GEOPHYSICS®

Traveltime-table compression using artificial neural networks for Kirchhoff-migration processing of microseismic data

Journal:	*Geophysics*
Manuscript ID	GEO-2019-0427.R3
Manuscript Type:	Technical Paper
Keywords:	microseismic, migration, neural networks, traveltime, interpolation
Area of Expertise:	Seismic Velocities and Traveltimes, Seismic Migration

Traveltime-table compression using artificial neural networks for Kirchhoff-migration processing of microseismic data

Serafim I. Grubas[1 2], **Georgy N. Loginov**[1 2], **Anton A. Duchkov** [1 2]

[1] *Trofimuk Institute of Petroleum Geology and Geophysics SB RAS,*

Koptyuga av., b. 3, Novosibirsk, Russia, 630090

[2]*Novosibirsk State University,*

Pirogova str., b. 1, Novosibirsk, Russia, 630090

E-mail: serafimgrubas@gmail.com, loginovgeorgy@gmail.com,

duchkovaa@ipgg.sbras.ru

(June 2, 2020)

GEO-2019-0427

Running head: **Neural-network compression**

ABSTRACT

Massive computation of seismic traveltimes is widely used in seismic processing, e.g. for the Kirchhoff migration of seismic and microseismic data. Implementation of the Kirchhoff migration operators utilizes large pre-computed traveltime tables (for all sources, receivers and densely sampled imaging points). We test the idea of using Artificial Neural Networks for approximating these traveltime tables. The neural network has to be trained for each velocity model, but then the whole traveltime table can be compressed by several orders of magnitude (up to six orders) to the size of less than one megabyte. This makes it convenient to store, share, and use such approximation for processing large data volumes. We discuss some aspects of choosing the neural-network architecture, training procedure, and optimal hyperparameters. On synthetic tests, we demonstrate reasonably accurate approximation of traveltimes by neural networks for various velocity models. Final synthetic test shows that using the neural-network traveltime approximation results in good accuracy of microseismic event localization (within the grid step) in 3D case.

INTRODUCTION

Seismic traveltimes are widely used in seismic processing. In particular, traveltimes are used in the Kirchhoff-type depth migration operators in the reflection seismology (Biondo, 2006) as well as microseismic monitoring (Duncan, 2005; Van Der Baan et al., 2013; Maxwell et al., 2015). For implementing the Kirchhoff migration, one needs to know seismic-wave traveltimes for a dense sampling of sources, receivers, and imaging points. For horizontally layered models, there exist analytic formulas for the traveltime approximation: from the simplest hyperbolic moveout to multi-parameter approximations (Stovas and Fomel, 2018). However, for laterally inhomogeneous models, one usually has to pre-compute a large number of traveltimes and save them to a hard disk for further use during the migration. This becomes an additional serious effort to organize efficient access to this traveltime table during the migration procedure, especially when optimizing it for modern high-performance computing systems (Panetta et al., 2012; Rastogi et al., 2017). It is emphasized that compression of the traveltime table is important for improving the migration performance (Alkhalifah, 2011). It also suggested using the traveltime-table interpolation between sparse samples (Li and Fomel, 2013).

Artificial neural networks (ANN) seem to be a promising tool for approximating the traveltime tables to reduce their size. The advantage of using ANNs is that they effectively implement a mapping approximating a function which is learned based on a given set of input-output value pairs. Modern neural network architectures proved to be useful in solving complicated problems such as image and speech recognition, translation of texts (Goodfellow et al., 2016), and solving differential equations (Raissi et al., 2017). Machine-learning and deep-learning methods are also used in geophysical applications including traveltime

moveout parameterization (Ivanov and Stovas, 2017), seismic traveltime tomography (De Wit et al., 2013; Bianco and Gerstoft, 2018), seismic migration (Vamaraju and Sen, 2019), and velocity model building directly from seismic data (Araya-Polo et al., 2018).

In this paper, we use ANN for approximating the traveltime table employed for the processing of microseismic data using the Kirchhoff migration operator. We start with a brief overview of the theory of surface microseismic data processing using migration operators. Next, describe the proposed approach of approximating large seismic-wave traveltime tables by ANNs. Then we perform several synthetic tests of the proposed ANN traveltime approximation method showing reasonable accuracy and great compression rate. This testing includes synthetic microseismic data processing by the Kirchhoff migration.

THEORY

Microseismic data processing by migration

Here we give a brief overview of surface microseismic data processing. The surface microseismic monitoring data are characterized by a large number of recording channels, but a low signal-to-noise ratio. In this case, for locating microseismic sources one can use either the reverse-time imaging (Artman et al., 2010) or the Kirchhoff-type migration operators (Duncan, 2005). In the latter case, we define a target volume for microseismic-event localization and a scanning grid for building a migration image within this target volume. Then we apply the migration operator, i.e. perform summation for each grid point:

$$\rho(\boldsymbol{\xi}_\alpha, t) = \frac{1}{N_r} \sum_{x_r} u\left(t + \tau(\boldsymbol{\xi}_\alpha, \mathbf{x}_r), \mathbf{x}_r\right), \tag{1}$$

where $\boldsymbol{\xi}_\alpha = (\xi_{x\alpha}, \xi_{y\alpha}, \xi_{z\alpha})$ are the points of the scanning grid, α is a linear index for this grid, $\mathbf{x}_r = (\mathtt{x}_r, \mathtt{y}_r, \mathtt{z}_r)$ is location of the receiver r, N_r is the number of receivers, t is time, $u(t, \mathbf{x}_r)$ is microseismic data, $\tau(\boldsymbol{\xi}_\alpha, \mathbf{x}_r)$ is traveltime of seismic wave from the source $\boldsymbol{\xi}_\alpha$ to the receiver $\mathbf{x}_r$, further in the paper we will consider only $P-$waves.

For short time intervals, we construct an analog of the semblance measure by selecting maximum in time:

$$\mathtt{S}(\boldsymbol{\xi}_\alpha) = \max_t \mathtt{S}(\boldsymbol{\xi}_\alpha, t). \tag{2}$$

Finally, localization of the hypocenter $\widetilde{\boldsymbol{\xi}}$ of the microseismic event can be found as a location of the semblance maximum value:

$$\widetilde{\boldsymbol{\xi}} = \arg\max_\alpha \mathtt{S}(\boldsymbol{\xi}_\alpha). \tag{3}$$

There is the alternative approach to find the hypocenter position and its error estimates in different directions (Anikiev et al., 2014):

$$\begin{aligned}
\widetilde{\xi}_x &= \sum_x x \cdot \sum_{y,z} P(\boldsymbol{\xi}_\alpha), \quad \sigma_x = \sqrt{\sum_x (x - \widetilde{\xi}_x)^2 \cdot \sum_{y,z} P(\boldsymbol{\xi}_\alpha)}, \\
\widetilde{\xi}_y &= \sum_y y \cdot \sum_{x,z} P(\boldsymbol{\xi}_\alpha), \quad \sigma_y = \sqrt{\sum_y (y - \widetilde{\xi}_y)^2 \cdot \sum_{x,z} P(\boldsymbol{\xi}_\alpha)}, \\
\widetilde{\xi}_z &= \sum_z z \cdot \sum_{x,y} P(\boldsymbol{\xi}_\alpha), \quad \sigma_z = \sqrt{\sum_z (z - \widetilde{\xi}_z)^2 \cdot \sum_{x,y} P(\boldsymbol{\xi}_\alpha)},
\end{aligned} \tag{4}$$

where $\widetilde{\boldsymbol{\xi}}$ is the hypocenter location estimate, $(\sigma_x, \sigma_y, \sigma_z)$ are standard deviations in orthogonal directions, and

$$P(\boldsymbol{\xi}_\alpha) = C \exp\left\{ -\left[\mathtt{S}(\boldsymbol{\xi}_\alpha) - \max_\alpha(\mathtt{S}(\boldsymbol{\xi}_\alpha)) \right]^2 / (2\sigma^2) \right\}, \tag{5}$$

where σ is standard deviation of $\mathtt{S}(\boldsymbol{\xi}_\alpha)$, C is a normalizing constant.

Traveltime table

Seismic-wave traveltimes are described by the eikonal equation:

$$|\nabla \tau| = \frac{1}{V}, \tag{6}$$

where $V = V(x, y, z)$ is seismic-wave velocity, $\tau = \tau(x, y, z)$ is seismic-wave traveltime, $\nabla = (\partial/\partial x, \partial/\partial y, \partial/\partial z)$ is the gradient operator.

For computing traveltimes one can use either ray tracing or numerical schemes for the eikonal equation, e.g. Fast Marching Method (FMM) (Sethian, 1996) or Fast Sweeping Method (FSM) (Zhao, 2005). The accuracy of the traveltime computation depends on the grid step size.

For microseismic-monitoring applications, one usually wants to localize hypocenters of microseismic events with the accuracy of about ten meters. Let us assume several thousands of receivers for the surface microseismic survey and a 3D scanning grid about 200-250 points in each dimension in the subsurface. Then we come up with about 10^{10} values for the traveltime table (number scanning grid points $\times$ number of receivers). Such traveltime table is about 200 GB in size and needs to be stored on a hard disk and requires a careful

organization of the input/output procedures during the migration processing.

Artificial neural networks

An artificial neural network is a system of interconnected artificial neurons. Each neuron is a computational unit that performs the simplest mathematical operations on the input data. The units can form connections with each other, forming complex-organized systems. The type of action of a single unit and the type of interaction between units can be chosen appropriately.

Modern ANN usually has three types of layers: input, hidden, output. Each layer has a set of units, and the hidden layers execute the main transformations over data:

$$\begin{cases} h_0 = \mathbf{X}, \\ h_{l+1} = f_{l+1}(h_l, \mathbf{\Theta}_{l+1}), \end{cases} \tag{7}$$

where $\mathbf{X}$ is input data, l is index of the hidden layer, h_l is input to the layer l (hidden state), h_{l+1} is output to the layer l, f_l and $\mathbf{\Theta}_l$ are the activation function and the weights correspondingly.

The union of weights $\mathbf{\Theta}$ for all hidden layers should be determined during the process of ANN training. Usually, there exists a training dataset (known as input-output pairs). Then, ANN training tries to minimize a loss function measuring misfit between the ANN prediction and the true values for the elements from the training dataset. The simplest implementation of the training procedure would be the iterative gradient descent method. We update the weights at each iteration (epoch) according to the equation:

$$\mathbf{\Theta}_{i+1} = \mathbf{\Theta}_i - \alpha \nabla J(\mathbf{\Theta}_i), \tag{8}$$

where i is epoch number, $\mathbf{\Theta}_i$ is ANN weights at epoch i, α is learning rate, $J(\mathbf{\Theta})$ is the loss function, $\nabla J(\mathbf{\Theta})$ is gradient of the loss function.

To speed up the ANN training procedure it is recommended to normalize the input data first:

$$\widetilde{\mathbf{X}} = (\mathbf{X} - \mu)/\sigma, \tag{9}$$

where μ, σ are the mean value and standard deviation for each column of $\mathbf{X}$.

From this overview, one can see that there are many parameters that define the ANN structure and may affect its performance: number of hidden layers, number of units in the hidden layers, the form of the activation function, the form of the loss function, and optimization method used for training.

Approximation of traveltime table by neural networks

A set of points can be approximated via linear regression with some accuracy. So one can build a hypothetical function that describes the given data, but the linearity leads to limitations of choosing the hypothesis. At the same time, ANN uses nonlinear regression, which can approximate more complex functions. Let us consider the problem of approximating the traveltime table. Then, ANN should represent the function $\tau(\boldsymbol{\xi}_\alpha, \mathbf{x}_r)$ describing seismic-wave traveltime from the subsurface grid point $\boldsymbol{\xi}_\alpha$ to the receiver $\mathbf{x}_r$. Then, we suggest the following procedure for approximating such multi-variable traveltime function by neural networks. First, we prepare the training dataset by computing

traveltimes $\tau(\boldsymbol{\xi}_\alpha, \mathbf{x}_r)$ on a coarse grid (any standard method of traveltime computation can be used for this purpose). Second, we train ANN on the training dataset (supervised learning approach).

For 2D case the traveltime depends on three parameters, $\tau(\xi_{x\,\alpha}, \xi_{z\,\alpha}, \mathtt{x}_r)$; for 3D case it depends on five parameters, $\tau(\xi_{x\,\alpha}, \xi_{y\,\alpha}, \xi_{z\,\alpha}, \mathtt{x}_r, \mathtt{y}_r)$. Therefore, the input data for training takes the form:

$$\mathbf{X} = \begin{pmatrix} \mathtt{x}_1 & \xi_{x\,1} & \xi_{z\,1} \\ \vdots & \vdots & \vdots \\ \mathtt{x}_m & \xi_{x\,m} & \xi_{z\,m} \end{pmatrix} \quad \text{in 2D,}$$
$$\mathbf{X} = \begin{pmatrix} \mathtt{x}_1 & \mathtt{y}_1 & \xi_{x\,1} & \xi_{y\,1} & \xi_{z\,1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathtt{x}_m & \mathtt{y}_m & \xi_{x\,m} & \xi_{y\,m} & \xi_{z\,m} \end{pmatrix} \quad \text{in 3D,} \tag{10}$$

where m is the number of training examples (or, number of source-receiver pairs). Before training, the input data are normalized according to equation 9.

Note that ANN training should be repeated for each velocity model. But after training we get a highly compressed representation of the function $\tau(\boldsymbol{\xi}_\alpha, \mathbf{x}_r)$ for any pair of $\boldsymbol{\xi}_\alpha$ and $\mathbf{x}_r$. So that it can be used to compute traveltimes on-the-fly for any dense grids required for the migration operator (equation 1). In this paper, we address the problem of choosing the optimal ANN structure. For this, we want to maintain the accuracy of ANN-based traveltime approximation that is sufficient for the migration processing of microseismic data. At the same time, we want to minimize ANN size to achieve better compression and faster on-the-fly computation during the migration step. In the following section, we show several tests for choosing optimal ANN parameters. We also estimate the accuracy of

the ANN approximation of seismic-wave traveltimes and how it can affect the accuracy of microseismic processing results.

TESTING

In this paper, we specify ANN type – fully connected neural network (each layer propagates information to each unit of only the next layer) because this architecture is supposed to be optimal for the function approximation problems (Cybenko, 1992). For building the training dataset, we computed traveltimes on a coarse scanning grid $\boldsymbol{\xi}_\alpha$. For this, we used the numerical solver of the eikonal equation from (Nikitin et al., 2018), which provides the first-arrival traveltimes.

After training ANN, we get the approximation of the traveltime table for a particular velocity model. We first choose optimal ANN architecture. We perform several tests for a simple velocity model checking the different number of hidden layers, the number of units in the layer, some other hyperparameters. After choosing optimal architecture, we test how the ANN approximation errors affect the hypocenter localization error. Then, we study the influence of velocity model complexity on the ANN approximation accuracy. For this, we used tests with the Marmousi model. Finally, we show compression rates and the results of the migration processing of synthetic microseismic data in 2D and 3D.

Most of the tests contain the following steps: composing the training set by computing the traveltimes from a coarse subsurface scanning grid to the receiver array (numerical eikonal solver); composing the validation set by computing the traveltimes from a fine subsurface scanning grid to the same receiver array (numerical eikonal solver); training ANN on the training set for approximating traveltimes; validating ANN approximation

on the validation set, i.e. check accuracy comparing ANN-predicted and pre-computed traveltimes.

Selection of hyperparameters

Let us discuss the problem of choosing an optimal ANN structure. In all tests, we used the Adam optimization method for ANN training (Kingma and Ba, 2014), the mean-squared-error (MSE) as a loss function and the mean-absolute-error (MAE) as a quality metric on the validation set. For initial tests, we chose a laterally inhomogeneous but comparatively simple 2D velocity model (see Figure 1). For the training set, we computed traveltimes from the coarse subsurface grid (20×20 points with 100-m step); for the validation set, we computed traveltimes from the detailed subsurface grid (201×201 points with 10-m step).

On a series of preliminary tests, we chose several parameters for stabilizing and speeding-up the training procedure. The final choice for the learning rate was 10^{-3} with a decay of this learning rate by 10^{-4} after each epoch (this decay should result in faster convergence). Moreover, the additional reduction of the learning rate by a factor of 1.3 is applied in case when convergence stagnates for three epochs. The batch size of 128 gave the best training convergence in most of the preliminary tests. For the activation function in the hidden layers, we used $ReLU(x) = max(0, x)$ because it yielded the best accuracy in the preliminary tests. The linear function was used as the activation function for the output layer as we address the regression type of problem. Some of these recommendations may be suitable for other problems of approximation, but it is not a generalization. For example, reducing the learning rate often increases accuracy in many tasks, but using the activation ReLU is not necessary. We can only recommend these parameters, at least as best-practice

for traveltime approximation, based on numerous tests for different models.

We also performed tests for choosing two important ANN parameters: the number of hidden layers and the number of units in these hidden layers. Note that both parameters affect the ANN size that we would like to minimize. Accordingly, we seek the smallest ANN that provides reasonable accuracy but retains a reasonable time for training. We show testing results in Table 1. We checked ANNs using one-to-three layers. For each number of hidden layers, we also iterated over different numbers of units in the hidden layers ranging from 50 to 3000. Then for each case, we performed five independent training and validation runs and averaged the values for approximation accuracy (MAE), training time, and ANN model size. These values are shown in Table 1 (note that labels for columns are the same for middle and lower sub-tables). To keep training time reasonably short, we used a smaller number of units and fewer training epochs for ANNs with a larger number of hidden layers. One can see that overall the training time and the ANN size grow significantly when we increase the number of hidden layers and units. At the same time, ANN approximation accuracy does not improve significantly. Thus, we conclude that the optimal choice would be to use fewer hidden layers, but include more units. In particular, for this example, the ANN with 3000 units provides reasonable approximation accuracy (1 ms) at reasonable training time and extremely small size of about 60 KB. All of these tests were performed on Intel Core i7-3770S CPU (3.1 GHz).

Table 1: ANN performance for traveltime approximation

Approximation and localization errors

Here, we consider the influence of ANN approximation error on the localization accuracy for 2D case. We use the velocity model in Figure 1. For this velocity model, we prepared synthetic microseismic data for true source location at point $(500, 1000)$ m. For this data, we used correct traveltimes, unit amplitudes, and the Berlage source wavelet with the central frequency of 40 Hz (period ΔT is 25 ms). The sampling rate was 1 ms. Random noise was added with the level of 20% of the maximum amplitude.

The testing parameters were the same as in the previous subsection. But here we consider only one ANN model with one hidden layer and 3000 units. The learning curve is shown in Figure 2. We saved several ANN models at a different stage of training as marked in the figure (i.e., 10, 30, 50, 70 and 100 epochs). Then, we used ANN approximated traveltimes to run the migration of the synthetic data using equation 1. Figures 3a-3c show the results of the migration (semblances) using traveltime approximation by three ANN models saved after: 10, 50, and 100 epochs, respectively. The color shows the semblance level, and the position of the maximum gives the source location estimate (see equation 3); the true source position is marked by red crosses.

From Figure 2 we see how ANN traveltime approximation accuracy gradually improves with the number of training epochs: MAE of 5.4 ms after 10 epochs (14% of ΔT), 3.2 ms after 50 epochs (10% of ΔT), and 1.2 ms after 100 epochs (5% of ΔT). Note that all these errors are considerably lower than the signal dominant period. Thus, it is not surprising that we see well-localized semblance in all panels. Moreover, estimating the location error using equation 4 gives the same value for all panels – about 14 m. This appears to be incorrect as we can compare the location of the maximum with the true source location in this case. For

three panels, the location errors are 30 m, 12 m, and 5 m correspondingly. So we conclude that in course of training, ANN first produces over-smoothed traveltime approximations that start focusing the semblance but producing a systematic bias in its maximum location. During the final training phase, this systematic bias is gradually removed. Also, note that if ANN training was not completed, then one can get clear semblance panels that may produce source location estimates with a systematic shift.

Complexity of the velocity model and approximation error

We next test how the training grid step and the velocity-model complexity affect the accuracy of the ANN approximation for 2D case. For this, we used the slightly smoothed Marmousi model (Figure 4a). We used 1361 receivers that were evenly distributed at the surface. For the validation set, we computed traveltimes from the detailed subsurface grid (1361×281 points with a step of 12.5 m spanning the entire model space). The validation traveltime table has about $5 \cdot 10^8$ values (3.9 GB). For this velocity model, we also prepared synthetic microseismic data for true source location at point $(6250, 2500)$ m. This simplistic data was generated by shifting to the corresponding time the same Berlage source wavelet with unit amplitude. The central frequency of the source wavelet was 40 Hz; the sampling rate was 2 ms; random noise was added with the level of 20% of the maximum amplitude.

To account for the complexity of the model, we used ANN with two hidden layers and 500 units in these layers. For the training set, we computed traveltimes from the coarse subsurface grid (ten times larger step, i.e. 135×28 points with 125-m step). The training took five epochs. Errors of ANN approximation are shown in Figure 4b. Each point shows ANN approximation error for the traveltimes from this location – MAE for all

receivers at the surface. The largest errors are located in the area of the sharp structural features (horizontal coordinate around 10 km). As an alternative approach, we use linear interpolation from the coarse training grid to the fine validation grid instead of the ANN approximation. Errors of such interpolation approach are shown in Figure 4c. Here again, each point shows MAE for traveltimes from this point to all surface receivers. One can see that the error distribution looks "patchy" in this case: zero errors at the points of the coarse grid and rapidly increasing errors in between. However, despite the resultant better accuracy of linear interpolation, the number of its coefficients depends on the grid step, whereas the ANN approximation does not. With this grid step, the ANN takes only 1 MB, when the linear interpolation requires about 300 MB, and more sophisticated interpolations providing better accuracy require greatly more memory to store.

ANN approximated traveltimes were used for the synthetic data migration which took about 3 hours on the CPU. The resultant semblance is shown in Figure 5a, and the zoom-in of the area around the source is shown in Figure 5b. One can see that it has a quite complicated structure. Note that we get the similar complicated structure of the semblance when using the correct traveltimes (for the fine grid). So they must be caused by the complexity of the model. We used equation 3 for estimating the source location. This gave us the correct source position within the grid step accuracy (12.5 m).

Training grid step and approximation error

We generated several training sets by using coarse grids spanning the same domain but with a different grid-step size. We then used traveltimes on these coarse grids for the ANN approximation and linear interpolation to a fine grid. The errors with respect to the grid

step are shown in Figure 6. Errors are presented in MAE (ms) for the whole fine validation set. Grid step is measured with respect to the step of the validation grid, e.g. a value of 12 means that the coarse grid step consists of every 12th point of the fine grid. The blue line shows errors of the ANN approximation, while the red line shows the errors of the linear interpolation. Figure 6a shows approximation errors for the simple model (see Figure 1); Figure 6b shows approximation errors for the Marmousi model (see Figure 4a). One can see that linear interpolation works better for a fine version of the training grid (small grid size), i.e. when the training grid is very close to the validation grid. However, for a coarse training grid, the ANN approximation starts outperforming the linear approximation. Also, note that ANN approximation performs very well for the simple velocity model.

Testing on 3D model

We have built the 3D velocity model by taking the part of the Marmousi model, see Figure 7, and duplicating it in the third direction (y-axis). We used 128×128 regular receiver grid at the surface. For the validation set, we computed traveltimes from the detailed subsurface grid ($128 \times 128 \times 128$ points with a step of 27.5 m). Then the validation traveltime table has 128^5 values and size of 256 GB. For the training set, we computed traveltimes from the coarse subsurface grid ($13 \times 13 \times 13$ points with 275-m step); then the training set size is about 275 MB. For this velocity model, we also prepared synthetic microseismic data for true source location at point $(1925, 1925, 1925)$ m. Synthetic data was generated by simply shifting to the corresponding time with the same Berlage source wavelet with unit amplitude. The central frequency of the source wavelet was 40 Hz; the sampling rate was 2 ms; random noise was added with the level of 20% of the maximum amplitude.

We used ANN with one hidden layer and 3000 units. The training took about 15 minutes (three epochs), and MAE for all subsurface points was about 2 ms. Then ANN approximated traveltimes were used for processing the synthetic data using the migration procedure. This migration took about 120 hours. The normalized summation energy from equation 1 (analog of semblance) is shown in Figure 8. The yellow color indicates the area of probable location. The estimated source location (maximum of the semblance) coincides with the true source position.

We then compared two implementations of the migration procedure. We show the results of in Table 2. The first strategy (labeled as "Table") relies on reading the huge traveltime table (256 GB) from the disk for using them in the migration. The second strategy (labeled as "ANN") uses ANN approximation of the traveltimes. The final ANN size is very small (82 KB) which corresponds to the dramatic compression rate of the order of 10^6. Traveltimes are then computed on-the-fly during the migration. We checked two migration implementations: CPU (Intel Core i7-3770S) and GPU (NVIDIA Tesla P4). These tests show that using ANN-compressed representation of the traveltime table has only a slight effect on the computational time (presented in hours). It is only 2 % slower on the GPU (although it may be 12 % slower on the CPU).

Table 2: 3D migration implementation using pre-computed (Table) and ANN-approximated (ANN) traveltimes.

DISCUSSION

In this section we wanted to make a few comments on the testing results from the previous section. ANN approximations of the traveltime table were trained on the traveltimes that were pre-computed on a coarse grid. Alternatively, various smart interpolation methods may be used for predicting traveltimes from the same values on the coarse grid. In our tests we used for comparison only simple linear interpolation. However, we speculate that ANN is a more promising tool for traveltime approximation because of several reasons. The main observation of this paper is that the ANN approximation provides unique strong compression together with efficient way of extracting required traveltime values during imaging.

Another aspect is that ANN training can be performed for irregular grid without any change. In our tests traveltimes were pre-computed on the regular grid. But we observed that the approximation errors tend to be concentrated in the areas with high velocity model complexity. One can locally add points with pre-computed traveltimes and add them to the ANN training without any change of the procedure. This may be a flexible tool for increasing accuracy of the ANN traveltime approximation. We have also tried to speed up the ANN training convergence by using the retraining methodology (it means that we start from the ANN that was trained for some other velocity model). Unfortunately, in our experiments the retraining idea showed bad results – training ANN from the scratch for a new velocity model was always better.

Finally, we want to mention that the ANN approximation does not only provide high compression rate, but it is also very efficient in computing derivatives of the approximated function. In particular, the ANN approximation provides easy access to kinematic attributes of the traveltime table, e.g. apparent velocities with respect to the receiver and the source

coordinates. So it might be useful for implementing various types of beam migration.

CONCLUSIONS

Implementation of the Kirchhoff migration operators utilizes large pre-computed traveltime tables (for a dense sampling of sources, receivers, and imaging points). In this paper, we used fully connected neural networks for approximating these traveltime tables. The neural network has to be trained for each velocity model, but then we achieved strong compression – up to six orders of magnitude (reducing size from 256 GB to 100 KB). We discussed the choice of optimal neural-network hyperparameters for boosting the training process. Our testing showed that one hidden layer is sufficient for approximating traveltimes in simple velocity models. Two hidden layers are required for approximating traveltimes in more complex velocity models. Note that here we considered only isotropic cases, but it is straightforward to apply the proposed approach for approximating and compressing traveltimes in anisotropic velocity models as well.

Highly compressed neural-network approximation makes it convenient to store, share, and use such traveltime approximation in processing large data volumes. During the migration procedure, traveltimes are then computed on-the-fly from this approximation. We showed tests of using the neural-network approximated traveltimes for microseismic data processing - source localization using the migration operator in 2D and 3D. Note that computing traveltimes on-the-fly from the compressed neural-network representation has a very small impact on the migration computational time – it adds about 2 % of the computational burden when the migration is implemented on a GPU.

ACKNOWLEDGEMENTS

We thank Yuriy Ivanov and two unknown reviewers for their valuable comments that have helped to improve the paper.

REFERENCES

Alkhalifah, T., 2011, Efficient traveltime compression for 3d prestack kirchhoff migration: Geophysical Prospecting, **59**, 1–9.

Anikiev, D., J. Valenta, F. Staněk, and L. Eisner, 2014, Joint location and source mechanism inversion of microseismic events: Benchmarking on seismicity induced by hydraulic fracturing: Geophysical Journal International, **198**, 249–258.

Araya-Polo, M., J. Jennings, A. Adler, and T. Dahlke, 2018, Deep-learning tomography: The Leading Edge, **37**, 58–66.

Artman, B., I. Podladtchikov, and B. Witten, 2010, Source location using time-reverse imaging: Geophysical Prospecting, **58**, 861–873.

Bianco, M. J., and P. Gerstoft, 2018, Travel time tomography with adaptive dictionaries: IEEE Transactions on Computational Imaging, **4**, 499–511.

Biondo, B., 2006, 3D seismic imaging: Society of Exploration Geophysicists.

Cybenko, G., 1992, Approximation by superpositions of a sigmoidal function: Mathematics of Control, Signals, and Systems (MCSS), **5**, 455–455.

De Wit, R. W., A. P. Valentine, and J. Trampert, 2013, Bayesian inference of earth's radial seismic structure from body-wave traveltimes using neural networks: Geophysical Journal International, **195**, 408–422.

Duncan, P. M., 2005, Is there a future for passive seismic?: First Break, **23**, 111–115.

Goodfellow, I., Y. Bengio, and A. Courville, 2016, Deep learning: MIT press.

Ivanov, Y., and A. Stovas, 2017, Traveltime parameters in tilted orthorhombic medium: Geophysics, **82**, no. 6, C187–C200.

Kingma, D. P., and J. Ba, 2014, Adam: A method for stochastic optimization: arXiv preprint arXiv:1412.6980.

Li, S., and S. Fomel, 2013, Kirchhoff migration using eikonal-based computation of traveltime source derivatives: Geophysics, **78**, no. 4, S211–S219.

Maxwell, S., D. Chorney, and S. Goodfellow, 2015, Microseismic geomechanics of hydraulic-fracture networks: Insights into mechanisms of microseismic sources: The Leading Edge, **34**, 904–910.

Nikitin, A. A., A. S. Serdyukov, and A. A. Duchkov, 2018, Cache-efficient parallel eikonal solver for multicore CPUs: Computational Geosciences, **22**, 775–787.

Panetta, J., T. Teixeira, P. de Souza Filho, C. da Cunha Filho, D. Sotelo, F. da Motta, S. Pinheiro, A. Rosa, L. Monnerat, L. Carneiro, and C. de Albrecht, 2012, Accelerating time and depth seismic migration by CPU and GPU cooperation: International Journal of Parallel Programming, **40**, 290–312.

Raissi, M., P. Perdikaris, and G. E. Karniadakis, 2017, Physics informed deep learning: Data-driven discovery of nonlinear partial differential equations: arXiv preprint arXiv:1711.10566.

Rastogi, R., A. Londhe, A. Srivastava, K. Sirasala, and K. Khonde, 2017, 3D kirchhoff depth migration algorithm: A new scalable approach for parallelization on multicore CPU based cluster: Computers & Geosciences, **100**, 67–75.

Sethian, J. A., 1996, A fast marching level set method for monotonically advancing fronts: Proceedings of the National Academy of Sciences, **93**, 1591–1595.

Stovas, A., and S. Fomel, 2018, Generalized velocity approximation: Geophysics, **84**, 1–65.

Vamaraju, J., and M. K. Sen, 2019, Unsupervised physics-based neural networks for seismic migration: Interpretation, **7**, no. 3, SE189–SE200.

Van Der Baan, M., D. Eaton, and M. Dusseault, 2013, Microseismic monitoring developments in hydraulic fracture stimulation, *in* Bunger, A. P., J. McLennan, and

R. Jeffrey, eds., Effective and sustainable hydraulic fracturing: International Society for Rock Mechanics and Rock Engineering: 439–466.

Zhao, H., 2005, A fast sweeping method for eikonal equations: Mathematics of computation, **74**, 603–627.

List of Figures

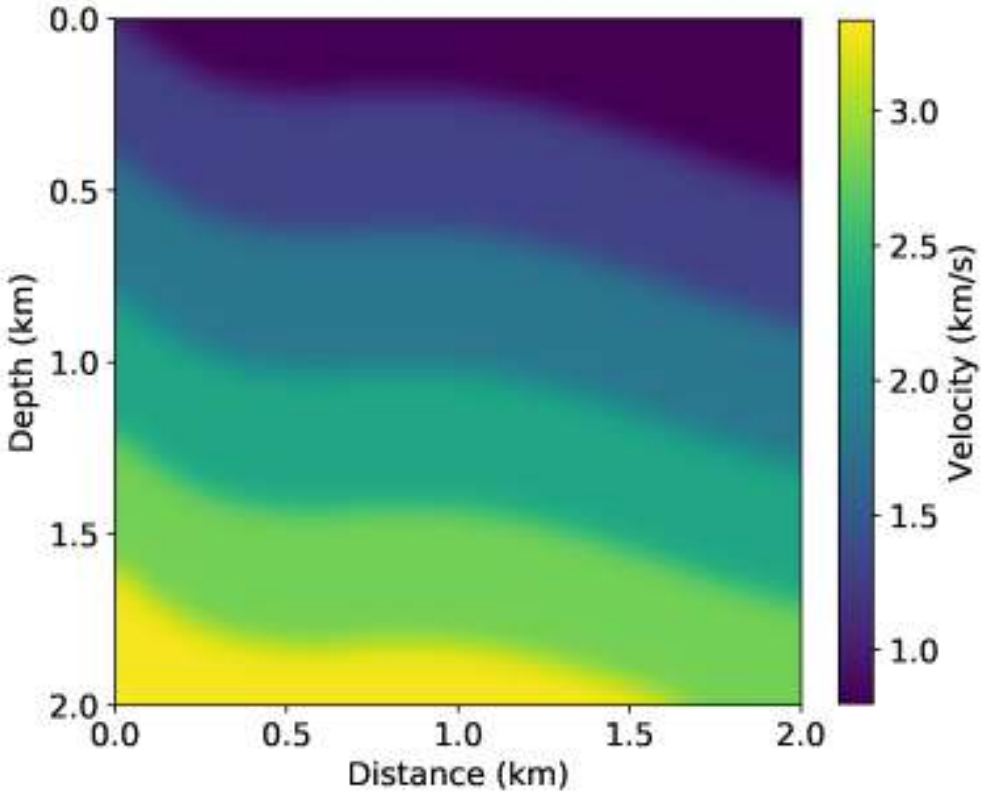

Figure 1: Velocity model used for choosing optimal ANN parameters.

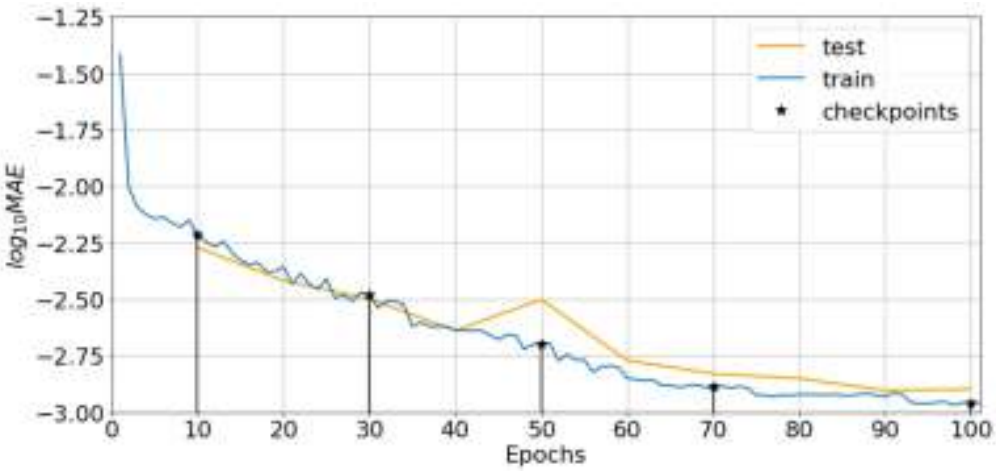

Figure 2: ANN learning curves on train (blue) and test (orange) datasets; black stars indicate moments of saving intermediate ANN models.

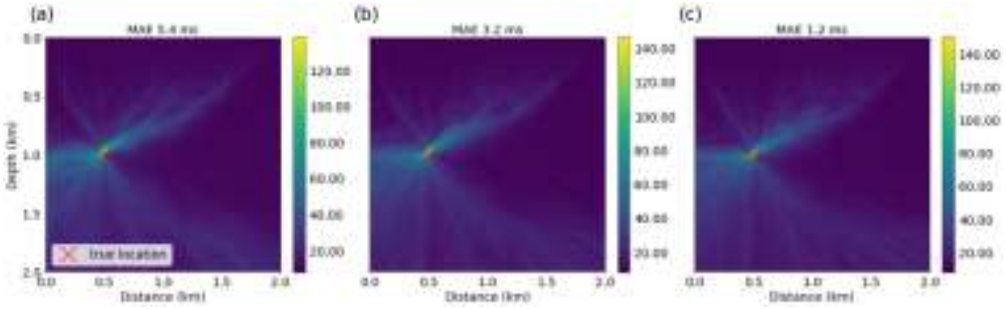

Figure 3: The results of migration (semblances) using traveltime approximation by different ANN models saved after: 10 epochs (a), 50 epochs (b), 100 epochs (c); approximation accuracy (MAE) is shown in the panel title; color shows semblance level; red cross - true source location.

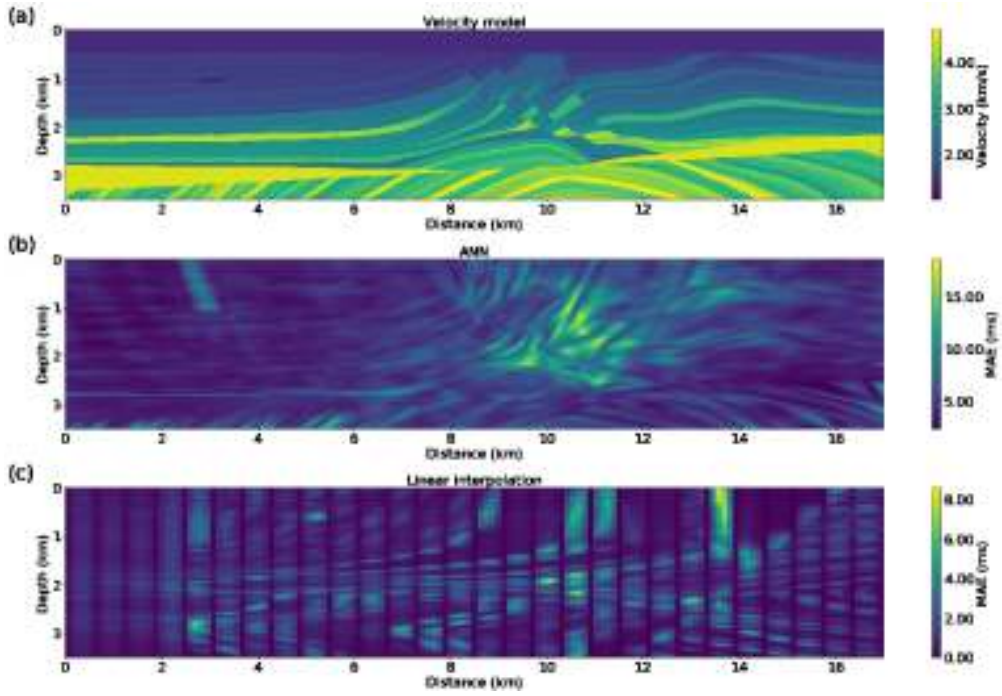

Figure 4: Approximation of traveltime table for Marmousi: velocity model (a); the map of errors of the ANN approximation for subsurface grid (MAE in ms) (b); the map of the errors of the linear interpolation (MAE in ms) (c).

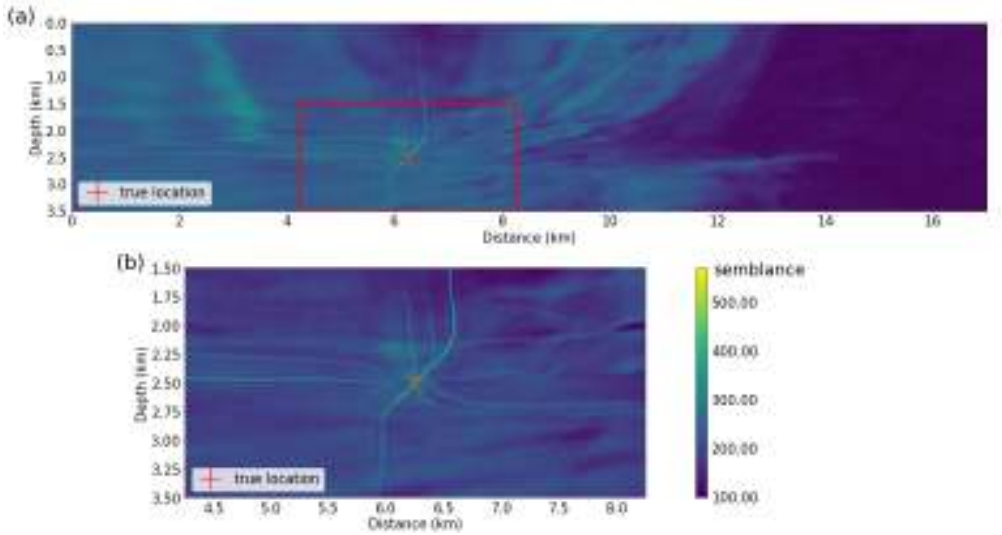

Figure 5: The semblance for the Marmousi model: for the whole area (a), zoom-in around the source area (b).

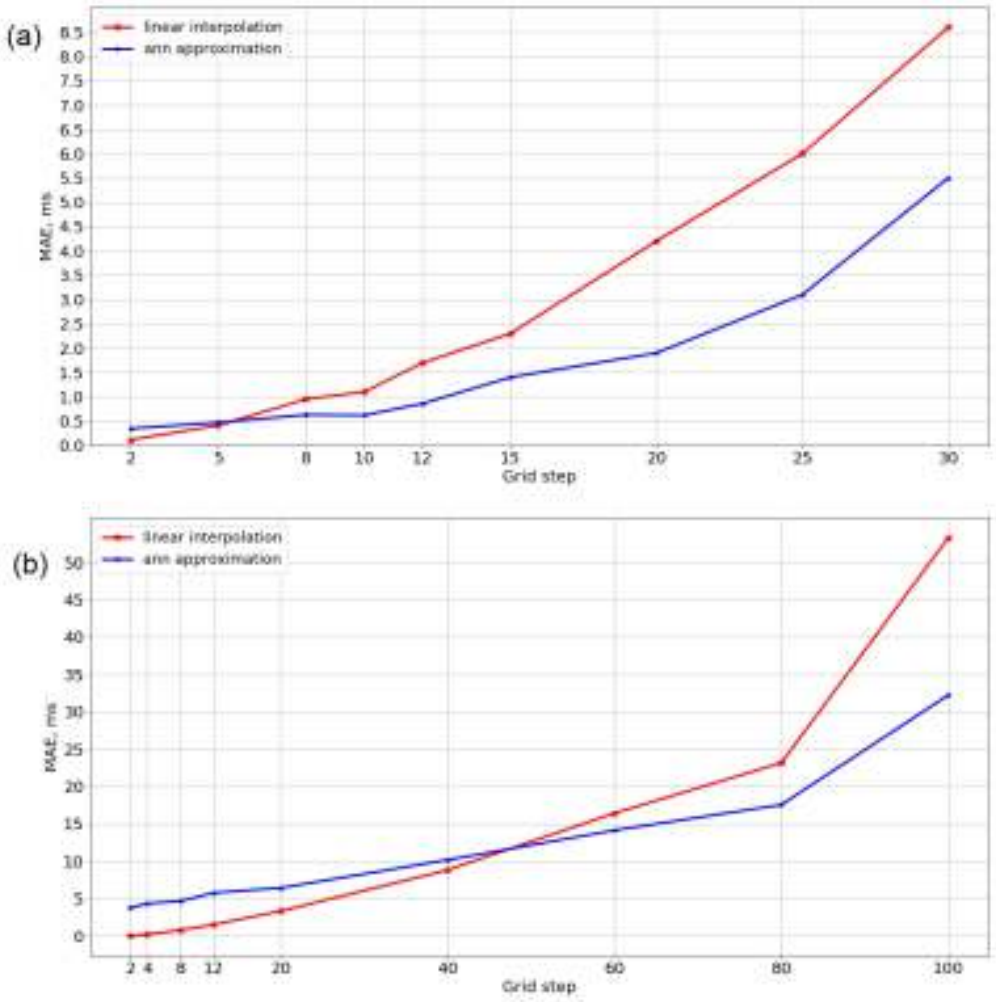

Figure 6: Traveltime approximation error for different grid step of the training set: simple model from Figure 1 (a), Marmousi model (b). ANN approximation (blue), linear interpolation (red).

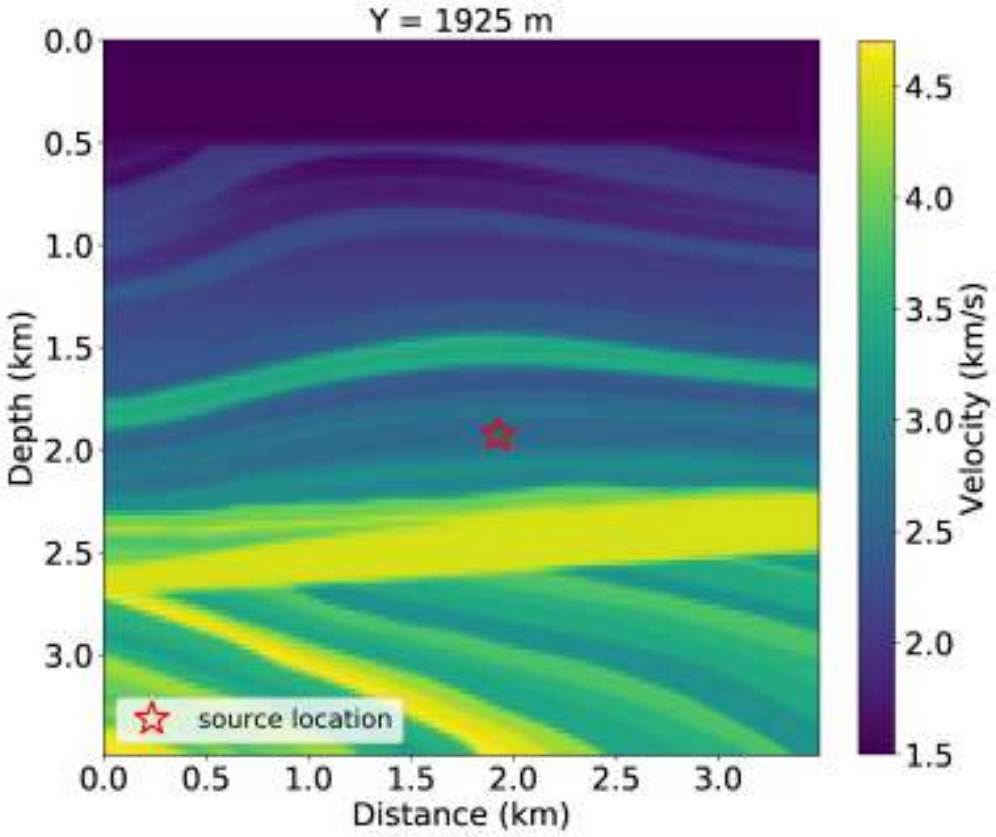

Figure 7: Section of the 3D velocity model at y=1925 m. The model was taken from the right part of the Marmousi and repeated for Y-axis. The red star is the source location.

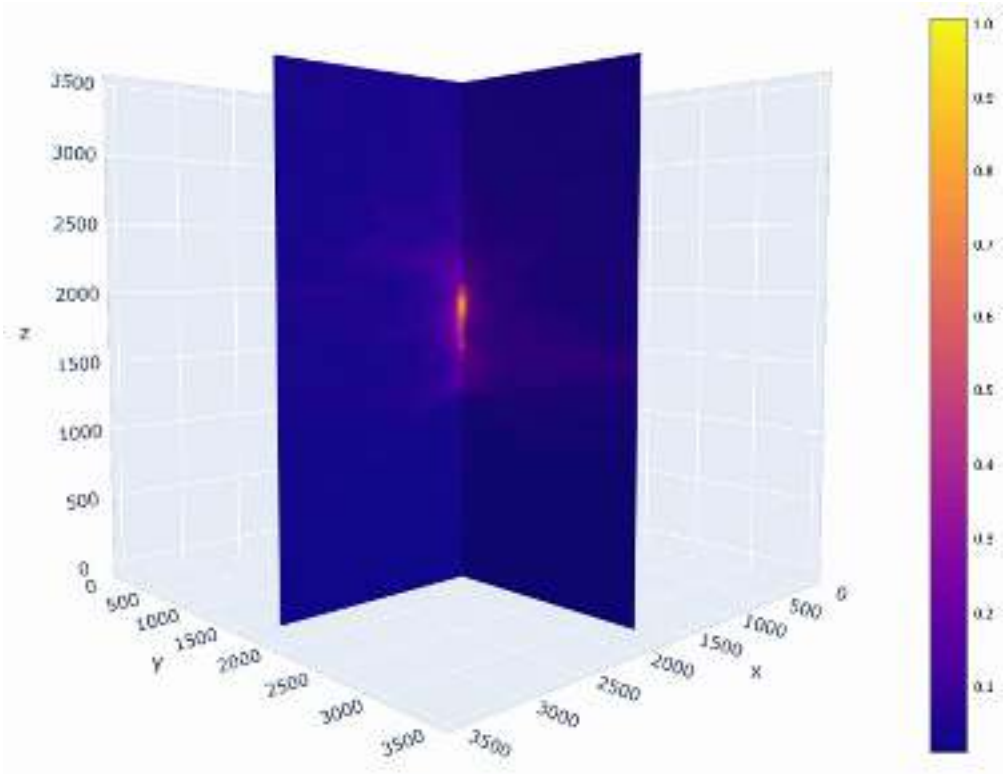

Figure 8. The result of migration in the 3D model using the ANN traveltime approximation; the color shows normalized summation energy from equation 1.

Table 1: ANN performance for traveltime approximation

1 hidden layer, 100 epochs								
Units	**100**	**250**	**500**	**1000**	**1500**	**2000**	**2500**	**3000**
MAE, ms	5.4	3.4	2.2	1.6	1.3	1.2	1.1	1.0
Time, sec	170	175	185	245	270	320	370	460
Size, KB	2	5	10	20	29	39	49	59
2 hidden layers, 80 epochs								
Units	**50**	**75**	**100**	**150**	**200**	**300**	**400**	**500**
MAE, ms	3.8	2.6	2.5	1.4	1.2	0.8	0.6	0.6
Time, sec	160	170	190	210	230	245	280	380
Size, KB	11	24	41	91	161	359	634	988
3 hidden layers, 60 epochs								
MAE, ms	3.8	2.7	1.9	1.3	1.0	0.7	0.6	0.5
Time, sec	130	145	170	210	230	250	290	480
Size, KB	21	46	81	180	318	711	1261	1967

Table 2: 3D migration implementation using pre-computed (Table) and ANN-approximated (ANN) traveltimes.

	Table	**ANN**
Size	256 GB	82 KB
Compression	$1.6 \cdot 10^6$ times	
CPU	105.9 h	119.7 h
GPU	99.1 h	100.7 h

DATA AND MATERIALS AVAILABILITY

Data associated with this research are available and can be obtained by contacting the corresponding author.