RNN-based Seismic Velocity Model Building: Improving Generalization using Hybrid Training Data

Hani Alzahrani & Jeffrey Shragge

Center for Wave Phenomena, Department of Geophysics, Colorado School of Mines, Golden CO 80401
Corresponding Author Email: halzahrani@mines.edu

ABSTRACT

Neural networks are emerging as an alternative approach for seismic velocity model building that do not inherently suffer from the challenges associated with deterministic methods. However, they have their own set of drawbacks, chief among which is a difficulty of network generalization. For seismic velocity model building, constructing training models that closely resemble the local geology can be as challenging as developing a sufficient starting model for full waveform inversion (FWI). We present a multi-scale FWI-inspired approach that uses recurrent neural networks to invert frequency-domain seismic data using a frequency-stepping scheme. We combine this approach with a hybrid training data set that consists of geologically realistic and purely geometrical models to significantly improve the network generalization. We demonstrate this by testing the proposed method on subsets from two seismic benchmark models and showing that the trained hybrid network is capable of constructing complex salt bodies that are not incorporated in the training models.

1 INTRODUCTION

Seismic velocity model building is one of the most challenging tasks in seismic exploration. Traditionally, the problem has been approached using deterministic methods that either use a subset of the recorded seismic waveform (e.g., traveltime tomography) or the full waveform (e.g., full waveform inversion (FWI)). FWI aims to build a high-resolution subsurface velocity model that minimizes the misfit between observed and modeled seismic data (Lailly, 1983; Tarantola, 1984). The greatest challenge to FWI is the fact that the wave equation is non-linear with respect to model parameters. This leads to a non-linear objective function with numerous local minima. Such challenges are exacerbated by the oscillatory nature of seismic data, which demands using starting models that, when used in seismic forward modeling, can simulate waveforms to within half a period of recorded data and thereby avoid cycle skipping that can cause the inversion to converge toward a local minimum that may be far away from the true global solution. Finally, FWI is computationally expensive because each iteration usually requires (tens of) thousands of seismic wave simulations.

A number of approaches have been proposed in the past decades to overcome the non-linearity challenges associated with the seismic velocity inversion problem. In FWI, many approaches follow a multi-scale strategy (Bunks et al., 1995; Sirgue and Pratt, 2004; Brenders and Pratt, 2007; Sirgue et al., 2010) that first inverts lower-frequency data to promote the recovery of smoother velocity models (i.e., longer wavelength structure) that ideally moves the optimization search direction toward the global minimum. The updated smooth velocity models are then used as input to invert higher-frequency data to increase model resolution.

The recent resurgence of neural networks, though, is introducing new paradigms for automated velocity-model building. These methods aim to take advantage of artificial neural networks (ANNs) to bypass challenges associated with deterministic model building methods. ANNs offer the potential of improved efficiency because the main cost overhead comes from ANN training, after which model predictions are made at the cost of matrix-vector multiplication.

There have been numerous successful applications of ANNs for solving the seismic velocity model building problem, including inverting common mid-point (CMP) semblance cubes (Araya-Polo et al., 2018) and using different neural networks architectures to invert seismic data (Phan and Sen, 2018; Zhang and Alkhalifah, 2019; Sun et al., 2020; Fabien-Ouelle and Sarkar, 2020; He and Wang, 2021). A common feature in these approaches is the use of time-domain shot gathers for network input, which requires all time-domain data be simultaneously input into the network.

In this work, we propose a novel frequency-domain ANN approach that uses a recurrent neural network (RNN) to feed frequency slices incrementally (rather than in its entirety) from the lowest available to highest usable. While this approach reduces the amount of data needed to estimate the underlying velocity model and produces improved results compared to other frequency-domain network architectures, it still suffers from the common challenge of generalization. In this context, generalization means the following: if a network is trained on a training data set and then successfully tested on data drawn from the same distribution but not used in the training data set, the network can be said to have generalized well. This limited definition of generalization, though, is not very useful in the context of seismic inversion because in most cases one does not have the required local geological information to construct models sufficiently representative of those being estimated.

To help minimize generalization issues, we present a novel training approach that combines a FWI-inspired frequency-stepping approach with a hybrid velocity model training scheme based on two velocity distribution classes: geological and purely geometrical. We show that compiling and training ANNs with such a hybrid data set leads to improved model estimation and a corresponding improvement in network generalizability. We begin by describing how each class of training models is constructed. We then describe the neural network architecture that we use in this work, and explain the rationale behind our choice. Next, we briefly describe the training process and show testing results comparing two neural networks; one is trained using geological models only, the other using the proposed hybrid data. We show that using hybrid training data enriches the training process and allows the network to recover a wider range of velocity structures than using geological training data only.

2 GENERATING TRAINING DATA

To demonstrate the benefits of combining different velocity model classes in the network training, we first specify the procedures used to generate models of the two distinct classes. We construct geologically realistic models (Figure 1c) from top to bottom in a number of steps. First, the number of model layers is randomly specified within a certain range. The interface boundary between the first and second layers is constructed through linear interpolation between control points, the number of which as well as their lateral and depth locations are randomly specified within a range of accepted values. The interface is also permitted to follow the topology of any geological structure. We then apply 1-D Gaussian smoothing to the resulting interpolated interface. The next interface is similarly constructed by linearly interpolating between control points. The horizontal locations of these points are the same as those on the first layer, while their depth is constrained to keep the layer thickness within a certain range. These two constraints prevent the second layer from developing completely different structure than the overlying layer. These steps are repeated for all subsequent layers. After constructing all layer interfaces, velocity values are randomly assigned to each layer. Following this, a randomly generated number specifies whether the current model should include faults as well as the vector offset such a fault should exhibit. The fault line is drawn by randomly specifying and interpolating between two points, located at the top and bottom boundaries, with the layers shifted by a randomly specified offset.

The second training model class is purely geometrical (Figure 1b). Alzahrani and Shragge (2022) demonstrates that purely geometrical training models are not biased toward specific geologic structures, which makes them useful in situations where geological structures in the testing models cannot be predicted. Instead of using squares as in Alzahrani and Shragge (2022), here we construct our purely geometrical models using a more elaborate process that involves a sequence of random numbers. We start by randomly specifying the number of squares in each direction. To understand how velocities are assigned to each square, imagine an ant is dropped on one square and allowed to randomly walk until reaching the domain boundary. A randomly selected velocity is then assigned to each square on which it stepped. The same process is repeated until all squares have been assigned a velocity, each time starting from a square not yet assigned a velocity value. This process ensures that the resulting velocity model consists of continuous structures, and not just randomly scattered velocity values. Another sequence of random processes determines the degree and orientation of the applied model smoothing operators to ensure the structures at different scales point in varying directions (see examples in Figure 1b). A subset of purely geometrical models includes models with a constant velocity value, as well as models with a either a vertical or a diagonal velocity gradient. Whether the velocity increases or decreases with depth is randomly specified. The background velocity value as well as the starting and ending velocity values for the velocity gradient are randomly specified (Figure 1a)

We generated $2,000$ models for each velocity class. The three training model sets were then combined to form the $6,000$ training models used for network training. Each training batch had one third of each class sorted in a random order. The models have the size of 125×400 grid points with a uniform 10.0 m model discretization. We simulated seismic data by solving the 2-D Helmholtz equation using a 15 Hz Ricker wavelet. Data were generated for six frequencies ranging from 1 Hz to 11 Hz in 2 Hz increments. Sources and receivers were respectively located three and six grid points below the free surface.

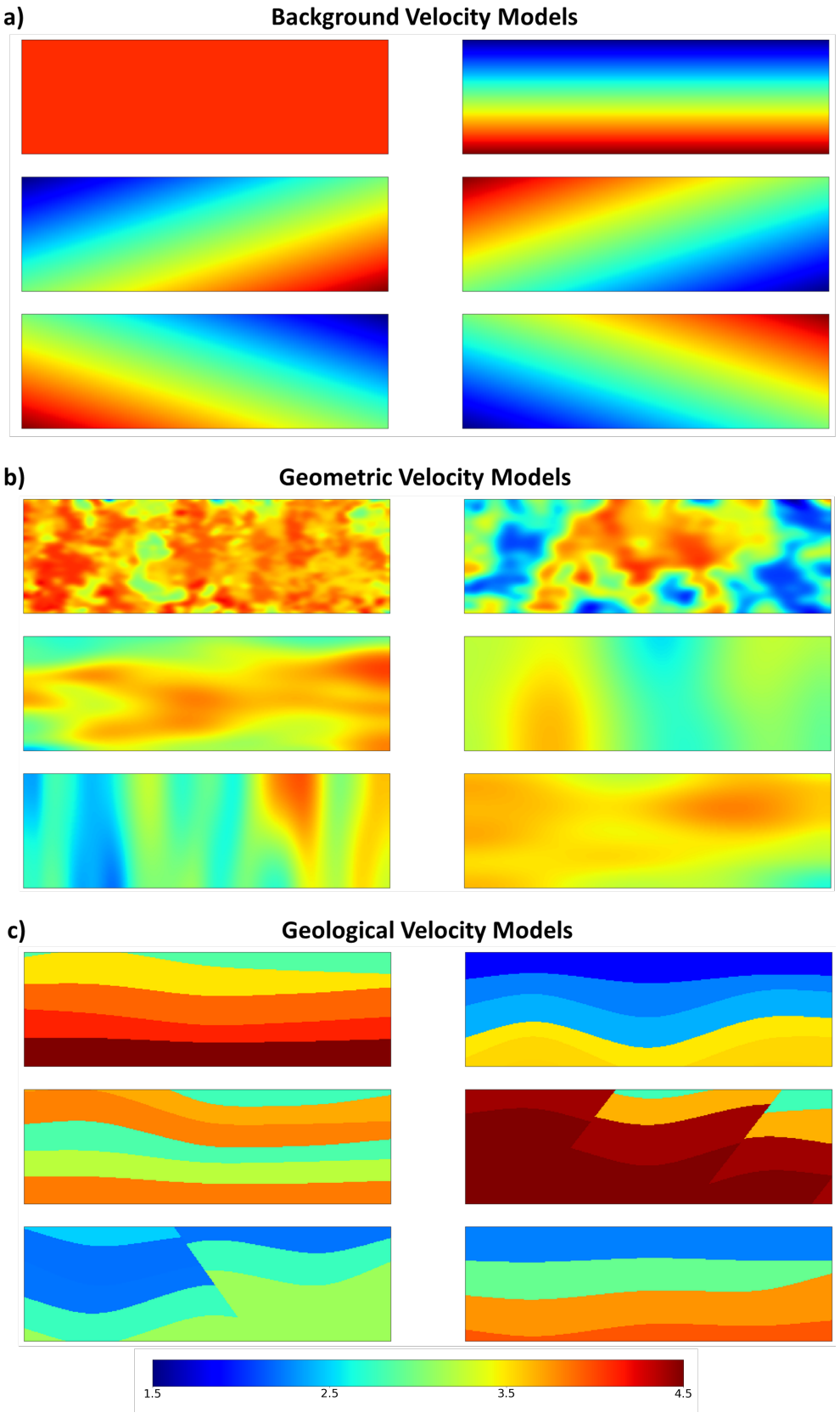

Figure 1. Representative 1.25 km×4.0 km examples of drawn from the three training model classes. (a) Background velocity models with either a constant velocity value or a velocity gradient in the vertical or the diagonal direction. (b) Purely geometrical models at different scales with the generated structures oriented in different directions. (a) Geological models showing layered faulted structures.

3 NEURAL NETWORK ARCHITECTURE

Convolutional neural networks (CNNs) have been used to successfully invert time-domain seismic data. However, when using CNNs to invert frequency-domain data, the natural procedure would be to feed each frequency slice to a different channel of the input layer. A problem with this approach is a lack of communication between different frequency slices, meaning that one would effectively train independent connections between each frequency slice and the velocity model. Moreover, such an approach does not exploit the multi-scale FWI strategy where one first constructs a smooth velocity model from lower-frequency data and then adds in finer details from higher-frequency data.

RNNs are one class of ANNs that allows information flow from prior input stages. These architectures incorporate a hidden-state vector that acts as network memory and carries information 'learned' from one input to the next. Thus, this class of ANNs is highly prospective for exploiting the FWI multi-scale approach. RNNs can be classified into two categories based on their type of constituent layers: conventional and convolutional (Figure 2). Because RNNs handle two vectors/feature maps at each step, an RNN layer actually consists of two layers: one handling the input data, the other the hidden-state vector. As illustrated in Figure 2, the only difference between conventional and convolutional RNNs is that in latter case the two RNN layers are convolutional, while the former uses fully connected layers.

Convolutional RNNs combine the advantages of CNNs (including a significant reduction in required memory) that have made them successful in a wide range of applications, while simultaneously benefiting from including a hidden-state feature map to propagate information from the initial to the final input. RNNs may also be classified based on how they propagate information between steps. Among these different classifications, long short-term memory (LSTM) networks are the most suitable for the present application owing to their stability for long sequences. This is because RNNs suffer from the vanishing or exploding gradient problem, which makes them incapable of carrying forward information learned for long sequences (Hochreiter and Schmidhuber, 1997). The LSTM network circumvents these issues by incorporating four layers (each with two constituent layers for eight in total) that propagate information between steps as well as two memory vectors. This is in contrast with the two layers and one memory vector found in conventional RNNs. Among the four layers constituting an LSTM cell, the forget layer performs a crucial task in that it acts as a gate for the information flowing from one step to the next. Because the network has limited memory, it needs to discard information that the layer believes are unnecessary. This allows relevant information to reach the end of the sequence.

We use a convolutional LSTM neural network architecture that we henceforth refer to as ConvLSTM. More specifically, we use a sequence-to-sequence ConvLSTM, in which each input is constrained by an output with the connection between the two guiding the training process (Figure 3). Following the logic of multi-scale FWI, we inject the lowest usable data frequency at the first step. Ideally, the output velocity model should have wavenumber components that are sensitive to at most the injected frequency. However, because constructing such model is not straightforward for heterogeneous media, we approximate this process by applying 2-D Gaussian smoothing at each step with the operator width dependent on the input frequency. We use a wider operator for lower frequencies to relatively upweight the wavenumber components sensitive to model structure at the current frequency. After injecting the smooth velocity model as the output for the first frequency, we proceed by following the same process for higher frequencies. For a more detailed description of the network dynamics, we refer readers to Alzahrani and Shragge (2021).

The network used in this work consists of three ConvLSTM layers (Figure 4). The input frequency slice contains data from all sources and receivers corresponding to a single frequency arranged in a 2D array with 76 elements along the y-axis representing twice the number of sources used in the experiment (since there is a real and an imaginary component for each source) and 400 elements along the x-axis representing the number of receivers. Because the network is a convolutional LSTM, each layer has the same hyper-parameters as a typical convolutional layer. It is only differentiated from a typical convolutional layer by a number of parameters that control the recurrence process. Thus, the two parameters controlling layer behavior are the number of 2D convolutional operators and their sizes. Figure 4 shows our choice of these two parameters for each layer. We found that applying convolutions with a stride larger than one produces more stable results than using pooling layers. To downscale the input data along the receiver axis, we use stride values of 6, 4, and 2 for the layers ConvLSTM 1, ConvLSTM 2, and ConvLSTM 3, respectively. These values strike a balance between model resolution and computational efficiency. Each ConvLSTM layer is followed by a batch normalization layer to reduce overfitting and improve network generalization. The 2-D feature maps corresponding to ConvLSTM 3 are flattened into a fully connected layer, which in is connected to the output velocity layer.

Training the ANN involves inputting data to the network, performing a forward run to compute the outputs, and using the true and predicted output difference to update the network weights through backpropagation. Because we use a sequence-to-sequence network, performing a forward run produces a velocity model for each input frequency. As stated previously, these models are smoothed to a level that is suitable for the input frequency. To ensure that the network only fits structures at the appropriate scale for the input frequency, we use a structural similarity index measure (SSIM) loss function with a varying window that is a function of the input frequency. That is, we use larger SSIM windows for lower frequencies and decrease the window size with

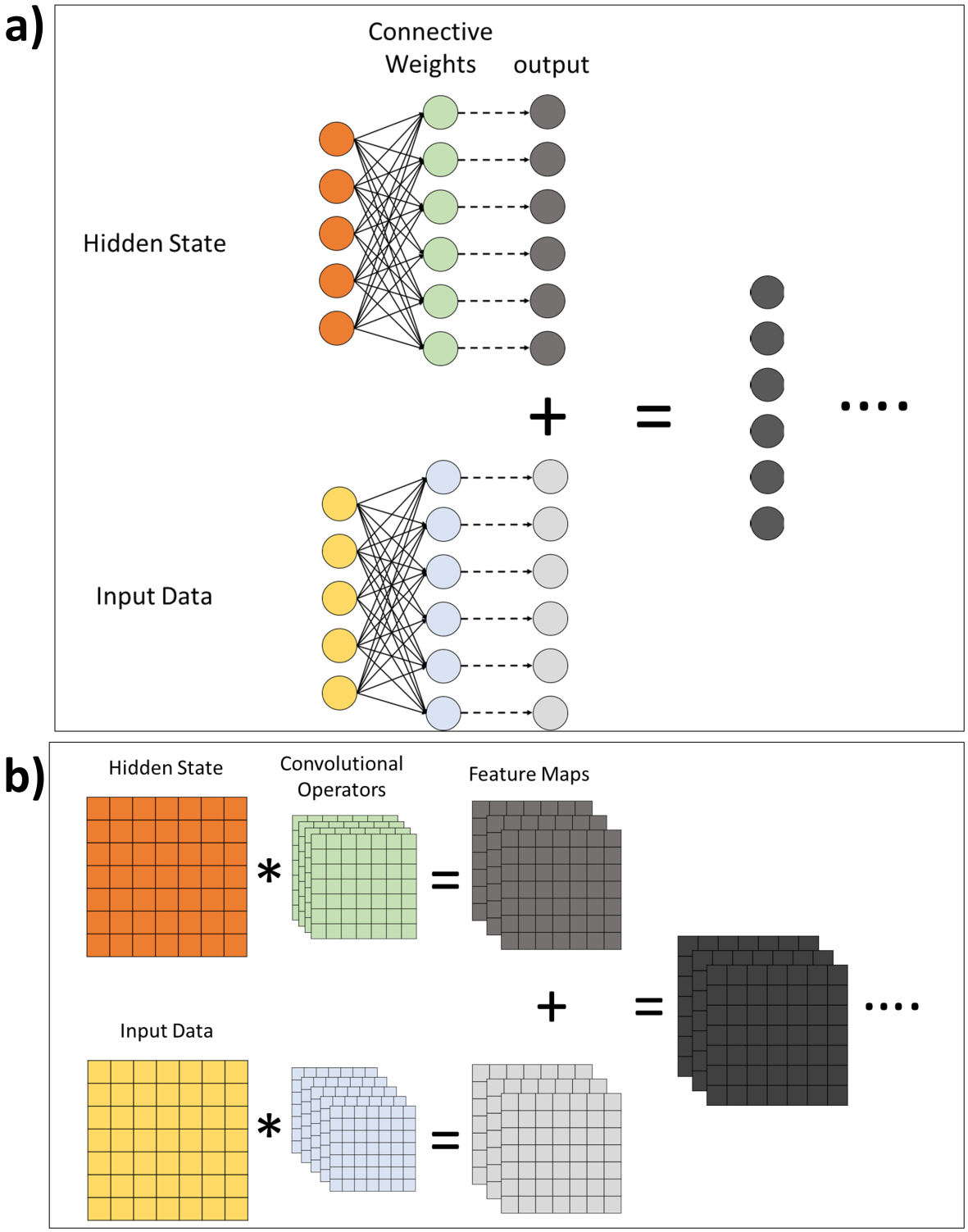

Figure 2. A comparison between conventional and convolutional RNNs. (a) The two fully connected layers comprising the conventional RNN. Information flows from one step to the next by adding the output of the hidden-state layer to the output of the layer handling the input data. (b) The same process as (a) with convolutional operators connecting the input and the output instead of connective weights. Items with the same role are indicated with the same color.

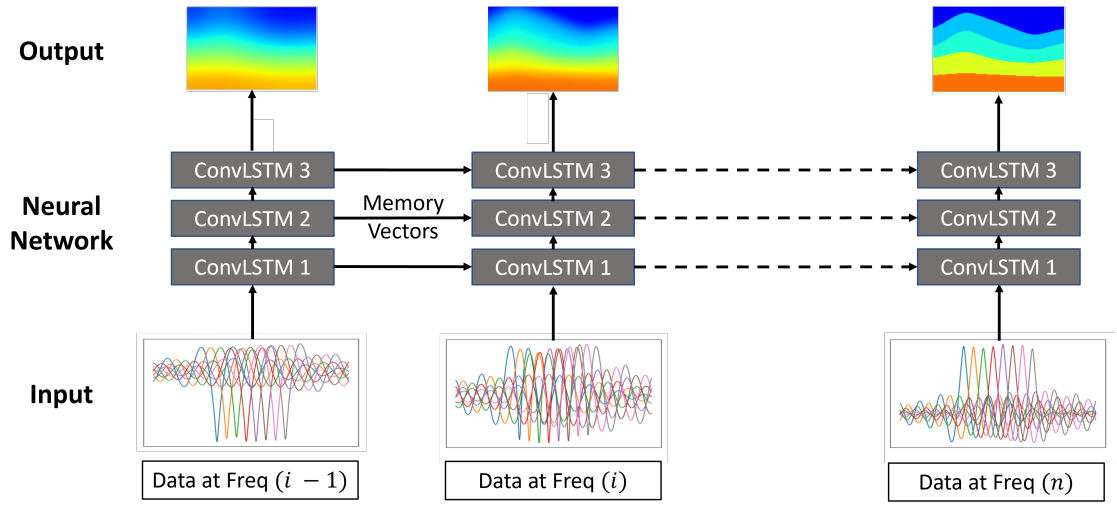

Figure 3. A sketch showing the network architecture used in this work. The network consists of three ConvLSTM layers. Information flows from one frequency to the next using two memory vectors, the hidden- and cell-state vectors. Each input frequency is constrained by its corresponding smoothed velocity model. The smoothing operator is larger for lower frequencies and gets progressively smaller as the input frequency increases.

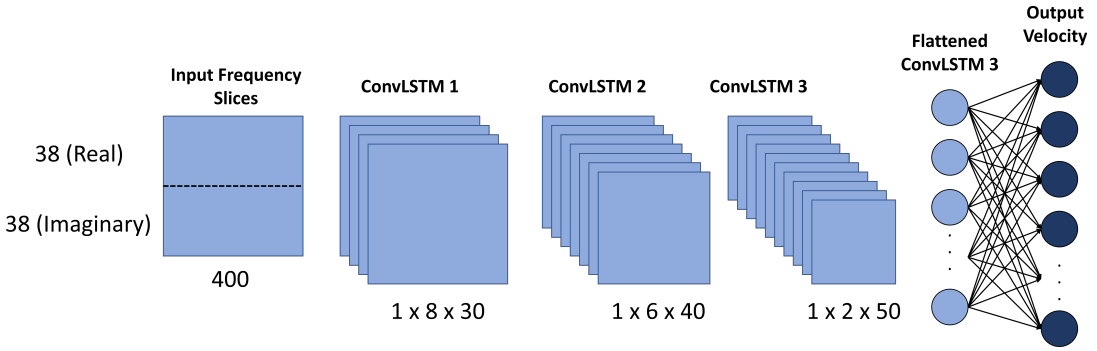

Figure 4. A diagram illustrating the convolutional parameters for each layer in the network used in this work. The input frequency slice consists of seismic data originated from 38 sources and recorded at 400 receivers. The top and bottom 38 rows of the input frequency slice consists of the real and imaginary components of the input seismic data, respectively. The first two numbers below each ConvLSTM layer represent the 2-D convolutional filter size in the y and x directions. The last number represents the number of filters, and consequently the number of output feature maps for each network. The output 2D feature maps for ConvLSTM3 are flattened into a vector in order for them to be connected to the output layer, which is a fully-connected layer.

increasing frequency. We also use a mean-squared-error loss function to measure the misfit between predicted models and their corresponding smoothed true models. To measure the value added to the training process by non-geological training data, we train two networks with the same hyperparameters but different training classes: (1) only geological and (2) hybrid (i.e., geological+geometric+background) data.

4 TESTING

To investigate the degree of generalization of the two networks, we test each using numerous subsets from different seismic benchmark data sets (BP 2004 and Marmousi). None of these subsets were involved in the training process and some have significantly different structures from those found in the training models. Therefore, the degree to which each network has generalized is measured by its performance on these subsets. Figure 5a-5c presents a comparison between the two networks in how well they recover a salt body of arbitrary shape. In all three cases, the network trained using the hybrid data recovers an approximate representation of the salt body in both shape and velocity value. It also recovers the true subsalt velocity gradient. Conversely, the velocity models re-

covered using the network trained only on geological models exhibit minimal structure and the estimated velocity values are nearly constant throughout the models. In addition to recovering complex salt bodies, the network trained using hybrid training data produces superior results when encountering a thin layer whose velocity is much greater than the surrounding medium (Figure 5d-5f). In particular, this network recover a velocity inversion as well as the shape and thickness of the fast layer.

5 CONCLUSIONS

We develop an ANN scheme that mimics an FWI-inspired multi-scale frequency-stepping RNN approach and combines hybrid geological-geometrical training data. We test the method by training two networks with the same architecture and hyper-parameters using two types of training data; one is trained using geological-only training models, the other using a combination of geological and geometric models. Testing results corroborate the assertion that using hybrid training data improves network generalization. By introducing a large number of random structures at different scales we demonstrate that this approach complements the geological training data and extends the range of structures recoverable by the trained ANN.

6 ACKNOWLEDGEMENTS

We thank Saudi Aramco (HA) and the other CWP sponsors whose support made this research possible. Computations were completed using the CSM *Wendian* facilities, the TensorFlow package, and the Madagascar software environment (www.ahay.org).

REFERENCES

Alzahrani, H., and J. Shragge, 2021, Neural network seismic velocity model building: A frequency-stepping approach: First International Meeting for Applied Geoscience & Energy Expanded Abstracts, 3561–3565.

———, 2022, Seismic velocity model building using neural networks: Training data design and learning generalization: Geophysics, **87**, no. 2, R193–R211.

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

Brenders, A. J., and R. G. Pratt, 2007, Waveform tomography of marine seismic data: What can limited offset offer?: SEG Technical Program Expanded Abstracts, 3024–3029.

Bunks, C., F. M. Saleck, S. Zaleski, and G. Chavent, 1995, Multiscale seismic waveform inversion: Geophysics, **60**, 1457–1473.

Fabien-Ouelle, G., and R. Sarkar, 2020, Seismic velocity estimation: A deep recurrent neural-network approach: Geophysics, **85**, no. 1, U21–U29.

He, Q., and Y. Wang, 2021, Reparameterized full-waveform inversion using deep neural networks: Geophysics, **86**, no. 1, V1–V13.

Hochreiter, S., and J. Schmidhuber, 1997, Long short-term memory: Neural Computation, **9**, no. 8, 1735–1780.

Lailly, P., 1983, The seismic inverse problem as a sequence of pre-stack migration, *in* Bednar, J.B. and R. Redner, E. Robinson, and A. Weglein, Eds., Conference on inverse scattering: Theory and Applications: Society of Industrial and Applied Mathematics.

Phan, S., and M. Sen, 2018, Hopfield networks for high-resolution prestack seismic inversion: SEG Technical Program Expanded Abstracts, 526–530.

Sirgue, L., O. I. Barkved, J. Dellinger, J. Etgen, U. Albertin, and J. H. Kommedal, 2010, Full-waveform inversion: the next leap forward in imaging at Valhall: First Break, **28**, 65–70.

Sirgue, L., and R. Pratt, 2004, Efficient waveform inversion and imaging: A strategy for selecting temporal frequencies: Geophysics, **69**, no. 1, 231–248.

Sun, J., Z. Niu, K. Innanen, J. Li, and D. Trad, 2020, A theory-guided deep-learning formulation and optimization of seismic waveform inversion: Geophysics, **85**, no. 2, R87–R99.

Tarantola, A., 1984, Inversion of seismic reflection data in the acoustic approximation: Geophysics, **49**, 1259–1266.

Zhang, Z., and T. Alkhalifah, 2019, Regularized elastic full-waveform inversion using deep learning: Geophysics, **84**, no. 5, R741–R751.

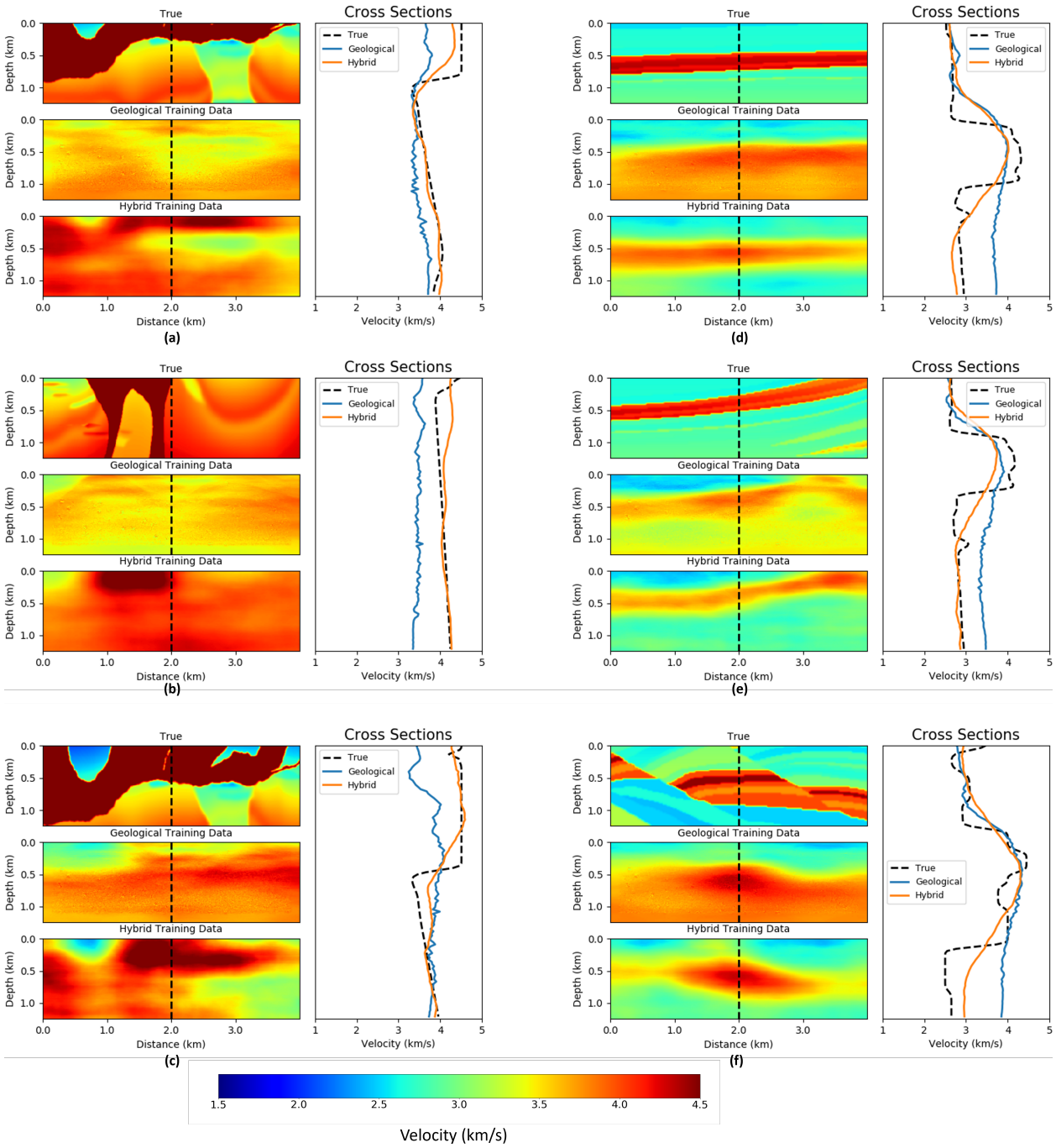

Figure 5. Testing results comparing the recovered models using the geological-trained and hybrid-trained networks. (a)-(c) Three examples of 2004 BP model subsets that show the superior performance of hybrid-trained network compared to the geological-trained network in recovering long-wavelength estimates of complex salt bodies and subsalt structure. (d)-(f) Three examples of Marmousi model subsets that similarly illustrate the improved recovery of strong velocity inversion features by the hybrid-trained network.