# EikoNet: Solving the Eikonal Equation With Deep Neural Networks

Jonathan D. Smith, Kamyar Azizzadenesheli, and Zachary E. Ross

***Abstract*— The recent deep learning revolution has created enormous opportunities for accelerating compute capabilities in the context of physics-based simulations. In this article, we propose EikoNet, a deep learning approach to solving the Eikonal equation, which characterizes the first-arrival-time field in heterogeneous 3-D velocity structures. Our grid-free approach allows for rapid determination of the travel time between any two points within a continuous 3-D domain. These travel time solutions are allowed to violate the differential equation—which casts the problem as one of optimization—with the goal of finding network parameters that minimize the degree to which the equation is violated. In doing so, the method exploits the differentiability of neural networks to calculate the spatial gradients analytically, meaning that the network can be trained on its own without ever needing solutions from a finite-difference algorithm. EikoNet is rigorously tested on several velocity models and sampling methods to demonstrate robustness and versatility. Training and inference are highly parallelized, making the approach well-suited for GPUs. EikoNet has low memory overhead and further avoids the need for travel-time lookup tables. The developed approach has important applications to earthquake hypocenter inversion, ray multipathing, and tomographic modeling, as well as to other fields beyond seismology where ray tracing is essential.**

***Index Terms*— Geophysics, partial differential equations (PDEs), ray tracing, travel time.**

## I. Introduction

THE 3-D ray tracing is a fundamental component of modern seismology, having direct applications to earthquake hypocenter inversions [8], seismic tomography [29], and earthquake source properties [4]. These derived products further form the basis for many downstream seismological applications. The Eikonal equation is a well-known nonlinear partial differential equation (PDE) that characterizes the first-arrival-time field for a given source location in a 3-D medium [17]. The Eikonal formulation can be solved with several finite-difference algorithms [18], [20], [27], with varying computational demands and stabilities to the solutions.

Manuscript received March 25, 2020; revised August 11, 2020 and October 16, 2020; accepted November 6, 2020. This work was supported in part by United States Geological Survey (USGS). The work of Kamyar Azizzadenesheli was supported in part by Raytheon and in part by Amazon Web Services. *(Corresponding author: Jonathan D. Smith.)*

Jonathan D. Smith and Zachary E. Ross are with the Seismological Laboratory, California Institute of Technology, Pasadena, CA 91125 USA (e-mail: jon_smith83@hotmail.co.uk).

Kamyar Azizzadenesheli is with the Department of Computer Science, Purdue University, West Lafayette, IN 47907 USA.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TGRS.2020.3039165.

Digital Object Identifier 10.1109/TGRS.2020.3039165

Recent advances in deep learning have shown to be extremely promising in the context of physics-based simulations [7], [31]. This technology has started to be applied to geophysics as well, for example, to predict the acoustic wave response of a medium given a velocity model as input [15], forecast the next time step of a wavefield conditional on its history [15], and accelerate viscoelastic simulations [5]. However, these techniques rely on inputs from precomputed physics-based models, which could themselves contain modeling-based artifacts and input bias. Instead, we wish to learn the underlying physics by incorporating the formulations into the neural network architecture and loss function. These physics-based procedures [1], [13], [19], [22], [28] take advantage of the backpropagation procedure to compute the gradient of the neural network output relative to the input terms. Such physics-informed neural networks (PINNs) [19], [22] are mesh independent, giving a continuous function output related to the inputs. These techniques have been used to learn the parameterization for formulations of the Burgers, Schrodinger, and Navier–Stokes equations, with comparisons made to the numerical derivatives [22].

In this article, we propose EikoNet, an approach to solving the factored Eikonal equation. EikoNet can be trained to learn the travel time between any two points in a truly continuous 3-D medium, avoiding the use of grids. We leverage the differentiability of neural networks to analytically compute the spatial gradients of the travel-time field and train the network to minimize the difference between the true and the predicted velocity model for the factored Eikonal formulation. EikoNet is massively parallelized and therefore well-suited for GPUs, has low memory overhead, and avoids the need for travel-time lookup tables. In addition, EikoNet has several novel advantages that are not currently offered with conventional finite-difference schemes.

## II. Eikonal Formulation

The Eikonal equation is a nonlinear first-order PDE representing a high-frequency approximation to the propagation of waves in heterogeneous media [17]. The equation takes the general form

$$\|\nabla T_{s\to r}\|^2 = \frac{1}{V(\vec{x}_r)^2} = S(\vec{x}_r)^2 \tag{1}$$

where $\|\cdot\|^2$ is the Euclidean norm, $T_{s\to r}$ is the travel time through the medium from a source location $s$ to a receiver location $r$, $V_r$ is the velocity of the medium at the receiver

location, $S_r$ is the slowness of the medium at the receiver location, and $\nabla_r$ is the gradient at the receiver location.

The value of travel time is computed by minimizing the misfit of a travel-time field that satisfies the user imposed velocity model, with the additional boundary condition that the travel time at the source location equals zero, $T_{s\to s} = 0$. Solutions to 1 have a strong singularity at the source location [25], leading to numerical errors close to the source. To mitigate such singularity effects, a factored formulation is often used, with solutions representing the travel-time deviation from a homogeneous medium with $V = 1$ [25]. The factored travel-time form can then be represented by

$$T_{s\to r} = T_0 \cdot \tau_{s\to r} \tag{2}$$

where $T_0 = \|\vec{x_r} - \vec{x_s}\|$, representing the distance function from the source location, and $\tau$ is the deviation of the travel-time field from a model travel time with homogeneous unity velocity. Substituting the formulation of 2 into 1 and expanding using the chain rule, then the velocity can be represented by

$$V(\vec{x_r}) = \left[T_0^2 \| \underset{r}{\nabla} \tau_{s\to r}\|^2 + 2\tau_{s\to r}(\vec{x_r} - \vec{x_s}) \cdot \underset{r}{\nabla} \tau_{s\to r} + \tau_{s\to r}^2\right]^{-\frac{1}{2}}. \tag{3}$$

The partial differential terms in 3 are typically solved using a finite-difference approach and will be discussed in Section IV.

## III. Methods

### A. Network Architecture and Training

Our approach to solving the Eikonal equation trains a deep neural network, $f_\theta$, to predict the travel-time field, $\tau$, between an input pair of source–receiver coordinates, $\vec{x} = [\vec{x_s}, \vec{x_r}]$. The deviation of the travel-time field is then represented by

$$\tau = f_\theta(\vec{x}) \tag{4}$$

with the corresponding travel time between source and receiver location represented by 2.

If $\tau$ was known, a neural network could be trained for catalog source–receiver pairs. However, $\tau$ itself is unknown, being the solution to the equation that we want to solve. Instead, only the velocity model and a differential equation specifying how $\tau$ relates to $V$ are known, but this can be used to train the neural network. Thus, we cast the problem as one where we aim to accurately predict the local velocity, assuming that the output of $f_\theta$ is indeed $\tau$, and use this value to compute $V$ from 3, defining this predicted velocity as $\hat{V}$. Here, we exploit the differentiability of deep neural networks to analytically determine the spatial gradient of 4 with respect to $\vec{x_r}$. This is possible because the layers and activations of the neural network are chosen precisely to be analytically differentiable. In this study, we use Pytorch to calculate the gradients and perform all network training.

Solving the factored Eikonal equation is therefore reduced to training a neural network with supervision on the velocity model, by iteratively updating the parameters $\theta$ to minimize some loss function. Once trained, the Eikonal equation is no longer needed, as $f_\theta$ outputs $\tau$ directly [see Fig. 1(a)]. In the process, the details of the velocity model will be encoded in the network, requiring the network to be retrained if the velocity model is to be changed. As such, the algorithm is fully capable of solving the Eikonal equation from scratch, without needing to run a separate finite-difference simulation to generate training data.

The model architecture is a feedforward network consisting of a series of residual blocks with fully connected layers [10] followed by nonlinear units [see Fig. 1(a)]. We use an exponential linear unit (ELU) as the activation [3] function on all hidden layers. This activation was chosen as the method is aligned with the notion of natural gradients, which uses the geometry induced by Fisher information to adjust the gradient direction [3], pushing the activation to stay in a region that is not saturated.

The optimal number of layers (or residual blocks) generally depends on the complexity of the training data. Here, we use ten residual blocks, which was found to provide the best results for the tests herein, comprising a total of 7 913 249 parameters to be optimized. For highly complex velocity structures, this number may need to increase.

The neural network learns the travel time between any source–receiver pair and must contain adequate sampling from across the 3-D medium. The input features are organized into a vector of six components

$$\vec{x} = [X_s, Y_s, Z_s, X_r, Y_r, Z_r] \tag{5}$$

where $X$, $Y$, and $Z$ are the Cartesian coordinates of the source or receiver. The features are paired with the seismic velocity at the receiver location

$$y = V(X_r, Y_r, Z_r). \tag{6}$$

A training data set therefore consists of many $(\vec{x}, y)$ samples, taken from across the 3-D volume [see Fig. 1(c)]. We discuss how these data sets are constructed in Section III-B.

The misfit between the predicted $\hat{V}$, as determined from 3, and observed $V$ velocities are then minimized using a mean-squared error loss function

$$L = \|V - \hat{V}\|^2. \tag{7}$$

We use the Adam optimization algorithm for training with a learning rate of $5 \times 10^{-5}$ [12] [see Fig. 1(d)]. The batch and data set size are set to 752 and $10^6$, respectively, with their variability discussed further in Section VI-B. The data set is separated into training and validation data, with the validation data set at 10% of the total size. An additional test data set is created representing $10^4$ source–receiver pairs that are blind to the user prior to loss calculation. For all the simulations, we use a single Nvidia Tesla V100 GPU, with models taking 66 s per epoch given the parameters mentioned earlier. The effects of parameter values on computation cost can be found in Section VI-B

Once trained, the network can be applied to a series of user-defined source–receiver pairs to determine the travel time and predicted velocity, as shown in Fig. 1(e).

### *B. Building a Training Data Set*

Our approach builds a training data set by randomly sampling source and receiver points from the continuous 3-D medium and labeling each with the velocity at the receiver location. Since the velocity model is gridded, we use linear interpolation to map these values to a continuous domain. In this study, we explore three different methods for sampling the velocity model. In these formulations, we investigate two sampling techniques from the classical theory of Bertrand paradox: sampling random points across the model space and sampling points at random distances.

*1) Random Distance:* The data set is composed of a series of source locations selected randomly across the model space. Once a source point is selected, a receiver point in space is sampled at a random distance away from the source location along a random vector. This method inherently allows the source–receivers pairs to have a distance distribution that is uniform across the model space.

*2) Random Locations:* The data set could also be composed of a series of randomly selected source and receiver locations. This allows for a more random distribution of points across the model space, but inherently has a nonuniform distribution of distances between source and receiver points, with a lower sampling at short distances, as expressed by the Bertrand paradox.

*3) Weighted Sampling:* In complex velocity contrasting models, the training procedure could quickly learn areas of simple velocity variations, and we act to improve the training efficiency of this procedure by allowing dynamic resampling of the training source–point pairs for values of greatest misfit. For the first epoch of training, we minimize the misfit between the actual and predicted velocity estimates using an L2-norm but also determine an importance weight parameter, $w$, for each training sample defined by

$$w = \frac{\left|\hat{V} - V\right|}{V} \tag{8}$$

where $\hat{V}$ is the neural-network predicted velocity value and $V$ is the actual imposed velocity model. The weight value is high for the samples with the greatest relative misfit. For subsequent training epochs, training samples are selected based on the weight value normalized by the maximum weight in the training data set. To mitigate stagnation in the extremes of the weight distribution, we project the weights between a user-defined minimum and maximum, represented by $[\min, \max] = [0.1, 0.9]$. This bound was chosen to mitigate the undersampling of regions with low misfit and oversampling of regions that could contain singularities.

### *C. Model Verification*

The accuracy of the solution to the Eikonal equation is given directly by the loss, which quantifies the degree to which the solution violates the PDE in a least-squares sense. Thus, after training is finished, we can predict the travel time to all points desired within the 3-D medium and use 1 to calculate the learned velocity model. Comparing the learned velocity model to the actual velocity model provides a visual and rigorous quantitative approach to understanding the accuracy of the solution [see Fig. 1(e)].

## IV. Baselines

In this section, we discuss some current approaches for solving the Eikonal formulation. While there have been many techniques developed for solving the Eikonal equation, a number of these can be broadly classified as either fast marching methods (FMMs) or fast sweeping methods (FSMs). The FMM is a grid-based numerical scheme that uses a special finite-difference operator to track the evolution of the minimum travel time using an advancing interface scheme [20], [23]. In contrast, the FSM is an iterative method for solving the Eikonal equation via upwind differencing using a Gauss–Seidel iteration scheme [30]. This method groups the wavefronts by the sweeping directions in addition to meeting the requirement of increasing travel times orthogonal to the wavefronts. Typically, this sweeping procedure is implemented six times, for each of the positive and negative directions in the Cartesian coordinate system. As this method does not have to track an advancing interface, the computational cost is lower than the FMM, although this procedure can breakdown in the presence of strongly heterogeneous velocity structures [2]. A more in-depth comparison between the FMM and FSM methods computational costs and solution misfit can be found in [2].

Throughout this study, we utilize the python FMM as a baseline since it is less sensitive to sharp velocity contrasts with only minor differences in the computational time to the FSMs [2]. To compute the FMM travel times, we use the CPU toolkit scikit-fmm (https://pythonhosted.org/scikit-fmm/) to formulate the travel time from a source location on a receiver geometry at 0.1 km grid spacing in the $X$, $Y$, and $Z$ dimensions. The root-mean-squared (rms) travel-time difference between the FMM and EikoNet approaches is calculated for each of the models. The FMM simulations are only used for comparison and are not used in the training of the neural network.

## V. Velocity Model Experiments

Outlined are a series of experiments designed to demonstrate the versatility of our approach for learning travel-time fields in complex 3-D velocity models. We consider four different velocity models and examine the performance of the trained network. Each network is trained with a batch size of 752, using a data set with $10^6$ random distance source–point pairs with a weighting range of $[0.1, 0.9]$. The effects of dynamic sampling and weighting are discussed further in Section VI.

### *A. Homogeneous Velocity*

The first model we consider is a homogeneous 3-D velocity structure with a value of 5 km/s [see Fig. 2(a)]. For demonstration purposes, a source is placed at $[X, Y, Z] = [10\text{ km}, 10\text{ km}, 1\text{ km}]$, with both $X$–$Z$ and $X$–$Y$ slices of the travel-time field shown. At each receiver point, we plot the learned velocity using 1 and use this to determine the

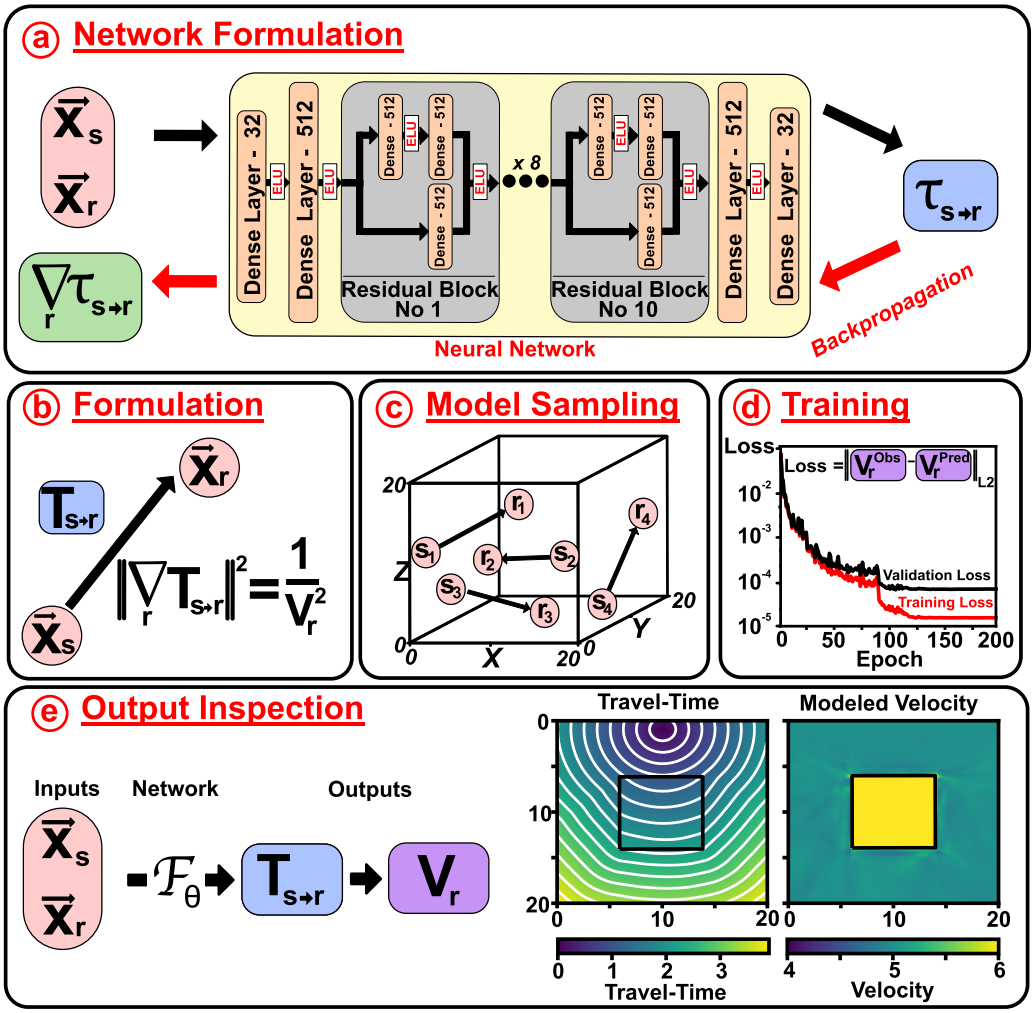



Fig. 1. Overview of processing workflow. (a) Neural network architecture composed of fully connected layers and residual blocks. Each residual block is composed of three fully connected layers with 512 neurons. ELU activations are applied on all hidden layers. (b) Summary of Eikonal equation for $T_{s\to r}$ and $V_r$. (c) Sampling of source–receiver pairs across the 3-D volume to build the training data set. (d) Network training through the minimization of loss function relating predicted and observed velocity values. (e) Inspection of neural network outputs by passing user-defined source–receiver pairs.

percent error. For comparison, we also show the FMM solution calculated on a grid of receivers with 0.1 km grid spacing in the $X$, $Y$, and $Z$ dimensions, providing a travel-time rms with the EikoNet travel times. The training loss goes to zero after 110 training epochs, showing that the network has learned the travel-time field perfectly. When comparing the EikoNet solution with the analytical solution, the rms travel-time error is zero. The FMM solution has an rms travel-time error of 0.0113 s.

### B. Graded Velocity

Next, we consider a velocity model that increases linearly with depth [see Fig. 2(b)]. The model increases from 3 km/s at the surface to 7 km/s at 20 km depth. Comparison with the conventional finite-difference scheme shows good agreement, with a mean difference between the imposed and recovered velocity models of 0.00209 km/s. The rms travel time with the FMM solution reaches a value of 0.0189 and 0.0072 s with the analytical solution.

### C. Block Model

The third test conducted is a 3-D model containing a central cube embedded within a homogeneous background [see Fig. 2(c)]. The cube has a constant velocity of 7 km/s, whereas the background velocity is 5 km/s. Here, the neural network approach has excellent misfit between the actual and learned velocities, with only minor disagreement close to sharp velocity gradients. The neural-network travel-time field is similar to that of the finite-difference scheme, with the effects of the sharp velocity contrast shown in the deflection of the travel-time fronts, a low travel-time rms of 0.0311 s, and a low mean velocity difference of 0.094 km/s. This model demonstrates that the neural network approach is able to reconcile even difficult cases with the sharp velocity changes of 20% of the mean value.

### D. Checkerboard Velocity

The fourth test conducted contains a 3-D checkerboard, which we use to demonstrate that the proposed algorithm is

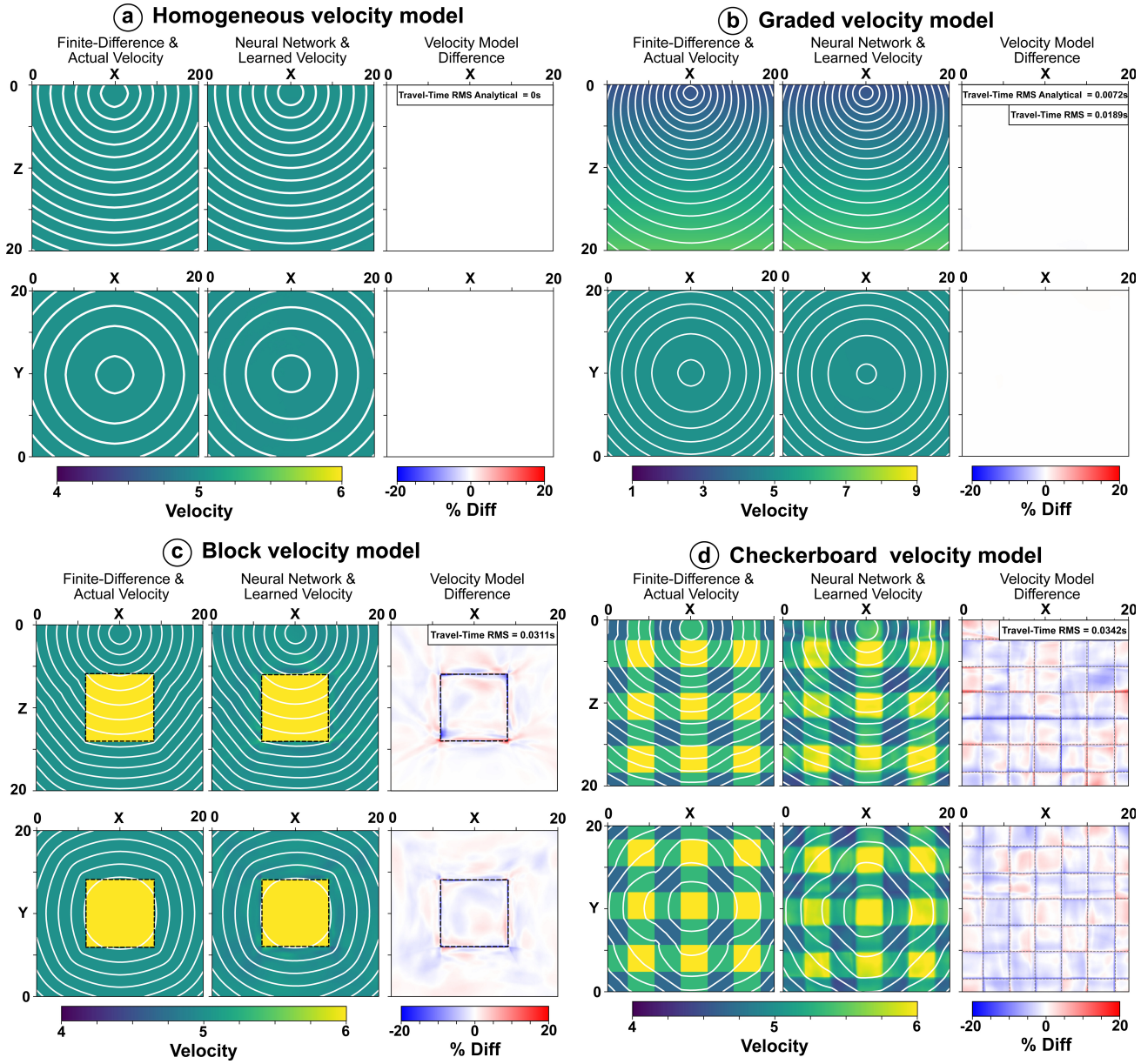



Fig. 2. Velocity model experiments with comparison to finite-difference and imposed velocity models. (Left) $X$–$Z$ and $X$–$Y$ slice from the imposed velocity model, overlayed by the finite-difference expected travel-time. (Middle) $X$–$Z$ and $X$–$Y$ slice from the neural network recovered velocity model and neural-network travel time. (Right) $X$–$Z$ and $X$–$Y$ slice velocity model differences between the imposed and recovered velocity models.

able to reconcile positive and negative velocity anomalies varying spatially within the domain. The velocity varies between 6 and 4 km/s, with a grid spacing of 6 km, meaning that the model is not symmetrical about the central point [see Fig. 2(d)]. We compare the solution with the finite-difference scheme and actual velocity model, showing that the deep learning approach is able to reconcile the velocity model with a mean velocity difference of 0.19 km/s and a travel-time rms of 0.0342 s.

### *E. Industry Velocity Model—Marmousi2*

The final velocity model that we test is the Marmousi2 2-D model. The Marmousi2, expanding on the original Marmousi model [26], is based on a geophysical profile through the North Quenguela Trough in the Cuanza Basin, Angola. This model represents a 17 km $\times$ 3.5 km cross section of strongly heterogeneous velocity contrasts attributed to changes in subsurface geology, offsets from fault structures, and reservoir levels [14]. We investigate the specific use of EikoNet on the S-wave velocity, as shown in Fig. 3(a). Due to the complexity in the velocity structure, we expand our network to use 20 residual blocks, to enable EikoNet to encapsulate even the smallest resolution velocity contrasts. The network is trained across 400 epochs with 1000 batches of 752 pairs randomly sampled per epoch. We apply this method to minimize the undersampling on sharp velocity contrasts, expected from a fixed sample data set. Training for the 400 epochs required a total of $\sim$6 h on a single Nvidia V100 GPU. Fig. 3(a)–(c) shows the EikoNet travel-time and velocity fields for a series of source locations, on a 0.001-km grid spacing. For comparison, FMM finite-difference travel-time simulations

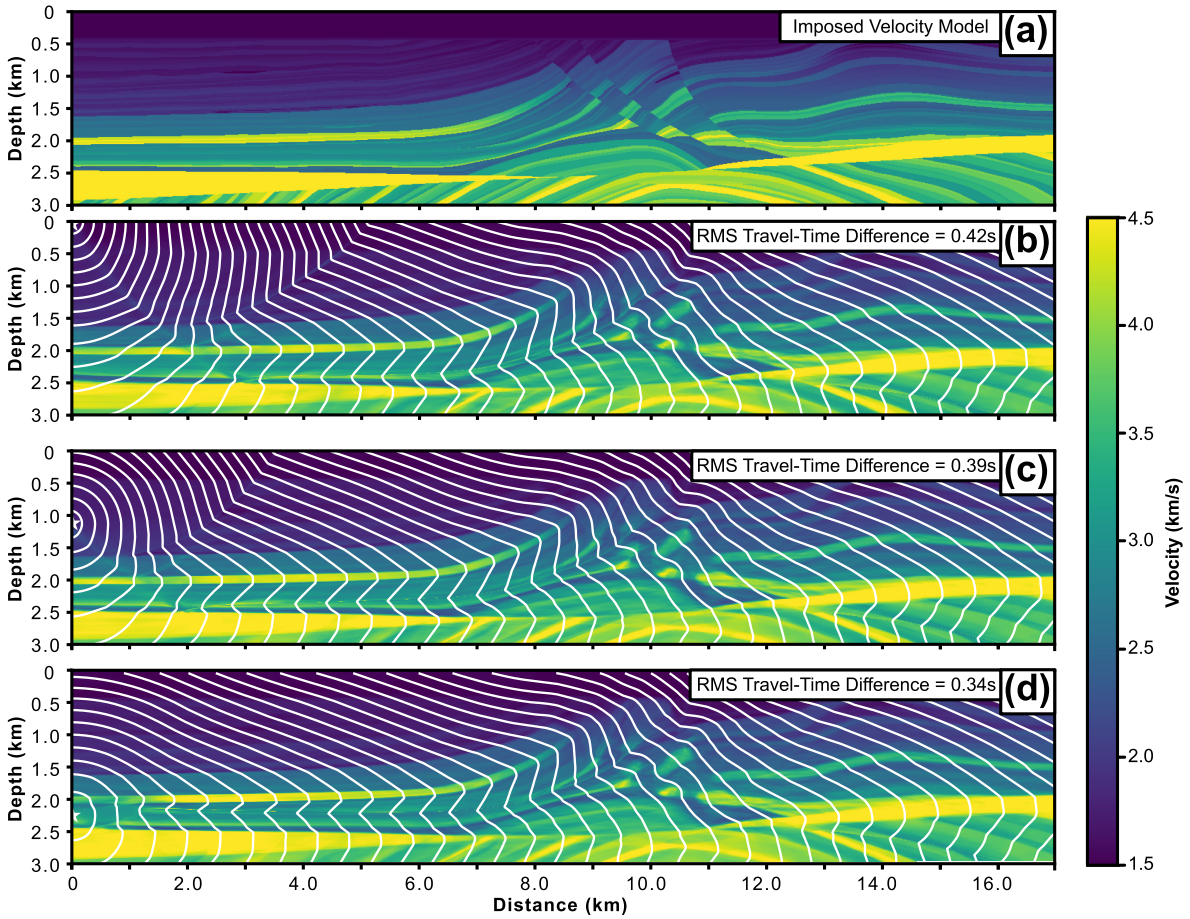



Fig. 3. Marmousi2 2-D travel-time formulations using EikoNet and finite-difference methods. (a) Colormap of the imposed 2-D velocity model. (b) Recovered travel-time and velocity model fields from a source location at [0 km, 0 km] to a point grid at 0.001 km spacing. Colormap represents the recovered velocity and white contours represent the travel time at 0.1-s spacing. (c) and (d) Similar plot to (b) but with the different source locations of [0 km, 1.1 km] and [0 km, 2.25 km] respectively.

are created for each of the source locations on the same point grid spacing. Travel-time rms is computed with values ranging between 0.34 and 0.42 s and a mean velocity model difference of 0.17 km/s. These simulations demonstrate that the EikoNet formulation is able to reconcile the broad-scale sharp velocity contrasts of the complex subsurface geology, as shown on the edges of the velocity model, but are unable to reconcile the sharp small-scale velocity contrasts, as shown at the center of the model. An underrepresentation of these features is expected to be attributing to the 0.34–0.42 s travel-time variations between the FMM and EikoNet travel-time formulations. Although misfit still remains, we envisage that future research in more complex dynamic sampling schemes, with an adaptive sampling around sharp velocity contrasts and velocity misfit, could help mitigate the underrepresentation of these small-scale sharp velocity features.

## VI. Sampling Experiments

Having demonstrated that deep neural networks can indeed learn to directly solve the Eikonal equation, we now examine the effect of the sampling scheme on the learned solution. Here, we use the block velocity model from Section V and train separate models using each of the three sampling techniques described in Section III-B. We also separately test how the total number of training samples and batch size affect the performance.

### A. Sampling Schemes

Fig. 4 shows the validation results for each of the three sampling methods. The weighted random sampling approach achieves the lowest loss of the three sampling methods, yet converges in a similar number of training epochs. However, the other two methods still perform well, as the differences in validation loss are relatively small. For the random location procedure, there is a greater misfit closer to the source location, expected due to the bias of the length scale to longer distances due to the Bertrand paradox, and as such, we use the random distance metric for sampling. Although the weighted sampler has little effect on the loss values and travel-time rms value, we expect that the weighted random sampling will be of increasing importance in very complex 3-D models and recommend it for selection of source–point pairs.

### B. Size of Training Data Set

We investigate how the number of training samples and batch size affect the network performance (see Fig. 5).

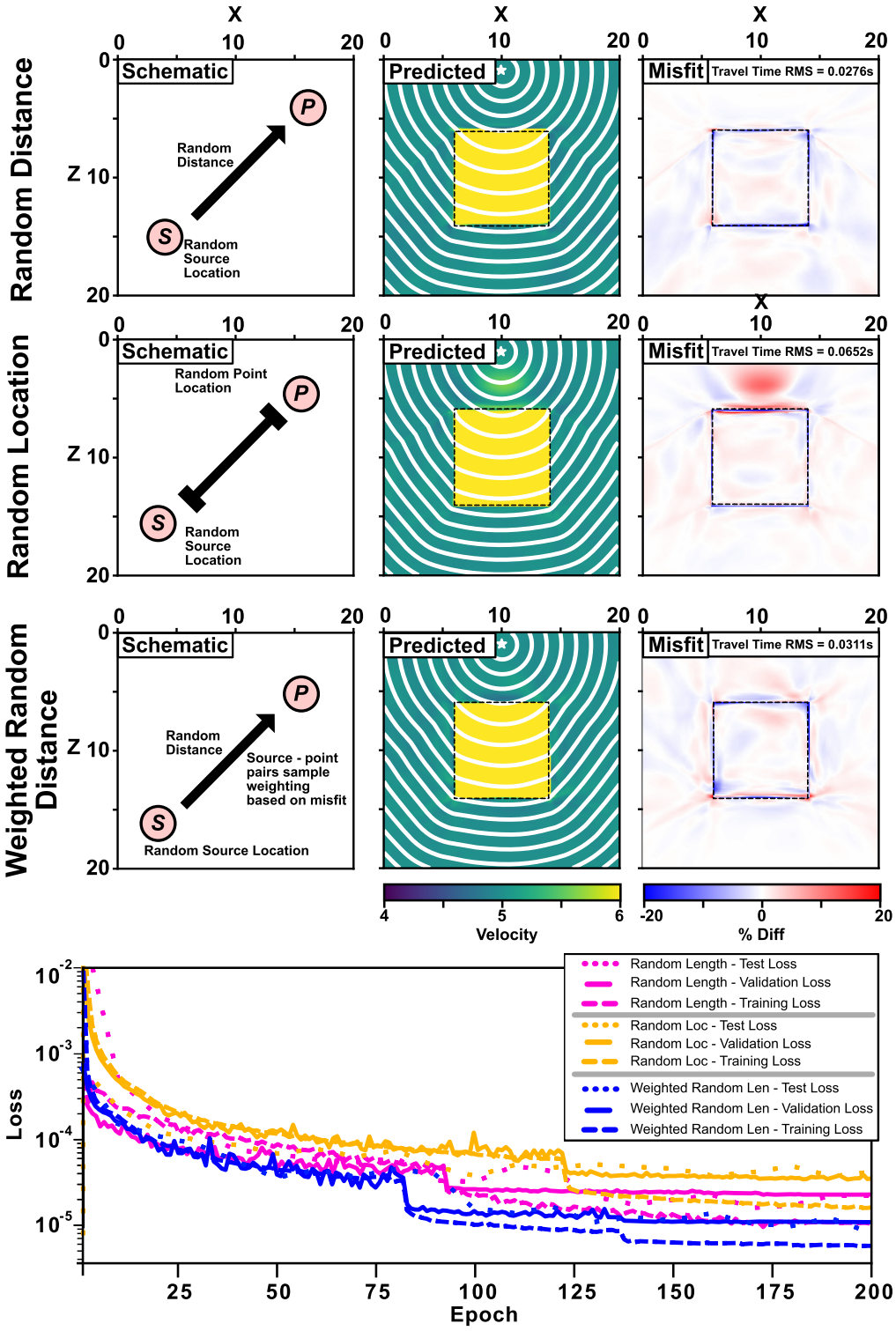



Fig. 4. Sampling schemes and their influence on the network performance. (Left Column) Schematic of the different type of sampling. (Middle Column) Learned velocity model for each simulation. (Right Column) Misfit between the predicted and the imposed velocity model. (Bottom) Training, validation, and testing loss for each of the simulations.

We rerun the simulation with three data set sizes ($10^4$, $10^5$, and $10^6$) and three batch sizes (64, 256, and 752), inspecting the recovery of the final solution and the validation loss for each simulation. Fig. 5(i) shows the variation of the optimal recovered solution for the different batch and sample sizes, with columns representing the increasing number of samples and rows representing the increase batch size. Fig. 5(j) shows the validation loss for each of the separate simulations with the line color corresponding to the panel color.

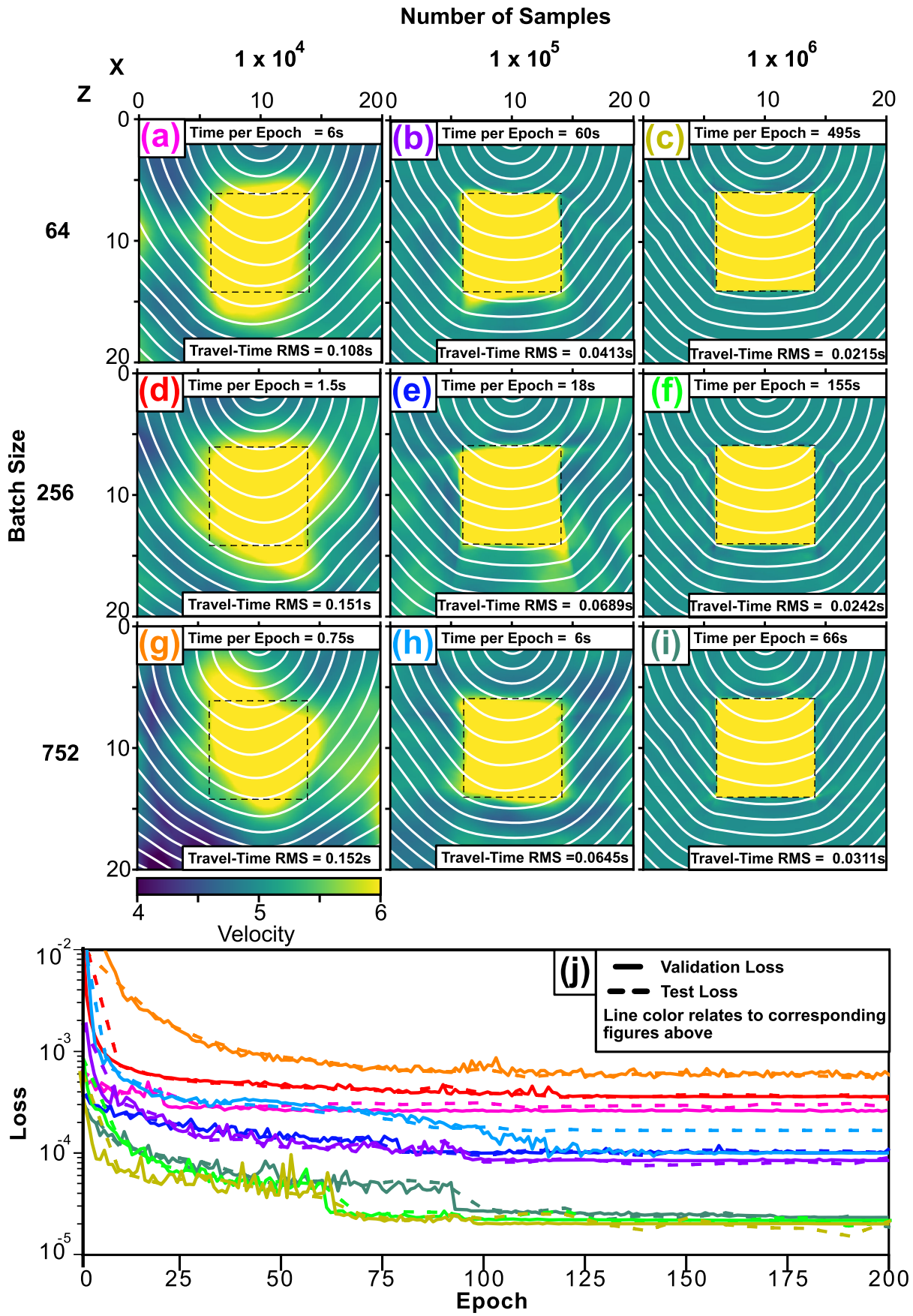



Fig. 5. Training loss effects with changing sample number and batch size. (a)–(i) Different models runs, with a number of samples increasing by columns left to right and batch size increasing by rows from top to bottom. (j) Validation loss, along with test loss for each of the separate model runs with color matching the panel labels.

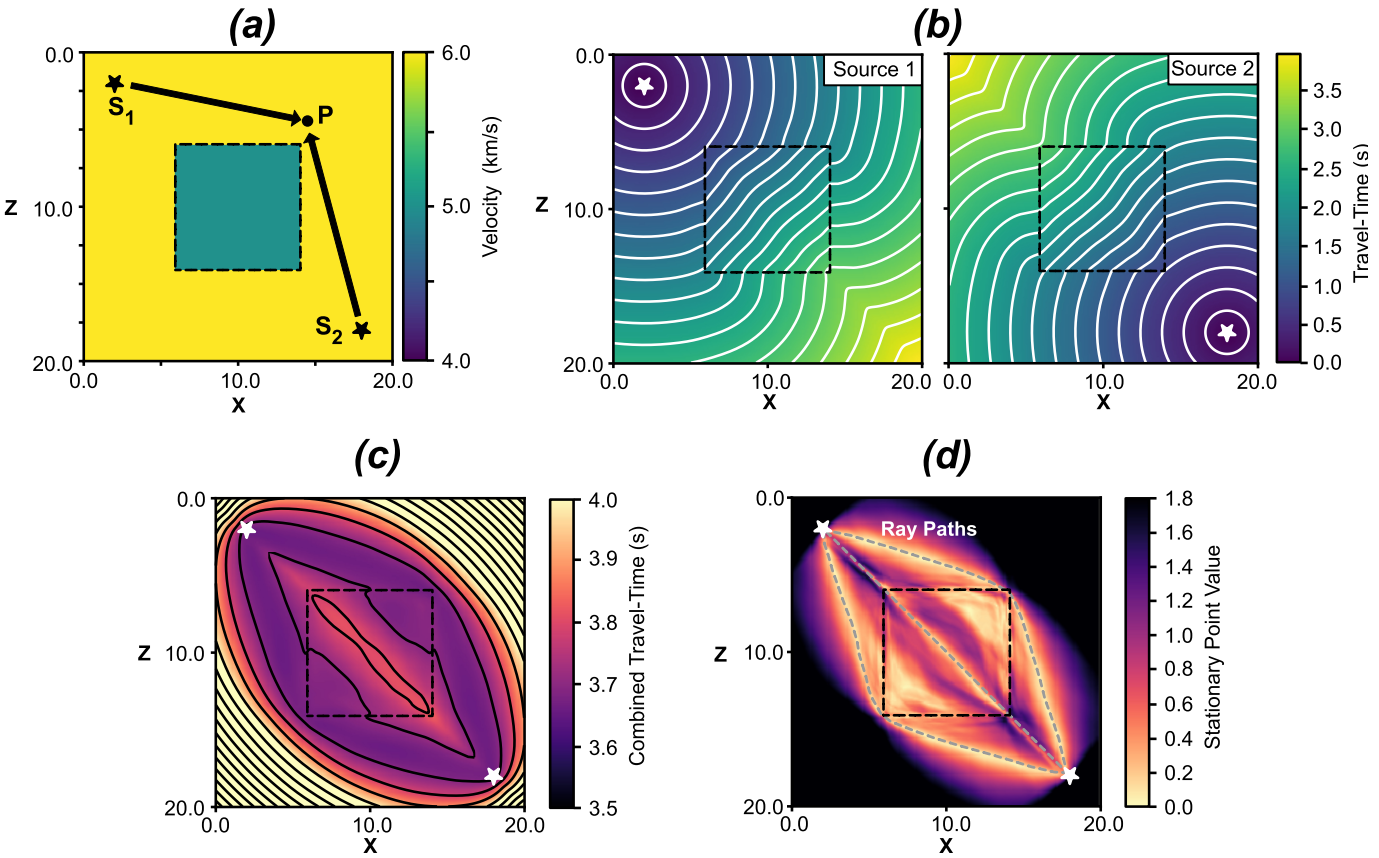



Fig. 6. Seismic ray multipathing procedure applied to an adapted block model velocity experiment with low-velocity block and high-velocity background. (a) Imposed velocity structure and schematic of the travel times between source locations $S_1$ and $S_2$, to some point $P$. (b) Travel times from each of the receiver locations to a grid of receiver locations. (c) Combined travel-time field from the source locations to each of the receiver locations, $\mathrm{TT}_P$. (d) Gradient of the combined travel-time field at each of the receiver locations, $(\partial \mathrm{TT}_P/\partial P)$. Stationary points are represented where this value is zero, with these points representing possible secondary arrival ray paths.

From Fig. 5, it is clear that the number of training samples has a profound influence on the network performance. The data set with $10^6$ samples achieves a loss that is about an order of magnitude lower than the data set with $10^4$ samples. In addition, through inspection of the recovered velocity model, we can see that the low sample size is unable to reconcile the complex velocity structure of the sharp velocity contrast of the block model. Therefore, it is crucial that an adequate number of training samples number should be used when constructing the data set. Future work could investigate dynamic sampling of the velocity model space to reconcile regions of greatest misfit.

While the batch size is seen to influence the training results early on, the final best loss value is seen to be insensitive to this hyperparameter (see Fig. 5). This is important as larger batch sizes are much more computationally efficient.

## VII. Future Applications

In this section, we discuss the application of EikoNet to a series of travel-time required problems, outlining the advantage of EikoNet over conventional finite-difference methods and how these procedures would be implemented.

### A. Earthquake Location

In an earthquake location theory, a series of seismic instruments is used to record the arrival time of the incoming seismic wave. These instrument arrival times are then inverted using a velocity model to determine an earthquake location and location uncertainty. In recent years, advances in seismological instrumentation have allowed for the incorporation of distributed acoustic sensing (DAS), using optical fibers as a series of nondiscretized station locations [24]. Current travel-time finite-difference techniques are not tractable for handling the tens of thousands of virtual receivers that DAS arrays provide. This would require a large computational cost and disk storage space. In comparison, our machine learning technique scales independently of the number of receivers, has a compact storage footprint, and can rapidly evaluate forward predictions from the network.

### B. Ray Multipathing

The Eikonal formulation represents the first arrival between source and receiver locations. However, in complex velocity models, sharp gradients in the velocity structure can produce a multipathing effect with the energy partitioned between multiple ray paths. Rawlinson *et al.* [21] acted to mitigate this effect by employing FMM to track the evolution of the seismic wavefront for a narrowband of nodes, representing an interface of interest. Once the outgoing wave traverses one of these node locations, an additional simulation is triggered and the combined wavefield used to determine a possible secondary

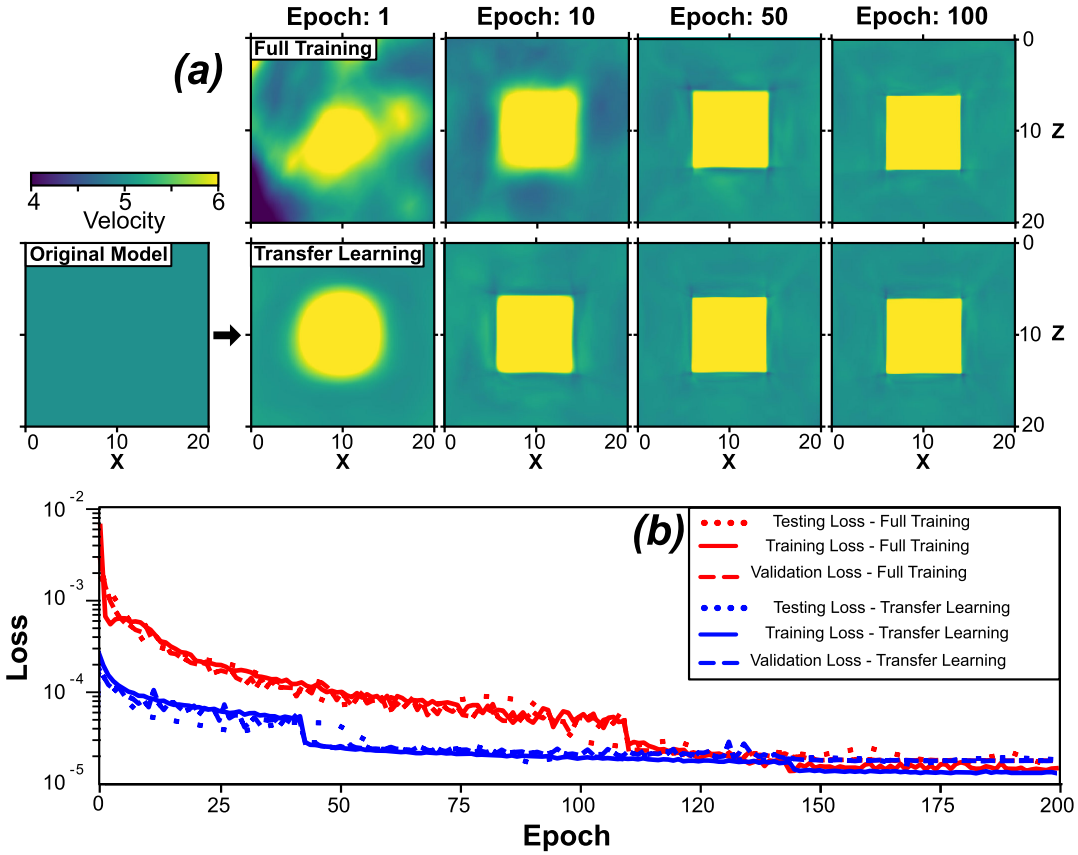



Fig. 7. Effects of transfer learning on the training loss for the Block model velocity experiment. (a) Training snapshots of a full-training procedure and transfer learning from the homogenous model. Rows represent the full-training and transfer learning approaches. Columns represent the snapshot of the velocity model for a series of training epochs. (b) Training, validation, and testing loss for each of the separate simulations.

pathway. The deep learning approach can be adapted to include additional secondary arrivals by determining the travel time from two separate source locations to the same point in space. Fig. 6(a) and (b) shows the travel-time field from the two source locations to a series of points within the block model velocity experiment but now altered with a slow velocity center and faster velocity background. By combining the travel time for the source locations to each of the points, we acquire the combined travel-time field, as shown in Fig. 6(c). By taking the gradient of this combined-travel time across the network relative to the receiver locations [see Fig. 6(d)], we can determine whether the points are stationary [21], having a gradient value close to zero and therefore representing a possible secondary phase arrival pathway. EikoNet is able to quickly formulate travel-time fields and take gradients relative to the receiver locations with each possible multipathing point calculated in 7 s for $1 \times 10^6$ points on a single CPU, making this procedure beneficial for multipathing simulations.

### C. Tomographic Modeling

For tomographic inversions that undergo many iterations successively, new travel-time fields must be computed from scratch for each iteration. Our approach allows for the neural network model from the previous iteration to be used as the starting point for the next training procedure, which could rapidly converge to the new velocity model if the perturbations are relatively small. This would effectively remove ray tracing as a computational burden from this part of the tomography, as nearly all of the compute time would be spent on the very first tomography iteration. We outline the use of a transfer learning technique on the Block model velocity experiment, comparing the training of a model from scratch relative to updating originally trained homogeneous model. Fig. 7(a) shows the comparison of the returned models for a series of training snapshots. Fig. 7(b) shows that the transfer learning approach is $\sim$3–4$\times$ faster than training a full model from scratch.

The computation cost in the training procedure is in learning the complexities of the velocity model space. If the velocity model is unknown but the user has some prior knowledge of possible arrival time differences, then this approach could be updated to do some form of tomographic inversion. This procedure instead would learn the velocity model to fit some known travel-time values. We perceive that this addition can be made in the loss function term itself, where an additional loss term can be used to update the velocity model to try and mitigate known observations. Prior finite-difference methods would have difficulty with this procedure as the travel-time field would have to be recalculated for each update to the velocity model.

## VIII. Conclusion and Discussion

We have demonstrated that deep neural networks can solve the Eikonal equation to learn the travel-time field in heterogeneous 3-D velocity structures. The method has been

shown to produce solutions that are consistent with those of the FMM.

Finite-difference approaches require computing the travel-time field separately for every source location of interest, without any ability to pass along knowledge about the wavefield or velocity structure between simulations. The computation cost for the finite-difference approach therefore increases based on the number of source locations and receiver nodes, with the storage of these travel-time tables increasing drastically with grid size. The deep learning approach instead is able to learn from and generalize the knowledge acquired between multiple source locations, as much of the structure of the problem is highly similar. For example, if two sources are placed very close to each other, the travel-time field will be generally quite similar between them, and the information learned from solving the equation for the first source can be used to more rapidly learn the solution for the second source. This is not the case for fast-marching methods and only holds for small distances in fast-sweeping methods. Implicitly, this means that the neural network is learning the velocity model. The disk storage size required is equal to the size of the neural network (∼90 MB for ten residual layers) as only the weights of the network have to be retained. For more complex velocity structures, the neural network size may require a larger model, with the disk storage scaling by a linear function of the network size. However, for finite-difference approaches, the storage size scales with the product of the number of source and nodal points; therefore, for a complex velocity structure, the model would require a fine-resolution grid spacing or dynamic meshing, making this method intractable with large lookup table storage sizes. One of the distinguishing features of EikoNet is that the travel-time solutions are valid for any two points within the 3-D continuous domain. This means that it is never necessary to store a travel-time grid and interpolate it to achieve the desired result away from the grid nodes. Here, EikoNet automatically learns an optimized interpolation scheme during the training process, drawing on context from across the entire data set.

A second important aspect is that solutions to the Eikonal equation, as learned by EikoNet, are guaranteed to be differentiable back through the network with respect to the source or receiver locations. This has a variety of important practical applications, such as in locating earthquakes, as the inverse problem can be formulated as one of gradient descent, by analytically calculating the gradients of an objective function with respect to the source locations.

The Eikonal equation is not just used in seismology, but numerous other domains of wave physics such as optics [9], medical imagery [6], and video-game rendering [11]. It is expected that EikoNet would be just as suitable in these fields.

The computational cost of predicting the travel time from a source-to-receiver location is equal to the time required to pass across the network ($4.047 \times 10^{-4}$ s per source–point pair on a single 2.3-GHz Intel Core i5 CPU). The low computation cost of a forward prediction means that the neural network model can be used to significantly increase the computation speed of typically used ray-based procedures. Our approach is massively parallel and very well suited for GPUs (taking 0.424 s for $10^6$ source–point pairs on a single Nvidia Tesla V100).

## ACKNOWLEDGMENT

EikoNet is avaliable at github https://github.com/Ulvetanna/EikoNet. The authors would like to thank Jack Muir for interesting discussions about finite-difference methods and limitations.

## REFERENCES

[1] L. Bar and N. Sochen, "Unsupervised deep learning algorithm for PDE-based forward and inverse problems," 2019, *arXiv:1904.05417*. [Online]. Available: http://arxiv.org/abs/1904.05417

[2] A. Capozzoli, C. Curcio, A. Liseno, and S. Savarese, "A comparison fast marching, fast sweeping fast iterative methods for solution eikonal equation," in *Proc. 21st Telecommun. Forum Telfor (TELFOR)*, Nov. 2013, pp. 685–688.

[3] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUs)," 2015, *arXiv:1511.07289*. [Online]. Available: http://arxiv.org/abs/1511.07289

[4] S. Das, "Three-dimensional spontaneous rupture propagation and implications for the earthquake source mechanism," *Geophys. J. Int.*, vol. 67, no. 2, pp. 375–393, Nov. 1981.

[5] P. M. R. DeVries, T. B. Thompson, and B. J. Meade, "Enabling large-scale viscoelastic calculations via neural network acceleration," *Geophys. Res. Lett.*, vol. 44, no. 6, pp. 2662–2669, 2017, doi: 10.1002/2017GL072716.

[6] M. Droske, B. Meyer, M. Rumpf, and C. Schaller, "An adaptive level set method for medical image segmentation," in *Proc. Inf. Process. Med. Imag.* Berlin, Germany: Springer, 2001, pp. 416–422.

[7] X. Guo, W. Li, and F. Iorio, "Convolutional neural Convolutional neural networks for steady flow approximation," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 481–490, doi: 10.1145/2939672.2939738.

[8] E. Hauksson, W. Yang, and P. M. Shearer, "Waveform relocated Earthquake catalog for Southern California (1981 to June 2011)," *Bull. Seismol. Soc. Amer.*, vol. 102, no. 5, pp. 2239–2244, 2012, doi: 10.1785/0120120010.

[9] J. A. Hoffnagle and D. L. Shealy, "Refracting the k-function: Stavroudis's solution to the eikonal equation for multielement optical systems," *J. Opt. Soc. Amer. A, Opt. Image Sci.*, vol. 28, no. 6, pp. 1312–1321, 2011.

[10] F. Huang, J. T. Ash, J. Langford, and R. E. Schapire, "Learning deep ResNet blocks sequentially using boosting theory," 2017, *arXiv:1706.04964*. [Online]. Available: https://arxiv.org/abs/1706.04964

[11] I. Ihrke, G. Ziegler, A. Tevs, C. Theobalt, M. Magnor, and H. P. Seidel, "Eikonal rendering: Efficient light transport in refractive objects," *ACM Trans. Graph.*, vol. 26, no. 3, p. 59-es, 2007.

[12] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980

[13] Z. Li *et al.*, "Neural operator: Graph kernel network for partial differential equations," 2020, *arXiv:2003.03485*. [Online]. Available: http://arxiv.org/abs/2003.03485

[14] G. S. Martin, R. Wiley, and K. J. Marfurt, "Marmousi2: An elastic upgrade for Marmousi," *Lead. Edge*, vol. 25, no. 2, pp. 156–166, 2006, doi: 10.1190/1.2172306.

[15] B. Moseley, A. Markman, and T. Nissen-Meyer, "Fast approximate simulation of seismic waves with deep learning," 2018, *arXiv:1807.06873*. [Online]. Available: https://arxiv.org/abs/1807.06873

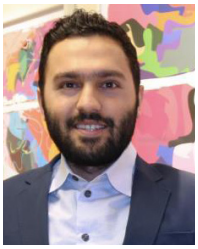
[16] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Int. Conf. Mach. Learn.*, 2010, pp. 807–814.

[17] M. M. Noack and S. Clark, "Acoustic wave and eikonal equations in a transformed metric space for various types of anisotropy," *Heliyon*, vol. 3, no. 3, Mar. 2017, Art. no. e00260.

[18] P. Podvin and I. Lecomte, "Finite difference computation of traveltimes in very contrasted velocity models: A massively parallel approach and its associated tools," *Geophys. J. Int.*, vol. 105, no. 1, pp. 271–284, Apr. 1991.

[19] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *J. Comput. Phys.*, vol. 378, pp. 686–707, Feb. 2019.
[20] N. Rawlinson and M. Sambridge, "Wave front evolution in strongly heterogeneous layered media using the fast marching method," *Geophys. J. Int.*, vol. 156, no. 3, pp. 631–647, 2004.
[21] N. Rawlinson, M. Sambridge, and J. Hauser, "Multipathing, reciprocal traveltime fields and raylets," *Geophys. J. Int.*, vol. 181, no. 2, pp. 1077–1092, 2010.
[22] S. H. Rudy, S. L. Burton, J. L. Proctor, and J. N. Kutz, "Data-driven discovery of partial differential equations," *Sci. Adv.*, vol. 3, no. 4, 2017, Art. no. e1602614.
[23] J. A. Sethian, "A fast marching level set method for monotonically advancing fronts," *Proc. Nat. Acad. Sci. USA*, vol. 93, no. 4, pp. 1591–1595, Feb. 1996.
[24] E. F. Williams *et al.*, "Distributed sensing of microseisms and teleseisms with submarine dark fibres," *Nature Commun.*, vol. 10, p. 5778, 2019.
[25] E. Treister and E. Halder, "A fast marching algorithm for the factored eikonal equation," *J. Comput. Phys.*, vol. 324, pp. 210–225, Nov. 2016.
[26] R. Versteeg, "The Marmousi experience: Velocity model determination on a synthetic complex data set," *Lead. Edge*, vol. 13, no. 9, pp. 927–936, 1994.
[27] J. E. Vidale, "Finite-difference calculation of traveltimes in three dimensions," *Geophysics*, vol. 55, no. 5, pp. 521–526, 1990.
[28] E. Weinan and B. Yu, "The deep ritz method: A deep learning-based numerical algorithm for solving variational problems," *Commun. Math. Statist.*, vol. 6, no. 1, pp. 1–12, 2018.
[29] H. Zhang, C. Thurber, and P. Bedrosian, "Joint inversion for Vp, Vs, and Vp/Vs at SAFOD, Parkfield, California," *Geochem., Geophys., Geosyst.*, vol. 10, no. 11, 2009, Art. no. Q11002, doi: 10.1029/2009GC002709.
[30] H. Zhao, "A fast sweeping method for eikonal equations," *Math. Comput.*, vol. 74, no. 250, pp. 603–627, 2004.
[31] Y. Zhu and N. Zabaras, "Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification," *J. Comput. Phys.*, vol. 366, pp. 415–447, Aug. 2018.

**Jonathan D. Smith** is a Post-Doctoral Researcher with the California Institute of Technology, Pasadena, CA, USA, where he is investigating dynamics of microseismic earthquakes. His research entails leveraging machine learning applications for phase detection, phase association, and location, in order to improve the resolution of microseismic earthquake catalogs. He is interested in the dynamics and forcing mechanics of earthquake sequences, with applications to a tectonic setting and induced seismicity.



**Kamyar Azizzadenesheli** is an Assistant Professor with the Computer Science Department, Purdue University, West Lafayette, IN, USA, where he uses machine learning, from theory to practice, with the main focus on reinforcement learning. He works on deep learning, control theory, spectral method, optimization, online learning, active learning, safety, adversarial attacks, and generative models through both learning theory and core scientific lenses.

**Zachary E. Ross** is an Assistant Professor of geophysics with the California Institute of Technology, Pasadena, CA, USA, where he uses machine learning and signal processing techniques to better understand earthquakes and fault zones. He is interested in the dynamics of seismicity, earthquake source properties, and fault zone imaging.