A deep learning formulation of elastic FWI with numerical and parameterization analysis

Tianze Zhang, Kris Innanen, Jian Sun, and Daniel Trad

ABSTRACT

In this paper we extend recent work on formulating seismic full waveform inversion within a deep learning environment. We are motivated both by the possibilities of incorporating training of multiple datasets with the relatively low dimensionality of a theory-guided network design, and by the fact that by doing so we implement an FWI algorithm ready-made for new computational architectures. A recurrent neural network is set up with rules enforcing elastic wave propagation, with the wavefield projected onto a measurement surface acting as the labelled data to be compared with observed seismic data. Training this network amounts to carrying out elastic FWI. Based on the Automatic Differential method, the exact gradient can be constructed by inspection and use of the computational graph, a gradient which acts to update the elastic model. We prepare our approach to mitigate cross-talk, which is a general property of multiparameter full waveform inversion algorithms, by allowing relative freedom to vary the eFWI parameterizations. The influence of random noise has also been examined.

INTRODUCTION

In the past year, CREWES has been expanding on and following a *theory-guided* approach to our analysis and implementation of machine-learning and/or AI methods for seismic data. In a theory-guided machine learning algorithm, the very large number of degrees of freedom represented by the weights of a ML network are reduced through assumptions regarding the physical origins of the output and input layers. For instance, theory-guiding within ML was used by Downton and Hampson (2018), in which a network designed to predict well log properties from seismic amplitudes was trained not only with real data but with synthetics derived from the Zoeppritz equations. Philosophically, theory-guided approaches are appealing to us, simply because it is difficult to believe that even the most modern AI algorithm will find the fact that deterministic physics connects the inputs and outputs of many geoscience problems unhelpful. The project reported in 2018 (and published this year) which formalized the approach for us was the work of Sun et al. (2019), in which a recursive neural network was set up to simulate the propagation of a seismic wave through a general acoustic medium. The network is set up in such a way that the trainable weights correspond to the medium unknowns (i.e., wave velocity model), and the non-trainable weights correspond to the mathematical rules (differencing etc.) enforcing wave propagation. The output layer was the field projected onto a measurement surface. Sun et al. (2019) discovered that training such a network with a single data set is very close to carrying out full waveform inversion.

We have put significant effort in 2019 then into extending and expanding this approach, and that expansion is the topic of this paper. The primary effort has gone into extending the acoustic work to the far more complex elastic problem. In this paper the isotropic elastic (2D) problem is broached; in a companion report, the viscoelastic problem is set up. There

are two kinds of motivation for doing this: (1) novel alterations to standard elastic FWI which can be accomplished only in such a milieu as the RNN; (2) the implementation of this seismic inversion problem within an ML environment tailor-made for computational architectures which are "the wave of the future". The importance of the second motivation is more or less self-evident. In the case of the first, there are aspects of a trainable ML that we have not yet broached and which may be extremely valuable. For instance, could the training aspect of a theory guided FWI-RNN be used to combat modelling errors? We do not answer these questions in this report - we simply formulate and stress-test an isotropic elastic machine learning full waveform inversion - the first of its kind to our knowledge.

Regardless of how it is implemented, full waveform inversion (FWI) is a challenging data fitting procedure aimed at extracting important information from seismic records. Key ingredients for conventional FWI are an efficient forward modeling method to simulate synthetic data, and a local differential approach, in which the gradient and direction are calculated to update the model.

Forward modelling/simulation

There are several ways to perform forward modeling, for example, the finite difference method, finite element method and the pseudo spectrum method. The finite difference method (FD method) is one of the most popular methods in seismic waveform simulation, reverse time migration and full-waveform inversion. FD method is straightforward to be implemented in programming and easy to be paralleled. FD method is also the first choice for high dimensional problems. Various kinds of FD methods have been developed by researchers to solve acoustic, elastic and viscoelastic wave equations, for instance, the staggered-grid method and the optimally accurate FD method (Moczo et al. (2007)). Forward modeling methods could influence the inversion results. The higher orders of time and space we use to discrete the partial derivatives, the more accuracy we will have to simulate waveform. Accurate synthetic data is essential for an inversion problem. However, the increase orders of time and space for the partial derivatives would inevitably increase computational cost. Furthermore, which kind of wave equation we would use to generate synthetic data could also influence the inversion results. For instance, if the area where we are investigating has a very high attenuation level and we only use the elastic wave equation to generate synthetic data. The inversion result would be inevitably influenced by this modeling error. The next essential step for FWI is the gradient calculation. The gradient for traditional FWI is achieved by calculating the zero-lag correlation of the forward propagating wavefields and the residual backpropagation wavefields. After we have the gradients, an optimization method could calculate the directions for FWI to accelerate the convergence rate. Next, a step length is usually obtained by line search. With the direction and step length, we can update our models and reduce the misfit.

Multiparameter updating

Issues faced by FWI are also challenging. The local minimum problem can be seen as one of the most important issues in FWI. FWI is a highly nonlinear problem, thus it may contain several local minimums for the misfit. When the initial model is not good or close

enough to the global minimum, the inversion could be trapped into the local minimum and can not jumpy out of it. When it comes to the elastic waveform inversion, which is a multi-parameter inversion, the trade-off problem is also an important issue. The trade-off problem in FWI can be considered as the conflation of the influence of one physical property on the data with another (Pan et al. (2019)). Implementing, second-order optimization method could mitigate this problem. Multi-parameter Hessian is a square and symmetric matrix with a block structure. Off diagonal blocks of the Hessian matrix blocks measure the correlation of Frechet derivative wavefields with respect to different physical parameters, which act to mitigate the trade-off effect of these parameters (Innanen (2014)). Tarantola introduced the use of scattering patterns for trade off problem (Tarantola (1986)), saying that by using different types of parameterization, the cross talk problem could be released. Gholami et al. (2010) studied the sensitivity of elastic FWI for VTI media. Pan et al. (2019) tested the performance of different parameterization for FWI by using both the synthetic data and field data.

MACHINE LEARNING METHODS IN GEOPHYSICS

Basic ideas of machine learning methods have been used in seismic inversion since the 1980s. Röth and Tarantola (1994) presented an application of neural networks to invert from time-domain seismic data to a depth profile of acoustic velocities. Due to the development of hardware, machine learning methods are becoming more commonly used in the study of Geophysics problems in recent years. Machine learning methods are studied widely in a variety of areas in Geophysics, such as fault detecting, denoising, reservoir predicting and inverting velocity models. Jin et al. (2019) implemented machine learning-based models using supervised learning techniques to identify fracture zones. A convolutional neural network (CNN) was trained by Zheng et al. (2019) to pick faults automatically in 3D seismic volumes, and the CNN was trained to make predictions of 1D velocity and density profiles from input seismic records. Peters et al. (2019) explained the similarities and differences between deep networks and other geophysical inverse problems and showed their utility in solving problems such as lithology interpolation between wells, horizon tracking, and segmentation of seismic images. Chen et al. (2019) leveraged the unsupervised learning philosophy of the autoencoding method to adaptively learn the seismic signals from the noisy observations. Li et al. (2019) proposed a novel method that is applicable to attenuating both incoherent noise, such as environmental noise, and coherent noise, such as ground roll and scattered noise, under a unified learning-based framework. Zhang et al. (2019a) proposed an interpolation method based on the denoising convolutional neural network (CNN) for seismic data. Parra et al. (2019) presented a method based on an inversion algorithm that automatically inverts subsidence signatures for tunnel radius, depth, Poisson's ratio, and the gap parameter. Smith et al. (2019) developed a machine-learning workflow that incorporates geophysical and geologic data, as well as engineering completion parameters, into a model for predicting well production. By using a supervised machine learning method, Shustak and Landa (2018) developed a method uses numerical backpropagation of the time-reversed registered wavefield followed by an analysis of its obtained focusing.

When it comes to the inversion of the velocity model, Sun and Demanet (2018) applied deep learning method to the challenging bandwidth extension problem that is essential for

FWI.Zhang et al. (2019b) designed an end-to-end framework, the velocityGAN, which can generate high-quality velocity images directly from the raw seismic waveform data. Wu and Lin (2018) trained a network with an encoder-decoder structure to model the correspondence from seismic data to subsurface velocity structures. Yang and Ma (2019), based on convolution neural network, investigated a method based on the supervised deep fully convolutional neural network for velocity-model building directly from raw seismograms. However, in these methods, the theories of wave propagation and inversion methodology have been ignored, which means that the nonlinear relationship between the seismic records and the velocity models are based on large amount of training. Jian sun et. al (2019) proposed a theory-guided machine leaning method, which used the recurrent neural network (RNN) to perform scalar wave FWI. This RNN is consisted of several RNN cells, and each RNN cell is designed according to the scalar wave equation. This method combines the knowledge we know about wave propagation and the advantages of machine learning methods. In this study, we extend their idea into elastic media to perform a more challenging inversion test.

INTRODUCTION TO RECURRENT NEURAL NETWORK

The recurrent neural network is a powerful model for sequential data. RNN achieves state of the art performance on tasks that include language modeling (Mikolov (2012)), speech recognition (Graves et al. (2013)), and machine translation (Kalchbrenner and Blunsom (2013)), (Zaremba et al. (2014)).

A recurrent neural network is consisted of several recurrent neural network cells. Each cell could be considered as a time step with some mathematical operations inside. The operations in each cell would use some information that is generated from the previous cells. Thus, the data generated in a cell is used recurrently by other cells. Figure 1 (a) shows a simple structure of a recurrent neural network to solve a problem. The mathematical operations between the internal variables are also listed at the bottom of the figure. The problem may need several cells to achieve the final goal, but only three cells of the network are demonstrated in this figure for simplification. In this RNN, sequence $\mathbf{S} = [s_0, s_1, s_2, \cdots]$ is the input data. Sequence $\mathbf{w} = [w_0, w_1, w_2, \cdots]$ contains the parameters that need to be optimized. O is an initial input. Elements in sequence $\mathbf{O} = [O_1, O_2, O_3 \cdots]$ are the internal variables calculated in cells. Sequence $\mathbf{H} = [h_0, h_1, h_2, \cdots]$ is the labeled data. Sequence $\mathbf{V} = (v_1, v_2, v_3, \cdots)$ evaluates the difference between output data of the cell and the labeled data by using mean square objective function. We wish to make labeled data $\mathbf{H}$ and RNN output $\mathbf{O}$ as close as possible, by optimizing the trainable parameters $\mathbf{w} = [w_0, w_1, w_2, \cdots]$.

In order to optimize the trainable parameters, $\mathbf{w} = [w_0, w_1, w_2, \cdots]$, we need to calculate the partial derivative of the residual with respect to these parameters. This process is achieved by using the backpropagation method according to the Chain's Rule. For instance, we want to calculate the partial derivative of the residual V_3 with respect to w_1. According to the computational graph we have:

$$\begin{aligned}\frac{\partial V_3}{\partial w_1} = \frac{\partial V_3}{\partial O_9}\frac{\partial O_9}{\partial w_1} =& \frac{\partial V_3}{\partial O_9}\frac{\partial O_9}{\partial O_8}\frac{\partial O_8}{\partial O_5}\frac{\partial O_5}{\partial w_1} \\ =& \frac{\partial V_3}{\partial O_9}\frac{\partial O_9}{\partial O_8}\frac{\partial O_8}{\partial O_5}\frac{\partial O_5}{\partial O_2}\frac{\partial O_2}{\partial w_1} = -2cos(w_1 s_1)s_1(h_3 - O_9)\end{aligned} \tag{1}$$

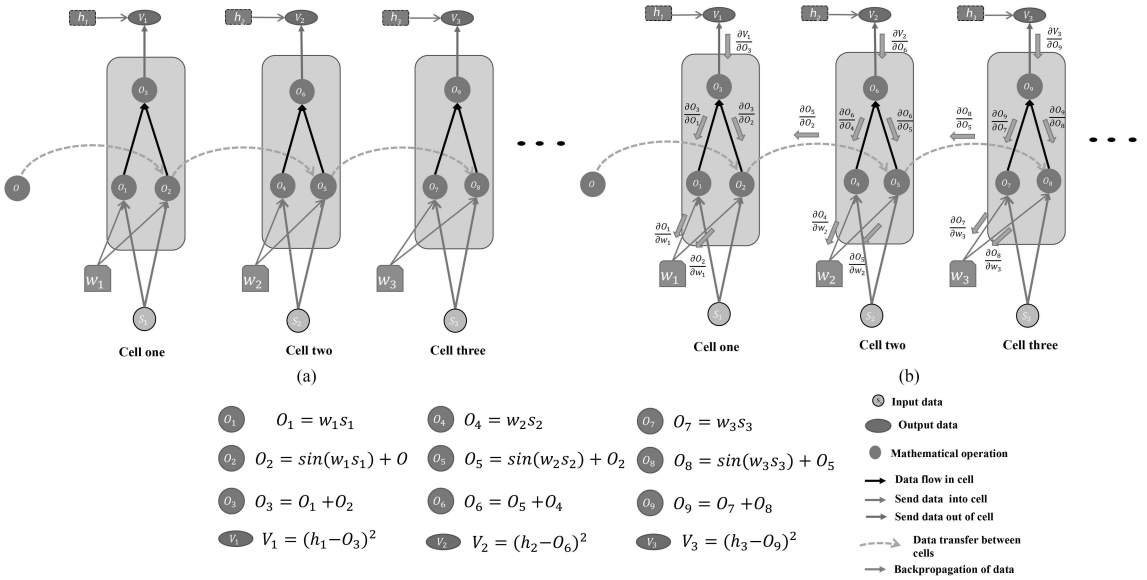

FIG. 1. Forward propagating of RNN

The gradients of w_1, and w_2 are also achieved in the same way. After we have the gradients, according to an optimization method we can update the weights and start another iteration.

We do not need to calculate all the partial derivatives by ourselves. If we are using a machine learning library of Pytorch or Tensor Flow, the partial derivative would be calculated automatically calculated by using the Automatic Differential (AD) engine built inside of these machine learning libraries. During the forward propagation, a Dynamic Computational Graph (DCG) would be built. This Dynamic Computational Graph records how every internal parameter is calculated from its previous one. When we need to calculate the partial derivative of the output with respect to the trainable parameter, Automatic Differential engine would use the DCG to backpropagate the computational graph to that parameter, and calculate the partial derivative along the computational graph using Chain's rule.

According to the modeling framework of finite difference method, seismic waveform modeling could also be considered as a kind of sequential data, since the wave equation can be discrete in time and the wavefields at the current time step can be calculated from previous time steps. Thus, we could also form a kind of recurrent neural network for seismic waveform modeling. When we want to simulate synthetic data, RNN would propagation forwardly to generate synthetic wavefields, and the wavefields would be recorded at the surface of the model, which forms the synthetic shotrecords. At the same time of forward propagating, the mathematical operations between each internal variables would be saved for backpropagation. When it comes to inversion, the residual between the synthetic shotrecords and observe records could be calculated. With the Automatic Differential method, partial derivative of the residual with respect to the velocity model would be gained. Then, we can use an optimization method to update the model and reduce the misfit.

ISOTROPIC ELASTIC WAVE EQUATION

Equation (2) shows the equation of the isotropic elastic wave equation. $\mathbf{v}_x$ and $\mathbf{v}_z$ are the elastic velocity fields in x and z direction. $\boldsymbol{\sigma}_{xx}$, $\boldsymbol{\sigma}_{zz}$ and $\boldsymbol{\sigma}_{xz}$ are the stress tensors. ρ is the density of the model. λ and μ are the first and the second Lame parameter for the elastic media.

$$\begin{cases} \dfrac{\partial \mathbf{v}_x}{\partial t} = \dfrac{1}{\rho}\left(\dfrac{\partial \boldsymbol{\sigma}_{xx}}{\partial x} + \dfrac{\partial \boldsymbol{\sigma}_{xz}}{\partial z}\right) \\ \dfrac{\partial \mathbf{v}_z}{\partial t} = \dfrac{1}{\rho}\left(\dfrac{\partial \boldsymbol{\sigma}_{xz}}{\partial x} + \dfrac{\partial \boldsymbol{\sigma}_{zz}}{\partial z}\right) \\ \dfrac{\partial \boldsymbol{\sigma}_{xx}}{\partial t} = (\lambda + 2\mu)\dfrac{\partial \mathbf{v}_x}{\partial x} + \lambda \dfrac{\partial \mathbf{v}_z}{\partial z} \\ \dfrac{\partial \boldsymbol{\sigma}_{zz}}{\partial t} = (\lambda + 2\mu)\dfrac{\partial \mathbf{v}_z}{\partial z} + \lambda \dfrac{\partial \mathbf{v}_x}{\partial x} \\ \dfrac{\partial \boldsymbol{\sigma}_{xz}}{\partial t} = \mu\left(\dfrac{\partial \mathbf{v}_x}{\partial z} + \dfrac{\partial \mathbf{v}_z}{\partial x}\right) \end{cases} \tag{2}$$

To simulate the wave propagation in the underground world, a PML (Perfect Matching Layer) damping coefficient is used to absorb energy when the waves hit the boundary of the model. Thus, we need to discrete the velocity fields and the stress fields in both x and z direction. So we have equation (3) and (4):

$$\begin{cases} \left(\dfrac{\partial}{\partial t} + d_x\right)\mathbf{v}_x^x = \dfrac{1}{\rho}\dfrac{\partial \boldsymbol{\sigma}_{xx}}{\partial x}, & \left(\dfrac{\partial}{\partial t} + d_z\right)\mathbf{v}_x^z = \dfrac{1}{\rho}\dfrac{\partial \boldsymbol{\sigma}_{xz}}{\partial z} \\ \left(\dfrac{\partial}{\partial t} + d_x\right)\mathbf{v}_z^x = \dfrac{1}{\rho}\dfrac{\partial \boldsymbol{\sigma}_{xz}}{\partial x}, & \left(\dfrac{\partial}{\partial t} + d_z\right)\mathbf{v}_z^z = \dfrac{1}{\rho}\dfrac{\partial \boldsymbol{\sigma}_{zz}}{\partial z} \\ \left(\dfrac{\partial}{\partial t} + d_x\right)\boldsymbol{\sigma}_{xx}^x = \Pi\dfrac{\partial \mathbf{v}_x}{\partial x}, & \left(\dfrac{\partial}{\partial t} + d_z\right)\boldsymbol{\sigma}_{xx}^z = \lambda\dfrac{\partial \mathbf{v}_z}{\partial z}, \\ \left(\dfrac{\partial}{\partial t} + d_x\right)\boldsymbol{\sigma}_{zz}^x = \lambda\dfrac{\partial \mathbf{v}_x}{\partial x}, & \left(\dfrac{\partial}{\partial t} + d_z\right)\boldsymbol{\sigma}_{zz}^z = \Pi\dfrac{\partial \mathbf{v}_z}{\partial z} \\ \left(\dfrac{\partial}{\partial t} + d_x\right)\boldsymbol{\sigma}_{xz}^x = \mu\dfrac{\partial \mathbf{v}_z}{\partial x}, & \left(\dfrac{\partial}{\partial t} + d_z\right)\boldsymbol{\sigma}_{xz}^z = \mu\dfrac{\partial \mathbf{v}_x}{\partial z} \end{cases} \tag{3}$$

where

$$\begin{cases} \mathbf{v}_x = \mathbf{v}_x^x + \mathbf{v}_x^z \\ \mathbf{v}_z = \mathbf{v}_z^x + \mathbf{v}_z^z \\ \boldsymbol{\sigma}_{xx} = \boldsymbol{\sigma}_{xx}^x + \boldsymbol{\sigma}_{xx}^z \\ \boldsymbol{\sigma}_{zz} = \boldsymbol{\sigma}_{zz}^x + \boldsymbol{\sigma}_{zz}^z, \\ \boldsymbol{\sigma}_{xz} = \boldsymbol{\sigma}_{xz}^x + \sigma_{xz}^z, \end{cases} \tag{4}$$

In the above equation d_x and d_z are the PML boundary damping coefficients in x and z direction respectively. $\Pi = \lambda + 2\mu$. The mathematical expression of the damping coefficient

is:

$$d_\eta(i) = d_{0\eta}\left(\frac{i}{n_{pml\eta}}\right)^P, \tag{5}$$

where η means x or z direction, and i is the length of the effective PML layers form the simulation boundary. $n_{pml\eta}$ is the number of the PML layers in η direction. The value of P is usually between 1 and 4. The value of $d_{0\eta}$ is:

$$d_{0\eta} = \log\left(\frac{1}{R}\right)\frac{\tau \mathbf{v}_s}{n_{pml\eta}\Delta\eta}, \tag{6}$$

where R is the theoretical reflection coefficients, and τ is the tuning parameter which has the **v**alue between 3 and 4. $\mathbf{v}_s$ is the shear velocity value. $\Delta\eta$ is the grid length in η direction.The combination of the R, $n_{pml\eta}$, can be expressed as:

$$\begin{cases} R = 0.01, & n_{pml\eta} = 5, \\ R = 0.001, & n_{pml\eta} = 10, \\ R = 0.0001, & n_{pml\eta} = 20 \end{cases} . \tag{7}$$

Derivative calculation based on image convolution

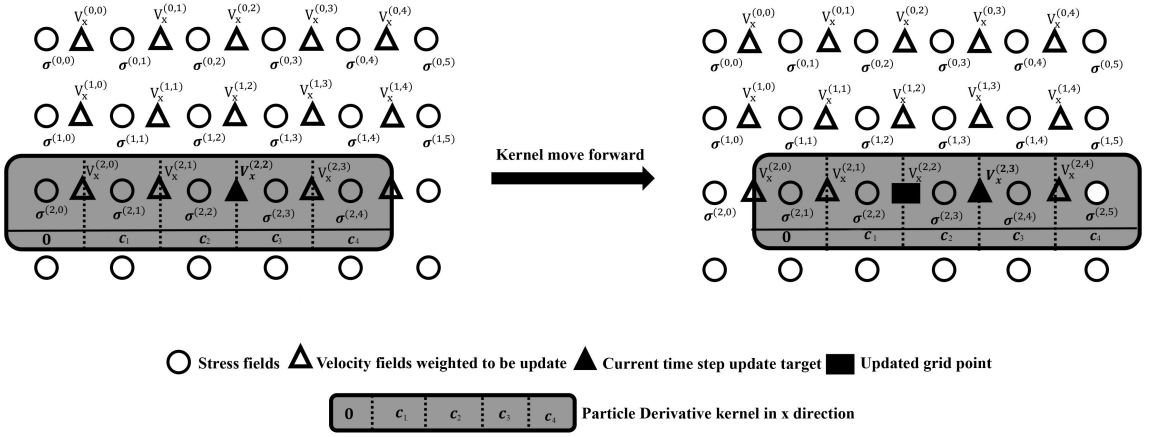

FIG. 2. Staggered grid finite difference method based on convolution

In this study, we use the staggered grid finite difference method to simulate synthetic data, and we use image convolution operation in machine learning to calculate partial derivative. Figure 2 shows how we use image convolution to update the velocity fields in x direction at grid point $(2, 2)$ and $(2, 3)$. According to the staggered grid format we first need to calculate the partial derivative fields for the stress fields. From the figure, we can see that grids for particle velocity field are located in the middle of stress fields grids, which form the staggered grid format. To calculate the partial derivative by using image convolution, first, we need to create a kernel. The kernel is designed according to the staggered grid finite-difference coefficients and the staggered stencil. Kernel would move from the edge of the original matrix element by element to calculate the partial derivative. In figure 2, to update the velocity fields at point $(2, 2)$, $V_x^{(2,2)}$, a kernel which has the value of $[0, c_1, c_2, c_3, c_4]$ is designed. At first, the kernel covers five stress fields elements which are $(\sigma^{(2,0)}, \sigma^{(2,1)}, \sigma^{(2,2)}, \sigma^{(2,3)}, \sigma^{(2,4)})$. This means that these five elements of the stress fields

would be used to update the velocity fields. By summing up all the elements of the doc product of these five elements with the kernel matrix, we can get the value of the partial derivative stress fields to update the velocity fields, Thus, the updated velocity fields at point $(2, 2)$ can be calculated as:

$$V_x^{(2,2)}\big|^{k+\frac{1}{2}} = V_x^{(2,2)}\big|^{k-\frac{1}{2}} + \frac{\Delta t}{\rho}(c_1\sigma^{(2,1)}\big|^k + c_2\sigma^{(2,2)}\big|^k + c_3\sigma^{(2,3)}\big|^k + c_4\sigma^{(2,4)}\big|^k) \quad (8)$$

After this operation, the kernel would move forward to calculate the partial derivative for next point, which can be expressed as:

$$V_x^{(2,3)}\big|^{k+\frac{1}{2}} = V_x^{(2,3)}\big|^{k-\frac{1}{2}} + \frac{\Delta t}{\rho}(c_1\sigma^{(2,2)}\big|^k + c_2\sigma^{(2,3)}\big|^k + c_3\sigma^{(2,4)}\big|^k + c_4\sigma^{(2,5)}\big|^k) \quad (9)$$

This operation would continue until the partial derivative of the last grid point of the field is calculated. Thus, the partial derivative of a field can be expressed in an image convolution way:

$$\partial_x\sigma_x = \sigma_x * K_{x1}, \quad (10)$$

where K_{x1} is the kernel we create to calculate the partial derivative in x direction and $*$ means image convolution. By using machine learning packages like Tensor Flow or Pytorch, the image convolution function is well designed and we do not need to program everything manually. All we need to do is to choose which field we want to calculate partial derivative and design the kernels.

By using the image convolution method to calculate the partial derivative and implement PML boundary, the isotropic elastic wave equation in an RNN cell can be expressed

as:

$$\begin{cases} \mathbf{v}_x^x|^{k+\frac{1}{2}} = (1-\Delta t d_x)\mathbf{v}_x^x|^{k-\frac{1}{2}} + \dfrac{\Delta t}{\rho}(\boldsymbol{\sigma}_{xx}|^k * K_{z1}) \\ \mathbf{v}_x^z|^{k+\frac{1}{2}} = (1-\Delta t d_z)\mathbf{v}_x^z|^{k-\frac{1}{2}} + \dfrac{\Delta t}{\rho}(\boldsymbol{\sigma}_{xz}|^k * K_{z2}) \\ \mathbf{v}_z^x|^{k+\frac{1}{2}} = (1-\Delta t d_x)\mathbf{v}_z^x|^{k-\frac{1}{2}} + \dfrac{\Delta t}{\rho}(\boldsymbol{\sigma}_{xz}|^k * K_{x2}) \\ \mathbf{v}_z^z|^{k+\frac{1}{2}} = (1-\Delta t d_z)\mathbf{v}_z^z|^{k-\frac{1}{2}} + \dfrac{\Delta t}{\rho}(\boldsymbol{\sigma}_{zz}|^k * K_{x1}) \\ \mathbf{v}_x|^{k+\frac{1}{2}} = \mathbf{v}_x^x|^{k+\frac{1}{2}} + \mathbf{v}_x^z|^{k+\frac{1}{2}} \\ \mathbf{v}_z|^{k+\frac{1}{2}} = \mathbf{v}_z^x|^{k+\frac{1}{2}} + \mathbf{v}_z^z|^{k+\frac{1}{2}} \\ \boldsymbol{\sigma}_{xx}^x|^{k+1} = (1-\Delta t d_x)\boldsymbol{\sigma}_{xx}^x|^k + (\lambda+2\mu)\Delta t(\mathbf{v}_x|^{k+1/2} * K_{x2}) \\ \boldsymbol{\sigma}_{xx}^z|^{k+1} = (1-\Delta t d_z)\boldsymbol{\sigma}_{xx}^z|^k + \lambda\Delta t(\mathbf{v}_z|^{k+1/2} * K_{z2}) \\ \boldsymbol{\sigma}_{xx}|^{k+1} = \boldsymbol{\sigma}_{xx}^x|^{k+1} + \boldsymbol{\sigma}_{xx}^z|^{k+1} \\ \boldsymbol{\sigma}_{zz}^x|^{k+1} = (1-\Delta t d_x)\boldsymbol{\sigma}_{zz}^x|^k + \lambda\Delta t(\mathbf{v}_x|^{k+1/2} * K_{x2}) \\ \boldsymbol{\sigma}_{zz}^z|^{k+1} = (1-\Delta t d_z)\boldsymbol{\sigma}_{zz}^z|^k + (\lambda+2\mu)\Delta t(\mathbf{v}_z|^{k+1/2} * K_{z2}) \\ \boldsymbol{\sigma}_{zz}|^{k+1} = \boldsymbol{\sigma}_{zz}^x|^{k+1} + \boldsymbol{\sigma}_{zz}^z|^{k+1} \\ \boldsymbol{\sigma}_{xz}^x|^{k+1} = (1-\Delta t d_x)\boldsymbol{\sigma}_{xz}^x|^k + \mu\Delta t(\mathbf{v}_x|^{k+1/2} * K_{z1}) \\ \boldsymbol{\sigma}_{xz}^z|^{k+1} = (1-\Delta t d_z)\boldsymbol{\sigma}_{xz}^x|^k + \mu\Delta t(\mathbf{v}_z|^{k+1/2} * K_{x1}) \\ \boldsymbol{\sigma}_{xz}|^{k+1} = \boldsymbol{\sigma}_{xz}^x|^{k+1} + \boldsymbol{\sigma}_{xz}^z|^{k+1} \end{cases} \tag{11}$$

,where the kernels are

$$K_{z1} = \begin{bmatrix} 0, \\ (1/24)/dz, \\ (-9/8)/dz, \\ (9/8)/dz, \\ (-1/24)/dz, \end{bmatrix} \tag{12}$$

$$K_{z2} = \begin{bmatrix} (1/24)/dz, \\ (-9/8)/dz, \\ (9/8)/dz, \\ (-1/24)/dz, \\ 0 \end{bmatrix} \tag{13}$$

$$K_{x1} = \begin{bmatrix} 0, & (1/24)/dx, & (-9/8)/dx, & (9/8)/dx, & (-1/24)/dx, \end{bmatrix} \tag{14}$$

$$K_{x2} = \begin{bmatrix} (1/24)/dx, & (-9/8)/dx, & (9/8)/dx, & (-1/24)/dx, & 0 \end{bmatrix} \tag{15}$$

Equation (11) is also what is happening in an elastic RNN cell. Details about the structure of an elastic RNN cell, please see appendix two. Figure 3 shows how elastic RNN generates seismic records. Every RNN cell is designed according to the isotropic elastic wave equation, which is equation (11). In figure 3, only the velocity fields at the x direction is shown, however, at each time step we need to generate all the following fields:

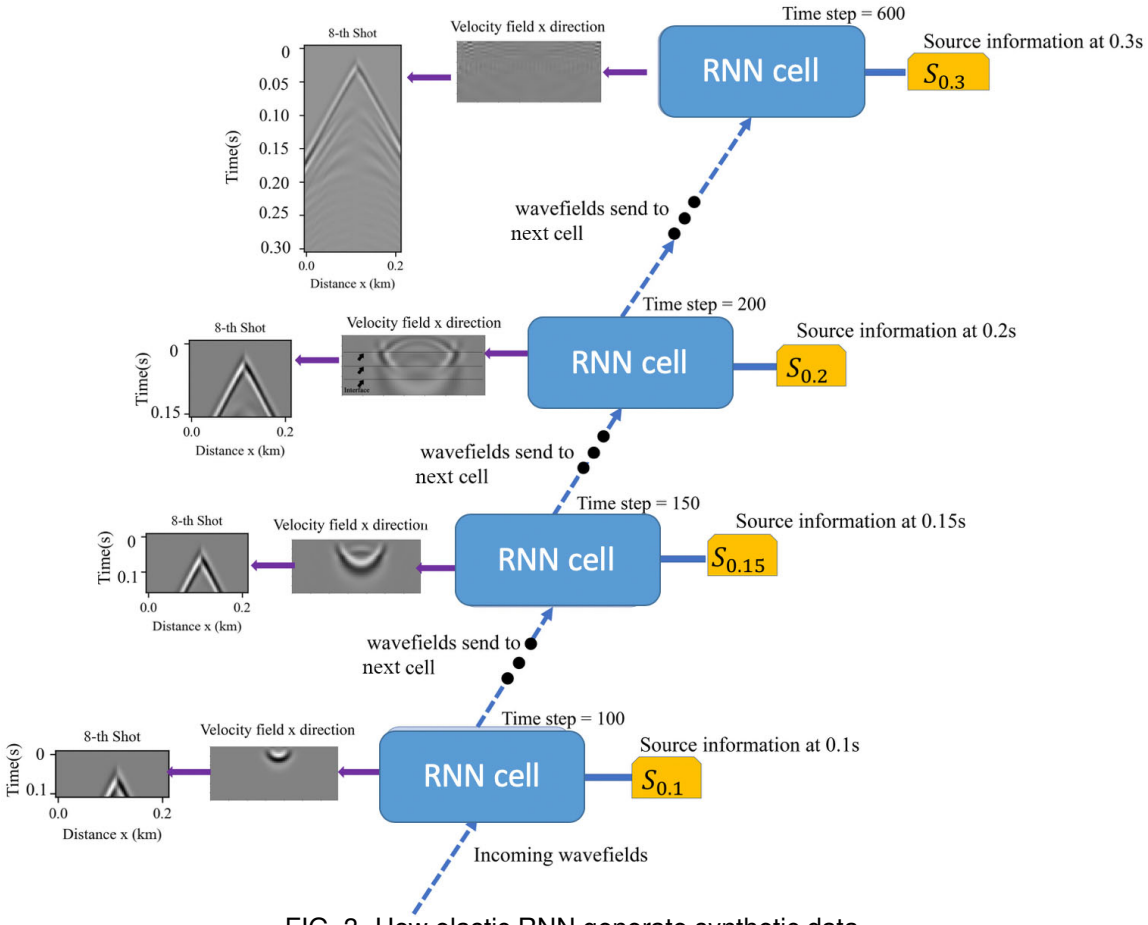

FIG. 3. How elastic RNN generate synthetic data

$(\boldsymbol{\sigma_{xx}}|^k, \boldsymbol{\sigma^x_{xx}}|^k, \boldsymbol{\sigma^z_{xx}}|^k, \boldsymbol{\sigma_{zz}}|^k, \boldsymbol{\sigma^x_{zz}}|^k, \boldsymbol{\sigma^z_{zz}}|^k, \boldsymbol{\sigma_{xz}}|^k, \boldsymbol{\sigma^x_{xz}}|^k, \boldsymbol{\sigma^z_{xz}}|^k)$
and $(\boldsymbol{V^x_x}|^{k-\frac{1}{2}}, \boldsymbol{V^z_x}|^{k-\frac{1}{2}}, \boldsymbol{V^x_z}|^{k-\frac{1}{2}}, \boldsymbol{V^z_z}|^{k-\frac{1}{2}})$

Table 1. Testing model parameters

Physical parameters	Physical meaning	Value
V_p	Pressure wave velocity	$2000m/s$
V_s	Shear wave velocity	$1400m/s$
ρ	density	$1000kg/m^3$

Table 1 lists the property of an elastic media, and figure 4 shows the velocity fields in the horizontal x direction and vertical z direction calculated at different time steps generated by RNN. The source of the wavelet is a Ricker wavelet with a main frequency of $35Hz$. The size of model is 100×100, and the grid length is $dx = dz = 4m$. The source is located at the center of the model. Figure 4 (a) are the horizontal and vertical velocity fields generated at time at $0.1s$, and figure 4 (b) are the fields calculated at $0.2s$. We can see that RNN has provided us with right wave fields, and when the waves hit the boundary, they are absorbed properly.

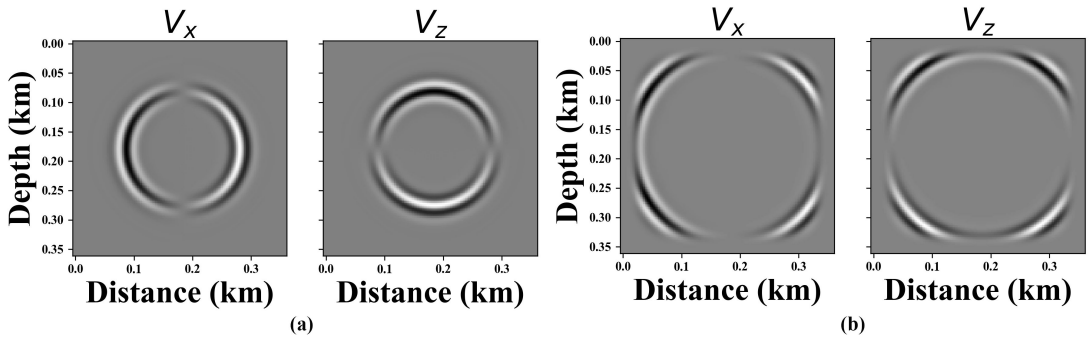

FIG. 4. (a)Velocity fields generated at time at $0.1s$,(b) velocity fields calculated at $0.2s$

NUMERICAL INVERSION

In this section, we test the efficiency of the method we introduce by doing numerical tests. Figure 5, (a), (d) and (g) demonstrate the true model for V_p, V_s and density model. The source of the wavelet is Ricker's wavelet with a main frequency of $35Hz$. The model has 54×101 grid points, and the grid length of the model is $dx = dz = 4m$. Ten sources are evenly distributed at the top of the model, and there are 101 receivers positioned at the top of the model. The total time to receive the shotrecords is $0.3s$, and length for the time step is $dt = 0.0003s$. Initial models are obtained through Gaussian smooth. The maximum iteration time is 200.

Gradient calculation is one of the most important steps for FWI. The gradients in this study are calculated by the Automatic Differential engine in the machine learning library, Pytroch. The mathematical principle under this Automatic Differential engine is the Chain's Rule. Gradients calculated in this way are the exact gradients based on forward computational graph. During forward propagation, RNN would generate synthetic shotrecords at each time step, at the same time, a Dynamic Computational Graph (DCG) would be built to record the mathematical operation between each variable, for instance, how the stress field is calculated according to the stress field at the last time step and the partial derivative of the velocity field. Then, we can start the backpropagation method. According to the Dynamic Computational Graph saved in memory, residual between observe data and the synthetic data would be calculated and backpropagated to the trainable parameters. The partial derivative of the residual with respect to the trainable parameters would be calculated by using the Chain's rule, in the same way as we present in the first section. After calculating the gradient, we need to use an optimization method to calculate the direction and update the trainable parameters.

In this test, the optimization method we use is the Adam algorithm. Algorithm 1 shows the basic step about Adam algorithm. One of the advantages of Adam algorithm is that it has the ability to shrink the 'step-length' of model updates during the iteration process, which means a relatively large learning rate can also be tolerated with oscillation occurred at first few iterations and we do not need to search the step length in every iteration. A traditional line search method, like the Wolfe Principle, would require extra forward modeling calculation, which would inevitably increase the computational cost. With Adam algorithm, the extra computational cost for step length would be mitigated (Jian Sun et. al.

2019).

Algorithm 1 Adam algorithm. Recommended setting for hyper-parameters: $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$, All operations on vectors are element-wise. β_1^k and β_2^k donated the k power of β_1 and β_2

Input: Current velocity model $\mathbf{v}_t$; Learning Rate :α; Exponential decay rates for the moment estimates: β_1, $\beta_2 \in [0, 1)$; Perturbation models achieved from gradient descent method : $\delta\mathbf{v}$

Output: Updated model: $\mathbf{v}_{t+1}$

1: Update biased 1st momentum: $\boldsymbol{m}_k \leftarrow \beta_1 \cdot \boldsymbol{m}_{k-1} + (1 - \beta_1) \cdot \boldsymbol{\delta}\mathbf{v}_k$
2: Update biased 2nd momentum $\boldsymbol{r}_k \leftarrow \beta_2 \cdot \boldsymbol{r}_{k-1} + (1 - \beta_2) \cdot \boldsymbol{\delta}\mathbf{v}_k^2$
3: Bias correction for 1st momentum: $\tilde{\boldsymbol{m}}_k \leftarrow \boldsymbol{m}_k / \left(1 - \beta_1^k\right)$
4: Bias correction for 2nd momentum $\tilde{\boldsymbol{r}}_k \leftarrow \boldsymbol{r}_k / \left(1 - \beta_2^k\right)$
5: Parameters updates $\mathbf{v}_k \leftarrow \mathbf{v}_{k-1} - \alpha \cdot \tilde{\boldsymbol{m}}_k / \left(\sqrt{\tilde{\boldsymbol{r}}}_k + \epsilon\right)$

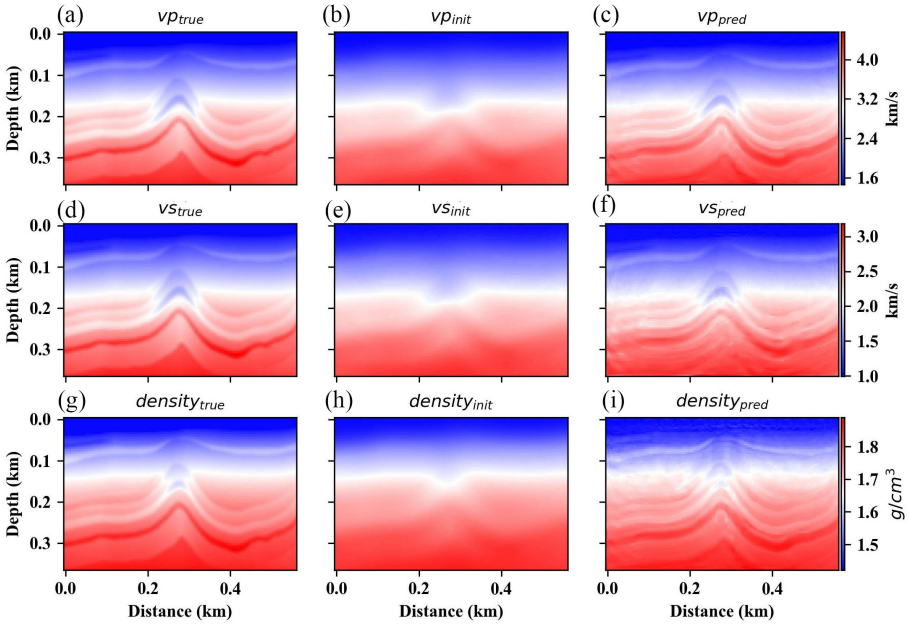

FIG. 5. Velocity parameterization (a), (b), (c) True, initial and inversion result for V_p model respectively, (d), (e), (f) True, initial and inversion result for V_s model respectively,(g), (h), (i) True, initial and inversion result for ρ model respectively.

Figure 5 (b), (e), (h) are the initial models we use for V_p, V_s and ρ model. Figure 5 (c) is the inversion result of the V_p model. Figure 5 (f) is the inversion result of the V_s model. Figure 5 (i) is the inversion result of the density ρ model. Compared with the true model, the layers of the subsurface have been correctly updated, which means that this elastic FWI deep learning method based on recurrent neural network could provide us with promising inversion results. The matching between the inversion results and the true models are very satisfactory. Compared with other models, the inversion results for density is not as good as the V_p and V_s model. In figure 5 (i), we can see the blurred layers on the top of the density model. This may due to the cross-talk problem. The cross-talk problem happens when we need to update several parameters simultaneously, and influences the inversion

results between each model. We will talk about how to release this problem in the next section.

Different parameterization

Estimating multiparameter models is an essential step for lithologic characterization and reservoir monitoring. However, if we intend to update several parameters simultaneously, the update of one parameter would influence other parameters, and this is the trade-off problem or also referred to the cross-talk problem. One of the main reasons for this problem is that the nature of seismic response for several different parameters are coupled together from transmission to reflection. Furthermore, the magnitude for different parameter can have different orders and different strength, which would make the inversion poorly conditioned (Operto et al. (2013))

A second-order optimization method could be used to mitigate the cross talk problem because the inverse of the Hessian is used to suppress the parameters that would be incorrectly updated, (Innanen (2014), Yang et al. (2016)). However, the direct inverse of the Hessian matrix based on the Newton's optimization method would cost a huge amount of memory. The implementation of the Truncated-Newton method would to some extent release this computational cost problem since the inverse of Hessian would be solved with a matrix-free scheme of the conjugate-gradient algorithm (Métivier et al. (2013), Boehm and Ulbrich (2015)). Quasi-newton methods, for instance, the l-BFGS method, could also be used to decrease the computational cost by approximating the inverse Hessian matrix with the gradient information from previous iterations (Brossier et al. (2009)).

Different parameterization method is also proposed to deal with the coupling effects in multiple physical parameter FWI. The coupling effects between different physical parameters are controlled by the physical relationships of these parameters and the subsurface model parameterization choice. Tarantola (1986), by using the "scattering" (or "radiation") pattern, systematically analyzed the effect of different model parameterizations on isotropic FWI. It is usually believed that the more different the scattering pattern between each parameter is, the better the inversion results we should have. Köhn et al. (2012) showed that in different geophysical parameterization of isotropic FWI, density is always the hardest to get good inversion result due to the energy leakage. Pan et al. (2019) used different parameterizations to test their effect on Hussar Land data.

The objective of this section is to test the performance of different parameterizations under the framework of the recurrent neural network for isotropic-elastic FWI. By modifying the RNN cell and changing the trainable parameters, we combine the deep learning method and parameterization theory. Three parameterization models are considered: velocity model (V-Dmodel) (P-wave velocity , S-wave velocity , and density), modulus model (M-D model) (First Lame parameter λ, second Lame parameter μ, and density), and the stiffness matrix model (S-D model) (c_{11}, c_{44} and density ρ).

We first use a toy model to test the problem. Figure 6 (a), (d) and (g) are the true P wave velocity model, S wave velocity model and density model we use in this test respectively. We can see that three box anomalies are located at different positions of the layers. The

initial models are obtained through Gaussian smooth. The size of the model is 40×90, and the grid length of the model is $dx = dz = 4m$. The type of the source wavelet is Ricker's wavelet with the main frequency of $35Hz$. The source is located at every 5 grid point of the model, and every grid point at the top of the model is positioned a receiver. The total receiving time is $0.25s$, and the length for the time step is $0.0005s$. Figure 6 (c), (f) and (i) are the inversion results of P wave velocity, S wave velocity and density ρ. From the comparison of the inversion results with the true models, we can see that all the models have been correctly updated. RNN based FWI has given us the right inversion results. However, we can also see that the V_s model and ρ model have influenced the inversion result of V_p, as the black arrows pointing out. At these positions, V_p model has been incorrectly updated due to the influence of V_s model and density model. From the inversion result for V_s, we can also see that V_s has been badly influenced by density model, and ρ model has also been severely contaminated by V_s model. These are the problems caused by the cross-talk.

Table 2. Different parameterization

Model	Parameters	Physical Meaning
V-D model	V_p,V_s, ρ	P wave velocity, S wave velocity and density
M-D model	λ,μ,ρ	Lamé's first and second parameter, density
S-D model	c_{11}, c_{44}, ρ	Isotropic-elastic stiffness matrix coefficients and density

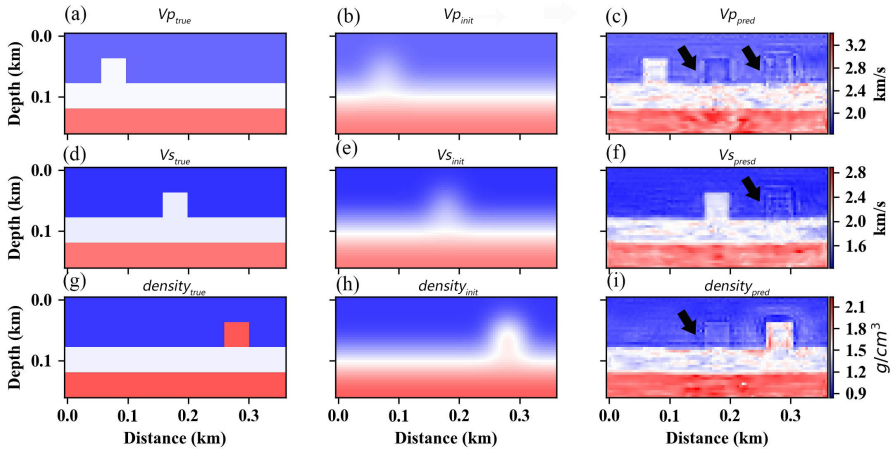

FIG. 6. V_p,V_s, ρ parameterization. (a) True V_p model, (b) Initial V_p model, (c) V_p inversion result, (d) True V_s model, (e) Initial V_s model, (f) V_s inversion result, (g) True ρ model, (h) Initial ρ model, (i) ρ inversion result

Figure 7 shows the modulus parameterization by using λ, μ and ρ parameters to represent the model, where $\lambda = V_p^2\rho - 2V_s^2\rho$, $\mu = V_s^2\rho$. By modifying the RNN cell, we can change the trainable parameters from velocity parameters to modules parameters. Figure 7 (a), (d) and (g) are the true models of λ, μ and ρ. Figure 7 (c), (f) and (i) are the inversion results of modulus parameters. The matching between the inversion results and the true models are very satisfactory. We can see the improvements of the inversion results from the comparison of modulus parameterization and velocity parameterization. The trade-off problem has been released by using the modulus parameterization. The influence of density on λ model and μ model has been mitigated. The trade-off between λ and μ has also been

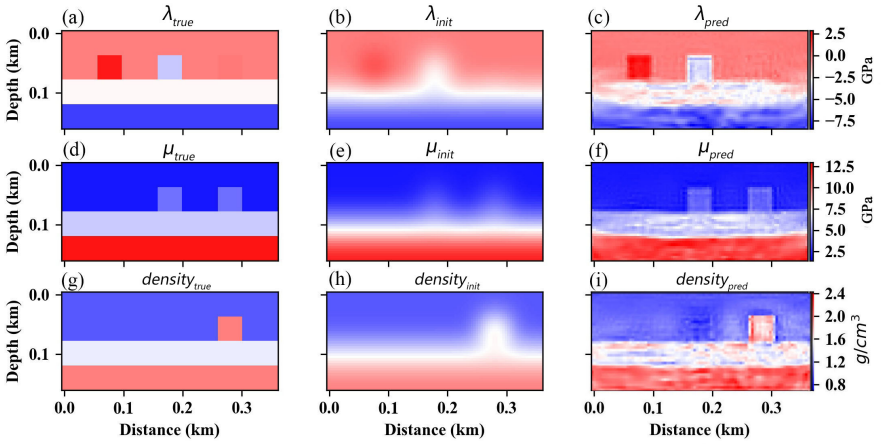

FIG. 7. λ, μ, ρ parameterization. (a) True λ model, (b) Initial λ model, (c) λ inversion result, (d) True μ model, (e) Initial μ model, (f) μ inversion result, (g) True ρ model, (h) Initial ρ model, (i) ρ inversion result

decreased. However, the density mode still suffers from the cross-talk. The influence of μ model on ρ model is obvious.

Figure 8 shows the parameterization of the stiffness matrix coefficients. We can use two stiffness matrix parameters and density to demonstrate an isotropic elastic media, which are c_{11}, c_{44}, and density ρ ($c_{11} = \lambda + 2\mu$, $c_{44} = \mu$). Figure 8 (a), (d), (g) are the true models of c_{11}, c_{44}, and ρ respectively. Figure 8 (b), (e), (h) are the initial models for c_{11}, c_{44}, and ρ respectively. Figure 8 (c), (f), (i) are the inversion results of c_{11}, c_{44}, and ρ. The models have been correctly updated by using this parameterization. Also, compared with velocity parameterization, the cross-talk problem has been mitigated as well in S-D parameterization. However, the cross-talk in density still remains, which can not be completely reduced.

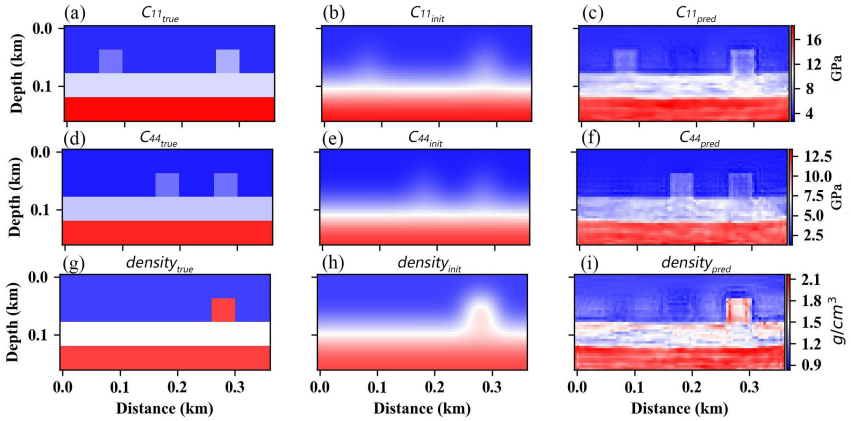

FIG. 8. Stiffness matrix parameterization (a) True c_{11} model, (b) Initial c_{11} model, (c) c_{11} inversion result, (d) True c_{44} model, (e) Initial c_{44} model, (f) c_{44} inversion result, (j) True ρ model, (k) Initial ρ model, (l) ρ inversion result.

Figure 9 shows the inversion of another model by using the M-D parameterization, and figure 10 demonstrates the inversion results of this model by using the S-D parameterization. We can see that the cross talk problem is not very severe in this model. However, we

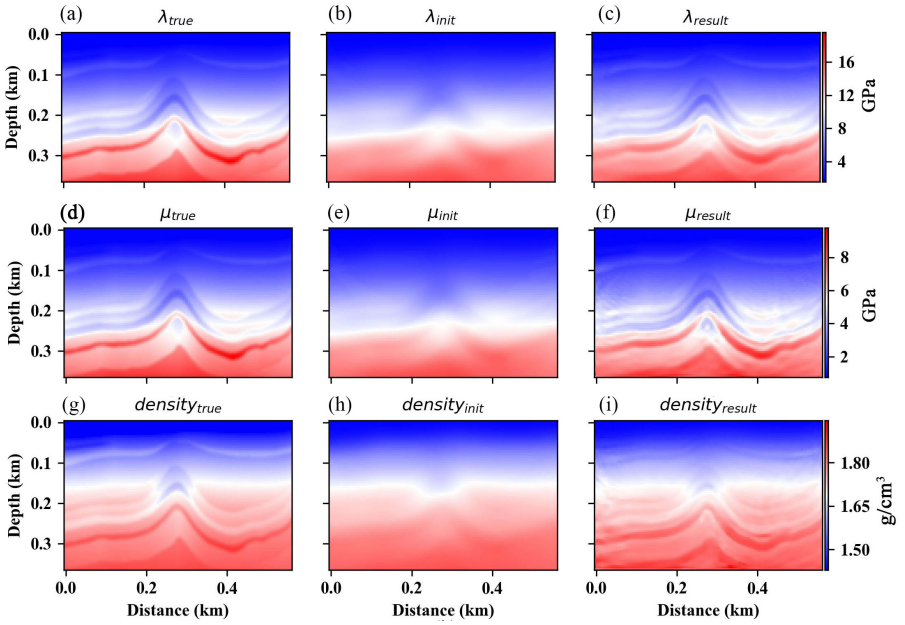

FIG. 9. Modulus parameterization (a) True λ model, (b) Initial λ model, (c) λ inversion result, (d) True μ model, (e) Initial μ model, (f) μ inversion result, (g) True ρ model, (h) Initial ρ model, (i) ρ inversion result.

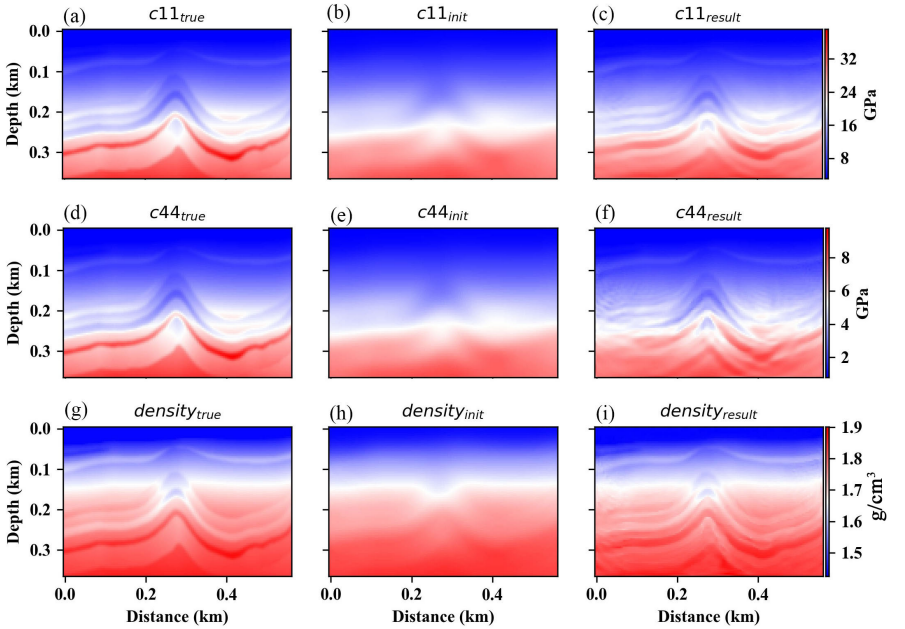

FIG. 10. Stiffness matrix parameterization (a) True c_{11} model, (b) Initial c_{11} model, (c) c_{11} inversion result, (d) True c_{44} model, (e) Initial c_{44} model, (f) c_{44} inversion result, (j) True ρ model, (k) Initial ρ model, (l) ρ inversion result.

can see that the inversion result for c_{44} is not very good compared with other parameterization. Compared with figure 5, the inversion results for density are also improved with the $M-D$ and $S-D$ parameterization. The cross-talk phenomenon, the blurred part at the top of the density model have been improved. Figure 11 shows the model misfits for V_p, λ, c_{11}, V_s, μ, c_{44} respectively. All the models have converged at the end of the inversion. We can also see that the convergence rate between V_p, λ and c_{11} are similar. Compared

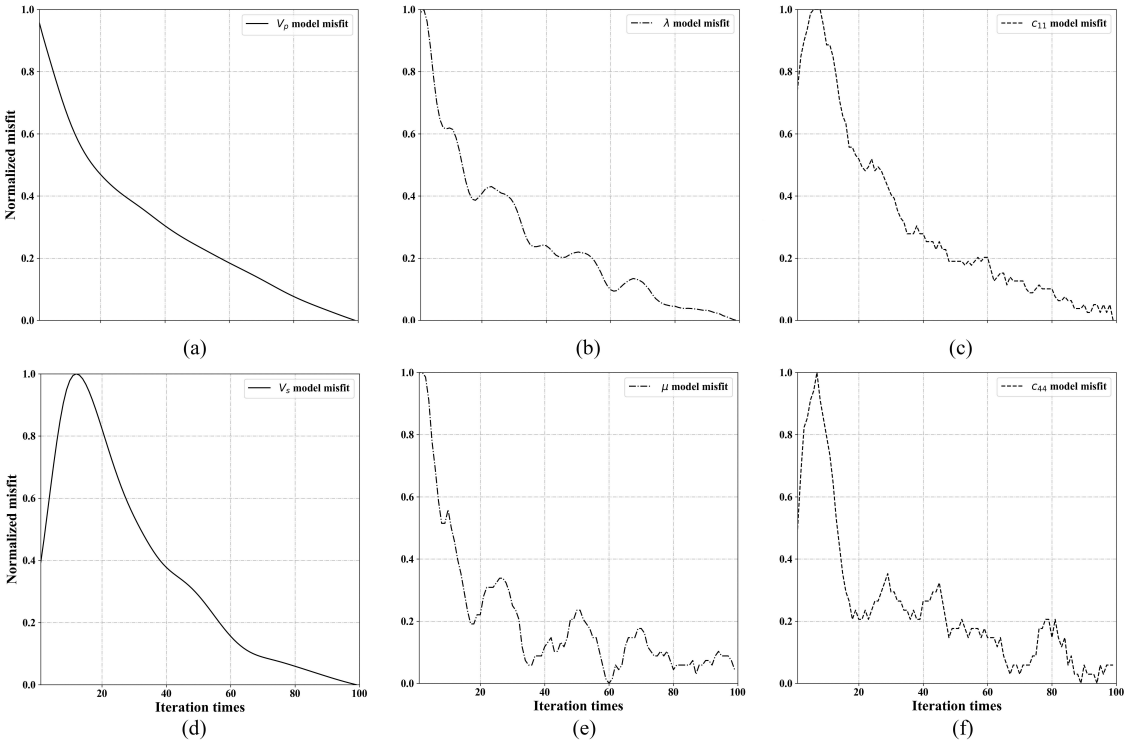

FIG. 11. Normalized model misfit (a) V_p model misfit, (b) λ model misfit, (c) c_{11} model misfit, (d) V_s model misfit, (e) μ model misfit, (f) c_{44} model misfit.

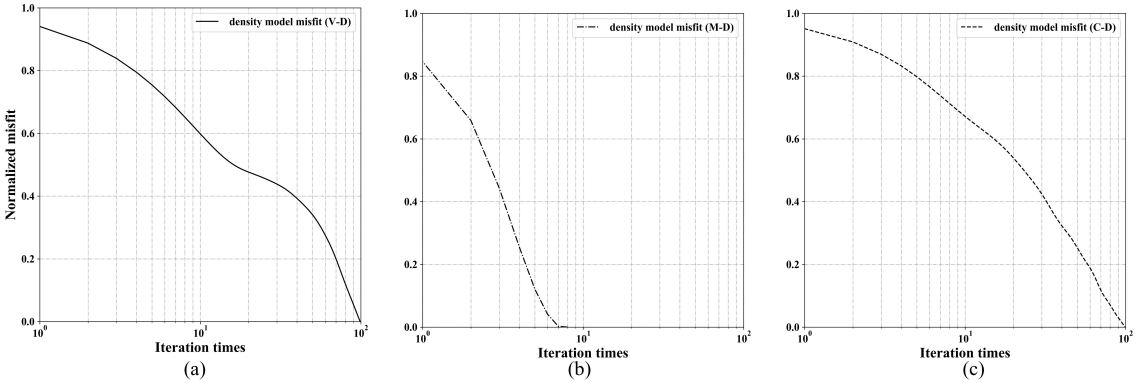

FIG. 12. Normalized density model misfit. (a) $V - D$ density misfit, (b) $M - D$ density misfit, (c) $S - D$ density misfit.

with the inversion for V_s, the inversion for μ and c_{44} have a better convergence rate, however, the inversion is much more stable by using the velocity parameterization. The model misfit curve for μ and c_{44} isolates several times before they converge. Figure 12 shows the model misfit convergence rate for density by using the three parameterizations. The decline rate for $V - D$ and $S - D$ are very similar and stable, and in this test, the density model with $M - D$ parameterization converged in around 10 iterations. Figure 13 demonstrates how the data misfits change during the iteration by these three parameterizations. The inversion by using the $M - D$ and $S - D$ parameterization have a better convergence rate for the model and data misfits, however, the inversion is more stable by using the $V - D$ parameterization.

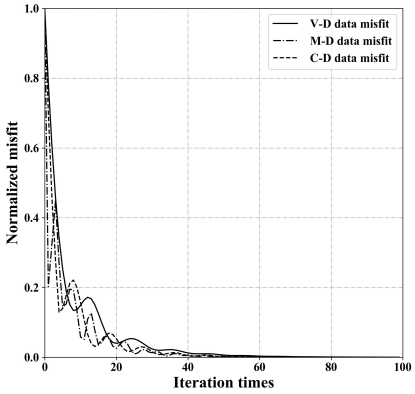

FIG. 13. Normalized data misfit. Solid line: V-D data misfit. Dotted line: S-D data misfit. Dotted solid line: M-D data misfit.

Choosing which parameterization is mainly based on the scattering pattern, however, the assumptions to calculate scattering patterns are not suitable in regular seismic data sets, for instance, the scattering pattern is calculated from an incident plane wave and background is usually considered as homogeneous. Meanwhile, the scattering pattern is usually calculated based on Born approximation, which means the higher scattering patterns are usually not considered here. Thus, in the future, we may need a more complete theory to choose which parameter we should use to tackle the cross-talk problem.

Noise stress test

In this section, we will test the sensitivity of this deep learning method we proposed with Gaussian noise. In order to show the influence of the noise on the data more clearly, in Figure 14, we plot a trace from the shot profile with different ratios of noise.

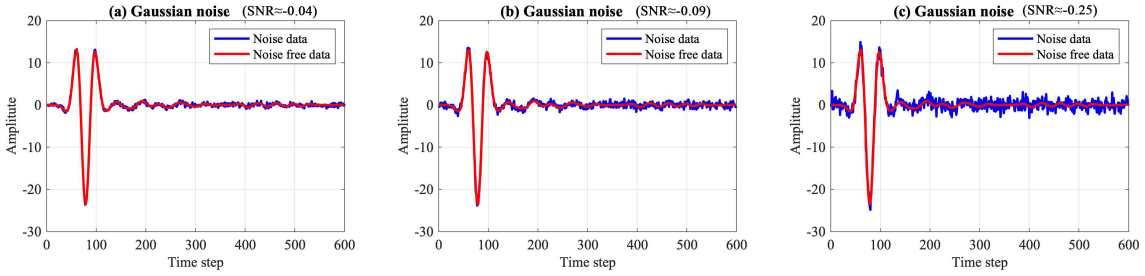

FIG. 14. Comparison between noise free data and Gaussian noise data. (a) Red line: noise free data, blue line: noise data with Gaussian noise $SNR\approx -0.04$ (d) Red line: noise free data, blue line: noise data with Gaussian noise $SNR\approx -0.09$ (g) Red line: noise free data, blue line: noise data with Gaussian noise $SNR\approx -0.25$

In figure 14 (a), the red line is the record without noise, and the blue line is the record with Gaussian random noise. The mean value of the noise is zero, and the standard divination of the noise is 0.3 (std = 0.3), and the signal to noise ratio is approximately -0.04. Figure 14 (b) shows the noise-free data and the noise data with $SNR\approx -0.09$. Figure 14 (c) shows the original data and the noise data with $SNR\approx -0.25$. From the records, we can see that as the increase of the noise level, we are having a harder time to see the useful

information from the shot records.

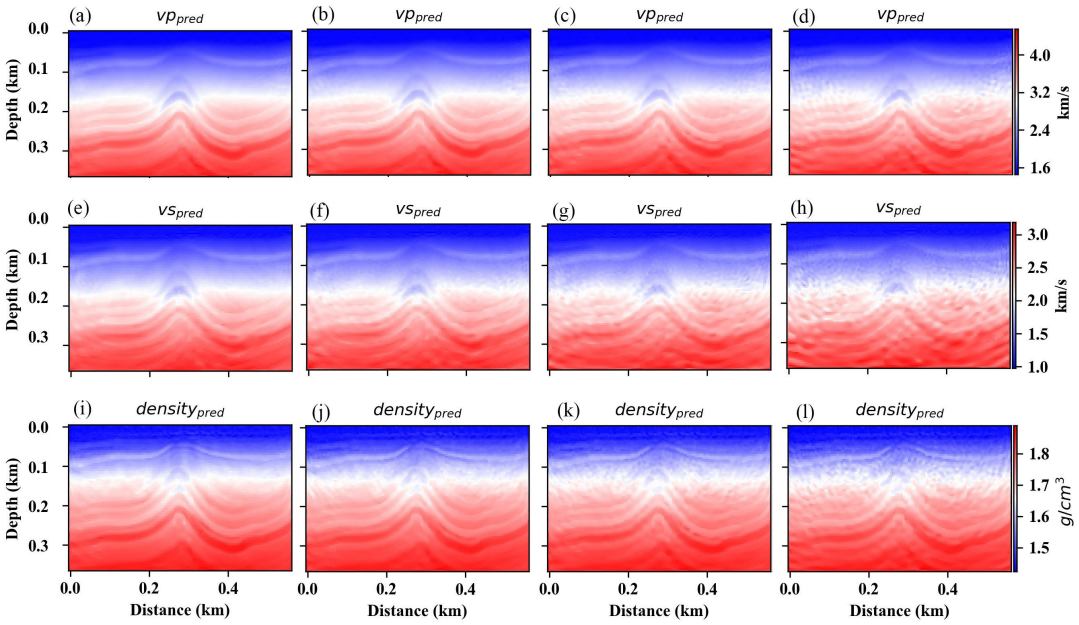

FIG. 15. (a), (c), (i) Inversion results for V_p, V_s, and ρ without noise. (b), (f), (j) Inversion result for V_p, V_s, and ρ with Gaussian noise $SNR \approx -0.04$. (c), (g), (k) Inversion result for V_p, V_s, and ρ with Gaussian noise $SNR \approx -0.09$. (d), (h), (l) Inversion result for V_p, V_s, and ρ with Gaussian noise $SNR \approx -0.25$.

Figure 15 (a), (e) and (i) show the inversion results of noise-free inversion based on RNN. Figure 15 (b), (f) and (j) show the inversion results with Gaussian noise $SNR \approx -0.04$ for V_p, V_s, and ρ model respectively. Figure 15 (c), (g) and (k) show the inversion results with Gaussian noise $SNR \approx -0.09$ for V_p, V_s, and ρ. Figure 15 (d), (h) and (l) show the inversion results with Gaussian noise $SNR \approx -0.25$ for V_p, V_s, and ρ. From the inversion results, we can see that even though, a certain amount of noise has been added to the records, the basic structure of the inversion results could still be seen, however, some layers have been blurred. Also, the inversion for V_s is much more sensitive to noise, compared with V_p and ρ This means that the inversion method we proposed is still very sensitive to noise. A small amount of noise would heavily influence the inversion results.

ACKNOWLEDGMENT

We thank the sponsors of CREWES for continued support. This work was funded by CREWES industrial sponsors, and NSERC (Natural Science and Engineering Research Council of Canada) through the grant CRDPJ 461179-13. We also thanks the support from the China Scholarship Council (CSC).

CONCLUSIONS

In this study, we introduce a theory-based deep learning framework, by using the recurrent neural network, to perform elastic full waveform inversion. The network is consisted of a series of RNN cells, and each RNN cell is coded according to the knowledge about

wave propagation. During the forward mapping process, a Dynamic Computational Graph would be built, which means the mathematical operation between different kinds of wavefields and trainable parameters would be saved in RAM. With the Automatic Differential engine, we calculate the exact gradient according to the Dynamic Computational Graph. This would avoid us from calculating the backpropagation of the residual wavefield in space. The inversion results on simple and complex models show that the inversion based on this RNN deep learning framework can provide us with promising inversion results.

To tackle to trade-off problem, we use different types of parameterization to mitigate this issue. Three different types of parameterization have been tested in this study, which are the velocity parameterization (V-D model), the modules parameterization (M-D model) and the stiffness coefficients parameterization (S-D). Inversion results show that by using different parameterization, the trade-off problem has been mitigated. However, we can not draw the universal conclusion on which one is the best parameterization for all models. The improvements made by different parameterization still various from model to model.

We also test the noise resistance level of this framework, by adding different levels of noise into the recorded data. From the inversion results, we can see that the RNN based inversion framework has a weak ability to tolerate noise. With a small amount of noise, the structure of the inversion results have been blurred.

This kind of machine learning framework is a theory-based machine learning method, which is quite different from the data based inversion. A data-based machine learning build their nonlinear relationship of the velocity model and shotrecords, by training the labeled data and the data generated by the network, and the nonliterary is represented with the weights and active function in the neural network cell. Thus, when the network is insufficiently trained or overtrained, the model can not be used as a universal situation. Also, data-based inversion requires huge amount of training data. The method we propose uses only the observe data as the labeled data, and the cells are designed according to the knowledge we know about wave propagation.

Which forms the theory-based machine learning method. It combines the knowledge we know about the forward modeling and inversion, which would give a more general solution for Geophysical inversion problems. Also, it uses the Automatic Differential method in machine learning to calculate the exact gradient based on the computational map. This kind of machine learning framework may be the pioneer to introduce more complex machine learning models into inversion problems and tackle the issues that the traditional inversion methods can not solve.

APPENDIX A

The detail of the staggered grid of the velocity stress wavefield equation can be expressed as:

$$
\begin{cases}
v_x^x|_{i,j}^{k+\frac{1}{2}} = (1-\Delta t\cdot d_x)v_x^x|_{i,j}^{k-\frac{1}{2}} + \dfrac{\Delta t}{\rho\Delta x}\displaystyle\sum_{n=1}^{N}\left\{C_n^{(N)}\left[\sigma_{xx}|_{i+\frac{2n-1}{2},j}^{k} - \sigma_{xx}|_{i-\frac{2n-1}{2},j}^{k}\right]\right\} \\
v_x^z|_{i,j}^{k+\frac{1}{2}} = (1-\Delta t\cdot d_z)v_x^z|_{i,j}^{k-\frac{1}{2}} + \dfrac{\Delta t}{\rho\Delta z}\displaystyle\sum_{n=1}^{N}\left\{C_n^{(N)}\left[\sigma_{xz}|_{i,j+\frac{2n-1}{2}}^{k} - \sigma_{xz}|_{i,j-\frac{2n-1}{2}}^{k}\right]\right\} \\
v_z^x|_{i+1/2,j+1/2}^{k+\frac{1}{2}} = (1-\Delta t\cdot d_x)v_x^z|_{i+1/2,j+1/2}^{k-\frac{1}{2}} + \dfrac{\Delta t}{\rho\Delta z}\displaystyle\sum_{n=1}^{N}\left\{C_n^{(N)}\left[\sigma_{xz}|_{i+1/2+\frac{2n-1}{2},j+1/2}^{k} - \sigma_{xz}|_{i+1/2-\frac{2n-1}{2},j+1/2}^{k}\right]\right\} \\
v_z^z|_{i+1/2,j+1/2}^{k+\frac{1}{2}} = (1-\Delta t\cdot d_z)v_z^z|_{i+1/2,j+1/2}^{k-\frac{1}{2}} + \dfrac{\Delta t}{\rho\Delta z}\displaystyle\sum_{n=1}^{N}\left\{C_n^{(N)}\left[\sigma_{zz}|_{i+1/2,j+1/2+\frac{2n-1}{2}}^{k} - \sigma_{zz}|_{i+1/2,j+1/2-\frac{2n-1}{2}}^{k}\right]\right\} \\
\sigma_{xx}^x|_{i+1/2,j}^{k+1} = (1-\Delta t\cdot d_x)\sigma_{xx}^x|_{i+1/2,j}^{k} + \dfrac{(\lambda+2\mu)\Delta t}{\Delta x}\displaystyle\sum_{n=1}^{N}\left\{C_n^{(N)}\left[v_x|_{i+1/2+\frac{2n-1}{2},j}^{k+1/2} - v_x|_{i+1/2-\frac{2n-1}{2},j+1/2}^{k+1/2}\right]\right\} \\
\sigma_{xx}^z|_{i+1/2,j}^{k+1} = (1-\Delta t\cdot d_z)\sigma_{xx}^z|_{i+1/2,j}^{k} + \dfrac{\lambda\Delta t}{\Delta z}\displaystyle\sum_{n=1}^{N}\left\{C_n^{(N)}\left[v_z|_{i+1/2,j+\frac{2n-1}{2}}^{k+1/2} - v_z|_{i+1/2,j+1/2-\frac{2n-1}{2}}^{k+1/2}\right]\right\} \\
\sigma_{zz}^x|_{i+1/2,j}^{k+1} = (1-\Delta t\cdot d_x)\sigma_{zz}^x|_{i+1/2,j}^{k} + \dfrac{\lambda\Delta t}{\Delta x}\displaystyle\sum_{n=1}^{N}\left\{C_n^{(N)}\left[v_x|_{i+1/2+\frac{2n-1}{2},j}^{k+1/2} - v_x|_{i+1/2-\frac{2n-1}{2},j+1/2}^{k+1/2}\right]\right\} \\
\sigma_{zz}^z|_{i+1/2,j}^{k+1} = (1-\Delta t\cdot d_z)\sigma_{zz}^z|_{i+1/2,j}^{k} + \dfrac{(\lambda+2\mu)\Delta t}{\Delta z}\displaystyle\sum_{n=1}^{N}\left\{C_n^{(N)}\left[v_z|_{i+1/2,j+\frac{2n-1}{2}}^{k+1/2} - v_z|_{i+1/2,j+1/2-\frac{2n-1}{2}}^{k+1/2}\right]\right\} \\
\sigma_{xz}^x|_{i,j+1/2}^{k+1} = (1-\Delta t\cdot d_x)\sigma_{xz}^x|_{i+1/2,j}^{k} + \dfrac{\mu\Delta t}{\Delta x}\displaystyle\sum_{n=1}^{N}\left\{C_n^{(N)}\left[v_z|_{i+\frac{2n-1}{2},j+1/2}^{k+1/2} - v_z|_{i-\frac{2n-1}{2},j+1/2}^{k+1/2}\right]\right\} \\
\sigma_{xz}^z|_{i,j+1/2}^{k+1} = (1-\Delta t\cdot d_z)\sigma_{xz}^x|_{i+1/2,j}^{k} + \dfrac{\mu\Delta t}{\Delta z}\displaystyle\sum_{n=1}^{N}\left\{C_n^{(N)}\left[v_z|_{i,j+1/2+\frac{2n-1}{2}}^{k+1/2} - v_z|_{i,j+1/2-\frac{2n-1}{2}}^{k+1/2}\right]\right\}
\end{cases}
\tag{16}
$$

,where the C_n^N can be expressed as:

$$
\begin{bmatrix}
1^1 & 3^1 & 5^1 & \cdots & (2N-1)^1 \\
1^3 & 3^3 & 5^3 & \cdots & (2N-1)^3 \\
1^5 & 3^5 & 5^5 & \cdots & (2N-1)^5 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1^{2N-1} & 3^{2N-1} & 5^{2N-1} & \cdots & (2N-1)^{2N-1}
\end{bmatrix}
\begin{bmatrix}
C_1^{(N)} \\ C_2^{(N)} \\ C_3^{(N)} \\ \vdots \\ C_N^{(N)}
\end{bmatrix}
=
\begin{bmatrix}
1 \\ 0 \\ 0 \\ \vdots \\ 0
\end{bmatrix}
\tag{17}
$$

and the solution is:

$$
C_m^{(N)} = \frac{(-1)^{m+1}\prod_{i=1,i\neq m}^{N}(2i-1)^2}{(2m-1)\prod_{i=1,i\neq m}^{N}|(2m-1)^2-(2i-1)^2|}
$$

APPENDIX B

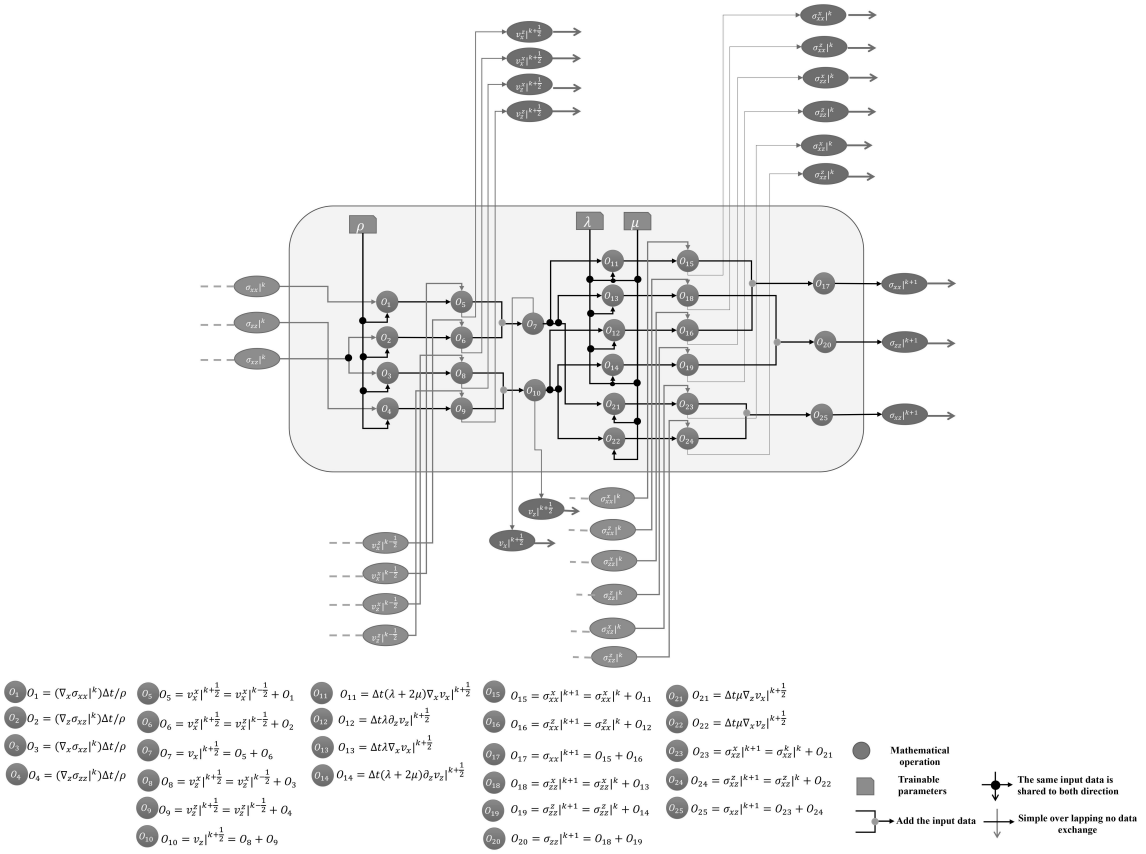

FIG. 16. Detail about the operations in isotropic elastic RNN cell

REFERENCES

Boehm, C., and Ulbrich, M., 2015, A semismooth newton-cg method for constrained parameter identification in seismic tomography: SIAM Journal on Scientific Computing, **37**, No. 5, S334–S364.

Brossier, R., Operto, S., and Virieux, J., 2009, Seismic imaging of complex onshore structures by 2d elastic frequency-domain full-waveform inversion: Geophysics, **74**, No. 6, WCC105–WCC118.

Chen, Y., Zhang, M., Bai, M., and Chen, W., 2019, Improving the signal-to-noise ratio of seismological datasets by unsupervised machine learning: Seismological Research Letters.

Downton, J. E., and Hampson, D. P., 2018, Deep neural networks to predict reservoir properties from seismic: CSEG Geoconvention.

Gholami, Y., Ribodetti, A., Brossier, R., Operto, S., and Virieux, J., 2010, Sensitivity analysis of full waveform inversion in vti media, *in* 72nd EAGE Conference and Exhibition incorporating SPE EUROPEC 2010.

Graves, A., Mohamed, A.-r., and Hinton, G., 2013, Speech recognition with deep recurrent neural networks, *in* 2013 IEEE international conference on acoustics, speech and signal processing, IEEE, 6645–6649.

Innanen, K. A., 2014, Seismic avo and the inverse hessian in precritical reflection full waveform inversion: Geophysical Journal International, **199**, No. 2, 717–734.

Jin, G., Mendoza, K., Roy, B., and Buswell, D. G., 2019, Machine learning-based fracture-hit detection algorithm using lfdas signal: The Leading Edge, **38**, No. 7, 520–524.

Kalchbrenner, N., and Blunsom, P., 2013, Recurrent continuous translation models, 1700–1709.

Köhn, D., De Nil, D., Kurzmann, A., Przebindowska, A., and Bohlen, T., 2012, On the influence of model parametrization in elastic full waveform tomography: Geophysical Journal International, **191**, No. 1, 325–345.

Li, C., Zhang, Y., and Mosher, C. C., 2019, A hybrid learning-based framework for seismic denoising: The Leading Edge, **38**, No. 7, 542–549.

Métivier, L., Brossier, R., Virieux, J., and Operto, S., 2013, Full waveform inversion and the truncated newton method: SIAM Journal on Scientific Computing, **35**, No. 2, B401–B437.

Mikolov, T., 2012, Statistical language models based on neural networks: M.Sc. thesis.

Moczo, P., Robertsson, J. O., and Eisner, L., 2007, The finite-difference time-domain method for modeling of seismic wave propagation: Advances in geophysics, **48**, 421–516.

Operto, S., Gholami, Y., Prieux, V., Ribodetti, A., Brossier, R., Metivier, L., and Virieux, J., 2013, A guided tour of multiparameter full-waveform inversion with multicomponent data: From theory to practice: The Leading Edge, **32**, No. 9, 1040–1054.

Pan, W., Innanen, K. A., Geng, Y., and Li, J., 2019, Interparameter trade-off quantification for isotropic-elastic full-waveform inversion with various model parameterizations: Geophysics, **84**, No. 2, R185–R206.

Parra, J., Parra, J., and Necsoiu, M., 2019, Nonlinear inversion of subsidence signatures induced by tunnels detected with surface and remote sensing measurements: The Leading Edge, **38**, No. 7, 550–553.

Peters, B., Haber, E., and Granek, J., 2019, Neural-networks for geophysicists and their application to seismic data interpretation: arXiv preprint arXiv:1903.11215.

Röth, G., and Tarantola, A., 1994, Neural networks and inversion of seismic data: Journal of Geophysical Research: Solid Earth, **99**, No. B4, 6753–6768.

Shustak, M., and Landa, E., 2018, Time reversal for wave refocusing and scatterer detection using machine learning: Geophysics, **83**, No. 5, T257–T263.

Smith, R., Mukerji, T., and Lupo, T., 2019, Correlating geologic and seismic data with unconventional resource production curves using machine learning: Geophysics, **84**, No. 2, O39–O47.

Sun, H., and Demanet, L., 2018, Low frequency extrapolation with deep learning, *in* SEG Technical Program Expanded Abstracts 2018, Society of Exploration Geophysicists, 2011–2015.

Sun, J., Niu, Z., Innanen, K. A., and Trad, D. O., 2019, A theory-guided deep learning formulation of seismic waveform inversion: Geophysics, **1**, No. 1.

Tarantola, A., 1986, A strategy for nonlinear elastic inversion of seismic reflection data: Geophysics, **51**, No. 10, 1893–1903.

Wu, Y., and Lin, Y., 2018, Inversionnet: A real-time and accurate full waveform inversion with cnns and continuous crfs: arXiv preprint arXiv:1811.07875.

Yang, F., and Ma, J., 2019, Deep-learning inversion: a next generation seismic velocity-model building method: Geophysics, **84**, No. 4, 1–133.

Yang, J., Liu, Y., and Dong, L., 2016, Simultaneous estimation of velocity and density in acoustic multiparameter full-waveform inversion using an improved scattering-integral approach: Geophysics, **81**, No. 6, R399–R415.

Zaremba, W., Sutskever, I., and Vinyals, O., 2014, Recurrent neural network regularization: arXiv preprint arXiv:1409.2329.

Zhang, H., Yang, X., and Ma, J., 2019a, Can learning from natural image denoising be used for seismic data interpolation?: arXiv preprint arXiv:1902.10379.

Zhang, Z., Wu, Y., Zhou, Z., and Lin, Y., 2019b, Velocitygan: Subsurface velocity image estimation using conditional adversarial networks, *in* 2019 IEEE Winter Conference on Applications of Computer Vision (WACV), IEEE, 705–714.

Zheng, Y., Zhang, Q., Yusifov, A., and Shi, Y., 2019, Applications of supervised deep learning for seismic interpretation and inversion: The Leading Edge, **38**, No. 7, 526–533.