Electronic Research Archive

https://www.aimspress.com/journal/era

ERA, 34(10): 7096–7117.
DOI: 10.3934/era.2026307
Received: 25 May 2026
Revised: 30 July 2026
Accepted: 11 August 2026
Published: 17 August 2026

Research article

Data-driven full waveform inversion with vision transformers

Feng Li, Shuo Peng, Zongzeng Li, Hongsun Fu* and Jinlong Yuan*

School of Science, Dalian Maritime University, Dalian 116026, China

* **Correspondence:** Email: fuhongsun@dlmu.edu.cn, yuanjinlong@dlmu.edu.cn.

Abstract: Full waveform inversion (FWI) is a powerful technique for reconstructing subsurface physical parameters. However, conventional FWI often suffers from cycle-skipping and local-minimum issues. With the rapid development of deep learning, data-driven methods have attracted increasing attention in FWI, where the inversion problem is reformulated as an end-to-end reconstruction task and neural networks are employed to directly estimate velocity models from seismic data. Most existing approaches rely on convolutional neural networks (CNNs) to extract local features from input data. However, compared with natural images, seismic data exhibit strong long-range dependencies that CNNs struggle to model effectively, since information associated with a single reflection interface is spatially distributed across the seismic data. Although CNNs perform well in computer vision, their local feature extraction mechanism limits reconstruction performance in FWI. To address this issue, we have proposed a novel FWI method based on the vision transformer (ViT). Unlike CNN-based approaches, the proposed method leverages the self-attention mechanism to model long-range dependencies in the input data. By stacking multiple transformer blocks, global physical information in seismic data is effectively encoded into the inversion process, resulting in improved reconstruction performance. Numerical results demonstrated that the proposed method outperforms CNN-based approaches in terms of reconstruction accuracy.

Keywords: full waveform inversion; data-driven method; deep neural network; long-range dependency; vision transformer

1. Introduction

Full waveform inversion (FWI) aims to estimate subsurface model parameters from the full waveform information contained in observed data [1]. By minimizing the residual between simulated and observed data, FWI can be formulated as an optimization problem constrained by the wave equation. Gradient-based iterative optimization methods, such as the L-BFGS algorithm [2, 3] and the conjugate gradient method [4, 5], are commonly employed to estimate the optimal model parameters.

To alleviate the ill-posedness of FWI, a series of regularization methods [6–8] have been proposed to obtain a reliable solution. In addition, due to the nonconvexity and nonlinearity of the objective function, gradient-based optimization methods often suffer from cycle-skipping and may converge to local minima. Moreover, the high computational cost remains a major challenge for practical FWI applications, especially in complex geological settings and large-scale problems.

With the rapid development of deep learning (DL), data-driven methods have attracted significant attention in the field of scientific computing [9–11]. Different from traditional physics-driven computational methods, data-driven approaches mainly rely on the powerful nonlinear representation and approximation capabilities of neural networks [12] to directly learn complex mapping relationships between variables through large-scale data training, thereby reducing the dependence on explicit physical modeling and manual feature engineering. In computational geophysics [13–15], a major research direction is to directly learn the inverse mapping from seismic data to model parameters by training neural networks on paired seismic waveforms and corresponding velocity models [16]. This type of learning approach does not depend on the numerical solution of the wave equation and treats FWI as an end-to-end image reconstruction problem.

Most DL-based methods rely on deep neural network architectures, particularly the encoder-decoder architecture. InversionNet [16] and Multiscale InversionNet [17] utilize an encoder-decoder architecture based on convolutional neural networks (CNNs), which is prevalent and effective in existing methods. InversionNet uses CNNs as the encoder to extract features and inverse convolution as the decoder to reconstruct the target velocity model from the features of the encoder. Building on InversionNet, Multiscale InversionNet integrates a low-resolution InversionNet for acquiring the low-frequency aspects of velocity models and a high-resolution InversionNet for reconstructing the high-frequency components from these low-frequency components. VelocityGAN [18] utilizes a generative adversarial network (GAN) based encoder-decoder architecture, applying a CNN as the discriminator and InversionNet as the generator. The discriminator in VelocityGAN continuously refines the generator's output, enhancing the quality of the reconstructed velocity models. InversionNet and VelocityGAN are two highly successful DL-based data-driven methods. OpenFWI [19] presents 12 open-access multiscale benchmark datasets for studying seismic FWI, which are synthesized from multiple sources, encompassing diverse geophysical domains and geological subsurface structures. OpenFWI also performed methods such as InversionNet and VelocityGAN on these datasets. These datasets and results have made significant contributions to reproducible research of DL-based FWI. Recently, Zhang et al. [20] proposed a dual-decoder network (DD-Net) with curriculum learning, in which one decoder reconstructs boundary information while the other recovers velocity values of subsurface structures.

Several studies have also explored the integration of deep learning with domain knowledge in geophysics [21, 22]. FWIGAN [23] combines a GAN with physical principles to generate 2D seismic velocity models, using the wave equation to simulate seismic data and the GAN to assess the discrepancy between simulated and observed data. Yang et al. [24] utilized a U-Net with sparsely distributed well-logs, high-frequency RTM images, and low-frequency FWI models as three channels of the network inputs to generate the high-resolution velocity models. UPFWI [25] integrates a CNN for modeling inverse mapping and a differentiable operator for approximating the forward modeling. By using the difference between the input and predicted seismic data as the loss, UPFWI shifts the learning paradigm from supervised to unsupervised. In addition, beyond geophysical applications,

Ren et al. [26] proposed a physics-embedded neural network with deep learning based FWI to achieve reliable imaging of brain tissues. Their framework uses a forward convolutional neural network to replace the expensive wavefield calculation process and an inversion sub-network to map the wavefield to the velocity model.

However, CNNs extract local features by applying convolutional kernels over limited receptive fields and progressively transform the input data into high-level abstract representations [27]. Although CNNs are widely adopted as encoder architectures, the inherent inductive bias of local connectivity in convolutional kernels may limit their ability to effectively capture features from seismic data. Compared with natural images, seismic data exhibit strong global correlations, since information associated with a single reflection interface is spatially distributed across the seismic data. Moreover, seismic waves interact differently with various subsurface materials, further increasing the complexity of feature representation. Therefore, convolution-based encoders face challenges in capturing global features from seismic data [28].

Transformers [29–31] represent another class of advanced neural network models. Compared with CNNs, the self-attention mechanism employed in transformers is capable of effectively capturing long-range dependencies in input data. Wang et al. [32] and Chen et al. [33] applied transformers to visual tasks by combining CNNs with self-attention mechanisms. Dosovitskiy et al. [34] introduced the vision transformer (ViT) for image classification, achieving superior performance compared to conventional CNN-based models. The ViT divides images into patches, thereby alleviating challenges associated with input sequence length and high dimensionality. Subsequently, the ViT has been extended to various visual tasks, such as semantic segmentation [35, 36] and object detection [37, 38]. Furthermore, transformer-based models have also been widely applied in the field of geophysics. Xiang et al. [39] proposed a SeismicTransformer, which improves the accuracy of seismic wavefield simulation by enhancing global feature extraction. Wang et al. [40] introduced a seismic data denoising transformer (SDT), which overcomes the limitations of CNNs in capturing long-range dependencies and improves the reconstruction performance of seismic data. Ning et al. [41] proposed an improved transformer encoder for seismic impedance inversion.

In this paper, motivated by the ViT's achievements in various domains, we propose a data-driven FWI method with vision transformers, named TFWI. The proposed network utilizes an encoder-decoder architecture to reconstruct velocity models. A ViT is used as the encoder to extract global features from seismic data. Our approach avoids using convolutional kernel sliding to obtain patches, aiming to reduce parameter overhead and simplify model training. We contend that randomly initialized convolutional kernels may struggle to extract meaningful information early in training, potentially leading to misinterpretations by subsequent transformer layers. Inverse convolution is used as the decoder to reconstruct the velocity model from the extracted features. A residual connection is introduced into the inverse convolution for feature fusion, designed to improve the quality of the velocity model. We also explore the effect of different loss functions and perform robustness tests to evaluate the model robustness. Numerical experiments have been performed on eight datasets in OpenFWI, and numerical results demonstrate that TFWI provides superior reconstruction performance both qualitatively and quantitatively.

The rest of the paper is organized as follows: In Section 2, we introduce our model, including a brief review of the physics-driven approach and a detailed discussion of our model structure. In Section 3, we outline the experimental settings and compare our method against existing data-driven methods.

Additionally, robustness tests, ablation experiments, and additional experiments are implemented to further assess our method. Section 4 discusses possible extensions of the proposed framework. We summarize our study in Section 5.

2. Methodology

2.1. Physics-driven methods

Physics-driven methods estimate subsurface model parameters from observed seismic data under the constraints of physical governing equations. To simulate seismic wave propagation in a medium, we consider the 2-D acoustic wave equation with a damping term:

$$\begin{cases} \dfrac{1}{\boldsymbol{v}^2(\boldsymbol{x})}\dfrac{\partial^2 \boldsymbol{u}(\boldsymbol{x},t)}{\partial t^2} + \zeta(\boldsymbol{x})\dfrac{\partial \boldsymbol{u}(\boldsymbol{x},t)}{\partial t} - \nabla^2 \boldsymbol{u}(\boldsymbol{x},t) = q_s(t)\delta(\boldsymbol{x}-\boldsymbol{x}_s), \\ \boldsymbol{u}(\boldsymbol{x},0) = 0, \\ \dfrac{\partial \boldsymbol{u}(\boldsymbol{x},0)}{\partial t} = 0, \end{cases} \tag{2.1}$$

where $\boldsymbol{x} = (x, z)$ denotes the spatial position, with x and z representing the horizontal and vertical coordinates, respectively, t denotes time, $\boldsymbol{v}(\boldsymbol{x})$ is the P-wave velocity model, $\boldsymbol{u}(\boldsymbol{x}, t)$ is the wavefield amplitude, $q_s(t)$ is the source function located at $\boldsymbol{x}_s$, $\delta(\cdot)$ denotes the Dirac delta function, and ∇^2 is the spatial Laplacian operator. The damping function $\zeta(\boldsymbol{x})$ is introduced to suppress artificial reflections from the boundaries of the finite computational domain and is typically nonzero only in the absorbing boundary region.

Given the velocity model and source function, the seismic data recorded at the receivers can be written as

$$\boldsymbol{d}(\boldsymbol{x}_r, t; \boldsymbol{v}) = \mathbf{R}\boldsymbol{u}(\boldsymbol{x}, t; \boldsymbol{v}), \tag{2.2}$$

where $\boldsymbol{u}(\boldsymbol{x}, t; \boldsymbol{v})$ is the solution of (2.1) and $\mathbf{R}$ is the observation operator that extracts the wavefield responses at the receiver positions $\boldsymbol{x}_r$. For simplicity, the wave equation and the observation process can be expressed as

$$\begin{cases} \mathbf{A}(\boldsymbol{v})\boldsymbol{u} = q, \\ \mathbf{R}\boldsymbol{u} = \boldsymbol{d}, \end{cases} \tag{2.3}$$

where $\mathbf{A}(\boldsymbol{v}) = (1/\boldsymbol{v}^2)(\partial^2/\partial t^2) + \zeta(\partial/\partial t) - \nabla^2$ denotes the wave-equation operator. After discretizing the computational domain into $n_x \times n_z$ grid points and the time dimension into N_t samples, the velocity model is represented as $\boldsymbol{v} \in \mathbb{R}^{n_x \times n_z}$. The observed seismic data $\boldsymbol{d} \in \mathbb{R}^{N_s \times N_r \times N_t}$ can then be obtained by solving (2.3) using a finite-difference scheme, where N_s and N_r denote the numbers of sources and receivers, respectively. Therefore, the forward modeling process can be written as

$$\mathbf{F}(\boldsymbol{v}) = \boldsymbol{d}, \tag{2.4}$$

where $\mathbf{F} : \mathbb{R}^{n_x \times n_z} \mapsto \mathbb{R}^{N_s \times N_r \times N_t}$ is the forward modeling operator that maps the velocity model to the observed seismic data.

Physics-driven FWI can be formulated as a regularized optimization problem:

$$\hat{\boldsymbol{v}} = \arg\min_{\boldsymbol{v} \in \mathbb{R}^{n_x \times n_z}} \left\{ \mathcal{D}\left(\boldsymbol{d}^{sim}(\boldsymbol{v}), \boldsymbol{d}^{obs}\right) + \lambda \mathcal{R}(\boldsymbol{v}) \right\}, \tag{2.5}$$

where $\mathcal{D}(\cdot)$ denotes the data-fidelity term, $\boldsymbol{d}^{obs}$ is the observed seismic data obtained by forward modeling with the true velocity model, $\boldsymbol{d}^{sim}(\boldsymbol{v})$ is the simulated seismic data generated from the current velocity model $\boldsymbol{v}$, $\mathcal{R}(\boldsymbol{v})$ is the regularization term, and λ controls the regularization strength. A widely used data-fidelity term is the l_2 norm:

$$\mathcal{D}\left(\boldsymbol{d}^{sim}, \boldsymbol{d}^{obs}\right) = \frac{1}{2} \sum_{s=1}^{N_s} \sum_{r=1}^{N_r} \sum_{t=1}^{N_t} \left\| \boldsymbol{d}^{sim}_{s,r,t} - \boldsymbol{d}^{obs}_{s,r,t} \right\|_2^2 . \tag{2.6}$$

However, conventional physics-driven FWI typically relies on gradient-based iterative algorithms to solve (2.5), which are computationally expensive and prone to convergence to local minima. In recent years, data-driven methods based on machine learning, particularly deep learning, have emerged as important alternatives for velocity model reconstruction.

2.2. Data-driven methods

We employ a data-driven approach to solve the FWI problem by using deep neural networks to infer velocity models from seismic data. Specifically, given a training set of paired seismic data and velocity models,

$$\left\{ (\boldsymbol{d}^i, \boldsymbol{v}^i) \mid \boldsymbol{d}^i = \mathbf{F}(\boldsymbol{v}^i) \right\}_{i=1}^{N}, \tag{2.7}$$

a deep neural network can be used to learn a direct mapping from seismic data $\boldsymbol{d}$ to the velocity model $\boldsymbol{v}$, thereby approximating an inverse mapping of the forward modeling operator $\mathbf{F}$ in (2.4). The training process is formulated as

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}\left(\mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{d}^i), \boldsymbol{v}^i \right), \tag{2.8}$$

where $\mathcal{G}_{\boldsymbol{\theta}} : \mathbb{R}^{N_s \times N_r \times N_t} \to \mathbb{R}^{n_x \times n_z}$ denotes the neural network, $\boldsymbol{\theta}$ represents its trainable parameters, and $\mathcal{L}(\cdot)$ is the training loss. Once trained, the network can directly predict the corresponding velocity model from input seismic data, avoiding the repeated forward modeling and iterative optimization required by conventional physics-driven FWI.

Our network adheres to the standard encoder-decoder architecture, as shown in Figure 1. The encoder extracts features from the seismic data employing vision transformers. The decoder then uses inverse convolution followed by convolution to translate those features to the desirable velocity model.

2.2.1. Encoder

The encoder part basically follows the design of vision transformers without the final classifier, which mainly includes a patch, linear embedding, and a transformer block.

Patch. The encoder starts with taking patches of the input image, i.e., cropping the image into 10×10 squares. Particularly, the patching performed here does not employ the common method of cropping with a convolution kernel. Instead, we used Python's built-in functions to avoid contaminating the initial information. Cropping with a convolution kernel would have introduced interference from the kernel parameters.

Embedding. Following patches, linear embedding uses a learnable embedding matrix to project patches into tokens of dimension 100. Additionally, positional embedding is appended to the embedded tokens to keep the spatial information of the patches as in the original image.

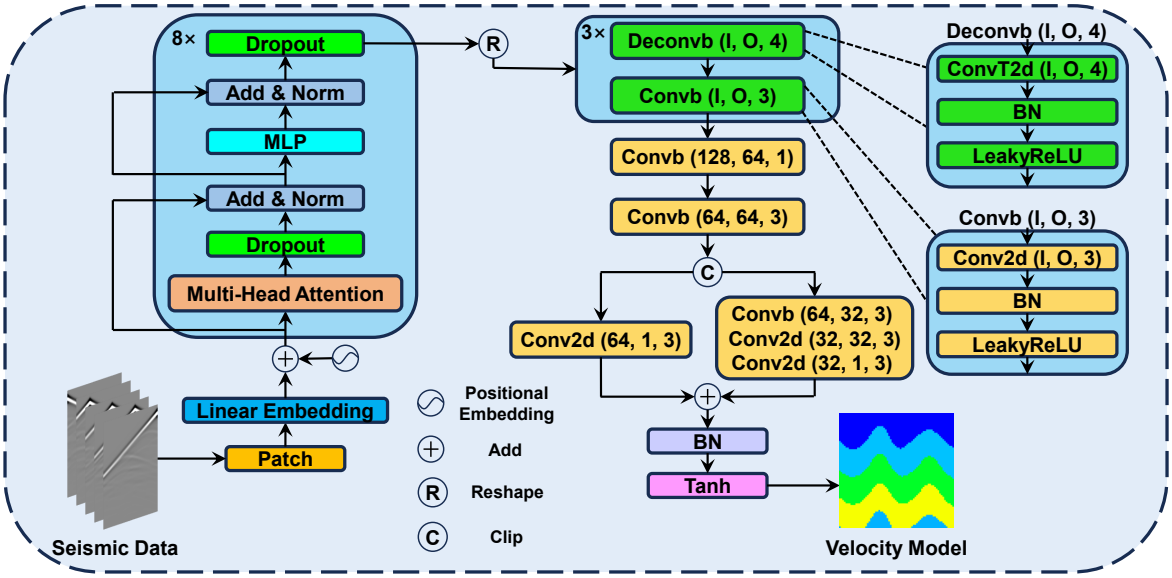


Figure 1. Architecture of FWI with vision transformers. In the attention module of the encoder, the feature dimension is 700×100. At the end of the encoder, it is reshaped into $700 \times 10 \times 10$ and used as the input to the decoder. The decoder begins with three consecutive pairs of deconvolution blocks (Deconvb) and convolution blocks (Convb), where ConvT2d(I, O, K) denotes a 2-D inverse convolutional layer with a stride of 2, and Conv2d(I, O, K) denotes a 2-D convolutional layer with a stride of 1. Here, I, O, and K represent the numbers of input and output channels and the kernel size, respectively. In these three pairs, the input channel numbers of the inverse convolutional layers are 700, 512, and 256, and their output channel numbers are 512, 256, and 128, respectively. The input and output channel numbers of the convolutional layers are 512, 256, and 128, respectively.

Transformer Block. The tokens are fed into the transformer block for feature extraction. Our transformer block adheres to the architecture of the vanilla transformer [29], consisting of a multi-head attention layer, as indicated in Figure 2, two normalization layers, a fully connected layer, and residual connections. To enhance the model generalization, we add a dropout after the second normalization layer and the multi-head attention layer. The attention calculation formula is as follows:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}})\mathbf{V}, \tag{2.9}$$

where $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ are the query matrix, key matrix, and value matrix of the attention layer, respectively. d_k represents the dimension of embedded patches. As shown in (2.10), multi-head attention means that each head performs the attention calculation separately and concatenates the results.

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\text{head}_1, \cdots, \text{head}_h)\mathbf{W}^O, \tag{2.10}$$

where $\mathbf{W}^O$ is the learnable parameter matrix for projecting the concatenated results to the desired dimension.

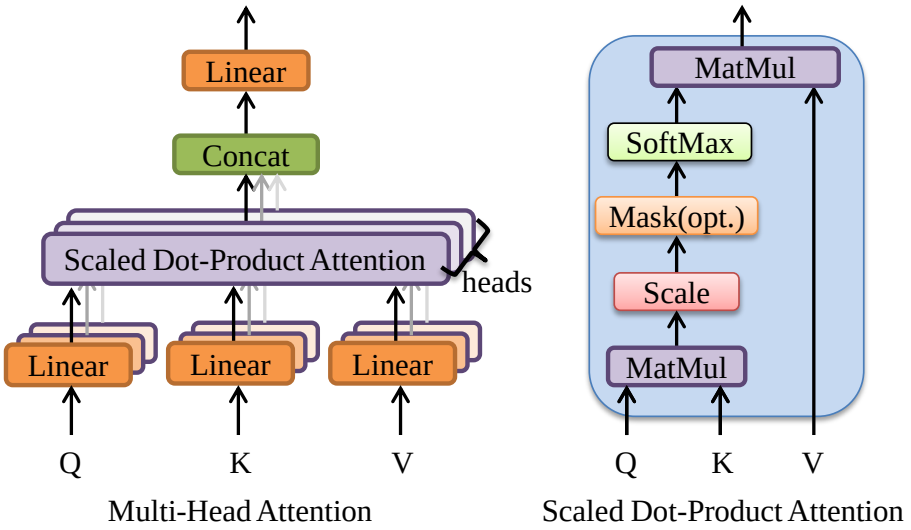


Figure 2. Architecture of multi-head attention (left) and scaled dot-product attention (right).

2.2.2. Decoder

The main component of the decoder consists of deconvolution and convolution blocks. The deconvolution block (Deconvb in Figure 1) includes an inverse convolutional layer with 4×4 kernels and a stride of 2, a batch normalization (BN) layer, and LeakyReLU activation. This block progressively increases the spatial resolution of the feature maps. The convolution block (Convb in Figure 1) employs a convolutional layer with a stride of 1, BN, and LeakyReLU activation to summarize and fuse the features from the deconvolution block. At the end of the decoder, we add a residual connection to split the feature map into two paths for parallel processing with different ways to preserve critical information.

2.2.3. Loss

The difference between the generated and the true velocity model is used as the reconstruction error to learn the model parameters. For clarity, assume $\mathbf{I} \in \mathbb{R}^{1\times5\times1000\times70}$ is an input seismic data with five channels, a height of 1000, and a width of 70. $\mathbf{I}$ is divided into 700 patches, each projected as an embedded token of dimension 100. A feature map $\mathbf{f} \in \mathbb{R}^{1\times700\times100}$ is obtained by processing these tokens through eight transformer blocks, each with four heads. The reshape function converts $\mathbf{f}$ into $\mathbf{f}_1 \in \mathbb{R}^{1\times700\times10\times10}$. $\mathbf{f}_1$ is then transformed into $\mathbf{f}_2 \in \mathbb{R}^{1\times128\times80\times80}$ through three sets of deconvolution and convolution blocks. After further operations and dual-path processing, a predicted velocity model $\hat{\mathbf{v}} \in \mathbb{R}^{1\times1\times70\times70}$ is ultimately generated.

Specifically, the loss function $\mathcal{L}(\cdot)$ uses a linear combination of l_1 loss and l_2 loss:

$$\begin{aligned} \text{Loss} &= \frac{1}{b}\sum_{i=1}^{b}\left(l_1(\mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{d}^i), \boldsymbol{v}^i) + l_2(\mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{d}^i), \boldsymbol{v}^i)\right) \\ &= \frac{1}{b}\sum_{i=1}^{b}\left(\left|\mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{d}^i) - \boldsymbol{v}^i\right|_1 + \left\|\mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{d}^i) - \boldsymbol{v}^i\right\|_2^2\right), \end{aligned} \tag{2.11}$$

where b denotes the number of samples used for training in each iteration, i.e., the batch size.

3. Experiments

We evaluate the performance of TFWI on the datasets from the OpenFWI project. We also use noise to check the robustness of models and present the ablation experiments and additional experiments to further analyze TFWI.

3.1. Datasets and experimental settings

3.1.1. Dataset description

We adopt eight datasets from the OpenFWI project to test the proposed method, including FlatVel-A/B, CurveVel-A/B, FlatFault-A/B, and CurveFault-A/B. All seismic data are obtained by forward modeling the generated velocity models using a finite-difference scheme with second-order accuracy in time and fourth-order accuracy in space, together with absorbing boundary conditions [42] to suppress boundary reflections. The velocity models are discretized into 70×70 grid points with a spatial interval of 10 m, corresponding to a 0.7 km$\times$0.7 km subsurface computational domain. Five sources are placed with a horizontal spacing of 140 m, and each source is excited by a Ricker wavelet with a dominant frequency of 15 Hz. The seismic data are recorded by 70 receivers with a horizontal spacing of 10 m. The recording duration is 1 s, and the temporal sampling interval is 0.001 s. The sources and receivers are both placed at the top surface of the velocity model. Thus, the seismic data for each sample have dimensions of $5 \times 1000 \times 70$.

The details of the datasets are summarized in Table 1. Figure 3 presents an example of seismic data containing five shot gathers along with the corresponding velocity model. Figure 4 shows several velocity models from the datasets, in which the numerical values correspond to the seismic velocities.

Table 1. Dataset summary.

Dataset	Train/Test	Size	Seismic data size	Velocity model size
FlatVel-A/B	24 K/6 K	43 GB	$5 \times 1000 \times 70$	70×70
CurveVel-A/B	24 K/6 K	43 GB	$5 \times 1000 \times 70$	70×70
FlatFault-A/B	48 K/6 K	77 GB	$5 \times 1000 \times 70$	70×70
CurveFault-A/B	48 K/6 K	77 GB	$5 \times 1000 \times 70$	70×70

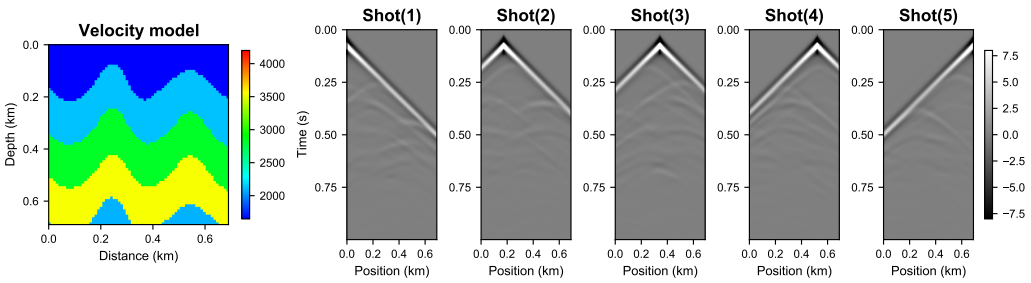


Figure 3. Visualization of seismic data and the corresponding velocity model.

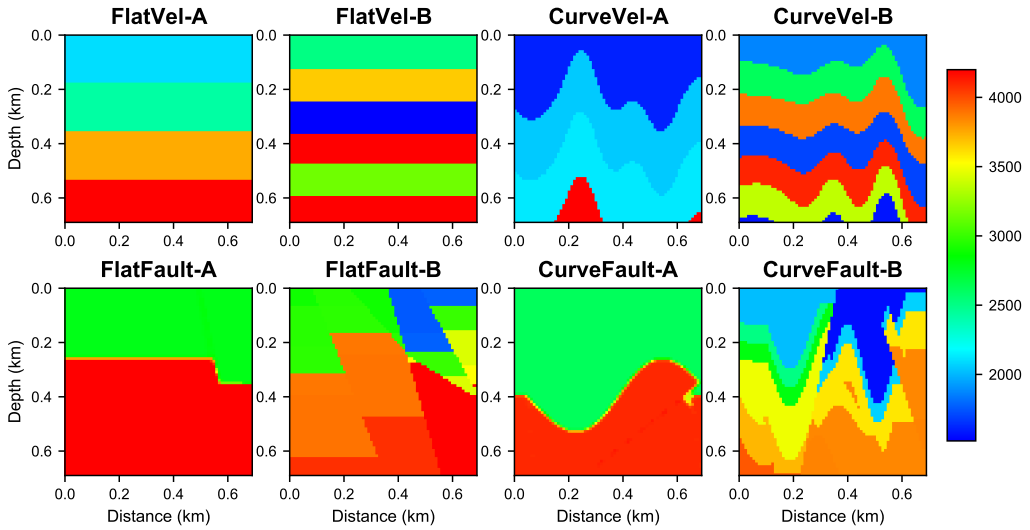


Figure 4. Visualization of example velocity models on the datasets.

3.1.2. Implementation details

To evaluate the performance of the proposed method on the above datasets, we compare it with two popular data-driven methods mentioned in [19], namely, InversionNet and VelocityGAN, both of which employ stacked convolutional layers as encoders. In addition, we quantitatively compare our method with a more advanced method, DD-Net70.

The details of the training and test sets are provided in Table 1. For the proposed method, 10% of the training samples are randomly selected as the validation set. The AdamW optimizer is used to solve (2.8) on the remaining training set and update the model parameters according to the corresponding loss function. During training, the model is evaluated on the validation set after each epoch, and the model with the lowest validation loss is selected as the best model for final testing. The optimizer parameters are set to $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. Following the data-processing approach in [19], we normalize the velocity models and seismic data to the range $[-1, 1]$ using min-max normalization, which are used as the training labels and inputs, respectively. The initial learning rate is set to 0.002 and is multiplied by 0.1 at the 110th and 180th epochs. The weight decay coefficient is set to 0.0001. The model is trained for a maximum of 200 epochs with a batch size of 128. In particular, we keep the same parameter settings on all the datasets. We employ the learning rate warm-up technique, performing 465 warm-up batches for the Vel Family data and 935 for the Fault Family data. Our models are trained in a PyTorch environment with an NVIDIA GTX 4090 GPU and an Intel i9 13,900 CPU.

For the comparison methods, we report the results from their public implementations. The training configurations reported for these methods are summarized as follows. Specifically, InversionNet, an encoder-decoder CNN, was trained with a learning rate of 0.0001, no learning rate decay, a weight decay coefficient of 0.0001, a maximum of 120 epochs, and a batch size of 256. VelocityGAN, a generative adversarial network whose generator shares the same architecture as InversionNet and whose discriminator is a nine-layer CNN, was trained with a learning rate of 0.0001, a weight decay coefficient of 0.0001, a maximum of 480 epochs, and a batch size of 64. DD-Net70, a CNN with one

encoder and two decoders, was trained with a learning rate of 0.0001, no weight decay, a maximum of 120 epochs, and a batch size of 64.

After training, we evaluate the model performance on the test sets. Specifically, for each seismic datum $\boldsymbol{d}$ in the test set, the trained network first predicts the velocity model as

$$\hat{\boldsymbol{v}} = \boldsymbol{G}_{\hat{\boldsymbol{\theta}}}(\boldsymbol{d}), \tag{3.1}$$

where $\hat{\boldsymbol{\theta}}$ denotes the learned network parameters. The error between the predicted velocity model $\hat{\boldsymbol{v}}$ and the true velocity model $\boldsymbol{v}$ is then computed and averaged over each test set. The evaluation metrics used in this work are defined as follows.

(1) Mean absolute error (MAE):

$$\text{MAE} = \frac{1}{n_x n_z} \sum_{k=1}^{n_x} \sum_{l=1}^{n_z} \left| \hat{\boldsymbol{v}}_{k,l} - \boldsymbol{v}_{k,l} \right|. \tag{3.2}$$

(2) Root mean squared error (RMSE):

$$\text{RMSE} = \sqrt{\frac{1}{n_x n_z} \sum_{k=1}^{n_x} \sum_{l=1}^{n_z} \left(\hat{\boldsymbol{v}}_{k,l} - \boldsymbol{v}_{k,l} \right)^2}. \tag{3.3}$$

(3) Structural similarity (SSIM):

$$\text{SSIM} = \frac{(2\mu_{\hat{\boldsymbol{v}}}\mu_{\boldsymbol{v}} + C_1)(2\sigma_{\hat{\boldsymbol{v}}\boldsymbol{v}} + C_2)}{(\mu_{\hat{\boldsymbol{v}}}^2 + \mu_{\boldsymbol{v}}^2 + C_1)(\sigma_{\hat{\boldsymbol{v}}}^2 + \sigma_{\boldsymbol{v}}^2 + C_2)}, \tag{3.4}$$

where $C_1 = (K_1 L)^2$ and $C_2 = (K_2 L)^2$, L refers to the dynamic range of the pixel values, and $K_1 \ll 1$ and $K_2 \ll 1$ are scalar constants. $\mu_{\hat{\boldsymbol{v}}}$ and $\mu_{\boldsymbol{v}}$ are the means of $\hat{\boldsymbol{v}}$ and $\boldsymbol{v}$, respectively, while $\sigma_{\hat{\boldsymbol{v}}}^2$ and $\sigma_{\boldsymbol{v}}^2$ denote their variances. $\sigma_{\hat{\boldsymbol{v}}\boldsymbol{v}}$ is the covariance between $\hat{\boldsymbol{v}}$ and $\boldsymbol{v}$.

In addition, to compare with DD-Net70, we introduce two additional evaluation metrics.

(4) Peak signal-to-noise ratio (PSNR):

$$\text{PSNR} = 10 \log_{10} \left(\frac{\text{MAX}^2}{\text{MSE}} \right), \tag{3.5}$$

where MAX denotes the maximum possible pixel value of the image, and MSE represents the mean squared error.

(5) Learned perceptual image patch similarity (LPIPS): LPIPS evaluates the perceptual difference between two images using a pretrained neural network. Specifically, both images are fed into the same network and the distance is computed as

$$\text{LPIPS} = \sum_p \frac{1}{H_p W_p} \sum_{h=1}^{H_p} \sum_{w=1}^{W_p} \left\| \boldsymbol{w}_p \circ \left(\hat{\boldsymbol{y}}_{h,w}^p - \boldsymbol{y}_{h,w}^p \right) \right\|_2^2, \tag{3.6}$$

where p denotes the p-th layer of the network, and H_p and W_p are the height and width of the corresponding feature map. $\hat{\boldsymbol{y}}^p \in \mathbb{R}^{H_p \times W_p \times C_p}$ and $\boldsymbol{y}^p \in \mathbb{R}^{H_p \times W_p \times C_p}$ represent the features of the predicted and true images at the p-th layer, respectively. $\boldsymbol{w}_p \in \mathbb{R}^{C_p}$ represents a vector for adjusting the weights of different layers.

3.2. Comparison of different methods

We compare our model with InversionNet and VelocityGAN. Instead of retraining these models, we use the results provided by [19], which include InversionNet with l_1 loss (InversionNet-l_1), InversionNet with l_2 loss (InversionNet-l_2), VelocityGAN with l_1 loss (VelocityGAN-l_1), and VelocityGAN with l_2 loss (VelocityGAN-l_2). Detailed results are shown in Table 2.

Table 2. Comparison of quantitative results of three methods.

Dataset	Loss	InversionNet			VelocityGAN			TFWI (ours)		
		MAE	RMSE	SSIM	MAE	RMSE	SSIM	MAE	RMSE	SSIM
FlatVel-A	l_1	0.0131	0.0211	0.9895	0.0118	0.0178	0.9916	**0.0068**	**0.0124**	**0.9955**
	l_2	0.0111	0.0180	0.9887	0.0605	0.0783	0.9453			
FlatVel-B	l_1	0.0351	0.0876	0.9461	0.0329	0.0807	0.9521	**0.0213**	**0.0609**	**0.9712**
	l_2	0.0417	0.0909	0.9402	0.0328	0.0787	0.9556			
CurveVel-A	l_1	0.0685	0.1273	0.8074	0.0482	0.1034	0.8624	**0.0447**	0.0987	**0.8888**
	l_2	0.0690	0.1202	0.8223	0.0510	**0.0976**	0.8758			
CurveVel-B	l_1	0.1497	0.2891	0.6727	**0.1268**	0.2618	0.7111	0.1287	0.2642	**0.7328**
	l_2	0.1624	0.2801	0.6661	0.1428	**0.2611**	0.6962			
FlatFault-A	l_1	0.0172	0.0426	0.9766	0.0868	0.1485	0.9313	**0.0098**	**0.0269**	**0.9883**
	l_2	0.0174	0.0362	0.9798	0.0319	0.0531	0.9798			
FlatFault-B	l_1	0.1055	0.1741	0.7208	0.0925	0.1600	0.7476	**0.0890**	**0.1548**	**0.7841**
	l_2	0.1106	0.1723	0.7186	0.0946	0.1553	0.7552			
CurveFault-A	l_1	0.0206	0.0650	0.9566	0.0258	0.0606	0.9613	**0.0202**	0.0516	**0.9700**
	l_2	0.0280	0.0620	0.9592	0.0216	**0.0505**	0.9687			
CurveFault-B	l_1	0.1646	0.2477	0.6163	**0.1571**	0.2427	0.5996	0.1582	0.2386	**0.6361**
	l_2	0.1669	0.2412	0.6053	0.1583	**0.2336**	0.6033			

TFWI achieves superior SSIM scores across all eight datasets compared to InversionNet and VelocityGAN, which can be attributed to its effective extraction of global features from seismic data. It also demonstrates significant improvements in MAE and RMSE on the FlatVel-A, FlatVel-B, FlatFault-A, and FlatFault-B datasets. On the CurveVel-A and CurveFault-A datasets, TFWI achieves the best MAE results while securing the second-best results on CurveVel-B and CurveFault-B. For RMSE, our model ranks second on the CurveVel-A, CurveFault-A, and CurveFault-B datasets and third on CurveVel-B. Notably, our non-first results are within 2.2% of the best-performing outcomes. Therefore, our method outperforms InversionNet and VelocityGAN in quantitative results.

For velocity model visualization comparison, we use the pre-trained weights from [19] for InversionNet and VelocityGAN. We randomly select seismic data from the FlatFault-A and CurveVel-A datasets, and process them with all three models to obtain the corresponding velocity models. Visualization results in Figure 5 show that our method produces velocity models with colors and shapes closely matching the true velocity models. Notably, our method captures finer details, as indicated by the rectangles, although some areas are still not perfectly reproduced compared to the true velocity models.

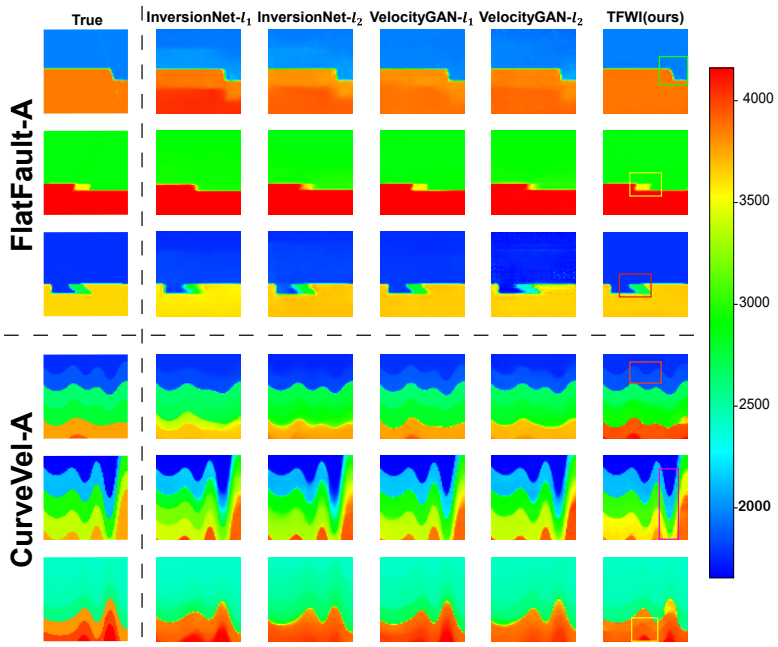


Figure 5. Comparison of visualization results of three methods. The left column shows randomly selected true velocity models from the FlatFault-A and CurveVel-A datasets. Each row on the right displays the predicted velocity models from InversionNet-l_1, InversionNet-l_2, VelocityGAN-l_1, VelocityGAN-l_2, and TFWI (ours), respectively.

To further highlight the importance of global feature extraction in velocity model reconstruction, we provide additional quantitative comparisons of the TFWI method with a more advanced approach, DD-Net70. Experiments were conducted on the FlatVel-A, FlatFault-A, CurveVel-A, and CurveFault-A datasets. Moreover, for a fair comparison, we modified our data processing method to be consistent with DD-Net70 and adopted the same window size for SSIM calculation. As shown in Table 3, TFWI outperforms DD-Net70 on all evaluation metrics. In particular, for the SSIM metric, TFWI achieves significant improvements on the CurveVel-A and CurveFault-A datasets. Notably, DD-Net70 employs a dual-branch network architecture and incorporates boundary-related loss functions, further demonstrating the critical role of global features in velocity model reconstruction.

Table 3. Quantitative results comparison between DD-Net70 and TFWI.

Dataset	Method	PSNR	SSIM	LPIPS ($\times 10^{-3}$)	MSE ($\times 10^{-3}$)	MAE ($\times 10^{-3}$)
FlatVel-A	DD-Net70	38.269	0.971	2.736	0.262	6.954
	TFWI (ours)	**45.930**	**0.991**	**0.423**	**0.038**	**3.400**
FlatFault-A	DD-Net70	33.026	0.951	17.628	0.606	11.566
	TFWI (ours)	**38.408**	**0.975**	**7.316**	**0.180**	**4.921**
CurveVel-A	DD-Net70	23.519	0.763	85.501	5.327	41.103
	TFWI (ours)	**25.795**	**0.836**	**30.709**	**2.435**	**22.327**
CurveFault-A	DD-Net70	28.086	0.896	27.089	1.887	19.629
	TFWI (ours)	**32.439**	**0.947**	**11.310**	**0.667**	**10.101**

3.3. Robustness tests

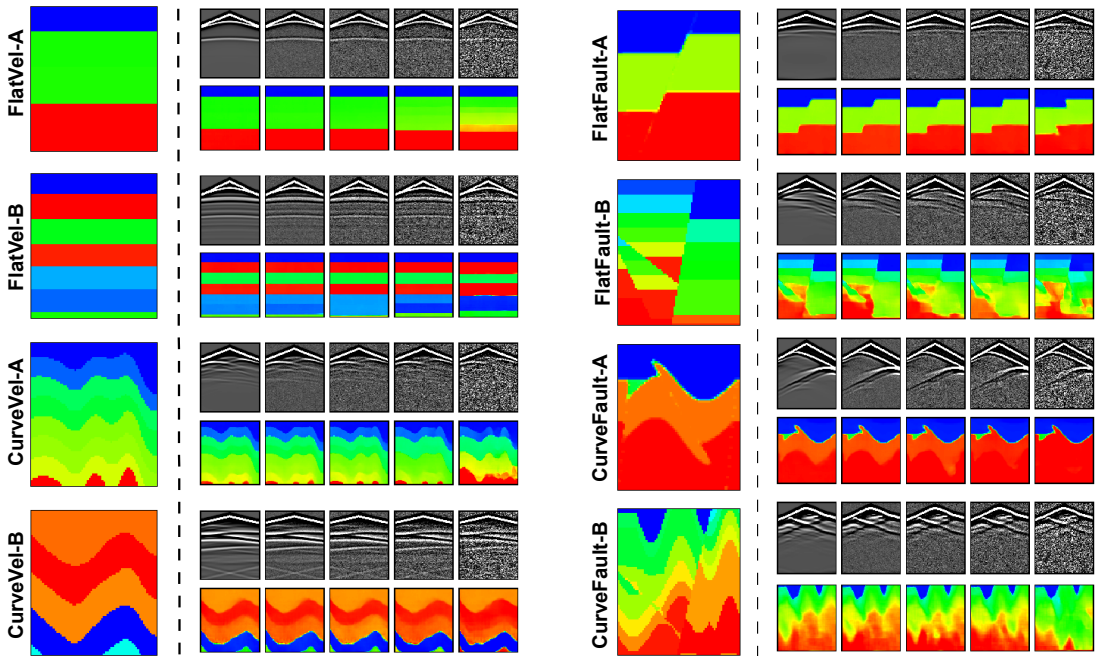
To further evaluate TFWI, we perform robustness tests using seismic data with noise to test TFWI. Following the noise addition method in [19], we generated data $\mathbf{X}$ following a standard normal distribution. Gaussian noise with different standard deviations, $\sqrt{\sigma}\mathbf{X}$, was added to the test data. We then compared changes in MAE, RMSE, and SSIM before and after noise addition and visualized the produced velocity models. Tables 4 and 5 and Figure 6 present the detailed results.

Table 4. Quantitative results of TFWI under different noise levels.

Dataset	$\sigma = 0$			$\sigma = 1 \times 10^{-4}$			$\sigma = 5 \times 10^{-4}$			$\sigma = 1 \times 10^{-3}$			$\sigma = 5 \times 10^{-3}$		
	MAE	RMSE	SSIM	MAE	RMSE	SSIM	MAE	RMSE	SSIM	MAE	RMSE	SSIM	MAE	RMSE	SSIM
FlatVel-A	0.0068	0.0124	0.9955	0.0151	0.0300	0.9834	0.0393	0.0702	0.9536	0.0628	0.1049	0.9272	0.1735	0.2759	0.8100
Degradation (%)	\	\	\	122.06	141.94	1.22	477.94	466.13	4.21	823.53	745.97	6.86	2451.47	2125.00	18.63
FlatVel-B	0.0213	0.0609	0.9712	0.0240	0.0662	0.9658	0.0331	0.0856	0.9478	0.0428	0.1053	0.9291	0.0970	0.1996	0.8324
Degradation (%)	\	\	\	12.68	8.70	0.56	55.40	40.56	2.41	100.94	72.91	4.33	355.40	227.75	14.29
CurveVel-A	0.0447	0.0987	0.8888	0.0453	0.0996	0.8875	0.0509	0.1071	0.8788	0.0613	0.1208	0.8634	0.1416	0.2359	0.7506
Degradation (%)	\	\	\	1.34	0.91	0.15	13.87	8.51	1.13	37.14	22.39	2.86	216.78	139.01	15.55
CurveVel-B	0.1287	0.2642	0.7328	0.1304	0.2666	0.7304	0.1421	0.2829	0.7172	0.1556	0.3031	0.7019	0.2065	0.3763	0.6378
Degradation (%)	\	\	\	1.32	0.91	0.33	10.41	7.08	2.18	20.90	14.72	4.22	60.45	42.43	12.96
FaltFault-A	0.0098	0.0269	0.9883	0.0145	0.0343	0.9872	0.0337	0.0676	0.9784	0.0469	0.0881	0.9682	0.0935	0.1515	0.9125
Degradation (%)	\	\	\	47.96	27.51	0.11	243.88	151.30	1.00	378.57	227.51	2.03	854.08	463.20	7.67
FaltFault-B	0.0890	0.1548	0.7841	0.1065	0.1762	0.7662	0.1187	0.1902	0.7440	0.1289	0.2023	0.7250	0.1696	0.2530	0.6536
Degradation (%)	\	\	\	19.66	13.82	2.28	33.37	22.87	5.11	44.83	30.68	7.54	90.56	63.44	16.64
CurveFault-A	0.0202	0.0516	0.9700	0.0224	0.0537	0.9693	0.0355	0.0700	0.9631	0.0509	0.0927	0.9519	0.1142	0.1943	0.8876
Degradation (%)	\	\	\	10.89	4.07	0.07	75.74	38.66	0.71	151.98	85.59	1.87	465.35	276.55	8.49
CurveFault-B	0.1582	0.2386	0.6361	0.1640	0.2459	0.6316	0.1722	0.2559	0.6215	0.1784	0.2633	0.6123	0.2068	0.2987	0.5707
Degradation (%)	\	\	\	3.67	3.06	0.71	8.85	1.73	2.30	12.77	10.60	3.74	30.72	25.19	10.28

Table 5. Averaged degradation percentage of quantitative results of InversionNet, VelocityGAN, and TFWI under different noise levels.

Method	$\sigma = 1 \times 10^{-4}$			$\sigma = 5 \times 10^{-4}$		
	MAE	RMSE	SSIM	MAE	RMSE	SSIM
InversionNet-l_1	114.70%	79.48%	9.38%	241.70%	168.26%	21.99%
InversionNet-l_2	85.54%	69.14%	6.59%	216.63%	173.35%	18.37%
VelocityGAN-l_1	33.35%	22.89%	2.33%	130.60%	96.35%	10.67%
VelocityGAN-l_2	18.60%	14.18%	2.01%	151.38%	125.17%	11.17%
TFWI	**9.09%**	**7.09%**	**0.65%**	**30.67%**	**24.38%**	**2.33%**



(a) Results of the Vel Family **(b)** Results of the Fault Family

Figure 6. Visualization of the robustness tests. The left side of the dividing line shows the true velocity model while the right side depicts variations in a specific channel of seismic data at different σ values, along with the corresponding velocity models of TFWI.

Table 4 demonstrates that the model performance degrades as σ increases. However, when σ is small, the performance degradation of TFWI remains manageable. Specifically, for $\sigma = 1 \times 10^{-4}$, 5×10^{-4}, and 1×10^{-3}, the decrease in SSIM is generally less than 5%. Similarly, except for the FlatVel-A and FlatFault-A datasets, reductions in MAE and RMSE are minor. The higher degradation percentages in MAE and RMSE for FlatVel-A and FlatFault-A are due to TFWI's exceptionally low MAE and RMSE at $\sigma = 0$. Despite the significant degradation percentages, TFWI still achieves commendable MAE and RMSE on these datasets.

Figure 6 illustrates that as σ increases, the noisy region in the seismic data expands, significantly impacting the encoder's feature extraction. Generally, higher σ values lead to smaller trajectory changes but more pronounced color changes in the generated velocity models. When σ is less than 5×10^{-3}, the quality degradation of the velocity models generated by TFWI is minimal. These findings demonstrate TFWI's strong robustness. However, at $\sigma = 5 \times 10^{-3}$, substantial decreases in MAE, RMSE, and SSIM are observed across seven datasets, except for CurveFault-B. This is due to higher noise levels severely blurring seismic data and obscuring valuable information, as shown in the last column of Figure 6(a),(b).

Table 5 presents the average performance degradation of InversionNet, VelocityGAN, and TFWI at $\sigma = 1 \times 10^{-4}$ and 5×10^{-4}. The degradation percentages in MAE, RMSE, and SSIM for TFWI are significantly smaller than those for InversionNet and VelocityGAN, demonstrating TFWI's superior robustness among the three methods.

Remark: In Table 5, the averaged degradation percentage for TFWI is not merely the average of

the percentages in Table 4. Following [19], we first average the MAE, RMSE, and SSIM results in Table 4 and then calculate the degradation percentage. The percentage for TFWI is averaged across the eight datasets used in this paper, whereas the percentages for InversionNet and VelocityGAN, as provided by [19], are averaged across ten datasets, including the eight used in this study.

3.4. Ablation experiments

We perform experiments on the Vel Family datasets to evaluate TFWI with the loss functions l_1 or l_2 alone. The numerical results in Table 6 demonstrate that TFWI with l_1 or l_2 alone can still outperform InversionNet and VelocityGAN across the four datasets, which further demonstrates the superiority of TFWI. Among the three loss functions for TFWI, l_1 exhibited the poorest performance, while $l_1 + l_2$ slightly outperformed l_2. Thus, $l_1 + l_2$ was selected as the loss function.

Table 6. Benefits of using a linear combination of l_1 and l_2.

Dataset	l_1			l_2			$l_1 + l_2$		
	MAE	RMSE	SSIM	MAE	RMSE	SSIM	MAE	RMSE	SSIM
FlatVel-A	**0.0066**	0.0121	0.9954	0.0072	**0.0116**	**0.9958**	0.0068	0.0124	0.9955
FlatVel-B	0.0214	0.0666	0.9687	0.0276	0.0670	0.9615	**0.0213**	**0.0609**	**0.9712**
CurveVel-A	0.0461	0.1016	0.8845	0.0486	**0.0954**	**0.8945**	**0.0447**	0.0987	0.8888
CurveVel-B	**0.1284**	0.2738	0.7254	0.1386	0.2652	0.7215	0.1287	**0.2642**	**0.7328**

Additionally, a residual connection was introduced in the decoder section for feature fusion. We tested the impact of the residual connection on the Vel Family datasets, as shown in Table 7. After adding the residual connection, we observed a significant improvement in SSIM for the CurveVel-B dataset and a slight improvement for the FlatVel-A dataset, but a slight decrease for the FlatVel-B and CurveVel-A datasets. This demonstrates that our decoder with the residual connection can further improve performance, especially for more challenging datasets.

Table 7. The role of residual connection.

Dataset	Without residual connection			With residual connection		
	MAE	RMSE	SSIM	MAE	RMSE	SSIM
FlatVel-A	0.0084	0.0148	0.9921	**0.0068**	**0.0124**	**0.9955**
FlatVel-B	0.0236	0.0624	**0.9714**	**0.0213**	**0.0609**	0.9712
CurveVel-A	0.0454	**0.0969**	**0.8912**	**0.0447**	0.0987	0.8888
CurveVel-B	0.1414	0.2751	0.7061	**0.1287**	**0.2642**	**0.7328**

3.5. Additional experiments

In our method, we do not employ the convolutional sliding technique commonly used in vision transformers for taking patches and instead directly segment the image. This is because using convolutional sliding for patch extraction initially increases the model's sensitivity to changes in the loss function, resulting in significant performance fluctuations. We tested this approach on the Vel Family dataset as shown in Table 8. We found that although patch extraction with convolutional sliding achieves slightly better results on the FlatVel-A and CurveVel-A datasets, it performs poorly

on the FlatVel-B and CurveVel-B datasets, especially for SSIM. Moreover, it is challenging to find a uniform learning rate that performs well across all datasets during training, which limits the applicability of the proposed method.

Table 8. The performance of different patch approaches.

Dataset	Convolutional sliding			Our approach		
	MAE	RMSE	SSIM	MAE	RMSE	SSIM
FlatVel-A	**0.0053**	**0.0106**	0.9951	0.0068	0.0124	**0.9955**
FlatVel-B	0.0256	0.0809	0.9603	**0.0213**	**0.0609**	**0.9712**
CurveVel-A	**0.0397**	**0.0909**	**0.9010**	0.0447	0.0987	0.8888
CurveVel-B	0.1417	0.2763	0.7023	**0.1287**	**0.2642**	**0.7328**

4. Discussion

In this work, we proposed a data-driven TFWI framework and evaluated it on several OpenFWI datasets. This section discusses possible extensions of the proposed framework to larger velocity models, 3-D FWI problems, and hybrid formulations with physics-based constraints.

4.1. Scalability to larger velocity models

In the present experiments, the predicted velocity models have a size of 70×70. The proposed framework could be extended to larger velocity models by modifying the network architecture. To illustrate this possibility, we use the Kimberlina-CO_2 dataset from OpenFWI as an example. As summarized in Table 9, the velocity model has a size of 141×401 and the seismic data have a size of $9 \times 1251 \times 101$, corresponding to 9 sources, 1251 time samples, and 101 receivers.

One possible adaptation of the TFWI architecture in Figure 1 is as follows. Before the patch embedding layer, one temporal sample and one receiver can first be removed from the input seismic data to make the input dimensions easier to transform, resulting in an input size of $9 \times 1250 \times 100$. The patch embedding layer then divides the data into 125×10 patches along the temporal and receiver dimensions, where each patch has a size of $9 \times 10 \times 10$. Each patch is flattened and projected into a 396-dimensional token with positional embedding. Thus, the transformer encoder processes a sequence of size 1250×396. The encoder output can then be reshaped into $1250 \times 18 \times 22$ and passed to the decoder. For the decoder, the number of input channels of the first inverse convolutional layer can be changed to 1250. The kernel sizes of the three inverse convolutional layers can be set to 4×4, 4×4, and 4×7, with strides of 2×2, 2×2, and 2×5, respectively. By keeping the other components the same as those of the original TFWI architecture and applying appropriate cropping, the final network output can be adjusted to the size of 141×401.

4.2. Scalability to 3-D FWI problems

The proposed framework could also be extended to 3-D FWI problems with appropriate modifications of the network architecture. As an example, we consider the 3-D Kimberlina dataset from OpenFWI. As summarized in Table 9, the velocity model has a size of $350 \times 400 \times 400$ and the seismic data have a size of $25 \times 5001 \times 40 \times 40$, corresponding to 25 sources, 5001 time samples,

and 1600 receivers.

In 3-D FWI, the self-attention mechanism can still be used to capture global dependencies in seismic data. However, the 2-D convolutional operations in TFWI should be replaced by their 3-D counterparts to effectively model volumetric features. One possible adaptation of the TFWI architecture in Figure 1 is as follows. Since self-attention is computationally expensive for large-scale seismic data, temporal downsampling can be introduced before the patch embedding layer to reduce the input sequence length. For example, the temporal dimension can be reduced from 5001 to 1250 by removing one temporal sample and then applying two consecutive temporal subsampling operations with a stride of 2, although this may discard part of the temporal information. The patch embedding layer then divides the data into $50 \times 4 \times 4$ patches along the temporal and receiver dimensions, where each patch has a size of $25 \times 25 \times 10 \times 10$. Each patch is flattened and projected into a 512-dimensional token with positional embedding. Thus, the transformer encoder processes a sequence of size 800×512. The encoded features can then be projected by a learnable matrix to a size of 800×7200 and reshaped into $800 \times 18 \times 20 \times 20$. For the decoder, the 2-D operations in Figure 1, including convolution, normalization, activation, and inverse convolution, can be replaced by their 3-D counterparts. The number of input channels of the first inverse convolutional layer is adjusted to 800. The kernel sizes of the three inverse convolutional layers can be set to $4 \times 4 \times 4$, $4 \times 4 \times 4$, and $7 \times 7 \times 7$, with strides of $2 \times 2 \times 2$, $2 \times 2 \times 2$, and $5 \times 5 \times 5$, respectively. By keeping the other components consistent with the corresponding 3-D architecture and applying appropriate cropping, the final network output can be adjusted to the target velocity model size of $350 \times 400 \times 400$.

Table 9. Dataset summary and parameter counts of the corresponding TFWI architectures.

Dataset	Seismic data size	Velocity model size	Parameters
Vel family	$5 \times 1000 \times 70$	70×70	11.908 M
Fault family	$5 \times 1000 \times 70$	70×70	11.908 M
Kimberlina-CO_2	$9 \times 1251 \times 101$	141×401	21.828 M
3-D Kimberlina	$25 \times 5001 \times 40 \times 40$	$350 \times 400 \times 400$	99.454 M

As the data and model architectures become more complex, the above extensions to larger velocity models and 3-D FWI would require substantially higher memory usage and computational cost than the original 2-D TFWI case. Although two linear projection operations are introduced in the possible 3-D adaptation to keep the input dimension of the self-attention module relatively small, the resulting network still contains a much larger number of parameters, as shown in Table 9. Therefore, more efficient computational modules, such as cross-covariance attention [43] and group convolution [44], may be necessary for substantially larger 2-D models, e.g., 512×512 or 1024×1024, and for practical 3-D applications.

4.3. Scalability to hybrid formulations

Currently, TFWI is a purely data-driven method that learns directly from paired data without explicitly exploiting physical information. To further improve the physical consistency of the inversion results, physical constraints can be incorporated into model training. For example, a loss term based on the wave equation can be introduced into the training objective. Let the velocity model predicted by the network be $\hat{\boldsymbol{v}}^i = \mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{d}^i)$. By applying the forward modeling operator $\mathbf{F}$ to $\hat{\boldsymbol{v}}^i$, the

corresponding synthetic seismic data can be generated. The training objective can then be formulated as

$$\mathcal{L}\left(\mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{d}^i), \boldsymbol{v}^i\right) = \lambda_1 \mathcal{L}_{data}\left(\mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{d}^i), \boldsymbol{v}^i\right) + \lambda_2 \mathcal{L}_{phy}\left(\mathbf{F}\left(\mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{d}^i)\right), \boldsymbol{d}^i\right), \tag{4.1}$$

where $\mathcal{L}_{data}$ denotes the data loss between the predicted velocity model and the true velocity model, $\mathcal{L}_{phy}$ denotes the physical consistency loss between the synthetic seismic data generated from the predicted velocity model and the observed seismic data, $\mathbf{F}$ is the forward modeling operator, and λ_1 and λ_2 balance the data loss and the physical consistency loss.

Compared with a purely data-driven method, this hybrid formulation can encourage the predicted velocity models to be consistent with the observed seismic data under the wave-equation forward model. Therefore, it may reduce nonphysical artifacts, improve reconstruction accuracy, enhance robustness to noisy data, and improve generalization to unseen velocity models.

In addition, velocity model reparameterization methods [45] based on CNNs have attracted increasing attention. These methods use the network architecture itself as an implicit regularization to improve the stability of inversion. TFWI can also be used as a reparameterization network to generate velocity models. We are currently extending TFWI to this reparameterization framework. Owing to the introduction of the self-attention mechanism, TFWI provides stronger modeling capability than CNN-only architectures, which has the potential to further improve inversion performance.

5. Conclusions

In this work, we propose TFWI, a novel data-driven method for seismic FWI based on the ViT. Unlike CNNs, our ViT-based encoder extracts global information from seismic data using the attention mechanism. We also enhance the ViT patch extraction method for velocity model generation and incorporate residual connections into the decoder for feature fusion, preserving critical information. We compare TFWI with existing data-driven methods for data quantization and velocity model visualization, demonstrating its superior performance. Furthermore, we discuss the impact of different loss functions and confirm TFWI's robustness through noise experiments.

Although the proposed method achieves promising performance, several issues remain for future investigation. In the current framework, the ViT is employed only as an encoder. Given the strong feature modeling capability of the self-attention mechanism, future work will explore incorporating the ViT into both the encoder and decoder to further improve reconstruction performance. In the Discussion section, we have presented possible extensions of the proposed framework. However, since these extensions would require substantially higher memory usage and computational cost, we only provide feasible architectural modifications without conducting additional experiments. In future work, we will investigate more efficient computational modules for TFWI and perform systematic experiments on the above extended applications. In addition, since the ViT operates on image patches, it may overlook spatial information within each patch. Future research will focus on enhancing the ability of TFWI to capture spatial information within patches, thereby enabling more effective utilization of seismic data.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 42274166), the Joint Program of the Liaoning Provincial Science and Technology Plan (Key R&D Program Project) (No. 2025JH2/101800002), and the Fundamental Research Funds for the Central Universities of China (Nos. 3132026192 and 3132026194).

Conflict of interest

The authors declare there are no conflicts of interest.

References

1. J. Virieux, S. Operto, An overview of full-waveform inversion in exploration geophysics, *Geophysics*, **74** (2009), WCC1–WCC26. https://doi.org/10.1190/1.3238367
2. D. C. Liu, J. Nocedal, On the limited memory BFGS method for large scale optimization, *Math. Program.*, **45** (1989), 503–528. https://doi.org/10.1007/BF01589116
3. H. S. Fu, Y. Zhang, X. L. Li, Adaptive overcomplete dictionary learning-based sparsity-promoting regularization for full-waveform inversion, *Pure Appl. Geophys.*, **178** (2021), 1–12. https://doi.org/10.1007/s00024-021-02662-w
4. R. Fletcher, C. M. Reeves, Function minimization by conjugate gradients, *Comput. J.*, **7** (1964), 149–154. https://doi.org/10.1093/comjnl/7.2.149
5. P. Mora, Nonlinear two dimensional elastic inversion of multioffset seismic data, *Geophysics*, **52** (1987), 1211–1228. https://doi.org/10.1190/1.1442384
6. A. N. Tikhonov, A. V. Goncharsky, V. V. Stepanov, A. G. Yagola, Regularization methods, in *Numerical Methods for the Solution of Ill-Posed Problems*, Springer Netherlands, (1995), 7–63. https://doi.org/10.1007/978-94-015-8480-7_2
7. A. Asnaashari, R. Brossier, S. Garambois, F. Audebert, P. Thore, J. Virieux, Regularized seismic full-waveform inversion with prior model information, *Geophysics*, **78** (2013), R25–R36. https://doi.org/10.1190/geo2012-0104.1
8. E. Esser, L. Guasch, T. van Leeuwen, A. Y. Aravkin, F. J. Herrmann, Total variation regularization strategies in full-waveform inversion, *SIAM J. Imaging Sci.*, **11** (2018), 376–406. https://doi.org/10.1137/17M111328X
9. W. Zhang, M. Z. Xu, H. X. Yang, X. Wang, S. S. Zheng, X. Li, Data-driven deep learning approach for thrust prediction of solid rocket motors, *Measurement*, **225** (2024), 114051. https://doi.org/10.1016/j.measurement.2023.114051
10. X. W. Wang, Y. H. Wang, X. C. Su, L. Wang, C. Lu, H. J. Peng, et al., Deep reinforcement learning-based air combat maneuver decision-making: Literature review, implementation tutorial and future direction, *Artif. Intell. Rev.*, **57** (2024). https://doi.org/10.1007/s10462-023-10620-2
11. M. Araya-Polo, J. Jennings, A. Adler, T. Dahlke, Deep-learning tomography, *Lead. Edge*, **37** (2018), 58–66. https://doi.org/10.1190/tle37010058.1

12. K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, *Neural Netw.*, **2** (1989), 359–366. https://doi.org/10.1016/0893-6080(89)90020-8

13. C. Song, T. Alkhalifah, Wavefield reconstruction inversion via physics-informed neural networks, *IEEE Trans. Geosci. Remote Sens.*, **60** (2021), 1–12. https://doi.org/10.1109/TGRS.2021.3123122

14. F. X. Wu, Y. Li, Z. W. Fu, Q. L. He, Y. Chen, B. Han, et al., Robust physics-informed full waveform inversion via a transformer-based autoencoder, *IEEE Trans. Geosci. Remote Sens.*, **63** (2025), 1–16. https://doi.org/10.1109/TGRS.2025.3597592

15. A. Adler, M. Araya-Polo, T. Poggio, Deep learning for seismic inverse problems: Toward the acceleration of geophysical analysis workflows, *IEEE Signal Process. Mag.*, **38** (2021), 89–119. https://doi.org/10.1109/MSP.2020.3037429

16. Y. Wu, Y. Z. Lin, Inversionnet: An efficient and accurate data-driven full waveform inversion, *IEEE Trans. Comput. Imaging*, **6** (2019), 419–433. https://doi.org/10.1109/TCI.2019.2956866

17. S. H. Feng, Y. Z. Lin, B. Wohlberg, Multiscale data-driven seismic full-waveform inversion with field data study, *IEEE Trans. Geosci. Remote Sens.*, **60** (2021), 1–14. https://doi.org/10.1109/TGRS.2021.3114101

18. Z. P. Zhang, Y. Z. Lin, Data-driven seismic waveform inversion: A study on the robustness and generalization, *IEEE Trans. Geosci. Remote Sens.*, **58** (2020), 6900–6913. https://doi.org/10.1109/TGRS.2020.2977635

19. C. Y. Deng, S. H. Feng, H. C. Wang, X. T. Zhang, P. Jin, Y. N. Feng, et al., OpenFWI: Large-scale multi-structural benchmark datasets for full waveform inversion, in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., (2022), 6007–6020. https://doi.org/10.52202/068431-0435

20. X. Y. Zhang, F. Min, S. L. Pan, Q. Xu, X. Y. Min, G. J. Song, et al., DD-Net: Dual decoder network with curriculum learning for full waveform inversion, *IEEE Trans. Geosci. Remote Sens.*, **62** (2024), 1–17. https://doi.org/10.1109/TGRS.2024.3358492

21. X. Yan, K. Wu, Z. Q. J. Xu, Z. Ma, A deep learning approach for solving the inverse problem of the wave equation, *CSIAM Trans. Appl. Math.*, **7** (2026), 29–61. https://doi.org/10.4208/csiam-am.SO-2024-0027

22. W. Ding, K. Ren, L. Zhang, Coupling deep learning with full waveform inversion, preprint, arXiv:2203.01799.

23. F. S. Yang, J. W. Ma, FWIGAN: Full-waveform inversion via a physics-informed generative adversarial network, *J. Geophys. Res.: Solid Earth*, **128** (2023), e2022JB025493. https://doi.org/10.1029/2022JB025493

24. S. L. Yang, T. Alkhalifah, Y. X. Ren, B. Liu, Y. Y. Li, P. Jiang, Well-log information-assisted high-resolution waveform inversion based on deep learning, *IEEE Trans. Geosci. Remote Sens. Lett.*, **20** (2023), 1–5. https://doi.org/10.1109/LGRS.2023.3234211

25. P. Jin, X. T. Zhang, Y. P. Chen, S. X. Huang, Z. C. Liu, Y. Z. Lin, Unsupervised learning of full-waveform inversion: Connecting cnn and partial differential equation in a loop, preprint, arXiv:2110.07584.

26. J. Ren, J. Li, C. Liu, S. Chen, L. Liang, Y. Liu, Deep learning with physics-embedded neural network for full waveform ultrasonic brain imaging, *IEEE Trans. Med. Imaging*, **43** (2024), 2332–2346. https://doi.org/10.1109/TMI.2024.3363144

27. X. S. Peng, X. M. Zhang, Y. P. Li, B. Q. Liu, Research on image feature extraction and retrieval algorithms based on convolutional neural network, *J. Vis. Commun. Image Represent.*, **69** (2020), 102705. https://doi.org/10.1016/j.jvcir.2019.102705

28. K. Han, Y. H. Wang, H. T. Chen, X. H. Chen, J. Y. Guo, Z. H. Liu, et al., A survey on vision transformer, *IEEE Trans. Pattern Anal. Mach. Intell.*, **45** (2022), 87–110. https://doi.org/10.1109/TPAMI.2022.3152247

29. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, et al., Attention is all you need, in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., (2017), 5998–6008.

30. S. K. Mondal, H. X. Zhang, H. D. Kabir, K. Ni, H. N. Dai, Machine translation and its evaluation: A study, *Artif. Intell. Rev.*, **56** (2023), 10137–10226. https://doi.org/10.1007/s10462-023-10423-5

31. Y. Li, X. Wu, J. C. Wang, Y. M. Bo, F. Ni, C. H. Jiang, Differential attention vision transformer with adaptive spatial feature conditioning for remote sensing scene classification, *Pattern Recogn.*, **178** (2026), 113461. https://doi.org/10.1016/j.patcog.2026.113461

32. Q. Wang, S. T. Liu, J. Chanussot, X. L. Li, Scene classification with recurrent attention of VHR remote sensing images, *IEEE Trans. Geosci. Remote Sens.*, **57** (2018), 1155–1167. https://doi.org/10.1109/TGRS.2018.2864987

33. M. Chen, A. Radford, R. Child, J. Wu, H. Jun, D. Luan, et al., Generative pretraining from pixels, in *Proceedings of the 37th International Conference on Machine Learning*, PMLR, (2020), 1691–1703.

34. A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, et al., An image is worth 16x16 words: Transformers for image recognition at scale, preprint, arXiv:2010.11929.

35. R. Strudel, R. Garcia, I. Laptev, C. Schmid, Segmenter: Transformer for semantic segmentation, in *2021 IEEE/CVF International Conference on Computer Vision*, IEEE, (2021), 7242–7252. https://doi.org/10.1109/ICCV48922.2021.00717

36. Z. Y. Wang, S. Wang, L. Y. Wu, D. Y. Liu, L. Gao, L. Qi, et al., Entroformer: An entropy-based sparse vision transformer for real-time semantic segmentation, *Comput. Vis. Image Underst.*, **260** (2025), 104482. https://doi.org/10.1016/j.cviu.2025.104482

37. N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, S. Zagoruyko, End-to-end object detection with transformers, in *European Conference on Computer Vision*, Springer, (2020), 213–229. https://doi.org/10.1007/978-3-030-58452-8_13

38. C. Hao, Z. T. Yu, X. Liu, J. Xu, H. J. Yue, J. Y. Yang, A simple yet effective network based on vision transformer for camouflaged object and salient object detection, *IEEE Trans. Image Process.*, **34** (2025), 608–622. https://doi.org/10.1109/TIP.2025.3528347

39. Y. J. Xiang, Z. L. Wang, Z. Song, R. Huang, G. J. Song, F. Min, Seismictransformer: An attention-based deep learning method for the simulation of seismic wavefields, *Comput. Geosci.*, **190** (2024), 105629. https://doi.org/10.1016/j.cageo.2024.105629

40. H. Z. Wang, J. Lin, Y. Li, X. T. Dong, X. Q. Tong, S. P. Lu, Self-supervised pretraining transformer for seismic data denoising, *IEEE Trans. Geosci. Remote Sens.*, **62** (2024), 1–25. https://doi.org/10.1109/TGRS.2024.3368282

41. C. Y. Ning, B. Y. Wu, B. H. Wu, Transformer and convolutional hybrid neural network for seismic impedance inversion, *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, **17** (2024), 4436–4449. https://doi.org/10.1109/JSTARS.2024.3358610

42. B. Engquist, A. Majda, Absorbing boundary conditions for the numerical simulation of waves, *Math. Comp.*, **31** (1977), 629–651. https://doi.org/10.1090/S0025-5718-1977-0436612-4

43. A. Ali, H. Touvron, M. Caron, P. Bojanowski, M. Douze, A. Joulin, et al., Xcit: Cross-covariance image transformers, in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., (2021), 20014–20027.

44. X. Zhang, X. Zhou, M. Lin, J. Sun, Shufflenet: An extremely efficient convolutional neural network for mobile devices, in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, (2018), 6848–6856. https://doi.org/10.1109/CVPR.2018.00716

45. A. Dhara, M. K. Sen, Elastic full-waveform inversion using a physics-guided deep convolutional encoder-decoder, *IEEE Trans. Geosci. Remote Sens.*, **61** (2023), 1–18. https://doi.org/10.1109/TGRS.2023.3294427

AIMS Press