Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

ARTICLE TYPE

Error Control and Loss Functions for the Deep Learning Inversion of Borehole Resistivity Measurements

M. Shahriari[1,2] | D. Pardo[3,4,5] | J. A. Rivera*[3,4] | C. Torres-Verdín[6] | A. Picon[7,3] | J. Del Ser[7,3,4] | S. Ossandón[8] | V. M. Calo[9]

[1]Software Competence Center Hagenberg GmbH (SCCH), Hagenberg, Austria
[2]Euskampus Fundazioa, Bilbao, Spain
[3]University of the Basque Country (UPV/EHU), Leioa, Spain
[4] Basque Center for Applied Mathematics (BCAM), Bilbao, Spain
[5]Ikerbasque (Basque Foundation for Sciences), Bilbao, Spain
[6]The University of Texas at Austin, USA
[7]Tecnalia, Basque Research & Technology Alliance, Derio, Spain
[8]Pontificia Universidad Católica de Valparaíso, Valparaíso, Chile
[9]Curtin University, Perth, Australia, Perth, Australia

Correspondence
*J. A. Rivera. Email: riverajonander@gmail.com

Summary

Deep learning (DL) is a numerical method that approximates functions. Recently, its use has become attractive for the simulation and inversion of multiple problems in computational mechanics, including the inversion of borehole logging measurements for oil and gas applications. In this context, DL methods exhibit two key attractive features: a) once trained, they enable to solve an inverse problem in a fraction of a second, which is convenient for borehole geosteering operations as well as in other real-time inversion applications. b) DL methods exhibit a superior capability for approximating highly-complex functions across different areas of knowledge. Nevertheless, as it occurs with most numerical methods, DL also relies on expert design decisions that are problem specific to achieve reliable and robust results. Herein, we investigate two key aspects of deep neural networks (DNNs) when applied to the inversion of borehole resistivity measurements: error control and adequate selection of the loss function. As we illustrate via theoretical considerations and extensive numerical experiments, these interrelated aspects are critical to recover accurate inversion results.

KEYWORDS:
error estimation, geophysical applications, real-time inversion, deep learning, deep neural networks

1 | INTRODUCTION

The number of research articles and industrial applications of Deep Learning algorithms have rapidly grown in the last decade due to their high performance in different applications such as computer vision[1], speech recognition[2], and biometrics[3], to mention a few. In recent years, there have been significant advances in the field of DL, with the appearance of residual neural networks (RNNs)[4], which prevent gradient degeneration during the training stage, and Encoder-Decoder (sequence-to-sequence) deep neural networks (DNNs), which have improved the DL work capability in computer vision applications[5]. Due to the high demand for DNNs from industry, dedicated libraries and packages such as Tensorflow[1], Keras[6], and Pytorch[7] have been developed. These libraries facilitate the use of DNNs across different industrial applications[8,9,10,11,12]. All these advances combined make DNNs one of the most powerful and fast-growing artificial intelligence (AI) tools presently.

In this work, we focus on the application of DNNs to geosteering operations[13,14,15]. In this oil & gas application, a logging-while-drilling (LWD) instrument records electromagnetic measurements, which are inverted in real time to produce a map of the Earth's subsurface. Based on the reconstructed Earth model, the operator adjusts the well-trajectory in real time to further

explore exploitation targets, including oil & gas reservoirs, and to maximize the posterior productivity of the available reserves. Due to the tremendous productivity increase achieved with this technique, nowadays geosteering plays an essential role in the oil & gas industry [16].

The main difficulty one faces when dealing with geosteering problems is the real-time adjustment of the well trajectory. For that, we require a rapid inversion technique. Unfortunately, traditional inversion methods have severe limitations, which force geophysicists to continuously look for new solutions to this problem (see, e.g., [17,18,14,19,20,21,13,15]). Gradient-based methods require simulating the forward problem dozens of times for each set of measurements. Moreover, these methods also estimate the derivatives of the measurements with respect to the inversion variables, which is challenging and time consuming [22]. To alleviate the high computational costs associated with this inversion method, simplified 1.5-dimensional (1.5D) methods are common (see, e.g., [23,14,15]). For the inversion of borehole resistivity measurements, an alternative is to apply statistics-based methods [24,25,26]. The statistical methods perform forward simulations hundreds of times, which also require large computation times [27]. Both gradient and statistics-based methods *only evaluate* the inverse operator. Thus, the entire inversion process is repeated at each new logging position.

Below, we employ DNNs to approximate the inverse operator. Although the training stage of a DNN may be time consuming, after the network is properly trained, it can forecast in a fraction of a second [13]. This rapid inversion facilitates geosteering operations.

DNNs also face important challenges when applied to the inversion of borehole resistivity problems. In particular, to properly train a DNN, we require a large dataset (also known as *ground truth*) with the solution of the forward problem for different Earth models [28,13,29]. Building a dataset may be time consuming, especially for two and three-dimensional problems. In those cases, we need to solve the forward problem using numerical simulation methods such as the finite element (FEM) [23,30,31] or finite difference (FDM) [32,33]. Moreover, we need to optimally sample the parameter space describing relevant Earth models. Additionally, the training stage can be time consuming. However, this is an *offline* cost. One additional challenge arises due to the nature of inverse problems: they are not well-defined, that is, there may exist multiple outputs for a given input [22,27]. As we shall illustrate in this work, when using a DNN equipped with a traditional loss function based on the data misfit, the corresponding DNN approximations may be far away from any of the existing solutions to the inverse operator. This can seriously compromise the reliability of the method and, consequently, the corresponding decision-making during geosteering operations.

In this work, we investigate the selection of the loss function to train a DNN when dealing with an inverse problem. We also introduce some error control during training. We focus on the inversion of borehole resistivity measurements. Nonetheless, most of the design decisions of such loss function are applicable to other inverse problems. To explain the main results stemming from this work, we first illustrate them with a simple mathematical example. Then, we apply the resulting DNN approximations to synthetic examples, which help us elucidate their main advantages and limitations. This work does not discuss optimal data sampling techniques nor the decision-making for the optimal selection of DNN architectures [34,35]. Those subjects are possible future work. However, for this article to be self-contained, we briefly describe in the appendix the architecture of the DNN we use.

The remainder of this article is organized as follows. Section 2 states the problem formulation and introduces two examples. In the first one, the exact solution of the inverse problem is the square root of the input data. This example serves to illustrate some of the main features and limitations of DL algorithms. The second example reproduces a realistic inversion scenario of borehole resistivity measurements. We also describe the selected parameterization of the Earth models and the borehole measurement acquisition system, including the employed logging instruments and recorded measurements. Section 3 describes the finite-dimensional input and output vector representations of the inverse operator that we approximate via DL. This section also discusses how we generate the ground truth dataset. Section 4 proposes a preprocessing of the input and output data variables to ensure that contributions to the loss (cost) function corresponding to different measurements and Earth parameters are comparable in magnitude. Section 5 describes the vector and matrix norms employed in this work, along with the corresponding absolute and relative errors. Section 6 analyzes various loss functions and illustrates their most prominent advantages and limitations. Section 7 describes the main implementation aspects of our DL inversion algorithm. We present several numerical inversion results of borehole logging measurements in Section 8. In addition to some conclusions and future work we describe in Section 9, the manuscript also contains two appendices. Appendix A details the selected DNN architectures.

2 | PROBLEM FORMULATION

2.1 | Forward problem

We fix the measurement acquisition system $\tilde{\mathbf{s}}$. Then, for a well trajectory $\tilde{\mathbf{t}}$, and an Earth model $\tilde{\mathbf{p}}$, the forward problem consists of finding the corresponding borehole resistivity measurements $\tilde{\mathbf{m}}$. We denote by $\tilde{\mathcal{F}}$ the associated forward function. That is:

$$\tilde{\mathcal{F}}(\tilde{\mathbf{t}}, \tilde{\mathbf{p}}) = \tilde{\mathbf{m}}, \qquad \text{where } \tilde{\mathbf{t}} \in \tilde{\mathbb{T}}, \tilde{\mathbf{p}} \in \tilde{\mathbb{P}}, \tilde{\mathbf{m}} \in \tilde{\mathbb{M}}. \tag{1}$$

In the above, we omit for convenience the explicit dependence of the function $\tilde{\mathcal{F}}$ upon the fixed input variable $\tilde{\mathbf{s}}$. $\tilde{\mathbb{P}} = \{\tilde{\mathbf{p}} = \tilde{\mathbf{p}}(x, y, z) \in \mathbb{R}^{3x3} : \forall (x, y, z) \in \Omega \subset \mathbb{R}^3\}$ – the set of all possible resistivity tensors – and $\tilde{\mathbb{M}} = \{\tilde{m} \in \mathbb{R}^m$, being m the number of measurements$\}$ – the set of all possible measurements – are normed vector spaces equipped with norms $||\cdot||_{\tilde{\mathbb{P}}}$ and $||\cdot||_{\tilde{\mathbb{M}}}$, respectively. $\tilde{\mathbb{T}} = \{\mathbf{t} = \mathbf{t}(s) : \mathbf{t}(s) \in \Omega \; \forall s \in (a, b) \subset \mathbb{R}\}$ – the set of all possible logging trajectories – is also a vector space. Function $\tilde{\mathcal{F}}$ consists of a boundary value problem governed by Maxwell's equations (see[23] for details).

2.2 | Inverse problem

In the inversion of borehole resistivity measurements, the objective is to determine the subsurface properties $\tilde{\mathbf{p}}$ corresponding to a set of measurements $\tilde{\mathbf{m}}$ recorded over a given trajectory $\tilde{\mathbf{t}}$. Again, the measurement acquisition system $\tilde{\mathbf{s}}$ is fixed. We denote that inverse operator as $\tilde{\mathcal{I}}$ (inverse of $\tilde{\mathcal{F}}$). Mathematically, we have:

$$\tilde{\mathcal{I}}(\tilde{\mathbf{t}}, \tilde{\mathbf{m}}) = \tilde{\mathbf{p}}, \qquad \text{where } \tilde{\mathbf{t}} \in \tilde{\mathbb{T}}, \tilde{\mathbf{m}} \in \tilde{\mathbb{M}}, \; \tilde{\mathbf{p}} \in \tilde{\mathbb{P}}. \tag{2}$$

Again, we omit for convenience the explicit dependence of function $\tilde{\mathcal{I}}$ upon input variable $\tilde{\mathbf{s}}$. The governing physical equation of operator $\tilde{\mathcal{I}}$ is unknown. However, we know that a given input may have multiple associated outputs. Thus, such inverse operator is not well-defined.

2.3 | Parameterization

We select a finite dimensional subspace of $\tilde{\mathbb{T}}$ parameterized with n_t real-valued numbers. The corresponding vector representation of an element from that subspace is $\mathbf{t} \in \mathbb{R}^{n_t}$. We similarly parameterize a finite dimensional subspace of $\tilde{\mathbb{P}}$ and $\tilde{\mathbb{M}}$ with n_p and n_m real-valued numbers, respectively. The corresponding vector representations of an element from those subspaces are $\mathbf{p} \in \mathbb{R}^{n_p}$ and $\mathbf{m} \in \mathbb{R}^{n_m}$, respectively.

The span of vector representations $\mathbf{p}$ and $\mathbf{m}$ constitute two subspaces of $\mathbb{R}^{n_p}$ and $\mathbb{R}^{n_m}$ with norms $||\cdot||_{\mathbb{P}}$ and $||\cdot||_{\mathbb{M}}$, respectively. Ideally, these norms should be inherited from those associated with the original infinite dimensional spaces. However, this is often a challenging task and an open area of research. We directly employ some existing (typically l_1 or l_2) finite dimensional norms.

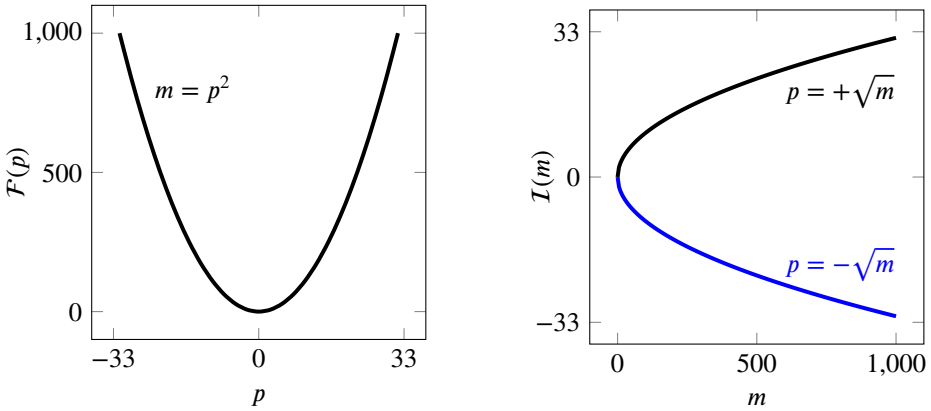
The function $\mathcal{F}$ associates a pair $(\mathbf{t}, \mathbf{p})$ (vector representations of $(\tilde{\mathbf{t}},\tilde{\mathbf{p}})$) with $\mathbf{m}$ (vector representation of $\tilde{\mathbf{m}}$) such that $\mathcal{F}(\mathbf{t}, \mathbf{p}) = \mathbf{m}$. We employ a similar notation for its inverse $\mathcal{I}$ acting on vector representations.

To provide context and guidance for future developments, we introduce simple examples that illustrate some of the shortcomings of the standard techniques when applied to these problems, and we explain how we seek to overcome the associated challenges. The first problem seeks to predict the inverse of squaring a number. The second example focuses on geosteering applications.

2.4 | Example A: Model problem with known analytical solution

We select $n_t = 0$, $n_p = n_m = 1$. The forward function is given by $\mathcal{F}(p) = p^2$, while the inverse has two solutions (branches): $\mathcal{I}(m) = +\sqrt{m}$, and $\mathcal{I}(m) = -\sqrt{m}$, as described in Figure 1.

This simple example contains a key feature exhibited by most inverse problems: it has multiple solutions. Thus, it illustrates the behaviour of DNNs when considering different loss functions. Results are enlightening and, as we show below, they provide clear guidelines to construct proper loss functions for approximating inverse problems.



(a) Forward function

(b) Inverse operator with two branches

FIGURE 1 Model problem with known analytical solution.

2.5 | Example B: Inversion of borehole resistivity measurements

In geosteering applications, multiple oil and service companies perform inversion assuming a piecewise 1D layered model of the Earth. In this case, there exist semi-analytic methods that can simulate the forward problem in a fraction of a second. Herein, we use the same approach. Thus, the evaluation of $\mathcal{F}$ is performed with a 1.5D semi-analytic code (see[23,36]). As a result, at each logging position, our inversion operator recovers the formation properties of a 1D layered medium[14,15].

In this work, as measurement acquistion system, we first consider a co-axial LWD instrument equipped with two transmitters and two receivers (see Figure 2). H^1_{zz} and H^2_{zz} are the zz-couplings of the magnetic field measured at the first and the second receivers, respectively (the first and second subscripts denote the orientation of the transmitter and receiver, respectively). Then, we define the attenuation and phase difference as follows:

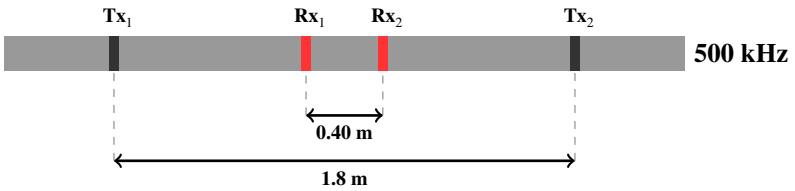


FIGURE 2 Conventional LWD logging instrument. Tx_i and Rx_i are the transmitters and the receivers, respectively.

$$\ln \frac{H^1_{zz}}{H^2_{zz}} = \underbrace{\ln \frac{|\,H^1_{zz}\,|}{|\,H^2_{zz}\,|}}_{\times 20\log(e) =: \text{attenuation } (dB)} + i \underbrace{\left(ph(H^1_{zz}) - ph(H^2_{zz})\right)}_{\times \frac{180}{\pi} =: \text{phase difference (degree)}}, \tag{3}$$

where *ph* denotes the phase of a complex number. We then record the average of the attenuations and phase differences associated with the two transmitters, and we denote these values as *LWD coaxial*.

Then, we consider a short-spacing configuration corresponding to a deep azimuthal instrument equipped with one transmitter and one receiver, as shown in Figure 3. In this logging instrument, the distance between transmitter and receiver is significantly larger than that of the previously considered LWD instrument. It also employs tilted receivers that are sensitive to the presence of bed boundaries. We record several measurements with this logging instrument: (a) the attenuation and phase differences, denotes as *deep coaxial*, computed using Equation (3) with $H^2_{zz} = 1$, and (b) the attenuation and phase differences of a directional

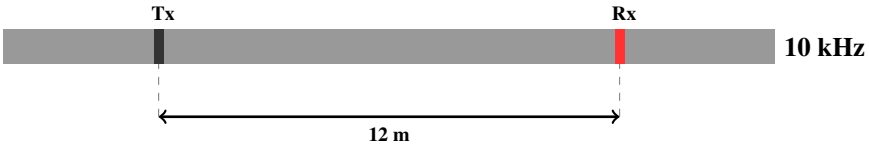


FIGURE 3 Short-spacing of a deep azimuthal logging instrument. Tx and Rx are the transmitter and the receiver, respectively.

measurement expressed as:

$$Geosignal = \ln \frac{H_{zz} - H_{zx}}{H_{zz} + H_{zx}} = \underbrace{\ln \frac{|\, H_{zz} - H_{zx} \,|}{|\, H_{zz} + H_{zx} \,|}}_{\times 20 \log(e) =: \text{attenuation } (dB)} + i \underbrace{\left(ph(H_{zz} - H_{zx}) - ph(H_{zz} + H_{zx}) \right)}_{\times \frac{180}{\pi} =: \text{phase difference (degree)}}. \tag{4}$$

We denote it as *geosignal*. These measurements exhibit a discontinuity as a function of the dip angle at 90 degrees. Indeed, such discontinuity is essential in the measurements if one wants to discern between top and bottom of the logging instrument (see Figure 4).

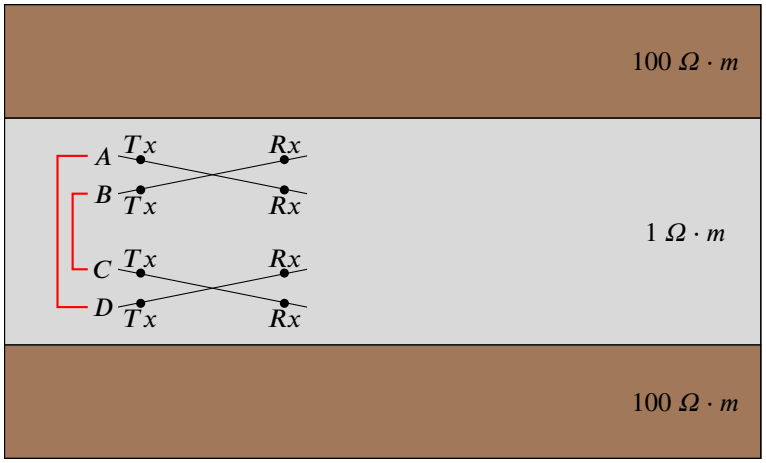


FIGURE 4 Illustration with four logging trajectories. By symmetry, measurements recorded with trajectories A and D are identical. The same occurs with trajectories B and C. If these measurements are continuous with respect to the dip angle, then at 90 degrees they all become identical, which disables the possibility of identifying if a nearby bed boundary is located on top or on the bottom of the logging instrument.

For our borehole resistivity applications, we consider a zero-thickness borehole embedded in a three-layer medium (see Figure 5). A common practice in the field is to characterize this medium with seven parameters, as described in Figure 5. In this work, to simplify the problem, we consider only five of them by restricting the search to isotropic formations ($\rho_v = \rho_h$) with zero dip angle ($\beta = 0$), as illustrated in Figure 6. Thus, $n_p = 5$.

In this example, we consider two cases (see Figure 6) according with different numbers of logging positions per data sample.

2.5.1 | Example B.1: one logging position

In this case, each trajectory consists of a single logging position. Therefore, for each sample, we record six real numbers (three attenuations and three phases), i.e., $n_m = 6$. At each logging position, the trajectory is described by one number: the trajectory dip angle. Thus, $n_t = 1$.

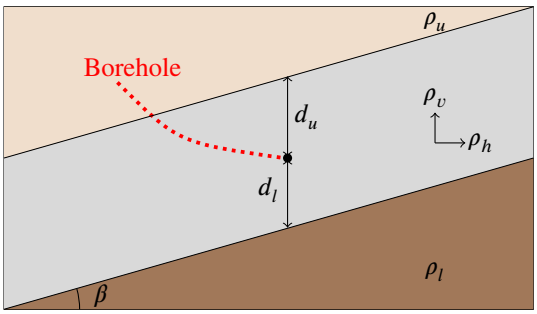
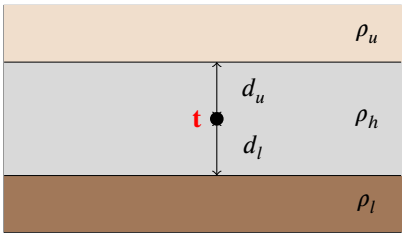
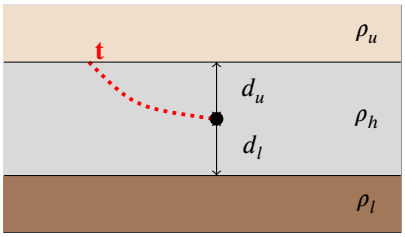


FIGURE 5 Well trajectory in a 1D medium. The black circle indicates the last trajectory position. ρ_h and ρ_v are the horizontal and vertical resistivities of the host layer where the final logging position is located, respectively. ρ_u and ρ_l are the resistivity values of the upper and lower layers to the host layer, respectively. d_u and d_l show the distance from the final logging position to the upper and lower bed boundaries, respectively.



(a) Example B.1: trajectory with 1 logging positions



(b) Example B.2: trajectory with 65 logging positions

FIGURE 6 Model problems corresponding to examples B.1 and B.2, respectively.

2.5.2 | Example B.2: sixty-five logging positions

In this case, the logging trajectory of each sample is formed by 65 logging positions with a logging step size of 0.3048 m (see [13,29] for further details). Thus, for each Earth model $\mathbf{p}$, we parametrize $\mathbf{m}$ with $6 \times 65 = 390$ real numbers ($n_m = 390$). For this example, we assume that the variation of the dip angle at a given logging position with respect to the previous one is constant. We denote that constant dip angle variation as α_v. Then, at the i-th logging position, the trajectory dip angle is $\alpha_i = \alpha_{ini} + (i-1)\alpha_v$, where α_{ini} is the initial dip angle. Hence, we have $n_t = 2$.

3 | DATA SPACE AND GROUND TRUTH

In this work, we employ a deep neural network (DNN) to approximate the discrete inverse operator $\mathcal{I}$. Given a supervised database of n-pairs $(\mathbf{m}_i, \mathcal{I}(\mathbf{t}_i, \mathbf{m}_i))$, $i = 1, ..., n$, the DNN builds an approximation of the unknown function $\mathcal{I}$. This section describes the construction of the supervised database.

We first select the number of samples, n, and two subspaces of $\mathbb{R}^{n_p}$ and $\mathbb{R}^{n_t}$, respectively. Then, we select the n samples in those subspaces, namely, $((\mathbf{t}_1, \mathbf{p}_1), ..., (\mathbf{t}_n, \mathbf{p}_n))$. To each of these samples, we apply the operator $\mathcal{F}$. That is, we compute $(\mathcal{F}(\mathbf{t}_1, \mathbf{p}_1), ..., \mathcal{F}(\mathbf{t}_n, \mathbf{p}_n))$. Finally, the n-pairs $(\mathbf{m}_i, \mathcal{I}(\mathbf{t}_i, \mathbf{m}_i)) := (\mathcal{F}(\mathbf{t}_i, \mathbf{p}_i), \mathbf{p}_i)$, $i = 1, ..., n$ form our supervised database.

We denote by $\mathbf{T} \in \mathbb{R}^{n_t \times n}$ to the set of all trajectory samples $(\mathbf{t}_1, ..., \mathbf{t}_n)$. In other words, $\mathbf{T}$ is a matrix with $\mathbf{t}_i$ being its i-th column. Similarly, we define $\mathbf{M} = (\mathbf{m}_1, ..., \mathbf{m}_n) \in \mathbb{R}^{n_m \times n}$ and $\mathbf{P} = (\mathbf{p}_1, ..., \mathbf{p}_n) \in \mathbb{R}^{n_p \times n}$.

Example A: Simple model problem with known analytical solution

We select $n = 10^3$ uniformly spaced samples within the subspace $[-33, 33] \subset \mathbb{R}$.

Example B: Inversion of borehole resistivity measurements

We select $n = 10^6$. Then, for the five parameters described in Section 2.5, we select random samples of the following rescaled variables over the corresponding intervals forming a subspace of $\mathbb{R}^5$:

$$\begin{aligned} \log(\rho_l), \log(\rho_u), \log(\rho_h) &\in [0, 3] \\ \log(d_l), \log(d_u) &\in [-2, 1]. \end{aligned} \tag{5}$$

We consider arbitrary high-angle trajectories. For each model problem, we randomly select the trajectory parameters within the following intervals:

$$\begin{aligned} \alpha_{ini} &\in [83°, 97°] \\ \alpha_v &\in [-0.045°, 0.045°] \text{ (only for Example B.2).} \end{aligned} \tag{6}$$

4 | DATA PREPROCESSING

Notation

For each output parameter of $\mathcal{F}$ and $\mathcal{I}$, we denote by $\mathbf{x} = (x_1, ..., x_n)$ the n-samples associated with that parameter. These x_i are real scalar values for $i = 1, ..., n$. For example, in the borehole resistivity example, each variable $\mathbf{x}$ contains n samples of each particular geophysical quantity such as resistivities, distances, or given measurements (attenuations, phases, etc.). Each dimension corresponds to a particular value (sample) of that variable, for example, the geosignal attenuation recorded at a specific logging position. From the algebraic point of view, the variable $\mathbf{x}$ denotes a row of either matrix $\mathbf{M}$ or $\mathbf{P}$.

Data preprocessing algorithm

This algorithm consists of three steps.

1. **Logarithmic change of coordinates**. We introduce the following change of variables:

$$\mathcal{R}_{\ln}(\mathbf{x}) := (\ln x_1, ..., \ln x_n). \tag{7}$$

   For some geophysical variables (e.g., resistivity), this change of variables ensures that equal-size relative errors correspond to similar-size absolute errors. Thus, this change of variables allows us to perform *local* (within a variable) comparisons.

2. **Remove outlier samples**. In practice, often outlier measurements are present in the sample database. These outliers appear due to measurement error or the physics of the problem. For example, in borehole resistivity measurements, some apparent resistivity measurements approach infinity, producing “horns" in the logs. When outlier measurements exists in any particular variable of the i-th sample x_i, then the entire sample should be removed. Otherwise, outlier measurements affect the entire minimization problem, leading to poor numerical results. The removal process may be automated using statistical indicators, or decided by the user based on a priori physical knowledge about the problem. We follow this second approach in this work.

3. **Linear change of coordinates**. We now introduce a linear rescaling mapping into the interval $[0.5, 1.5]$. We select this interval since it has unit length and the mean of a normal (or a uniform) distribution variable $\mathbf{x}$ is equal to one. Let $x_{\min} := \min_i x_i$, $x_{\max} := \max_i x_i$. We define

$$\mathcal{R}_{lin}(\mathbf{x}) := \left(\frac{x_1 - x_{\min}}{x_{\max} - x_{\min}} + 0.5, ..., \frac{x_n - x_{\min}}{x_{\max} - x_{\min}} + 0.5 \right), \tag{8}$$

   where the limits $x_{\min}$ and $x_{\max}$ are fixed for all possible approximations $\mathbf{x}^{app}$. This change of variables allows us to perform a *global* comparison between errors corresponding to different variables since they all take values over the same interval.

   Remark: $x_{\min}$ and $x_{\max}$ could also be selected based on the physically valid interval of each particular variable rather than on the training samples.

Variables classification

We categorize each input and output geophysical variable $\mathbf{x}$ into two types: either linear (A) or log-linear (B). When necessary, we shall indicate that a particular variable belongs to a specific category by adding the corresponding symbol as subindex of the variable, e.g., $\mathbf{x}_A$. Table 1 describes the domain of those variables as well as the rescaling employed for each of them. Variables of type A only require a global rescaling while those of type B require both a local and a global change of variables.

Geophysical Variables	Category	Domain	Rescaling
Angles, attenuations, phases, and geosignals	A	$\mathbb{R}^n$	$\mathcal{R}_{lin}(\mathbf{x})$
Apparent resistivities, resistivities, and distances	B	$(a,\infty)^n$ $a>0$	$\mathcal{R}_{lin}(\mathcal{R}_{\ln}(\mathbf{x}))$

TABLE 1 Categories for geophysical variables: types A or B. We apply a different rescaling to each of them.

For simplicity, we denote by $\mathcal{R}$ the result of the above rescalings, i.e., $\mathcal{R}(\mathbf{x}_A) := \mathcal{R}_{lin}(\mathbf{x}_A)$, and $\mathcal{R}(\mathbf{x}_B) := \mathcal{R}_{lin}(\mathcal{R}_{\ln}(\mathbf{x}_B))$. In general, given a variable $\mathbf{x}$ (of category A or B), we represent $\mathbf{x}_\mathcal{R} := \mathcal{R}(\mathbf{x})$. Given a matrix $\mathbf{X} \in \mathbb{R}^{n_x \times n}$, we abuse notation and denote by $\mathbf{X}_\mathcal{R} := \mathcal{R}(\mathbf{X}) \in \mathbb{R}^{n_x \times n}$ to the matrix that results from applying operator $\mathcal{R}$ row-wise.

Remark: Substituting in Equation 7 the natural logarithm by the base ten logarithm does not affect the definition of $\mathcal{R}$. Results are identical.

5 | NORMS AND ERRORS

We first introduce both the vector and the matrix norms that we use during the training process.

Norms

We introduce a norm $||\cdot||_\mathbb{X}$ associated with the variable $\mathbf{x}$. In general, we employ the l_1 or l_2 vector norms and, for matrices, the l_1 and Frobenius norms.

Absolute and relative errors

Let $\mathbf{x}^{app} = (x_1^{app}, ..., x_n^{app})$ be an approximation of $\mathbf{x}$. We define the absolute error A_e between $\mathbf{x}^{app}$ and $\mathbf{x}$ in the $||\cdot||_\mathbb{X}$ norm as

$$A_e^\mathbb{X}(\mathbf{x}^{app}, \mathbf{x}) := ||\mathbf{x}^{app} - \mathbf{x}||_\mathbb{X}. \tag{9}$$

This error measure has limited use since it is challenging to select an absolute error threshold that distinguishes between a *good* and a *bad* quality approximation. To overcome this issue, practitioners often employ relative errors. We define the relative error R_e in percent between $\mathbf{x}^{app}$ and $\mathbf{x}$ in the $||\cdot||_\mathbb{X}$ norm as:

$$R_e^\mathbb{X}(\mathbf{x}^{app}, \mathbf{x}) := 100\frac{||\mathbf{x}^{app} - \mathbf{x}||_\mathbb{X}}{||\mathbf{x}||_\mathbb{X}}. \tag{10}$$

Error control

For a variable $\mathbf{x}$ and its approximation $\mathbf{x}^{app}$, we want to control the relative error of the rescaled variable, that is:

$$R_e^\mathbb{X}(\mathbf{x}_\mathcal{R}^{app}, \mathbf{x}_\mathcal{R}). \tag{11}$$

The value $B = ||\mathbf{x}_\mathcal{R}||_\mathbb{X}$ is expected to be similar for all variables $\mathbf{x}$. Thus:

$$\sum_{\mathbf{x}_\mathcal{R}} A_e^\mathbb{X}(\mathbf{x}_\mathcal{R}^{app}, \mathbf{x}_\mathcal{R}) = \sum_{\mathbf{x}_\mathcal{R}} ||\mathbf{x}_\mathcal{R}^{app} - \mathbf{x}_\mathcal{R}||_\mathbb{X} \approx B \sum_{\mathbf{x}_\mathcal{R}} \frac{||\mathbf{x}_\mathcal{R}^{app} - \mathbf{x}_\mathcal{R}||_\mathbb{X}}{||\mathbf{x}_\mathcal{R}||_\mathbb{X}} = \frac{B}{100} \sum_{\mathbf{x}_\mathcal{R}} R_e^\mathbb{X}(\mathbf{x}_\mathcal{R}^{app}, \mathbf{x}_\mathcal{R}). \tag{12}$$

Therefore, the minimum of the first and last terms of the above equation coincide.

6 | LOSS FUNCTION

In this section, we consider a set of weights $\theta \in \Theta$ and a function $\mathcal{I}_{\mathcal{R},\theta}$ that depends upon the selected DNN architecture (see Appendix A). Then, we introduce a loss function $L(\mathcal{I}_{\mathcal{R},\theta})$. We define the minimizer of the loss function over all possible weight sets θ as:

$$\mathcal{I}_{\mathcal{R},\theta^*} := \arg\min_{\theta\in\Theta} L(\mathcal{I}_{\mathcal{R},\theta}). \tag{13}$$

Function $\mathcal{I}_{\theta^*} := \mathcal{R}^{-1} \circ \mathcal{I}_{\mathcal{R},\theta^*} \circ \mathcal{R}$ is the final DNN approximation of $\mathcal{I}$. In the following, we analyze the advantages and limitations associated with the use of different loss functions.

6.1 | Data misfit

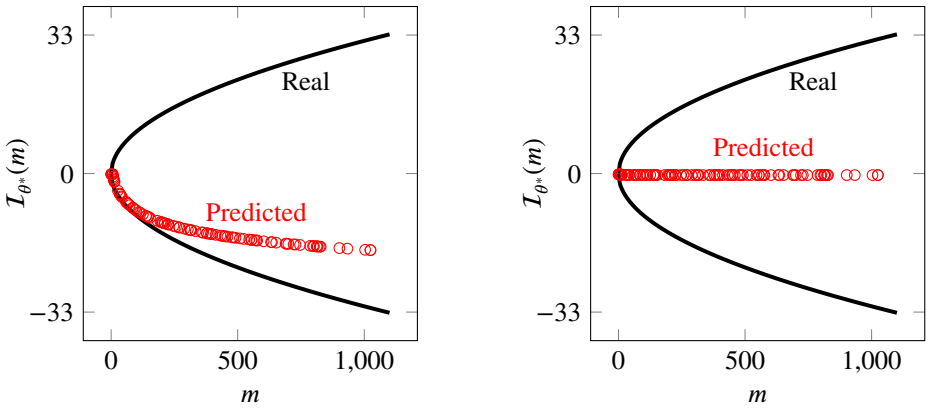
A simple loss function based on the data misfit is given by:

$$L(\mathcal{I}_{\mathcal{R},\theta}) := ||\mathcal{I}_{\mathcal{R},\theta}(\mathbf{T}_{\mathcal{R}}, \mathbf{M}_{\mathcal{R}}) - \mathbf{P}_{\mathcal{R}}||_P. \tag{14}$$

In the above equation, symbol $||\cdot||_P$ indicates l_1 or Frobenius norms introduced in Section 5.

6.1.1 | Example A: Model problem with known analytical solution

In this example, $n_p = 1$. Thus, matrix norms reduce to vector norms. Figure 7 illustrates the results we obtain using the l_1 and l_2 norms, respectively. These disappointing results are expected. Indeed, for the l_2-norm, it is easy to show that for a sufficiently flexible DNN architecture, the exact solution is $\mathcal{I}_{\theta^*} \approx 0$. To prove this, we assume that for every sample of the form $(m, \sqrt{m})$, there exist another one $(m, -\sqrt{m})$, which is satisfied in our dataset by construction (see Section 3). Then, for each pair of samples of this form, the exact point that minimizes the distance between both solutions ($\sqrt{m}$ and $-\sqrt{m}$) is 0. This argument can be extended to all pairs of samples. A similar reasoning shows that for the l_1-norm, any solution in between the two square root branches is an exact solution of the inverse problem. Our numerical solutions in Figure 7 confirm these simple mathematical observations. Thus, the data misfit loss function is unsuitable for inversion purposes.



(a) $\|\cdot\|_1$-norm **(b)** $\|\cdot\|_2$-norm

FIGURE 7 Analytical solution vs DNN predicted solution evaluated over the test dataset using the loss function based on the data misfit.

6.2 | Misfit of the measurements

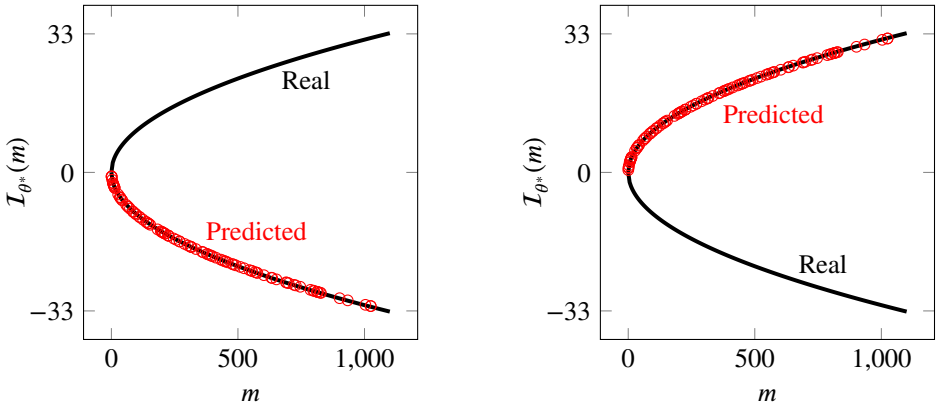
To overcome the aforementioned limitation, we consider the following loss function that measures the misfit of the measurements (see[37]):

$$L(\mathcal{I}_{\mathcal{R},\theta}) := \|(\mathcal{F}_{\mathcal{R}} \circ \mathcal{I}_{\mathcal{R},\theta})(\mathbf{T}_{\mathcal{R}}, \mathbf{M}_{\mathcal{R}}) - \mathbf{M}_{\mathcal{R}}\|_M, \tag{15}$$

where $\mathcal{F}_{\mathcal{R}} := \mathcal{R} \circ \mathcal{F} \circ \mathcal{R}^{-1}$, and $||\cdot||_M$ indicates a matrix norm of the type introduced in Section 5.

Example A: Model problem with known analytical solution

Figure 8 shows the inversion results when using the misfit of the measurements. We recover one of the possible solutions of the inverse operator. A regularization term could be introduced to select one solution branch over the other.



(a) $\|\cdot\|_1$-norm **(b)** $\|\cdot\|_2$-norm

FIGURE 8 Analytical solution vs DNN predicted solution evaluated over the test dataset using the loss function based on the measurements misfit.

Despite the accurate results exhibited for the above example, the proposed loss function has some critical limitations that affect its performance. Namely, during training, it is necessary to evaluate the forward problem multiple times. Depending upon the size of the training dataset and number of iterations required to converge, this may lead to millions of forward function evaluations. Solving the forward problem for such large number of times is time-consuming even with a 1.5D semi-analytic simulator. Moreover, most forward solvers are implemented for CPU architectures, while the training of the DNN normally occurs on GPUs. This requires a permanent communication between GPU and CPU, which further slows down the training process. Additionally, porting the forward solver $\mathcal{F}$ to a GPU may be complex to implement and bring additional numerical difficulties.

6.3 | Encoder-Decoder

To overcome the aforementioned implementation challenges, we propose to approximate the forward function using another DNN $\mathcal{F}_{\phi^*}$, where $\phi^* \in \boldsymbol{\Phi}$ are the parameters associated to the trained DNN. With this approach, we simultaneously train the forward and inverse operators solving the following optimization problem:

$$\begin{aligned}(\mathcal{F}_{\mathcal{R},\phi^*}, \mathcal{I}_{\mathcal{R},\theta^*}) := \arg\min_{\phi\in\boldsymbol{\Phi},\theta\in\boldsymbol{\Theta}} \{ &\|(\mathcal{F}_{\mathcal{R},\phi}\circ\mathcal{I}_{\mathcal{R},\theta})(\mathbf{T}_{\mathcal{R}}, \mathbf{M}_{\mathcal{R}}) - \mathbf{M}_{\mathcal{R}}\|_M \\ &+ \|\mathcal{F}_{\mathcal{R},\phi}(\mathbf{T}_{\mathcal{R}}, \mathbf{P}_{\mathcal{R}}) - \mathbf{M}_{\mathcal{R}}\|_M \},\end{aligned} \tag{16}$$

Function $\mathcal{F}_{\phi^*} := \mathcal{R}^{-1}\circ\mathcal{F}_{\mathcal{R},\phi^*}\circ\mathcal{R}$ is the final DNN approximation to $\mathcal{F}$. The first term in the above loss function constitutes an Encoder-Decoder DNN architecture[5] and ensures that function $\mathcal{I}_{\mathcal{R},\theta^*}$ shall be a inverse of $\mathcal{F}_{\mathcal{R},\phi^*}$. The second term imposes that the forward DNN approximates the ground truth data. In particular, it prevents situations in which both $\mathcal{I}_{\mathcal{R},\theta^*}$ and $\mathcal{F}_{\mathcal{R},\phi^*}$ approximate the identity operator.

Example A: Model problem with known analytical solution

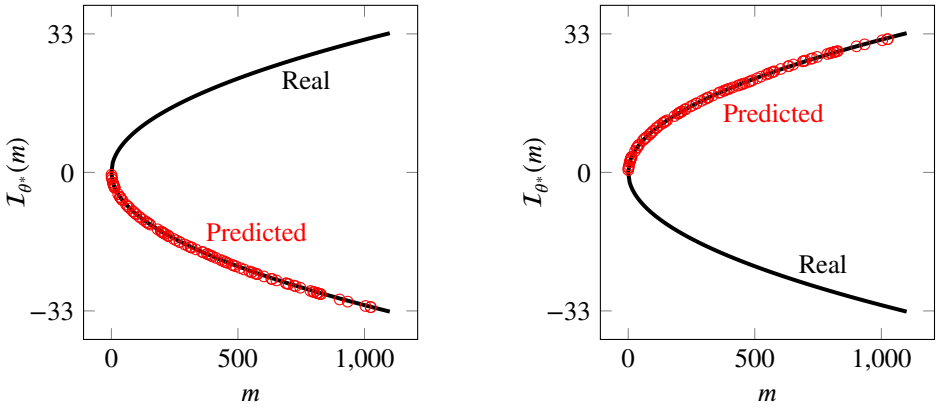
Figure 9 shows the results obtained with the Encoder-Decoder loss function. We recover accurate inversion results.

6.4 | Two-steps approach

It is possible to decompose the above Encoder-Decoder based loss function into two steps: the first optimization problem intends to approximate the forward function, and the second one determines the inverse operator:

$$\mathcal{F}_{\mathcal{R},\phi^*} := \arg\min_{\phi\in\boldsymbol{\Phi}} \|\mathcal{F}_{\mathcal{R},\phi}(\mathbf{T}_{\mathcal{R}}, \mathbf{P}_{\mathcal{R}}) - \mathbf{M}_{\mathcal{R}}\|_M, \tag{17a}$$

$$\mathcal{I}_{\mathcal{R},\theta^*} := \arg\min_{\theta\in\boldsymbol{\Theta}} \|(\mathcal{F}_{\mathcal{R},\phi^*}\circ\mathcal{I}_{\mathcal{R},\theta})(\mathbf{T}_{\mathcal{R}}, \mathbf{M}_{\mathcal{R}}) - \mathbf{M}_{\mathcal{R}}\|_M. \tag{17b}$$

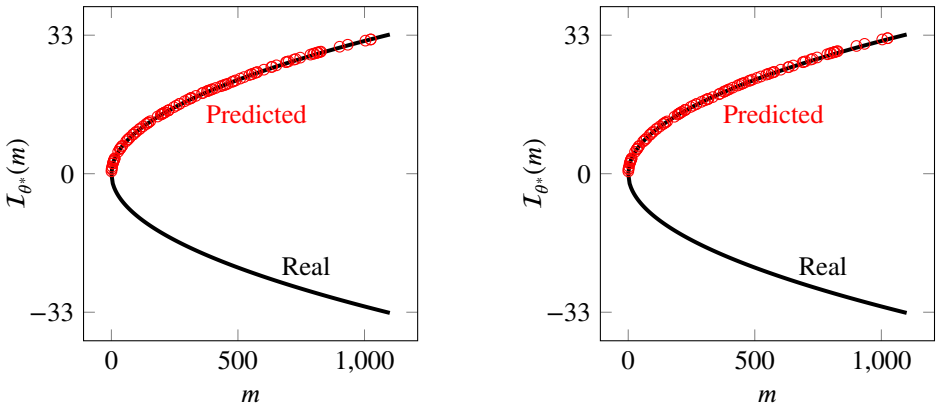


(a) $\|\cdot\|_1$-norm **(b)** $\|\cdot\|_2$-norm

FIGURE 9 Analytical solution vs DNN predicted solution evaluated over the test dataset using the Encoder-Decoder loss function.

Example A: Model problem with known analytical solution

Figure 10 shows the results of the inversion using the two-steps approach. We recover a faithful approximation of the inverse operator.



(a) $\|\cdot\|_1$-norm **(b)** $\|\cdot\|_2$-norm

FIGURE 10 Analytical solution vs DNN predicted solution evaluated over the test dataset using the two-step loss function.

Remark A: Based on the above discussion, it may seem that optimization problems given by either Equations 16 or 17 are ideal to solve inverse problems. However, there is a critical issue that needs to be addressed. In Equation 17a, the forward DNN $\mathcal{F}_{\mathcal{R},\phi}$ is trained only for the given dataset samples. However, the output of the DNN approximation of the inverse operator $\mathcal{I}_{\mathcal{R},\theta}$ will often deliver data far away from the data space used to produce the training samples. This may lead to catastrophic results. To illustrate this, we consider our model problem with known analytical solution. If we consider a dataset with only positive values of p, then the following approximations will lead to a zero loss function:

$$\mathcal{F}_{\phi^*}(p) = \begin{cases} p^2 & \text{if } p > 0 \\ ap^2 & \text{if } p < 0 \end{cases} \qquad \mathcal{I}_{\theta^*}(m) = -\sqrt{m/a}, \tag{18}$$

for any $a > 0$. However, if $a \neq 1$, this approximation is far away from both our original forward and inverse solutions. To prevent these undesired situations, one should ensure that the output space of $\mathcal{F}_{\mathcal{R},\theta^*}$ is sufficiently close to the space from which we obtain the training samples. However, this is often difficult to control.

6.5 | Regularization term

Inverse problems often exhibit non-unique solutions. Thus, in numerical methods, one introduces a regularization term to select a particular solution we prefer out of all the existing ones.

In DL applications, standard regularization techniques seek to optimize the model architecture (e.g., by penalizing high-valued weights). Herein, we regularize the system by adding the loss function of Equation 14 measured in the l_1-norm to either the optimization problem given by Equation 16 or 17b. This extra term guides the solution towards the ones considered in the training dataset, which may be convenient. Nevertheless, such a regularization term often hides the fact that other different solutions of the inverse problem may coexist. We study the advantages and limitations of including this regularization term in detail in Section 8.

7 | IMPLEMENTATION

To solve the forward problem, we employ a semi-analytic method[36] implemented in Fortran 90. It employs a Hankel transform to reduce the original 3D Maxwell system to a sequence of uncoupled 1D problems, whose solutions are analytical in the Hankel domain[38]. Then, we perform a numerical inverse Hankel transform with an adaptive Andersson quadrature rule[39]. With it, we produce a dataset containing one million samples (*ground truth*). Each sample consists of a randomly selected 1D layered model (see Section 3 for details). We use 80% of the samples for training the DNNs, 10% for validating them, and the remaining 10% for testing.

We consider two DNN architectures to approximate $\mathcal{F}$ and $\mathcal{I}$, respectively. The forward function $\mathcal{F}$ is well-posed and continuous, while the inverse operator $\mathcal{I}$ is not even well-defined. Thus, we employ a simpler DNN architecture to approximate $\mathcal{F}$ than to approximate $\mathcal{I}$. See Appendix A for details. We use the l_1 norm for the loss function.

We implement our DNNs using *Tensorflow 2.0*[40] and *Keras*[6] libraries. To train the DNNs, we use a *NVIDIA Quadro GV100* GPU. Using this hardware device, we require almost 70 hours to simultaneously train $\mathcal{F}_{\mathcal{R},\phi^*}$ and $\mathcal{I}_{\mathcal{R},\theta^*}$. While the training process is time-consuming, it is performed *offline*. Then, the *online* part of the process consists of simply evaluating the DNN, which can deliver an inverse model for thousands of logging positions in a few seconds. This low *online* computational cost makes the DNN approach an excellent candidate to perform inversion during geosteering operations in the field.

8 | NUMERICAL RESULTS

We perform a three step evaluation process of the results:

1. We first study the evolution of each term in the loss function during the training process. This analysis assesses the overall performance of the training process and, in particular, shows if any particular term of the loss function is driving the optimization procedure in detriment of other terms.
2. Second, we produce multiple cross-plots, which provide essential information about the adequacy of the selected loss function and dataset. These cross-plots indicate the possible non-uniqueness of the inverse problem at hand.
3. Finally, we apply the trained networks to invert three realistic synthetic models and analyze the overall success of the proposed DNN algorithm as well as its limitations.

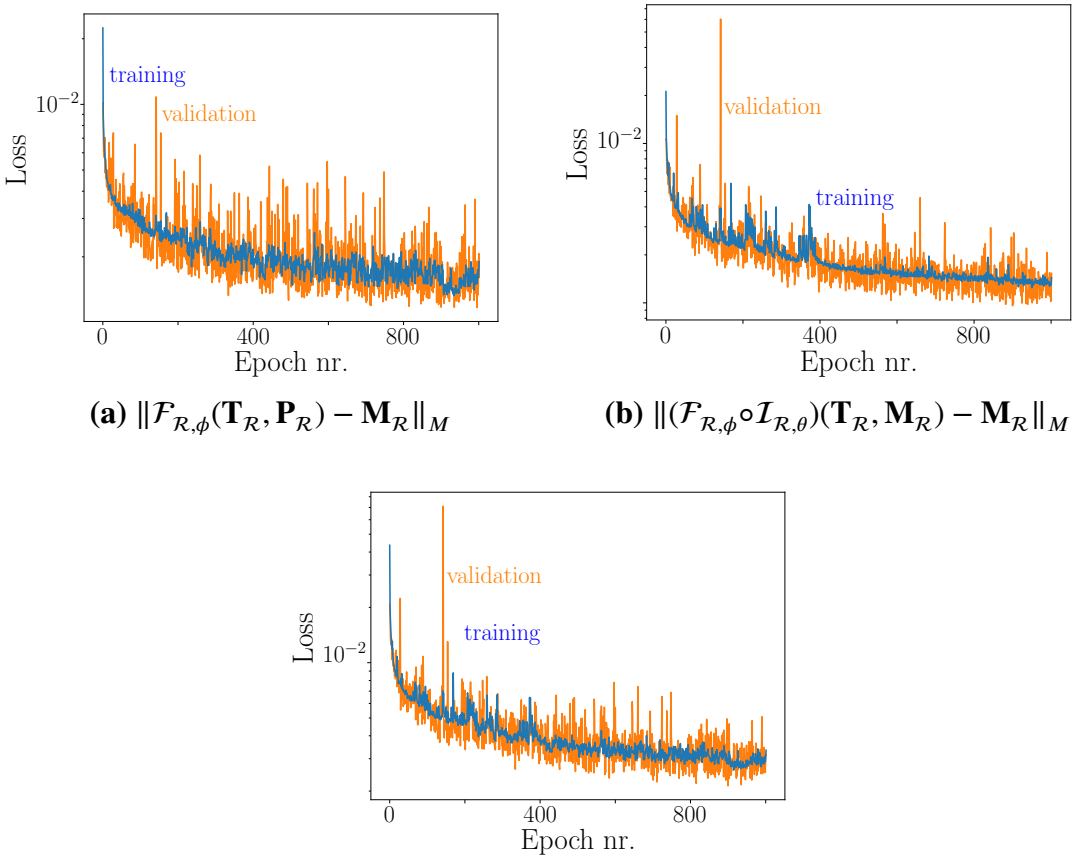
The above evaluation process provides a step-by-step assessment of the adequacy of the proposed strategy for solving inverse problems.

In most cases, we observe similar results when we consider the Encoder-Decoder loss function given by Equation 16 and the two-step loss function given by Equation 17. For brevity, we mostly focus on the Encoder-Decoder results. Additionally, we include one set of results using the two-step loss function, for which the observed behavior is essentially different from that of the Encoder-Decoder process.

8.1 | Evolution of the loss function

Figure 11 displays the evolution of the terms composing the Encoder-Decoder loss function described in Equation 16 for Example B.1. Figure 12 displays the corresponding results when we add the regularization term based on Equation 14. In both figures,

we observe: (a) a proper reduction of the total loss function, indicating that the overall minimization process is successful; (b) an adequate balance between the loss contribution of the different terms composing each loss function, suggesting that all terms of the loss functions are simultaneously minimized; and (c) a satisfactory match between the loss functions corresponding to the training and the validation data samples, which indicates we avoid overfitting. We observe a similar behavior with Example B.2, which we skip for brevity. We do not detail the results per variable since the applied rescaling of Section 4 guarantees a good balance between different variables.



(a) $\|\mathcal{F}_{\mathcal{R},\phi}(\mathbf{T}_{\mathcal{R}}, \mathbf{P}_{\mathcal{R}}) - \mathbf{M}_{\mathcal{R}}\|_M$

(b) $\|(\mathcal{F}_{\mathcal{R},\phi} \circ \mathcal{I}_{\mathcal{R},\theta})(\mathbf{T}_{\mathcal{R}}, \mathbf{M}_{\mathcal{R}}) - \mathbf{M}_{\mathcal{R}}\|_M$

(c) Total Loss

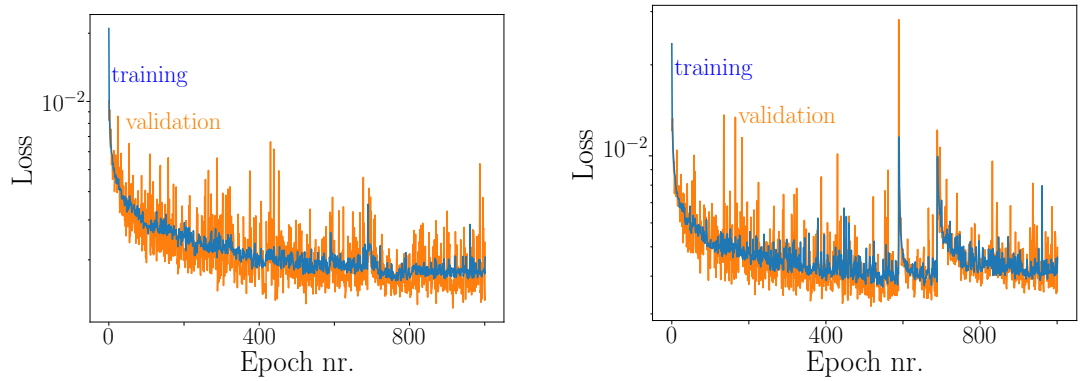
FIGURE 11 Example B.1. Evolution of the different terms of the Encoder-Decoder loss function given by Equation 16 without regularization.

8.2 | Cross-plots

We consider the following types of cross-plots:

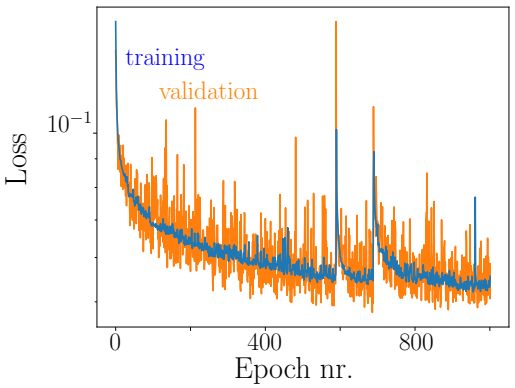
$$\begin{aligned} &\text{Cross-plot 1: } \mathcal{F}\circ\mathcal{I} \;\; vs \;\; \mathcal{F}_{\phi^*}\circ\mathcal{I} \\ &\text{Cross-plot 2: } \mathcal{F}\circ\mathcal{I} \;\; vs \;\; \mathcal{F}_{\phi^*}\circ\mathcal{I}_{\theta^*} \\ &\text{Cross-plot 3: } \mathcal{F}\circ\mathcal{I} \;\; vs \;\; \mathcal{F}\circ\mathcal{I}_{\theta^*} \\ &\text{Cross-plot 4: } \quad \mathcal{I} \;\; vs \;\; \mathcal{I}_{\theta^*} \end{aligned} \tag{19}$$

In the above, $\mathcal{F}$ and $\mathcal{I}$ are the exact functions and they define the *ground truth*, while the others are the *predictions* our DNNs deliver. In particular, in the first three types of cross-plots the ground truth is simply the identity mapping. We could display each type of cross-plot for the training, validation, and test data samples and for each variable. In our Example B, this makes a

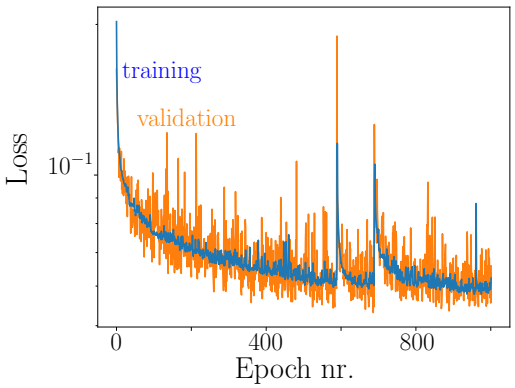


(a) $\|\mathcal{F}_{\mathcal{R},\phi}(\mathbf{T}_{\mathcal{R}}, \mathbf{P}_{\mathcal{R}}) - \mathbf{M}_{\mathcal{R}}\|_M$

(b) $\|(\mathcal{F}_{\mathcal{R},\phi} \circ \mathcal{I}_{\mathcal{R},\theta})(\mathbf{T}_{\mathcal{R}}, \mathbf{M}_{\mathcal{R}}) - \mathbf{M}_{\mathcal{R}}\|_M$



(c) $||\mathcal{I}_{\mathcal{R},\theta}(\mathbf{T}_{\mathcal{R}}, \mathbf{M}_{\mathcal{R}}) - \mathbf{P}_{\mathcal{R}}||_P$



(d) Total Loss

FIGURE 12 Example B.1. Evolution of the different terms of the Encoder-Decoder loss function given by Equation 16 with the regularization term prescribed by Equation 14.

total of 69 cross-plots. In addition, we need to repeat them for each considered loss function. To compress this information, we quantify each cross-plot with a single number: the statistical measure R-squared (R^2), which represents how much variation of the ground truth is explained by the predicted value. When this value is close to 1, indicating a perfect matching between the predicted value and the ground truth, we can safely omit these cross-plots. Otherwise, cross-plots display interesting information beyond what R^2 provides.

The proper interpretation of the cross-plots (or alternatively, R^2 factors) is of utmost importance. Cross-plots of type 1 (Equation 19_1) indicate how well the forward function is approximated over the given dataset. The cross-plots of type 2 (Equation 19_2) display how well the composition of the predicted forward and inverse mappings approximate the identity. These two types of cross-plots often deliver high R^2 factors, since the corresponding approximations are directly built into the Encoder-Decoder loss function given by Equation 16. Table 2 confirms those theoretical predictions for the most part.

An in-depth inspection of Table 2 reveals that for the the geosignal measurements (both attenuation and phase) corresponding to the Example B.1 without regularization, the cross-plots 2 exhibit significantly better R^2 factors than those corresponding to the cross-plots 1. Figure 13 shows the corresponding cross-plots. The anti-diagonal grey line shown in cross-plots of type 1 corresponds to dip angles of the logging instrument that are close to 90 degrees. At that angle, the geosignal is discontinuous. Thus, it is not properly approximated via DL algorithms, which approximate continuous functions. Cross-plots of type 2 seem to fix that issue by delivering higher R^2 factors and apparently nicer figures. However, they amplify the problem. In reality, the DL approximation of the inverse operator is inverting an incorrect forward approximation. Numerical results below illustrate this problem.

Cross-plots 1

R^2 factors	Atten. LWD Coaxial	Atten. Deep Coaxial	Atten. Deep Geosignal	Phase LWD Coaxial	Phase Deep Coaxial	Phase Deep Geosignal
Example B.1						
Training	0.9997	0.9992	**0.9509**	0.9996	0.9994	**0.9468**
Test	0.9995	0.9984	**0.9531**	0.9990	0.9991	**0.9487**
Without Reg.						
Example B.1						
Training	0.9998	0.9998	0.9897	0.9998	0.9998	0.9893
Test	0.9998	0.9998	0.9893	0.9998	0.9998	0.9890
With Reg.						
Example B.2						
Training	0.9959	0.9975	0.9872	0.9954	0.9980	0.9853
Test	0.9924	0.9960	0.9775	0.9920	0.9974	0.9765
Without Reg.						

Cross-plots 2

R^2 factors	Atten. LWD Coaxial	Atten. Deep Coaxial	Atten. Deep Geosignal	Phase LWD Coaxial	Phase Deep Coaxial	Phase Deep Geosignal
Example B.1						
Training	0.9997	0.9995	0.9998	0.9999	0.9996	0.9999
Test	0.9997	0.9994	0.9999	0.9999	0.9996	0.9999
Without Reg.						
Example B.1						
Training	0.9971	0.9980	0.9779	0.9970	0.9979	0.9798
Test	0.9970	0.9979	0.9785	0.9970	0.9978	0.9803
With Reg.						
Example B.2						
Training	0.9931	0.9958	0.9800	0.9933	0.9967	0.9821
Test	0.9890	0.9930	0.9701	0.9881	0.9944	0.9720
Without Reg.						

TABLE 2 R^2 factors for cross-plots 1 and 2 and Examples B.1 and B.2, with and without regularization, for training and test datasets. Numbers below 0.96 are marked in boldface.

Obtaining high R^2 factors associated to cross-plots of type 3 (Equation 19_3) is a challenging task as we discuss in Remark A of Section 6. Equation 18 shows a simple example in which cross-plots of type 1 and 2 deliver perfect R^2 marks and results, while cross-plots of type 3 are disastrous. This is also the situation that occurs in Example B.2. (see Table 3). While the original training dataset is based on 1D Earth models, the one obtained after the predicted DNN inversion is a *piecewise* 1D Earth model, for which $\mathcal{F}_{\phi^*}$ is untrained for. When this occurs, the training database should be upgraded, either by increasing the space of the data samples or by selecting a different parameterization (e.g., measurements) for each sample. In our case, we choose to parametrize each sample independently (the later stategy) and we move to Example B.1.

Table 3 shows mixed results for the Example B.1. Results without regularization are unremarkable with the geosignal forecasts showing poor results. The DNN inverse approximation accurately inverts for the outcome predicted by the DNN forward approximation. Nevertheless, since the DNN predicts solutions far from the true forward function, the predictions are poor. Again, this poor forecasting occurs because the DNN inverse approximation encounters subsurface models for which the forward DNN approximation is untrained. As a result, both the forward and inverse DDN approximations depart strongly from the

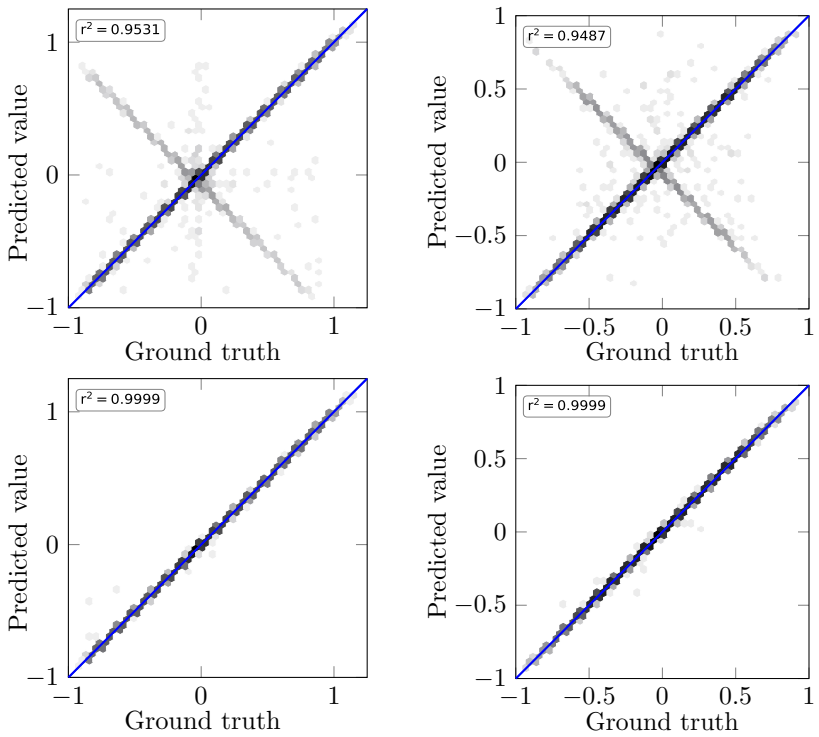


FIGURE 13 Geosignal cross plots for the Example B.1 without regularization for the test dataset. First row: Cross-plots 1. Second row: Cross-plots 2. First column: Attenuation. Second column: Phase.

true solutions. In other words, the inverse can only comply with their composition to be close to the identity, which is not robust to deliver accurate and physically relevant approximations.

Cross-plots 3

R^2 factors	Atten. LWD Coaxial	Atten. Deep Coaxial	Atten. Deep Geosignal	Phase LWD Coaxial	Phase Deep Coaxial	Phase Deep Geosignal
Example B.1						
Without Reg.	0.9468	0.7406	0.0013	0.9383	0.9116	0.0167
With Reg.	0.9971	0.9979	0.9807	0.9969	0.9979	0.9856
Example B.2						
Without Reg.	0.5721	0.8383	0.0253	0.4546	0.8611	0.0284
With Reg.	0.9010	0.9701	0.5901	0.8621	0.9618	0.5877

TABLE 3 R^2 factors for Cross-plots 3 and Examples B.1 and B.2, with and without regularization, for the test dataset.

To partially alleviate the above problem, we envision three possible solutions. First, we can increase the training dataset. This option is time-consuming and often impossible to achieve in practice. For example, herein, we already employ 1,000,000 samples. Second, we can include regularization. Results with regularization are of high quality (see Table 3). However, the regularization term may hide alternative physical solutions of the inverse problem. Thus, the regularization diminishes the ability to perform uncertainty quantification. Similarly, it may induce on the user excessive confidence in the results. A third option is to consider the two-step loss function given by Equation 17. Following this approach, we first adjust the forward DNN approximation before training the DNN inverse approximation. Fixing the forward DNN often provides a proper forecast even in areas with a lower rate of training samples before producing a DNN approximation that approximates the inverse of the DNN

forward approximation. Following this two-step approach without regularization, we obtain high R^2 factors for cross-plots of type 3: above 0.95 for the geosignal attenuation and phase, and above 0.99 for the remaining measurements.

Finally, the R^2 factors for the cross-plots of type 4 do not reflect on the accuracy of the DNN algorithm, but rather on the nature of the inverse problem at hand. Low R^2 factors indicate there exist multiple solutions. A regularization term (e.g., Equation 14) increases the R^2 indicator. Figure 14 clearly illustrates this fact. However, it is misleading to conclude that results without regularization are always worse. They may simply exhibit a different (but still valid) solution of the inverse problem.

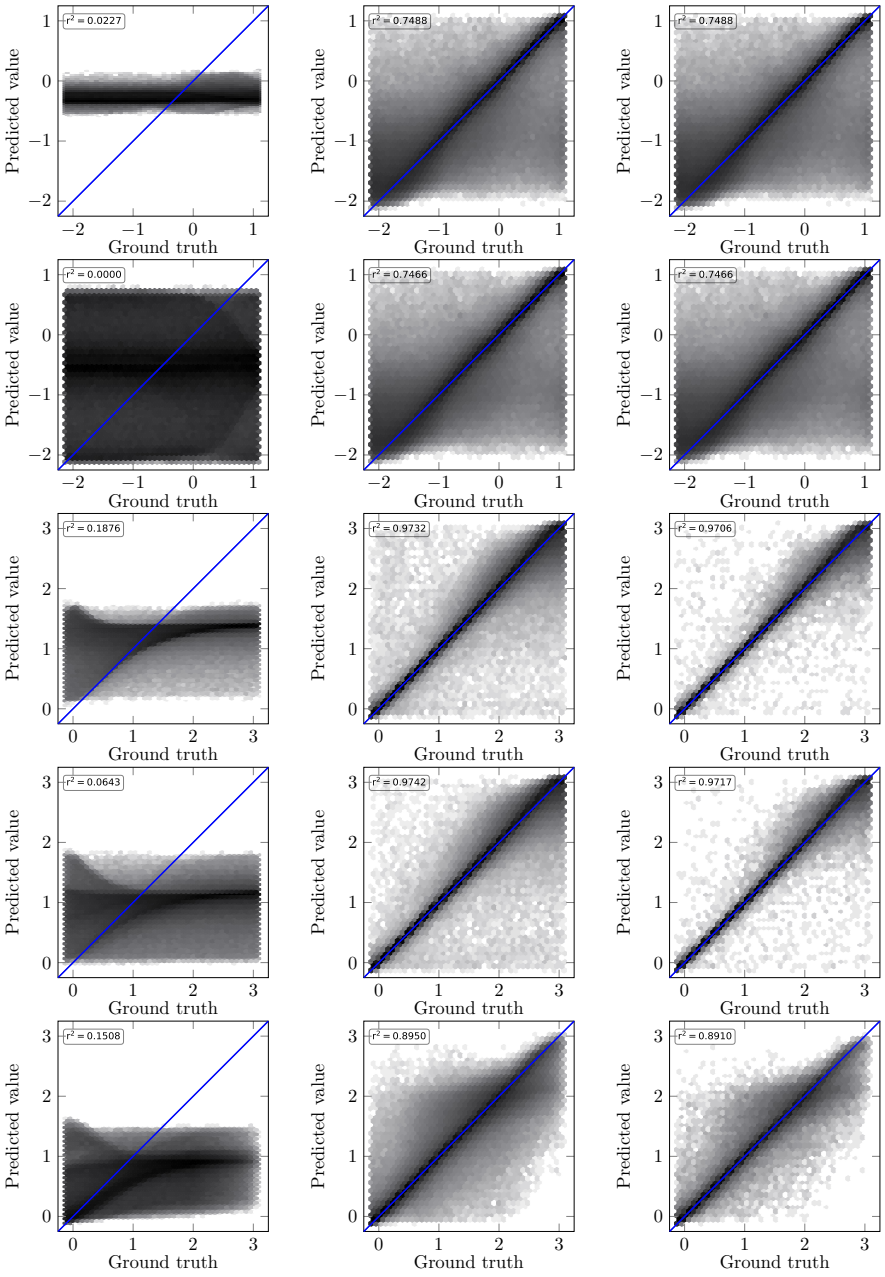


FIGURE 14 Cross-plots of type 4 for Example B.1 without regularization for the training dataset (first column), and with regularization for the training dataset (second column) and the test dataset (third column). First row: distance to the upper layer. Second row: distance to the lower layer. Third row: resistivity of upper layer. Fourth row: resistivity of lower layer. Fifth row: resistivity of central layer.

8.3 | Inversion of realistic synthetic models

We now consider three realistic synthetic examples to assess the performance of the inversion process. In terms of log accuracy, we observe qualitatively similar results for the attenuation and phase logs. Thus, in the following we only display the attenuation logs and omit the phase logs.

8.3.1 | Model Problem I

Figure 15 describes a well trajectory in a synthetic model problem. The model has a resistive layer with a water-bearing layer underneath, and exhibits two geological faults.

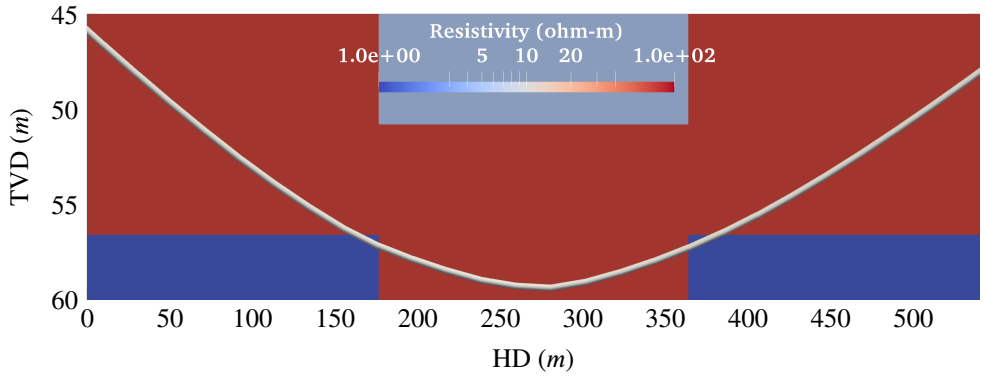
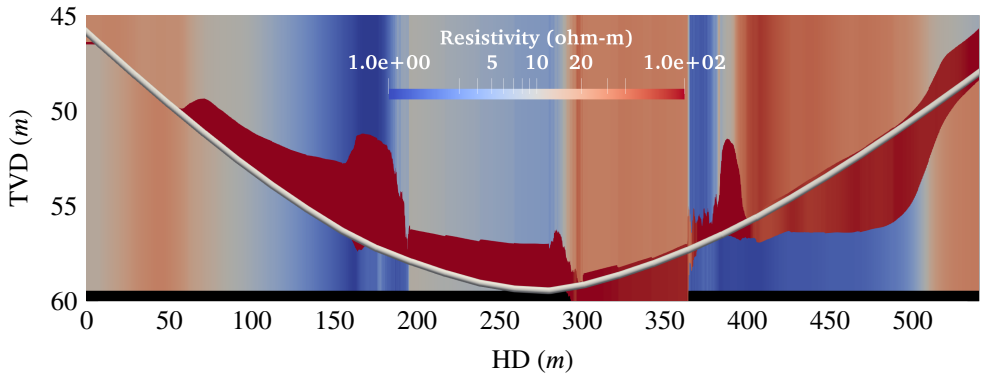


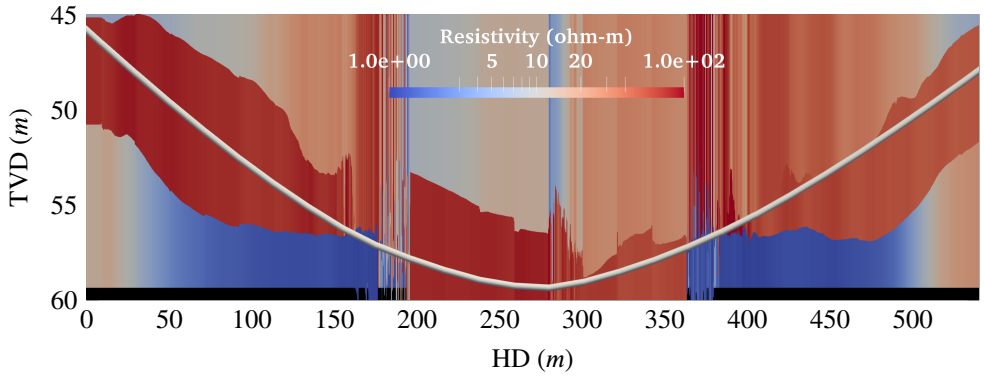
FIGURE 15 Formation of model problem I.

For the DNNs produced with the Example B.2 (with input measurements corresponding to 65 logging positions per sample), Figure 16 shows the corresponding inverted models using the Encoder-Decoder DNN with and without regularization. Results show inaccurate inversion results, specially for the case without regularization. Moreover, the predicted logs are far from the true logs, as Figure 17, and as expected from cross-plots 3 (see Table 3). The DNN inversion results are *piecewise* 1D models. However, the DNN approximation only trains with 1D models, not for piecewise 1D models, which explains the poor approximations they deliver (see Remark A on Section 6).

In the remainder of this section, we restrict to DNNs produced with Example B.1. That is, we parametrize all observations at one location using information from that location alone. Figure 18 shows the corresponding inverted models. For the case of the Encoder-Decoder loss function without regularization, we observe in Figure 18a an inverted model that is completely different from the original one. The corresponding logs (see Figure 19) are also inaccurate, as anticipated by the cross-plots results of type 3 shown in the previous subsection. When considering the two-step loss function without regularization, the recovered model (see Figure 18b) is still quite different from the original one. Nonetheless, we observe a superb matching in the logs (see Figure 20), which indicates the presence of a different solution for the inverse problem. This confirms that the given measurements are insufficient to provide a unique solution for the inverse problem. For the case with regularization, inversion results (see Figure 18b) match the original model, and the corresponding logs properly approximate the synthetic ones, see Figure 21. Figures 22 and 23 confirm that our methodology delivers a proper training of the forward function approximation and the composition $\mathcal{F}_{\phi^*} \circ \mathcal{I}_{\theta^*}$, respectively.

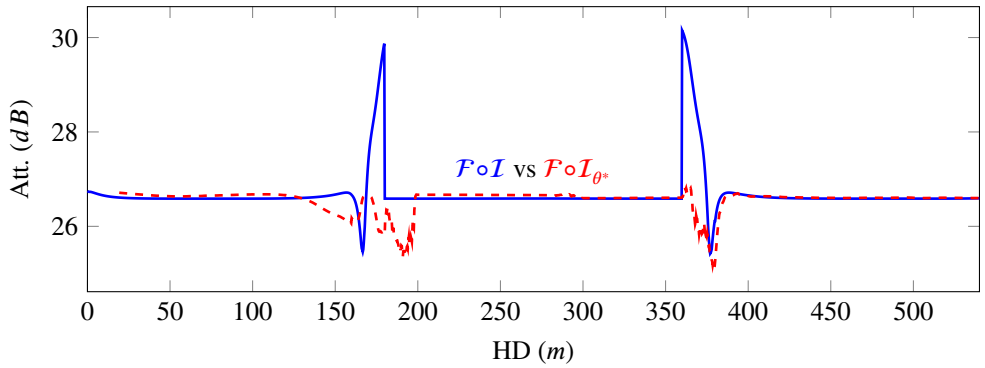


(a) Without regularization

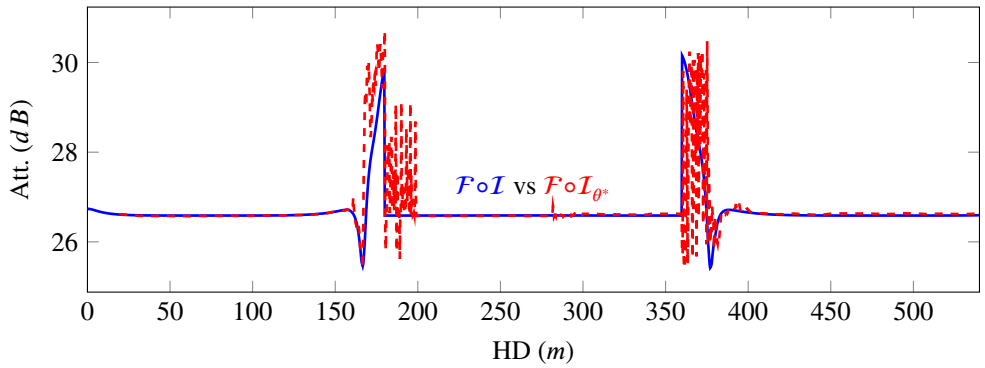


(b) With regularization

FIGURE 16 Inverted formation of model problem I using the inversion strategy of Example B.2, i.e., with input measurements corresponding to 65 logging positions per sample.

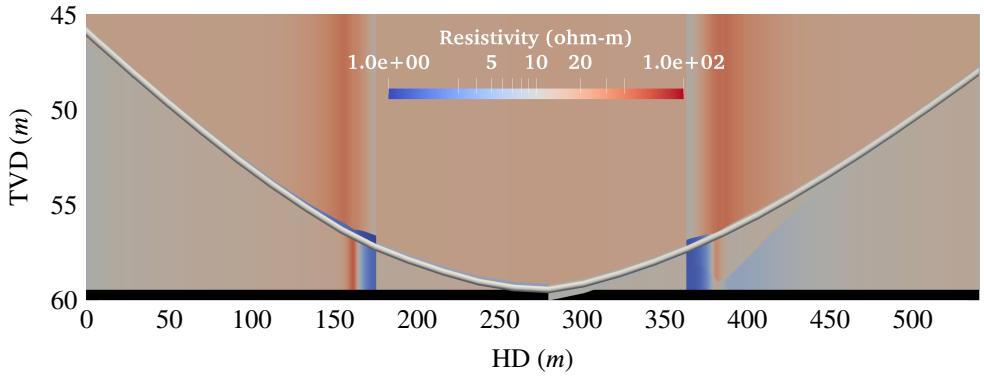


(a) LWD coaxial measurement. Without regularization

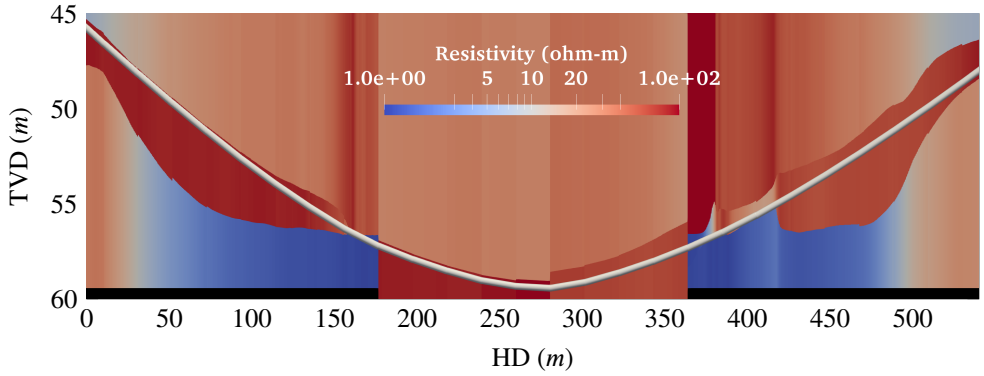


(b) LWD coaxial measurement. With regularization

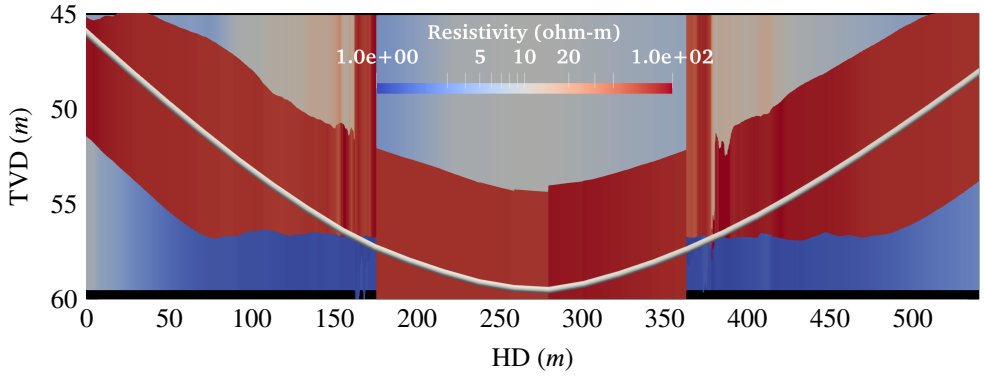
FIGURE 17 Model problem I. Comparison between $\mathcal{F}\circ\mathcal{I}$ and $\mathcal{F}\circ\mathcal{I}_{\theta^*}$ using the inversion strategy of Example B.2, i.e., with input measurements corresponding to 65 logging positions per sample.



(a) Predicted formation using the Encoder-Decoder loss function without regularization

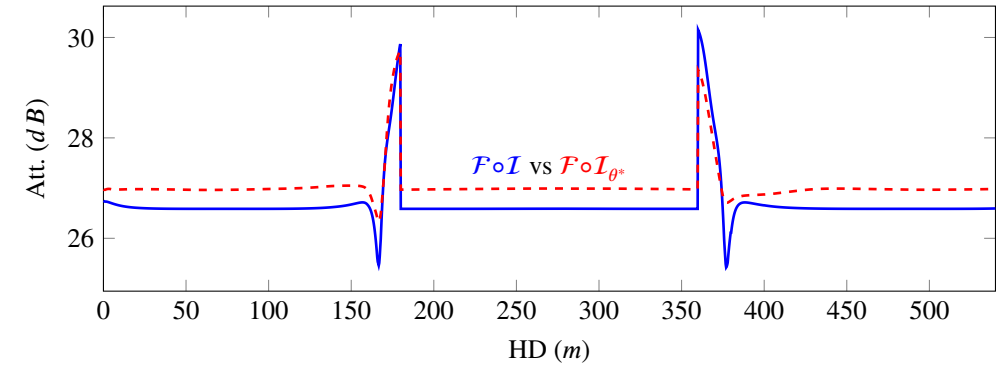


(b) Predicted formation using the two-step loss function without regularization

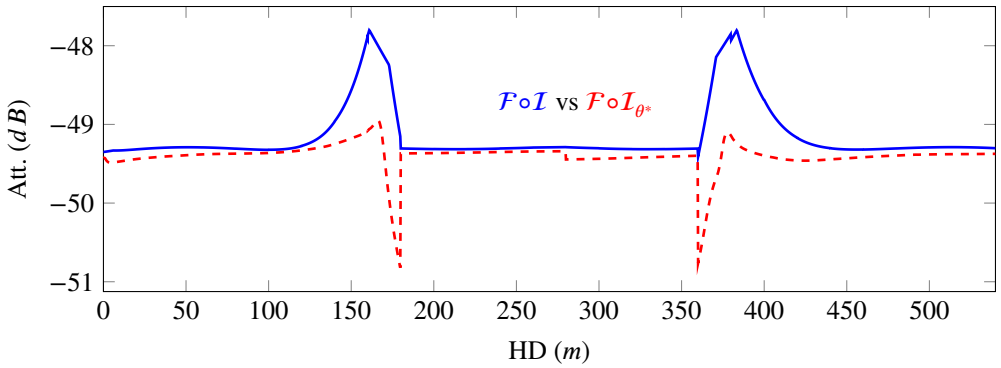


(c) Predicted formation using the Encoder-Decoder loss function with regularization

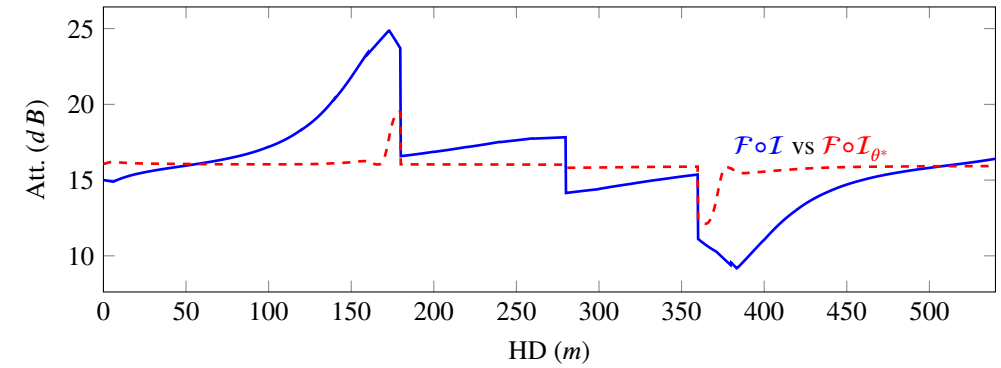
FIGURE 18 Inverted formation of model problem I using the inversion strategy of Example B.1, i.e., with input measurements corresponding to one logging positions per sample.



(a) LWD coaxial measurement

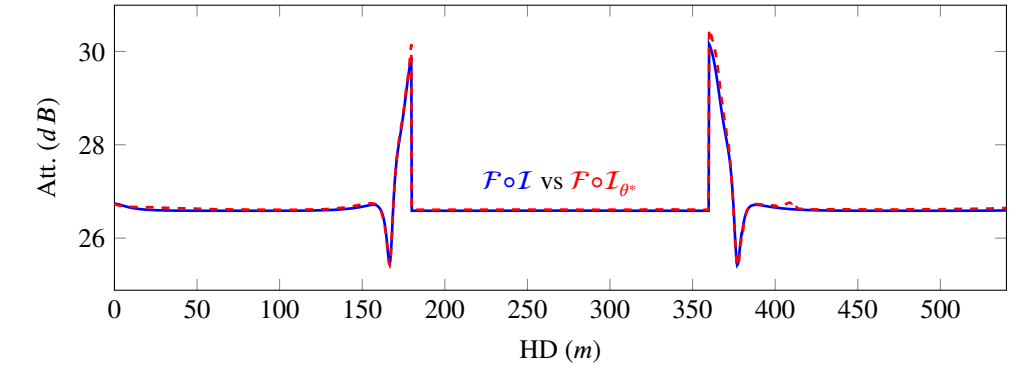


(b) Deep coaxial measurement

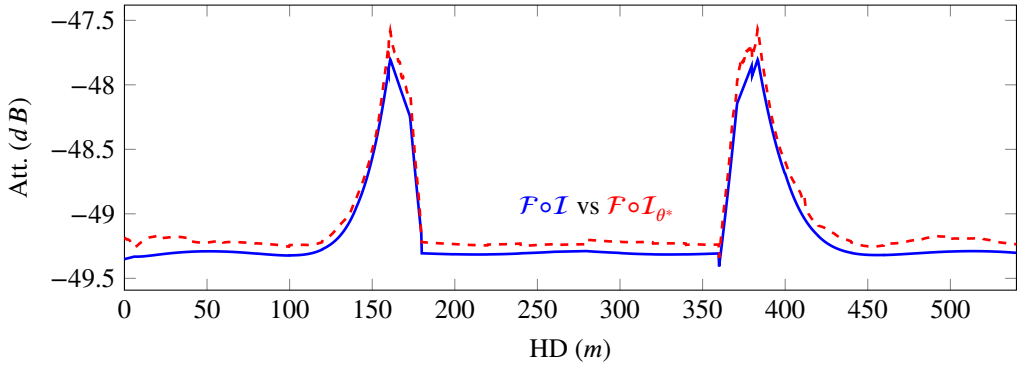


(c) Geosignal measurement

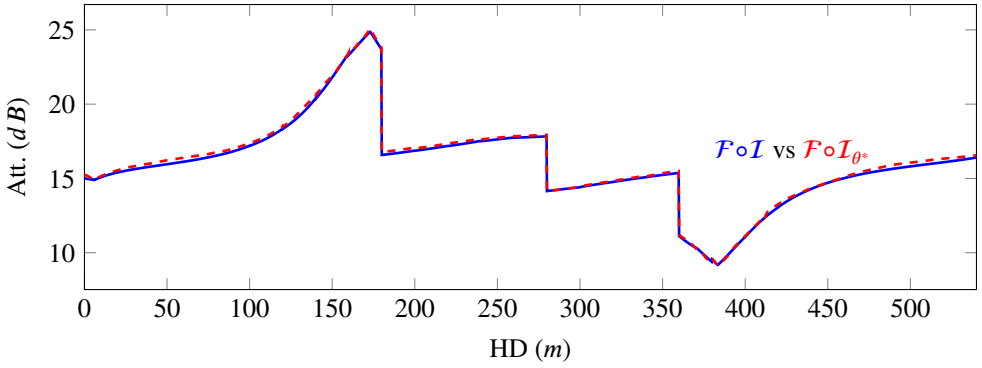
FIGURE 19 Model problem I. Comparison between $\mathcal{F}\circ\mathcal{I}$ and $\mathcal{F}\circ\mathcal{I}_{\theta^*}$ without regularization using the Encoder-Decoder loss function and the inversion strategy of Example B.1, i.e., with input measurements corresponding to one logging positions per sample.



(a) LWD coaxial measurement

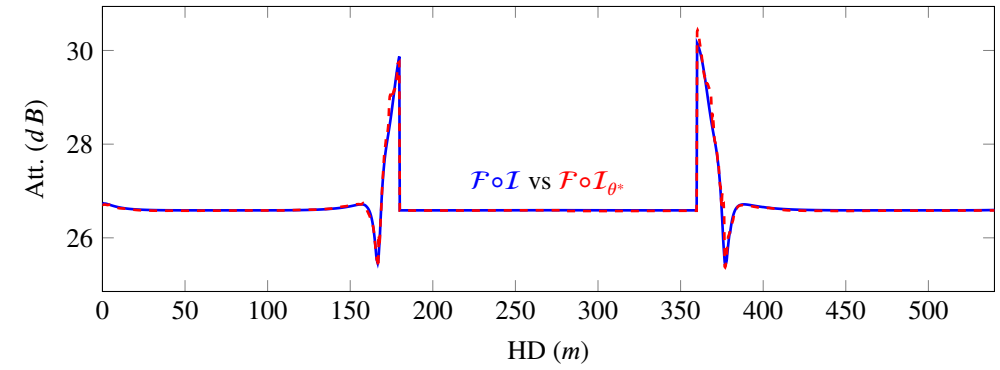


(b) Deep coaxial measurement

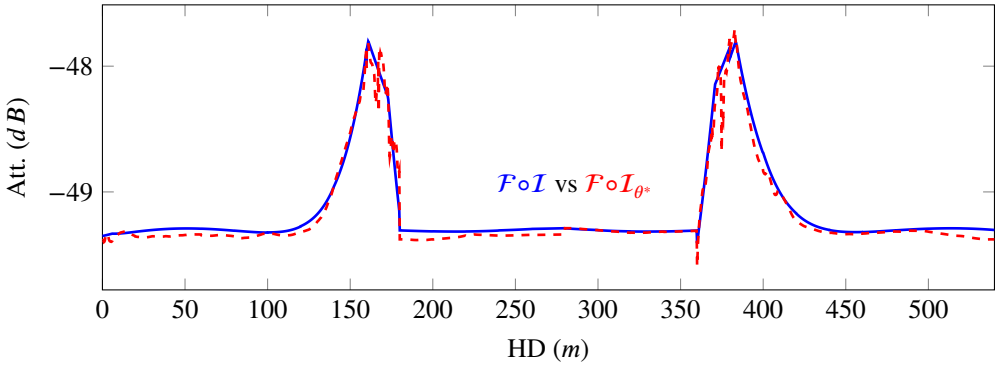


(c) Geosignal measurement

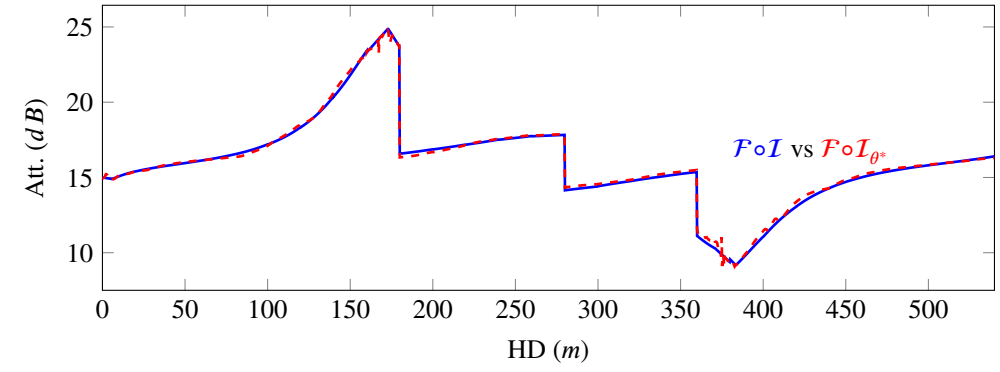
FIGURE 20 Model problem I. Comparison between $\mathcal{F}\circ\mathcal{I}$ and $\mathcal{F}\circ\mathcal{I}_{\theta^*}$ using the two-step loss function without regularization and the inversion strategy of Example B.1, i.e., with input measurements corresponding to one logging positions per sample.



(a) LWD coaxial measurement

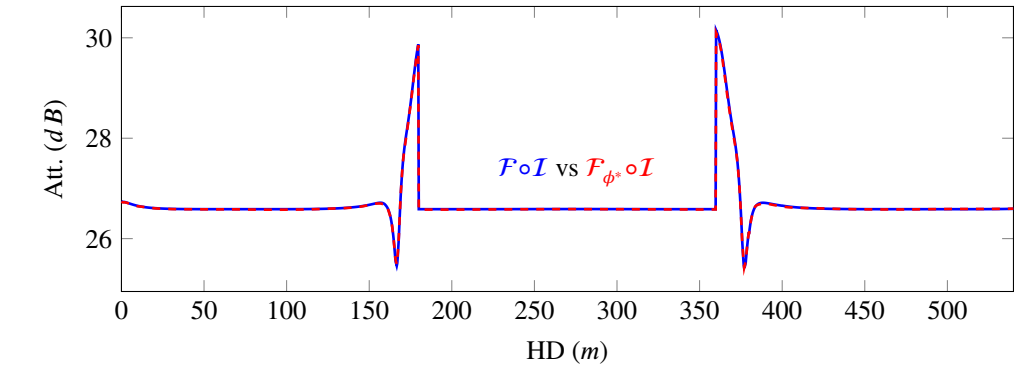


(b) Deep coaxial measurement

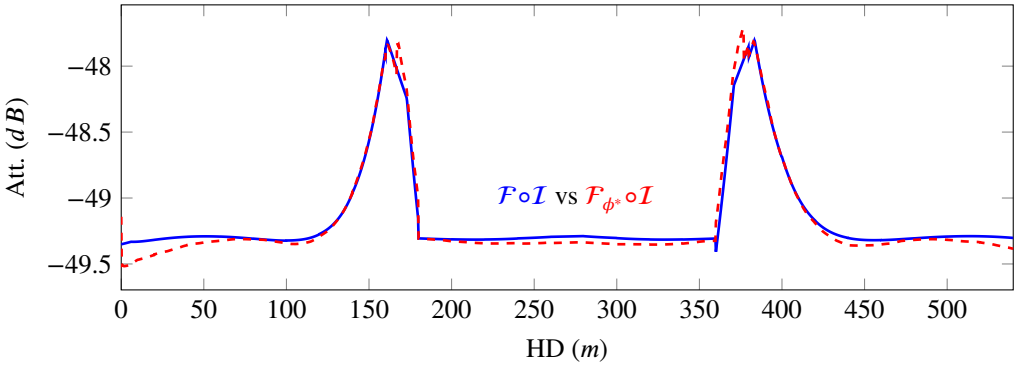


(c) Geosignal measurement

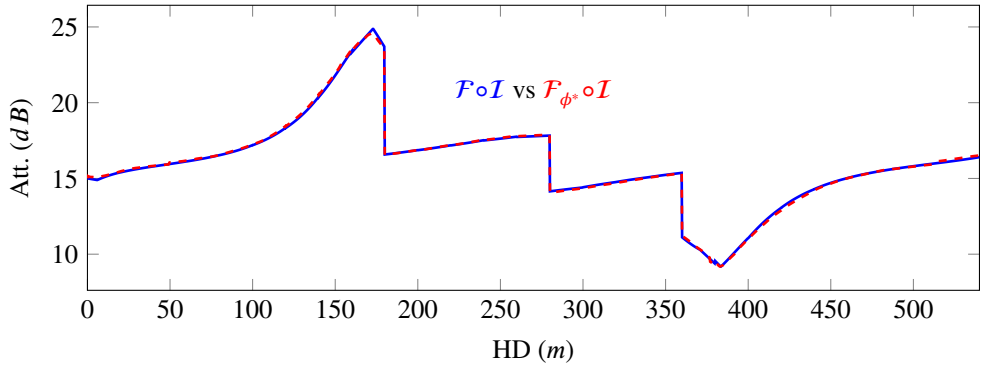
FIGURE 21 Model problem I. Comparison between $\mathcal{F}\circ\mathcal{I}$ and $\mathcal{F}\circ\mathcal{I}_{\theta^*}$ with regularization using the inversion strategy of Example B.1, i.e., with input measurements corresponding to one logging positions per sample.



(a) LWD coaxial measurement

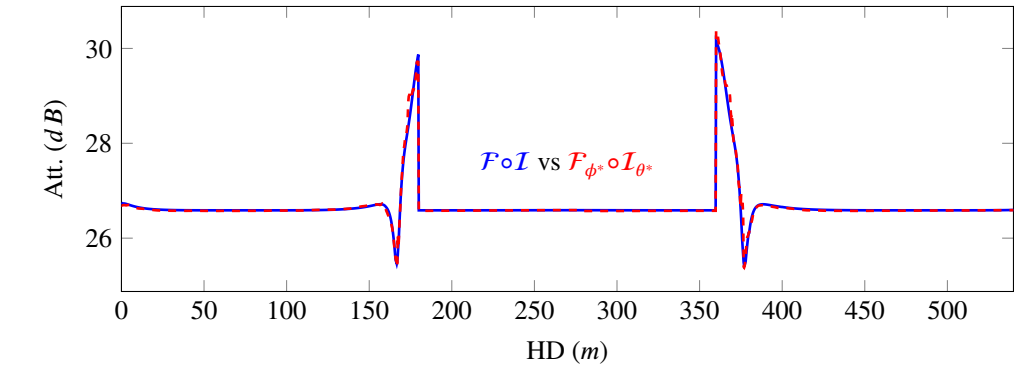


(b) Deep coaxial measurement

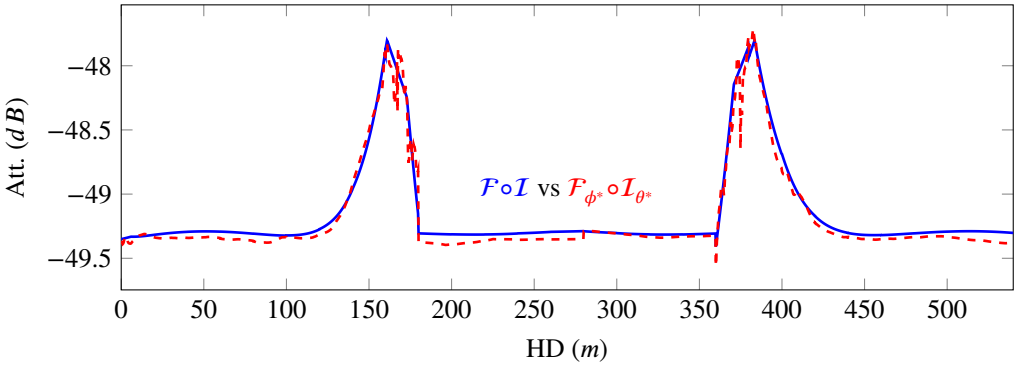


(c) Geosignal measurement

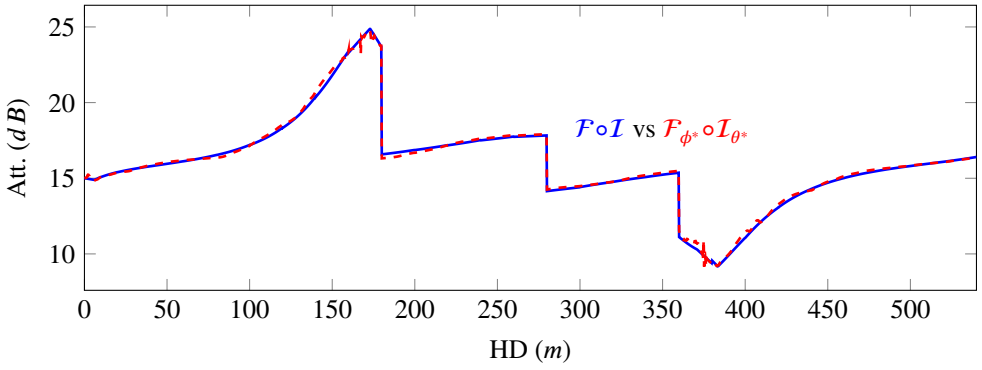
FIGURE 22 Model problem I. Comparison between $\mathcal{F}\circ\mathcal{I}$ and $\mathcal{F}_{\phi^*}\circ\mathcal{I}$ with regularization using the inversion strategy of Example B.1, i.e., with input measurements corresponding to one logging positions per sample.



(a) LWD coaxial measurement



(b) Deep coaxial measurement

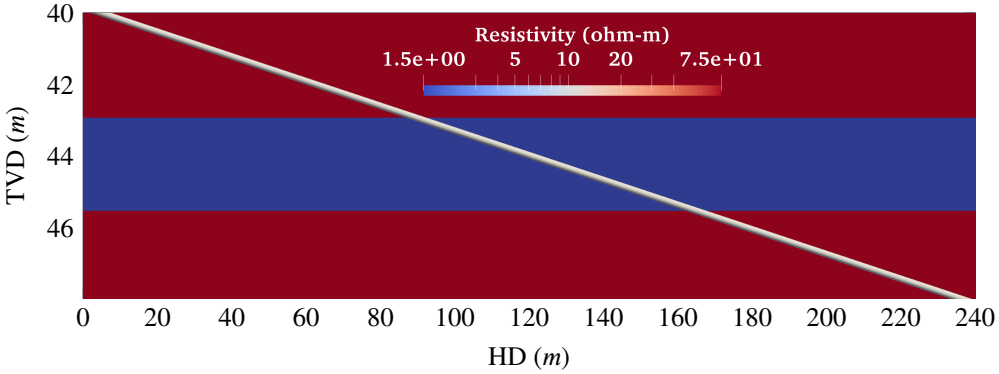


(c) Geosignal measurement

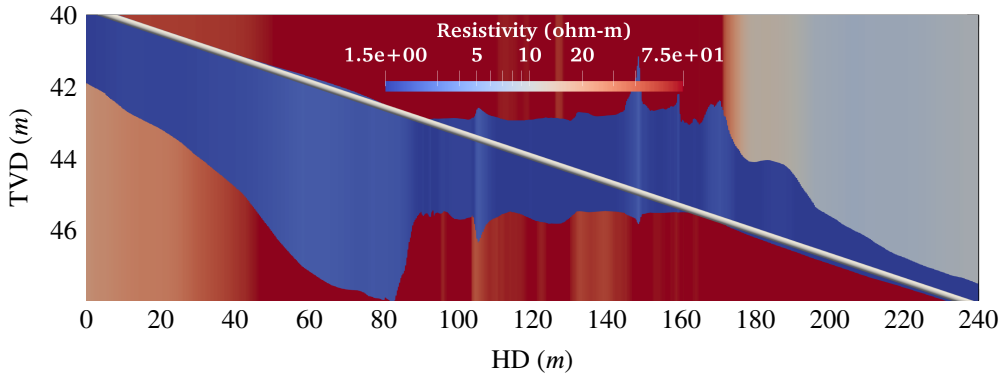
FIGURE 23 Model problem I. Comparison between $\mathcal{F}\circ\mathcal{I}$ and $\mathcal{F}_{\phi^*}\circ\mathcal{I}_{\theta^*}$ with regularization using the inversion strategy of Example B.1, i.e., with input measurements corresponding to one logging positions per sample.

8.3.2 | Model Problem II

In this problem, we consider a 2.5m-thick conductive layer surrounded by two resistive layers. A well trajectory with a dip angle equal to 87° crosses the formation. Figure 24 displays the original and predicted models by DL. This example illustrates some of the limitations of DNNs. In this case, the Earth models associated with part of the trajectory are outside the model problems considered in Section 2, which restrict to only one layer above and below the logging trajectory. Thus, the DNN is untrained for such models, and results cannot be trusted in those zones. Numerical results confirm these observations. Nonetheless, inaccurate inversion results are simple to identify by inspection of the logs (Figures 25 and 26).

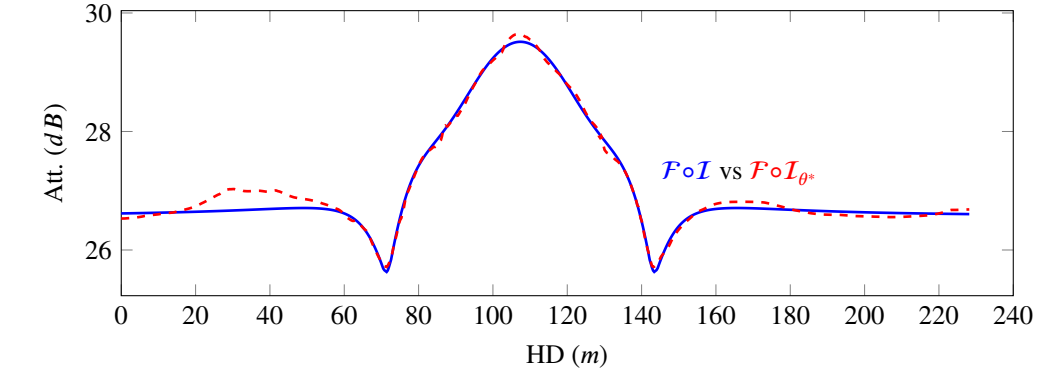


(a) Actual formation

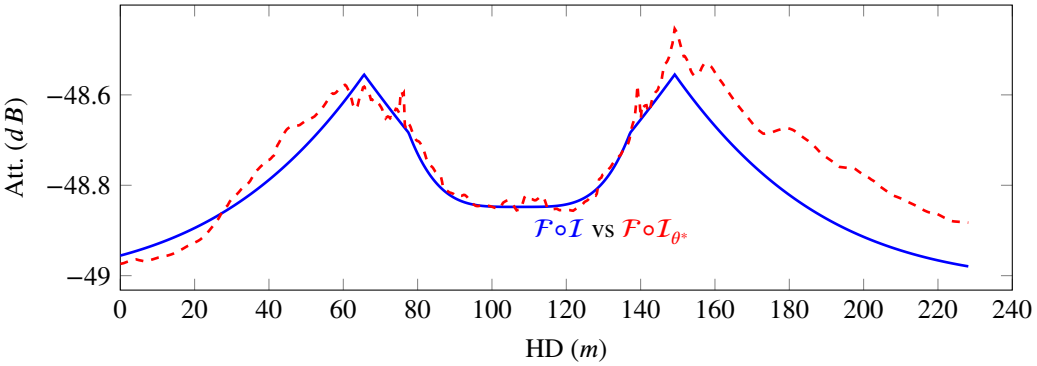


(b) Predicted formation using one logging position with regularization

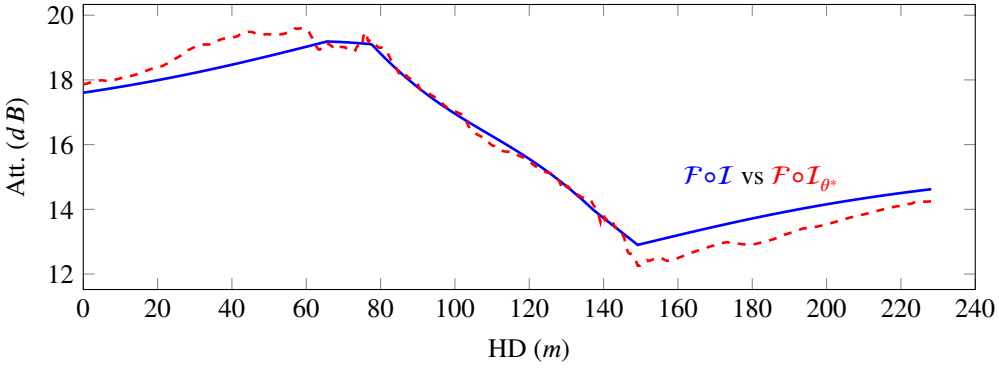
FIGURE 24 Model problem 2. Comparison between actual and predicted formations with regularization using the inversion strategy of Example B.1, i.e., with input measurements corresponding to one logging positions per sample.



(a) LWD coaxial measurement

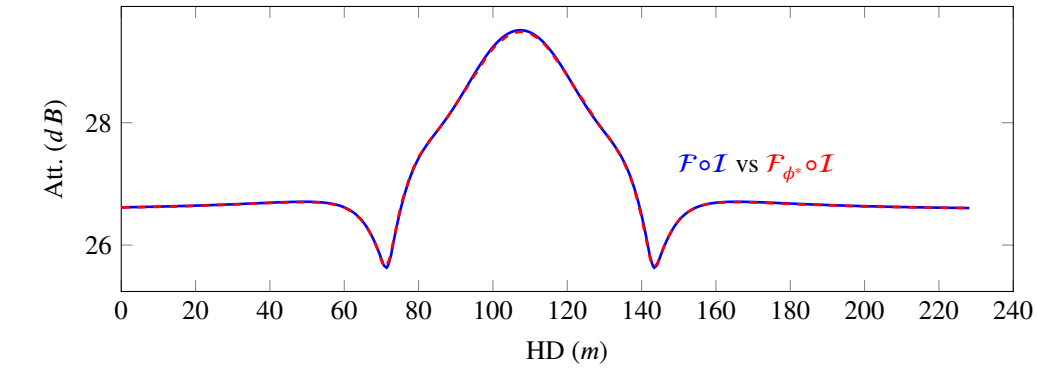


(b) Deep coaxial measurement

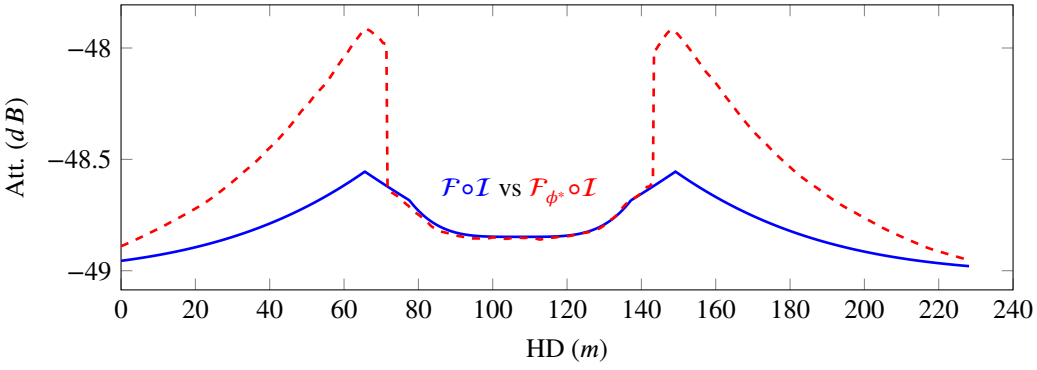


(c) Geosignal measurement

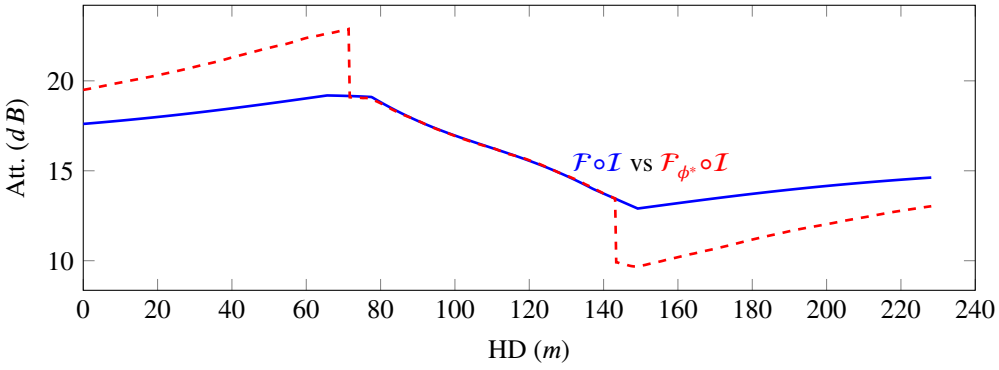
FIGURE 25 Model problem 2. Comparison between $\mathcal{F}\circ\mathcal{I}$ and $\mathcal{F}\circ\mathcal{I}_{\theta^*}$ with regularization using the inversion strategy of Example B.1, i.e., with input measurements corresponding to one logging positions per sample.



(a) LWD coaxial measurement



(b) Deep coaxial measurement

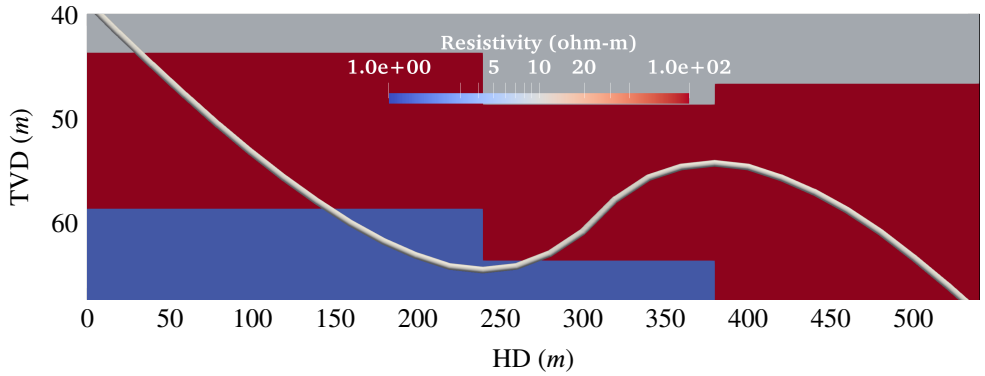


(c) Geosignal measurement

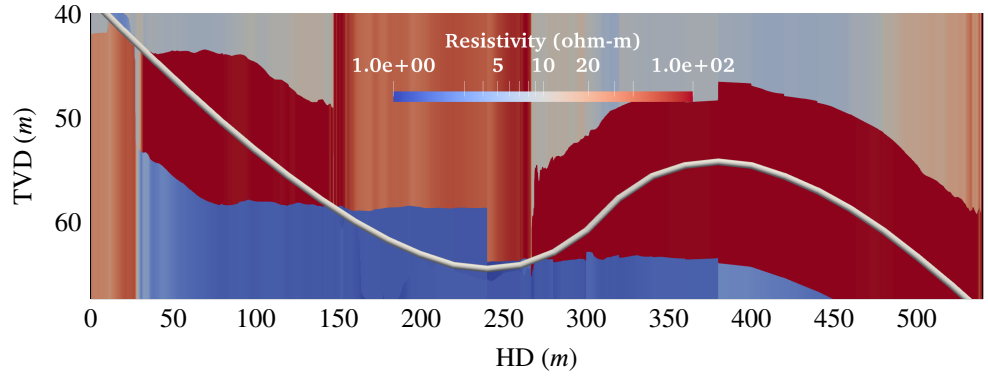
FIGURE 26 Model problem 2. Comparison between $\mathcal{F}\circ\mathcal{I}$ and $\mathcal{F}_{\phi^*}\circ\mathcal{I}$ with regularization using the inversion strategy of Example B.1, i.e., with input measurements corresponding to one logging positions per sample.

8.3.3 | Model Problem III

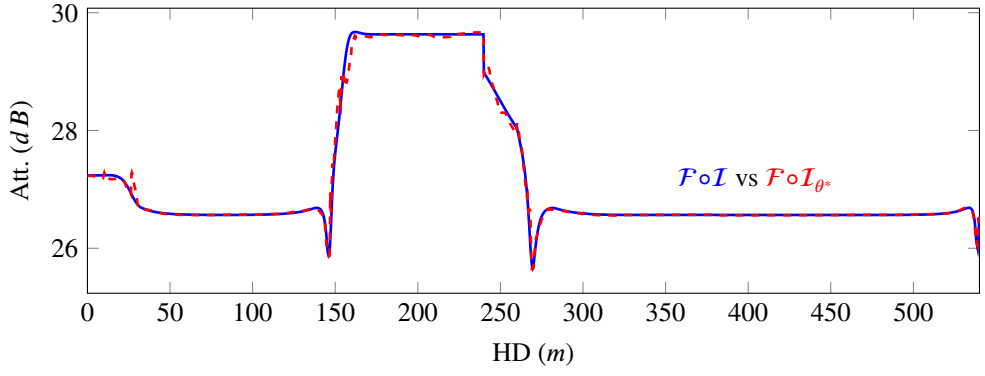
We now consider a model formation exhibiting geological faults and two different well trajectories. For well trajectory 1, Figure 27 shows the model problem, logging trajectory, inversion results, and coaxial attenuation logs. Inversion results are excellent. When considering the second well trajectory shown in Figure 28, we observe good inversion results except at the proximity of points with horizontal distance (HD) equals to 75m and 350m. These inaccurate inversion results are easily identified by examination of the corresponding logs.



(a) Actual formation

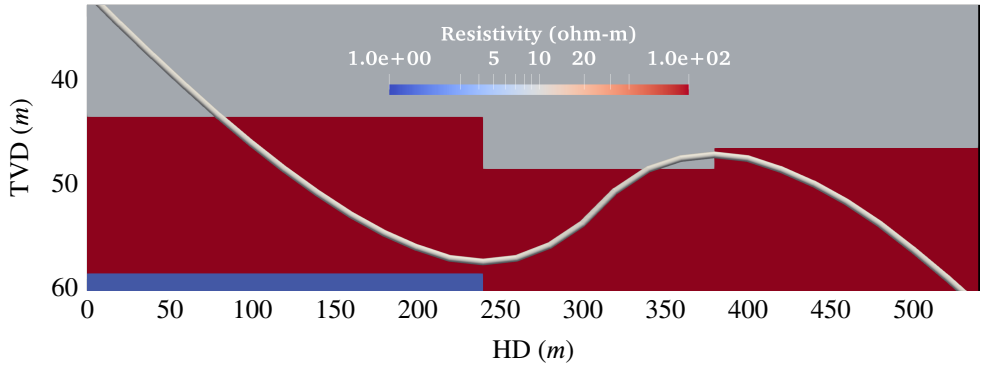


(b) Predicted formation

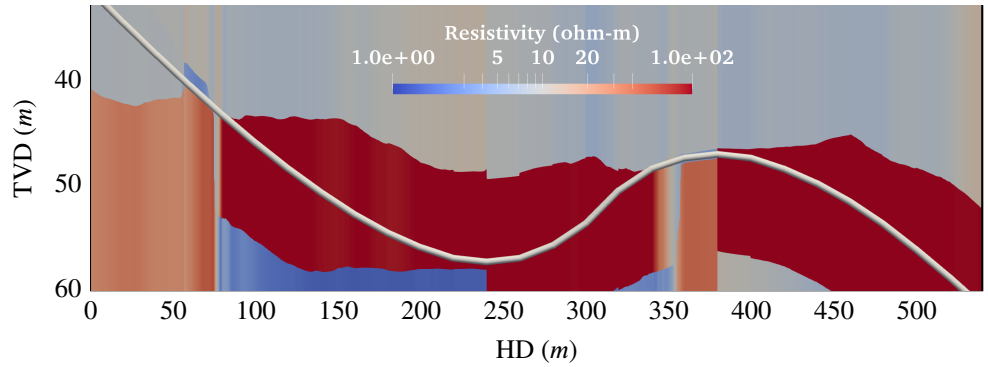


(c) LWD coaxial measurement

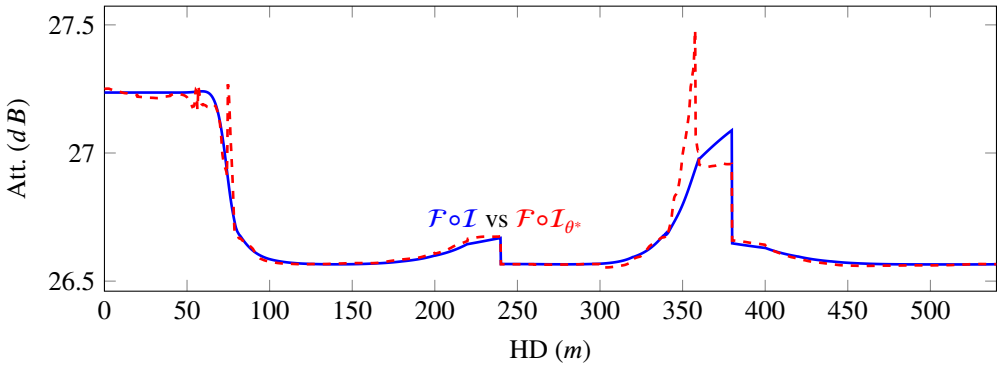
FIGURE 27 Model problem III, trajectory 1. Comparison between actual and predicted formations and the corresponding coaxial logs with regularization using the inversion strategy of Example B.1, i.e., with input measurements corresponding to one logging positions per sample.



(a) Actual formation



(b) Predicted formation



(c) LWD coaxial measurement

FIGURE 28 Model problem III, Trajectory 2. Comparison between actual and predicted formations and the corresponding coaxial logs with regularization using the inversion strategy of Example B.1, i.e., with input measurements corresponding to one logging positions per sample.

9 | DISCUSSION AND CONCLUSIONS

In this work, we focus on the use of deep neural networks (DNN) for the inversion of borehole resistivity measurements for geosteering applications. We analyze the strong impact that different loss functions have on the prediction results. We illustrate

via a simple benchmark example that a traditional data misfit loss function delivers poor results. As a remedy, we use an Encoder-Decoder or a two-step loss function. These approaches generate two DNN approximations: one for the forward function and another one for the inverse operator. We propose different neural network architectures for each approximation functions.

To guarantee that the inverse DNN approximation provides meaningful results, we need to ensure that the training dataset contains sufficient samples. Otherwise, both forward and inverse DNN operators may provide incorrect solutions while still ensuring the composition of both operators to be close to the identity. Thus, the approach is highly dependent on the existence of a sufficiently rich training dataset, which facilitates the learning process of the DNNs. In the case of 1D layered formations, it is often feasible to produce the required dataset. However, for more complicated cases, for example, the inversion of 2D and 3D geometries, a direct extension may be limited due to the larger number of inversion variables and the extremely time-consuming process of producing an exhaustive dataset.

As a partial remedy for this limitation, we find it highly beneficial to add a regularization term to the loss function based on the existing training dataset. This reduces the richness we need to guarantee within the training datasets. Nevertheless, such regularization terms may hide alternative feasible solutions for the inverse operator, which may provide excessive confidence on the results and minimize the capacity to perform a fair uncertainty quantification assessment. Another possibility to partially alleviate the aforementioned problem is to consider a two-step loss functions. Using this approach, we have shown that the inverse problem considered in this work admits different solutions that are physically feasible, a fact that was obscured when using the regularization term.

Other critical limitations of DNNs we encounter in this work are: (a) the limited approximation capabilities of DNNs to reproduce discontinuous functions, (b) the need of a new dataset and trained DNN for each subsurface parametrization, and (c) the poor results they exhibit when they are evaluated over a sample that is outside the training dataset space. More importantly, it is often difficult to identify the source of poor results, which may include inadequate selections of: (i) loss function, (ii) DNN architecture, (iii) regularization term, (iv) training dataset, (v) optimization algorithm, (vi) rescaling operator and norms, (vii) model parameterization, (viii) approximation capabilities of DNNs, or simply (ix) the nature of the problem due to a lack of adequate measurements. To deal with the aforementioned limitations, we propose a careful step-by-step error control based on: (a) selecting adequate norms, (b) proper rescaling of the variables, (c) selecting a well suited loss function possibly with a regularization term, (d) analyzing the evolution of the different terms of the loss function, (e) studying multiple cross-plots of different nature, and (f) performing an in-depth assessment of the results over multiple realistic test examples.

Finally, we show it is possible to obtain a good-quality inversion of geosteering measurements with limited *online* computational cost, thus, suitable for real-time inversion. Moreover, the quality of the inversion results can be rapidly evaluated to detect its possible inaccuracies in the field and select alternative inversion methods when needed.

Possible future research lines of this work include: (a) to study different DNN architectures when applied to these problems, for example, using automatic DNN architecture generators such as Automated Machine Learning techniques, (b) to design proper measurement acquisition systems and adequate Earth model parametrizations using the cross-plots delivered by the DNNs, (c) to consider more complex Earth models, possibly containing geological faults or other relevant subsurface features, (d) to develop optimal sampling techniques for inverse problems, possibly containing a different number of samples to train the forward and inverse operators, (e) to design and analyse new regularization techniques, (f) to use Bayesian DNNs for uncertainty quantification, and (g) to use transfer learning techniques for higher spatial dimensions, which can alleviate data requirements to train the corresponding DNNs. Finally, a natural step toward industrial applications is to evaluate the performance of our DNN approach when having noisy measurements. As mentioned above, we shall use our approach to design proper measurement acquisition techniques and adequate earth model parameterizations using the cross-plots delivered by the DNNs.

10 | ACKNOWLEDGMENTS

The research reported in this article has been funded by the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 777778 (MATHROCKS), the European POCTEFA 2014-2020 Project PIXIL (EFA362/19) by the European Regional Development Fund (ERDF) through the Interreg V-A Spain-France-Andorra programme, the Austrian Ministry for Transport, Innovation and Technology (BMVIT), the Federal Ministry for Digital and Economic Affairs (BMDW), the Province of Upper Austria in the frame of the COMET - Competence Centers for Excellent Technologies Program managed by Austrian Research Promotion Agency FFG, the COMET Module S3AI, the Project of the Spanish Ministry of Economy and Competitiveness with reference MTM2016-76329-R (AEI/FEDER, EU), the BCAM

"Severo Ochoa" accreditation of excellence (SEV-2017-0718), and the Basque Government through the BERC 2018-2021 program, the two Elkartek projects ArgIA (KK-2019-00068) and MATHEO (KK-2019-00085), the Consolidated Research Group MATHMODE (IT1294-19) given by the Department of Education, The University of Texas at Austin Research Consortium on Formation Evaluation, jointly sponsored by Anadarko, Aramco, Baker Hughes, BHP, BP, Chevron, China Oilfield Services Limited, CNOOC International, ConocoPhillips, DEA, Eni, Equinor ASA, ExxonMobil, Halliburton, INPEX, Lundin Norway, Occidental, Oil Search, Petrobras, Repsol, Schlumberger, Shell, Southwestern, Total, Wintershall Dea, and Woodside Petroleum Limited. Carlos Torres-Verdín is grateful for the financial support provided by the Brian James Jennings Memorial Endowed Chair in Petroleum and Geosystems Engineering. This publication acknowledges the financial support of the CSIRO Professorial Chair in Computational Geoscience at Curtin University and the Deep Earth Imaging Enterprise Future Science Platforms of the Commonwealth Scientific Industrial Research Organisation, CSIRO, of Australia. Additionally, at Curtin University, The Institute for Geoscience Research (TIGeR) and by the Curtin Institute for Computation, kindly provide continuing support.

REFERENCES

1. Lu L, Zheng Y, Carneiro G, Yang L. *Deep Learning for Computer Vision: Expert techniques to train advanced neural networks using TensorFlow and Keras*. Springer, Switzerland . 2017.

2. Yu D, Deng L. *Automatic Speech Recognition: A Deep Learning approach*. Springer, London . 2017.

3. Bhanu B, Kumar A. *Deep Learning for Biometrics*. Springer, Switzerland . 2017.

4. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. *arXiv:1512.03385* 2015.

5. Badrinarayanan V, Kendall A, Cipolla R. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2017; 39(12): 2481–2495.

6. Chollet F. Keras. https://github.com/fchollet/keras; 2015.

7. Paszke A, Gross S, Chintala S, et al. Automatic differentiation in PyTorch. In: ; 2017.

8. Zhao R, Yan R, Chen Z, Mao K, Wang P, Gao RX. Deep learning and its applications to machine health monitoring. *Mechanical Systems and Signal Processing* 2019; 115: 213 - 237. doi: https://doi.org/10.1016/j.ymssp.2018.05.050

9. Deng L, Li J, Huang J, et al. Recent advances in deep learning for speech research at Microsoft. In: ; 2013: 8604-8608.

10. Längkvist M, Karlsson L, Loutfi A. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters* 2014; 42: 11 - 24. doi: https://doi.org/10.1016/j.patrec.2014.01.008

11. Vargas R, Mosavi A, Ruiz R. Deep learning: A review. working paper, ; : 2017.

12. Jan B, Farman H, Khan M, et al. Deep learning in big data Analytics: A comparative study. *Computers & Electrical Engineering* 2019; 75: 275 - 287. doi: https://doi.org/10.1016/j.compeleceng.2017.12.009

13. Shahriari M, Pardo D, Picón A, Galdran A, Ser JD, Torres-Verdín C. A Deep Learning Approach to the Inversion of Borehole Resistivity Measurements. *Computational Geosciences* 2020. doi: 10.1007/s10596-019-09859-y

14. Pardo D, Torres-Verdin C. Fast 1D inversion of logging-while-drilling resistivity measurements for the improved estimation of formation resistivity in high-angle and horizontal wells. *Geophysics* 2014; 80 (2): E111–E124.

15. Ijasana O, Torres-Verdín C, Preeg WE. Inversion-based petrophysical interpretation of logging-while-drilling nuclear and resistivity measurements. *Geophysics* 2013; 78 (6): D473–D489.

16. Desbrandes R, Clayton R. Chapter 9 measurement while drilling. *Developments in Petroleum Science* 1994; 38: 251 – 279.

17. Ghasemi M, Yang Y, Gildin E, Efendiev Y, Calo VM. Fast multiscale reservoir simulations using POD-DEIM model reduction. *Society of Petroleum Engineers* 2015: 1–18.

18. Chemali R, Bittar M, Hveding F, Wu M, Dautel M. Improved geosteering by integrating in real time images from multiple depths of investigation and inversion of azimuthal resistivity signals. *Society of Petrophysicists and Well-Log Analysts* 2010: 1–7.

19. Dupuis C, Denichou JM. Automatic inversion of deep-directional-resistivity measurements for well placement and reservoir description. *The Leading Edge* 2015; 34(5): 504–512.

20. Zhang Z, Yuan N, Liu CR. 1-D Inversion of triaxial induction logging in layered anisotropic formation. *Progress In Electromagnetics Research B* 2012; 44: 383–403.

21. Seifert DJ, Dossary SA, Chemali RE, et al. Deep electrical images, geosignal, and real-time inversion help guide steering decisions. *Society of Petroleum Engineers* 2009: 1–9.

22. Tarantola A. *Inverse Problem Theory and Methods for Model Parameter Estimation*. Society for Industrial and Applied Mathematics . 2005.

23. Shahriari M, Rojas S, Pardo D, et al. A numerical 1.5D method for the rapid simulation of geophysical resistivity measurements. *Geosciences* 2018; 8(6): 1–28.

24. Wang L, Li H, Fan Y. Bayesian Inversion of Logging-While-Drilling Extra-Deep Directional Resistivity Measurements Using Parallel Tempering Markov Chain Monte Carlo Sampling. *IEEE Transactions on Geoscience and Remote Sensing* 2019; 57(10): 8026-8036.

25. Malinverno A, Torres-Verdín C. Bayesian inversion of DC electrical measurements with uncertainties for reservoir monitoring. *Inverse Problems* 2000; 16(5): 1343–1356. doi: 10.1088/0266-5611/16/5/313

26. Gunning J, Glinsky ME. Detection of reservoir quality using Bayesian seismic inversion. *GEOPHYSICS* 2007; 72(3): R37-R49. doi: 10.1190/1.2713043

27. Vogel C. *Computational Methods for Inverse Problems*. Society for Industrial and Applied Mathematics . 2002.

28. Higham CF, Higham DJ. Deep learning: An introduction for applied mathematicians. *Computing Research Repository* 2018; abs/1801.05894.

29. Shahriari M, Pardo D, Moser B, Sobieczky F. A Deep Neural Network as Surrogate Model for Forward Simulation of Borehole Resistivity Measurements. *Procedia Manufacturing* 2020; 42: 235 - 238. International Conference on Industry 4.0 and Smart Manufacturing (ISM 2019).

30. Alvarez-Aramberri J, Pardo D. Dimensionally adaptive hp-finite element simulation and inversion of 2D magnetotelluric measurements. *Journal of Computational Science* 2017; 18: 95–105.

31. Bakr SA, Pardo D, Mannseth T. Domain decomposition Fourier FE method for the simulation of 3D marine CSEM measurements. *J. Comput. Phys.* 2013; 255: 456–470.

32. Davydycheva S, Wang T. A fast modelling method to solve Maxwell's equations in 1D layered biaxial anisotropic medium. *Geophysics* 2011; 76 (5): F293–F302.

33. Davydycheva S, Homan D, Minerbo G. Triaxial induction tool with electrode sleeve: FD modeling in 3D geometries. *Journal of Applied Geophysics* 2004; 67: 98–108.

34. Puzyrev V. Deep learning electromagnetic inversion with convolutional neural networks. *Geophysical Journal International* 2019; 218(2): 817-832.

35. Moghadas D. One-dimensional deep learning inversion of electromagnetic induction data using convolutional neural network. *Geophysical Journal International* 2020; 222(1): 247-259.

36. Loseth LO, Ursin B. Electromagnetic fields in planarly layered anisotropic media. *Geophysical Journal International* 2007; 170: 44–80.

37. Jin Y, Wu X, Chen J, Huang Y, others . Using a Physics-Driven Deep Neural Network to Solve Inverse Problems for LWD Azimuthal Resistivity Measurements. In: Society of Petrophysicists and Well-Log Analysts. ; 2019.

38. Ursin B, Stovas A. Reflection and transmission responses of a layered isotropic viscoelastic medium. *GEOPHYSICS* 2002; 67(1): 307-323. doi: 10.1190/1.1451803

39. Anderson WL. A hybrid fast Hankel transform algorithm for electromagnetic modeling. *GEOPHYSICS* 1989; 54(2): 263-266. doi: 10.1190/1.1442650

40. Abadi M, Agarwal A, Barham P, et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems.; 2015. Software available from tensorflow.org.

41. Quan TM, Hildebrand DGC, Jeong WK. FusionNet: A deep fully residual convolutional neural network for image segmentation in connectomics.; 2016.

42. Krizhevsky A, Sutskever I, Hinton GE. ImageNet Classification with Deep Convolutional Neural Networks. In: Pereira F, Burges CJC, Bottou L, Weinberger KQ., eds. *Advances in Neural Information Processing Systems 25*Curran Associates, Inc. 2012 (pp. 1097–1105).

43. Jin KH, McCann MT, Froustey E, Unser M. Deep Convolutional Neural Network for Inverse Problems in Imaging. *IEEE Transactions on Image Processing* 2017; 26(9): 4509-4522.

44. Wiatowski T, Bölcskei H. A Mathematical Theory of Deep Convolutional Neural Networks for Feature Extraction. *IEEE Transactions on Information Theory* 2018; 64(3): 1845-1866.

45. Thévenaz P, Blu T, Unser M. Chapter 28-Image Interpolation and Resampling. In: Bankman I. , ed. *Handbook of MedicalImage Processing and Analysis, 2nd ed*. 4. MA, USA: Academic Press: Cambridge. 2009 (pp. 465–493).

46. Parker JA, Kenyon RV, Troxel DE. Comparison of Interpolating Methods for Image Resampling. *IEEE Transactions on Medical Imaging* 1983; 2(1): 31-39.

How to cite this article: M. Shahriari, D. Pardo, J. A. Rivera, C. Torres-Verdín, A. Picon, J. Del Ser, S. Ossandón, and V. M. Calo (2020), Error Control and Loss Functions for the Deep Learning Inversion of Borehole Resistivity Measurements, ,

APPENDIX

A DDN ARQUITECTURES

To approximate the forward and the inverse problems, we use DNN architectures based on residual-type blocks[4,41] with convolutional operators[28,42,43,44]. In the following, we first define the main operators of our DNN architectures, followed by a description of the forward and inverse DNN architectures.

We denote by $\mathcal{N}$ to our nonlinear *activation function*. In our case, we employ the *rectified linear unit (ReLU)*, defined component-wise for each entry x as $\max(0, x)$[28]. We now introduce a 1D convolutional operator $\mathbf{C}^{c,k}_{\psi}$, where c is the filter size (output dimensionality), k the kernel size, and ψ the weights[28,4]. Then, we define the following block:

$$\mathbf{B}^{c,k}_{\psi} := \left(\mathcal{N}\circ\mathbf{C}^{c,k}_{\psi^1}\circ\mathcal{N}\circ\mathbf{C}^{c,k}_{\psi^2} + \mathbf{C}^{c,k}_{\psi^3}\right), \tag{A1}$$

where now $\psi = (\psi^1, \psi^2, \psi^3)$ are all weights associated to block $\mathbf{B}^{c,k}_{\psi}$. We now define the following operators:

- A fully-connected layer $\mathbf{D}^{n}_{\psi}$ with n being its number of units and ψ its weights[28,40].
- A 1D upsampling operator $\mathbf{U}$ with upsampling factor equal to two (using the *TensorFlow* routine upsampling1D[40,45,46]).
- A bilinear resampling operator $\mathbf{L}$ with resampling factor equal to the number of logging positions[45,46], i.e., 1 for Example B.1 and 65 for Example B.2.
- A flattening layer $\mathbf{S}$ that receives a 2D matrix and outputs a 1D vector[40].

A.1 Forward Problem DNN Architecture

Each input sample has dimension $n_p + n_t$, and contains the variables representing the material properties and the trajectory. We define our DNN architecture as:

$$\mathcal{F}_{\mathcal{R},\phi} := \mathcal{N} \circ \mathbf{C}_{\phi_6}^{c_6,1} \circ \mathbf{L} \circ \mathbf{U} \circ \mathbf{B}_{\phi_5}^{c_5,3} \circ \mathbf{U} \circ \mathbf{B}_{\phi_4}^{c_4,3} \circ \cdots \circ \mathbf{U} \circ \mathbf{B}_{\phi_1}^{c_1,3}, \tag{A2}$$

where $c_i := 40i$, for $i = 1, \cdots, 5$, $c_6 = n'_m = 6$, where n'_m is the number of evaluated measurements per logging position, and $\phi = \{\phi_i : \ i = 1, \cdots, 6\}$ is a set of all weights associated to the forward DNN architecture. $\mathbf{L}$ expands (in case of 65 logging positions) or shrinks (in case of one logging position) its input dimension. The output of the mentioned bilinear resampling is a matrix in which its first dimension is equal to the number of logging positions[45,46]. All the resampling operators considered in the Equation A3 raise/shrink the dimension of their input gradually to avoid missing information due to a sudden dimension change. The output is a matrix of dimension (n_l, n'_m), where n_l is the number of logging positions.

A.2 Inverse Problem DNN Architecture

The input of the DNN is a matrix of dimension $(n_l, n'_m + 2)$, where n_l is the number of logging positions. The first two columns of the aforementioned matrix are the sine and cosine of the trajectory dip angle at each logging position. Analogously to the forward problem, we consider the following architecture:

$$\mathcal{F}_{\mathcal{R},\phi} := \mathcal{N} \circ \mathbf{D}_{\theta_7}^{n_p} \circ \mathbf{S} \circ \mathbf{B}_{\theta_6}^{c_6,3} \circ \mathbf{B}_{\theta_5}^{c_5,3} \circ \mathbf{B}_{\theta_4}^{c_4,3} \circ \cdots \circ \mathbf{B}_{\theta_1}^{c_1,3}, \tag{A3}$$

where $c_i := 40i$, for $i = 1, \cdots, 5$, and $\theta = \{\theta_i : \ i = 1, \cdots, 7\}$ is a set of all the weights associated to each block and layer. $\mathbf{D}_{\theta_7}^{n_p}$ performs the ultimate feature extraction and down-sampling. The output of this DNN is a vector consists of material properties.