ in logarithmic scale.

the step learning rate scheduler by reducing learning rate to 90 per cent after every 10 epochs. The weight decay is set as 0.0001 for the Adam optimizer.

3.3 Results

The history of the total loss and two individual losses in logarithmic scale is shown in Fig. 7. Note that $\mathcal{L}_{\text{MS-SSIM}}$ converges faster than \mathcal{L}_1 loss, thus \mathcal{L}_1 loss mainly governs the training process. To illustrate the performance of the neural network on the training data set, we select six input CIG volumes from the training data set and apply the trained network to them. The comparison between the ground truth and predictions from the trained network of these selected CIG volumes is shown in Fig. 8. The neural network has a remarkable performance on the training data. The subsurface velocity models obtained from the network are mostly the same as the true velocity models. Even faults can be easily picked from the fourth and sixth output velocity models in Fig. 8. To better visualize the performance of our network on the training data, in Fig. 9 we plot the corresponding velocity profiles for the true and the estimated velocity models shown in Fig. 8. The horizontal location of these velocity profiles is at 6.36 km. As shown in Fig. 9, lines representing the velocity profiles of predicted velocity models are mostly identical to those of the true velocity models, which means that values of the output velocity models match that of the true velocity models very well.

As previously discussed, in our method, we use the combination of \mathcal{L}_1 and MS-SSIM loss as the loss function to train the network. Next, we will compare the performance of the neural network using different loss functions, including \mathcal{L}_1 , \mathcal{L}_2 and their combination with $\mathcal{L}_{\text{MS-SSIM}}$. We train neural networks using these four types of loss functions with the same training setting, the training times for all formulations of loss functions are almost the same since calculating the loss only contributes a little compared with the forwarding pass and the back propagation computation of the network. Fig. 10 compares the results obtained from the four types of training, the ground truth models are included as the reference. From the comparison in Fig. 10, it is easy to see that the network trained using $\mathcal{L}_1 + \mathcal{L}_{\text{MS-SSIM}}$ has the best performance. Compared with predictions from the other three neural networks, the estimated velocity models obtained with our proposed network better resemble the ground truth model. For instance, in the first test example, the fault structure is much shaper and clearly recovered. For the second row in Fig. 10, we do note that the fault in the true velocity model can not be fully identified by all four networks. The small strike of this fault might be the reason for the difficulty with the networks. If we only compare the \mathcal{L}_1 loss and \mathcal{L}_2 loss results without $\mathcal{L}_{\text{MS-SSIM}}$, the trained network using \mathcal{L}_1 loss still can generate better results than the one using the \mathcal{L}_2 loss. From the predicted velocity models in the third column of Fig. 10 (\mathcal{L}_1), we can see that velocity values in each layer of these velocity models are more consistent than the second column (\mathcal{L}_2). Therefore, generally speaking, \mathcal{L}_1 performs better than \mathcal{L}_2 in our problem and the combination of \mathcal{L}_1 and $\mathcal{L}_{\text{MS-SSIM}}$ has the best performance. Similar to what we have done for the training data, we compare velocity profiles of the true velocity models and predicted velocity models from four networks in Fig. 11. The comparison in Fig. 11 shows that the values from our proposed method match the true values the best. Another conclusion we can obtain from the comparison is that the mixed loss function of \mathcal{L}_n and $\mathcal{L}_{\text{MS-SSIM}}$ can boost the performance of the network compared to only using the \mathcal{L}_n loss. This is because $\mathcal{L}_{\text{MS-SSIM}}$ could help impose more penalty on local structures.

For better quantitative comparison, we evaluate the performance of the trained network with each kind of loss function on the test data set using five metrics: root mean square error (RMSE), MAE, mean relative percentage difference (MRPD), SSIM and MS-SSIM. We have

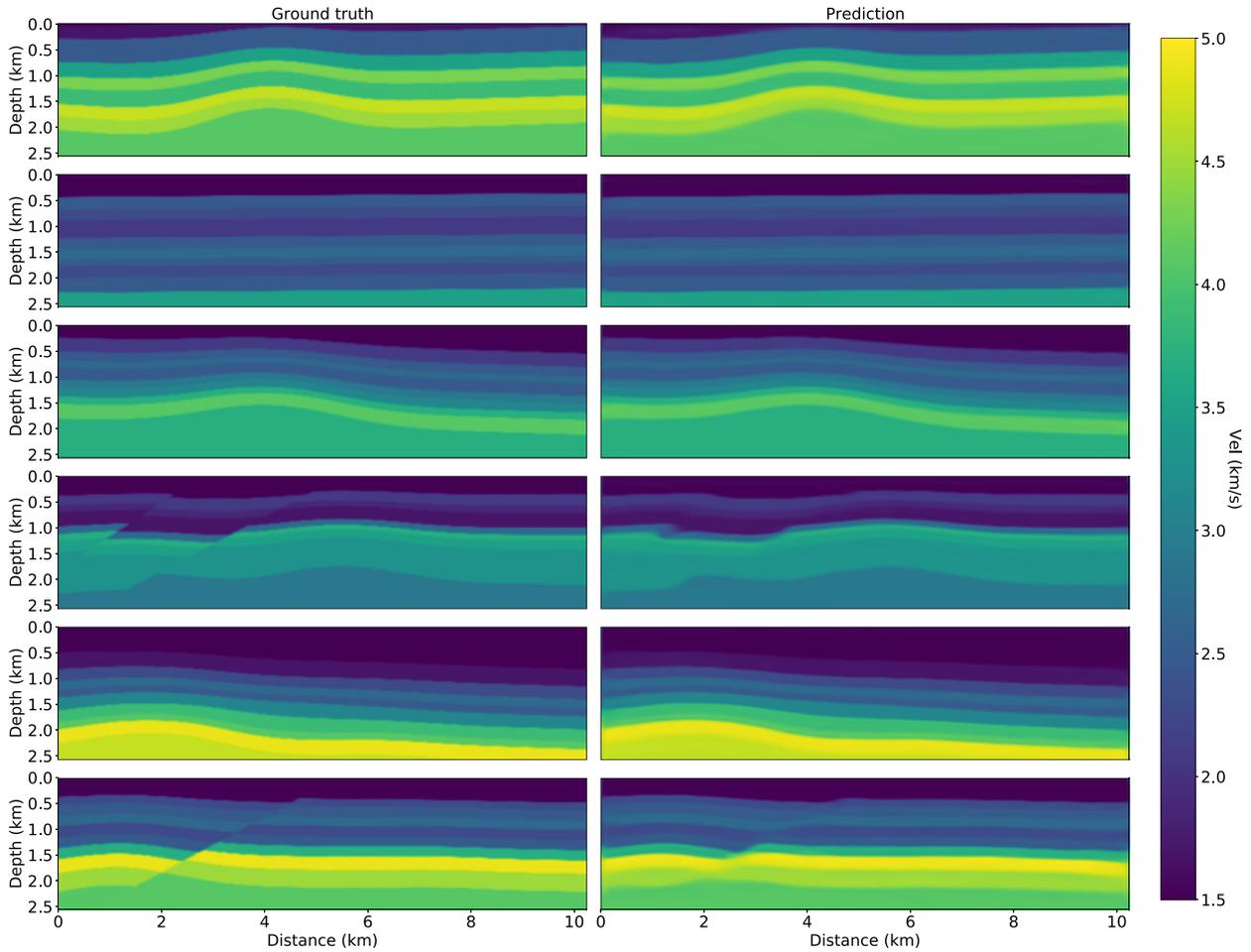


Figure 8. The comparison between the ground truth and the prediction from the trained network on six selected training input data.

already introduced MAE (eq. 1), SSIM (eq. 3) and MS-SSIM (eq. 5) in the previous section. RMSE is defined as the square root of MSE (eq. 2):

$$\mathcal{L}_2 = \sqrt{\frac{1}{N} \sum_{i=1}^N (v_i^p - v_i)^2}, \quad (10)$$

and MRPD is formulated as:

$$\text{MRPD} = \frac{2}{N} \sum_{i=1}^N \frac{|v_i^p - v_i|}{|v_i^p| + |v_i|}, \quad (11)$$

where N denotes the number of pixels in a single image. Table 2 shows the performance evaluation of four networks with different loss functions on the same test data set. Note that, for RMSE, MAE and MRPD metrics, lower score means better performance, while higher scores means better performance for SSIM and MS-SSIM. From Table 2, it is clear to see that the network training with the combination of \mathcal{L}_1 and $\mathcal{L}_{\text{MS-SSIM}}$ outperforms other networks on most metrics, which again shows that this loss function is more suitable for our task.

4 DISCUSSION

In this paper, we propose a novel and intuitive method of directly estimating velocity models from input CIG volumes by using a neural network. The flatness of seismic events along ray-parameter dimension and the spatial locations of seismic images provide information for subsurface velocity modelling building, which indicates that a CIG volume contains enough information for predicting subsurface velocity values. In addition, relationship between a CIG volume to a velocity model is a depth-to-depth transform, which avoids the domain conversion. Comparing with the deep learning methods that directly map data to model parameters, our new method does not involve dimension transform, which is closer to computer vision problems. Therefore, it is more reasonable to use the network to learn the relationship. One the other hand, directly mapping the data to model parameters is more close to an inverse problem, which needs much more efforts for researchers to develop advanced machine learning inverse theory and explain the behaviour of the network. We have shown that our method could produce reliable

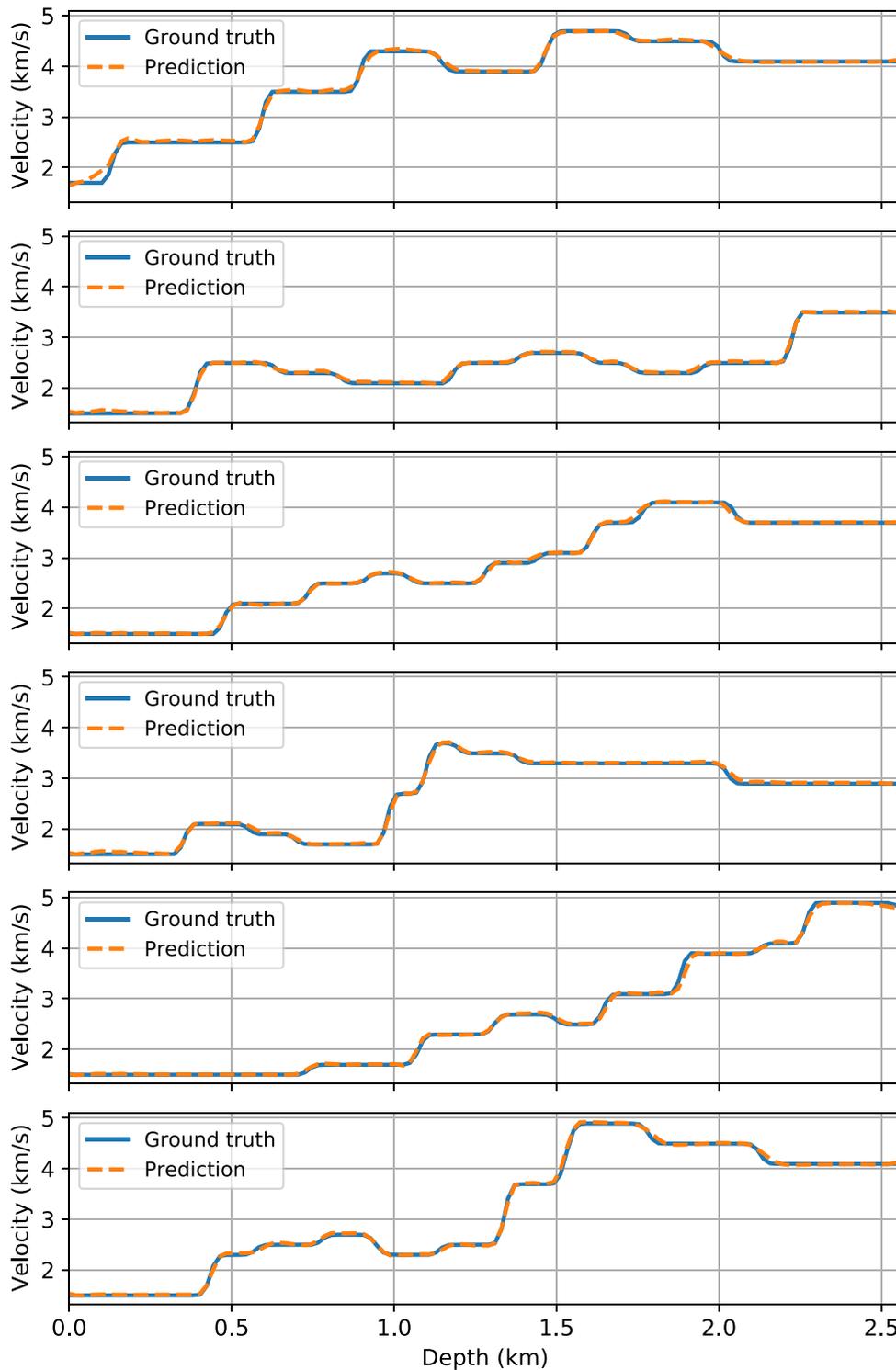


Figure 9. Velocity profiles of the ground truth and our prediction velocity models at 6.36 km.

results on test data. Field data, however, is much more complex than our training and test data set. More experiments might be necessary for applying the proposed method to more realistic problems. As demonstrated by the examples, even though not all features of models can be sharply recovered, the velocity values are in general accurately predicted. We believe that, with proper further investigation, the proposed method can be a powerful tool for the velocity building process.

One possible improvement for the proposed method is its generalization ability. In our method, we generate synthetic data for the training and testing, and some of them are close to each other due to the current limitation of the employed velocity model building method. Therefore, if the trained network has excellent performance on the training data, it can also obtain accurate velocity models from the test data set. In

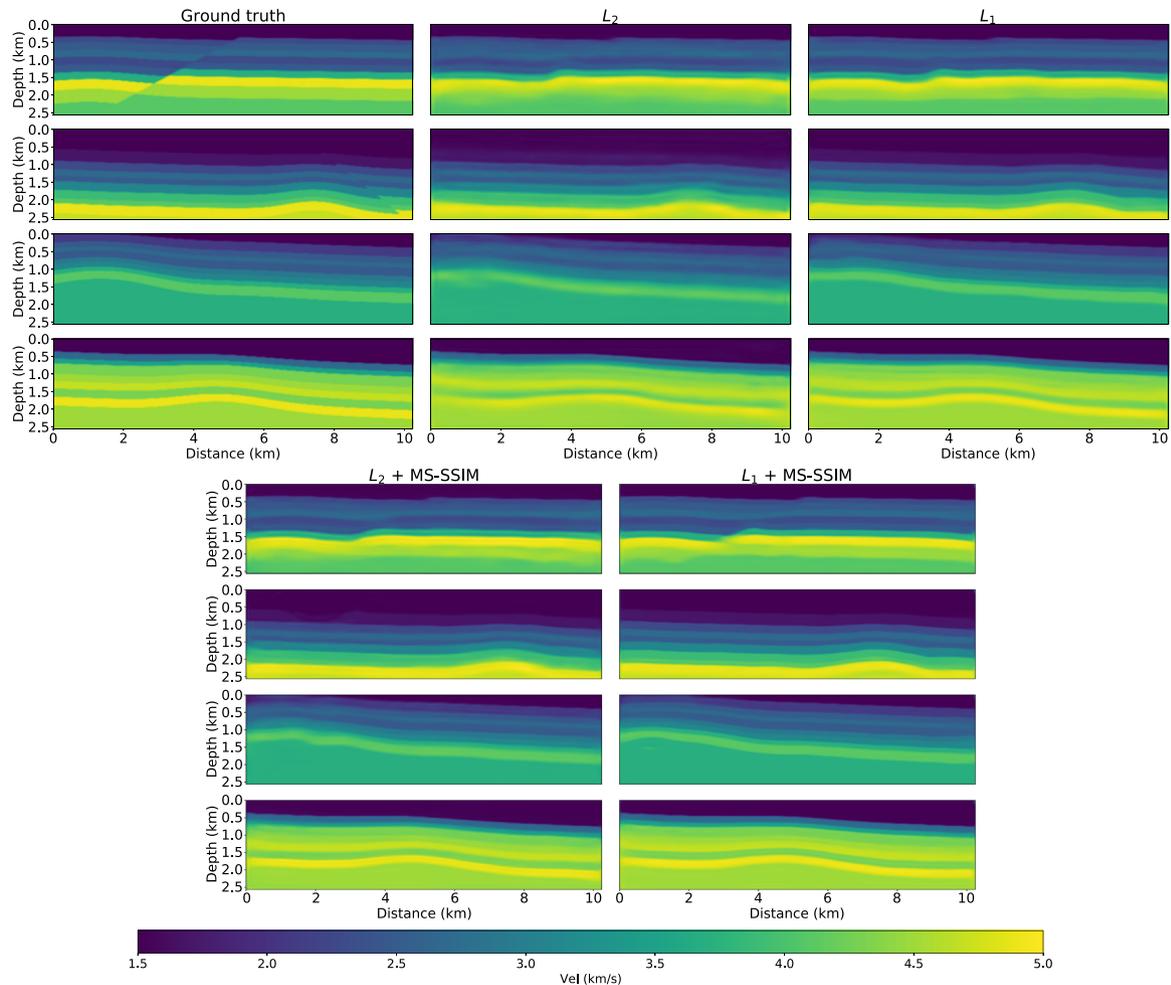


Figure 10. True velocity models and test results of four neural networks trained with \mathcal{L}_2 , \mathcal{L}_1 , $\mathcal{L}_2 + \mathcal{L}_{\text{MS-SSIM}}$ and $\mathcal{L}_1 + \mathcal{L}_{\text{MS-SSIM}}$.

theory, to successfully predict field data examples, the training data set should contain training samples that simulate subsurface structures that are similar to field data. The network can predict what it has seen or that is similar to what it has seen, which is a common limitation of deep learning in general. Therefore, to make deep learning generalized to field data set, one should build large enough training data set that simulated all realistic geological features, that is to say the training data set should contains samples across the entire parameter space. At this point, building realistic models that provide the network enough subsurface features is still one of the most challenging issue for all deep learning geophysical applications. Including field data into the training and test data sets could also help tackle this issue. However, the limited access to field data and the lack of true labels make this a difficult task for the academic research. Previous research on both computer vision and other geophysics fields has shown that good generalization ability of a neural network can be achieved by carefully handling the training process and training data set. Therefore, we believe that our method can be put into practice and provide reliable results with field data by further improvement including adding more realistic training data. Another possible solution to tackle the limited training data set issue is to use the transfer learning method. Transfer learning (Weiss *et al.* 2016) is a new surging method focusing on solving problems when the training data set and the test data set are not from the same domain. With transfer learning, one can deal with the scenarios where real data are much more different from the training data and their patterns are never seen by the trained mode by transferring the knowledge gained in the training data. It has been successfully applied in various geophysics applications (Naeini & Prindle 2018; Saad & Chen 2020; Zhang *et al.* 2021; Yu & Ma 2021). In the same manner, although we only have access to limited amount of synthetic data sets for training, we could use transfer learning technique to apply the proposed method to field data to overcome the limitation of training data sets. Further research are needed to successfully incorporate the transfer learning into the proposed workflow.

The dimension of the kernel in the network is also worth discussing. For 2-D examples, the input data is a 3-D cube, it seems more reasonable to use 3-D convolutional filters throughout the network. We tried this kind of 3-D CNN by replacing all the 2-D convolutional kernel in the current network architecture to 3-D and adding a $1 \times 1 \times 1$ convolutional layer at the end to transform 3-D feature maps to 2-D to obtain the final output velocity model. Compared with the 2-D network used in our method, however, this 3-D network, requiring much more memory and computation, can only obtain similar performance. Another network architecture, combining both 3-D and 2-D convolutional layers was also tested. The only difference between our 2-D network and this 2-D–3-D network is that three 3-D convolutional

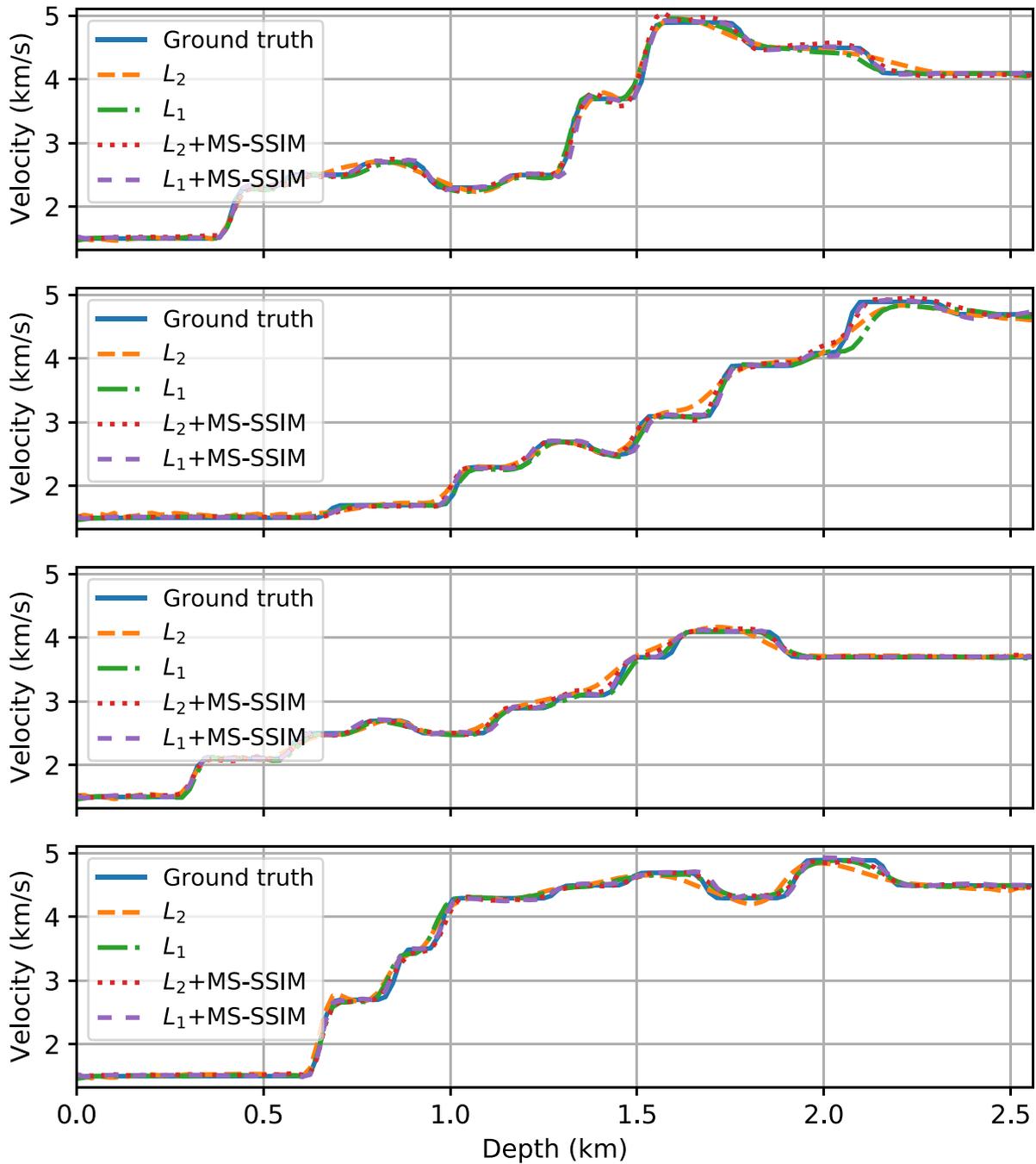


Figure 11. Velocity profiles of the ground truth and prediction velocity models from four networks at 6.36 km.

Table 2. Scores of neural networks using different loss functions on the test data set. For RMSE, MAE and MRPD metrics, lower score means better performance. For SSIM and MS-SSIM, higher score means better performance. The best and the second best results are highlighted in **bold** and *bold italic*.

Loss	RMSE	Lower is better		Higher is better	
		MAE	MRPD	SSIM	MS-SSIM
\mathcal{L}_1	0.089	0.036	0.075	0.920	0.980
\mathcal{L}_2	0.095	0.062	0.142	0.888	0.973
$\mathcal{L}_1 + \mathcal{L}_{MS-SSIM}$	0.075	<i>0.044</i>	<i>0.096</i>	0.946	0.988
$\mathcal{L}_2 + \mathcal{L}_{MS-SSIM}$	<i>0.082</i>	0.045	0.122	<i>0.921</i>	<i>0.986</i>

layers are added to the beginning. However, this network also did not outperform our 2-D network. The purpose of using 3-D convolutional layers is to capture information along the ray-parameter dimension with convolutional filters. The reason why a pure 2-D network can work in our problem is that a 2-D convolutional layer also can extract information along the channel (ray-parameter) dimension by using weighted sums or dot products instead of convolutional filter, which might be enough for our velocity model building task.

In our training setup, we choose ray-parameter CIGs as the input for the consideration of computation cost. We found ray-parameter CIGs can be easily obtained right after plane wave RTM without any extra computational cost, while angle domain CIGs generally requires higher computational cost due to the computation needed after the wave propagation. For instance, CIGs obtained with the extended imaging condition requires computing extra cross-correlations each time step, which can be costly for large-scale problems. However, as described in Section 2, the proposed velocity model building method is not limited to a specific type of migration method. Any method that can generate CIG volumes can be used as the training data, as long as the same migration method is used when applying the trained network on test data sets.

5 CONCLUSIONS

In this paper, we apply an encoder–decoder CNN to estimate subsurface velocity models directly and automatically from CIG volumes that are obtained using a reference velocity model. The proposed network is trained using a combined loss: $\mathcal{L}_1 + \mathcal{L}_{MS-SSIM}$, which proves to be suitable for our problem. A workflow from building velocity models to generating CIG volumes is proposed to generate the training data set for our proposed method. Good performance on the test data set shows a promising ability of our proposed method to help obtain accurate subsurface velocity models without large computation costs. Further improvements to our method can be made by adding more training data and extending it to 3-D. The extension of our proposed method to 3-D is straightforward by replacing 2-D convolutional filters with 3-D convolutional filters but requires additional computing resources.

ACKNOWLEDGEMENTS

We thank the editor Andrew Valentine, assistant editor Louise Alexander, reviewer Gareth O'Brien, and one anonymous reviewer for providing valuable suggestions. We thank the sponsors of the Texas Consortium for Computation Seismology (TCCS) and Institute for Geophysics for supporting the research. The Texas Advanced Computing Center (TACC) provided computational resources for this study.

DATA AVAILABILITY

The data underlying this article will be shared on reasonable request to the corresponding author.

REFERENCES

- Abadi, M. *et al.*, 2016. TensorFlow: Large-scale machine learning on heterogeneous distributed systems, preprint (arXiv:1603.04467).
- Al-Yahya, K., 1989. Velocity analysis by iterative profile migration, *Geophysics*, **54**(6), 718–729.
- Araya-Polo, M., Jennings, J., Adler, A. & Dahlke, T., 2018. Deep-learning tomography, *Leading Edge*, **37**(1), 58–66.
- Billette, F. & Lambaré, G., 1998. Velocity macro-model estimation from seismic reflection data by stereotomography, *Geophys. J. Int.*, **135**(2), 671–690.
- Biondi, B. & Sava, P., 1999. Wave-equation migration velocity analysis, in *SEG Technical Program Expanded Abstracts 1999*, pp. 1723–1726, Society of Exploration Geophysicists.
- Bishop, T. *et al.*, 1985. Tomographic determination of velocity and depth in laterally varying media, *Geophysics*, **50**(6), 903–923.
- Biswas, R., Arnulf, A.F., Sen, M.K., Datta, D., Zhao, Z., Mishra, P.K. & Jaysaval, P., 2020. Two-step velocity inversion using trans-dimensional tomography and elastic FWI, in *SEG Technical Program Expanded Abstracts 2020*, pp. 3628–3633, Society of Exploration Geophysicists.
- Bozdağ, E., Peter, D., Lefebvre, M., Komatitsch, D., Tromp, J., Hill, J., Podhorszki, N. & Pugmire, D., 2016. Global adjoint tomography: first-generation model, *Geophys. J. Int.*, **207**(3), 1739–1766.
- Bube, K.P. & Langan, R.T., 1999. On a continuation approach to regularization for crosswell tomography, in *SEG Technical Program Expanded Abstracts 1999*, pp. 1295–1298, Society of Exploration Geophysicists.
- Cerveny, V., 2001. *Seismic Ray Theory*, Cambridge University Press.
- Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K. & Yuille, A.L., 2017. Deeplab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs, *IEEE Trans. Pattern Anal. Mach. Intell.*, **40**(4), 834–848.
- Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F. & Adam, H., 2018. Encoder-decoder with atrous separable convolution for semantic image segmentation, in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 801–818, Springer.
- Clapp, R.G., Biondi, B.L. & Claerbout, J.F., 2004. Incorporating geologic information into reflection tomography, *Geophysics*, **69**(2), 533–546.
- Datta, D. & Sen, M.K., 2016. Estimating a starting model for full-waveform inversion using a global optimization method, *Geophysics*, **81**(4), R211–R223.
- Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. & Fei-Fei, L., 2009. ImageNet: a large-scale hierarchical image database, in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 248–255, IEEE Computer Society.
- Dines, K.A. & Lytle, R.J., 1979. Computerized geophysical tomography, *Proc. IEEE*, **67**(7), 1065–1073.
- Engquist, B. & Froese, B.D., 2013. Application of the Wasserstein metric to seismic signals, preprint (arXiv:1311.4581).
- Geng, Z., Wu, X., Shi, Y. & Fomel, S., 2020. Deep learning for relative geologic time and seismic horizons, *Geophysics*, **85**(4), WA87–WA100.
- Goodfellow, I., Bengio, Y. & Courville, A., 2016. *Deep Learning*, MIT press.
- He, K., Zhang, X., Ren, S. & Sun, J., 2016. Deep residual learning for image recognition, in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, IEEE Computer Society.
- He, K., Gkioxari, G., Dollár, P. & Girshick, R., 2017. Mask R-CNN, in *The IEEE International Conference on Computer Vision (ICCV)*, pp. 2961–2969, IEEE Computer Society.

- Isola, P., Zhu, J.-Y., Zhou, T. & Efros, A.A., 2017. Image-to-image translation with conditional adversarial networks, in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5967–5976, IEEE Computer Society.
- Jiao, J., Stoffa, P.L., Sen, M.K. & Seifoullaev, R.K., 2002. Residual migration-velocity analysis in the plane-wave domain, *Geophysics*, **67**(4), 1258–1269.
- Karras, T., Laine, S. & Aila, T., 2019. A style-based generator architecture for generative adversarial networks, in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4396–4405, IEEE Computer Society.
- Kingma, D.P. & Ba, J., 2014. Adam: a method for stochastic optimization, preprint (arXiv:1412.6980).
- Krizhevsky, A., Sutskever, I. & Hinton, G.E., 2012. ImageNet classification with deep convolutional neural networks, in *Advances in Neural Information Processing Systems 25*, pp. 1097–1105, Curran Associates.
- Lailly, P., 1983. The seismic inverse problem as a sequence of before stack migrations, in *Conference on Inverse Scattering: Theory and Application*, pp. 206–220, SIAM.
- Laina, I., Rupprecht, C., Belagiannis, V., Tombari, F. & Navab, N., 2016. Deeper depth prediction with fully convolutional residual networks, in *The IEEE International Conference on 3D Vision (3DV)*, pp. 239–248, IEEE Computer Society.
- LeCun, Y., Bengio, Y. & Hinton, G., 2015. Deep learning, *Nature*, **521**(7553), 436–444.
- Li, S., Ren, Y., Chen, Y., Yang, S., Wang, Y. & Jiang, P., 2020. Deep-learning inversion of seismic data, *IEEE Trans. Geosci. Remote Sens.*, **58**(3), 2135–2149.
- Lin, G., Milan, A., Shen, C. & Reid, I., 2017a. RefineNet: multi-path refinement networks for high-resolution semantic segmentation, in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1925–1934, IEEE Computer Society.
- Lin, T.-Y., Goyal, P., Girshick, R., He, K. & Dollár, P., 2017b. Focal loss for dense object detection, in *The IEEE International Conference on Computer Vision (ICCV)*, pp. 2980–2988, IEEE Computer Society.
- Liu, B., Yang, S., Xu, X., Ren, Y. & Jiang, P., 2019. Deep learning inversion of seismic data in tunnels, in *SEG 2019 Workshop: Mathematical Geophysics: Traditional vs Learning, Beijing, China, 5-7 November 2019*, pp. 52–55, Society of Exploration Geophysicists.
- Liu, Z. & Bleistein, N., 1995. Migration velocity analysis: Theory and an iterative algorithm, *Geophysics*, **60**(1), 142–153.
- Long, J., Shelhamer, E. & Darrell, T., 2015. Fully convolutional networks for semantic segmentation, in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3431–3440, IEEE Computer Society.
- Ma, Y. & Hale, D., 2013. Wave-equation reflection traveltime inversion with dynamic warping and full-waveform inversion, *Geophysics*, **78**(6), R223–R233.
- Mousavi, S.M., Ellsworth, W.L., Zhu, W., Chuang, L.Y. & Beroza, G.C., 2020. Earthquake transformer-an attentive deep-learning model for simultaneous earthquake detection and phase picking, *Nat. Commun.*, **11**(1), 1–12.
- Naeni, E.Z. & Prindle, K., 2018. Machine learning and learning from machines, *Leading Edge*, **37**(12), 886–893.
- Paszke, A. et al., 2019. PyTorch: an imperative style, high-performance deep learning library, in *Advances in Neural Information Processing Systems 32*, pp. 8024–8035, Curran Associates.
- Pestana, R.C. & Stoffa, P.L., 2010. Time evolution of the wave equation using rapid expansion method, *Geophysics*, **75**(4), T121–T131.
- Pratt, R.G., 1999. Seismic waveform inversion in the frequency domain, Part 1: theory and verification in a physical scale model, *Geophysics*, **64**(3), 888–901.
- Pratt, R.G., Shin, C. & Hick, G., 1998. Gauss–Newton and full Newton methods in frequency–space seismic waveform inversion, *Geophys. J. Int.*, **133**(2), 341–362.
- Ren, S., He, K., Girshick, R. & Sun, J., 2015. Faster R-CNN: towards real-time object detection with region proposal networks, in *Advances in Neural Information Processing Systems 28*, pp. 91–99, Curran Associates.
- Saad, O.M. & Chen, Y., 2020. Deep denoising autoencoder for seismic random noise attenuation, *Geophysics*, **85**(4), V367–V376.
- Sava, P. & Biondi, B., 2004. Wave-equation migration velocity analysis. I. Theory, *Geophys. Prospect.*, **52**(6), 593–606.
- Sava, P.C. & Fomel, S., 2003. Angle-domain common-image gathers by wavefield continuation methods, *Geophysics*, **68**(3), 1065–1074.
- Schmidhuber, J., 2015. Deep learning in neural networks: an overview, *Neural Netw.*, **61**, 85–117.
- Sen, M.K. & Stoffa, P.L., 1991. Nonlinear one-dimensional seismic waveform inversion using simulated annealing, *Geophysics*, **56**(10), 1624–1638.
- Shen, P. & Symes, W.W., 2008. Automatic velocity analysis via shot profile migration, *Geophysics*, **73**(5), VE49–VE59.
- Shi, Y., Wu, X. & Fomel, S., 2019. SaltSeg: automatic 3D salt segmentation using a deep convolutional neural network, *Interpretation*, **7**(3), SE113–SE122.
- Stoffa, P.L. & Sen, M.K., 1991. Nonlinear multiparameter optimization using genetic algorithms: inversion of plane-wave seismograms, *Geophysics*, **56**(11), 1794–1810.
- Stoffa, P.L., Sen, M.K., Seifoullaev, R.K., Pestana, R.C. & Fokkema, J.T., 2006. Plane-wave depth migration, *Geophysics*, **71**(6), S261–S272.
- Stork, C., 1992. Reflection tomography in the postmigrated domain, *Geophysics*, **57**(5), 680–692.
- Szegedy, C. et al., 2015. Going deeper with convolutions, in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, IEEE Computer Society.
- Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. & Wojna, Z., 2016. Re-thinking the inception architecture for computer vision, in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2826, IEEE Computer Society.
- Tarantola, A., 1984. Inversion of seismic reflection data in the acoustic approximation, *Geophysics*, **49**(8), 1259–1266.
- Tarantola, A., Crase, E., Jervis, M., Konen, Z., Lindgren, J., Mosegaard, K. & Noble, M., 1990. Nonlinear inversion of seismograms: state of the art, in *SEG Technical Program Expanded Abstracts 1990*, pp. 1193–1198, Society of Exploration Geophysicists.
- Varela, C.L., Stoffa, P.L. & Sen, M.K., 2002. Background velocity estimation using non-linear optimization for reflection tomography and migration misfit, *Geophys. Prospect.*, **46**(1), 51–78.
- Virieux, J. & Operto, S., 2009. An overview of full-waveform inversion in exploration geophysics, *Geophysics*, **74**(6), WCC1–WCC26.
- Wang, W. & Ma, J., 2020. Velocity model building in a cross-well acquisition geometry with image-trained artificial neural networks, *Geophysics*, **85**(2), U31–U46.
- Wang, Z., Simoncelli, E.P. & Bovik, A.C., 2003. Multiscale structural similarity for image quality assessment, in *The Thirty-Seventh Asilomar Conference on Signals, Systems Computers*, vol. 2, pp. 1398–1402, IEEE.
- Wang, Z., Bovik, A.C., Sheikh, H.R. & Simoncelli, E.P., 2004. Image quality assessment: from error visibility to structural similarity, *IEEE Trans. Image Process.*, **13**(4), 600–612.
- Warner, M. & Guasch, L., 2016. Adaptive waveform inversion: theory, *Geophysics*, **81**(6), R429–R445.
- Weiss, K., Khoshgoftaar, T.M. & Wang, D., 2016. A survey of transfer learning, *J. Big data*, **3**(1), 1–40.
- Wu, X., Liang, L., Shi, Y. & Fomel, S., 2019a. FaultSeg3D: using synthetic data sets to train an end-to-end convolutional neural network for 3D seismic fault segmentation, *Geophysics*, **84**(3), IM35–IM45.
- Wu, X., Liang, L., Shi, Y., Geng, Z. & Fomel, S., 2019b. Multitask learning for local seismic image processing: fault detection, structure-oriented smoothing with edge-preserving, and seismic normal estimation by using a single convolutional neural network, *Geophys. J. Int.*, **219**(3), 2097–2109.
- Wu, X., Geng, Z., Shi, Y., Pham, N., Fomel, S. & Caumon, G., 2020. Building realistic structure models to train convolutional neural networks for seismic structural interpretation, *Geophysics*, **85**(4), WA27–WA39.
- Xu, S., Zhang, Y. & Tang, B., 2011. 3D angle gathers from reverse time migration, *Geophysics*, **76**(2), S77–S92.

- Yang, F. & Ma, J., 2019. Deep-learning inversion: a next-generation seismic velocity model building method, *Geophysics*, **84**(4), R583–R599.
- Yang, Y., Engquist, B., Sun, J. & Hamfeldt, B.F., 2018. Application of optimal transport and the quadratic Wasserstein metric to full-waveform inversion, *Geophysics*, **83**(1), R43–R62.
- Yilmaz, Ö., 2001. *Seismic Data Analysis: Processing, Inversion, and Interpretation of Seismic Data*, Society of Exploration Geophysicists.
- Yu, S. & Ma, J., 2021. Deep learning for geophysics: current and future trends, *Rev. Geophys.*, **59**(3), e2021RG000742.
- Zhang, J., Li, J., Chen, X., Li, Y., Huang, G. & Chen, Y., 2021. Robust deep learning seismic inversion with a priori initial model constraint, *Geophys. J. Int.*, **225**(3), 2001–2019.
- Zhang, X., Zou, J., He, K. & Sun, J., 2015. Accelerating very deep convolutional networks for classification and detection, *IEEE Trans. Pattern Anal. Mach. Intell.*, **38**(10), 1943–1955.
- Zhao, H., Gallo, O., Frosio, I. & Kautz, J., 2016a. Loss functions for image restoration with neural networks, *IEEE Trans. Comput. Imag.*, **3**(1), 47–57.
- Zhao, Z. & Sen, M.K., 2021a. A gradient-based Markov chain Monte Carlo method for full-waveform inversion and uncertainty analysis, *Geophysics*, **86**(1), R15–R30.
- Zhao, Z. & Sen, M.K., 2021b. A hybrid optimization method for full-waveform inversion, in *First International Meeting for Applied Geoscience & Energy Expanded Abstracts*, pp. 767–771, Society of Exploration Geophysicists.
- Zhao, Z., Sen, M.K. & Stoffa, P.L., 2016b. Double-plane-wave reverse time migration in the frequency domain, *Geophysics*, **81**(5), S367–S382.
- Zhu, H. & Fomel, S., 2016. Building good starting models for full-waveform inversion using adaptive matching filtering misfit, *Geophysics*, **81**(5), U61–U72.
- Zhu, J.-Y., Park, T., Isola, P. & Efros, A.A., 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks, in *The IEEE International Conference on Computer Vision (ICCV)*, pp. 2242–2251, IEEE Computer Society.
- Zhu, W. & Beroza, G.C., 2019. PhaseNet: a deep-neural-network-based seismic arrival-time picking method, *Geophys. J. Int.*, **216**(1), 261–273.
- Zhu, W., Mousavi, S.M. & Beroza, G.C., 2019. Seismic signal denoising and decomposition using deep neural networks, *IEEE Trans. Geosci. Remote Sens.*, **57**(11), 9476–9488.