

---

# NEURAL EIKONAL SOLVER: IMPROVING ACCURACY OF PHYSICS-INFORMED NEURAL NETWORKS FOR SOLVING EIKONAL EQUATION IN CASE OF CAUSTICS

---

**Serafim Grubas**

Novosibirsk State University  
Russia, Novosibirsk, 630090  
serafimgrubas@gmail.com

**Anton Duchkov**

Dynamic analysis in seismology  
Institute of Petroleum Geology and Geophysics  
Russia, Novosibirsk, 630090  
DuchkovAA@ipgg.sbras.ru

**Georgy Loginov**

Dynamic analysis in seismology  
Institute of Petroleum Geology and Geophysics  
Russia, Novosibirsk, 630090  
loginovgeorgy@gmail.com

May 18, 2022

## ABSTRACT

The concept of physics-informed neural networks has become a useful tool for solving differential equations due to its flexibility. A few approaches are using this concept to solve the eikonal equation which describes the first-arrival traveltimes of acoustic and elastic waves in smooth heterogeneous velocity models. However, the challenge of the eikonal is exacerbated by the velocity models producing caustics, resulting in instabilities and deterioration of accuracy due to the non-smooth solution behaviour. In this paper, we revisit the problem of solving the eikonal equation using neural networks to tackle the caustic pathologies. We introduce the novel Neural Eikonal Solver (NES) for solving the isotropic eikonal equation in two formulations: the one-point problem is for a fixed source location; the two-point problem is for an arbitrary source-receiver pair. We present several techniques that provide relatively fast, stable, and accurate approximation of the eikonal in complex velocity models producing caustics: an improved factorization bounding the NES between the fastest and the slowest solutions to speed up the training; a non-symmetric loss function based on  $L_1$ -norm and a Hamiltonian of the eikonal to account for the errors caused by caustics; gaussian activation for a more accurate approximation of solution in caustics; a symmetrization to account for the reciprocity principle in the two-point problem. The tests on the Marmousi model showed that NES provides the relative mean absolute error of about 0.2-0.4% from the second-order factored Fast Marching Method, and outperforms existing neural-network solvers giving 10-60 times lower errors and 2-30 times faster training. With using a GPU, the training takes 1-5 minutes, and the inference time is comparable with the Fast Marching. The one-point NES provides the most accurate solution, whereas the two-point NES provides slightly lower accuracy but gives an extremely compact representation. It can be useful in various seismic applications where massive computations of traveltimes are required (millions of source-receiver pairs): ray modeling, traveltime tomography, hypocenter localization, and Kirchhoff migration. Source code is available at <https://github.com/sgrubas/NES>

**Keywords** Physics-Informed Neural Network · Eikonal equation · Seismic · Traveltimes

# 1 Introduction

The eikonal equation is a first-order non-linear partial differential equation (PDE) that is used in various applications: geometric optics, computer vision, image processing and others [1]. It is also used for computing the first-arrival traveltimes of seismic waves in many seismic applications: ray modeling [2]; ray multipathing [3]; traveltime tomography [4]; Kirchhoff migration [5]. There are well-known numerical techniques for solving the equation: Fast Marching Method [6, 7] and Fast Sweeping Method [8, 9]. Often, the seismic applications require massive computations of the traveltimes for many source-receiver pairs, demanding a high accuracy in complex velocity models.

The Physics-Informed Neural Networks (PINNs) were introduced for solving PDEs via artificial neural networks. To date, PINNs are used for many equations including Schrodinger, Burgers, Navier-Stokes and some others [10, 11]. PINNs are trained by minimizing a loss function which contains the governing PDE, corresponding boundary and initial conditions. The input data to the PINNs consist of collocation points in the domain of PDE. During the training, the neural network learns the PDE by minimizing the loss function on the training points in the domain of interest, giving a grid-free approximate solution to the PDE. Many enhancements for the PINNs have been proposed to improve its performance: adaptive activation [12, 13]; gradient of PDE [14]; loss balancing [15, 16]; improved architectures [16]; adaptive sampling [17, 18]. Some of the improvements almost do not increase computational cost, but others may considerably slow down the training.

PINNs have already been used for solving the eikonal equation [19, 20, 21]. Existing implementations of PINNs for the eikonal equation have low accuracy in the presence of caustics or require significant computational resources for the training. In this paper, we consider further development of a robust PINN design for solving the eikonal equation. The usage of PINN-based eikonal solver has several advantages in the seismic applications. For example, PINNs provide not only the traveltimes but the derivatives, which may speed up the solving of inverse problems: localization of earthquakes [22] and microseismic events [23]; traveltime tomography [24, 25]. Other advantage is compression of the traveltime tables for seismic imaging problems [26].

In this paper, we propose the novel Neural Eikonal Solver (NES) for solving the eikonal equation using the PINNs. First, we introduce a new version of factorizing the eikonal equation. It accounts for the singularity at the source and limits the classes of functions satisfying the eikonal equation. Second, we propose a new form of the loss function to account for the errors in caustic regions. Third, we propose the activation function for better approximation of the caustic regions. Fourth, we introduced a symmetrization of the solution to account for the reciprocity principle. Lastly, we proposed the input scaling for invariance to the units (m/s or km/s) and the random weight initialization for speeding up the convergence. The proposed techniques do not increase the computational overhead and are accompanied by systematic testing in different velocity models. In comparison to other PINN-based eikonal solvers the NES performance shows a higher traveltime accuracy and a faster training for an arbitrary velocity model.

Our paper is organized as follows: in chapter 2 we introduce the eikonal equation and our new PINN architecture (NES); in chapter 3 we test the NES performance for different velocity models and provide comparison with existing solutions; in chapter 4 we provide discussion of future developments and potential applications; chapter 5 summarises results and concludes our paper.

## 2 Theory

### 2.1 Eikonal equation and neural networks

The eikonal equation is used for computing traveltimes of seismic waves in a high-frequency approximation. For isotropic heterogeneous velocity models the equation is often written in the following form [27]:

$$\begin{aligned} \|\nabla_r \tau(\mathbf{x}_r)\| &= v(\mathbf{x}_r)^{-1}, \\ \tau(\mathbf{x}_s) &= 0, \end{aligned} \tag{1}$$

where  $\|\cdot\|$  is the Euclidean norm,  $\nabla_r \equiv (\partial_{x_r}, \partial_{y_r}, \partial_{z_r})$  is the gradient operator w.r.t. to coordinate  $\mathbf{x}_r = (x_r, y_r, z_r)$ ,  $v(\mathbf{x}_r)$  is a velocity model. This equation defines the first-arrival traveltimes for a fixed source location  $\mathbf{x}_s = (x_s, y_s, z_s)$ , constrained by the boundary condition  $\tau(\mathbf{x}_s) = 0$ . There are several numerical techniques for solving the equation in a stable manner, e.g. Fast Marching Method (FMM) [6] and Fast Sweeping Method (FSM) [8].

In seismic applications it is commonly needed to compute traveltimes for multiple sources [28, 29]. Therefore, we can consider traveltime field as function of two points  $T(\mathbf{x}_s, \mathbf{x}_r)$  that should satisfy the reciprocity principle for monotype

waves:

$$\begin{aligned}\|\nabla_r T(\mathbf{x}_s, \mathbf{x}_r)\| &= v(\mathbf{x}_r)^{-1}, \\ \|\nabla_s T(\mathbf{x}_s, \mathbf{x}_r)\| &= v(\mathbf{x}_s)^{-1}, \\ T(\mathbf{x}_s, \mathbf{x}_r) &= T(\mathbf{x}_r, \mathbf{x}_s), \\ T(\mathbf{x}_s, \mathbf{x}_s) &= 0,\end{aligned}\tag{2}$$

where  $\nabla_s \equiv (\partial_{x_s}, \partial_{y_s}, \partial_{z_s})$  and  $\nabla_r \equiv (\partial_{x_r}, \partial_{y_r}, \partial_{z_r})$  denote gradients with respect to the source  $\mathbf{x}_s = (x_s, y_s, z_s)$  and the receiver  $\mathbf{x}_r = (x_r, y_r, z_r)$  coordinates. With that,  $\nabla_s$  and  $\nabla_r$  define the source-part and the receiver-part eikonal equations. Unlike the eikonal for a fixed source (one-point form) (eq. 1), this system describes the first-arrival traveltimes for all the source-receiver pairs, and we will call it the two-point eikonal.

Physics-Informed Neural Networks (PINN) is a concept for solving PDEs usually by means of using the Fully-Connected Neural Networks (FCNNs) [11]. The FCNN is a universal approximator [30] representing the mapping  $\mathbf{f}_\theta(\mathbf{x}): \mathbb{R}^n \rightarrow \mathbb{R}^m$ . FCNN with  $K - 1$  hidden layers can be written in the following form:

$$\mathbf{h}_k = g_k(\Theta_k \mathbf{h}_{k-1} + b_k), \quad k = 0 \dots K,\tag{3}$$

where  $k$  is a hidden layer index,  $\mathbf{h}_k \in \mathbb{R}^{N_k \times 1}$  is a hidden state,  $N_k$  is a number of units on a hidden layer  $k$ ,  $\Theta_k \in \mathbb{R}^{N_k \times N_{k-1}}$  are kernel weights,  $b_k$  is a bias weight,  $g_k$  is an activation function providing the non-linearity. For an arbitrary regression problems the last activation  $g_K$  is usually the linear function. Note,  $\mathbf{h}_0 = \mathbf{x} \in \mathbb{R}^n$  is the input data, and  $\mathbf{f}_\theta(\mathbf{x}) = \mathbf{h}_K \in \mathbb{R}^m$  is the output, where  $\theta$  denotes all the weights of the network  $(\Theta_1, b_1, \dots, \Theta_K, b_K)$ . The weights  $\theta$  are selected during the training procedure, which implies adjusting the weights  $\theta$  by minimizing the loss function  $L(\theta)$  with respect to the training data set. This loss function represents the distance between the neural-network output and the training set. Since a neural network  $\mathbf{f}_\theta(\mathbf{x})$  is a nonlinear operator, training is often performed using the optimization methods based on a gradient descent.

In our work we used the Adam optimization method, which is the most popular and provides relatively fast and accurate convergence for the majority of problems [31, 32]. For the problems where the training-set size  $M$  is huge, the data are split into small batches to fit the machine memory (mini-batch gradient descent). In the computational frameworks such as Tensorflow [33] the algorithm of Automatic Differentiation (AD) is used, which allows computing the analytical derivatives of any order at a machine precision [34]. This Automatic Differentiation is crucial for PINNs because it allows both constructing a loss function  $L(\theta)$  for given PDE and computing gradient for its optimization.

## 2.2 New Neural Eikonal Solver (NES)

We introduce new Neural Eikonal Solver (NES) based on the PINN concept. This approach is mesh-free, unlike the finite-difference methods (FMM, FSM). The target grid of traveltimes can be initialized with an arbitrary distribution of collocation points and will not require any modifications of the algorithm. Existing neural-network eikonal solvers [21, 19] exhibit different performance depending on the complexity of the velocity model. They may require longer training time or accuracy deterioration for complicated velocity models. Main cause for this problems is that the eikonal equation is a non-linear PDE that can produce non smooth front shapes in heterogeneous velocity models producing caustics. Thus, we want to revisit the way of constructing the neural-network solver in order to make it fast and reach minimum number of trainable parameters providing high accuracy of the solution for complicated velocity models.

The NES has two versions:

- solving one-point problem with fixed source (NES-OP), equation 1;
- solving two-point problem for arbitrary source-receiver pair (NES-TP), equation 2.

### 2.2.1 Improved factorization of the eikonal equation

The solution to the eikonal equation has singularity at the source point which deteriorates the solution accuracy in this region. To avoid this problem, it was suggested to use factorization [9], i.e. look for the eikonal solutions in the form (for the one-point and the two-point problem respectively):

$$\tau(\mathbf{x}_r) = R \cdot f(\mathbf{x}_r),\tag{4}$$

$$T(\mathbf{x}_s, \mathbf{x}_r) = R \cdot F(\mathbf{x}_s, \mathbf{x}_r),\tag{5}$$

where  $R = \|\mathbf{x}_r - \mathbf{x}_s\|$  is the distance function and the solution in an isotropic homogeneous velocity model ( $v = 1$ ),  $f(\mathbf{x}_r)$  and  $F(\mathbf{x}_s, \mathbf{x}_r)$  are unknown functions representing deviations from the solution in homogeneous model (for the one-point and the two-point eikonal respectively). The factor  $R$  emulates the source-point singularity, which significantly enhances the accuracy of FMM [7] and FSM [9]. This factorization is also used in the neural-network eikonal solvers [19, 21].

If we use PINNs for the eikonal solution, the functions  $f(\mathbf{x}_r)$  and  $F(\mathbf{x}_s, \mathbf{x}_r)$  are to be approximated. However, their free form requires PINN to search the solution from the huge class of functions which may be time consuming resulting in the long training. Therefore, we propose the improved factorization which limits the class of functions for search, thereby significantly boosting the training speed of the NES. Let us consider the following expression for the NES-OP and NES-TP solutions respectively:

$$\tau_\theta(\mathbf{x}_r) = R \cdot \bar{\sigma}(f_\theta(\mathbf{x}_r)), \quad (6)$$

$$T_\theta(\mathbf{x}_s, \mathbf{x}_r) = R \cdot \bar{\sigma}(F_\theta(\mathbf{x}_s, \mathbf{x}_r)), \quad (7)$$

where  $f_\theta$  and  $F_\theta$  are the neural-network outputs,  $\bar{\sigma}$  is a bounding function or the output activation function ( $g_K$  in eq. 3). The NES solution must be strictly positive function  $\tau_\theta(\mathbf{x}_r) > 0 \forall \mathbf{x}_r \neq \mathbf{x}_s$  (same for  $T_\theta$ ). Since  $R \geq 0$ , then the second term must always be strictly positive  $\bar{\sigma} > 0$ . Note,  $R$  is the homogeneous solution for  $v = 1$  and  $\bar{\sigma}$  is supposed to be a small deviation from the homogeneity. If we assume that the unity velocity is the minimal possible value, so then  $R$  constrains the range of  $\bar{\sigma}$  to be in  $(0, 1]$ . Any velocity model  $v$  can be represented using its ranges:  $v_{min} < v < v_{max}$ , where  $v_{min}, v_{max}$  are minimal and maximal values in a considered velocity model respectively. This bounding allows to constrain  $\bar{\sigma}$  in the range of  $[1/v_{max}, 1/v_{min}]$ . It bounds the NES solution to be  $\tau_\theta, T_\theta \in [R/v_{max}, R/v_{min}]$ , i.e. constrains the NES between the fastest and the slowest solutions. We suggest the following bounding function  $\bar{\sigma}$ :

$$\bar{\sigma}(x) = \left( \frac{1}{v_{min}} - \frac{1}{v_{max}} \right) \sigma(x) + \frac{1}{v_{max}}, \quad (8)$$

where  $\sigma$  may be any function defined in the range of  $[0, 1]$ . We propose the sigmoidal function because it showed the best performance:

$$\sigma(x) = \frac{1}{1 + e^{-x}}, \quad (9)$$

The explanation of the improved factorization is shown in Figure 1, where the true solution is always bounded by the slowest and fastest solutions. That also means that if a velocity model is homogeneous  $v_{min} = v_{max} = v$  then the NES converges to the true solution, i.e.  $\tau_\theta = R/v$  everywhere (same for  $T_\theta$ ).

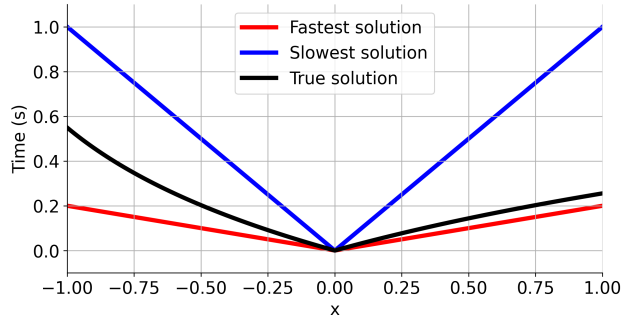


Figure 1: Solution of the eikonal equation for 1D velocity model with a linear gradient. Black line indicates the true solution which is bounded by the slowest (blue line) and the fastest (red line) solutions.

### 2.2.2 Imposing reciprocity principle

The solution to the two-point eikonal 2 can benefit when based on the reciprocity principle. The reciprocity  $T_\theta(\mathbf{x}_s, \mathbf{x}_r) = T_\theta(\mathbf{x}_r, \mathbf{x}_s)$  can be introduced as an additional constraint to the loss function or within the factorization. The first approach is a soft constraint which does not guarantee the reciprocity. The soft constraint means that it is introduced through the penalty so that the constraint is not strictly satisfied. The second approach is a hard constraint which can guarantee reciprocity. The hard constraint means that it is strictly satisfied through imposing a certain behaviour on the function. We suggest to use the hard type of constraint to guarantee the desirable solution behaviour. The reciprocity principle requires constructing a symmetric function with respect to permutation of the two arguments  $\mathbf{x}_s$  and  $\mathbf{x}_r$ , which can be done by including the averaging of the permutations in the NES-TP approximation:

$$T_\theta(\mathbf{x}_s, \mathbf{x}_r) = R \cdot \bar{\sigma} \left( \frac{F_\theta(\mathbf{x}_s, \mathbf{x}_r) + F_\theta(\mathbf{x}_r, \mathbf{x}_s)}{2} \right). \quad (10)$$

This guarantees reciprocity everywhere  $T_\theta(\mathbf{x}_s, \mathbf{x}_r) \equiv T_\theta(\mathbf{x}_r, \mathbf{x}_s)$  and combines the receiver- and source-part equations (eq. 2) into a single one. The last assumption is valid as the gradients  $\nabla_r T_\theta$  and  $\nabla_s T_\theta$  for 10 define the same function but measured at different spatial coordinates  $(\mathbf{x}_s, \mathbf{x}_r)$  and  $(\mathbf{x}_r, \mathbf{x}_s)$  respectively. Thus, the expression 10 provides the required reciprocity and only one equation is needed to solve the two-point eikonal system 2. For the simplicity we will consider only the receiver-part equation for the NES-TP hereafter.

### 2.2.3 Form of the loss function

Here we discuss a proper choice of the loss-function to achieve better training results for the NES. The factor  $R$  in 6 and 7 automatically satisfies the boundary condition allowing to exclude it from the loss function. Thus, we need to use only the eikonal equation while forming the loss function in NES. Note that the eikonal equation is zero-level set of a corresponding Hamiltonian that may take various forms [27]. In particular, we can write for the one-point 1 and the two-point formulations 2 correspondingly:

$$\mathcal{H}_p(\mathbf{x}_r, \tau) = \frac{1}{p} [v(\mathbf{x}_r)^p \|\nabla \tau(\mathbf{x}_r)\|^p - 1] = 0, \quad (11)$$

$$\mathcal{H}_p(\mathbf{x}_s, \mathbf{x}_r, T) = \frac{1}{p} [v(\mathbf{x}_r)^p \|\nabla_r T(\mathbf{x}_s, \mathbf{x}_r)\|^p - 1] = 0, \quad (12)$$

where  $p \neq 0$ .

So we have a family of possible loss functions depending on the parameter  $p$  (for the one-point and the two-point formulations respectively):

$$L_{op}(\theta, p) = \frac{1}{N_r} \sum_{\mathbf{x}_r} |\mathcal{H}_p(\mathbf{x}_r, \tau_\theta)|, \quad (13)$$

$$L_{tp}(\theta, p) = \frac{1}{N_{sr}} \sum_{\mathbf{x}_s, \mathbf{x}_r} |\mathcal{H}_p(\mathbf{x}_s, \mathbf{x}_r, T_\theta)|, \quad (14)$$

where subscripts  $op$  and  $tp$  refer to the one-point and two-point eikonals respectively,  $N_r$  is a number of receiver points,  $N_{sr}$  is a number of source-receiver pairs,  $|\cdot|$  denotes an absolute value,  $\mathcal{H}_p$  refers to a Hamiltonian form (eqs. 11 and 12). These loss functions are based on the  $L_1$ -norm (mean-absolute error) of the Hamiltonian. Below we explain the advantages of the proposed loss function.

#### Loss function for robust solution

In many PINNs application for different PDEs the  $L_2$ -norm (mean-squared error) in the loss function has been used [11, 12, 13, 14, 15, 16, 17, 18] including the eikonal equation [19, 21]. The  $L_2$ -norm is designed for a normal Gaussian distribution of the errors. In case of the eikonal equation the error distribution may not be Gaussian for an arbitrary heterogeneous velocity models, especially when the eikonal solution may have caustics. At the caustics the seismic-rays trajectories intersect each other forming singularity zone where the gradients are discontinuous and undefined, that is, the equation is not satisfied. In vicinity of the caustic singularities the solution itself is not stable which leads to the presence of errors. Unlike the source-point singularity, the caustics singularities cannot be localized in advance. Therefore, we have to account for the errors related to the caustic singularities using other type of error metric. In regression problems, the  $L_1$ -norm (mean-absolute error) is known to provide the solution which is robust to outliers, assuming that the error has random Laplace distribution [35]. That is why, we suggest to use  $L_1$ -norm (mean-absolute error) in the loss functions 13 and 14.

#### Non-symmetric error distribution

The errors due to the caustic singularities have a non-symmetric distribution. As we mentioned, the eikonal solution tends to be unstable at the caustic singularities which are the zones of a non-smooth behaviour of a solution. The NES approximation, which is a smooth function, tends to smooth this zone (see Figure 2). From the gradient of the solution we can obtain a predicted velocity model in a form of  $v_\theta = \|\tau_\theta\|^{-1}$ , and the difference with the actual velocity model  $v$  may show how well the eikonal equation is satisfied. The smoothing at caustic singularities means that the NES approximation has a lower magnitude of the solution gradients (Figure 2b), that is, the predicted velocity exceeds the actual velocity values ( $v_\theta > v$ ). Thus, the errors related to the caustic singularities are shifted to the zone of exceeding velocity values. This observation was confirmed with the internal experiments. Therefore, we should allow the high-velocity errors to exist with the higher probability than the low-velocity errors which do not represent caustics. To do so, we should make our Laplace distribution non-symmetric so that the high-velocity errors have higher probability.

The Hamiltonian form (eqs. 11 and 12) can help to take a non-symmetric distribution into account. The Hamiltonian  $\mathcal{H}_p$  uses the ratio of the predicted and actual velocities (the term  $v^p \|\tau_\theta\|^p$  which is equivalent to  $v^p/v_\theta^p$ ). If the predicted velocity is higher than the actual one, the ratio is less than unity  $v^p/v_\theta^p < 1$ . It means that we can reduce the ratio magnitude when it is less than unity (decrease the significance of high-velocity errors) and enhance when it is higher (increase the significance of low-velocity errors) by setting up the  $p$  value to be  $> 1$ . Our tests showed that the optimal value is  $p = 2$  (in extreme cases with many caustics it can be  $p = 3$ ). Noteworthy, the Hamiltonian  $\mathcal{H}_p$  with  $p = 2$  defines the ray-tracing system parameterized by traveltime [27].

#### Non-dimensional loss

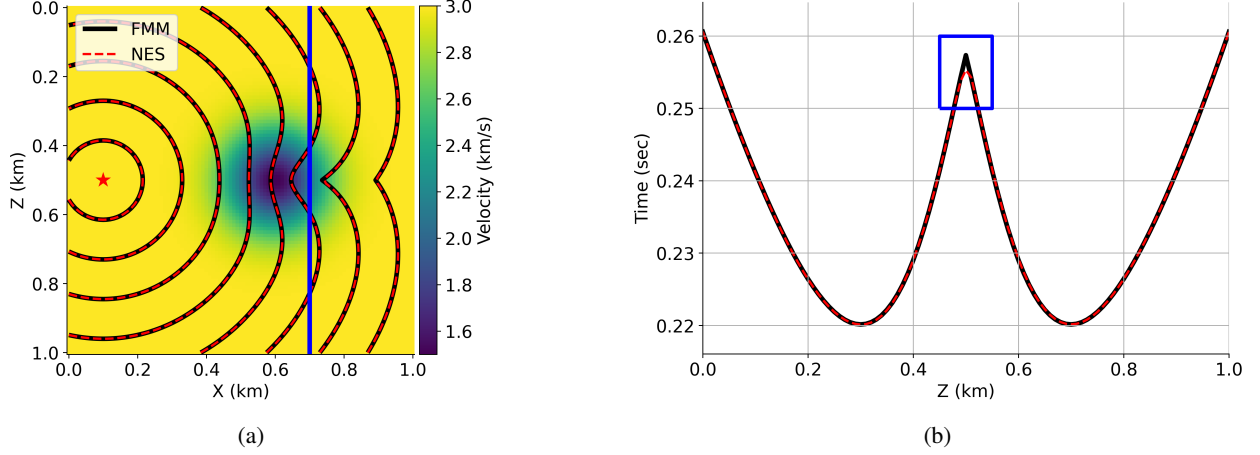


Figure 2: The example of the caustic singularities. **(a)** shows the velocity model, where red star indicates the source position, the black solid contours represent the reference solution (the second-order factored FMM), the red dashed contours represent the NES-OP solution, the blue vertical line at  $x = 0.7$  km indicates a section for **(b)**. **(b)** showcases the vertical section of the solutions where the blue box highlights the position of the caustic singularity. From **(b)** we can see that the NES (red dashed line) smooths the caustic singularity (in blue box) at  $z = 0.5$  km.

The Hamiltonian-based loss function (eqs. 13 and 14) is non-dimensional as it is scaled by the actual velocity model. The loss function becomes non-sensitive to the velocity model and may give the same loss values for the velocity models of different order of values. It allows setting a single tolerance value as a training conditioning for the loss function for any velocity model.

#### 2.2.4 Activation function

The choice of the hidden activation  $g_k$  (eq. 3) is important because it describes the basis functions for solution approximation. All the functions used for PINN should be smooth enough to provide the existence of the necessary derivatives. The neural-network output is differentiated w.r.t. inputs (spatial coordinates) in computing the losses (eqs. 13 and 14) and also w.r.t. weights during the training procedure. Thus, we need to consider at least two-times differentiable activation functions.

The shape of the caustic singularities in Figure 2b (at the center  $z = 0.5$  km) are smoothed by the NES approximation. We suggest to approximate these singularities by a set of the gaussian functions  $g_k(x) = e^{-x^2}$  because it has natural extremum resembling the singularity shape and also is infinitely differentiable function. The gaussian function is also called as Radial Basis Function which are universal approximators [36]. In all our tests, the *gauss* activation showed the significantly better performance than the other activation functions for the velocity models with caustics.

#### 2.2.5 Other neural-network parameters

There are many hyperparameters of the neural networks that may influence the performance of a training and accuracy of a result. The common practice is to test the different set of parameters and select it based on the best performance [37]. In this subsection we consider the choice of initial weights, input data scaling, and the type of connection between hidden layers. The discussed parameters are chosen among the other results of our research based on the most significant impact on the accuracy and performance.

The weights of neural network must be initialized before the training. These initial weights are used to be initialized in random manner, but the type of the distribution can be different. We tested different types of distribution ( $\Theta_k$  in equation 3) which are available in Keras framework [38]. The *he\_normal* and *he\_uniform* [39] types of random distributions provided a considerably faster convergence of the loss function, which starts decreasing from the first epochs and almost without any stagnation.

The input data scaling or normalization is another popular technique to improve the training process of neural networks: considerably speeds up the convergence of the gradient descent and standardizes the learning rate definition. We propose the scaling of the inputs  $\mathbf{x}_r$  (and  $\mathbf{x}_s$ ) by  $\max |\mathbf{x}_r|$  (or  $\max(|\mathbf{x}_s|, |\mathbf{x}_r|)$ ) which maps input data to the range of  $[-1, 1]$ . Such a scaling prevents the neural network to be sensitive to the units of the spatial coordinates ( $m$  or  $km$ ).

We have considered several types of architectures of neural networks and types of connection between the hidden layers. The fully-connected layers provide the best performance compared to the layers with the skip-connections (residual blocks) [40] and attention mechanism [16]. We tested the influence of an architecture complexity on the solution accuracy (number of hidden layers and units). Our tests showed that the total number of weights  $10^3 - 10^4$  is enough to provide the accuracy of 0.1%, and more weights  $> 10^4$  do not give any improvement. From this range, we chose 4 hidden layers with 75 units on each as a baseline because it provides the faster convergence with the least number of weights. However, one still can use less weights for simple velocity models, and more weights for complex velocity models.

### 2.2.6 Final architectures

We propose the architectures for the NES-OP and NES-TP which are visualized in Figure 3. The NES-OP approximates the equation for a fixed source location, when a finer grid of collocation points can be used to train for a smaller time to get a very accurate solution. The NES-TP provides the generalized solution for any source-receiver pair. It means that we may need more collocation points and training time, but if we need the solution for many sources, it is recommended to use the NES-TP. Our general recommendation is to use the NES-TP unless a special accuracy is needed. It is also worth noting that the NES package is written using Tensorflow backend [33] and Keras API [38], which supports parallelism on any heterogeneous systems with single or multiple CPUs, GPUs, or TPUs.

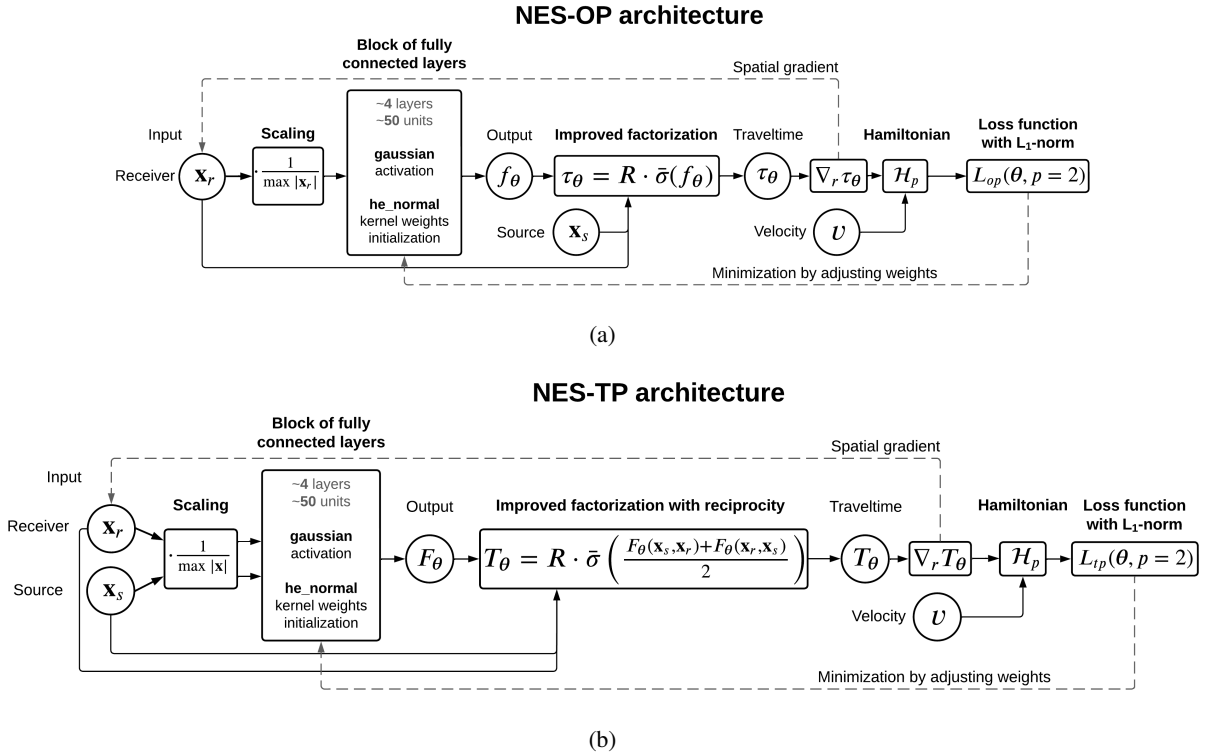


Figure 3: The diagrams of the NES-OP (a) and the NES-TP (b) architectures. The text in bold indicates our main contribution. The gray dashed arrow in the spatial gradient means that traveltime is differentiated with respect to the  $\mathbf{x}_r$ . The gray dashed arrow from the loss function indicates the backpropagation algorithm for the training. The symbol  $\mathbf{x}$  in the scaling layer in (b) indicates that the maximum value is calculated for both source and receiver locations.

## 3 Numerical testing

In this chapter, we provide: the testing of the proposed NES architectures (Figure 3) in the velocity model from Figure 2a; the benchmark and comparison of the NES-OP and the NES-TP performance in Marmousi model; the comparison of the NES with the existing packages PINNeik and EikoNet [21, 19]. All error estimations are based on the reference solution represented by the second-order factored FMM [7] implemented in the package *eikonal\_fm* [41].

The used error metric is relative mean absolute error (RMAE):

$$RMAE = \frac{\sum |\tau_{ref} - \tau_{\theta}|}{\sum |\tau_{ref}|}, \quad (15)$$

where  $\tau_{ref}$  is the reference solution,  $\tau_{\theta}$  is the NES-OP approximation and can be replaced with  $T_{\theta}$  for the NES-TP. The summation is performed for the set of points where we test the accuracy.

### 3.1 Advantage of NES architecture

In this section we show the influence of the proposed NES architectures (Figure 3). The velocity model for the test is presented in Figure 2a. The NES-OP was trained on the grid of  $61 \times 61 - 1$  evenly spaced receivers without the source location because of the singularity. The source is located at the same position as shown in Figure 2a. The RMAE for the NES-OP was measured on  $101 \times 101$  points (2.7 times finer grid). The NES-TP was trained on 25000 collocation points of source-receiver pairs randomly and uniformly distributed in the domain of the same velocity model. The RMAE for the NES-TP was measured on a set of  $3 \times 3$  sources evenly spaced in the domain with  $101 \times 101$  receivers for each. Batch size for both cases was 1/4 of the training set and Adam method for optimization was used [31] with the learning rate of  $2.5 \cdot 10^{-3}$ . For both scenarios, the neural networks contain 4 hidden layer with 75 units on each layer.

We test the cumulative influence of the proposed NES architectures (all highlighted features by bold font in Figure 3). It includes the improved factorization (eqs. 4 and 5), the loss function with a  $L_1$ -norm of the Hamiltonian form with  $p = 2$  (eqs. 13 and 14), gaussian activation of the hidden layers (subsection 2.2.4), the *he\_normal* weights initialization, the input data scaling (subsection 2.2.5), and the reciprocity principle (eq. 10). We show only cumulative effect because the influence of each component may differ for various set of other settings in the architecture. For example, influence of the gaussian activation with using improved factorization differs from one without the improved factorization. The systematic tests for various models show that the contributions of all the proposed features are approximately equal. The results are shown in Figure 4. It shows the training dynamics of the RMAE. Naive refers to an architecture without the improved factorization, with the loss function based on  $L_2$ -norm and with conventional forms of eikonal equations (eqs. 1 and 2), with hyperbolic tangent as hidden activation, and without the symmetrization for the reciprocity of the NES-TP. It is clearly seen that the RMAE with the NES architectures is drastically decreased by more than two orders of magnitude and the stability improved (narrower min-max bands). Overall, after 3000 epochs the RMAE was about 0.16% and 0.07% for the NES-OP and the NES-TP respectively. These tests were performed on the GPU RTX 3070 Laptop which took 43 and 75 seconds for the training of the NES-OP and the NES-TP respectively.

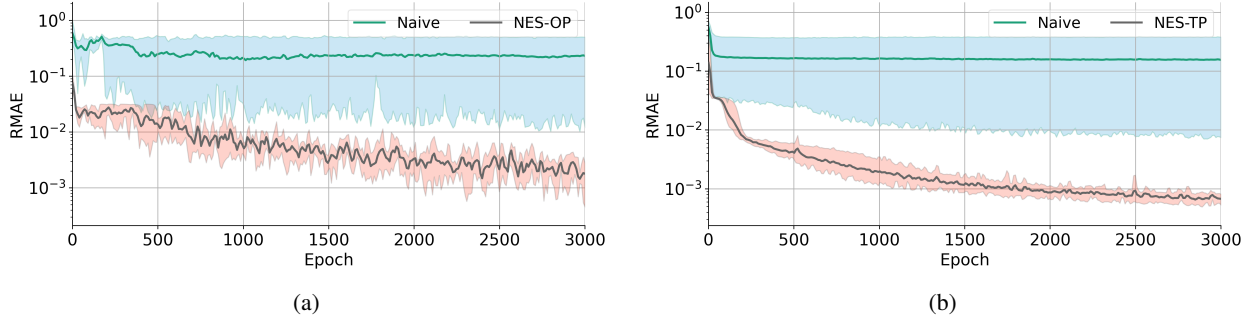


Figure 4: The cumulative influence of the NES architectures on the training dynamic in the velocity model from Figure 2a. **(a)** shows results for the NES-OP architecture (Figure 3a). **(b)** indicates results for the NES-TP architecture (Figure 3b). Lines indicate average value of RMAE over 5 independent launches of NES, transparent areas around lines indicate min-max band.

### 3.2 Benchmark on Marmousi 2D

Here we provide the test of the NES on the most heterogeneous part of the Marmousi model which was smoothed by a gaussian filter. For the test we chose  $7 \times 7$  sources (49 in total) with  $101 \times 101$  receivers to compare with the reference solution. Sources and receivers are evenly spaced in the domain of the velocity model. The sample solutions are presented in Figure 5 for the three different sources. Note, that a single source means a single NES-OP, that is, we have 49 different NES-OPs for 49 sources, whereas a single NES-TP is defined for any number of sources. All 49 NES-OPs and the single NES-TP were trained for 3000 epochs. For each NES-OP the training set was  $101 \times 101 - 1$  (10200), whereas for the NES-TP the training set contained  $49 \times 101 \times 101 - 49$  (499800) points. Again, the source



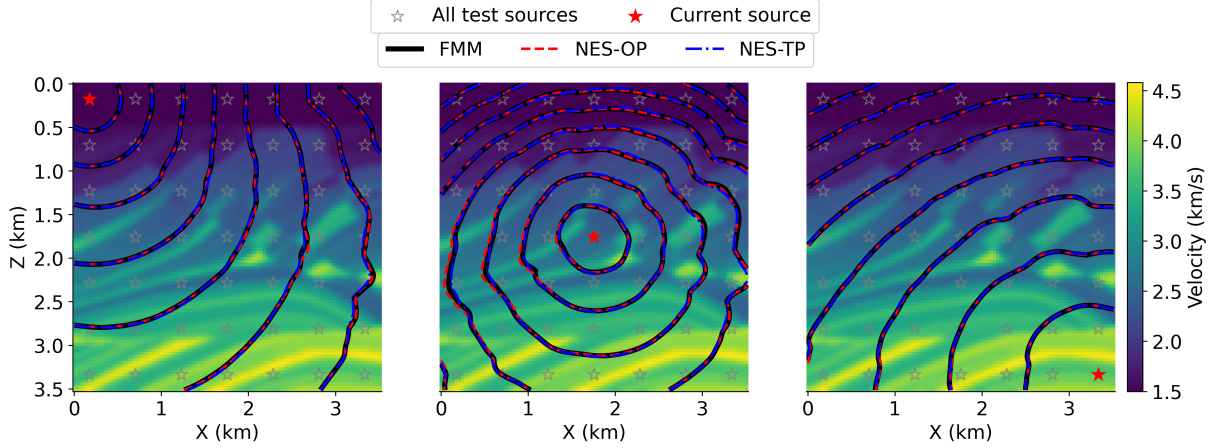


Figure 5: The solutions of NES-OP and NES-TP comparatively to the reference solution. Red star indicates the current source location, the gray star show all considered source locations.

points are always excluded from the training set due to the singularity. Overall, the average RMAE over 49 sources was about 0.5% and 0.8% for the NES-OP and the NES-TP respectively.

In addition, we tested the convergence of the NES-OP and the NES-TP depending on the sparsity of the receiver points in the training set and the results are shown in Figure 6. From the Figure 6 we can conclude that the difference between the NES-OP and the NES-TP is insignificant but systematic. The NES-TP provides slightly higher errors but more stable solution (min-max band is narrower). One can notice that the RMAE for both almost stopped decreasing for the grid sizes bigger than  $41 \times 41$ . That can be due to the fact that the number of the training epochs, the number of trainable weights, and the sampling of collocation points remained fixed throughout all the experiments, while the information about the velocity model was increasing. This problem of weak convergence is known for PINNs and can be partially resolved using proper techniques for the sampling of training points [14], but we did not consider this aspect in our paper.

The summary of the performance with the sizes of networks and elapsed time for the training is presented in Table 1. One can notice that the NES-TP is the most attractive as it has the fewest number of trainable weights, require the acceptable training time and provides the  $\text{RMAE} < 1\%$ . The NES-OP could be used to get a lower error for one specific source. Comparing the NES-TP with the FMM, one can notice that the NES-TP inference time (after training) is comparable with the FMM, and the NES-TP size in memory is significantly lower than the FMM. The FMM size will grow linearly with the increasing grid size, whereas the NES-TP will grow much slower (if we decide to increase number of hidden layers or units). These tests were performed with the fixed complexity of the NES-OP and the NES-TP, but to get higher accuracy more epochs of training and more hidden layers and units in networks can be used. Overall, the NES-TP can be a good alternative to the FMM for the problems with many sources as it gives the compact representation with good accuracy. In addition, although the training time will grow in 3D models, the inference time will not grow as fast as the time of FMM. Our preliminary tests showed that in 3D models, the NES-TP provides 4-5 times faster inference time.

### 3.3 Comparison with EikoNet and PINNeik

In this section, we give a comparison with the existing solutions for solving the eikonal equation: PINNeik [21] and EikoNet [19]. We do separate comparison because PINNeik works for the one-point eikonal, but the EikoNet uses two-point eikonal. For these tests, we used the same Marmousi model from Figure 5. In these tests, we used the different number of hidden layers and the training set sizes to show a possible variability in the performance. The results of the comparisons are shown in Table 2.

The PINNeik uses the conventional factorization (eq. 4) and the loss function based on the  $L_2$ -norm which includes the equation, the positivity and the boundary conditions. The PINNeik architecture consists of 10 hidden layers with 20 units on each, and locally-adaptive arctangent activation [13]. The NES-OP architecture contained 4 hidden layer with 50 units on each layer. The total number of collocation points for training is  $300 \times 281$  evenly spaced in the domain. The source is positioned at the center of the model as it is shown in central panel in Figure 5. Number of epochs was

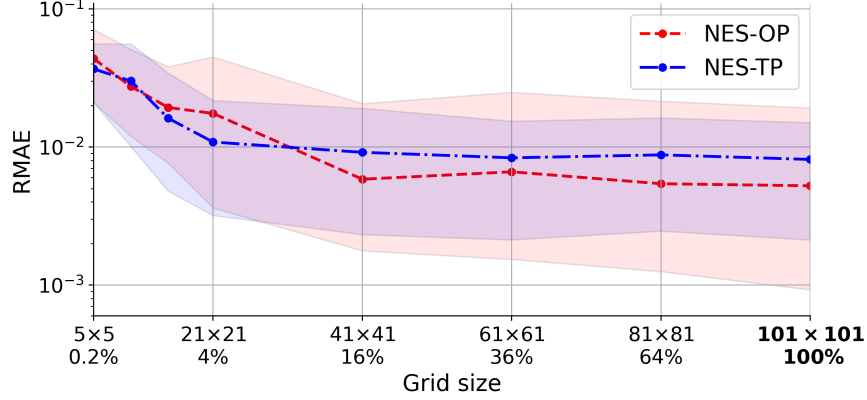


Figure 6: The convergence of the NES-OP and the NES-TP in the Marmousi model. The X-axis indicates the grid size of the receiver points for each source location and the percentage from the original grid size highlighted by bold font, and on which the RMAE was measured. The lines show the average RMAE over all 49 sources, the transparent areas indicate the min-max band over 49 sources.

Table 1: The summary of the performance of the NES-OP and the NES-TP in the Marmousi model for the considered 49 sources. The data are provided for the the grid sizes  $41 \times 41$  and  $101 \times 101$  in the first and the second rows for the NES-OP and the NES-TP respectively. These grid sizes are chosen as the lower and the upper bound when the RMAE almost stopped decreasing (see Figure 6). These tests were performed on GPU RTX 3070 Laptop. Batch size is always 1/4 of the total number of training points. Inference time column indicates the run-time on a given set of points once a network is trained. Size column indicates the number of the trainable weights for the NES-OP and the NES-TP, and the grid size for the FMM. Note, that the data for the NES-OP contain 49 separate networks.

Solver	RMAE, %	Elapsed time for 3000 epochs, (s)	Inference time, (s)	Training points	Size
<b>NES-OP</b>	0.6	2500	0.8	82320	852894
	0.5	3000	1	499800	
<b>NES-TP</b>	0.9	200	0.07	82320	17558
	0.8	960	0.3	499800	
FMM	-	-	0.2	-	499849

5000. We made 5 independent launches for both solvers to see the average result. From the Table 2 one can notice that the NES-OP is faster and significantly outperforms PINNeik in accuracy (60 times lower RMAE).

The EikoNet also uses the conventional factorization but for the two-point eikonal (eq. 5). The EikoNet loss function based on  $L_2$ -norm and is written for the difference between the predicted and true velocities with respect to the receiver-part equation only (based on the first equation in 2) and without the reciprocity. The EikoNet uses deep architecture with ten residual blocks and *elu* activation. The NES-TP contained 6 hidden layers with 100 units on each. For the training set, we used random 250000 source-receiver pairs. With that, EikoNet utilized random distance sampling, whereas NES-TP utilized random uniform distribution in the domain. We made five independent launches with 1000 epochs for both solvers. The errors were estimated on the three test sources with positions as shown in Figure 5. From the Table 2 one can notice that the NES-TP provides: 13 times lower error, 32 times faster training, 154 times more compact architecture.

## 4 Discussion

We presented the Neural Eikonal Solver for the one-point and the two-point eikonal equations. We discussed the crucial aspects of the proposed NES architecture: the improved factorization, the loss function, the activation function, the reciprocity principle, the type of a distribution for the random weights initialization, the input data scaling, and the type of connection between the hidden layers. These helped to enhance significantly the performance of the NES. Some aspects still remain unresolved, which we plan to consider as a future development.

Table 2: Comparison of NES-OP with PINNeik and NES-TP with EikoNet. RMAE indicates the average for five independent launches. Training time shows the average time: for 5000 epochs of NES-OP and PINNeik; for 1000 epochs of NES-TP and EikoNet. These tests were performed on GPU Tesla P100-PCIE.

Solver	RMAE, %	Elapsed time, (s)	Size
<b>NES-OP</b>	<b>0.2</b>	<b>240</b>	<b>7856</b>
PINNeik	12.4	330	4061
<b>NES-TP</b>	<b>0.4</b>	<b>300</b>	<b>51308</b>
EikoNet	5.4	9600	7913249

#### 4.1 Future developments

The sampling of the collocation points for the training is very important for the accuracy and the convergence. We considered only the influence of regular grids on the accuracy (see Figure 6), but thorough investigation is needed to better understand what sparsity of the training set is enough to get the highest performance of the NES. Our preliminary tests showed no visible and systematic improvements for the following techniques: gradients based sampling (more points where gradient of velocity model is high), importance-based sampling [18], and weighted sampling [19]. However, the Residual-based Adaptive Refinement (RAR) [17] (adding points where needed) showed promising results by providing better accuracy with lower number of collocation points.

The neural-network architecture is important for the performance. Preliminary, we found that the total number of trainable weights between  $10^3$ - $10^4$  provides good accuracy, but we did not consider how exactly the number of trainable weights increases with the complexity of the velocity model. In our convergence test on the Marmousi model (see Figure 6), we used the fixed architectures which could be a reason why the errors almost stopped decreasing after a certain grid size. Therefore, a deeper investigation is needed to understand how the neural network depth should grow with the increasing information about the velocity model heterogeneities.

It has been shown that the pre-trained neural network in one velocity model can be used for another model to provide a faster convergence [19, 21]. The transfer learning can be beneficial for reducing the training time, but we did not consider this problem in our work.

In our paper we mitigated the challenges related to the caustics using a non-symmetric loss function with the Hamiltonian form of the eikonal with  $p = 2$ . But, usually these challenges are tackled using the viscosity solution which provides the stability and uniqueness even for non-differentiable solutions [42]. The trick is to use the additional viscosity term in the eikonal equation  $-\varepsilon\Delta\tau$  (laplacian) by vanishing the degree of viscosity  $\varepsilon \rightarrow 0$ . The numerical algorithms, such as the FMM, do not solve the extended equation with viscosity term, but utilize the discrete approximation of the viscosity solution [43]. We tested the influence of the viscosity term  $-\varepsilon\Delta\tau$  on the NES training dynamic, but it only decreased the accuracy. The future investigations of the sophisticated neural-network architectures representing the viscosity solution [44] may be helpful for the eikonal equation.

#### 4.2 Potential applications

The scope of our work focused on the isotropic eikonal equation only, but the NES can be extended to the various anisotropic eikonal equations in a similar way as it was shown for the PINNeik [21]. As well, the massive computations of traveltimes for large number of source-receiver pairs (look-up tables) require a lot of memory for storage, whereas the NES-TP solution has the fixed size defined by the complexity of the neural-network architecture thereby providing drastic compression opportunities. The compression of the look-up tables becomes very useful in the Kirchhoff migration procedures [45], which may require large tables (several Tb), while the neural network takes hundreds Kb, which was shown on the example of the approximation of the eikonal solution [26, 20]. Other possible applications for the NES-TP: traveltimes simulation of the reflected waves according to the Huygens principle; the ray multipathing analysis [3, 19]; the earthquake localization [22, 23]; the traveltimes tomography [24, 25].

## 5 Conclusions

In this paper, we introduced the Neural Eikonal Solver framework for solving the one-point and the two-point eikonal equation using the PINN concept. We suggested several novelties to the neural-network architecture to improve its accuracy and to speed up training. We proposed the improved factorization of the eikonal equation to speed up the training by constraining the NES between the fastest and the slowest solutions. To address the intrinsic pathologies related to the caustic singularities, the NES involves a non-symmetric loss function based on the  $L_1$ -norm and a

Hamiltonian form of the eikonal equation, and uses the gaussian activation function. To account for the reciprocity principle we proposed a symmetrization that provides a strict observance of the reciprocity. We also suggested the random weight initialization based on *he\_normal* distribution and the scaling of the input data by a maximum value. Almost all of the proposed features require no additional computational cost. This resulted in the improvement of the accuracy of traveltime computations by the two orders of magnitude.

The benchmark testing conducted on a simple model with caustics showed that the NES reaches the relative mean absolute error of about 0.07-0.16% from the second-order factored FMM. For complicated Marmousi model the NES reaches error  $< 0.5\%$  after several minutes of training. The test showed that the NES significantly outperforms existing PINN solutions to the eikonal equation showing 10-60 times smaller error and 2-30 times faster training. The NES-OP provides higher accuracy than the NES-TP for one specific source. The NES-TP provides slightly lower accuracy than the NES-OP but gives a compact representation of traveltimes for arbitrary source-receiver pairs which is perspective for applications in seismic exploration. In this paper, we showed 2D examples for the simplicity but extension to 3D is straightforward with the accuracy and the performance remaining the same.

## Acknowledgement

NES package with examples and tutorials is available at <https://github.com/sgrubas/NES>.

## References

- [1] James Albert Sethian. *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*, volume 3. Cambridge university press, 1999.
- [2] Rémi Abgrall and Jean-David Benamou. Big ray-tracing and eikonal solver on unstructured grids: Application to the computation of a multivalued traveltime field in the marmousi model. *Geophysics*, 64(1):230–239, 1999.
- [3] Nicholas Rawlinson, Malcolm Sambridge, and Jurg Hauser. Multipathing, reciprocal traveltime fields and raylets. *Geophysical Journal International*, 181(2):1077–1092, 2010.
- [4] Gerard T Schuster and Aksel Quintus-Bosz. Wavepath eikonal traveltime inversion: Theory. *Geophysics*, 58(9):1314–1323, 1993.
- [5] Samuel H Gray and William P May. Kirchhoff migration using eikonal equation traveltimes. *Geophysics*, 59(5):810–817, 1994.
- [6] James A Sethian. Fast marching methods. *SIAM review*, 41(2):199–235, 1999.
- [7] Eran Treister and Eldad Haber. A fast marching algorithm for the factored eikonal equation. *Journal of Computational physics*, 324:210–225, 2016.
- [8] Hongkai Zhao. A fast sweeping method for eikonal equations. *Mathematics of computation*, 74(250):603–627, 2005.
- [9] Sergey Fomel, Songting Luo, and Hongkai Zhao. Fast sweeping method for the factored eikonal equation. *Journal of Computational Physics*, 228(17):6440–6455, 2009.
- [10] Justin Sirignano and Konstantinos Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375:1339–1364, 2018.
- [11] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. 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, 2019.
- [12] Ameya D Jagtap, Kenji Kawaguchi, and George Em Karniadakis. Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. *Journal of Computational Physics*, 404:109136, 2020.
- [13] Ameya D Jagtap, Kenji Kawaguchi, and George Em Karniadakis. Locally adaptive activation functions with slope recovery for deep and physics-informed neural networks. *Proceedings of the Royal Society A*, 476(2239):20200334, 2020.
- [14] Jeremy Yu, Lu Lu, Xuhui Meng, and George Em Karniadakis. Gradient-enhanced physics-informed neural networks for forward and inverse pde problems. *Computer Methods in Applied Mechanics and Engineering*, 393:114823, 2022.
- [15] Rafael Bischof and Michael Kraus. Multi-objective loss balancing for physics-informed deep learning. *arXiv preprint arXiv:2110.09813*, 2021.

- [16] Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, 2021.
- [17] Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. Deepxde: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228, 2021.
- [18] Mohammad Amin Nabian, Rini Jasmine Gladstone, and Hadi Meidani. Efficient training of physics-informed neural networks via importance sampling. *Computer-Aided Civil and Infrastructure Engineering*, 2021.
- [19] Jonathan D Smith, Kamyar Azizzadenesheli, and Zachary E Ross. Eikonet: Solving the eikonal equation with deep neural networks. *IEEE Transactions on Geoscience and Remote Sensing*, 2020.
- [20] Serafim Grubas. Seismic-wave traveltime computation by supervised and unsupervised training of artificial neural networks. In *SPE Annual Technical Conference and Exhibition*. OnePetro, 2020.
- [21] Umair bin Waheed, Ehsan Haghighat, Tariq Alkhalifah, Chao Song, and Qi Hao. Pinneik: Eikonal solution using physics-informed neural networks. *Computers & Geosciences*, page 104833, 2021.
- [22] Jonathan D Smith, Zachary E Ross, Kamyar Azizzadenesheli, and Jack B Muir. Hyposvi: Hypocenter inversion with stein variational inference and physics informed neural networks. *arXiv*, 2021.
- [23] S Grubas, S Yaskevich, and A Duchkov. Localization of microseismic events using the physics-informed neural-network for traveltime computation. In *82nd EAGE Annual Conference & Exhibition*, volume 2021, pages 1–5. European Association of Geoscientists & Engineers, 2021.
- [24] Rômulo M Silva and Alvaro LGA Coutinho. Physics-informed neural networks for the factored eikonal equation. 2020.
- [25] Umair bin Waheed, Tariq Alkhalifah, Ehsan Haghighat, Chao Song, and Jean Virieux. Pinntomo: Seismic tomography using physics-informed neural networks. *arXiv preprint arXiv:2104.01588*, 2021.
- [26] Serafim I Grubas, Georgy N Loginov, and Anton A Duchkov. Traveltime-table compression using artificial neural networks for kirchhoff-migration processing of microseismic data. *Geophysics*, 85(5):U121–U128, 2020.
- [27] Vlastislav Cervený. *Seismic ray theory*. Cambridge university press Cambridge, 2001.
- [28] Jon F Claerbout. *Imaging the Earth’s interior*. Blackwell scientific publications Oxford, 1985.
- [29] S.V. Goldin. *Seismic Traveltime Inversion*, volume 1. SEG: Tulsa, 1986.
- [30] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.
- [31] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [32] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*, 2019.
- [33] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283, 2016.
- [34] Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *Journal of machine learning research*, 18, 2018.
- [35] Peter J Rousseeuw and Annick M Leroy. *Robust regression and outlier detection*, volume 589. John wiley & sons, 2005.
- [36] Jooyoung Park and Irwin W Sandberg. Universal approximation using radial-basis-function networks. *Neural computation*, 3(2):246–257, 1991.
- [37] Tom O’Malley, Elie Bursztein, James Long, François Chollet, Haifeng Jin, Luca Invernizzi, et al. Kerastuner. <https://github.com/keras-team/keras-tuner>, 2019.
- [38] François Chollet et al. Keras. <https://keras.io>, 2015.
- [39] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.
- [40] Furong Huang, Jordan Ash, John Langford, and Robert Schapire. Learning deep resnet blocks sequentially using boosting theory. In *International Conference on Machine Learning*, pages 2058–2067. PMLR, 2018.
- [41] Kevin Ganster. Eikonal fast marching. [https://github.com/kevinganster/eikonal\\_fm](https://github.com/kevinganster/eikonal_fm), 2020.

- [42] Guy Barles. An introduction to the theory of viscosity solutions for first-order hamilton–jacobi equations and applications. In *Hamilton-Jacobi equations: approximations, numerical analysis and applications*, pages 49–109. Springer, 2013.
- [43] Régis Monneau. Introduction to the fast marching method. 2010.
- [44] Jérôme Darbon and Tingwei Meng. On some neural network architectures that can represent viscosity solutions of certain high dimensional hamilton–jacobi partial differential equations. *Journal of Computational Physics*, 425:109907, 2021.
- [45] Tariq Alkhalifah. Efficient traveltimes compression for 3d prestack kirchhoff migration. *Geophysical Prospecting*, 59(1):1–9, 2011.