Geophysical Journal International

Geophys. J. Int. (2023) **234,** 2487–2500
Advance Access publication 2023 June 23
GJI General Geophysical Methods

https://doi.org/10.1093/gji/ggad249

Neural network architecture optimization using automated machine learning for borehole resistivity measurements

M. Shahriari,[1,2] D. Pardo,[3,4,5] S. Kargaran[2] and T. Teijeiro[3,4]

[1] *GE Healthcare Austria GmbH, Tiefenbach* 15, 4871 *Zipf, Austria. E-mail: m.shahriari.sh@gmail.com*
[2] *Software Competence Center Hagenberg GmbH (SCCH), Softwarepark* 32*a,* 4232 *Hagenberg, Austria*
[3] *Department of Mathematics, University of the Basque Country (UPV/EHU), Leioa* 48940*, Spain*
[4] *Basque Center for Applied Mathematics, (BCAM),* 48009 *Bilbao, Spain*
[5] *Ikerbasque (Basque Foundation for Sciences),* 48009 *Bilbao, Spain*

Accepted 2023 June 13. Received 2023 May 6; in original form 2022 July 19

SUMMARY

Deep neural networks (DNNs) offer a real-time solution for the inversion of borehole resistivity measurements to approximate forward and inverse operators. Using extremely large DNNs to approximate the operators is possible, but it demands considerable training time. Moreover, evaluating the network after training also requires a significant amount of memory and processing power. In addition, we may overfit the model. In this work, we propose a scoring function that accounts for the accuracy and size of the DNNs compared to a reference DNNs that provides good approximations for the operators. Using this scoring function, we use DNN architecture search algorithms to obtain a quasi-optimal DNN smaller than the reference network; hence, it requires less computational effort during training and evaluation. The quasi-optimal DNN delivers comparable accuracy to the original large DNN.

Key words: Inverse theory; Machine learning; Neural networks, fuzzy logic; Downhole method; Wave propagation.

1 INTRODUCTION

Oil and gas companies use geosteering to increase the productivity of their wells (Beer *et al.* 2010; Bittar & Aki 2015). In this application, a logging-while-drilling (LWD) instrument helps us to navigate the well trajectory inside the oil reservoir to maximize its production. A LWD instrument incorporates transmitters and receivers, in our case, electromagnetic (EM) ones (Desbrandes & Clayton 1994; Spies 1996).

In geosteering, we encounter forward (Shahriari *et al.* 2018, 2020a; Alyaev *et al.* 2021) and inverse problems (Shahriari *et al.* 2020b). Some traditional methods to solve inverse problems include gradient-based and statistics-based approaches (Malinverno & Torres-Verdín 2000; Tarantola 2005; Watzenig 2007; Ijasana *et al.* 2013; Pardo & Torres-Verdín 2014; Jahani *et al.* 2022). Artificial intelligence (AI) algorithms, particularly deep learning (DL), have recently become popular to solve inverse problems (Jin *et al.* 2019b; Puzyrev 2019; Shahriari *et al.* 2020c, b; Moghadas 2020; Hu *et al.* 2020; Rammay *et al.* 2022). In this work, we use a deep neural network (DNN) to approximate the solution of an inverse problem.

The main difficulty when solving an inverse problem arises due to the non-uniqueness of its solution, that is when there exist multiple outputs for each input (Tarantola 2005). It turns out that the DNN approximation may become an average of all the existing solutions, which can be far from any of them. To partially overcome this problem and obtain one of the possible existing solutions, in (Shahriari *et al.* 2020c), we proposed a specific loss function based on the misfit of the measurements that incorporates both the inverse and forward solutions. To recover a set of solutions, one would need to use more sophisticated methods like Bayesian DNNs (Snoek *et al.* 2012), or mixture density networks (Alyaev & Elsheikh 2022), which are out of the scope of this work.

Designing DNN architectures by hand is difficult (Goodfellow *et al.* 2016; Higham & Higham 2019). An excessively large DNN may achieve the required accuracy, but we may incur in excessive computational costs and possibly in overfitting. On the other side, using a small DNN may limit the accuracy. Here, we use automated machine learning (AutoML) algorithms (Hutter *et al.* 2019; He *et al.* 2021). More precisely, we make use of DNN architecture search algorithms (O'Malley *et al.* 2019; Jin *et al.* 2019a; Elsken *et al.* 2019) to build quasi-optimal DNN architectures, where the optimal DNN architecture belongs to a specific search space defined by the user *a priori*. We perform this optimization process by balancing the size and accuracy of the networks. These search techniques allow us to find well-suited DNNs with limited knowledge about the DNN architectures.

In this work, we have considered as a reference model the DNN described in Shahriari *et al.* (2022), and we have evaluated the explored architectures by assessing the error variation and the number

of trainable parameters. This is tightly related to model compression (Cheng *et al.* 2018), a research field that explores methods for reducing the complexity of a reference model without affecting its accuracy. In particular, our approach can be linked to knowledge distillation (Ba & Caruana 2014), a family of techniques consisting of training a more compact model that integrates the output of the reference model in the loss function. In our case, we rely on AutoML for exploring the space of compact models. The resulting DNN architecture is a parametric representation of the desired operator, that is forward and inverse. Although DNN architecture search algorithms are time-intensive, having a smaller model makes it computationally cheaper to train a DNN on new data sets (new scenarios). Moreover, smaller DNNs require less memory to save and fewer computational resources to evaluate. Therefore, saving and evaluating on portable devices, for example LWD instruments, is more versatile.

In this work, we propose a search space of DNN architectures based on convolutional blocks. Moreover, we introduce a scoring function that accounts for both the loss and size of the DNN. By minimizing this scoring function, we find the optimal DNN within the selected search space. We use two standard architecture search algorithms: random and Bayesian searches (Elsken *et al.* 2019). We compare the results of the obtained DNN versus our original DNN designed by hand. Results show that the AutoML DNN algorithms deliver smaller DNN networks that preserve the accuracy of the larger DNN created by hand. Throughout this work, we consider noise-free synthetic measurements. Nonetheless, we expect smaller DNNs to be more resilient towards noise than large DNNs that are more prone to overfitting.

The remaining of this work is organized as follows: Section 2 defines the forward and inverse problems in the inversion of borehole resistivity measurements. Section 3 describes a two-step training strategy that we use in this work to obtain the DNN approximation of the inverse operator. Section 4 discusses the considered DNN architecture components, the definition of the scoring function, and the DNN architecture search algorithms that we use in this work. Section 5 verifies the proposed techniques by showing the training results and the model's output for some synthetic models. Section 6 is dedicated to the conclusion.

2 PROBLEM DEFINITION

Let $\mathbf{p}$ be the subsurface properties. In this application, since the inversion should be performed in real-time, to reduce the computational complexity of the problem it is common to consider a 1D-layered formation around the logging position (Davydycheva & Wang 2011; Pardo & Torres-Verdín 2014; Shahriari *et al.* 2018). Hence, $\mathbf{p}$ is a vector of variables parametrizing the 1D formation as follows:

$$\mathbf{p} = (\rho_c, \rho_u, \rho_l, d_u, d_l), \tag{1}$$

where ρ_c is the resistivity of the host (central) layer of the logging instrument, ρ_u and ρ_l are the resistivities of the upper and lower layers, respectively, and d_u and d_l are the vertical distances to the upper and lower bed boundaries, respectively. Fig. 1 shows the Earth subsurface parametrization and corresponding variation intervals selected based on their occurrence on the geological targets (Pardo & Torres-Verdín 2014; Shahriari *et al.* 2020b; Shahriari & Pardo 2020).

The selection of the measurements is of paramount importance to: (1) avoid using redundant or highly correlated measurements,

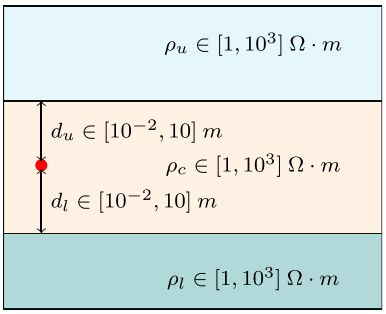


Figure 1. 1D subsurface formation, its parametrization and the range of variation of the parameters. The logging position indicated by the red circle. ρ_u, ρ_c and ρ_l are the resistivities of the upper, central and lower layers, respectively. d_u and d_l are vertical distances from the current logging position to the upper and lower bed boundaries, respectively.

which are unnecessary and (2) be able to obtain an efficient DNN approximation of the forward and the inverse operators. Shahriari *et al.* (2022) describes an automatic algorithm to select seven complex-valued measurements from a pool of 45 measurements obtained with two logging instruments shown in Fig. 2: a conventional LWD and an azimuthal one. Each logging instrument incorporates various transmitter-receiver spacings. Let $\mathbf{m}$ be the measurements obtained at the receivers using the aforementioned logging instruments and the algorithm described in Shahriari *et al.* (2022). Table 1 describes these measurements for each transmitter–receiver set. For each measurement shown in Table 1, we obtain a real and an imaginary part, except for the geosignal as the imaginary part is discontinuous (see Shahriari *et al.* 2022, for more details). Therefore, $\mathbf{m}$ contains 13 real numbers per logging position. Table 2 defines the type of measurements that we consider in this work. Moreover, we consider high-angle (almost horizontal) trajectories, hence, for the case of trajectory dip angle, we have $t \in [83^\circ, 97^\circ]$. Then, we have the following separate problems:

(i) Forward problem: Given $\mathbf{p}$ and t, we obtain $\mathbf{m}$ at the receivers, that is $\mathcal{F}(\mathbf{p}, t) = \mathbf{m}$, where $\mathcal{F}$ is the solution of Maxwell's equations with a zero Dirichlet boundary condition far away from the transmitters (Davydycheva *et al.* 2004; Davydycheva & Wang 2011; Shahriari *et al.* 2018; Shahriari & Pardo 2020; Shahriari *et al.* 2020a; Alyaev *et al.* 2021). Hence, given the aforementioned definitions of material properties, trajectory dip angle and measurements, the forward function has an input dimension of 6 and an output dimension of 13.

(ii) Inverse problem: Given the measurements acquired at the receivers and the trajectory dip angle, the inverse operator $\mathcal{I}$ delivers the subsurface properties, that is $\mathcal{I}(\mathbf{m}, t) = \mathbf{p}$ (Ijasana *et al.* 2013; Pardo & Torres-Verdín 2014; Shahriari *et al.* 2020b, c; Jin *et al.* 2019b). Therefore, the input and output sizes of the inverse operator are 14 (considering the measurement and trajectory dip angle) and 5, respectively.

In this work, we use DNNs to approximate the forward function $\mathcal{F}$ and inverse operator $\mathcal{I}$. Training a DNN requires a large data set. Hence, given the above subsurface parametrization, trajectory, and measurements, we produce a data set of 300 000 randomly selected samples using a fast semi-analytic solver (Loseth & Ursin 2007). We then express the values of the subsurface properties in the logarithmic scale (Shahriari *et al.* 2020c), and rescale all the variables (i.e. subsurface properties in the logarithmic scale and the measurements) to the interval [0.5,1.5] (see Shahriari *et al.* 2022, for details).

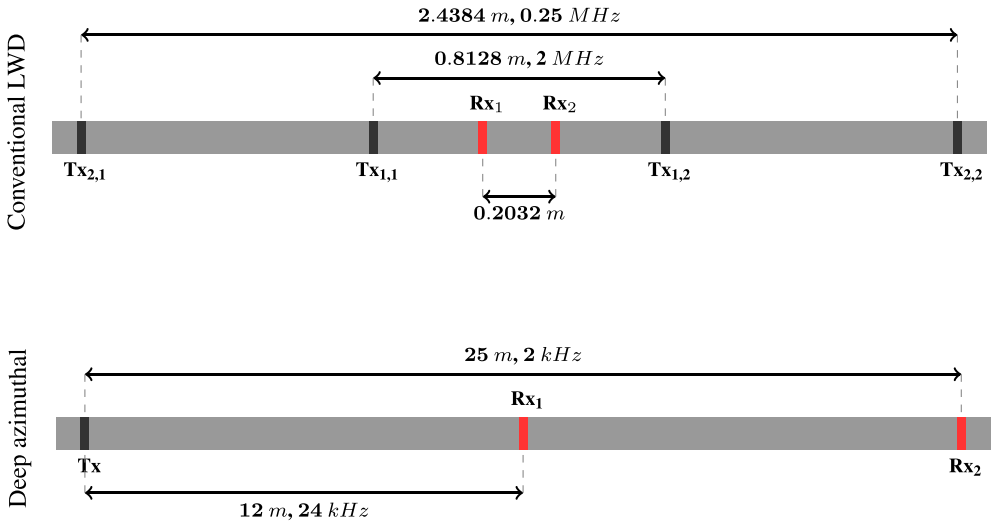


Figure 2. LWD instruments. **Tx**, and $\mathbf{Tx_{i,j}}$, **i**, **j**=1,2, denote the transmitters. $\mathbf{Rx_1}$, $\mathbf{Rx_2}$ are the receivers.

Table 1. Evaluated measurements for each transmitter–receiver set.

Transmitter–receiver	Measured component
$(Tx_{1,1}, Tx_{1,2}, Rx_1, Rx_2)$	zz, yy, Geosignal, symmetrized directional
$(Tx_{2,1}, Tx_{2,2}, Rx_1, Rx_2)$	Symmetrized directional
(Tx, Rx_1)	Symmetrized directional
(Tx, Rx_2)	zz

Table 2. Definition of each type of measurement. H_{ij} is the complex-valued magnetic field, where i and j indicate the orientations of transmitters and receivers, respectively.

Name	Measurement definition
zz	H_{zz}
yy	H_{yy}
Geosignal	$\frac{H_{zz} - H_{zx}}{H_{zz} + H_{zx}}$
Symmetrized directional	$\frac{H_{zz} + H_{zx}}{H_{zz} - H_{zx}} \cdot \frac{H_{zz} - H_{xz}}{H_{zz} + H_{xz}}$

3 TWO-STEP TRAINING STRATEGY

Due to the non-uniqueness of the inverse operator's output, using conventional loss functions (based on the misfit of the inversion variables **p**) may produce an inaccurate solution (Tarantola 2005; Shahriari *et al.* 2020c). To guarantee that the trained DNN delivers one of the true solutions of the inverse operator, we use a two-step training strategy, as described in Shahriari *et al.* (2020c). First, we approximate the forward function $\mathcal{F}$ as:

$$\mathcal{F}_{\alpha^*} := \arg\min_{\alpha} \sum_{i=1}^{n_t} L(\mathcal{F}_{\alpha}(t_i, \mathbf{p}_i), \mathbf{m}_i), \tag{2}$$

where for given vectors $\mathbf{x}$ and $\mathbf{y}$, we define $L(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_{l_1}$, $\{(\mathbf{p}_i, \mathbf{m}_i, t_i)\}_1^{n_t}$ is the training data set consisting of n_t samples, and α is the set of weights and biases corresponding to the DNN. Then, using the trained DNN approximation of $\mathcal{F}$, we use the following loss function based on the misfit of the measurements to obtain the DNN approximation of the inverse operator:

$$\mathcal{I}_{\beta^*} := \arg\min_{\beta} \sum_{i=1}^{n_t} L(\mathcal{F}_{\alpha^*} \circ \mathcal{I}_{\beta}(t_i, \mathbf{m}_i), \mathbf{m}_i), \tag{3}$$

where β represents the weights and biases corresponding to the DNN, and o indicates the composition of the functions.

4 DNN ARCHITECTURE OPTIMIZATION

4.1 Space of DNN architectures

There exist endless possibilities to select the search space for the DNN architectures. One can select different types of DNN layers/blocks, for example convolutional layers and dense layers, and their corresponding variables, for example kernel size and the number of neurons, and the number of layers/blocks that build the DNN architectures. Beyond that, one can also search for the best training hyperparameters, such as learning rate, which can lead to a more efficient or faster training process. Hence, the search space selection is arbitrary, and there is no concrete approach for selecting the best search space among all the possibilities. Therefore, as it is only guaranteed that the obtained DNN is optimal inside a proposed search space, we call it quasi-optimal DNN architecture. A different selection of the search space will likely lead to a different quasi-optimal DNN architecture, since this choice is not unique.

In this work, we only focus on the architectural components. To have a fair comparison between our quasi-optimal DNN and our previous results, we form convolutional blocks analogous to the ones shown in Shahriari *et al.* (2022). A 1D convolutional layer consists of two main components: (1) kernel size: an integer being the size of the sliding window that moves along the input data during the convolution operation and (2) number of filters: an integer corresponding to the number of output channels produced by the convolutional layer, that is the dimension of the output tensor. Herein, we define the convolutional block B_{k_0,k_1} shown in Fig. 3 as our main architectural component of our DNNs, where k_0 and k_1 are

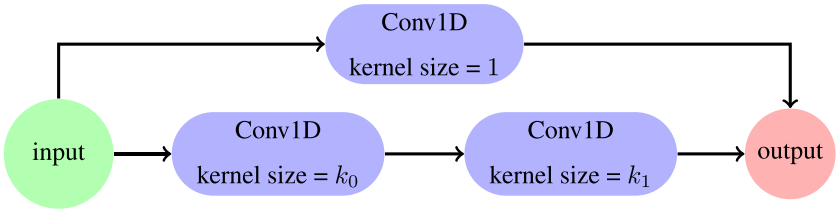
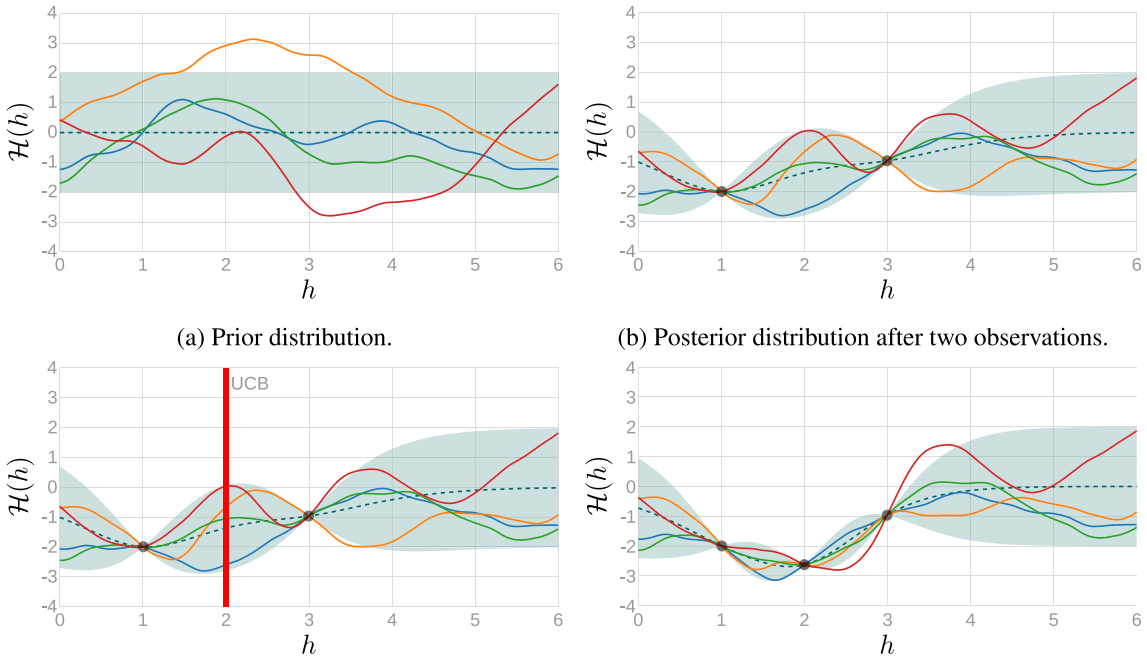


Figure 3. Our convolutional block B_{k_0,k_1} consists of three convolutional layers. k_0 and k_1 are kernel sizes of the convolutional layers that can vary. We consider the number of filters of the convolutional layers to be constant.



(a) Prior distribution.

(b) Posterior distribution after two observations.

(c) Upper Confidence Bound (UCB) criteria for input acquisition.

(d) Distribution after acquiring the selected input.

Figure 4. Illustration of a GP for fitting and optimizing a scoring function with a single parameter. The dashed line is the expected value of the modelled function, while the coloured lines correspond to random sampled functions from the GP distribution. The shaded area represents the 95 per cent confidence interval at each input value (2σ).

Table 3. Comparison of the time required to perform a random search versus a Bayesian approach.

Problem	Random search [hr]	Bayesian approach [hr]
Forward	18.02	16.3
Inverse	5.97	4.80

the kernel sizes of two 1D convolutional layers (He *et al.* 2016). As the tuning process is highly time-intensive, in this work, we do not consider the number of filters as part of our search space to reduce its dimension. We consider $\mathcal{F}_{h_f,\alpha}$ to be the DNN approximation of $\mathcal{F}$ given the set of hyperparameters h_f. Then, for $h_f = \{n, k_0, k_1, l\}$, we define $\mathcal{F}_{h_f,\alpha}$ as:

$$\mathcal{F}_{h_f,\alpha} = B^n_{k_0,k_1} \circ \cdots \circ B^0_{k_0,k_1} \circ C_l, \tag{4}$$

where n is the number of residual blocks and C_l is a 1D convolutional layer with l being its kernel size. We define the search space of hyperparameters as:

$$\mathcal{S}_{\mathcal{F}} = \{n \in \{1, 2, 3, 4\}, k_0, k_1, l \in \{3, 5, 7\}\}. \tag{5}$$

To achieve a symmetric convolution, it is a common practice to consider an odd-size kernel when computing the convolution. In this case, for each architecture, $T = \{t_i = 40 \times i, i \in \{1, \cdots, n\}\}$ are the set of numbers of filters, where t_i is the number of filters of all the convolutional layers in ith block. The number of filters of the final convolutional layer is equal to the number of outputs, that is 13.

Analogously, we consider the DNN approximation of the inverse operator $\mathcal{I}_{h_i,\beta}$ for a given set of hyperparameters $h_i = \{n, k_0, k_1\}$ to be as follows:

$$\mathcal{I}_{h_i,\beta} = B^n_{k_0,k_1} \circ \cdots \circ B^0_{k_0,k_1} \circ r \circ d, \tag{6}$$

where r is a flattening layer that converts a 2D tensor to a 1D one, and d is a fully connected layer with its number of nodes being the size of the vector of subsurface properties $|\mathbf{p}| = 5$. Therefore, our search space is:

$$\mathcal{S}_{\mathcal{I}} = \{n \in \{1, 2, 3, 4, 5\}, k_0, k_1 \in \{3, 5, 7\}\}. \tag{7}$$

Analogous to the forward model, we consider $T = \{t_i = 40 \times i, i \in \{1, \cdots, n\}\}$ to be the set of numbers of filters, where t_i is the number of filters of all the convolutional layers in the i th block.

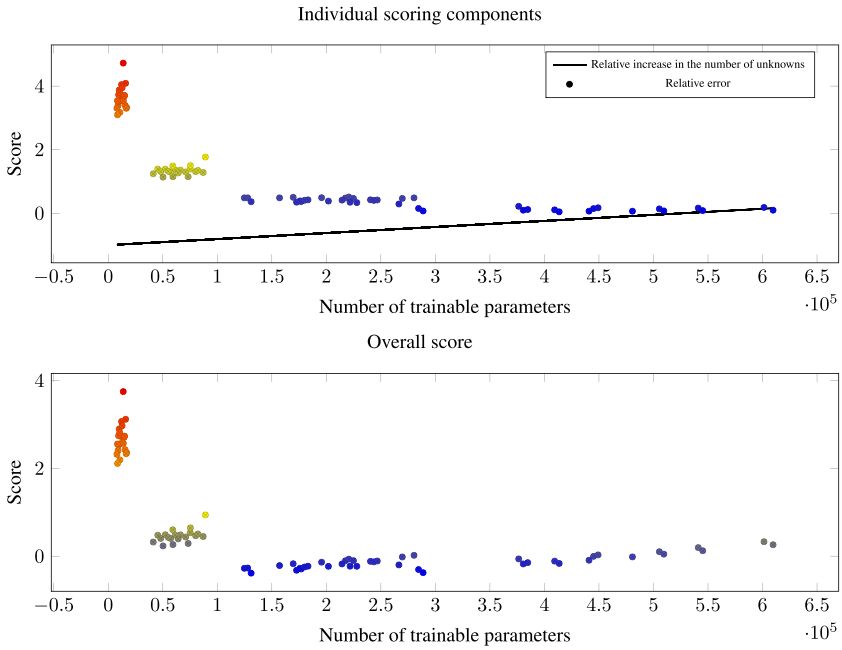


Figure 5. DNN optimization of the forward function $\mathcal{F}$ using a random search. The colours indicate separate clusters of points.

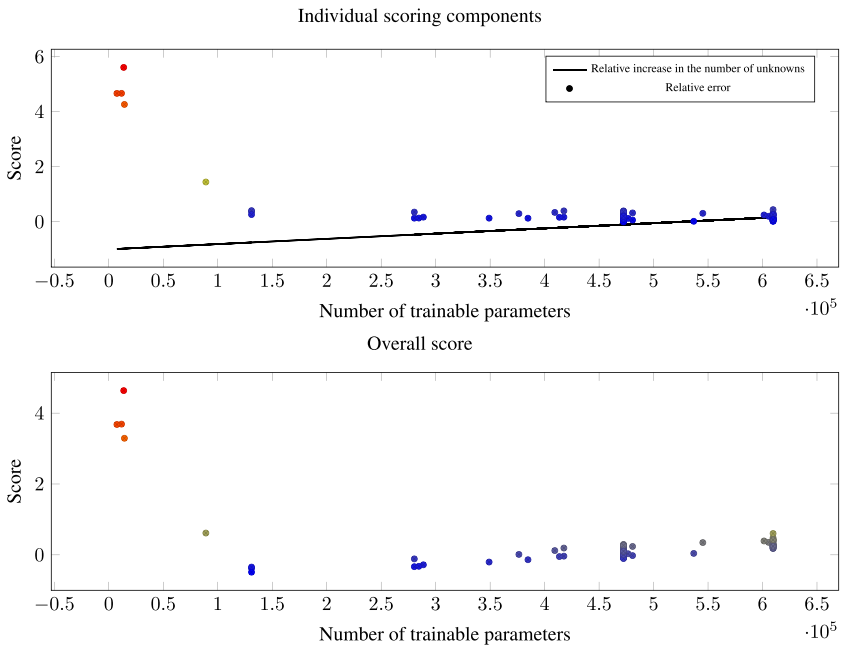


Figure 6. DNN optimization of the forward function $\mathcal{F}$ using a Bayesian approach. The colours indicate separate clusters of points.

4.2 DNN hyperparameter tuning

This work aims to find DNN approximations of $\mathcal{F}$ and $\mathcal{I}$ such that their corresponding architectures use a minimum number of unknowns (weights and biases) and provide comparable (or better) accuracy than the excessively large reference DNN used in Shahriari *et al.* (2022), that corresponds to the hyperparameters h_f^o and h_i^o for the DNN approximations of $\mathcal{F}$ and $\mathcal{I}$, respectively. For a set of hyperparameters $h_f \in \mathcal{S}_{\mathcal{F}}$—see eq. (2)—we train its corresponding DNN defined by eq. (4) to obtain $\mathcal{F}_{h_f,\alpha^*}$. Then, we compute the following scoring function:

$$R_f(h_f) = \underbrace{\frac{\mathcal{H}_f(h_f) - \mathcal{H}_f(h_f^o)}{\mathcal{H}_f(h_f^o)}}_{\text{relative error}} - \underbrace{\frac{N_p(h_f^o) - N_p(h_f)}{N_p(h_f^o)}}_{\substack{\text{relative decrease in} \\ \text{the number of unknowns}}}, \tag{8}$$

where

$$\mathcal{H}_f(h_f) = \sum_{i=1}^{n_v} L(\mathcal{F}_{h_f,\alpha^*}(t_i, \mathbf{p}_i), \mathbf{m}_i) \tag{9}$$

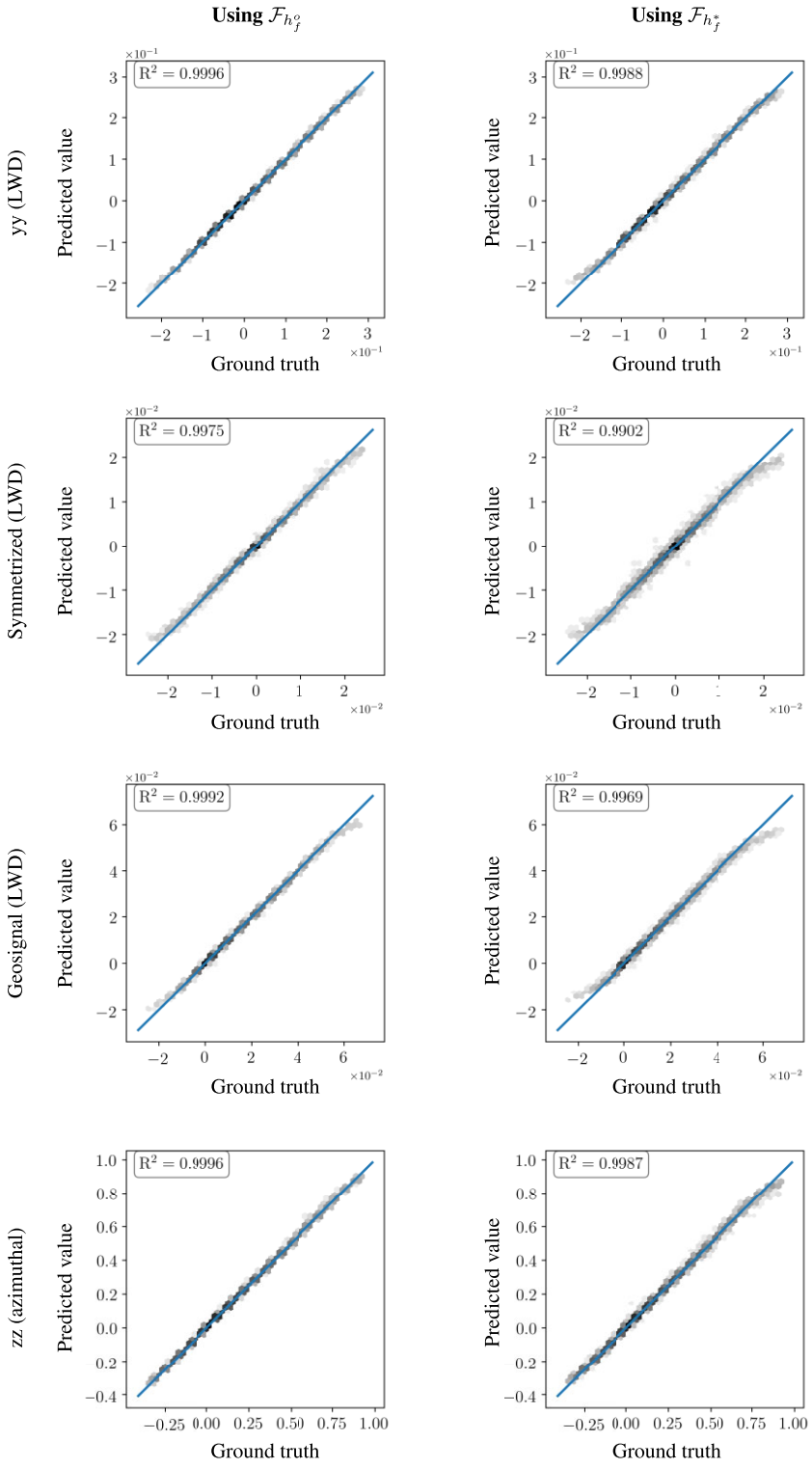


Figure 7. DNN approximation of $\mathcal{F}$. Comparison between the original network and the quasi-optimal one for the real part of selected measurements.

Table 4. Comparison of the training time between the original DNN and the quasi-optimal one.

Problem	Original DNN training time [hr]	Quasi-optimal DNN training time [hr]
Forward	18.69	4.65
Inverse	34.41	4.16

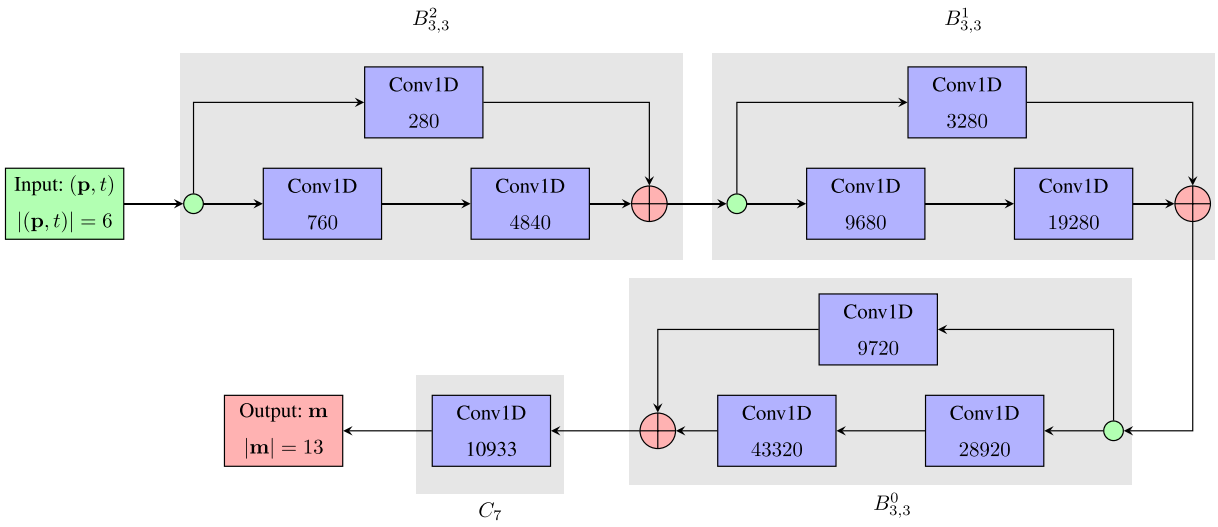


Figure 8. DNN architecture of $\mathcal{F}_{h^*_f}$. The number in each layer corresponds to the number of trainable parameters.

for $\{(\mathbf{p}_i, \mathbf{m}_i, t_i)\}_1^{n_v}$ being a validation data set distinct from the training data set with n_v being its size, and $N_p(h)$ is the number of unknowns of the DNN corresponding to the hyperparameter h. Then, the hyperparameter tuning consists of solving the following minimization problem:

$$h^*_f = \underset{h_f \in \mathcal{S}_\mathcal{F}}{\arg\min}\, R_f(h_f). \tag{10}$$

According to the two-step training strategy, after obtaining $\mathcal{F}_{h^*_f,\alpha^*}$, we need to minimize the following problem for the hyperparameter tuning of the inverse operator:

$$h^*_i = \underset{h_i \in \mathcal{S}_\mathcal{I}}{\arg\min}\, R_i(h_i), \tag{11}$$

where

$$R_i(h_i) = \frac{\mathcal{H}_i(h_i) - \mathcal{H}_i(h^o_i)}{\mathcal{H}_i(h^o_i)} - \frac{N_p(h^o_i) - N_p(h_i)}{N_p(h^o_i)}, \tag{12}$$

and

$$\mathcal{H}_i(h_i) = \sum_{i=1}^{n_v} L(\mathcal{F}_{h^*_f,\alpha^*} \circ \mathcal{I}_{h_i,\beta^*}(t_i, \mathbf{m}_i), \mathbf{m}_i). \tag{13}$$

The above optimization problems have no explicit gradient formulations. The simplest method to solve these problems is a grid search, which evaluates the scoring function over all the possible combinations of the hyperparameters. However, as the evaluation of the scoring function requires a complete training of a DNN and it can be costly, it is a common practice to rely on random search and a Bayesian approach to speed-up the optimization. These approaches are detailed in the next section.

4.3 AutoML algorithms

In this section, for simplicity in the notation, we denote $\mathcal{S} = \{h_0, h_1, \cdots, h_n\}$ as the search space of hyperparameters and $\mathcal{H}$ as the scoring function.

4.3.1 Random search

In this iterative approach, at the i-th iteration, we randomly select $h_i \in \mathcal{S}$ as our set of hyperparameters. By training the corresponding DNN to the selected set of hyperparameters, we compute $\mathcal{H}(h_i)$. Generally speaking, in the case of a massive search space, it is possible to interrupt the search algorithm as soon as we achieve our goal, for instance, a specific accuracy. In our case, we repeat this process until the search space is exhausted (Elsken *et al.* 2019; Li & Talwalkar 2020). We consider a search space exhausted when in five consecutive iterations, the randomly selected set of hyperparameters are amongst the ones we have already tried.

Using a random search approach to tune the hyperparameters could become excessively costly as we need to compute the score, that is to train a new DNN at each iteration. Moreover, we do not use the information we obtain during the previous iterations to select the next hyperparameter. As a result of such a blind selection process, this approach imposes a high computational cost, especially when considering a massive search space. Furthermore, as the selection is entirely random, and we may not try all the search space, there is no guarantee that we obtain the quasi-optimal set of hyperparameters. However, if stopping criteria while tuning is imposed, for example the tuning stops when we achieve a specific value of the scoring function, it is possible that a random search obtains the hyperparameters sooner than a grid search, hence, the possibility of reduced computational cost. In the worst-case scenario, random and grid searches impose the same computational cost.

4.3.2 Bayesian approach

Random search constitutes an improvement over grid search in terms of performance. However, it requires a large number of samples to properly characterize the search space. Given the high cost of calculating the scoring function for each sample, we consider a surrogate model—also known as *performance predictor* (White *et al.* 2021)—to: (1) estimate the scoring function without having to train the associated DNN and (2) to select the most promising

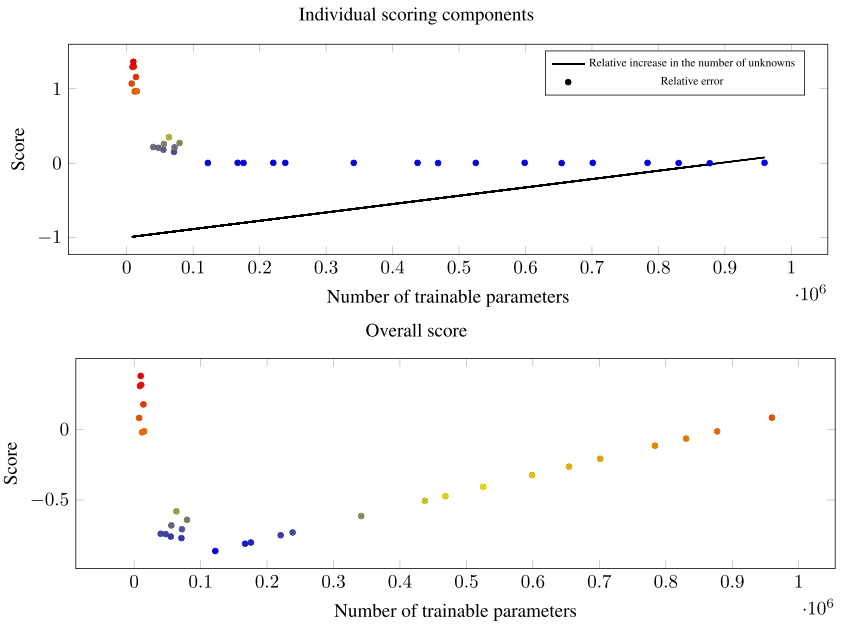


Figure 9. DNN optimization of the inverse function $\mathcal{I}$ using a random search. The colours indicate separate clusters of points.

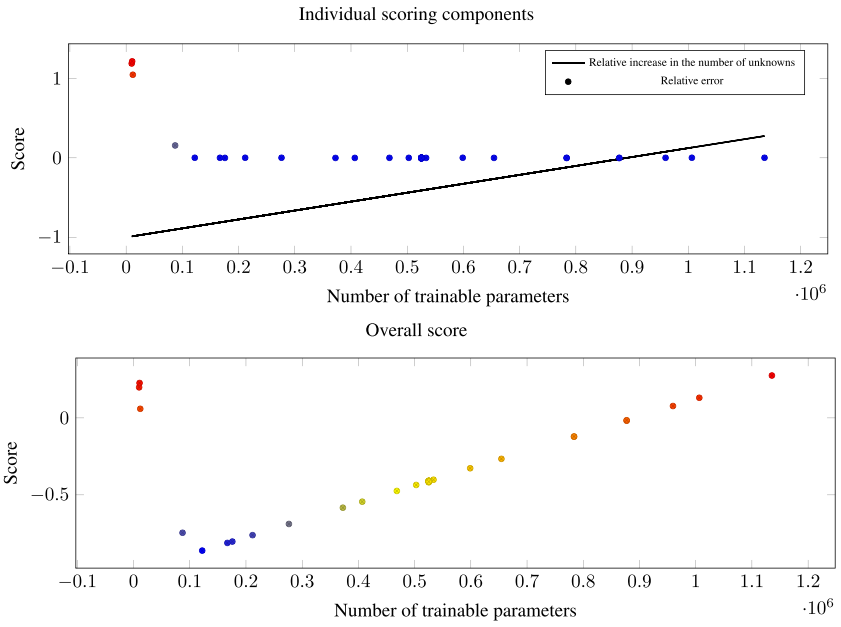


Figure 10. DNN optimization of the inverse function $\mathcal{I}$ using a Bayesian approach. The colours indicate separate clusters of points.

set of hyperparameters to test. For this, we use a probabilistic performance predictor based on Gaussian Processes (GPs; Snoek *et al.* 2012; Kandasamy *et al.* 2018; Elsken *et al.* 2019).

4.3.3 Gaussian processes as performance predictors

GPs are a generalization of multivariate Gaussian distributions to infinite dimensions, and therefore they can model a probability distribution over continuous functions. Thus, they allow us to estimate the value of a function and its uncertainty at any point in the domain. In our case, the function to model is the scoring function $\mathcal{H} : \mathcal{S} \to \mathbb{R}$.

A GP assumes that any finite set of n points has an associated n-variate Gaussian distribution, which is completely determined by its mean vector μ and covariance matrix Σ. Since a GP is a model on a potentially infinite set of points, it is characterized by a mean function $m(h) = \mathbb{E}(\mathcal{H}(h))$ and a covariance function (a.k.a kernel) $k(h, h') = \mathbb{E}[(\mathcal{H}(h) - m(h))(\mathcal{H}(h') - m(h'))]$, where $\mathbb{E}(X)$ is the expected value of X. These functions can be used to derive μ and Σ for any set of points $\{h_0, \ldots, h_n\}$. All the relevant properties of the GP including continuity, differentiability, and periodicity are determined by the covariance function.

Fig. 4 illustrates the concept for a hypothetical GP with a single input variable $h \in [0, 6]$. We use a zero mean function $m(h) = 0$ and a Matérn covariance function (Rasmussen 2004) with $\nu = 5/2$,

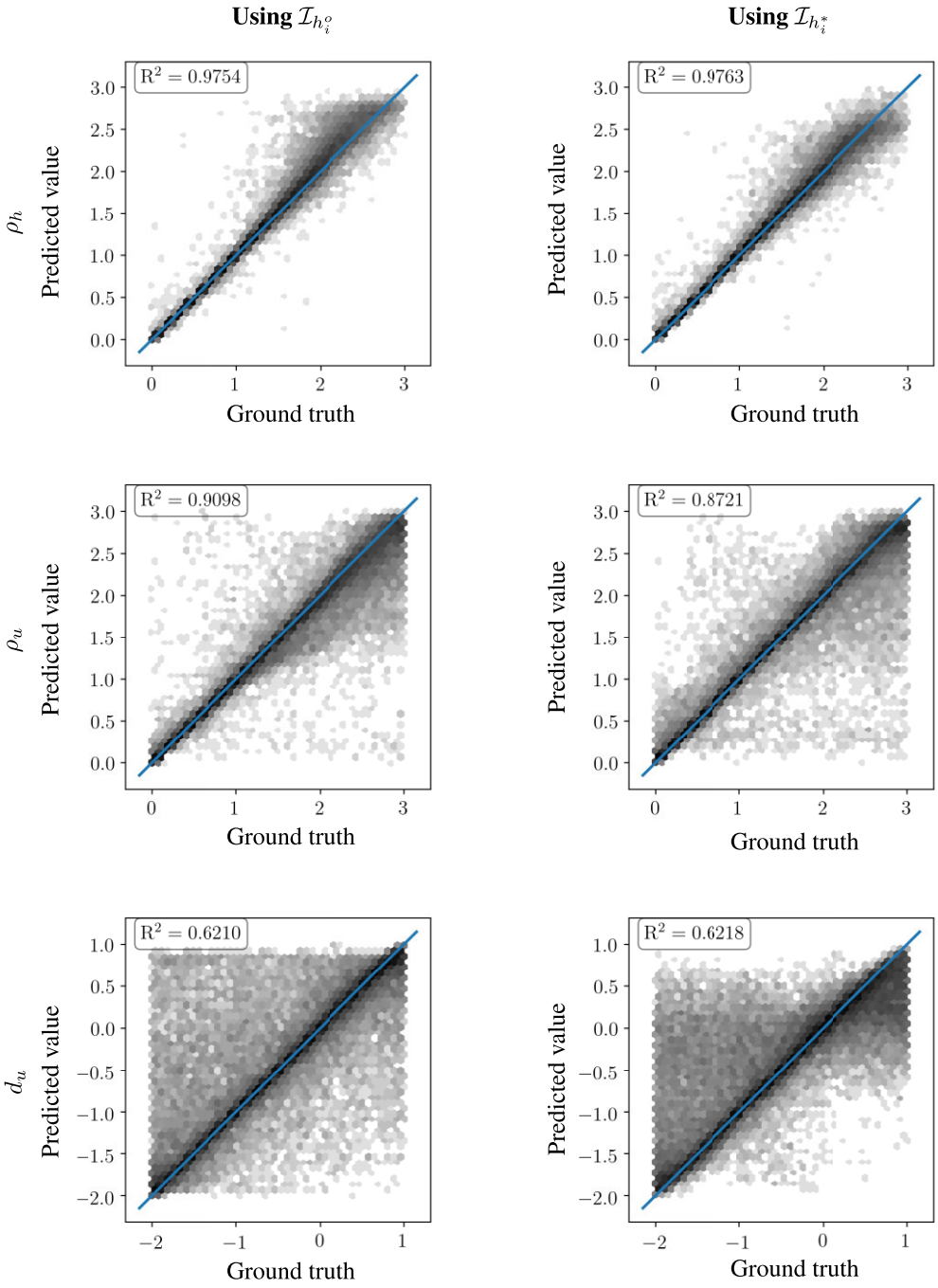


Figure 11. DNN approximation of $\mathcal{I}$. Comparison between the original network and the quasi-optimal one for a selected set of material properties.

which is widely used in hyperparameter optimization (O'Malley *et al.* 2019). This kernel is defined as follows:

$$k(h, h') = \left(1 + \sqrt{5}\|h - h'\|_{l_2} + \frac{5\|h - h'\|_{l_2}^2}{3}\right) * \exp(-\sqrt{5}\|h - h'\|_{l_2}), \quad (14)$$

Fig. 4(a) shows the prior distribution of the $\mathcal{H}(h)$ function and four random sampled functions following that prior. All samples are relatively smooth (this is determined by the selected kernel), but there is great variability among them. This gives us an idea of the flexibility of the GP in modelling $\mathcal{H}(h)$, but it also shows that, as expected, these priors will not provide useful predictions.

However, as soon as we start measuring some actual values of the scoring function, the model quickly converges to a well-constrained curve. For example, in Fig. 4(b) we incorporate the constraints (measurements) $\mathcal{H}(1) = -2$ and $\mathcal{H}(3) = -1$. Then, the posterior allows us to estimate the value of the scoring function more accurately, especially for the inputs that are closer to the observations.

The calculation of the posterior is based on the assumption that the observations and the desired estimations follow a joint Gaussian

Table 5. R^2-score comparison of the forward model between the original DNN (h_f^o) and quasi-optimal one (h_f^*) using the measurement system shown in Table 1.

$(TX-RX)$	Measurement	$\mathcal{F}_{h_f^o}$ (R^2 score) (Real/Imaginary)	$\mathcal{F}_{h_f^*}$ (R^2 score) (Real/Imaginary)
	zz	0.99/0.99	0.99/0.99
$(Tx_{1,1}, Tx_{1,2}, Rx_1, Rx_2)$	yy	0.99/0.99	0.99/0.99
	Geosignal	0.99/-	0.99/-
	Symmetrized directional	0.99/0.99	0.99/0.99
$(Tx_{2,1}, Tx_{2,2}, Rx_1, Rx_2)$	Symmetrized directional	0.99/0.99	0.99/0.99
(Tx, Rx_1)	zz	0.99/0.99	0.99/0.99
(Tx, Rx_2)	Symmetrized directional	0.99/0.99	0.99/0.99

distribution. Let us assume we have observed $\mathbf{z}_1 = \mathcal{H}(H_1)$ observations, with $|H_1| = n_1$, and we want to estimate the posterior $\mathbf{z}_2$ for a set H_2 of inputs, with $|H_2| = n_2$. Since $\mathbf{z}_1$ and $\mathbf{z}_2$ are jointly Gaussian, we can write:

$$\begin{bmatrix}\mathbf{z}_1\\ \mathbf{z}_2\end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix}\mu_1\\ \mu_2\end{bmatrix}, \begin{bmatrix}\Sigma_{11} & \Sigma_{12}\\ \Sigma_{21} & \Sigma_{22}\end{bmatrix}\right) \tag{15}$$

with:

$$\begin{aligned}
\mu_1 &= m(H_1) \quad (n_1 \times 1)\\
\mu_2 &= m(H_2) \quad (n_2 \times 1)\\
\Sigma_{11} &= k(H_1, H_1) \quad (n_1 \times n_1)\\
\Sigma_{22} &= k(H_2, H_2) \quad (n_2 \times n_2)\\
\Sigma_{12} &= k(H_1, H_2) = \Sigma_{21}^\top \quad (n_1 \times n_2)
\end{aligned}$$

Then, we can calculate the conditional distribution:

$$\begin{aligned}
p(\mathbf{z}_2 \mid \mathbf{z}_1) &= \mathcal{N}(\mu_{2|1}, \Sigma_{2|1})\\
\mu_{2|1} &= \mu_2 + \Sigma_{21}\Sigma_{11}^{-1}(\mathbf{z}_1 - \mu_1)\\
\Sigma_{2|1} &= \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12}
\end{aligned} \tag{16}$$

In this way, for each $h^* \in H_2$ we can calculate its posterior expected value μ_{h^*} and standard deviation σ_{h^*} according to eq. (16), as illustrated in Fig. 4(b) for $H_2 = [0, 6]$. It is important to highlight that according to eq. (16), $p(\mathbf{z}_1 \mid \mathbf{z}_1) \sim \mathcal{N}(\mu_1, 0)$, and therefore it is guaranteed that all the functions taken from the above distribution pass through the observation points.

4.3.4 Optimizing the hyperparameter search

In addition to a probabilistic estimation of the scoring function, the use of GPs as a surrogate model allows us to determine the next set of hyperparameters to test during the optimization. First, we evaluate the scoring function for a small number of random hyperparameter sets, and we construct the initial estimate of the surrogate model. Then, at each iteration, the Bayesian approach consists of the following steps: (1) to select the hyperparameter set to estimate the scoring function; (2) to evaluate the scoring function for the selected hyperparameter set; and (3) to update the surrogate model using the observation obtained in the previous step (Tipping 2004; Fox & Roberts 2012; Theodoridis 2015). For step (1), since an explicit formulation of the model is inaccessible, we need to rely on the so-called *acquisition functions*, which estimate an expected loss from evaluating $\mathcal{H}$ at a point h^*. Here, we consider the Upper Confidence Bound (UCB) (Srinivas *et al.* 2012) as our acquisition function, defined as follows:

$$UCB_{\mathcal{H}}(h^*) = \mu_{h^*} - \alpha\sigma_{h^*}, \tag{17}$$

where α is a calibration variable to balance exploration and exploitation. The concept of exploration is here related to the uncertainty derived from the standard deviation σ_{h^*}. Therefore, the higher the weight assigned to this factor (larger α), the more exploratory the behaviour of the *UCB*, selecting points further away from those already known. A lower α, on the other side, will give more weight to the points observed so far, which corresponds to exploitation. We empirically select α to be 2.6 (Snoek *et al.* 2012; O'Malley *et al.* 2019; Turner *et al.* 2021). Fig. 4(c) shows the minimum *UCB* value for the model fitted in Fig. 4(b), and Fig. 4(d) shows the updated model after incorporating the new scoring function result. Since GPs assume a continuous domain, we need to restrict the *UCB* and posterior evaluation to those sets of hyperparameters that are actually acceptable for our DNN architecture. With this approach, we aim to optimize the hyperparameters by learning from previous experiments, and hence we expect to require fewer iterations and lower computational time compared to random search.

5 NUMERICAL RESULTS

5.1 Hyperparameter tuning

To increase the speed of the hyperparameter tuning algorithms, we execute them using only 30 000 samples. Then, we train the quasi-optimal DNNs selected by the random search and the Bayesian algorithm using 300 000 samples to find the final DNN approximations of $\mathcal{F}$ and $\mathcal{I}$.

We consider the DNN architecture with hyperparameters $h_f^o = \{n = 5, k_0 = 3, k_1 = 3, l = 1\}$ that leads to 525 373 parameters as our reference approximation of $\mathcal{F}$. Analogously, $h_i^o = \{n = 6, k_0 = 3, k_1 = 3\}$ is the set of hyperparameters corresponding to the DNN architecture of the reference DNN approximating $\mathcal{I}$. The aforementioned DNN architecture consists of 890 925 parameters (see (Shahriari *et al.* 2022) for more details). To increase the computational efficiency, we also enforce two stopping criteria:

(i) An early stopping condition with the validation loss variation threshold and the patience being 10^{-3} and 30, respectively. This means that if the change in the loss is below the threshold during 30 consecutive epochs, the training stops.
(ii) If $\mathcal{H}(h) \leq 1.1 \times \mathcal{H}(h^o)$, where h is the hyperparameter set under trial, we also stop the training.

Table 3 shows the computational time of the hyperparameter tuning using both random search and the Bayesian approach. Results show that the Bayesian approach is less expensive than the random search. Note these time differences will increase as we augment the number of unknowns (i.e. measurements in the forward problem, and inverted parameters in the inverse problem).

Table 6. R^2-score comparison of the inverse model between the original DNN (h_i^o) and quasi-optimal one (h_i^*) using the measurement system shown in Table 1.

Material properties	$\mathcal{I}_{h_i^o}$ (R^2 score)	$\mathcal{I}_{h_i^*}$ (R^2 score)
ρ_u	0.90	0.87
ρ_c	0.97	0.97
ρ_l	0.90	0.85
d_u	0.62	0.62
d_l	0.61	0.63

Fig. 5 shows the results of hyperparameter tuning using random search to approximate $\mathcal{F}$. It represents the score value for each selection of hyperparameters from the search space $\mathcal{S}_{\mathcal{F}}$ and its corresponding number of trainable parameters. We also display the effect of individual factors involved in the scoring function, that is the relative increase in the number of unknowns and the relative error. Analogously, Fig. 6 shows the results of tuning using the Bayesian approach to approximate $\mathcal{F}$. In the case of random search, the algorithm repeatedly considers DNN architectures with less than 100 000 parameters even when their score is not significantly improving. However, in the Bayesian approach, the algorithm rapidly learns that this cluster of DNN architectures leads to unacceptable scores. Considering the results of both algorithms, we witness that the quasi-optimal set of hyperparameters is $h_f^* = \{n = 3, k_0 = 3, k_1 = 3, l = 7\}$, which corresponds to 131 013 parameters. Using this DNN, we achieve a comparable score to the reference one with approximately 25 per cent of the trainable parameters used by the reference DNN. Fig. 7 shows cross-plots comparing the accuracy of the DNN approximation of $\mathcal{F}$ using the quasi-optimal DNN and the reference one for a selected set of measurements. The accuracy of the two DNNs is similar, that is the R^2 scores of the prediction versus ground truth are comparable. Moreover, according to the training time shown in Table 4, training the reference DNN takes almost four times more computational time compared to the quasi-optimal one. Fig. 8 shows the DNN architecture of $\mathcal{F}_{h_f^*}$.

Figs 9 and 10 show the process of hyperparameter tuning to obtain a quasi-optimal DNN architecture to approximate $\mathcal{I}$. Analogous to the tuning for the forward function, the Bayesian approach selects a less redundant set of hyperparameters compared to the random search. By comparing the scores of all the DNN architectures, the quasi-optimal set of hyperparameters is $h_i^* = \{n = 3, k_0 = 3, k_1 = 3\}$ with 121 965 parameters. The quasi-optimal DNN architecture contains more than seven times fewer parameters than the reference one. Fig. 11 compares the accuracy of the quasi-optimal DNN architecture and the reference one for some inversion variables. Table 4 shows that we spend almost eight times more computational time to train the reference DNN compared to the quasi-optimal one. Tables 5 and 6 summarize the comparison between the quasi-optimal and original DNNs for the forward and the inverse operators, respectively. We obtain a similar accuracy between the original and quasi-optimal DNNs. Fig. 12 shows the DNN architecture of $\mathcal{I}_{h_i^*}$.

Table 7. Second set of evaluated measurements for each transmitter–receiver set.

Transmitter–receiver	Measured component
$(Tx_{1,1}, Tx_{1,2}, Rx_1, Rx_2)$	Geosignal, symmetrized directional
$(Tx_{2,1}, Tx_{2,2}, Rx_1, Rx_2)$	zz, yy, symmetrized directional
(Tx, Rx_1)	Symmetrized directional
(Tx, Rx_2)	zz

To investigate the effectiveness of our obtained DNNs when using a new set of measurements, we consider the measurements presented in Table 7. Table 8 shows the results of the trained DNN for $\mathcal{F}_{h_f^*}$ and $\mathcal{F}_{h_f^*} \circ \mathcal{I}_{h_i^*}$. Our obtained DNNs deliver high-quality results for this set of measurements. Moreover, Table 9 shows the results for $\mathcal{I}_{h_i^*}$. The DNN shows a slightly better accuracy when applied to the set of measurements of Table 1 because we chose them based on an optimization algorithm to maximize the R^2-score of the DNN output—see (Shahriari *et al.* 2022) for more details. In other words, the slight reduction in the performance is related to the selection of the measurements, not the DNN architecture, since $\mathcal{F}_{h_f^*} \circ \mathcal{I}_{h_i^*}$ is showing high accuracy.

5.2 Synthetic example

Fig. 13 compares the inversion results using $\mathcal{I}_{h_i^*}$ and $\mathcal{I}_{h_i^o}$ to the actual formation for a synthetic model. Both inversion models can adequately predict the material properties up to a sufficient depth of

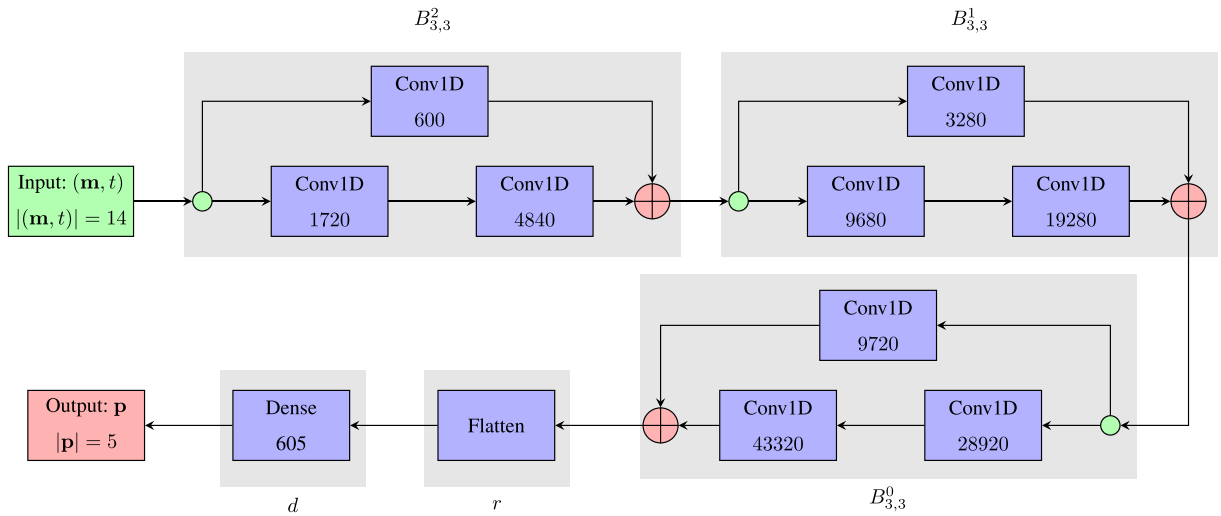


Figure 12. DNN architecture of $\mathcal{I}_{h_i^*}$. The number in each layer corresponds to the number of trainable parameters.

Table 8. R^2-score comparison between $\mathcal{F}_{h_f^*}$ and $\mathcal{F}_{h_f^*} \circ \mathcal{I}_{h_i^*}$ using the measurement system shown in Table 7.

$(TX-RX)$	Measurement	$\mathcal{F}_{h_f^*}$ (R^2 score) (Real/Imaginary)	$\mathcal{F}_{h_f^*} \circ \mathcal{I}_{h_i^*}$ (R^2 score) (Real/Imaginary)
$(Tx_{1,1}, Tx_{1,2}, Rx_1, Rx_2)$	Geosignal	0.98/-	0.96/-
	Symmetrized directional	0.99/0.99	0.98/0.98
	zz	0.99/0.99	0.99/0.99
$(Tx_{2,1}, Tx_{2,2}, Rx_1, Rx_2)$	yy	0.99/0.99	0.98/0.99
	Symmetrized directional	0.98/0.98	0.98/0.96
(Tx, Rx_1)	zz	0.98/0.98	0.96/0.98
(Tx, Rx_2)	Symmetrized directional	0.99/0.98	0.99/0.98

Table 9. R^2-score of $\mathcal{I}_{h_i^o}$ using the measurement system shown in Table 7.

Material properties	$\mathcal{I}_{h_i^o}$ (R^2 score)
ρ_u	0.84
ρ_c	0.94
ρ_l	0.83
d_u	0.54
d_l	0.55

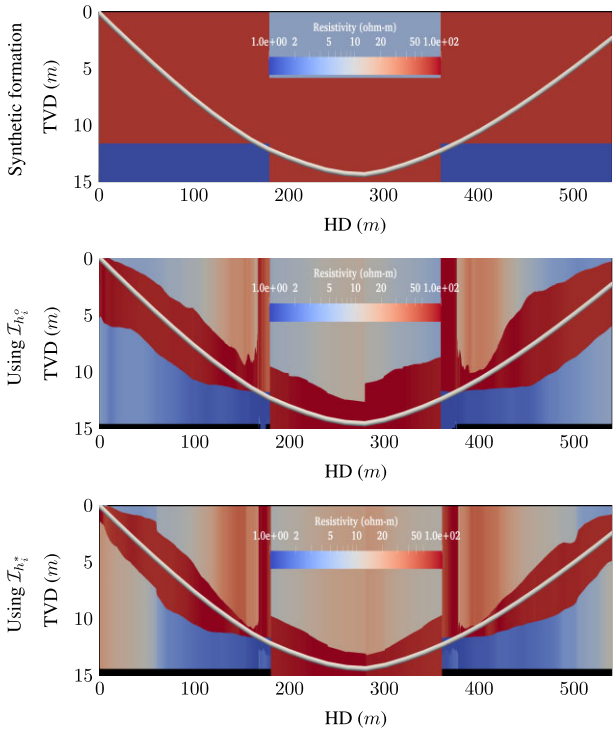


Figure 13. Model problem 1. Comparison amongst the synthetic (original) formation, and the formations predicted by the original (reference) DNN, and the quasi-optimal DNN.

investigation. The quasi-optimal DNN predicts the material properties around the trajectory. Moreover, it detects the bed boundary corresponding to oil-to-water contact from a few meters away from the trajectory. Fig. 14 shows similar results for a second synthetic formation.

6 CONCLUSIONS

In this work, we used AutoML—specifically, DNN architecture search algorithms– to obtain quasi-optimal DNN architectures for the inversion of borehole resistivity measurements. A quasi-optimal

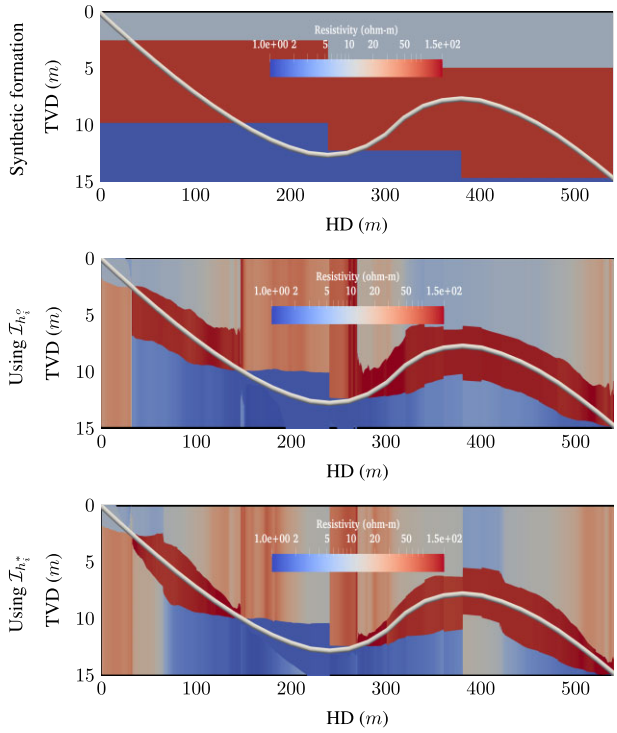


Figure 14. Model problem 2. Comparison amongst the synthetic (original) formation, and the formations predicted by the original (reference) DNN, and the quasi-optimal DNN.

DNN provides accurate results with a minimum number of unknowns (trainable parameters). We introduced a scoring function that accounts both for the accuracy of the trained DNN and its size compared to a reference large DNN. We introduced convolutional blocks as the main components of the DNN architecture.

We used two standard search algorithms to find our quasi-optimal hyperparameters: random search and a Bayesian approach based on Gaussian Processes. Both automatic search algorithms deliver quasi-optimal DNN architectures with reduced hand-design. Random search performs an arbitrary selection of the hyperparameters. In contrast, the Bayesian approach purposefully selects the hyperparameters using the information obtained from the previous iterations. Thus, it performs a less redundant selection of hyperparameters, and it typically requires fewer iterations to achieve the quasi-optimal DNN architecture, thereby, requiring less computational time than random search.

In this work, both algorithms converged to the same architecture because the search space is relatively small, and we imposed no stopping criteria while searching for the quasi-optimal DNN (tuning). Although the quasi-optimal DNN architecture contains

significantly fewer trainable parameters, it still delivers a performance comparable to the original DNN. Moreover, it substantially reduces the computational time required to train the DNN.

In future work, we shall consider a more general DNN architecture component, including for example, fully connected layers in the exploration and different number of filters in the CNN layers. Furthermore, we shall investigate the effect of using noisy data for training and evaluating the DNN. Moreover, we shall consider more complicated scenarios, for example a more general 2-D and 3-D subsurface parametrization, possibly combined with transfer learning. Although we considered a 1D-layered formation, our strategy can be extended to more complex geological targets. In those cases, we may require a larger data set and DNN to approximate the forward and inverse operators adequately. In addition, we shall study the possibility of an automated approach based on active learning to efficiently sample the space of subsurface properties using the minimum number of samples, that is the minimum data set's size.

ACKNOWLEDGMENTS

Mostafa Shahriari and Somayeh Kargaran have been supported by the Federal Ministry for Climate Action, Environment, Energy, Mobility, Innovation and Technology (BMK), the Federal Ministry for Digital and Economic Affairs (BMDW), the State of Upper Austria in the frame of the COMET - Competence Centers for Excellent Technologies Programme managed by Austrian Research Promotion Agency FFG and the 'Austrian COMET-Program' (Project InTribology, no. 872176).

David Pardo has received funding from: the Spanish Ministry of Science and Innovation projects with references TED2021-132783B-I00, PID2019-108111RB-I00 (FEDER/AEI) and PDC2021-121093-I00 (MCIN/AEI/10.13039/501100011033/Next Generation EU), 'BCAM Severo Ochoa' CEX2021-001142-S/MICIN/AEI/10.13039/501100011033; the Spanish Ministry of Economic and Digital Transformation with Misiones Project IA4TES (MIA.2021.M04.008/NextGenerationEU PRTR); and the Basque Government through the BERC 2022-2025 program, the Elkartek project SIGZE (KK-2021/00095) and the Consolidated Research Group MATHMODE (IT1456-22).

Tomas Teijeiro is supported by the grant RYC2021-032853-I funded by MCIN/AEI/10.13039/501100011033 and by the European Union NextGenerationEU/PRTR.

DATA AVAILABILITY

Data and codes used in this work can be shared with any interested party on request.

REFERENCES

Alyaev, S. & Elsheikh, A.H., 2022. Direct multi-modal inversion of geophysical logs using deep learning, *Earth Space Sci.,* **9**(9), e2021EA002186, doi:10.1029/2021EA002186.

Alyaev, S., Shahriari, M., Pardo, D., Omella, A.J., Larsen, D.S., Jahani, N. & Suter, E., 2021. Modeling extra-deep EM logs using a deep neural network, *Geophysics,* **86**(3), E269–E281.

Ba, J. & Caruana, R., 2014. Do deep nets really need to be deep?, in *Advances in Neural Information Processing Systems,* Vol. **27,** Curran Associates, Inc.

Beer, R. *et al.*, 2010. Geosteering and/or reservoir characterization the prowess of new-generation LWD tools, in *Proceedings of the SPWLA Annual Logging Symposium,* Perth, Australia, 19–23 June 2010, Paper Number: SPWLA-2010-93320.

Bittar, M. & Aki, A., 2015. Advancement and economic benefit of geosteering and well-placement technology, *Leading Edge,* **34**(5), 524–528.

Cheng, Y., Wang, D., Zhou, P. & Zhang, T., 2018. Model compression and acceleration for deep neural networks: the principles, progress, and challenges, *IEEE Signal Proc. Mag.,* **35,** 126–136.

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

Davydycheva, S., Homan, D. & Minerbo, G., 2004. Triaxial induction tool with electrode sleeve: FD modeling in 3D geometries, *J. appl. Geophys.,* **67,** 98–108.

Desbrandes, R. & Clayton, R., 1994. Chapter 9 measurement while drilling, *Dev. Petrol. Sci.,* **38,** 251–279.

Elsken, T., Metzen, J.H. & Hutter, F., 2019. Neural architecture search: a survey, *J. Mach. Learn. Res.,* **20,** 1–21.

Fox, C. & Roberts, S., 2012. A tutorial on variational Bayesian inference, *Artif. Intell. Rev.,* **38**(2), 85–95.

Goodfellow, I., Bengio, Y., Courville, A. & Bengio, Y., 2016. *Deep Learning,* Vol. **1,** MIT Press Cambridge.

He, K., Zhang, X., Ren, S. & Sun, J., 2016. Deep residual learning for image recognition, in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 27–30 June 2016, Las Vegas, NV, USA, pp. 770–778, IEEE.

He, X., Zhao, K. & Chu, X., 2021. AutoML: a survey of the state-of-the-art, *Knowledge-Based Syst.,* **212,** doi:10.1016/j.knosys.2020.106622.

Higham, C.F. & Higham, D.J., 2019. Deep learning: an introduction for applied mathematicians, *SIAM Rev.,* **61**(4), doi:10.1137/18M1165748.

Hu, Y., Guo, R., Jin, Y., Wu, X., Li, M., Abubakar, A. & Chen, J., 2020. A supervised descent learning technique for solving directional electromagnetic logging-while-drilling inverse problems, *IEEE Trans. Geosci. Remote Sens.,* **58**(11), 8013–8025.

Hutter, F., Kotthoff, L. & Vanschoren, J., 2019. *Automated Machine Learning: Methods, Systems, Challenges,* Springer Nature.

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

Jahani, N., Garrido, J.A., Alyaev, S., Fossum, K., Suter, E. & Torres-Verdín, C., 2022. Ensemble-based well-log interpretation and uncertainty quantification for well geosteering, *Geophysics,* **87**(3), IM57–IM66.

Jin, H., Song, Q. & Hu, X., 2019a. Auto-Keras: an efficient neural architecture search system, in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining,* pp. 1946–1956, ACM.

Jin, Y., Wu, X., Chen, J. & Huang, Y., 2019b. Using a physics-driven deep neural network to solve inverse problems for LWD azimuthal resistivity measurements, in *Proceedings of the SPWLA Annual Logging Symposium,* Day 5 Wed, 19 June 2019.

Kandasamy, K., Neiswanger, W., Schneider, J., Póczos, B. & Xing, E.P., 2018. Neural architecture search with bayesian optimisation and optimal transport, in *Proceedings of the 32nd International Conference on Neural Information Processing Systems, NIPS'18,* pp. 2020–2029, Curran Associates Inc.

Li, L. & Talwalkar, A., 2020. Random search and reproducibility for neural architecture search, in *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference, Vol. 115 of Proceedings of Machine Learning Research,* pp. 367–377, PMLR.

Loseth, L.O. & Ursin, B., 2007. Electromagnetic fields in planarly layered anisotropic media, *Geophys. J. Int.,* **170,** 44–80.

Malinverno, A. & Torres-Verdín, C., 2000. Bayesian inversion of DC electrical measurements with uncertainties for reservoir monitoring, *Inverse Problems,* **16**(5), 1343–1356.

Moghadas, D., 2020. One-dimensional deep learning inversion of electromagnetic induction data using convolutional neural network, *Geophys. J. Int.,* **222**(1), 247–259.

O'Malley, T., *et al.*, 2019. *Keras Tuner*. https://keras.io/keras_tuner/

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

Puzyrev, V., 2019. Deep learning electromagnetic inversion with convolutional neural networks, *Geophys. J. Int.,* **218**(2), 817–832.

Rammay, M.H., Alyaev, S. & Elsheikh, A.H., 2022. Probabilistic model-error assessment of deep learning proxies: an application to real-time inversion of borehole electromagnetic measurements, *Geophys. J. Int.,* **230**(3), 1800–1817.

Rasmussen, C.E., 2004. *Gaussian Processes in Machine Learning,* pp. 63–71, Springer Berlin Heidelberg.

Shahriari, M. & Pardo, D., 2020. Borehole resistivity simulations of oil-water transition zones with a 1.5D numerical solver, *Comput. Geosci.,* **24,** 1285–1299.

Shahriari, M., Rojas, S., Pardo, D., Rodríguez-Rozas, A., Bakr, S.A., Calo, V.M. & Muga, I., 2018. A numerical 1.5D method for the rapid simulation of geophysical resistivity measurements, *Geosciences,* **8**(6), 1–28.

Shahriari, M., Pardo, D., Moser, B. & Sobieczky, F., 2020a. A deep neural network as surrogate model for forward simulation of borehole resistivity measurements, *Proc. Manufact.,* **42,** 235–238.

Shahriari, M., Pardo, D., Picon, A., Galdran, A., Del Ser, J. & Torres-Verdín, C., 2020b. A deep learning approach to the inversion of borehole resistivity measurements, *Comput. Geosci.,* **24,** 971–994.

Shahriari, M., Pardo, D., Rivera, J.A., Torres-Verdín, C., Picon, A., Del Ser, J., Ossandón, S. & Calo, V.M., 2020c. Error control and loss functions for the deep learning inversion of borehole resistivity measurements, *Int. J. Numer. Methods Eng.,* **122**(6), 1629–1657.

Shahriari, M., Hazra, A. & Pardo, D., 2022. A deep learning approach to design a borehole instrument for geosteering, *Geophysics,* **87**(2), D83–D90.

Snoek, J., Larochelle, H. & Adams, R.P., 2012. Practical bayesian optimization of machine learning algorithms, in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2, NIPS'12,* pp. 2951–2959, Curran Associates Inc.

Spies, B.R., 1996. Electrical and electromagnetic borehole measurements: a review, *Surv. Geophys.,* **17**(4), 517–556.

Srinivas, N., Krause, A., Kakade, S.M. & Seeger, M.W., 2012. Information-theoretic regret bounds for gaussian process optimization in the bandit setting, *IEEE Trans. Inform. Theory,* **58,** 3250–3265.

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

Theodoridis, S., 2015. *Machine Learning: A Bayesian and Optimization Perspective,* 1st edn, Academic Press, Inc..

Tipping, M.E., 2004. *Bayesian Inference: An Introduction to Principles and Practice in Machine Learning,* pp. 41–62, Springer Berlin Heidelberg,

Turner, R., Eriksson, D., McCourt, M., Kiili, J., Laaksonen, E., Xu, Z. & Guyon, I., 2021. Bayesian optimization is superior to random search for machine learning hyperparameter tuning: analysis of the black-box optimization challenge 2020, *CoRR,* arXiv:2104.10201.

Watzenig, D., 2007. Bayesian inference for inverse problems- statistical inversion, *Elektrotech. Informationstech.,* **124,** 240–247.

White, C., Zela, A., Ru, R., Liu, Y. & Hutter, F., 2021. How powerful are performance predictors in neural architecture search?, in *Advances in Neural Information Processing Systems,* Vol. **34,** pp. 28 454–28 469, Curran Associates, Inc.