EikoNet: Solving the Eikonal equation with Deep Neural Networks

Jonathan D. Smith *, Kamyar Azizzadenesheli †, Zachary E. Ross *

Abstract

The recent deep learning revolution has created an enormous opportunity for accelerating compute capabilities in the context of physics-based simulations. Here, we propose EikoNet, a deep learning approach to solving the Eikonal equation, which characterizes the first-arrival-time field in heterogeneous 3D velocity structures. Our grid-free approach allows for rapid determination of the travel time between any two points within a continuous 3D 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 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 multi-pathing, and tomographic modeling, as well as to other fields beyond seismology where ray tracing is essential.

1 Introduction

Three-dimensional ray tracing is a fundamental component of modern seismology, having direct applications to earthquake hypocenter inversions [6], seismic tomography [24], and earthquake source properties [2]. These derived products further form the basis for many downstream seismological applications. The Eikonal Equation is a well-known nonlinear partial differential Equation that characterizes the first-arrival-time field for a given source location in a 3D medium [13]. The Eikonal formulation can be solved with several finite difference algorithms [22],[14],[16], with varying computational demands and stabilities to the solutions.

Recent advances in deep learning have shown to be extremely promising in the context of physics-based simulations [5, 25]. 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 [**?**], forecast the next time step of a wavefield conditional on its history [**?**], and accelerate viscoelastic simulations [3]. However, these techniques rely on inputs from pre-computed physics based models, which could themselves contain modeling-based artifacts and input bias. Instead, we wish to learn the

*Seismological Laboratory, California Institute of Technology, Pasadena, CA, USA

†Department of Computing and Mathematical Sciences, California Institute of Technology, Pasadena, CA, USA

underlying physics by incorporating the formulations into the neural network architecture and loss function. These physics-based procedures [18, 23, 15, 1, 11] 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 [18, 15] are mesh independent, giving a continuous function output related to the inputs. These techniques have been used to learn the parametrization for formulations of the Burgers, Schrodinger, and Navier-Stokes Equations, with comparisons made to the numerical derivatives [18].

In this paper, we propose EikoNet, an approach to solving the factored Eikonal Equation directly with deep neural networks. EikoNet can learn the travel-time between any two points in a truly continuous 3D 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 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. Additionally, EikoNet has several novel advantages that are not currently offered with conventional finite differencing schemes.

2 Eikonal Formulation

The Eikonal Equation is a nonlinear first-order partial differential Equation representing a high-frequency approximation to the propagation of waves in heterogeneous media [13]. The formulation takes the general form;

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

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 ∇_r the gradient at the reciever 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 Equation 1 have a strong singularity at the source location [21], 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 $V = 1$ [21]. 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 τ the deviation of the travel-time field from a model travel-time with homogeneous unity velocity. Substituting the formulation of Equation 2 into Equation 1 and expanding using the chain rule, then the velocity can be represented by;

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

The partial differential terms in Equation 3 are typically solved using a finite-difference approach and will be discussed in Section 4

3 Methods

3.1 Network architecture and training

Our approach to solving the Eikonal Equation trains a deep neural network, f_θ, to predict the deviation of the travel-time field, τ, between an input pair of source-receiver coordinates, $\vec{x} = [\vec{x_r}, \vec{x_s}]$. 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 Equation 2 (Figure 1.1.).

If τ was known, a neural network could be trained for a catalog source-receiver pairs. However, τ 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 τ 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_θ is indeed τ; and use this value to compute V from Equation 3, defining this predicted velocity as $\hat{V}$. Here, we exploit the differentiability of deep neural networks to analytically determine the spatial gradient of Equation 4 with respect to $\vec{x_r}$.

Solving the factored Eikonal Equation is therefore reduced to training a neural network with supervision on the velocity model, by iteratively updating the parameters θ to minimize some loss function. Once trained, the Eikonal Equation is no longer needed, as f_θ outputs τ directly (Figure 1.2). In the process, the details of the velocity model will be encoded in the network, requiring the network to be re-trained if the velocity model is to be changed. The model architecture used in this study is a feed-forward network consisting of a series of residual blocks with fully-connected layers [8] followed by nonlinear units (Figure 1.2). We apply ReLu activation function as the nonlinear unit to all hidden layers [12].

The neural network learns the travel time between any source receiver pair and must contain an adequate sampling from across the 3D 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, 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 dataset therefore consists of many $(\vec{x}, y)$ samples, taken from across the 3D volume (Figure 1.3). We discuss how these datasets are constructed in the following section.

The misfit between the predicted $\hat{V}$, as determined from Equation 3, and observed, V, velocities are then minimised 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 5×10^{-5}[10] (Figure 1.4). The batch and dataset size are set to 752 and 10^6 respectively, with their variability discussed further in Section 6.2. The dataset is sepearted into training and validation data, with the validation

dataset at 10% of the total size. An addition test dataset is created representing 10^4 source reciever pairs which are blind to the user prior to loss calculation.

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 Figure 1.5.

3.2 Building a training dataset

Our approach builds a training dataset by randomly sampling source-receiver points from the continuous 3D 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.

3.2.1 Random Variable Length

The dataset is composed of a series of source locations selected randomly across the model space. Once selected , the receiver locations are selected 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.

3.2.2 Random Locations

The dataset 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 non-uniform distributions of distances between source-receiver points as expressed by the Bertrand Paradox.

3.2.3 Weighted Sampling

In complex velocity contrasting models the training procedure could quickly learn areas of simple velocity variations, we act to improve the training speed of this procedure by allowing dynamic resampling of the training source-receiver pairs for values of greatest misfit. For the first epoch of training we minimise the misfit between the actual and predicted velocity estimates using a L2-norm, but also determine a 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 dataset. 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.

3.3 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 3D medium and use Equation 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 (Figure 1.5).

4 Baselines

The Fast Marching Method (FMM) is a grid based numerical scheme to determine the solution to the Eikonal Equation [19]. The FMM uses a finite difference approach to track the evolution of the minimum travel-time using a advancing interface scheme [16]. This solution returns the minimum travel-time from any source location to a receiver node in a user-defined mesh or regular grid. This formulation relies on a solution to the Eikonal formulation without singularities and therefore can fail if sharp velocities or under sampling. Typically to mitigate these effects a smoothed velocity model is supplied with adequate receiver node sampling to reconcile any velocity contrasts. As this procedure computes the travel-time from a source location to a series of receiver locations this procedure can be repeated for a number of source locations, with the computational cost and memory/storage requirements increasing with the number of source and receiver locations.

Throughout this study we utilize the python fast marching method toolkit scikit-fmm (`https://pythonhosted.org/scikit-fmm/`) to formulate the travel-time from a source location on a receiver geometry at $0.1km$ grid spacing in the X, Y, Z dimensions.

5 Velocity Model Experiments

Outlined are a series of experiments designed to demonstrate the versatility of our approach for learning travel time fields in complex 3D 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 dataset with 10^6 randomly located source-receiver pairs. The effects of dynamic sampling and weighting are discussed further in Section 6.

5.1 Homogeneous Velocity

The first model we consider is a homogeneous 3D velocity structure with a value of $5km/s$ (Figure 2.1). For demonstration purposes, a source is placed at $[X, Y, Z] = [10, 10, 1]$, with both $X - Z$ and $X - Y$ slices of the travel time field shown. At each receiver point, we plot the learned velocity using Equation 1, and use this to determine the percent error. For comparison, we also show the FMM solution calculated on a grid of receivers with $0.1km$ grid spacing in the X,Y and Z dimensions. The training loss goes to zero after 110 training epochs, showing that the network has learned the travel time field perfectly.

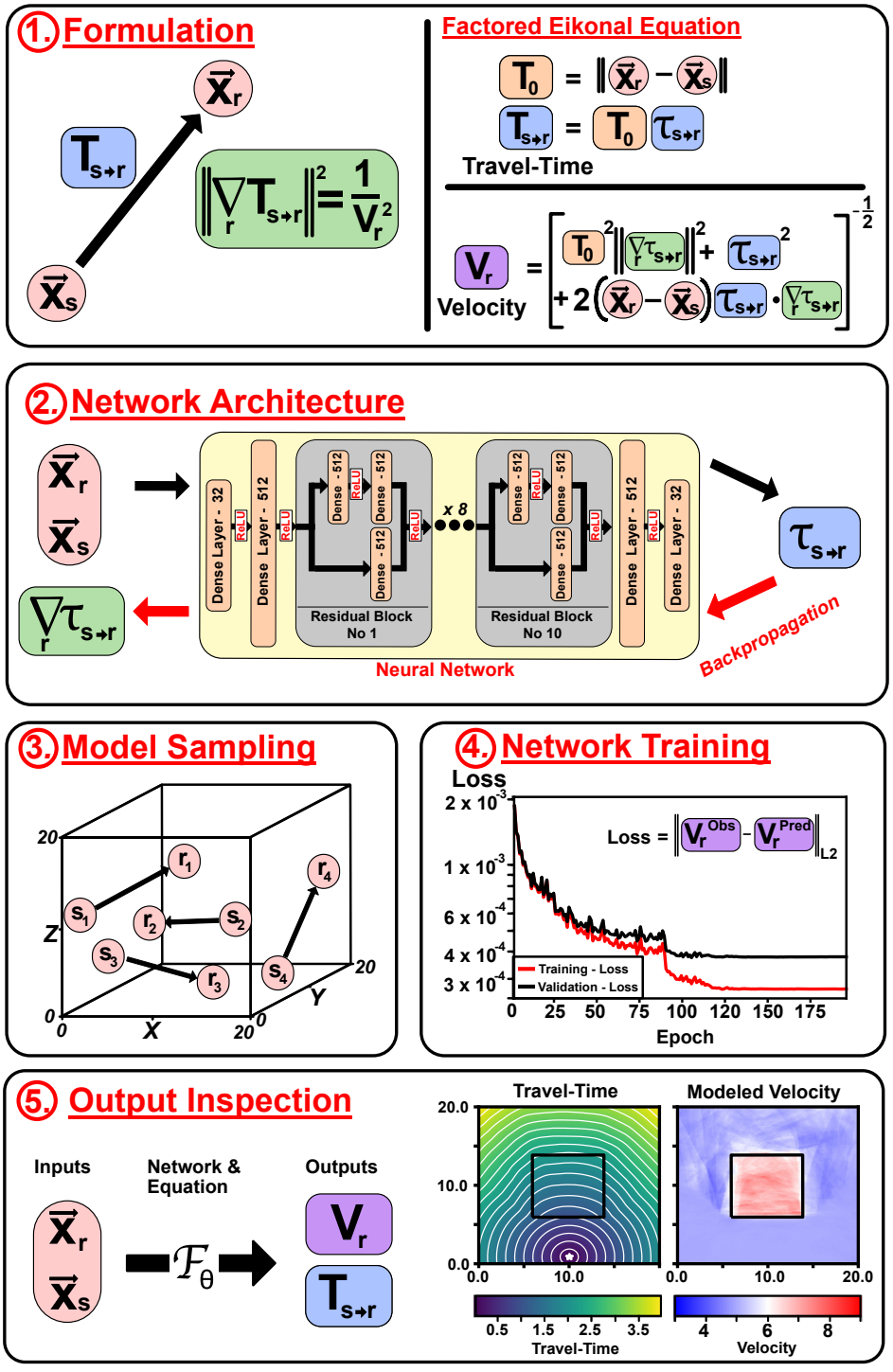

Figure 1: Overview of processing workflow. (1) Summary of Eikonal and factored Eikonal Equations for $T_{s \to r}$ and V_r. (2) Neural network architecture composed of fully-connected layers and residual blocks. Each residual block is composed of 3 fully-connected layers with 512 neurons. ReLu activations are applied on all hidden layers. (3) Sampling of source-receiver pairs across the 3D volume to build the training dataset. (4) Network training through the minimization of loss function relating predicted and observed velocity values. (5) Inspection of neural network outputs by passing user defined source receiver pairs.

5.2 Graded Velocity

Next, we consider a velocity model that increases linearly with depth. The model increases from $3km/s$ at the surface, to $7km/s$ at $20km$ depth. Comparison with the conventional finite-difference scheme shows good agreement, with a maximum difference between imposed and recovered velocity model of $0.18km/s$.

5.3 Block Model

The third test conducted is a 3D model containing a central cube embedded within a homogeneous background (Figure 2.3). The cube has a constant velocity of $7km/s$, while the background velocity is $5km/s$. Here, the neural network approach has excellent misfit between the actual and learned velocity, with only some 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, demonstrating that the neural network approach is able to reconcile even difficult cases with sharp velocity changes of 20% of the mean value.

5.4 Checkerboard Velocity

The final velocity model experiment contains a checkerboard, which we use to demonstrate that the proposed algorithm is able to reconcile positive and negative velocity anomalies varying spatially within the domain. This velocity model is implemented using the following Equation:

$$Vp = \frac{1}{4}\left[\sin\left(\frac{5\pi}{2}X\right) + \sin\left(\frac{5\pi}{2}Y\right) + \sin\left(\frac{5\pi}{2}Z\right)\right] + 5, \tag{9}$$

where Vp is the velocity at a point. 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 maximum velocity difference of $0.48km/s$.

6 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 the previous section and train separate models using each of the three sampling techniques described in Section 3.2. We also separately test how the total number of training samples and batch size affect the performance.

6.1 Sampling Schemes

Figure 3 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. We expect that the weighted random sampling will be of increasing importance in very complex 3D models and recommend it for selection of source-receiver pairs.

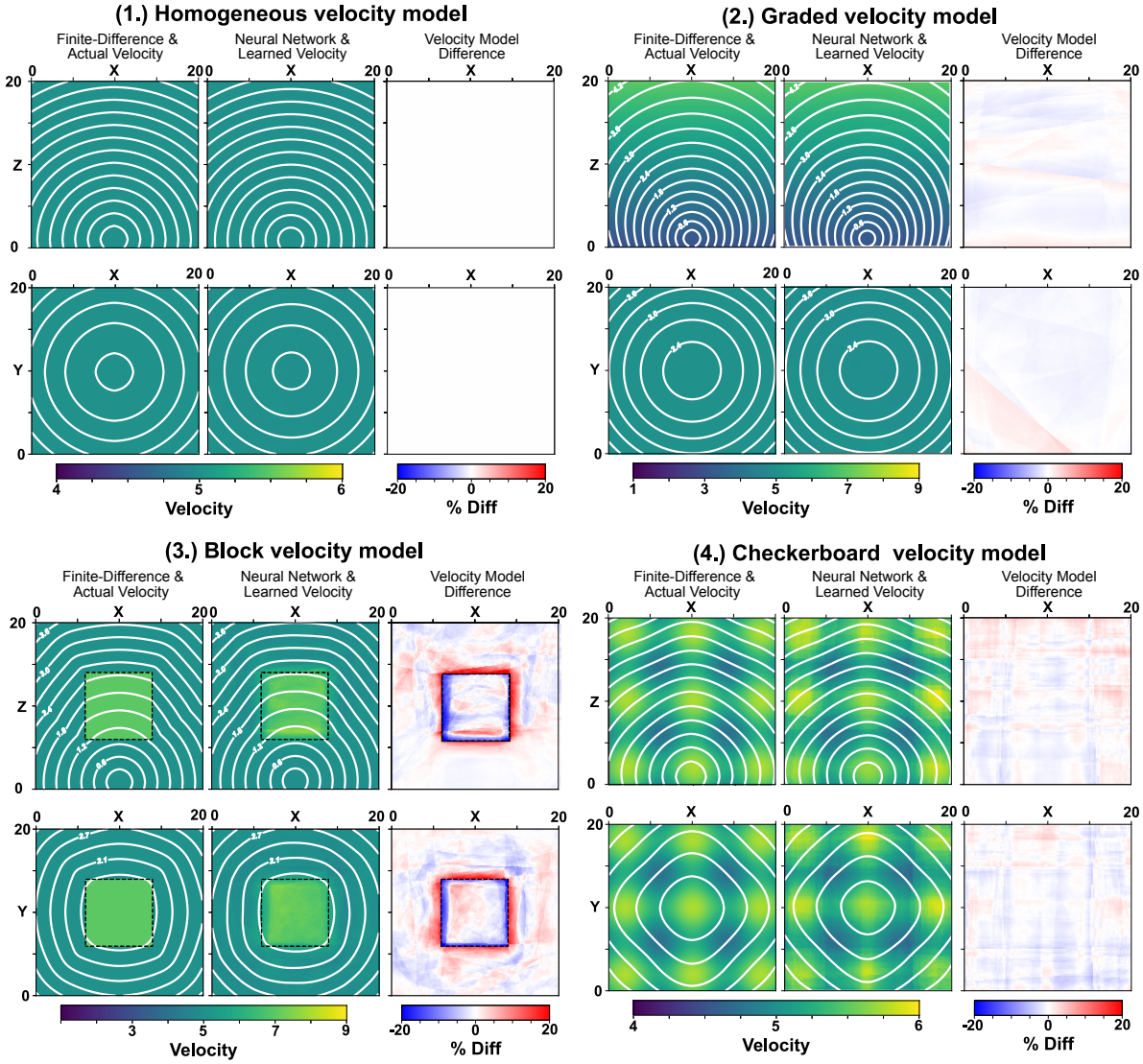

Figure 2: Velocity model experiments with comparison to finite-difference and imposed velocity models. *Left* panels represent the X-Z and X-Y slice from the imposed velocity model, overlayed by the finite-difference expected travel-time. *Middle* panels represent the X-Z and X-Y slice from the neural network recovered velocity model and neural-network travel-time. *Right* panels represent X-Z and X-Y slice velocity models differences between the imposed and recovered velocity model

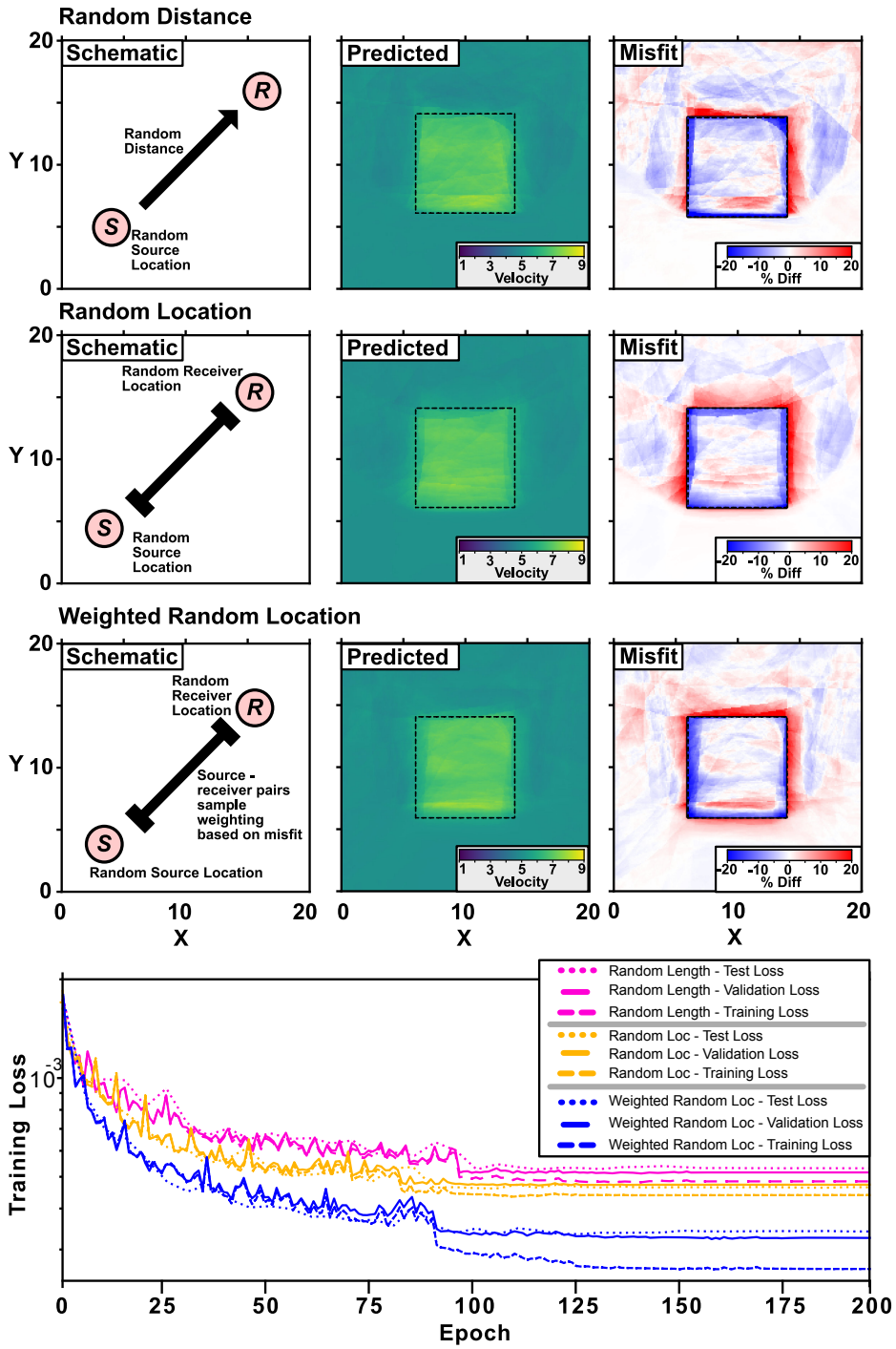

Figure 3: Sampling schemes and their influence on the network performance. *Left Column* represents a schematic of the different type of sampling, *Middle Column* represents the learned velocity model for each simulation and *Right Column* represents the misfit between predicted and imposed velocity model. *Bottom Panel* represents the training, validation, and testing loss for each of the simulations.

6.2 Size of training dataset

We investigate how the number of training samples and batch size affect the network performance. We re-run the simulation with three dataset 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. Figure 4i 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 the increase batch size. Figure 4j shows the validation loss for each of the separate simulations with the line colour corresponding to the panel color.

From Figure 4, it is clear that the number of training samples has a profound influence on the network performance. The dataset with 10^6 samples achieves a loss that is about an order of magnitude lower than the dataset 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 be used when constructing the dataset. 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 (Figure 4). This is important as larger batch sizes are much more computationally efficient.

7 Discussion and Conclusion

We have demonstrated that deep neural networks can solve the Eikonal Equation to learn the travel-time field in heterogeneous 3D velocity structures. This procedure has demonstrated consistent calculations with prior finite-difference approaches for travel-time formulation.

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 the finite difference algorithms, which have no notion of context. 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 ($\sim 90MB$) as only the weights of the network have to be retained. Figure 5 demonstrates a schematic representing the disk storage required for the finite-difference and machine learning approach.

One of the distinguishing features of EikoNet is that the travel time solutions are valid for any two points within the 3D continuous domain. This means 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 dataset.

A second important aspect is that solutions to the Eikonal Equation, as learned by EikoNet,

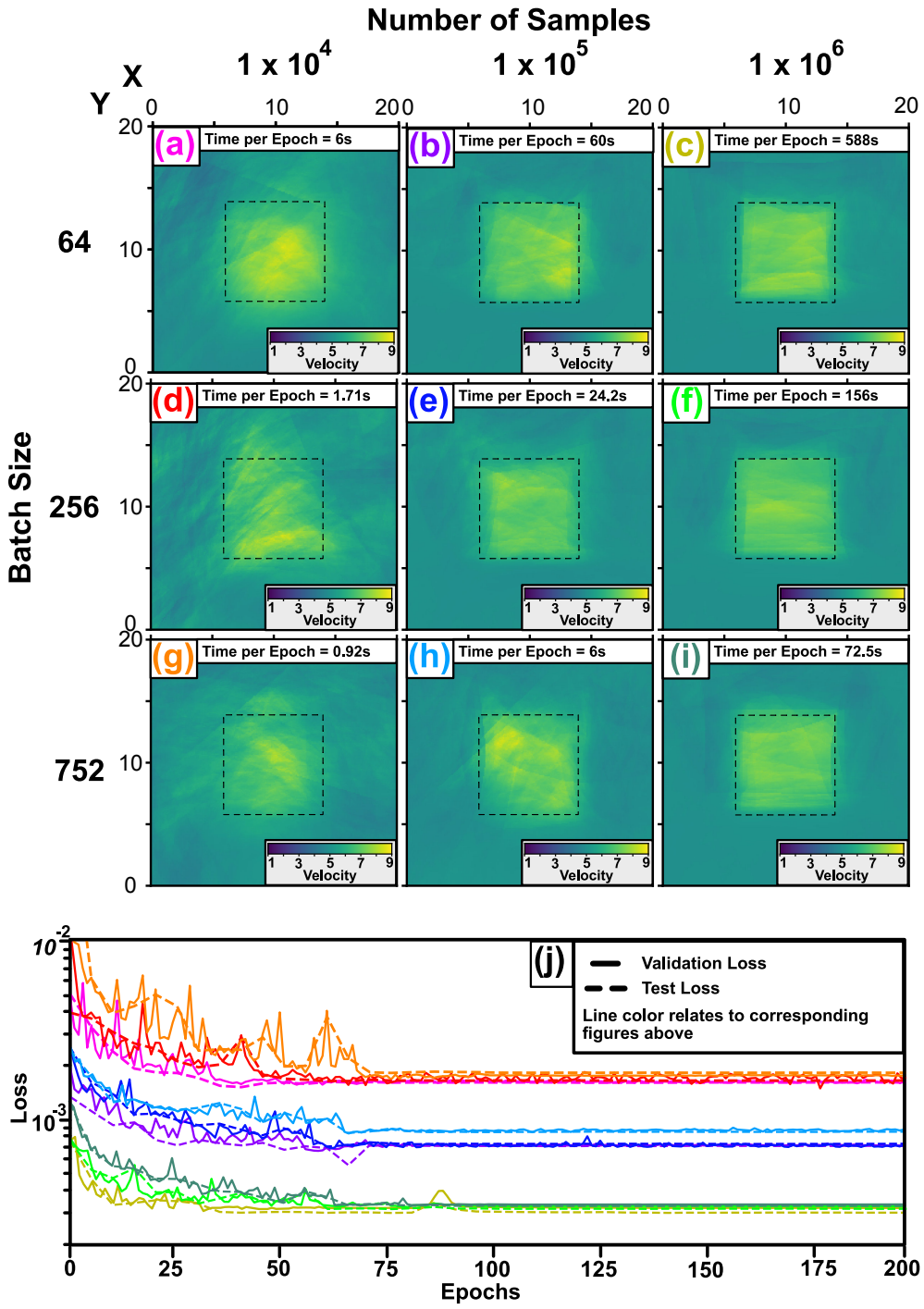

Figure 4: Training loss effects with changing sample number and batch size. Panels (a) - (i) represent the different models runs, with number of samples increasing by columns left to right and batch-size increasing by rows from top to bottom. Panel (j) represents the validation loss, along with test loss for each of the separate model runs with color matching the panel labels.

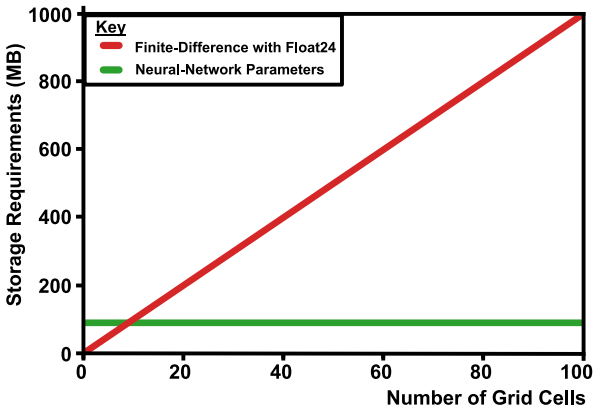

Figure 5: Schematic of storage requirements difference between a finite difference scheme and the neural-network parameters.

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 [7], medical imagery [4] and video-game rendering [9]. 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. 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. Below, we outline several areas that could benefit from this deep learning approach.

7.1 Earthquake Location

In earthquake location theory a series of seismic instruments are 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 Accoustic Sensing (DAS), using optical fibres as a series of non-discretized station locations [20]. 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 has a fixed disk storage space and the computation cost only that of the forward prediction from the network.

7.2 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 multi-pathing

effect with the energy partitioned between multiple ray paths.[17] acted to mitigate this effect by employing FMM to track the evolution of the seismic wavefront for a narrow band of nodes, representing a 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 pathway. The deep learning approach can be adapted to include additional secondary arrivals by integrating the travel-time along thousands of arbitrary pathways at a small additional computational cost. Each pathway can be inspected to determine its plausibility of being a secondary arrival, possible arrival time and corresponding amplitudes.

7.3 Tomographic Modeling

For tomographic inversions which 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.

The computation cost in the training procedure is in learning the complexities of the velocity model space. If the velocity model is an 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.

References

[1] L. Bar, and N. Sochen, 2019, *Unsupervised deep learning algorithm for PDE-based forward and inverse problems*, arXiv preprint arXiv:1904.05417.

[2] S. Das, 2012, *Three-dimensional spontaneous rupture propagation and implications for the earthquake source mechanism*, Geophysical Journal International, Volume 67, Issue 2, November 1981, Pages 375–393.

[3] P.M.R. DeVries, T. Thompson, T. Ben, and B.J. Meade, 2017, *Enabling large-scale viscoelastic calculations via neural network acceleration*, Geophysical Research Letters, 44, 2662 – 2669, 10.1002/2017GL072716.

[4] M. Droske, B. Meyer, M. Rumpf, and C.Schaller, 2001, *An Adaptive Level Set Method for Medical Image Segmentation*, Information Processing in Medical Imaging, Springer Berlin Heidelberg, 416–422.

[5] X. Guo, W. Li, and F. Iorio, 2016, *Convolutional Neural Networks for Steady Flow Approximation*,P roceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 481–490, 10.1145/2939672.2939738

[6] E. Hauksson, W. Yang, and P.M. Shearer, 2012, *Waveform Relocated Earthquake Catalog for Southern California (1981 to 2011)*, Bull. Seismol. Soc. Am., Vol. 2012, doi: 10.1785/0120120010.

[7] J.A. Hoffnagle, and D.L. Shealy, 2011, *Refracting the k-function: Stavroudis's solution to the eikonal Equation for multielement optical systems*, JOSA, 28, 1312-1321.

[8] F. Huang, J.T. Ash, J. Langford, and R.E. Schapire, 2017, *Learning deep resnet blocks sequentially using boosting theory*. CoRR abs/1706.04964.

[9] I. Ihrke, G. Ziegler, A. Tevs, C. Theobalt, M. Magnor, and H.P. Seidel, 2007, *Eikonal rendering: Efficient light transport in refractive objects*, ACM Transactions on Graphics (TOG), 26(3), 59-es.

[10] D. Kingma, and J. Ba, 2015, *Adam: A method for stochastic optimization*, International Conference on Learning Representations (ICLR).

[11] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, 2020, *Neural Operator: Graph Kernel Network for Partial Differential Equations*, arXiv preprint arXiv:2003.03485.

bMoseley, Benjamin, Markman, Andrew, and Nissen-Meyer, Tarje, 2018, *Fast approximate simulation of seismic waves with deep learning*, ArXiv, 1807.06873.

[12] V. Nair, and G.E. Hinton, 2010, *Rectified linear units improve restricted boltzmann machines*, Proceedings of the 27th International Conference on International Conference on Machine Learning (ICML'10), 807–814.

[13] M.M. Noack and S. Clark, 2017, *Acoustic wave and eikonal Equations in a transformed metric space for various types of anisotropy* Heliyon, 3.

[14] P. Podvin, and I. Lecomte, 1991. *Finite difference computation of traveltimes in very contrasted velocity models: a massively parallel approach and its associated tools*, Geophys. J. Int., 105, 271–284.

[15] M. Raissi, P. Perdikaris and G. E. Karniadakis, 2019, *physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential Equations*, Journal of Computational physics, 378, 686-707.

[16] N. Rawlinson, and M. Sambridge, 2004, *Wave front evolution in strongly heterogeneous layered media using the fast marching method*, Geophysical Journal International, Volume 156, Issue 3, Pages 631–647

[17] N. Rawlinson, M. Sambridge and J. Hauser, 2010, *Multipathing, reciprocal traveltime fields and raylets*, Geophysical Journal International, 181, 2, 1077–1092.

[18] S.H. Rudy, S. L. Burton, J. L. Proctor and J. N. Kutz, 2017, *Data-driven discovery of partial differential Equations*, Science Advances, 3.

[19] J.A. Sethian, 1996, *A fast marching level set method for monotonically advancing fronts*, Proceedings of the National Academy of Sciences, Feb 1996, 93 (4) 1591-1595.

[20] E. F. Williams, M. R. Fernández-Ruiz, R. Magalhaes, R. Vanthillo, Z. Zhan, M. González-Herráez, and H. F. Martins, Nature Communications, 2019, 10 (1) 1-11.

[21] E. Treister and E. Halder, 2016, *A fast marching algorithm for the factored eikonal Equation*, Journal of Computational physics, 324, 210-225.

[22] J.E. Vidale, 1988, *Finite-difference traveltime calculation*, Seis. Sot. Am., 78, 2062-2076.

[23] E. Weinan, and Y. Bing, 2018, *he deep Ritz method: a deep learning-based numerical algorithm for solving variational problems*, Communications in Mathematics and Statistics, 6, 1–12.

[24] H. Zhang, C. Thurber, and P.A. Bedrosian, 2009, *Joint inversion for Vp, Vs, and Vp/Vs at SAFOD, Parkfield, California.*, Geochem. Geophys. Geosyst.,10(11):Q11002, doi:10.1029/2009GC002709.

[25] Y. Zhu, and N. Zabaras, 2018, *Bayesian deep convolutional encoderdecoder networks for surrogate modeling and uncertainty quantification*, Journal of Computational Physics, 366, 415 – 447