An implicit neural representation for full waveform inversion

Tianze Zhang, Jian Sun, Daniel Trad and Kris Innanen

ABSTRACT

We introduce and analyze implicit full waveform inversion (IFWI), which uses a neural network to generate velocity models and perform full waveform inversion. IFWI has two main parts: a neural network that generates velocity models, and a recurrent neural network FWI to perform the inversion. IFWI is distinct from conventional waveform inversion in two key ways. First, it does not require an initial model as does conventional FWI. Instead, it requires general information about the target area, for instance means and standard deviations of medium properties in the target area, or alternatively well-log information in the target area. Second, within IFWI, we update the weights in the neural network, unlike in the conventional FWI, which updates the velocity model directly. The neural network we use to generate velocity models is a fully connected set of sinusoidal activation layers, which has been shown to outperform $Relu$ and $tanh$ because of its ability to learn high-order spatial derivatives. Through numerical tests, we demonstrate that, by controlling the random initialization of the weights in the network and the scale of the velocity the network generated, the IFWI can in principle build accurate models in the absence of an initial model. In practice IFWI itself may be a useful tool for building initial models for conventional or high-frequency FWI.

INTRODUCTION

Full waveform inversion (FWI) is an optimization based inverse procedure with the capacity to produce high-resolution velocity models and subsurface images (Virieux and Operto, 2009). Although very promising as a technology for exploration and monitoring, practical FWI still faces several key challenges. On land, adequate near-surface models typically require elastic or viscoelastic propagation models to be considered, and within these, accurately-determined shallow heterogeneities (Teodor et al., 2021). However in multi-parameter inversion, different sensitivities across parameter classes cause crosstalk in reconstructed models (e.g., Keating and Innanen, 2020, 2019). Even in single-parameter problems challenges arise. The commonly used ℓ_2 norm objective function typically contains many local minima, requiring either very accurate initial models or high-fidelity broadband data (Lailly and Bednar, 1983; Tarantola, 1984; Tromp et al., 2005; Plessix, 2006; Virieux and Operto, 2009).

A great deal of recent research has been focused on leveraging machine learning (ML) techniques to address these FWI challenges. For instance Sun and Alkhalifah (2020) used a recurrent neural network to train an optimization method for FWI; a convolution neural network has been used to fill in missing low frequency information to stabilize FWI (Sun and Demanet, 2020); also Zhang and Alkhalifah (2020) has developed a deep neural network for reservoir characterization. Directly merging FWI and ML is not straightforward, but natural linkages exist. For instance, Richardson (2018) and Sun et al. (2020) formulated waveform inversion as a training procedure in a constrained recursive neural network, and Sun et al. showed that training such a network with a single data set is equivalent to

conventional FWI. Subsequently Zhang et al. (2020, 2021)produced a significant extension of this work to multiparameter elastic inversion with a range of optimizers. This RNN approach is an example of theory-guided machine learning, in which the neural network is structured based on prior knowledge, and the desired medium parameters are the trainable weights in the network.

Among these various kinds of the neural network, the coordinate-based multilayer perceptron (MLP) appears to be of particular interest. Coordinate-based MLP takes low-dimensional coordinates as input (usually in $\mathbb{R}^2$ or $\mathbb{R}^3$) and are trained to output a representation of the targets (Tancik et al., 2020). These networks (e.g., Rahaman et al., 2019; Stanley, 2007; Genova et al., 2020) have been developed and applied in image representation problems, shape representation in texture synthesis, shape inference from images, and novel view synthesis, and have achieved state-of-the-art results (Rahaman et al., 2019; Stanley, 2007; Genova et al., 2020; Park et al., 2019; Liu et al., 2019; Sitzmann et al., 2019). The property networks of this kind most often report is a very strong ability to resolve low frequencies in the loss function; difficulties are reported in its ability to resolve high frequency information (Basri et al., 2020; Rahaman et al., 2019). We suggest that therefore such networks hold promise in complementing FWI, which conventionally recovers high-resolution details well but requires an accurate long-wavelength initial model.

In coordinate-based MLP networks, initialization and activation function choices have a strong influence on the accuracy and the convergence properties of the training step. A proper initialization of the weights and activation function can force the data to be generated within a desirable range, which in turn tends to have a positive impact on convergence. Conventionally, activation functions for neural networks are the Relu, Sigmoid, or Tanh, but Sitzmann et al. (2020) has demonstrated that the MLP with periodic sinusoidal activation function is better able to recover derivatives of the target image compared with Relu and Tanh.

In this report, we set up what we call an implicit full waveform inversion (IFWI) inversion method. This involves combining a coordinate abased MLP network, activated with sinusoidal activation functions, with a recurrent neural network FWI. The IFWI set up like this proceeds in the manner of a waveform inversion, but does not require an explicit initial model to begin. The MLP takes in the spatial positions of the grid cells associated with the model we wish to reconstruct as coordinate information, and, through judicious initialization of the weights and some degree of prior model information, generates a range of velocity models with expected statistical properties. These velocity models are then sent to the RNN and through it data are simulated. The residual between the observed and simulated data are then sent back to the MLP, through the RNN, and updating occurs. The key is that the iterative procedure involves updating the weights of the MLP, which in turn produces the suite of velocity models based on these, as opposed to updating the velocity model explicitly.

We carry out a range of numerical tests to develop the behaviour of this IFWI scheme. There is significant freedom in applying IFWI. Medium property models can be generated in a variety of ways, differing in their general or in detailed features. We observe that shallow model structures tend to be well-recovered in particular. Although the method is in

its early stages of analysis, this is suggestive that it in the presence of surface seismic data it may have particular applicability in producing shallow model information for initialization of conventional FWI workflows.

THEORY

Implicit neural representation with sinusoidal activation layer

The neural network we use in this study is a fully connected layer with the activation of the sin function, which can be formulated as:

$$h_i(\mathbf{X}) = \phi_i(\mathbf{W}_i\mathbf{X}_i + \mathbf{b}_i) \tag{1}$$

Here, $h_i : \mathbb{R}^{m_i} \rightarrow \mathbb{R}^{n_i}$ means the i^{th} layer output, with $\mathbf{X}_i \in \mathbb{R}^{m_i}$ as the input. The transform of h_i is defined with the weight matrix $\mathbf{W}_i \in \mathbb{R}^{n_i \times m_i}$ and the bias $\mathbf{b}_i \in \mathbb{R}^{n_i}$. ϕ_i is the sinusoidal activation function. The network with sinusoidal activations has been demonstrated to produce favourable convergence properties. In each layer, the data can be generated such that it falls within a specified range, with the mean and the standard deviation of the network changing though out the network. Suppose that we have four layers of the network, and in the first layer, we initialize the weight matrix with uniform distribution form $[-1, 1]$. The weights in other layers are initialized uniformly in the interval $[-c, c]$, and we use a linear range of 256 points in the range of $[-1, 1]$ as input and plot the histogram of each linear transform and sinusoidal activation. We show in Figure 5 that the activation throughout the networks is standard normal distributed before each sine non-linearity and arc-sine distributed after each sine non-linearity, irrespective of the depth of the network. We will discuss this property further in the next section.

Training a network to predict images or signals with a fully connected neural network activated with sinusoidal functions, is equivalent to using a series of sinusoidal functions with different frequencies and phases to represent models and images. This is similar to a Fourier-based method wherein sine and cosine functions with different amplitudes, frequencies, and phases represent complex signals. In equation (1), $\mathbf{W}_i$ controls the range of the frequencies of the sinusoidal functions that we use to represent the outputs, and the bias $\mathbf{b}_i$ is the phase.

Neural tangent kernel

The neural tangent kernel (NTK) is a quantity that captures the approximate dynamics of the neural network during training. Suppose that, like in equation (1), h is the fully-connected deep neural network which is parameterized with weights θ initialized with Gaussian distribution $\mathcal{N}$.Tancik et al. (2020) indicates that if we have infinite numbers of layers for the fully-connected network, optimized with the gradient descent method, the convergence property of the network can be described with the neural tangent kernel, which is defined as:

$$k_{NTK}(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle \tag{2}$$

$\phi(x) = \partial_\theta h(x; \theta)$, which is the partial derivative of the activation function with respect to parameters, and $\langle :, : \rangle$ means inner product. x_i and x_j represents two input values.

The NTK is used to approximate the dynamics of a deep network during training, which brings us the detail of how the loss would change during training. Tancik et al. (2020) gives the variation of the training error $\mathbf{U} = \hat{\mathbf{Y}} - \mathbf{Y}$, with iteration time (approximating it as a continuous variable) as

$$\partial_t \mathbf{U} = -\mathbf{K}\mathbf{U}, \tag{3}$$

where the positive semidefinite matrix $\mathbf{K} \in \mathbb{R}^{n\times n}$ is the neural tangent kernel of the network. $\hat{\mathbf{Y}}$ and $\mathbf{Y}$ are the output of the network and the labeled data respectively. $\mathbf{U}, \hat{\mathbf{Y}}, \mathbf{Y} \in \mathbb{R}^n$

Let the eigen-decomposition of the kernel be $\mathbf{K} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T = \sum_{n=1}^{n} \lambda_i V_i V_i^T$. We express the training loss $\mathbf{U}$ by using the eigen vector with $\mathbf{U} = \sum_{n=1}^{n} e_i V_i$, where the e_i means the i^{th} components of the training loss,we have $\partial_t \sum_{i=1}^{n} e_i V_i = -\sum_{i=1}^{n} \lambda_i V_i V_i^T \sum_{i=1}^{n} e_i V_i$, based on the orthogonal properties of the eigen vectors and removing the summation notation, we have:

$$\partial_t e_i = -e_i \lambda_i; \rightarrow e_i = e^{-\lambda_i t} + C \tag{4}$$

Equation (4) demonstrate a very important conclusion for this study. It shows that, if we consider training convergence in the eigen-basis of the NTK, the i^{th} components of the training loss will decay exponentially with the rate of λ_i, indicating components of the target function that correspond to kernel eigenvectors with larger eigenvalues will decay faster compared with components with smaller eigenvalue. This means that the components loss function's eigenvectors with larger eigenvalues will be learned faster.

Implicit FWI: constructing velocity models with neural network

The inversion procedure is driven by the idea of "implicit FWI", or IFWI, wherein what is solved for are weights in a network that more and more effectively produces realizations of velocity models. This distinguishes it from conventional FWI, in which the velocity model parameters are solved for explicitly. IFWI consists of two parts. The first part is a fully connected neural network with sinusoidal activation functions. The input of the network is the coordinate information of the model informing the size of the model we aim to generate. Thus, this is actually a coordinate-based multilayer perceptron (MLP). The output would be the velocity model, which would be sent to the second part of the network. The second part of IFWI, is an RNN forward modeling method. After the forward modeling, the residuals between the synthetic data and the observed data would be calculated. The update would be performed on the weights in the network, unlike the velocity model like the conventional FWI. Thus, we are actually training a fully connected network that can generate velocity models with a very complex objective function. Since in the network objective function, we use the wave equation to project the velocity model from model space to the seismic data space and calculate the norm of the residuals there.

With the conclusion from the last section, we can discuss the rationality of the IFW network. It is because that the eigenvectors with larger eigenvalues always correspond to the larger scale or low frequencies information, and smaller eigenvalues always correspond to the small scale or high frequencies information. If the loss function's eigenvectors with larger eigenvalues are always learned faster. It means that the low frequencies information of the objective function would be first recovered. This is exactly very reasonable for FWI,

because low frequencies information is critical for reducing the non-linearity of FWI. The multi-scale FWI methods are doing the same process. In multi-scale FWI, we usually first extract low frequencies information from the data and perform the inversion. The higher frequency components would be later included in the inversion. This means that performing the inversion with the IFW alliances with the philosophy of multi-scale FWI.

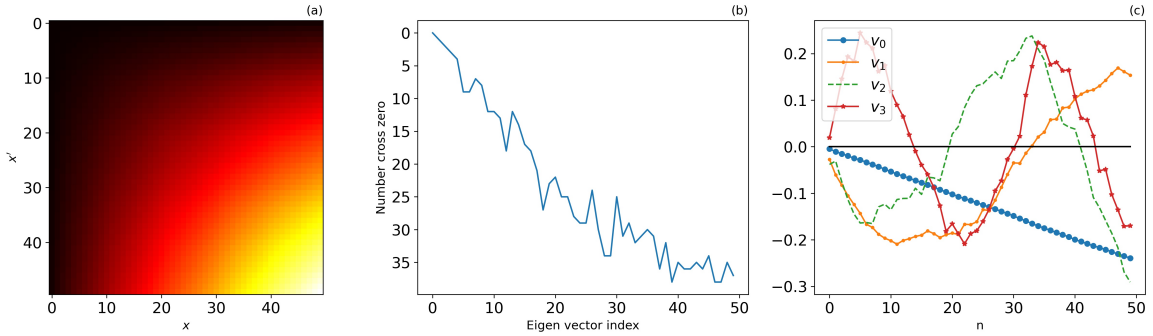

FIG. 1. The NTK analysis of a two layer fully connected layer with Relu activation function. (a) The NTK kernel of Relu function. (b) The plot of numbers for eigenvector of the Relu NTK crossing the zero point indicating the oscillation properties of the eigenvector (index 0 stands for the largest eigenvalue). (c) The plot of the eigenvectors fpr V_0, V_1,V_2,V_3 for Relu NTK, with eigenvalue $\lambda_0 > \lambda_1 > \lambda_2 > \lambda_3$

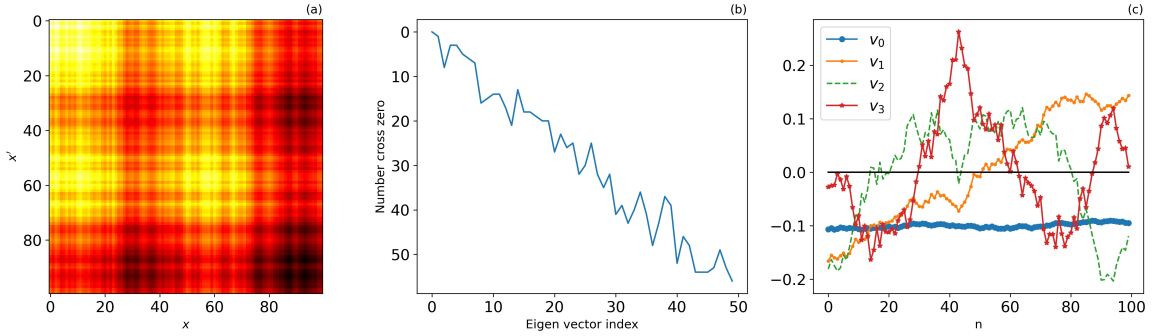

FIG. 2. The NTK analysis of a two layer fully connected layer with sin activation function. (a) The NTK kernel of sine function. (b) The plot of numbers for eigenvector of the sin NTK crossing the zero point indicating the oscillation properties of the eigenvector (index 0 stands for the largest eigenvalue). (c) The plot of the eigenvectors for V_0, V_1,V_2,V_3 for Relu NTK, with eigenvalue $\lambda_0 > \lambda_1 > \lambda_2 > \lambda_3$

To further explain our analysis, we compute two the neural tangent kernel for two simple neural networks and do the eigen-decomposition of the NTK. Figure 1 (a) shows the NTK for a network with Relu activation function consisting of two neurons. Figure 1 (b) shows the oscillation property of the eigenvectors. The vertical axis shows the number of the eigenvector passing the zero point, and the horizontal axis shows the index of the eigen vector (index 0 has the largest eigenvalue and index 49 has the smallest eigenvalue). We can see that the eigenvector shows a stronger oscillation as the eigenvalue becomes smaller. Figure 1 (c) we plot the four of the eigenvectors, V_0, V_1,V_2 and V_3, their corresponding eigenvalues are $\lambda_0 > \lambda_1 > \lambda_2 > \lambda_3$. We can see that a bigger eigenvalue corresponds to a eigenvector with small little oscillation property, and as the eigenvalue becomes smaller, the eigenvector would show more oscillations. We observed the same phenomenon in 2, which is the neural tangent kernel analysis with a neural network activated with the sin

function, (Figure 2, and Figure 1 shares the same initialization weights and input). The major differences between these two figures are the NTK. With the same weight and input, Figure 1 shows a larger area with small NTK values compared with Figure 2. It means that the neural network with Relu function is less sensitive with some input data compared with the sin activation neural network. Thus these two figures further explained how NTK demonstrates the dynamic behavior of the neural network.

To further clarify our statement, we train a coordinate-based MLP with sin activation functions to generate a single image of the velocity model. The objective function of the network is the ℓ_2 norm of the image generated with the network and the target velocity model. The maximum iteration time is 200 times, and we plot the prediction results in Figure 3 for different training epochs. We can see that the network gradually generates the model with information from general to detail, which alliances with our discussion in the last paragraph.

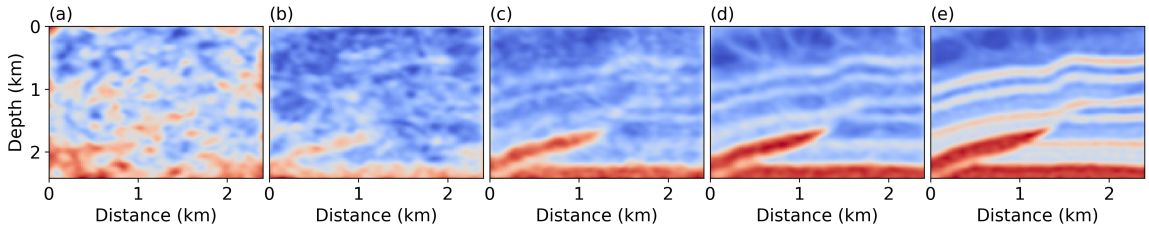

FIG. 3. The prediction results of training a network to generate a velocity model in different epoch. From left to right are the training prediction with, (a) 10 epoch, (b)40 epoch,(c) 80 epoch, (d) 120 epoch, (e) 200 epoch. In the early stage of the iteration, the network tends to recover the general information of the target image, corresponding to our discussion that the eigenvectors with larger eigenvalues are learned faster, which gives the general information for the model.

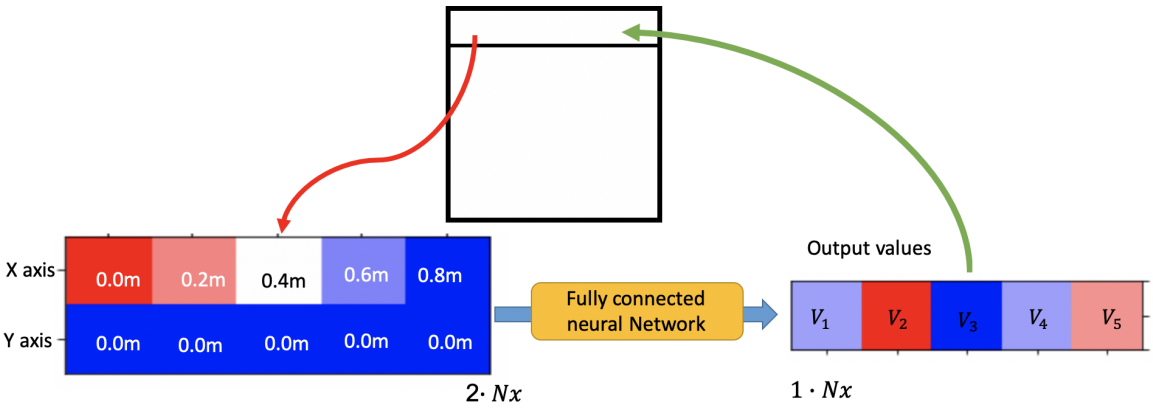

FIG. 4. The input of the network are the X and Y coordinates, in this figure, the matrix with the dimension of 2$\times$Nx. This matrix represents all the points in the first row of the velocity mode. We take them into the neural network and it will generate the model with the dimension of 1$\times$Nx. The output would be put into the first row of the velocity model, If we do this for very row then we can using the coordinate information to generate a velocity model. This velocity model would be sent into the RNN to calculate synthetic data and weights in the neural network would be updated according to the residual.

The weight initialization of neural network

Note that building the coordinate-based MLP without carefully chosen weights yielded poor performance both in accuracy and in convergence speed. The key idea in our initialization scheme is to preserve the distribution of activation through the network so that the final output at initialization does not depend on the number of layers. In this study, we use the sin function as the activation layer. Thus, according to equation 1, weights we generated determine the range of the frequencies of the sin function that we use to represent imagines. Figure 5 shows the initialization of the weights of a four-layer network and the histogram of the data before and after the sin activation function. In the first layer, the weights are generated with a Uniform distribution, $\mathbf{W}_1 \sim \mathcal{U}(-1, 1)$, and in the rest of the network, the weights are initialized with $\mathbf{W}_{2\sim4} \sim \mathcal{U}(-\sqrt{\frac{c}{n}}, -\sqrt{\frac{c}{n}})$, where n is the number of the weights in the inversion and c is the hyper-parameter of the network. In Figure 5, $c = 6$, we can see that, in each layer, the data generated before the sin activation is the Gaussian's distribution. After the sin activation layer, it becomes the U shape arcsine distribution. Such a pattern has been preserved throughout the following layer. We can see that in the first column, except for the first layer, the range of the weights are approximately from $-5 \times 10^3 \sim 5 \times 10^3$, It means that the frequencies we use here are very narrow. Figure 6 shows the histograms of the network with $c = 20$. We can see that, with a larger value of c, a broader range of the weights are generated, indicating we are using a wider range of frequencies with sin functions for the network to generate velocity models. It is helpful for the recovering of the details of our target. Also, we can see that the distribution of the output data remains the same throughout the network.

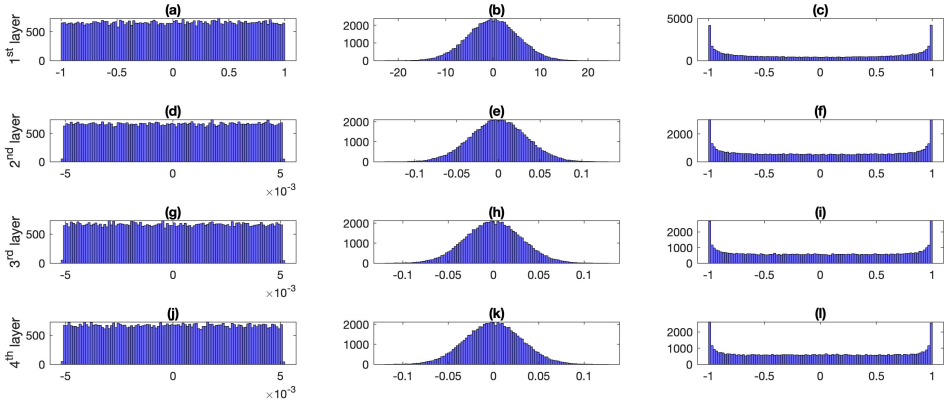

FIG. 5. Activation statistics of a four layer neural network with $c = 6$. The first column: the initialization of the weights $\mathbf{W}_i, i \in (1, 2, 3, 4)$. The second column: the histogram of $\mathbf{W}_i^T \mathbf{X}_i$, where $\mathbf{X}_i$ is the input of the layer, before sin activation. Third column: the histogram of $(\mathbf{W}_i^T \mathbf{X}_i)$ after sin activation.

To further demonstrate the influence of the initialization, we did the experiments in Figure 3 again with a different c value. The maximum iteration time is 200 times, and we test five c values with, $c = 1$, $c = 3$, $c = 9$, $c = 15$, $c = 20$, and the prediction results are plotted in Figure 7. We a relatively small c we only recover the general information of the velocity model, and as the increase of the c value, we can see that more details of the model have been reconstructed.

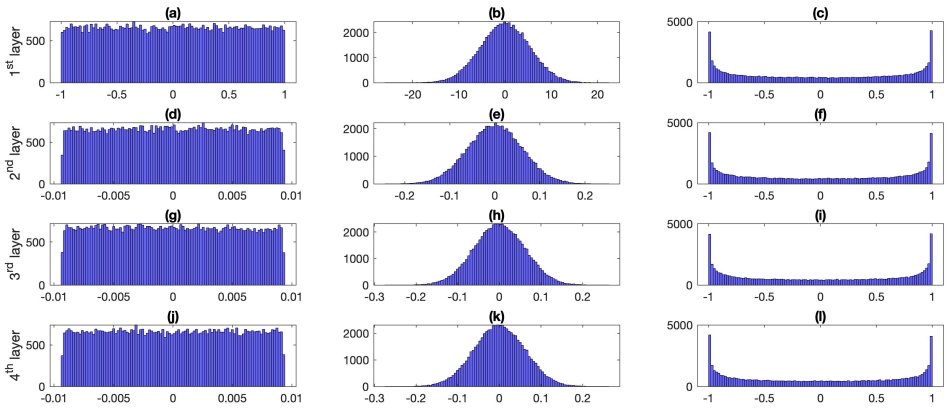

FIG. 6. Activation statistics of a four layer neural network with $c = 20$. The distribution of in each layer alliance with Figure 5, however, with a different c value a broader ranges of weights are generated meaning we can use more frequencies of sin functions to represent the target.

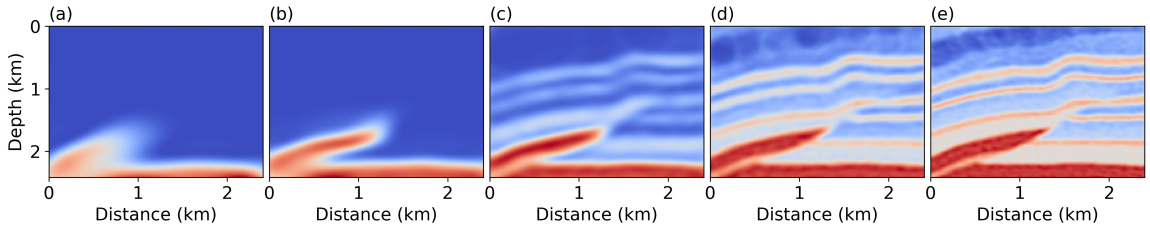

FIG. 7. The prediction result training a network to generate a velocity model with different c value with a four layer coordinate based MLP. (a) $c = 1$, (b) $c = 3$, (c) $c = 9$, (d) $c = 15$, (e) $c = 20$. With a broader range of weights are generated, more detail of the image can be resolved.

NUMERICAL TESTS

In this section, we will use the IFWI to perform the elastic full waveform inversion. The neural network is a four-layers fully connected neural network with sin activation function that takes in the grid coordinate information in the x and z direction as input, and the outputs of the network are the velocity models that would be sent to the RNN to perform forward modeling obtaining the synthetic data. The residual between the observed data and the synthetic data is calculated with the ℓ_2 norm. The residual would be sent back to the fully connected neural networks through the RNN, to update the weights in the fully connected network. Thus, in this inversion test, we update the weights in the neural network, not the velocity models generated with networks. The 2D VP model is obtained through the 3D Overthrust model, and the VS and density models are calculated through scaling the true VP model. The size of elastic models is 125×125, and we use $dx = dz = 20m$ as the grad length of the model. The wavelet is Ricker's directional wavelet with the main frequency of $15Hz$. The maximum receiving time is 2.6 seconds, with the $dt = 0.002$. We use the staggered grid stress velocity finite difference method with 10 layers of PML boundaries to calculate the synthetic data. The maximum iteration time is 2000 times.

Figure 8 shows inversion results with the surface acquisition, which means that the source and the receivers are located on the surface of the model. The subfigures in Figure 8 from left to right, in columns, are the inversion results at 60, 100, 200, 500, and 2000

iteration, and from top to bottom, in rows, are VP, VS, and density. We can see that after 60 iterations, the network could actually generate the general background of the model, and arbitrarily the first layer of VS can be roughly seen. After 100 iterations, we can see some geological-meaningful structures at $1km$ depth in the models generated with the network. The areas that are near the source tends to have more update than other areas. As for the inversion results at 200 iterations, the general upper structures, with depth less than $1km$ of the model, have been recovered. After 500 iterations, more details are added to the velocity models. The inversion results at 2000 iteration do not show much more updates compared with the inversion results after 500 iterations. Figure 9 shows the vertical inversion profile of the inversion results for VP, VS, and ρ, and we plot the inversion results at 20, 200, and 500 iterations at $2km$ distance of the model to see how the inversion evolves throughout the iteration time. We can see that at iteration 20 (yellow line), we can only generate a velocity model that has the mean value of the model. As the increase of the iteration time, the network progressively adds the details into the velocity model. However, we can see that the inversion shows poor resolution at around $2km$ of the model.

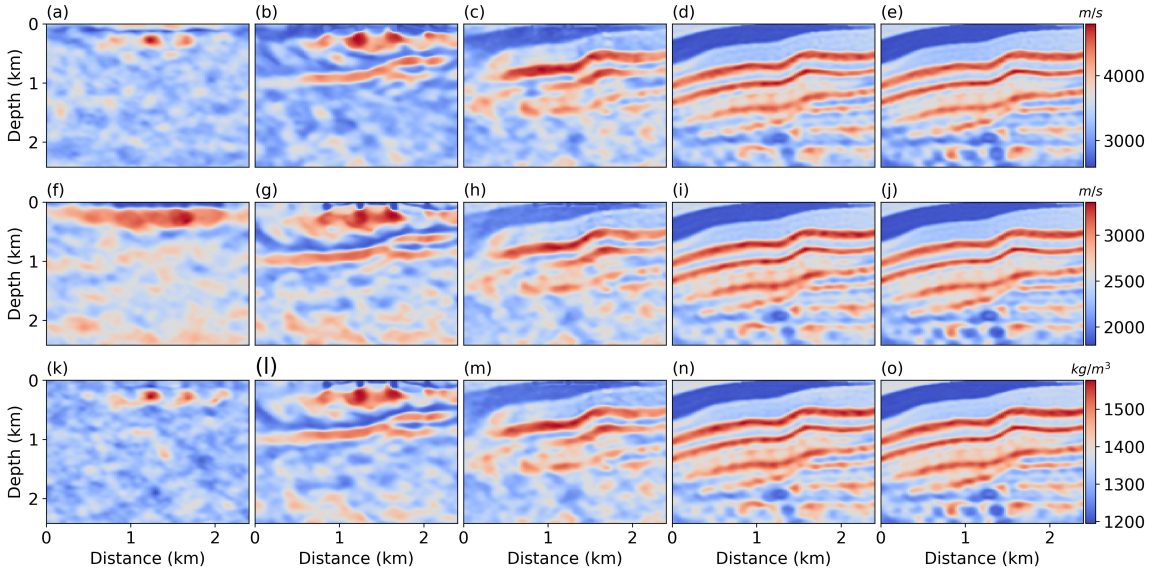

FIG. 8. Elastic IFW inversion results with surface acquisition. Figure (a)-(e) shows the inversion results of the VP model at 60, 100, 200, 500 and 2000 epochs. The second and the third row shows the inversion results for VS and density.

We test another acquisition system that has a vertical well of shots at around $2.5km$, in the distance, in the model. The interval of shots in the well is $z = 30$ grid points which means that every $600m$ in the well we have a shot. The receivers are still located on the surface. All other variables in this test are the same as the surface acquisition test. The inversion results are shown in Figure 9. In Figure 9 we observed a similar pattern of how the velocity model generated with the network updates from general to detail. We observe that in the well acquisition tends to produce more updates around the well, while the surface acquisition test tends to first update around the surface. Compared with the inversion results at 200 iterations, (c),(h),(m) in Figure 8 and 10, it shows that the layers around the sources are better resolved as the red ecliptic marks. Also, we can see that in the final results of Figure (e), (j), (0). The bottom layer has been better recovered as marked in the plot. Figure 11 shows the vertical profile of the inversion results at $2km$ of the model.

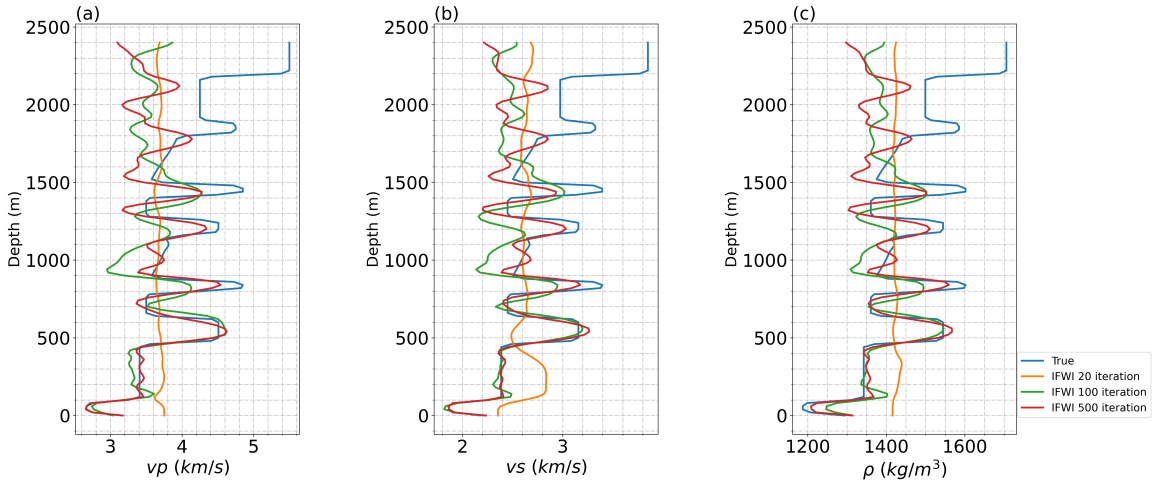

FIG. 9. Vertical profile of the IFWI surface acquisition results for VP, VS and density at 20, 200, and 500 for VP, VS and density respectively.

We can see the improvements made with the well at around $2km$ in the depth of the model, which is a better fit for the velocity model compared with Figure 9. Although both the tests fail to recover the bottom part of the inversion, we still consider this is a successful recovery for the elastic velocities. Because we do not have any initial models, the IFWI we introduce could become a valid method that provides the larger scale initial model for the conventional FWI.

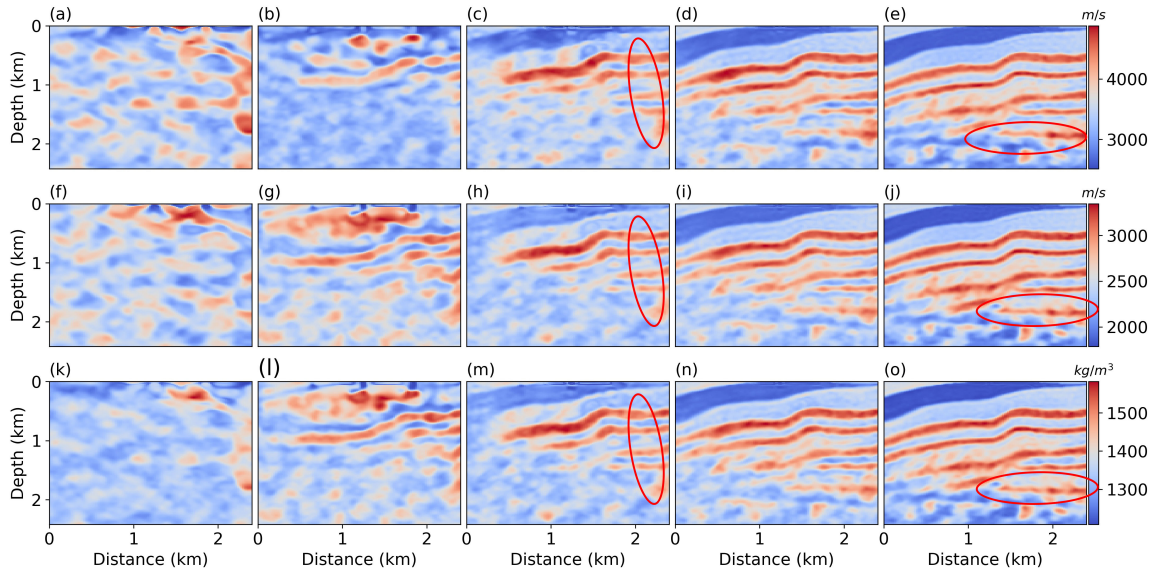

FIG. 10. Elastic IFW inversion results with surface and well acquisition. Figure (a)-(e) shows the inversion results of the VP model at 60, 100, 200, 500 and 2000 epochs. The second and the third row shows the inversion results for VS and density.

COMPUTATIONAL COST

All our inversion tests are calculated with GPU provided by ARC Cluster in the University of Calgary with Nvidia Tesla V100 (16GB). In the surface acquisition test, we use 6 shots on the surface, and the maximum receiving time is 2.6s with 1300 time steps. It takes

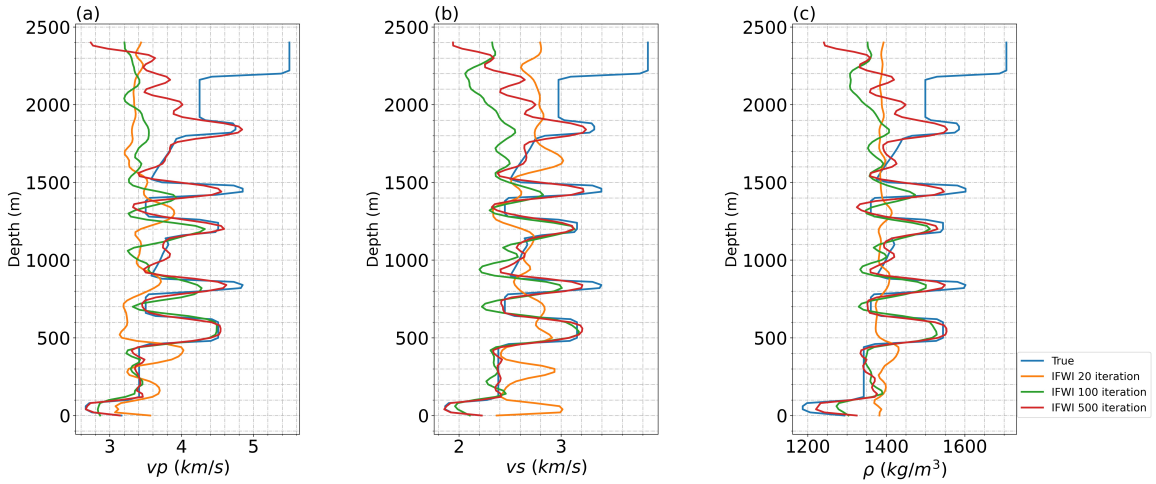

FIG. 11. Vertical profile of the IFWI with surface and well acquisition results for VP, VS and density at 20, 200, and 500 for VP, VS and density respectively.

about 73% of the total RAM, 11.68G, on the GPU. The surface acquisition inversion test takes about 4 hours, 43 minutes, and 21 seconds to finish 2000 iterations. In the well log acquisition test, we use 4 shots on the surface and 4 shots in the well, also with a maximum receiving time 2.6s with 1300 time steps. It uses approximately 87%, 13.92G, of the total RAM. And it takes 5 hours, 23 minutes and 47 seconds to finish 2000 iterations.

CONCLUSION

In this study, we introduce the implicit FWI to perform inversion without using an initial model. The stricture of the network consists of two parts. The first part is a coordinated-based MLP with the sinusoidal activation function. And the second part is an RNN based neural network forward modeling method. The velocity model generated with the MLP would be sent into the RNN for obtaining the synthetic data, and the residuals between the observed data and the synthetic data would be sent back to the MLP, through the RNN, to update the weights in the MLP. The analysis of the training for MLP indicates that the components of the target function that correspond to kernel eigenvectors with larger eigenvalues will decay faster compared with components with smaller eigenvalue. This means that the components loss function's eigenvectors with larger eigenvalues will be learned faster. This explains why the training with MLP has a good ability to provide general information in the velocity model for inversion. We also demonstrate how the initialization of the weights could influence the approximation of training. The Numerical test section shows how the velocity model generated with the MLP evolves throughout iterations. The Computational cost section shows that IFW method is a computational affordable inversion method.

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 543578-19. The first author is also supported by the Chine Scholarship Council (CSC).

REFERENCES

Basri, R., Galun, M., Geifman, A., Jacobs, D., Kasten, Y., and Kritchman, S., 2020, Frequency bias in neural networks for input of non-uniform density, *in* International Conference on Machine Learning, PMLR, 685–694.

Genova, K., Cole, F., Sud, A., Sarna, A., and Funkhouser, T., 2020, Local deep implicit functions for 3d shape, *in* Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 4857–4866.

Keating, S., and Innanen, K. A., 2019, Parameter crosstalk and modeling errors in viscoacoustic seismic full-waveform inversion: Geophysics, **84**, No. 4, R641–R653.

Keating, S., and Innanen, K. A., 2020, Parameter crosstalk and leakage between spatially separated unknowns in viscoelastic full-waveform inversion: Geophysics, **85**, No. 4, R397–R408.

Lailly, P., and Bednar, J., 1983, The seismic inverse problem as a sequence of before stack migrations.

Liu, S., Saito, S., Chen, W., and Li, H., 2019, Learning to infer implicit surfaces without 3d supervision: arXiv preprint arXiv:1911.00767.

Park, J. J., Florence, P., Straub, J., Newcombe, R., and Lovegrove, S., 2019, Deepsdf: Learning continuous signed distance functions for shape representation, *in* Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 165–174.

Plessix, R.-E., 2006, A review of the adjoint-state method for computing the gradient of a functional with geophysical applications: Geophysical Journal International, **167**, No. 2, 495–503.

Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F., Bengio, Y., and Courville, A., 2019, On the spectral bias of neural networks, *in* International Conference on Machine Learning, PMLR, 5301–5310.

Richardson, A., 2018, Seismic full-waveform inversion using deep learning tools and techniques: arXiv preprint arXiv:1801.07232.

Sitzmann, V., Martel, J., Bergman, A., Lindell, D., and Wetzstein, G., 2020, Implicit neural representations with periodic activation functions: Advances in Neural Information Processing Systems, **33**.

Sitzmann, V., Zollhöfer, M., and Wetzstein, G., 2019, Scene representation networks: Continuous 3d-structure-aware neural scene representations: arXiv preprint arXiv:1906.01618.

Stanley, K. O., 2007, Compositional pattern producing networks: A novel abstraction of development: Genetic programming and evolvable machines, **8**, No. 2, 131–162.

Sun, B., and Alkhalifah, T., 2020, Ml-descent: An optimization algorithm for full-waveform inversion using machine learning: Geophysics, **85**, No. 6, R477–R492.

Sun, H., and Demanet, L., 2020, Extrapolated full-waveform inversion with deep learning: Geophysics, **85**, No. 3, R275–R288.

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

Tancik, M., Srinivasan, P. P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J. T., and Ng, R., 2020, Fourier features let networks learn high frequency functions in low dimensional domains: arXiv preprint arXiv:2006.10739.

Tarantola, A., 1984, Inversion of seismic reflection data in the acoustic approximation: Geophysics, **49**, No. 8, 1259–1266.

Teodor, D., Comina, C., Khosro Anjom, F., Brossier, R., Valentina Socco, L., and Virieux, J., 2021, Challenges in shallow target reconstruction by 3d elastic full-waveform inversion—which initial model?: Geophysics, **86**, No. 4, R433–R446.

Tromp, J., Tape, C., and Liu, Q., 2005, Seismic tomography, adjoint methods, time reversal and banana-doughnut kernels: Geophysical Journal International, **160**, No. 1, 195–216.

Virieux, J., and Operto, S., 2009, An overview of full-waveform inversion in exploration geophysics: Geophysics, **74**, No. 6, WCC1–WCC26.

Zhang, T., Innanen, K. A., Sun, J., and Trad, D. O., 2020, Numerical analysis of a deep learning formulation of multi-parameter elastic full waveform inversion, *in* SEG International Exposition and Annual Meeting, OnePetro.

Zhang, T., Sun, J., Innanen, K. A., and Trad, D. O., 2021, A recurrent neural network for l1 anisotropic viscoelastic full-waveform inversion with high-order total variation regularization, *in* First International Meeting for Applied Geoscience & Energy, Society of Exploration Geophysicists, 1374–1378.

Zhang, Z.-d., and Alkhalifah, T., 2020, High-resolution reservoir characterization using deep learning-aided elastic full-waveform inversion: the north sea field data example: Geophysics, **85**, No. 4, WA137–WA146.