Deep learning in standard least-squares theory of linear models
Perspective, development and vision

Amiri-Simkooei, Alireza; Tiberius, Christian; Lindenbergh, Roderik

DOI
10.1016/j.engappai.2024.109376

Publication date
2024

Document Version
Final published version

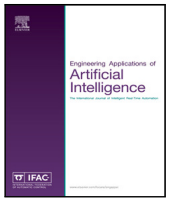
Published in
Engineering Applications of Artificial Intelligence

Engineering Applications of Artificial Intelligence 138 (2024) 109376

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai



Research Paper

Deep learning in standard least-squares theory of linear models: Perspective, development and vision

Alireza Amiri-Simkooei [a,*], Christian Tiberius [b], Roderik Lindenbergh [b]

[a] Department of Control and Operations, Faculty of Aerospace Engineering, Delft University of Technology, The Netherlands
[b] Department of Geoscience and Remote Sensing, Delft University of Technology, The Netherlands

ARTICLE INFO

Keywords:
Least-squares-based deep learning
Explainable artificial intelligence
Geostatistical learning
Design matrix
Quality control measures

ABSTRACT

Inspired by the attractive features of least-squares theory in many practical applications, this contribution introduces least-squares-based deep learning (LSBDL). Least-squares theory connects explanatory variables to predicted variables, called observations, through a linear(ized) model in which the unknown parameters of this relation are estimated using the principle of least-squares. Conversely, deep learning (DL) methods establish nonlinear relationships for applications where predicted variables are unknown (nonlinear) functions of explanatory variables. This contribution presents the DL formulation based on least-squares theory in linear models. As a data-driven method, a network is trained to construct an appropriate design matrix of which its entries are estimated using two descent optimization methods: steepest descent and Gauss–Newton. In conjunction with interpretable and explainable artificial intelligence, LSBDL leverages the well-established least-squares theory for DL applications through the following three-fold objectives: (i) Quality control measures such as covariance matrix of predicted outcome can directly be determined. (ii) Available least-squares reliability theory and hypothesis testing can be established to identify mis-specification and outlying observations. (iii) Observations' covariance matrix can be exploited to train a network with inconsistent, heterogeneous and statistically correlated data. Three examples are presented to demonstrate the theory. The first example uses LSBDL to train coordinate basis functions for a surface fitting problem. The second example applies LSBDL to time series forecasting. The third example showcases a real-world application of LSBDL to downscale groundwater storage anomaly data. LSBDL offers opportunities in many fields of geoscience, aviation, time series analysis, data assimilation and data fusion of multiple sensors.

1. Introduction

Big data is undoubtedly the twenty-first century phenomenon, which deals with massive amounts of data in many areas of science, engineering and industry. Though this provides great opportunities, it also confronts us with unprecedented challenges regarding their processing and interpretation (Williams, 2017). As a part of artificial intelligence, machine learning (ML) algorithms have been widely used in a variety of applications in computer vision, geoscience, medicine, voice and face recognition, and email filtering. ML is one of today's most rapidly growing technical fields that lies at the core of artificial intelligence (AI) and data science (Jordan and Mitchell, 2015). ML builds a model based on sample data itself, where it is rather difficult to develop a (linear) mathematical model to connect the explanatory variables to predicted ones (Hu et al., 2020). There are therefore demanding needs to develop novel machine learning and artificial intelligence methods that help in processing and interpretation of such data. This contribution is an attempt to formulate the ML method using the standard least squares theory.

Deep learning (DL) describes a specific family of machine learning algorithms that are used to train complex prediction models (Hinton et al., 2006). DL, successfully used to several application areas, is a relatively new and novel methodology receiving much attention. For example, in the classification of handwritten digits of the Modified National Institute of Standards and Technology (MNIST) data set, a DL method has set a record to have an error rate of only 0.21% (Wan et al., 2013). There are different algorithms for DL including deep feedforward neural networks (D-FFNN), convolutional neural networks (CNNs), deep belief networks (DBNs), autoencoders (AEs), and long short-term memory (LSTM) networks. For an introductory review on these DL techniques the reader may refer to Emmert-Streib et al. (2020).

Extracting meaningful patterns from big data for decision-making, prediction and approximation poses a few challenges in DL. They range

* Corresponding author.
E-mail address: a.amirisimkooei@tudelft.nl (A. Amiri-Simkooei).

https://doi.org/10.1016/j.engappai.2024.109376
Received 5 April 2024; Received in revised form 15 August 2024; Accepted 21 September 2024
Available online 30 September 2024

from format variation of raw data, noisy and poor quality data, high dimensionality, and scalability of algorithms (Najafabadi et al., 2015) to implementation problems like overfitting, statistical inference of results and the so-called saturation phenomenon. For example, it is rather difficult to detect and identify outliers from training data in machine learning techniques (Belhadi et al., 2020). These are mainly related to the 'black-box' problem associated to this technique. The opposite of black-boxness is transparency and there is ongoing research to increase transparency in the current AI domain. For a review on eXplainable AI (XAI), its concepts, taxonomies, opportunities and challenges we refer to Arrieta et al. (2020). To address parts of the above problems we put an attempt to make a connection between input and output by establishing a linear structure in the deep learning black boxes. The linear least squares theory can lay this relation.

Although linear model theory has a variety of successful applications, its applicability is limited. This is because the relation between the explanatory and predicted variables is not always linear. As previously explained, the use of machine learning in general and deep learning in particular has a variety of applications in handling these kinds of problems. These learning methods are an application of artificial intelligence (AI) in which a network is trained on a series of training examples, and then applied to make informed decisions. The performance of the trained network is usually tested on a testing dataset. The technique is both formulated in a supervised and unsupervised framework. In its supervised variant, the training datasets are designed to train algorithms into classifying data, or predicting outcomes accurately. Unsupervised learning uses the algorithms to analyze and cluster unlabeled data sets by discovering hidden patterns in data without the need of human intervention. This contribution considers the supervised framework.

Supervised learning can be categorized as two types of problems: classification and regression (Criminisi et al., 2012). The former uses an algorithm to accurately assign test data into specific categories, such as separating hand-written digits from 0 to 9. The latter is a supervised method that uses an algorithm to understand the relationship between dependent and independent variables. They are usually helpful for predicting numerical values based on different data points, such as stock/sale/weather forecasting formulated in a linear regression or stochastic-based regression models. To further highlight the difference between the two, we note that classification is about predicting a label and regression is about predicting a quantity. Therefore the output variable in regression is numerical (or continuous), whereas for classification, it is categorical (or discrete) (Liaw et al., 2002). Regression is the subject of discussion in this contribution.

Least squares (LS) and deep learning (DL) have been widely used in many engineering applications. A complete review of these methods is beyond the scope of the present contribution, instead we briefly review some relevant literature on the intersection. The available literate on the theory of LS and DL mainly relies on the individual use (and comparison) of these methods (Zeng et al., 2018; Chang and Song, 2023), DL with an LS cost function (Cai et al., 2020; Jin et al., 2024), DL with an assisted LS method (Wang et al., 2021b; Singh et al., 2024), or a combination of LS and DL (Zhang et al., 2019b; Jin et al., 2024). There is also research that explores the integration of LS optimization techniques with deep learning models to enhance interpretability, reliability, and performance (Mao et al., 2017). They introduced an LS generative adversarial network that improved training stability, and hence could address issues such as vanishing gradients. The LS support vector machine has been used to enhance classification tasks in high-dimensional data. This method assigns specific penalties to samples based on their susceptibility to misclassification and their position relative to the decision boundary (Zhao et al., 2022). Although LS has been widely used in many engineering problems, its combination with DL is rather limited. There is no rigorous formulation of LS and DL methods that enables to take advantage of the combination of both methods: DL methods to capture complex non-linearity and LS theory for appropriate statistical inference of results. This contribution represents a step forward in integrating these two approaches within a theoretical framework. This alignment enhances the interpretability and transparency of the model, and hence contributes to the goals of XAI.

A question arises as to whether one can formulate the DL algorithm in the standard least squares theory. This turns out to be the case and is the focus of the present contribution. We will elaborate to see how such relation can be made. We make use of a feature (input) matrix, consisting of explanatory variables, and apply the gradient descent methods to establish a design matrix. The design matrix is estimated through a convolution of the feature matrix with a so-called weight matrix, which is unknown. To allow for possible non-linearity, the convolved matrix is fed to an appropriate 'activation function' (see later Eq. (15)). Having DL formulated in the framework of the linear least squares theory opens a wide range of opportunities and challenges. The method allows to apply the existing theory of the least squares method to the DL problem. For example, the precision of the predicted values/decisions (prediction error) can directly be evaluated through this formulation. Also hypothesis testing referring to detection, identification and validation (DIA) can be extended to the DL problem (Teunissen, 2000a, 2018).

As mentioned, DL methods face challenges like the black-box problem and the need for vast amounts of training data. These issues underscore the necessity for developing DL models that are able to effectively capture fundamental physical phenomena, such as structure and symmetry. There is increasing interest in integrating physics principles into DL, leading to the development of physics-informed neural networks (PINNs) models (Raissi et al., 2019; Karniadakis et al., 2021). PINNs embed mathematical models of physical laws, often represented as partial differential equations (PDEs), into the loss function during training (Lehmann et al., 2023). They can therefore integrate physics laws into the learning process, which improves performance with limited data (Karniadakis et al., 2021) and addresses the explainability and interpretability of DL methods (Cuomo et al., 2022). They provide prior information through a linear system of equations with uncertainty represented by a covariance matrix (Menke, 2015). By incorporating prior knowledge, PINNs reduce the need for extensive data. The least squares method can embed this prior knowledge using soft and hard physics-based constraints into DL through a unified least-squares formulation (Amiri-Simkooei, 2019).

This paper is organized as follows. Section 2 reviews the linear least squares theory. In particular, the univariate and multivariate model will be discussed. Section 3 presents the rationale behind the development of least-squares-based deep learning (LSBDL) and its transparency. Section 4 generalizes the linear(ized) least squares theory to LSBDL. The design matrix is trained to minimize the least squares objective function using the steepest descent (SD) method. Special cases, linked to what is already known in literature, will also be discussed. Section 5 gives the Gauss–Newton (GN) formulation of LSBDL, known for its faster convergence rate compared with SD. Section 6 embeds LSBDL within the least squares framework, which is twofold. (i) determining quality control measures like covariance matrix of prediction, (ii) using least-squares theory to detect model errors and outliers. Section 7 elaborates to further developed different aspects of LSBDL in deeper networks, for example exploration of multi-layer LSBDL. Section 8 provides an overview of highlights, opportunities, and challenges associated with LSBDL, along with its XAI perspective. For example, the use of covariance matrix is crucial to express (co)variances among observations, particularly when fusing different data sources. This matrix plays an important role in training networks, even when the data are inconsistent, heterogeneous, or statistically correlated. Section 9 illustrates the performance of the proposed method using three examples. In Section 10 we draw some conclusions.

2. Standard linear least squares theory

This section reviews the least squares theory in linear model of observation equations. Consider the following linear model

$$y = Ax + e,\ \mathsf{D}(y) = Q_y \tag{1}$$

where $y \in \mathbb{R}^m$ is a vector of m observations, $x \in \mathbb{R}^n$ is a vector of n unknown parameters, $e \in \mathbb{R}^m$ is a vector of m residuals (errors), $A \in \mathbb{R}^{m\times n}$ is a given design matrix, and $Q_y \in \mathbb{R}^{m\times m}$ is the positive semi-definite covariance matrix of observables y. D is the dispersion operator. The least squares estimate of the unknown parameters is

$$\hat{x} = (A^T Q_y^{-1} A)^{-1} A^T Q_y^{-1} y \tag{2}$$

The least squares estimates of the observations and residuals are respectively as follows:

$$\hat{y} = P_A y \ \text{ and } \ \hat{e} = P_A^{\perp} y \tag{3}$$

where $P_A = A(A^T Q_y^{-1} A)^{-1} A^T Q_y^{-1}$ and $P_A^{\perp} = I - P_A$ are two orthogonal projectors (Teunissen, 2000b). The covariance matrices of the above estimators are

$$Q_{\hat{x}} = (A^T Q_y^{-1} A)^{-1},\ \ Q_{\hat{y}} = P_A Q_y \ \text{ and } \ Q_{\hat{e}} = P_A^{\perp} Q_y \tag{4}$$

The above model of observation equations is referred to as the univariate model. It is because the model relates one observation vector y to one unknown vector x. To keep the generality of our formulation, we use a so-called multivariate linear model. Such a model can generally relate r observation vectors, all collected in an $m\times r$ observation matrix $Y = [y_1, \dots, y_r]$ (each y_i, $i = 1, \dots, r$ is an m-vector), to r unknown vectors, all collected in an $n \times r$ unknown matrix $X = [x_1, \dots, x_r]$. A multivariate linear model is of the form (Koch, 1999; Amiri-Simkooei, 2009)

$$Y = AX + E \tag{5}$$

in which the same $m \times n$ design matrix A connects y_i to x_i for all $i = 1, \dots, r$, and $E = [e_1, \dots, e_r]$ is an $m \times r$ residual matrix. The covariance matrix of Y is referred to as the (co)variance among entries y_{ij} and y_{kl}, for $i, k = 1, \dots, m$ and $j, l = 1, \dots, r$. It is more convenient to express this covariance matrix as a Kronecker product $\otimes$ of two matrices as

$$\mathsf{D}(Y) = Q_{\mathrm{vec}(Y)} = \Sigma \otimes Q \tag{6}$$

where vec(.) is the vector operator converting a matrix into a column vector, the $r\times r$ matrix Σ expresses covariances among different classes (output), and the $m\times m$ matrix Q expresses covariances among different observations within a class. For example, in global navigation satellite system (GNSS) position time series, the (co)variances among three coordinate components (north, east and up) can be expressed by a 3×3 matrix Σ, whereas the time correlation of individual time series is expressed by an $m \times m$ matrix Q (Amiri-Simkooei, 2013). The least squares estimate of the unknown parameters is (Amiri-Simkooei, 2007)

$$\hat{X} = (A^T Q^{-1} A)^{-1} A^T Q^{-1} Y \tag{7}$$

The least squares estimates of the observation and residual matrices are

$$\hat{Y} = P_A Y, \ \text{ and } \ \hat{E} = P_A^{\perp} Y \tag{8}$$

where $P_A = A(A^T Q^{-1} A)^{-1} A^T Q^{-1}$ and $P_A^{\perp} = I - P_A$ are two orthogonal projectors. The covariance matrices of the above estimates are

$$\begin{aligned} Q_{\mathrm{vec}(\hat{X})} &= \Sigma \otimes (A^T Q^{-1} A)^{-1}, \\ Q_{\mathrm{vec}(\hat{Y})} &= \Sigma \otimes P_A Q, \\ Q_{\mathrm{vec}(\hat{E})} &= \Sigma \otimes P_A^{\perp} Q, \end{aligned} \tag{9}$$

having dimensions $nr \times nr$, $mr \times mr$, and $mr \times mr$, respectively. The univariate model is a special case of the multivariate model with $r = 1$, $Y = y$, $X = x$, $E = e$, $A = A$, $\Sigma = \sigma^2$, $Q = Q$, giving $Q_y = \sigma^2 Q$ (σ^2 is called the variance of unit weight). Therefore we stay with the general formulation of the multivariate model.

The least squares formulation allows to apply hypothesis testing to the observations and the linear model. In particular one can test for the presence of inconsistencies in the functional model such as outliers in observations. If Σ and Q are known, a commonly used measure is the overall model test (OMT). It is based on the following measure of discrepancy on the quadratic form of the residuals (Teunissen, 2000a; Amiri-Simkooei, 2007)

$$T_{r(m-n)} = \mathrm{tr}(\hat{E}^T Q^{-1} \hat{E} \Sigma^{-1}) \tag{10}$$

where tr(.) is the trace of a matrix. The above test statistic has a chi-squared distribution χ^2 with $df = r(m - n)$ degrees of freedom, i.e. $T_{r(m-n)} \sim \chi^2(r(m - n), 0)$, provided the observations are normally distributed. In the univariate case, we have ($r = 1$)

$$T_{m-n} = \hat{e}^T Q_y^{-1} \hat{e} \sim \chi^2(m - n, 0) \tag{11}$$

where $Q_y = \sigma^2 Q$ is assumed known.

The above test can be performed at a given confidence level. If the test is rejected it indicates that there are unresolved issues on the consistency between the model and observations. This is referred to as the 'detection' step in the DIA (detection, identification and adaptation) steps (Teunissen, 2000a). This can possibly be due to the presence of outliers in the observations, which can be identified in the 'identification' step using the w-test statistics. The w-test is widely used as a datasnooping procedure to screen individual observations for the presence of an outlier (Baarda, 1968). To test the i^{th} observation, the w-test statistics is of the form (Teunissen, 2000a)

$$w_i = \frac{c_i^T Q_y^{-1} \hat{e}}{\sqrt{c_i^T Q_y^{-1} Q_{\hat{e}} Q_y^{-1} c_i}} \tag{12}$$

which has a standard normal distribution under the null hypothesis. The vector $c_i = [0, \dots, 1, \dots, 0]^T$ is a canonical unit vector, having all zeros with a one as its i^{th} entry; i ranges from 1 to m to check for all observations.

A recently extended DIA method takes into account the detection, identification and adaptation steps at the same time. Estimation and testing are thus combined in this procedure (Teunissen, 2018).

3. Problem description of LSBDL

The theory of the previous section will be applied to the least-squares-based deep learning (LSBDL) problem, thus making LSBDL transparent and explainable. LSBDL, as a nonlinear problem, should be formulated in the nonlinear least squares framework. Two gradient descent methods of LSBDL are proposed: (1) Steepest descent method (Section 4), and (2) Gauss–Newton method (Section 5), see Teunissen (1990).

3.1. Definition and notation

Before we start formulating LSBDL, we briefly define some standard terms used in AI: Training involves estimating unknown parameters X and weights (W). The training rate α controls the iterative update of weight parameters in the gradient methods. Activation functions are non-linear differentiable functions that introduce non-linearity. Regularization techniques enhance generalization capability by using a parameter κ to balance bias and variance. The momentum parameter μ enhances training speed and accuracy by incorporating the history of gradients. For more information, we refer to Bishop and Bishop (2024).

For the sake of brevity/convenience, the following notations are interchangeably used in the context of this paper:

$x = \mathrm{vec}(X)$: an nr-vector of unknown parameters reshaped from the $n \times r$ unknown matrix X; $\hat{x} = \mathrm{vec}(\hat{X})$ is the least squares estimate for

x (nr entries). It is noted that vec(.), the vector operator, converts the matrix X into a column vector x, which is convenient to express the multivariate variables as the univariate variables.

$y = \mathrm{vec}(Y)$: an mr-vector of observations reshaped from the $m \times r$ observation matrix Y; $\hat{y} = \mathrm{vec}(\hat{Y})$ is the least squares estimate for y (mr entries).

$e = \mathrm{vec}(E)$: an mr-vector of residuals reshaped from the $m \times r$ residual matrix E; $\hat{e} = \mathrm{vec}(\hat{E})$ is the least squares estimate for e (mr entries).

$w = \mathrm{vec}(W)$: a kn-vector of unknown weights reshaped from the $k \times n$ weight matrix W; $\hat{w} = \mathrm{vec}(\hat{W})$ is the least squares estimate for w (kn entries).

The above notations can accordingly be generalized to the covariance matrix of the above variables; for example $Q_y = Q_{\mathrm{vec}(Y)}$ is the covariance matrix of $y = \mathrm{vec}(Y)$.

An activation function and its derivative are denoted as $\mathtt{a}(.)$ and $\mathtt{a}'(.)$, respectively, with their matrix counterparts as $\mathtt{A}(.)$ and $\mathtt{A}'(.)$. For example, a sigmoid activation function and its first derivative are

$$A = \mathtt{A}(u) = 1/(1+e^{-u}),\ \ A' = \mathtt{A}'(u) = A - A \odot A \tag{13}$$

where $\odot$ is the element-wise Hadamard product. In principle, other activation functions can be used as well (Ramachandran et al., 2017).

3.2. Problem description

In the classical set-up of the multivariate linear model $Y = AX + E$, the design matrix A and the observation matrix Y are assumed known. This is however not the case for our deep learning model as the design matrix A is unknown. The design matrix is then to be trained in an iterative procedure in a supervised learning algorithm. We would still stick on capital A for this design matrix, which is also abbreviated for the 'activation' function. A combination of different activation functions are usually used in DL algorithms (see later Eq. (104)).

To train the design matrix A, we need input data. Each row of the input data can in principle create one row of A. In the classical linear model, this is referred to as the explanatory variables or the so-called input variables. For example, in a linear regression model $y(u) = x_1 u + x_2$ these explanatory variables are $[u,\ 1]$, which is a linear function of the input variable u. There are however examples that are not linear functions of the input variable. The sine fitting problem $y(t) = x_1 \sin(\omega t) + x_2 \cos(\omega t) + x_3$, with the basis functions $[\sin(\omega t),\ \cos(\omega t),\ 1]$, leads to a nonlinear function of input variable t. In a curve fitting problem, a p-order polynomial $y(u) = x_1 u^p + \cdots + x_p u + x_{p+1}$ can lead to the basis functions $[u^p,\ \cdots,\ u,\ 1]$, which are also nonlinear functions of input variable u. Although the above basis functions are nonlinear, they are entries of the design matrix A, and therefore the linear least squares theory can directly be used to estimate x_i's.

For deep learning applications, the above-mentioned basis functions, and therefore elements of A, are not directly available. We start with the explanatory variables (input features) $d = [d_1, \ldots, d_k]^T$. In the DL literature they are also called neurons. The observation (output) y is assumed to be an (unknown) nonlinear function $y = f(d, x)$ of the input variables. An unknown linear combination of the input variables, $d^T w + b$, can be used to approximate y, where the entries of w, the weights, and b, the bias term are unknown. This equation can provide a linear relation between input d and output y. To introduce the nonlinearity, an activation function $\mathtt{a}(.)$ can be used. The input of the activation function is $d^T w + b$, and therefore we can have $y = \mathtt{a}(d^T w + b)$. An important feature of an activation function is its ability to add nonlinearity and hence to allow capturing any complexity in the data. This is the basics of all data-driven ML techniques. A generalized form of the above equation can be expressed as

$$y = f(d, x) = x_1 \mathtt{a}(d^T w_1 + b_1) + \cdots + x_n \mathtt{a}(d^T w_n + b_n) \tag{14}$$

where x_i's along with w_i's and b_i's are to be estimated. Eq. (14) is in conjunction with the *universal approximation theorem* expressing that finite linear combinations of appropriate activation functions (sigmoid for example) can uniformly approximate any continuous real-valued function of k real variables with an arbitrary precision (Cybenko, 1989; Hornik et al., 1989; Kruse et al., 2011; Costarelli et al., 2013).

We denote the feature matrix as an $m \times k$ matrix D, where m is the number of observations and k is the number of features (k features per observation). We present a methodology to train a design matrix A and hence to establish a linear model of observation equations. As indicated in Eq. (14) the entries of A, the basis functions, are indeed nonlinear functions of the input variables d's. This is conceptually a simple idea, similar to the 'sine' or 'curve' fitting problems mentioned above. Although we take advantage of the linear least squares theory to estimate x_i's, the entries of A are (unknown) nonlinear functions of input variables.

For an observation y_i with input variables d_i (i^{th} row of D), the entries of A are expressed as $a_{ij} = \mathtt{a}(d_i^T w_j + b_j), i = 1, \ldots, m,\ j = 1, \ldots, n$, where d_i is a given k-vector, w_j is a k-vector of unknown weights and b_j is a scalar bias term. The weights and biases (both included in weight matrix $W \in \mathbb{R}^{k \times n}$) are the commonly used learnable parameters of all ML models. In matrix notation, this can be reformulated to the following system of equations:

$$Y = \mathtt{A}(DW)X + E,\ \ Q_{\mathrm{vec}(Y)} = \Sigma \otimes Q \tag{15}$$

where $\mathtt{A}(.) = \mathtt{a}_{ij}(.) \in \mathbb{R}^{m \times n}$, the design matrix, is an element-wise activation function introduced above. The design matrix $A = \mathtt{A}(DW)$ is not known because the weights and biases are unknown beforehand, and they need to be trained in the estimation process. The above system is nonlinear with respect to W. However, when these two matrices are given, the system becomes a linear system of equations with respect to X. We note that the weight matrix W also includes unknowns related to the biases b_j, and therefore the feature matrix D is augmented to contain the summation vector (a vector containing all ones).

In summary, our choice for Eq. (15) is motivated by the following three-fold arguments: (i) This equation is a multivariate version of Eq. (14), which can be expressed as $y = Ax + e$ (univariate model). The same basis functions and therefore the same design matrix A are used to estimate the coefficients x_i's, which can vary from one model to another. This can happen in multiple output regression models, applied for example to forecast the multi-step ahead time series variables. (ii) In DL regression literature, the activation function of the last layer is usually the *fully connected layer*, which is set to be an identity map as $\mathbf{I}(U) = U$ (Selmic and Lewis, 2002; Kawaguchi, 2016; Bakhshi and Chalup, 2021). Identity maps provide a structural regularity between the input and output and helps the network to produce a better representation of the output (He et al., 2016b; Kumar and Ningombam, 2018; Zhang et al., 2019a). The formulation of Eq. (15) automatically imposes the identity map in the last layer because we can have $\mathbf{I}(\mathtt{A}(DW)X) = \mathtt{A}(DW)X$. Therefore this equation is seen as the transposed version of a fully connected feed-forward neural network having one hidden layer and an identity activation function for its output layer (Kawaguchi, 2016). A similar formulation also holds true later in Eq. (104) for the multilayer model. (iii) The linear model structure introduced in Eq. (15) allows to apply the existing least squares theory to DL applications. For example, the global minimizer of the least squares problem for X is automatically ensured (see Eq. (7)). Hypothesis testing and other existing theories in the DIA procedure can directly be applied to this model, see Baarda (1968), Teunissen (2000a, 2018). These characteristics of LSBDL are closely tied to XAI.

A final remark on the formulation of LSBDL is in order. The relation between different matrix dimensions k, n and m plays an important role in this formulation. As an overdetermined system of equations, the number of observations should be larger than the number of unknowns in a least squares problem ($m > n$). The number of unknowns n, (columns of DW), depends on the complexity of the problem. If we assume D is of full-column rank, we have $\mathrm{rank}(D) = k$. It is known, from linear algebra, that $\mathcal{R}(D) = \mathcal{R}(DW)$, with $\mathcal{R}(.)$ the range space of a matrix, provided the $k \times (n = k)$ matrix W is a regular matrix (Strang,

1988). This indicates that $D\,W$ is also of full-column rank. However when W has more columns than rows, i.e. when $n > k$, matrix $D\,W$ cannot be of full-column rank. This does not however hold for $\mathrm{A}(D\,W)$, as it can still be of full-column rank. This is due to the non-linearity nature of the activation function $\mathrm{A}(.)$ applied, which violates the 'linear dependence' theory. This can make the LSBDL theory more attractive and flexible.

4. Steepest descent (SD) formulation of LSBDL

4.1. Derivation of SD gradient

The design matrix $A = \mathrm{A}(D\,W)$ is unknown, which is to be trained/estimated using the least squares method. Having matrices Y and D available, the supervised learning leads to the following least squares criterion:

$$(\hat{W}, \hat{X}) = \arg\min_{(W,X)} \frac{1}{2}\|Y - \mathrm{A}(D\,W)X\|^2 \tag{16}$$

where $\|.\|^2$ is weighted by $\Sigma \otimes Q$. If we assume that $A = \mathrm{A}(D\,W)$ is given, we may then solve for the following minimization problem $\|E\|^2 = \|Y - AX\|^2 \rightarrow \min$. We thus follow a two-step procedure to estimate X and W. The global minimizer for X is known to be $\hat{X} = (A^T Q^{-1} A)^{-1} A^T Q^{-1} Y$. This will then accordingly give

$$\begin{aligned} \hat{W} &= \arg\min_W \tfrac{1}{2}\|Y - A(A^T Q^{-1} A)^{-1} A^T Q^{-1} Y\|^2 \\ &= \arg\min_W \tfrac{1}{2}\|Y - A\hat{X}\|^2 \end{aligned} \tag{17}$$

where $A = \mathrm{A}(D\,W)$ and $Y - A\hat{X} = \hat{E}$. Our aim is thus firstly to minimize an objective function that depends only on the nonlinear parameters W, and then to proceed to estimate the linear parameters X (the first two representations in Eq. (17)). This two-step procedure is conceptually similar to solving a nonlinear least squares problem with separable variables (Golub and Pereyra, 1973; Teunissen, 1988a). For example, in the sine fitting problem $y(t) = x_1 \sin(\omega t) + x_2 \cos(\omega t) + x_3$, we may follow the above two-step procedure to estimate the nonlinear unknown ω along with the linear unknowns x_i, $i = 1, 2, 3$. Alternatively we may also firstly estimate $\hat{X}$, and then proceed to estimate W (the second representation in Eq. (17)). Both methods can be shown to provide identical results for SD.

The objective function ϕ should be minimized with respect to the unknown weights W. This nonlinear minimization problem can be solved by different optimization methods. Here, we use the SD method, which is a first-order iterative optimization algorithm for finding a local minimum of a differentiable function. The SD method modifies the weights using the following iterative manner (Teunissen, 1990):

$$W^{(t+1)} = W^{(t)} - \alpha \nabla\phi(W) \tag{18}$$

where $W^{(t+1)}$ is the updated weight matrix at iteration $t + 1$, α is the learning rate, a predefined positive scalar, and ∇ is the gradient operator. While we derive an analytical form for the gradient to ensure simplicity and ease of implementation, users may alternatively employ automatic differentiation for numerical differentiation of the cost function in Eq. (17), see Baydin et al. (2018). To calculate $\nabla\phi(W)$, a few partial derivatives are required to be obtained, which are based on the following theorem:

Theorem 1. *Let the objective functions be of the forms*

$$\phi_1 = \mathrm{tr}(B\mathrm{A}(D\,W)C) = \mathrm{tr}(BAC) \tag{19}$$

and

$$\phi_2 = \mathrm{tr}\big(E(B\mathrm{A}(DW))^{-1}C\big) = \mathrm{tr}\big(E(BA)^{-1}C\big) \tag{20}$$

where B, C, D, E, and W are given matrices of appropriate size, W contains running unknown variables, and $\mathrm{A}(.)$, the activation function, is assumed to be differentiable. Then their partial derivatives with respect to matrix W are

$$\frac{\partial\phi_1}{\partial W} = D^T(B^T C^T \odot A') \tag{21}$$

and

$$\frac{\partial\phi_2}{\partial W} = -D^T\big(B^T(A^T B^T)^{-1} E^T C^T (A^T B^T)^{-1} \odot A'\big) \tag{22}$$

respectively, where $\odot$ is the element-wise Hadamard product applied to two matrices of the same dimension, $A = \mathrm{A}(DW)$ is the design matrix, and A' is the first derivative of A (see e.g. Eq. (13)).

Proof. See Appendix A.

The first representation in Eq. (17) is used to obtain the gradient $\nabla\phi$. After a few simple mathematical operations, the objective function $\phi = \frac{1}{2}\|Y - A(A^T Q^{-1} A)^{-1} A^T Q^{-1} Y\|^2$ can be simplified to

$$\phi(W) = \frac{1}{2}\mathrm{tr}(Y^T Q^{-1} Y \Sigma^{-1}) - \frac{1}{2}\mathrm{tr}(U^T N^{-1} U \Sigma^{-1}) \tag{23}$$

where $N = A^T Q^{-1} A$ is an $n \times n$ matrix and $U = A^T Q^{-1} Y$ is an $n \times r$ matrix. Both N and U are functions of the unknown A and hence W, but we note that the first term in this equation is not a function of W. Eq. (23) gives the gradient $\nabla\phi$ as

$$\nabla\phi = \frac{\partial\phi(W)}{\partial W} = -\frac{1}{2}\frac{\partial\mathrm{tr}(U^T N^{-1} U \Sigma^{-1})}{\partial W} \tag{24}$$

To derive the above partial derivative with respect to W, the chain rule should be used because both N and U are functions of $A = \mathrm{A}(D\,W)$. In principle, four partial derivatives are to be calculated; one in U, one in U^T, and two in N. Due to the symmetric properties of the matrices $U^T N^{-1} U$ and Σ^{-1}, only two partial derivatives are required for computing the gradient $\nabla\phi$.

After a few simple mathematical operations, the above theorem, along with Eq. (24), will then give

$$\begin{aligned} \nabla\phi &= D^T\big(Q^{-1} A N^{-1} U \Sigma^{-1} U^T N^{-1} \odot A'\big) \\ &- D^T\big(Q^{-1} Y \Sigma^{-1} U^T N^{-1} \odot A'\big) \end{aligned} \tag{25}$$

Knowing that $\hat{X} = N^{-1}U$ is the least squares estimate of the unknowns X and $\hat{Y} = AN^{-1}U$ is the least squares estimate of the observations, Eq. (25) reads then $\nabla\phi = -D^T(Q^{-1}(Y - \hat{Y})\Sigma^{-1}\hat{X}^T \odot A')$, or simply

$$\nabla\phi = -D^T(Q^{-1}\hat{E}\Sigma^{-1}\hat{X}^T \odot A') \tag{26}$$

where $\hat{E} = Y - \hat{Y}$ are the least squares residuals. This, with Eq. (18), will then give

$$W^{(t+1)} = W^{(t)} + \alpha\, D^T(Q^{-1}\hat{E}\Sigma^{-1}\hat{X}^T \odot A') \tag{27}$$

which iterates to improve W in the SD algorithm. It incorporates the matrices D and A', the least squares estimates of $\hat{E}$ and $\hat{X}$ and the matrices Σ and Q of the observation covariance matrix $Q_{\mathrm{vec}(Y)}$.

The gradient $\nabla\phi$ can also be obtained from the second representation $\phi(W) = \frac{1}{2}\|Y - A\hat{X}\|^2$ in Eq. (17). The derivation is not given here but the gradient will be the same as in Eq. (26). Having $\nabla\phi$ available, we may implement the SD in an iterative procedure to estimate W (see Section 4.2).

Algorithm 1: Implementation of LSBDL using SD method

Input:
- obtain data/feature matrix D
- observation matrix Y
- matrices Σ and Q

Initialization:
- initialize weight matrix $W^{(0)}$ and $\Delta W^{(0)} = 0$
- set regularization parameter κ
- set learning rate parameter α
- set momentum parameter μ
- set softening parameter s
- set maximum iteration t_{max}
- provide convergence threshold ϵ

BEGIN

Do for $t = 0$ till t_{max}
- compute the activation matrix A and its derivative A'
- obtain the least squares estimates $\hat{X} = (A^T Q^{-1} A + \kappa I_n)^{-1} A^T Q^{-1} Y$
- compute the least squares residuals $\hat{E} = Y - A\hat{X}$
- compute the gradient matrix $\nabla\phi = -D^T(Q^{-1}\hat{E}\Sigma^{-1}\hat{X}^T \odot A')$
- soften the gradient by transformation $\nabla\phi = \text{sgn}(\nabla\phi) \odot |\nabla\phi|^s$
- compute weights' corrections $\Delta W^{(t+1)} = -\alpha\nabla\phi + \mu\Delta W^{(t)}$
- update the weights $W^{(t+1)} = W^{(t)} + \Delta W^{(t+1)}$

if $\|W^{(t)} - W^{(t-1)}\| > \epsilon$
- break loop

else
- increase counter $t = t + 1$

end if

End do

END

4.2. Implementation of SD method

To implement the above SD algorithm, the procedure is started with an appropriate initial weight $W^{(0)}$, and iterated through the above equations to modify the weights W. The iterations will continue till the convergence, for example when $\|W^{(t+1)} - W^{(t)}\|$ becomes smaller than a given threshold ϵ, or when the objective function $\phi(W)$ has sufficiently decreased (Teunissen, 1990). The method is applied to the weights (and biases) to finally compute the activation (design) matrix A. A summary of these steps is provided in Algorithm 1. Having the linear system of equations $Y = AX + E$ available, we may finally use the least squares method to estimate the unknown parameters X. Its least squares estimate is

$$\hat{X} = (A^T Q^{-1} A)^{-1} A^T Q^{-1} Y \tag{28}$$

where $A = \mathbb{A}(D\,W)$ is the trained design matrix and Q is a given $m \times m$ matrix. We may then also estimate $\hat{Y} = A\hat{X}$ and $\hat{E} = Y - \hat{Y}$.

The following remarks highlight the pros and cons of the SD method applied to LSBDL. One of the advantages of the SD method is its great simplicity. Only the first derivative of the objective function is required and no matrix inversion is needed. However, the rate of convergence of SD is at most linear even for a quadratic cost function. The SD can converge to a local minimum and slow down in a neighborhood of a saddle point. It experiences a zig-zag pattern as iterations progress, resulting in a low convergence rate (Teunissen, 1990). There are methods that improve the convergence speed of SD and can directly be applied to LSBDL.

The SD algorithm modifies the weights at time step $t+1$ as $W^{(t+1)} = W^{(t)} + \Delta W^{(t+1)}$ where $\Delta W^{(t+1)} = -\alpha\nabla\phi_t$. Modifications of gradient descent have been proposed to handle the above-mentioned deficiencies. A widely used method that helps the network out of local minima and breaks the zig-zag pattern is to use a momentum term as follows (Rumelhart et al., 1985):

$$\Delta W^{(t+1)} = -\alpha\nabla\phi_t + \mu\Delta W^{(t)} \tag{29}$$

Table 1
Typical range and the values used in this study for tuning hyper-parameters: learning rate, regularization parameter, momentum parameter, and softening parameter.

Parameter	Symbol	Typical range	Values used here
Regularization	κ	$(10^{-10}, 1)$	10^{-4}, 10^{-6}, 10^{-8}
Learning rate	α	$(10^{-10}, 1)$	10^{-2}, 10^{-3}, 10^{-4}
Momentum	μ	(0, 0.99)	0.9, 0.95
Softening	s	(0.1, 1)	0.5

where $\Delta W^{(t)}$ is the weight modification at time step t and $\mu \in (0, 1)$ is the momentum parameter, to be determined. The inclusion of a momentum term has been shown to increase the convergence rate dramatically (Qian, 1999; Sutskever et al., 2013).

There are alternative methods. It is generally unfavorable if training leads to large weight values. It is firstly because large weights can easily lead to the saturation region in which the gradients almost vanish, and secondly, large weights increase the risk of overfitting. We propose a method to soften the gradient by a fractional exponent transformation of the gradient matrix entries as follows:

$$G = \text{sgn}(G) \odot |G|^s \tag{30}$$

where sgn(G) and |.| are the sign and absolute operators and $s \in (0, 1)$ is the tunable softening parameter. We found $s = 0.5$, the square root, to be an efficient parameter in the above 'softening' method.

Two aspects on the implementation of LSBDL notably require attention from users when applying LSBDL in various contexts. (i) Effective implementation of LSBDL relies on appropriately setting the tuning parameters. Table 1 provides a list of typical values for the tuning parameters, consisting of learning rate, regularization parameter, momentum parameter, and softening parameter. Furthermore, for the three examples considered in our study, we have also presented the optimal values for these parameters in the same table. (ii) Selecting appropriate network architectures, also affecting the method's performance, depends on the complexity of the data. For example, simpler datasets perform well with a network consisting of just one layer with a few neurons per layer, whereas more complex datasets may require deeper networks with more neurons per layer. Users should adjust the number of layers and neurons based on the performance observed during validation (see e.g. Section 9.1). By iterating and tuning these parameters, users can tailor the network architecture to best fit the specific characteristics of the available data.

No training for X: In DL regression problems, the activation function of the output layer is typically set as an identity function $\mathbf{I}(U) = U$ (Kawaguchi, 2016). Traditionally both the output and hidden layer(s) are updated by the gradient descent methods using the back-propagation equations (Rumelhart et al., 1986; Hastie et al., 2009). In contrast to what is commonly used in literature, we followed a different procedure. The identity output layer with variables X does not require training in LSBDL. Although one can use the back-propagation method (see Section 7.3 on multilayer) to first update $X^{(t+1)} = X^{(t)} - \alpha\nabla\phi(X)$ and then W, we did not follow this procedure. Because the global minimizer for X is already known from the linear least squares theory, in Eq. (17), we substituted $\hat{X}$ with its global minimizer from Eq. (28). Therefore the objective function $\phi = \frac{1}{2}\|Y - A(A^T Q^{-1} A)^{-1} A^T Q^{-1} Y\|^2$ is only a function of W. Although Eq. (27) is a function of $\hat{X}$ and $\hat{E}$, they are both linear functions of Y; so they were introduced just for implementation convenience. This indicates that we just need to train W and therefore back-propagation is not applied when having one hidden layer.

4.3. Special cases of LSBDL

The general formulation of Eq. (27) can take into consideration special cases. We may simply consider $Q = I_m$ and $\Sigma = I_r$, resulting in $Q_{\text{vec}(Y)} = I_r \otimes I_m$. It then follows that

$$W^{(t+1)} = W^{(t)} + \alpha\, D^T(\hat{E}\hat{X}^T \odot A') \tag{31}$$

which can be applied to independent observations having identical precision.

For a better understanding of the LSBDL formulation, we denote $Z = Q^{-1}\hat{E}\Sigma^{-1}\hat{X}^T \odot A'$, an $m \times n$ matrix. Eq. (27) will then read $W^{(t+1)} = W^{(t)} + \alpha\, D^T Z$. The individual entries w_{uv}, $u = 1, \dots, k$, $v = 1, \dots, n$ can be obtained from multiplication of the u^{th} row of D^T with the v^{th} column of Z as

$$w_{uv}^{(t+1)} = w_{uv}^{(t)} + \alpha \sum_{i=1}^{m} d_{ui}^T z_{iv} = w_{uv}^{(t)} + \alpha \sum_{i=1}^{m} d_{iu} z_{iv} \tag{32}$$

Two special cases will be addressed for the general formulation. (1) When Q and Σ are diagonal matrices, i.e., $Q = \text{diag}(\sigma_{1,q}^2, \dots, \sigma_{m,q}^2)$, uncorrelated samples, and $\Sigma = \text{diag}(\sigma_{1,s}^2, \dots, \sigma_{r,s}^2)$, uncorrelated classes, the entries z_{iv} simplify to

$$z_{iv} = a'_{iv} \sum_{j=1}^{r} \frac{\hat{x}_{vj}\hat{e}_{ij}}{\sigma_{i,q}^2 \sigma_{j,s}^2} \tag{33}$$

which, with Eq. (32), leads to

$$w_{uv}^{(t+1)} = w_{uv}^{(t)} + \alpha \sum_{i=1}^{m} d_{iu} a'_{iv} \sum_{j=1}^{r} \frac{\hat{x}_{vj}\hat{e}_{ij}}{\sigma_{i,q}^2 \sigma_{j,s}^2} \tag{34}$$

This equation indicates that the residuals of all observations will contribute when training w_{uv}. However, observations with higher precision (smaller $\sigma_{i,q}$ and $\sigma_{j,s}$) will have a larger contribution in the training. (2) When observations are equally weighted $\sigma_{i,q} = 1$, $i = 1, \dots, m$ and $\sigma_{j,s} = 1$, $j = 1, \dots, r$, the above equation reads

$$w_{uv}^{(t+1)} = w_{uv}^{(t)} + \alpha \sum_{i=1}^{m} d_{iu} a'_{iv} \sum_{j=1}^{r} \hat{x}_{vj}\hat{e}_{ij} \tag{35}$$

which is indeed identical to the element-wise weight estimation proposed in the literature (Rumelhart et al., 1986; Hastie et al., 2009).

Another special case considers the univariate model, so $r = 1$, $Y = y$ and therefore $Q_{\text{vec}(Y)} = Q_y = \sigma^2 Q$. It then follows that

$$W^{(t+1)} = W^{(t)} + \alpha\, D^T (Q_y^{-1}\hat{e}\hat{x}^T \odot A') \tag{36}$$

For uncorrelated observations, $Q_y = \text{diag}(\sigma_1^2, \dots, \sigma_m^2)$, this will then result in ($u = 1, \dots, k$ and $v = 1, \dots, n$)

$$w_{uv}^{(t+1)} = w_{uv}^{(t)} + \alpha \sum_{i=1}^{m} d_{iu} a'_{iv} \frac{\hat{x}_v \hat{e}_i}{\sigma_i^2} \tag{37}$$

indicating again the impact of observations' precision σ_i's in the training.

5. Gauss–Newton (GN) formulation of LSBDL

The steepest descent (SD) method was elaborated in Section 4. The second gradient-based method to train the design matrix $A = \mathtt{A}(D\,W)$ is Gauss–Newton (GN) (Teunissen, 1990). GN methods in the realm of DL, as discussed in the literature, utilize second-order derivatives known as the Hessian matrix (Botev, 2020). This approach is often referred to as Newton method in the broader literature (Xu et al., 2020). Calculating and storing the Hessian matrix for AI applications is usually infeasible due to their large size, which renders these methods computationally expensive and complex and therefore less practical for large-scale applications (Botev et al., 2017).

Our GN method is classified among the gradient-based method (first-order derivative), where the Jacobian matrix is needed (Hartley, 1961; Ruhe, 1979; Gratton et al., 2007). The GN idea is to approximate/linearize the objective function (within the norm) using the Taylor series expansion. GN has a faster convergence rate than SD (Teunissen, 1990) (see later Fig. 2).

5.1. Derivation of Jacobian matrix

Similar to Section 4, we consider two-step procedures to estimate X and W. Our aim is firstly to estimate the nonlinear parameters W, and then to proceed to estimate the linear parameters X, and vice versa. The starting point is again Eq. (17), now reformulated as $\hat{W} = \arg\min_W \frac{1}{2}\|Y - P_A^{\perp} Y\|^2$, where the design matrix $A = \mathtt{A}(D\,W)$ is to be trained using the GN method by estimating the weight matrix W. The Jacobian matrix is derived in this subsection for which two formulas are presented.

The non-linearity of $\|Y - P_A^{\perp} Y\|^2$ is due to the non-linearity of $P_A = P_{\mathtt{A}(DW)}$, because the quadratic form $\|.\|^2$ will disappear in the differentiation. The GN idea is then to approximate/linearize P_A (within the norm) using the Taylor series expansion as follows (Teunissen, 1990):

$$P_A^{(t+1)} = P_A^{(t)} + \partial_W P_A^{(t)} (W - W^{(t)}) \tag{38}$$

where $P_A^{(t+1)} = P_{\mathtt{A}(DW^{(t+1)})}$, and so is $P_A^{(t)}$. The above equation, with Eq. (17), gives

$$\hat{W} = \frac{1}{2} \arg\min_W \|\hat{E} - \partial_W P_{\mathtt{A}(DW)} Y (W - W^{(t)})\|^2 \tag{39}$$

where the residuals $\hat{E} = Y - P_A Y$ are calculated at $A = \mathtt{A}(DW^{(t)})$.

Four partial derivatives play a role in computing the Jacobian matrix. It is known that the differentiation, $\mathtt{d}(.)$, of an inverse matrix, $N^{-1} = (A^T Q^{-1} A)^{-1}$, is

$$\mathtt{d}(N^{-1}) = -N^{-1}\mathtt{d}(N)N^{-1} = -N^{-1}\mathtt{d}(A^T Q^{-1} A)N^{-1} \tag{40}$$

which leads to

$$\begin{aligned} \partial_W P_{\mathtt{A}(DW)} Y \;&=\; [\partial_W A] N^{-1} A^T Q^{-1} Y \\ &-\; A N^{-1} A^T Q^{-1} [\partial_W A] N^{-1} A^T Q^{-1} Y \\ &-\; A N^{-1} [\partial_W A^T] Q^{-1} A N^{-1} A^T Q^{-1} Y \\ &+\; A N^{-1} [\partial_W A^T] Q^{-1} Y \end{aligned} \tag{41}$$

This can further be simplified to

$$\partial_W P_{\mathtt{A}(DW)} Y = P_A^{\perp} [\partial_W A] \hat{X} + A N^{-1} [\partial_W A^T] Q^{-1} \hat{E} \tag{42}$$

or

$$\partial_W P_{\mathtt{A}(DW)} y_i = P_A^{\perp} [\partial_W A] \hat{x}_i + A N^{-1} [\partial_W A^T] Q^{-1} \hat{e}_i \tag{43}$$

for $i = 1, \dots, r$. To obtain the Jacobian matrix, the above partial derivatives should be derived. They are based on the following theorem:

Theorem 2. *Let the m-vector h_1 and h_2 be of the forms*

$$h_1 = BAz = B\mathtt{A}(DW)z \tag{44}$$

and

$$h_2 = CA^T z = C\mathtt{A}(W^T D^T)z \tag{45}$$

where B, C and D are given matrices of appropriate sizes, z is a given vector, W contains running unknown variables and $\mathtt{A}(.)$, the activation function, is assumed to be differentiable. Then their Jacobian matrices are

$$J_1 = \partial_w h_1 = B[\partial_W A]z = B\big((u_m z^T \otimes u_k^T) \odot M\big) \tag{46}$$

and

$$J_2 = \partial_w h_2 = C[\partial_W A^T]z = (C \otimes u_k^T) \odot (u_m z^T M) \tag{47}$$

where $w = \text{vec}(W)$, and

$$M = (A' \otimes u_k^T) \odot (u_n^T \otimes D) \tag{48}$$

with u_k, u_n, u_m the summation vectors of sizes k, n and m, respectively.

Proof. See Appendix B.

For the sake of convenience, we introduce the kn-vector $w = \text{vec}(W)$. Eq. (39) can then be reformulated to

$$\hat{W} = \frac{1}{2} \arg\min_W \|\text{vec}(\hat{E}) - J_w(\hat{X}, \hat{E})(w - w^{(t)})\|^2 \tag{49}$$

where the $mr \times kn$ Jacobian matrix $J_w = J_w(\hat{X}, \hat{E})$ can be obtained as

$$J_w = \begin{bmatrix} J_1(\hat{x}_1, \hat{e}_1) \\ \vdots \\ J_r(\hat{x}_r, \hat{e}_r) \end{bmatrix} = \sum_{i=1}^{r} c_i \otimes J_i(\hat{x}_i, \hat{e}_i) \tag{50}$$

with c_i the canonical unit vector. The individual $m \times kn$ Jacobian matrices $J_i = J_i(\hat{x}_i, \hat{e}_i)$, $i = 1, \dots, r$, can be obtained from multiple application of Theorem 2 to Eq. (43). This will then give

$$\begin{aligned} J_i(\hat{x}_i, \hat{e}_i) &= P_A^{\perp}\left((u_m \hat{x}_i^T \otimes u_k^T) \odot M\right) \\ &+ (AN^{-1} \otimes u_k^T) \odot (u_m \hat{e}_i^T Q^{-1} M) \end{aligned} \tag{51}$$

where

$$M = (A' \otimes u_k^T) \odot (u_n^T \otimes D) \tag{52}$$

is an $m \times kn$ matrix.

An alternative formula for the two-step procedure is to firstly estimate $\hat{X}$ and then $\hat{W}$. We will thus use the last formula in Eq. (17): $\hat{W} = \arg\min_W \frac{1}{2}\|Y - A\hat{X}\|^2$. The non-linearity is then due to the non-linearity of $A = \mathbb{A}(DW)$. The GN linearizes A (within the norm) using the Taylor series expansion, which leads to

$$\partial_W \mathbb{A}(DW)\hat{x}_i = [\partial_W A]\hat{x}_i \tag{53}$$

for $i = 1, \dots, r$. The Jacobian matrix can then be obtained from Eq. (50), where the individual $m \times kn$ matrices are obtained from the application of Theorem 2 to the above equation (in J_1 of Theorem 2, take $B = I$ and $z = \hat{x}_i$)

$$J_i = (u_m \hat{x}_i^T \otimes u_k^T) \odot M \tag{54}$$

The above Jacobian matrices can now be used to estimate W in an iterative GN method.

5.2. Implementation of GN method

The Jacobian matrix J_w along with the covariance matrix $Q_{\mathrm{vec}(Y)}$ can now be used to estimate $\delta\hat{w} = w^{(t+1)} - w^{(t)}$ as follows:

$$\delta\hat{w} = \left(J_w^T Q_{\mathrm{vec}(Y)}^{-1} J_w\right)^{-1} J_w^T Q_{\mathrm{vec}(Y)}^{-1} \mathrm{vec}(\hat{E}) \tag{55}$$

Having substituted $J_w^T = \sum_{i=1}^r c_i^T \otimes J_i^T$, $Q_{\mathrm{vec}(Y)}^{-1} = \Sigma^{-1} \otimes Q^{-1}$, $J_w = \sum_{j=1}^r c_j \otimes J_j$ and $\mathrm{vec}(\hat{E}) = \sum_{j=1}^r c_j \otimes \hat{e}_j$, it follows that

$$\delta\hat{w} = \left(\sum_{i=1}^{r}\sum_{j=1}^{r} \sigma_{ij}^{-1} J_i^T Q^{-1} J_j\right)^{-1} \sum_{i=1}^{r}\sum_{j=1}^{r} \sigma_{ij}^{-1} (J_i^T Q^{-1} \hat{e}_j) \tag{56}$$

where $\sigma_{ij}^{-1} = \left(\Sigma^{-1}\right)_{ij}$ are the entries of Σ^{-1} (just for notational convenience). In a special case where $\Sigma = \mathrm{diag}(\sigma_{11}, \dots, \sigma_{rr})$ is a diagonal matrix, it follows that

$$\delta\hat{w} = \left(\sum_{i=1}^{r} \sigma_{ii}^{-1} J_i^T Q^{-1} J_i\right)^{-1} \sum_{i=1}^{r} \sigma_{ii}^{-1} (J_i^T Q^{-1} \hat{e}_i) \tag{57}$$

In a special case where $\Sigma = \sigma^2 I_r$, it follows that

$$\delta\hat{w} = \left(\sum_{i=1}^{r} J_i^T Q^{-1} J_i\right)^{-1} \sum_{i=1}^{r} J_i^T Q^{-1} \hat{e}_i \tag{58}$$

Further simplification of the above equation can consider uncorrelated and equally-weighted observations, i.e. $Q = I_m$, which leads to

$$\delta\hat{w} = \left(\sum_{i=1}^{r} J_i^T J_i\right)^{-1} \sum_{i=1}^{r} J_i^T \hat{e}_i \tag{59}$$

The estimated $\delta\hat{w}$ from the above equations can be used to update $w^{(t+1)} = w^{(t)} + \delta\hat{w}$, resulting in an update for $W^{(t+1)}$. This can accordingly be used to update $A = \mathbb{A}(DW^{(t+1)})$ and therefore $\hat{X}$. The iterative procedure will continue till the convergence, for example when $\|w^{(t+1)} - w^{(t)}\|$ becomes smaller than the convergence threshold ϵ, or when the objective function $\phi(W)$ has sufficiently decreased. A summary of these steps is provided in Algorithm 2.

The following remarks highlight the pros and cons of the GN method applied to LSBDL. The GN method is known to have a faster convergence rate compared to the SD method. The rate of convergence of GN is at least linear, but can be quadratic for special cases of flat manifolds and/or consistent data (Teunissen, 1990). The method takes advantage of the quadratic-form structure of the objective function. The GN method allows to directly apply the existing body of knowledge of the least squares method to LSBDL. There are also disadvantages for the GN-based LSBDL. Because the $kn \times kn$ normal matrix $J_w^T Q_{\mathrm{vec}(Y)}^{-1} J_w$ is to be inverted in GN, the computational burden can drastically increase even for moderate values of k and n.

Algorithm 2: Implementation of LSBDL using GN method

Input:
- obtain data/feature matrix D
- observation matrix Y
- matrices Σ and Q

Initialization:
- initialize weight matrix $W^{(0)}$ and $\Delta W^{(0)} = 0$
- set regularization parameter κ
- set learning rate parameter, usually $\alpha = 1$
- set maximum iteration t_{max}
- provide convergence threshold ϵ

BEGIN

Do for $t = 0$ till t_{max}
- compute activation matrix A and its derivative A'
- obtain least squares estimates $\hat{X}$ using Eq. (69)
- compute least squares residuals $\hat{E} = Y - A\hat{X}$
- compute Jacobian matrix J_w from Eqs. (50) and (51)
- estimate $\delta\hat{w}$ from Eq. (55)
- update $w^{(t+1)} = w^{(t)} + \alpha\delta\hat{w}$
- reshape $w^{(t+1)}$ to obtain $k \times n$ matrix $W^{(t+1)}$

if $\|W^{(t)} - W^{(t-1)}\| > \epsilon$
- break loop

else
- increase counter $t = t + 1$

end if

End do

END

6. Embedding of LSBDL in least squares framework

LSBDL is a transparent and explainable machine learning method, as will be highlighted in this section. Its formulation in the least squares framework allows for direct application of well-established theories to LSBDL. Here, we extend some of these theories for LSBDL.

6.1. Covariance matrix of estimates

In the machine learning domain research is ongoing to include data uncertainty in machine learning and its applications to geodetic data science (Shahvandi and Soja, 2022). Having the LSBDL formulation allows to obtain the precision of the least squares estimates $\hat{X}$, $\hat{Y}$ and $\hat{E}$ using the Gauss–Newton formulation. As a first approximation, we may directly use Eq. (9) to obtain the covariance matrices $Q_{\mathrm{vec}(\hat{X})}$, $Q_{\mathrm{vec}(\hat{Y})}$ and $Q_{\mathrm{vec}(\hat{E})}$, if we ignore the uncertainty of $\hat{W}$.

On the other hand, if we take a one-step procedure to simultaneously estimate X and W, we may use the following linearized system of equations:

$$\mathrm{vec}(Y) = \begin{bmatrix} A_x & J_w \end{bmatrix} \begin{bmatrix} \hat{x} \\ \delta\hat{w} \end{bmatrix} + \mathrm{vec}(E) \tag{60}$$

where $A_x = I \otimes A$ and $\hat{x} = \mathrm{vec}(\hat{X})$. This equation can be used to investigate the estimability of X and W. If the columns of the

design matrix $[A_x\ J_w]$ are linearly independent (full column rank), the parameters $\hat{x} = \mathrm{vec}(\hat{X})$ and $\hat{w} = \mathrm{vec}(\hat{W})$ are unbiased estimable. Then, the inverse of the normal matrix directly gives the covariance matrix of the estimated parameters

$$\mathrm{D}\begin{bmatrix}\hat{x}\\ \hat{w}\end{bmatrix} = \begin{bmatrix} Q_{\hat{x}} & Q_{\hat{x}\hat{w}} \\ Q_{\hat{w}\hat{x}} & Q_{\hat{w}}\end{bmatrix} = \begin{bmatrix} N_{xx} & N_{xw} \\ N_{wx} & N_{ww}\end{bmatrix}^{-1} \tag{61}$$

where

$$\begin{aligned} N_{xx} &= A_x^T Q_y^{-1} A_x = \Sigma^{-1} \otimes A^T Q^{-1} A \\ N_{xw} &= A_x^T Q_y^{-1} J_w \\ N_{wx} &= J_w^T Q_y^{-1} A_x \\ N_{ww} &= J_w^T Q_y^{-1} J_w \end{aligned} \tag{62}$$

with $Q_y = Q_{\mathrm{vec}(Y)}$. Based on Eq. (61), the covariance matrix of $\hat{x} = \mathrm{vec}(\hat{X})$ can be obtained as (Teunissen, 2000b)

$$Q_{\hat{x}} = \left(N_{xx} - N_{xw}N_{ww}^{-1}N_{wx}\right)^{-1} \tag{63}$$

which is a generalized form of the covariance matrix of estimates $\hat{X}$ in Eq. (9), taking also the uncertainties of $\hat{W}$ into account. The covariance matrix of $\hat{w} = \mathrm{vec}(\hat{W})$ can also be approximated as

$$Q_{\hat{w}} = \left(N_{ww} - N_{wx}N_{xx}^{-1}N_{xw}\right)^{-1} \tag{64}$$

In a similar manner, the covariance matrix of $\hat{y} = \mathrm{vec}(\hat{Y})$ and $\hat{e} = \mathrm{vec}(\hat{E})$ can accordingly be determined. Without any derivation we have

$$\begin{aligned} Q_{\hat{y}} &= [A_x,\ J_w]\begin{bmatrix} Q_{\hat{x}} & Q_{\hat{x}\hat{w}} \\ Q_{\hat{w}\hat{x}} & Q_{\hat{w}}\end{bmatrix}\begin{bmatrix} A_x^T \\ J_w^T\end{bmatrix} \\ &= A_x Q_{\hat{x}} A_x^T + A_x Q_{\hat{x}\hat{w}} J_w^T + J_w Q_{\hat{w}\hat{x}} A_x^T + J_w Q_{\hat{w}} J_w^T \end{aligned} \tag{65}$$

and

$$Q_{\hat{e}} = Q_y - Q_{\hat{y}} \tag{66}$$

both extracted from the standard least squares method. Two special cases of the above formulations can occur as follows:

Case 1: In a special case when $N_{ww} = 0$ (e.g. when W is known), Eq. (63) yields

$$Q_{\hat{x}} = N_{xx}^{-1} = \Sigma \otimes (A^T Q^{-1} A)^{-1} \tag{67}$$

which is identical to $Q_{\mathrm{vec}(\hat{X})}$ in Eq. (9).

Case 2: Another special case occurs when $N_{xx} = 0$ (e.g. when X is known). In this case, Eq. (64) simplifies to

$$Q_{\hat{w}} = Q_{\mathrm{vec}(\hat{W})} = N_{ww}^{-1} = \left(J_w^T Q_{\mathrm{vec}(Y)}^{-1} J_w\right)^{-1} \tag{68}$$

indicated also in Eq. (55). $Q_{\hat{y}}$ and $Q_{\hat{e}}$ can accordingly be obtained for the above two special cases.

6.2. Statistical hypothesis testing

The LSBDL formulation allows to apply the hypothesis testing to its linear(ized) model (see some challenges in Section 8.2). In fact, the existing theories in the DIA procedure can directly follow as (Baarda, 1968; Teunissen, 2000a, 2018):

Detection: The overall model test (OMT) in Eqs. (10) and (11) can be performed to test the general consistency between the observations and the model. If the test is rejected this indicates that there are unresolved issues on the consistency between the model and observations, of which outliers could be a cause.

Identification: When the OMT is rejected, the w-test statistics in Eq. (12) can directly be applied to specified alternative hypotheses, pointing to individual observations for example, to identify the potential source of model error such as systematic errors or outlying observations.

Adaptation: When outliers are confidently identified, they are then compensated for in the functional model; adaptation of matrix A by removing outliers (less rows in A), or introducing new unknowns (more columns in A). The final $\hat{X}$ and its statistical inference can then be presented, with all outlying samples removed.

In Section 9 we use the above DIA procedure in a surface fitting problem and a time series example using the LSBDL.

6.3. Regularization of solution

There could arise complications on the application of deep learning in real-word problems. When there are too many unknown weights and biases in W the system of equations $Y = AX + E$ becomes ill-posed, referred to as over-parametrization or overfitting. For the deep learning formulation, the design matrix $A = \mathtt{A}(D\,W)$ is then nearly rank deficient and therefore the normal matrix $A^T Q^{-1} A$ or that in Eq. (61) cannot be regularly inverted. This can happen when the number of columns n is (much) larger than the number of rows k in W. To obtain a regular solution, regularization methods can be used. It is a technique to prevent the model from overfitting by posing extra information to the model.

In deep learning algorithms, data augmentation is often considered as a type of regularization (Hernández-García and König, 2018). In image classification, data augmentation concerns randomly selecting cropped regions from images, flipping images horizontally, introducing scaling and rotation to images, which adds observations to Y (increasing m). Although data augmentation can also be implemented in the LSBDL formulation, one other way to obtain a regular solution is to take possible prior information on the unknown parameters into account. The Tikhonov regularization method has been widely used with least squares (Tikhonov, 1963). The least squares estimate of X in Eq. (28) is then modified to

$$\hat{X} = (A^T Q^{-1} A + \kappa I_n)^{-1} A^T Q^{-1} Y \tag{69}$$

where κ is the regularization parameter, to be determined. One way to determine κ is to use the L-curve method (Hansen, 1999, 2001). An alternative method that avoids introducing the regularization parameter is to use the theory of generalized inverses (Penrose, 1955; Teunissen, 1985; Ben-Israel and Greville, 2003). The generalized inverse of the normal matrix $N = A^T Q^{-1} A$ can be used to obtain a regularized solution as

$$\hat{X} = (A^T Q^{-1} A)^{-} A^T Q^{-1} Y \tag{70}$$

where $(.)^{-}$ is the generalized inverse of a matrix. A particular class of generalized inverses is called pseudoinverse or Moore–Penrose inverse (Greville, 1959; Barata and Hussein, 2012). The above equation will then read

$$\hat{X} = (A^T Q^{-1} A)^{+} A^T Q^{-1} Y \tag{71}$$

where $(.)^{+}$ is the pseudoinverse of a matrix.

The above formulations are an extension of the regularized least squares solution to deep learning. This can be used to estimate the final X, but also to train the network in earlier steps. The above regularization methods can also be used in Eq. (55) of the GN method and in Eq. (61).

6.4. Variance component estimation

So far we assumed that the covariance matrix $Q_{\mathrm{vec}(Y)}$ of observations is known. This however will not be the case for many practical applications. We consider the case when the covariance matrix has a Kronecker structure as $Q_{\mathrm{vec}(Y)} = \Sigma \otimes Q$, see Eq. (6). We may have different possibilities to estimate its components. The matrix Σ expresses covariances among different classes, whereas Q expresses (co)variances among different sequences within a specific class.

In practice a few (co)variance components can be assumed unknown in Σ and Q. Least squares variance component estimation (LS-VCE) can be used to estimate such components (Teunissen, 1988b; Teunissen and

Amiri-Simkooei, 2008). For example, if Q is known, we may just simply estimate Σ. The least squares estimate of Σ is (Amiri-Simkooei, 2009)

$$\hat{\Sigma} = \frac{\hat{E}^T Q^{-1} \hat{E}}{m-n} \tag{72}$$

where $\hat{E} = Y - \hat{Y}$ is the least squares estimate of matrix E.

The $m \times m$ matrix Q contains (co)variance information of different observations. In the context of scientific and engineering applications, with real-valued observations (real numbers), Q can play the role of the covariance matrix. The inverse of Q, i.e. Q^{-1}, is used to weigh the contribution of different observations in the final estimates $\hat{X}$ and $\hat{W}$, obtained by SD and GN, see Eqs. (27) and (57). There are also possibilities to estimate unknown components in Q, for example when combining data from different sensors. By merging heterogeneous information, data fusion has been proven to improve the performance of ML models in many application fields (Arrieta et al., 2020). LS-VCE can be used to estimate possible variances for each group of observations or each sensor. The variances are then the unknown components in Q, and the univariate (Teunissen and Amiri-Simkooei, 2008) or multivariate (Amiri-Simkooei, 2009) formulation of LS-VCE can be used to estimate such variances in the data fusion stochastic model.

Matrix Q can also play a significant role in classification problems. In ML classification methods, we mostly encounter to predict discrete targets such as a binary decision of 'yes' or 'no' (or '1' and '0'). In this condition, Q^{-1} can also play an important role in down-weighting or up-weighting specific samples. It is well possible that some samples are identified as gross errors (blunders), which are to be removed or down-weighted in the estimation process. Also within a specific class, the distribution of discrete values can vary significantly. Then, due to the distribution difference in each class (or among different classes), the algorithms tend to get biased towards the majority values present and likely do not perform well on the minorities. Up-weighting is then a way to fully address the presence and impact of the minorities on the training performance.

6.5. Performance on test data

In ML, data sets are usually split into two subsets: training and testing. The training data are fed into LSBDL to estimate W and X. Then, the trained network can directly be used to predict future events. Prior to prediction, the performance of the trained network is investigated on the test data for which we have both Y_t and D_t available. With the estimated $\hat{W}$ and $\hat{X}$, we may then predict

$$\begin{aligned} \hat{Y}_t &= \mathtt{A}(D_t\hat{W})\hat{X} = A_t\hat{X}, \\ \hat{E}_t &= Y_t - \mathtt{A}(D_t\hat{W})\hat{X} = Y_t - \hat{Y}_t \end{aligned} \tag{73}$$

where $A_t = \mathtt{A}(D_t\hat{W})$, $\hat{Y}_t$ is a prediction for Y_t, which is the test data but is unavailable for future events, and $\hat{E}_t$ is the prediction error. $\hat{E}_t$ includes both the errors of new samples Y_t (sampling noise) and the errors of the trained network (network noise due to $\hat{W}$ and $\hat{X}$).

Knowing that $\hat{Y}_t = \mathtt{A}(D_t\hat{W})\hat{X}$, it follows that the GN design matrix of the test data is $[A_{xt},\ J_{wt}]$. Similar to Eq. (65), this allows to apply the error propagation law to obtain the covariance matrix of $\hat{y}_t = \mathrm{vec}(\hat{Y}_t)$ as

$$Q_{\hat{y}_t} = A_{xt}Q_{\hat{x}}A_{xt}^T + A_{xt}Q_{\hat{x}\hat{w}}J_{wt}^T + J_{wt}Q_{\hat{w}\hat{x}}A_{xt}^T + J_{wt}Q_{\hat{w}}J_{wt}^T \tag{74}$$

To obtain the covariance matrix of the predicted error $\hat{e}_t = \mathrm{vec}(\hat{E}_t)$ an extra term due to the presence of $y_t = \mathrm{vec}(Y_t)$ should be added to the preceding equation. This leads to

$$Q_{\hat{e}_t} = Q_{y_t} + Q_{\hat{y}_t} \tag{75}$$

meaning that the $Q_{\hat{y}_t}$ is indeed added to Q_{y_t}; note in Eq. (66), the term $Q_{\hat{y}}$ is subtracted from Q_y to obtain $Q_{\hat{e}}$. The first term $Q_{y_t=\mathrm{vec}(Y_t)}$ in the above equation indicates errors of the new samples Y_t (sampling noise), whereas the second term $Q_{\hat{y}_t}$ expresses the error of the trained network (network noise).

The estimated Σ in Eq. (72) is based on the training data. ML techniques are known for a tendency to suffer from overfitting problems, leading to the residuals of the training data to be quite small and hence to an underestimation of Σ. A more realistic estimate of Σ may rely on the test residuals $\hat{E}_t$. This will then give

$$\hat{\Sigma}_t = \frac{\hat{E}_t^T Q^{-1} \hat{E}_t}{m_t} \tag{76}$$

where m_t is the number of samples in the test data. The above equation includes both the network noise and the sampling noise. This is a more realistic measure to investigate the performance of the trained network. For example, the covariance matrices of $\hat{X}$, $\hat{Y}$, $\hat{E}$, $\hat{Y}_t$ and $\hat{E}_t$ can all be adapted when $\hat{\Sigma}_t$ is used instead of $\hat{\Sigma}$. In this case, the covariance matrices in Eq. (9) can be replaced by

$$\begin{aligned} \hat{Q}_{\mathrm{vec}(\hat{X})} &= \hat{\Sigma}_t \otimes (A^T Q^{-1} A)^{-1} \\ \hat{Q}_{\mathrm{vec}(\hat{Y})} &= \hat{\Sigma}_t \otimes P_A Q \\ \hat{Q}_{\mathrm{vec}(\hat{E})} &= \hat{\Sigma}_t \otimes P_A^{\perp} Q \end{aligned} \tag{77}$$

which use $\hat{\Sigma}_t$ from Eq. (76).

7. Further development of LSBDL

In this section, we will further develop on the LSBDL formulation based on the steepest descent (SD) method. The subject on GN is beyond the scope of this contribution. Three topics will be covered.

7.1. Reduction of computational burden in DL model

There are occasions of the linear system of equations being very large. This can happen when 'too many' data points are available (m is too large), which is often the case with machine learning techniques as a large number of training samples is available. Then, the observations Y can be split into p groups as $Y = [Y_1^T, \ldots, Y_p^T]^T$, where each group contains m_i observations as $Y_i \in \mathbb{R}^{m_i \times r}$, $i = 1, \ldots, p$, in conjunction with the mini-batch training used in DL (Li et al., 2014). The feature matrix can accordingly be row-wise partitioned to smaller matrices D_i. In this case, the linear system of equations can be rewritten as follows:

$$Y = AX + E = \begin{bmatrix} Y_1 \\ \vdots \\ Y_p \end{bmatrix} = \begin{bmatrix} A_1 \\ \vdots \\ A_p \end{bmatrix} X + \begin{bmatrix} E_1 \\ \vdots \\ E_p \end{bmatrix} \tag{78}$$

where $A_i = \mathtt{A}(D_i W) \in \mathbb{R}^{m_i \times n}$ is the design matrix corresponding to observations Y_i. Matrix Q is assumed to be of the form $Q = \mathrm{blkdiag}(Q_1, \ldots, Q_p)$, where $Q_i \in \mathbb{R}^{m_i \times m_i}$ and blkdiag is the block diagonal operator. We note that the unknowns X and W are the same among all groups.

To handle the above group-wise model, the batch and recursive least squares theory can be used to solve for X and W. Its batch formulation reads

$$\hat{X} = \Big(\sum_{i=1}^{p} A_i^T Q_i^{-1} A_i\Big)^{-1} \Big(\sum_{i=1}^{p} A_i^T Q_i^{-1} Y_i\Big) \tag{79}$$

Given $\hat{X}$, the weight matrix W of Eq. (27) can be updated using the following iterative procedure:

$$W^{(t+1)} = W^{(t)} + \alpha \sum_{i=1}^{p} D_i^T (Q_i^{-1} \hat{E}_i \Sigma^{-1} \hat{X}^T \odot A_i') \tag{80}$$

where $\hat{E}_i = Y_i - A_i\hat{X}$ and A_i' is the derivative of A_i. Having a new weight matrix W available, we may update $A_i = \mathtt{A}(D_i W)$, and repeat the procedure for a new iteration over t.

This problem can also be solved in a recursive form. In this scenario, the unknowns X and W are both updated in a recursive form. Using the first set of data Y_1, we obtain

$$\hat{X}_{(1)} = (A_1^T Q_1^{-1} A_1)^{-1} A_1^T Q_1^{-1} Y_1 \tag{81}$$

and

$$W_{(1)} = W_{(0)} + \alpha D_1^T (Q_1^{-1} \hat{E}_1 \Sigma^{-1} \hat{X}_{(1)}^T \odot A_1') \tag{82}$$

where $\hat{E}_1 = Y_1 - A_1 \hat{X}_{(1)}$ and A_1' is the derivative of A_1. Having $\hat{X}_{(i-1)}$ and $W_{(i-1)}$ available from a set of observations $Y_1, \dots, Y_{i-1}$, we may add the new set Y_i, to update X and W in a recursive form as follows (Teunissen, 2001; Amiri-Simkooei, 2019)

$$\hat{X}_{(i)} = \hat{X}_{(i-1)} - R_{i-1} A_i^T (Q_i + A_i R_{i-1} A_i^T)^{-1} (Y_i - A_i \hat{X}_{(i-1)}) \tag{83}$$

where $A_i = \mathrm{A}(D_i W_{(i-1)})$. Matrix W is then updated as

$$W_{(i)} = W_{(i-1)} + \alpha D_i^T (Q_i^{-1} \hat{E}_i \Sigma^{-1} \hat{X}_{(i)}^T \odot A_i') \tag{84}$$

where $\hat{E}_i = Y_i - A_i \hat{X}_{(i)}$. In the above equations, starting from $R_1 = (A_1^T Q_1^{-1} A_1)^{-1}$, R_i is updated as follows:

$$R_i = R_{i-1} - R_{i-1} A_i^T (Q_i + A_i R_{i-1} A_i^T)^{-1} A_i R_{i-1} \tag{85}$$

Having finished the first iteration, with $i = 1, \dots, p$ while $t = 1$, we may restart the same procedure to do the next iteration over t.

For the above recursive formulation, as an alternative method, we may update only X, while W is kept fixed. We then update $\hat{X}_{(i)}$, $i = 1, \dots, p$, and the final $\hat{X}_{(p)}$ is in fact $\hat{X}$. Eq. (80) can then be used to update W, and the next iteration over t can be scheduled.

7.2. Local feature extraction in DL model

Deep learning is known to be a powerful local feature extractor. Feature extraction aims to reduce the number of explanatory variables (features), in case there are too many variables. For example, a convolutional neural network (CNN), a class of deep neural networks, can successfully capture the spatial and temporal dependencies in an image through the application of relevant filters (Albawi et al., 2017). Its architecture can perform a better fitting due to the reduction in the number of parameters involved in a local area. This is referred to as the local feature extraction of CNN.

Although, in general, different local feature extractors can be defined, we propose a column-wise partitioning of the feature matrix. Matrix D can be decomposed into smaller sub-features as $[D_1, \dots, D_p]$, with D_i, $i = 1, \dots, p$ an $m \times k_i$ matrix. We note that we may have in general $\sum_{i=1}^{p} k_i \geq k$, indicating overlaps among the columns of D, consistent with the CNN stride structure (Kong and Lucey, 2017). The individual matrices D_i are then convolved with their individual weight matrices as $D_i W_i$. They will then be fed to an activation function and yield the linear model $Y = AX + E$ as

$$\begin{aligned} Y &= [\mathrm{A}(D_1 W_1), \dots, \mathrm{A}(D_p W_p)] \begin{bmatrix} X_1 \\ \vdots \\ X_p \end{bmatrix} + E \\ &= \textstyle\sum_{i=1}^{p} A_i X_i + E \end{aligned} \tag{86}$$

where $A_i = \mathrm{A}(D_i W_i)$ are the individual design matrices to be trained. This is a so-called column-wise partitioned model of observation equations (Teunissen, 2000b). The least squares criterion leads to the following partitioned normal equations ($N\hat{X} = U$):

$$\begin{bmatrix} N_{11} & \cdots & N_{1p} \\ \vdots & \ddots & \vdots \\ N_{p1} & \cdots & N_{pp} \end{bmatrix} \begin{bmatrix} \hat{X}_1 \\ \vdots \\ \hat{X}_p \end{bmatrix} = \begin{bmatrix} U_1 \\ \vdots \\ U_p \end{bmatrix} \tag{87}$$

where $N_{ij} = A_i^T Q^{-1} A_j$ and $U_i = A_i^T Q^{-1} Y$. There are different ways to solve these equations. The straightforward manner is to solve for X simply from the inverse of N, i.e. $\hat{X} = N^{-1} U$. An alternative is to use the block-triangular decomposition of $N = LDL^T$, where

$$L = \begin{bmatrix} I_{n_1} & 0 & \cdots & 0 \\ L_{21} & I_{n_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ L_{p1} & L_{p2} & \cdots & I_{n_p} \end{bmatrix} \tag{88}$$

and

$$D = \mathrm{blkdiag}\left(D_{11},\ D_{11},\ \cdots,\ D_{pp}\right) \tag{89}$$

where L is a lower block-triangular unit matrix and D is a block-diagonal matrix. The entries of $L = L_{ij}$ and $D = D_{ii}$ are explained in Appendix C.

To solve for $\hat{X}$, rewriting $NX = LDL^T X = LZ = U$, we first solve for Z from $LZ = U$ as

$$\begin{bmatrix} I_{n_1} & 0 & \cdots & 0 \\ L_{21} & I_{n_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ L_{p1} & L_{p2} & \cdots & I_{n_p} \end{bmatrix} \begin{bmatrix} Z_1 \\ Z_2 \\ \vdots \\ Z_p \end{bmatrix} = \begin{bmatrix} U_1 \\ U_2 \\ \vdots \\ U_p \end{bmatrix} \tag{90}$$

which gives

$$Z_i = U_i - \sum_{j=1}^{i-1} L_{ij} Z_j,\ i = 1, \dots, p \tag{91}$$

We then need to use $DL^T \hat{X} = Z$, resulting in

$$\begin{bmatrix} D_{11} & D_{11} L_{21}^T & \cdots & D_{11} L_{p1}^T \\ 0 & D_{22} & \cdots & D_{22} L_{p2}^T \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & D_{pp} \end{bmatrix} \begin{bmatrix} \hat{X}_1 \\ \hat{X}_2 \\ \vdots \\ \hat{X}_p \end{bmatrix} = \begin{bmatrix} Z_1 \\ Z_2 \\ \vdots \\ Z_p \end{bmatrix} \tag{92}$$

which finally gives

$$\hat{X}_i = D_{ii}^{-1} \left(Z_i - \sum_{j=i+1}^{p} D_{ij} L_{ji}^T \hat{X}_j \right),\ i = p, \dots, 1 \tag{93}$$

The estimated $\hat{X}_i$, $i = 1, \dots, p$ can be used to update the individual weights (see below, with Eq. (95)).

We may also propose an alternative method that considers the individual $\hat{X}_i$, $i = 1, \dots, p$ in each iteration. Because the entire problem must be solved through an iterative manner, due to non-linearity of W_i's for example, this method estimates both $\hat{X}_i$ and W_i's in an iterative method (we first update $\hat{X}_i$ and then W_i). The i^{th} block row of the normal equations $N\hat{X} = U$ is $\sum_{j=1}^{p} N_{ij} \hat{X}_j = U_i$, which is reformulated to $N_{ii} \hat{X}_i + \sum_{j=1}^{p,\, j \neq i} N_{ij} \hat{X}_j = U_i$. This finally provides an update for $\hat{X}_i$ as

$$\hat{X}_i = N_{ii}^{-1} \left(U_i - \sum_{j=1}^{p,\, j \neq i} N_{ij} \hat{X}_j \right) \tag{94}$$

where i runs from 1 to p. It is known that the final estimates $\hat{X}_i$'s are to be calculated through an iterative method.

The latest estimates of $\hat{X}_i$'s (using either of the above methods), can then be employed to update the individual weights as

$$W_i^{(t+1)} = W_i^{(t)} + \alpha D_i^T (Q^{-1} \hat{E} \Sigma^{-1} \hat{X}_i^T \odot A_i') \tag{95}$$

where $\hat{E} = Y - \sum_{j=1}^{p} A_j \hat{X}_j$ can also be updated when an individual $\hat{X}_i$ is updated. As an alternative, $\hat{E}$ can be updated only once over each iteration t in which all $\hat{X}_j$ have been estimated.

7.3. Multilayer DL model

So far we considered the single-layer model as $Y = \mathrm{A}(DW)X + E$, to be as simple as possible. Our observations indicate that this model can indeed be applied to a series of problems having moderate complexity. If the above model is not suitable, we may consider to use a more sophisticated model. Considering more complexity, possibly due to complicated non-linearity, we consider a model that includes weights in deeper layers. The adjective 'deep' refers to the use of multiple layers in a deep learning network (Kawaguchi, 2016). As a starting point, we consider a double-layer deep learning model as

$$Y = \mathrm{A}_2\left(\mathrm{A}_1(DW_1)W_2\right)X + E, \tag{96}$$

where in addition to W_1 we have convolved a new weight matrix W_2, having generally a new activation function $\mathtt{A}_2(.)$. In this formulation, the unknown matrices are then X, W_1 and W_2. We first have to solve for X, then for W_2, and finally for W_1. Given $\hat{X} = (A^T Q^{-1} A)^{-1} A^T Q^{-1} Y$, with $A = A_2 = \mathtt{A}_2(.)$ it follows that $\hat{Y} = A\hat{X}$ and $\hat{E} = Y - \hat{Y}$. We then update the weight matrix W_2 as $W_2^{(t+1)} = W_2^{(t)} - \alpha_2 \nabla_{W_2} \phi$, with (cf. Eq. (26))

$$\nabla_{W_2} \phi = -A_1^T (Q^{-1} \hat{E} \Sigma^{-1} \hat{X}^T \odot A_2') \tag{97}$$

where α_2 is the W_2 learning rate and $A_1 = \mathtt{A}_1(DW_1)$ is a given $m \times k$ matrix. Having the W_2 updated, we may then update $\hat{X}$, $\hat{Y}$ and $\hat{E}$ again. The last step concerns updating W_1. This is to be done using the following equation:

$$W_1^{(t+1)} = W_1^{(t)} - \alpha_1 \nabla_{W_1} \phi(W_2^{(t+1)}, W_1^{(t)}) \tag{98}$$

where α_1 is the W_1 learning rate and $\phi(W_1, W_2) = \frac{1}{2}\mathrm{tr}(Y^T Q^{-1} Y) - \frac{1}{2}\mathrm{tr}(U^T N^{-1} U \Sigma^{-1})$ is given in Eq. (23). Its derivative with respect to W_1 is then needed, which can follow from the following theorem:

Theorem 3. *Let the objective functions be of the forms*

$$\phi_1 = \mathrm{tr}(B\mathtt{A}_2(\mathtt{A}_1(D W_1)W_2)C) = \mathrm{tr}(BA_2C) \tag{99}$$

and

$$\phi_2 = \mathrm{tr}\big(E(B\mathtt{A}_2(\mathtt{A}_1(D W_1)W_2))^{-1}C\big) = \mathrm{tr}\big(E(BA_2)^{-1}C\big) \tag{100}$$

where B, C, D, E, W_1 and W_2 are given matrices of appropriate size, W_1 contains running unknown variables, and $\mathtt{A}_1(.)$ and $\mathtt{A}_2(.)$, the activation functions, are assumed to be differentiable. Then their partial derivatives with respect to the matrix W_1 are

$$\frac{\partial \phi_1}{\partial W_1} = D^T\big((B^T C^T \odot A_2')W_2^T \odot A_1'\big) \tag{101}$$

and

$$\frac{\partial \phi_2}{\partial W_1} = -D^T\big((B^T H^T E^T C^T H^T \odot A_2')W_2^T \odot A_1'\big) \tag{102}$$

respectively, where $H = (BA_2)^{-1}$, $\odot$ is the element-wise Hadamard product applied to two matrices of the same dimension, $A_2 = \mathtt{A}_2(.)$ is the design matrix, and A_1' and A_2' are the first derivatives of A_1 and A_2, respectively.

Proof. See Appendix D.

After a few simple mathematical operations, the above theorem along with $N = A^T Q^{-1} A$ and $U = A^T Q^{-1} Y$ provide the gradient $\nabla_{W_1} \phi$ as

$$\nabla_{W_1} \phi = -D^T\big((Q^{-1} \hat{E} \Sigma^{-1} \hat{X}^T \odot A_2')W_2^T \odot A_1'\big) \tag{103}$$

where A_1' and A_2' are the derivatives of the activation functions $\mathtt{A}_1(.)$ and $\mathtt{A}_2(.)$, respectively.

We are now in a position to generalize the above formulation. Let us convolve the feature matrix D sequentially with a series of weight matrices $W_1, \ldots, W_p$ and using the activation functions $\mathtt{A}_1, \ldots, \mathtt{A}_p$. The model of observation equations $Y = AX + E$ reads then

$$Y = \mathtt{A}_p\Big(\mathtt{A}_{p-1}\big(\ldots \mathtt{A}_1(D W_1) \ldots W_{p-1}\big)W_p\Big)X + E, \tag{104}$$

where the final $m \times n$ design matrix $A = \mathtt{A}_p(.)$ is to be trained. The weight matrices W_j, should be estimated using the gradient descent method as

$$W_j^{(t+1)} = W_j^{(t)} - \alpha_j \nabla_{W_j} \phi(W_p^{(t+1)}, \ldots, W_{j+1}^{(t+1)}, W_j^{(t)}, \ldots, W_1^{(t)})$$

where j runs from p to 1. We first update W_p using the gradient ∇_{W_p}, with (cf. Eq. (26))

$$\nabla_{W_p} = -A_{p-1}^T (Q^{-1} \hat{E} \Sigma^{-1} \hat{X}^T \odot A_p') = -A_{p-1}^T H_p \tag{105}$$

where $A_p' = A' = \mathtt{A}_p'(A_{p-1}(.))$ is the derivative of A and $H_p = Q^{-1} \hat{E} \Sigma^{-1} \hat{X}^T \odot A_p'$. We then update W_p, $\hat{X}$, $\hat{Y}$ and $\hat{E}$. The gradient descent method is again used to update $W_{p-1}^{(t+1)} = W_{p-1}^{(t)} - \alpha_{p-1} \nabla_{W_{p-1}} \phi$, with (cf. Eq. (103))

$$\nabla_{W_{p-1}} \phi = -A_{p-2}^T\big(H_p W_p^T \odot A_{p-1}'\big) = -A_{p-2}^T H_{p-1} \tag{106}$$

where $H_{p-1} = H_p W_p^T \odot A_{p-1}'$. From the mathematical induction, we may further prove that the $\nabla_{W_q} \phi$ can be obtained as

$$\nabla_{W_q} \phi = -A_{q-1}^T\big(H_{q+1} W_{q+1}^T \odot A_q'\big) = -A_{q-1}^T H_q \tag{107}$$

where q runs from $p-1$ to 1 and

$$H_q = H_{q+1} W_{q+1}^T \odot A_q' \tag{108}$$

It is noted that $\nabla_{W_1} \phi$ requires A_0, which is indeed the feature matrix D. The above formulation is a generalized form of the back-propagation algorithm, widely used to train feed-forward artificial neural networks (Buscema, 1998). This generalization has all the properties of the LSBDL we highlighted in Section 6. Using the chain rule, the gradients were obtained with respect to the weight matrices W_i's and it iterates to modify the weights backward from the last (W_p) to the first (W_1) layer.

8. Opportunities and challenges of LSBDL

The general formulation of LSBDL has attractive properties and offers new application and research areas, which we highlight in this section. These properties will indeed render LSBDL transparent and interpretable, which align closely with the rapidly growing field of XAI.

8.1. Directions for further research

GN method: This contribution mainly focused to formulate LSBDL using the SD method. In Section 5, we also addressed the Gauss–Newton (GN) method, a widely used nonlinear least squares problem solver (Hartley, 1961; Ruhe, 1979; Teunissen, 1990; Gratton et al., 2007). The method requires the Jacobian matrix to be derived. The minimization problem in Eq. (17), as a nonlinear problem, was solved by GN, which is known to have a faster convergence rate compared to the SD method (Teunissen, 1990). GN can be used in small- to medium-sized ML problems.

In Section 9.1 we numerically showcase that GN has a higher convergence rate compared to SD (see later Fig. 2). Future studies should however aim to rigorously investigate the performance of GN in terms of convergence rate, computational burden, and scalability across different problem sizes. This exploration provides new opportunities and challenges for the GN formulation of LSBDL. For example, by comparing its efficiency in large-scale ML problems, we can gain a better understanding of its practical limitations and potential.

Deeper networks: The word 'deep' in neural networks refers to either multilayer dense neural networks or convolutional neural networks (CNN). Deep learning considers usually many hidden layers leading to lots of training parameters, and hence posing a few challenges (Chen and Lin, 2014). LSBDL can offer research directions of which we name three. (i) When there exist too many training samples (big data), DL is known to be computationally expensive (Elthakeb et al., 2018; Thompson et al., 2020; Matsubara et al., 2022). The mini-batch training, with a small number of training examples, is used iteratively to train the entire network on all groups of samples until they are all used (Li et al., 2014). LSBDL can handle this problem using recursive least squares. (ii) DL is known to be a powerful local feature extractor of which CNN is widely used (Albawi et al., 2017; Gu et al., 2018). We provided column-wise partitioning of the input matrix to handle this problem. (iii) So far, we mainly considered a single hidden layer. The word 'deep' can also refer to as multiple hidden layers rather than a single layer. The above

three topics were introduced in Section 7, but further development and applications can be the focus of future research.

Embedding of LSBDL into least-squares framework: We formulated LSBDL in the least-squares framework. This allows to directly apply the existing theory of the least squares method to deep learning problems. This is in conjunction with the transparent and explainable AI algorithms that help understanding the quality of DL output (Arrieta et al., 2020; Vilone and Longo, 2021; Chou et al., 2022). For example, quality control measures such as the covariance matrix and precision of predicted results can be determined by LSBDL. Also the available least-squares reliability theory and hypothesis testing, to identify mis-specification and outlying observations, can directly be established in the LSBDL framework (Baarda, 1968). Generalization of some of these theories was elaborated in Section 6, which offers new directions for future research.

Convergence speed and overfitting: There are still challenges to be addressed by LSBDL. The challenges originate from three closely related problems associated with deep learning: overfitting, local minimum and convergence speed. Some of the future research directions in LSBDL are itemized as follows: (1) *Batch normalization* is a widely adopted technique that normalizes activations in intermediate layers of deep learning to improve accuracy and speed up the training process (Ioffe and Szegedy, 2015; Santurkar et al., 2018; Bjorck et al., 2018). (2) *Dropout* is a technique to address the overfitting problem, where randomly selected neurons (along with their connections) are temporally ignored during the training process (Baldi and Sadowski, 2013; Srivastava et al., 2014; Garbin et al., 2020). (3) *Adam* optimization is one of the most popular momentum-based SD variants that modifies individual adaptive learning rates from estimates of the first and second moments of the gradients (Kingma and Ba, 2014; Zhang, 2018). These are some research directions to be addressed in future.

Physics-informed neural networks: DL methods are known to be opaque black-box methods, which prevent obtaining a realistic assessment of the quality of predicted results. Ongoing research aims at increasing transparency by addressing this DL's black-box problem (Arrieta et al., 2020). One aspect of transparency is to incorporate the physics laws into the DL models while training, referred to as physics-informed neural networks (PINNs) (Raissi et al., 2019; Karniadakis et al., 2021; Cuomo et al., 2022). PINN is a scientific machine learning (SciML) technique used to incorporate physics laws, like partial differential equations (PDEs), in the training (Lehmann et al., 2024; Thiyagalingam et al., 2022). A primary challenge for PINN is its generalization beyond the training domain, especially when complex physical problems are represented by PDEs. This issue arises for example in geophysical inversion problems, when the neural network must learn the general solution of PDEs like the elastic wave equation with varying parameters (Lehmann et al., 2023). Other physics laws include radiative transfer modeling (RTM) (Gege, 2004) and the linear dispersion relation (LDR) (Santos et al., 2022; Daly et al., 2022) in satellite-derived bathymetry methods. LSBDL can incorporate prior knowledge using soft and hard physics-based constraints into DL through a unified least-squares formulation (Amiri-Simkooei, 2019). As LSBDL can also incorporate the covariance matrix of observations within the training process, it explores the weighting of constraints in the PINN model using least squares variance component estimation (LS-VCE) (Teunissen and Amiri-Simkooei, 2008; Amiri-Simkooei, 2007).

Out-of-Distribution: Out-of-distribution (OOD) detection using deep learning has gained increasing attention in the field of AI (Cui and Wang, 2022; Berend et al., 2020). LSBDL offers a robust framework to handle OOD predictions by leveraging its foundation in the least squares theory. Here, we outline four key aspects of OOD detection and management that can be effectively handled by LSBDL. (i) One of the strengths of LSBDL is its ability to directly determine quality control measures such as the covariance matrix and the precision of predicted outcomes. For example, high uncertainty in predictions can serve as an indicator of OOD data. (ii) LSBDL can employ available reliability theory and hypothesis testing to identify and detect (and hence remove) model mis-specification and outlying observations (refer to Section 6.2 on DIA). This is particularly useful for OOD detection, as OOD data often manifest as outliers or anomalies in the input. (iii) The LSBDL framework exploits the covariance matrix of observations to train the network with inconsistent, heterogeneous, and statistically correlated data. This approach is particularly beneficial to handle OOD data because it allows for higher weights to be given to more precise data and less weight to less precise data, as indicated by the role of Q and Σ in Eq. (27). (iv) Similar to all AI methods, LSBDL can also be trained to be robust against OOD data. Techniques such as data augmentation, adversarial training (Bai et al., 2021), and regularization methods (dropout, weight decay, and batch normalization) can be easily implemented within the LSBDL framework.

8.2. Challenges due to nonlinearity of LSBDL

The nonlinearity of the LSBDL model, originating from the utilization of nonlinear activation functions and the depth of the layers used, poses several challenges that warrant further research. Some research directions in the field of LSBDL involve the following interrelated fourfold challenges:

Quality control measures: We have provided a few measures for quality control of the estimates in the DL models. For example, the covariance matrix of estimates and predictions has been derived under different scenarios (see Eqs. (61) to (66)). It is noted that this measure for precision description is based on a one-layer learning model. Further derivation of such uncertainty quantification measures needs to be developed for deeper networks. This also poses other challenges as a multi-layer deep network can be severely nonlinear and hence the estimates can be biased (Teunissen and Knickmeyer, 1988), the covariance matrices cannot be too much representative of the actual covariance matrix (Wang and Zhao, 2017; Amiri-Simkooei et al., 2016), and the convergence (rate) can severely be affected (Teunissen, 1990). The deeper the network is, the more severe these challenges will become. This offers new challenges to be addressed in the future.

Hypothesis testing: In Section 6.2, we introduce the application of statistical hypothesis testing to detect mis-specification and outliers in the LSBDL model (Baarda, 1968). The nonlinearity of the model will affect the distribution of its underlying test statistics. For example, the distribution of the w-test statistic in Eq. (12) is not normal, even if the original observables are normally distributed. Furthermore, under the normality assumption, the test statistics in Eq. (11) has a chi-square distribution. However, this cannot hold for nonlinear least squares problems. For LSBDL, the type of activation function and the depth of the layers used have a significant effect on the distributional assumptions. This opens new research challenges. For example, to reject the null hypothesis and hence to identify mis-specification, it is always safe to use the Chebyshev inequality, which is independent of the distribution and gives an upper bound for a specified level of significance α, see Casella and Berger (2021). An alternative is based on the use of Markov chain Monte Carlo (MCMC) methods (Gilks et al., 1995; Jones and Qin, 2022), a powerful tool to estimate features of probability distributions. MCMC comprises a class of algorithms for sampling from a probability distribution. By constructing a Markov chain one can obtain a sample of the desired distribution by recording states from the chain. The more steps that are included, the more closely the distribution of the sample matches the actual desired distribution. Various algorithms exist to construct chains; the Metropolis–Hastings algorithm is most popular (Metropolis et al., 1953; Wu et al., 2017).

Error in input D: Possible uncertainties in the input variables D have been ignored so far. A more realistic error propagation should also

take uncertainty of D into consideration. This can be implemented by the weighted total least squares methods for which uncertainties of both A (due to D) and Y can simultaneously be considered in the estimation process and its uncertainty quantification (Amiri-Simkooei et al., 2016). In one hand, the covariance matrices given in Eqs. (61)–(66) only include the uncertainty of observations, which are propagated to the parameters of interest through the linear(ized) model. On the other hand, approximation using DL methods is subject to approximation errors, which is an inherent characteristic of all approximation methods (Elbrächter et al., 2021; Lu et al., 2021). The above challenges will offer future research directions in this field.

Vanishing gradient: The DL models significantly suffer from the problem of vanishing gradient. The vanishing gradient is a challenge that occurs when training deep neural networks, especially those with many layers. It arises when the gradients of the loss function with respect to the parameters of the network become small as they are backpropagated through the layers, because the gradients of the loss function are multiplied by the derivatives of the activation functions as they are backpropagated (see Eq. (108)). These challenges offer new research directions in which the weight initialization (Glorot and Bengio, 2010), activation functions (He et al., 2015), batch normalization (Ioffe and Szegedy, 2015), and skip connections (He et al., 2016a) are among some possible methods to address these challenges. For example, the use of the Rectified Linear Unit (ReLU) activation function, having a derivative of one for positive inputs, helps prevent the vanishing gradient problem (He et al., 2015).

8.3. Directions for some applications

LSBDL versus ML methods: ML methods have been applied to different application fields. A review of those methods is beyond the scope of present contribution. Here we refer to random forest (RF), support vector regression (SVR) and multilayer perceptron (MLP), as widely used methods. The RF regression is a supervised ensemble-learning technique that generates thousands of decision trees, which act as regression functions on their own, and the final RF output is the average of outputs from all decision trees (Breiman, 2001; Ho, 1995). SVR is based on the support vector machine (SVM) whose goal is to model and then predict complex relationship between input and output through mapping the data into a high-dimensional feature space (Awad and Khanna, 2015; Smola and Schölkopf, 2004). MLP is a fully connected feed-forward artificial neural network that consists of an input layer, an output layer and at least one hidden layer. MLP can be used to approximate complex nonlinear functions (Rumelhart et al., 1986; Ezugwu et al., 2005; Kawaguchi, 2016).

In a recent work, the performance of the above three methods was investigated for downscaling the groundwater storage anomaly (GWSA) extracted from the gravity recovery and climate experiment (GRACE) mission (Sabzehee et al., 2023). The methods predicted GWSA as a function of hydro-climatic variables such as precipitation, evapotranspiration, land surface temperature and normalized difference vegetation index. RF outperformed the other two methods to downscale GWSA. Our findings show that LSBDL outperforms the RF method using the same dataset (see Section 9.3). This needs further investigation in similar application fields such as data fusion methods (Meng et al., 2020), remote sensing applications (Belgiu and Drăguţ, 2016), mapping of mineral prospectivity (Rodriguez-Galiano et al., 2015) and solar energy systems (Ahmad et al., 2018).

Geoscience applications: LSBDL particularly offers new opportunities in geoscience. The applications cover a variety of problems in which the ordinary linear model theory cannot directly be applied. Such applications range from current challenges in global navigation satellite systems (GNSS), the gravity recovery and climate experiment (GRACE), multi-sensor integrated unmanned ship navigation, atmospheric monitoring and forecasting, multi-frequency multi-beam echo-sounders (MF-MBES) to satellite-based Lidar, SAR and multi-spectral imagery. For some of these applications, we may specifically refer to GNSS multipath detection (Hsu, 2017; Suzuki and Amano, 2021), estimation of precipitable water vapor (Lee et al., 2019; Zhang and Yao, 2021; Razin and Voosoghi, 2022), downscaling of GRACE-derived terrestrial water storage anomalies (Jyolsna et al., 2021; Foroumandi et al., 2022; Sabzehee et al., 2023), soil moisture estimation (Kim and Lakshmi, 2018; Senyurek et al., 2020; Nabi et al., 2022), wind speed prediction (Chen et al., 2018; Ibrahim et al., 2021; Asgarimehr et al., 2022), applications in synthetic aperture radar (SAR) images (Schwegmann et al., 2016; Chen et al., 2019a; Fracastoro et al., 2021), total electron content forecasting (Chen et al., 2019c; Natras et al., 2022) and time series analysis (Torres et al., 2021; Gao et al., 2022; Shahvandi and Soja, 2022).

Inverse problems: Inverse problems, encountered in many scientific and engineering fields including geophysics, astronomy, remote sensing and acoustics, require estimating system parameters from observed (noisy) data (Tarantola, 2005). In geophysics, they involve deducing unknown subsurface properties from observed data, such as seismic waves, gravity, or electromagnetic fields (Muir, 2022). Traditional approaches often rely on linear approximations and regularization techniques to manage ill-posedness and ensure stable solutions. Deep learning techniques have been shown to be superior to the traditional methods in seismic tomography (Araya-Polo et al., 2018) and 3D elastic wave propagation (Lehmann et al., 2024a). This becomes even more appealing when using SciML with PINN and PDE (Lehmann et al., 2024a). Ongoing research demonstrates how neural networks can be trained to predict subsurface velocity models from seismic data, highlighting the potential of AI to improve the accuracy and efficiency of geophysical inversions.

LSBDL addresses the inverse problem by training a neural network to establish the design matrix that links input data (e.g. velocity models) to predicted outcomes (e.g. wavefields) through a linear least-squares framework. This linear foundation is enhanced by the nonlinear capabilities of deep learning, allowing the method to handle complex, real-world geophysical relationships more effectively than purely linear methods. LSBDL applies the well-established least-squares theory to geophysical inverse problems in a three-fold manner. (i) LSBDL leverages the covariance matrix of observations during training, which makes it well-suited to handle geophysical data, often being inconsistent and heterogeneous. (ii) LSBDL provides quality control measures like the covariance matrix and precision of predicted outcomes. (iii) LSBDL can embed prior knowledge (e.g. of PDEs) using soft and hard physics-based constraints into the model through a unified least-squares formulation (Amiri-Simkooei, 2019).

Aviation: LSBDL is expected to contribute to various aviation acoustic and operational applications. (i) The use of AI methods in aviation noise modeling is gaining momentum (Feng et al., 2023). ML and DL play a crucial role in this field, which develop more accurate and adaptable noise prediction models (Cha et al., 2023). By leveraging large datasets, including noise measurements, flight trajectories, and meteorological conditions, these models can optimize predictions of noise levels and their impact on communities (Wang and Ma, 2024). (ii) In aviation operations, ML methods, such as MLP (Alla et al., 2021) and RF (Dai, 2024), have also been employed to predict flight delays. Similar ML and DL methods have been employed to build explainable RF classifiers to predict flight anomalies (E-Silva and Murça, 2023), predict safety-critical landing metrics using supervised machine learning (Puranik et al., 2020), and use ML methods for aircraft maintenance (Karaoğlu et al., 2023; Helgo, 2023). LSBDL has the high potential to accomplish the above tasks, which offers new opportunities and challenges in aviation noise modeling and operations.

Time series analysis: Time series analysis is a hot research topic in various scientific and engineering domains, where the goal is to model

sequential data points over time. Within this field, prediction and anomaly detection are two prominent research topics. (i) ML methods such as SVM (Smola and Schölkopf, 2004), RNN methods like long short-term memory (LSTM) and echo state network (ESN), and random forest (RF) (Bagnall et al., 2017) have shown promising results in time series forecasting. For example, LSTM has been employed to accurately predict stock prices (Nelson et al., 2017) and air quality indices (Wang et al., 2021a). (ii) ML methods have also been used to anomaly detection in time series data (Li and Jung, 2023), which is crucial in various domains like ionospheric monitoring (Zhang et al., 2024), among others.

LSBDL offers opportunities for time series analysis, which covers both forecasting and anomaly detection. For time series prediction, LSBDL utilizes past observations as features to predict future values. Anomaly detection, essential for identifying unusual patterns or outliers, leverages LSBDL's least squares reliability theory and hypothesis testing to identify mis-specifications and outlying observations. This dual capability of LSBDL in time series forecasting and outlier detection is showcased in Section 9.2. LSBDL will thus offer new opportunities and challenges for time series analysis.

Data fusion using LSBDL: Data fusion of multiple sensors, having inconsistent, heterogeneous and statistically correlated uncertainties, is still a challenging problem in ML methods (Meng et al., 2020; Qiu et al., 2022). The challenges require effective processing methods that compensate the heterogeneity and non-stationary properties of a large amount of data obtained from multiple sensors (Salcedo-Sanz et al., 2020). For some ongoing research in data fusion we may refer to the role of uncertainty in decision making and trust in automated analytics (Stracuzzi et al., 2018), unresolved challenges in weather forecasts using multiple data sources with different uncertainties (Watson-Parris, 2021), and data fusion of multiple sensors used in autonomous vehicles (Kurzidem et al., 2020). Parts of uncertainty, heterogeneity and correlation of observations can be explained in the data covariance matrix. For example, high-precision observations bring a larger contribution in the training than less precise observations (see Eq. (34)). LSBDL allows to take into consideration the covariance matrix $Q_{\text{vec}(Y)} = \Sigma \otimes Q$ of observations in the training. Matrix Σ expresses (co)variances among different classes (output), whereas matrix Q expresses variances of observations from different samples/sensors. This will thus offer new applications when combining observations from different sensors in the training.

9. Three illustrative examples

The efficacy of the proposed LSBDL method is demonstrated through two basic examples and a real-world geoscience application. The results primarily focus on the SD method, while GN is also used to train the network in Example 1. To investigate the performance of the methods, we primarily use the root mean squared error (RMSE) metric. RMSE quantifies the average magnitude of prediction errors, with lower values indicating better model performance.

9.1. Surface fitting problem

Approximating a given surface or an unstructured point cloud by parametric methods has been widely investigated in scientific and engineering problems (Pottmann and Leopoldseder, 2003). In parametric methods, an unknown function $z = f(u, v)$ is approximated with a series of given data points or samples specified as $z_i = y_i,\ i = 1, \dots, m$. The surface is usually approximated using a summation as

$$f(u, v) = \sum_{i=1}^{p} B_i(u, v) x_i \tag{109}$$

where x_i's are the unknown coefficients and the basis functions $B_i(u, v), i = 1, \dots, p$ are usually polynomial, piecewise polynomial or piecewise rational. Such basis functions include at least pure splines (Amiri-Simkooei et al., 2018), B-splines (Wang et al., 2006), Bézier basis splines (Bercovier and Jacobi, 1994), radial basis functions (Carr et al., 2001), PHT-splines (Wang et al., 2011), bivariate splines (Zeilfelder, 2002), and triangular B-splines (He and Qin, 2004).

The basis functions $B_i(u, v)$ in Eq. (109) are indeed elements of the design matrix A. This indicates that the basis functions are known a priori, and they can in principle be linear/nonlinear functions of the variables u and v. This however indicates that the entries of A are known a priori. Although the final approximating surface can generally be a nonlinear function (possibly due to nonlinearity of the basis functions), we use the linear least squares theory to estimate its coefficients.

Known function: We follow a similar method using the non-parametric method LSBDL in which the linear least squares theory is used. The main difference is that the entries of A (basis functions) are not known a priori, and therefore they should be trained in an iterative procedure. In principle, function values $z = f(.)$, measured as a function of positions u and v, known as a point cloud, are approximated by a 2D surface using LSBDL. The efficiency of the method is investigated through a numerical example, which starts from a known mathematical function. For simulation of a 2D surface, consider the following mathematical function

$$f(u, v) = ue^{-(u^2+v^2)} \tag{110}$$

where $e^{(.)}$ is the exponential function.

Data generation: Fig. 1 [left] shows this function on a regular grid of $41 \times 41 = 1681$ points on $u = -2 : 0.1 : 2$ and $v = -2 : 0.1 : 2$. A data set of $m = 500$ points, sampled from Eq. (110) and generated using the uniform distribution over the region is used to illustrate the efficacy of the proposed method (Fig. 1 [right]). Random noise with a standard deviation of 0.01 is added to the generated data set. Three observations, namely observations 50, 250 and 350, are intentionally contaminated with outliers of −0.1, −0.1 and 0.1, respectively. We therefore have $y = [z_1, \dots, z_m]^T$, where z_i is measured at coordinates $(u_i, v_i),\ i = 1, \dots, m$. The goal is to approximate/regenerate this data set using the LSBDL theory proposed in Sections 4 and 5. We will assume $Q = I_m$, an identity matrix of size m.

Two sets of features: To train a design matrix, we use the features $[1, u, v]$ (Case 1) and $[1, u, v, uv, u^2, v^2]$ (Case 2). Case 1 is indeed the simplest case of which the features are just the point coordinates and a bias. The i^{th} row of the feature matrix D is then $D_i = [1, u_i, v_i]$ or $D_i = [1, u_i, v_i, u_i v_i, u_i^2, v_i^2]$. The convolution of DW is then activated by the sigmoid function to make the design matrix $A = \text{A}(DW)$. As indicated before, we use $m = 500$ data points to train the network. As a test data set, we again use the regular grid of $41 \times 41 = 1681$ points, to see how we can reproduce the known function $f(u, v) = ue^{-(u^2+v^2)}$ in Fig. 1 [left]. To train the design matrix, we set the hyper-parameters to $\alpha = 0.01$, $\kappa = 10^{-6}$, $\mu = 0.9$ and $s = 0.5$. The weight matrix W is of size $3 \times n$ and $6 \times n$, for Case 1 and Case 2, respectively, with their initial entries generated randomly using the uniform distribution between −1 and 1. To investigate the effect of under-parametrization on the results, different values of n are suggested. This value is set to $n = 5$, 10 and 20. We set $Q_y = \sigma^2 I_m$, with $\sigma = 0.01$ (hence $\Sigma = \sigma^2$ and $Q = I_m$).

Training using SD and GN: For Case 1 ($k = 3$), with $n = 20$, the training is performed to see its performance using the SD and GN

Table 2
Overall model test (OMT) on test statistics $\hat{s}^2 = T/(m-n) = \hat{e}^T Q_y^{-1} \hat{e}/(m-n)$ for Cases 1 and 2 (TS1 and TS2) versus critical values (CV) under three scenarios with $n = 5$, 10, and 20; observations 50, 250 and 350 contain blunders.

n	CV	$\hat{s}^2$ = TS1	Result	$\hat{s}^2$ = TS2	Result
5	1.154	17.34	reject	1.675	reject
10	1.155	4.39	reject	1.646	reject
20	1.156	1.77	reject	1.618	reject

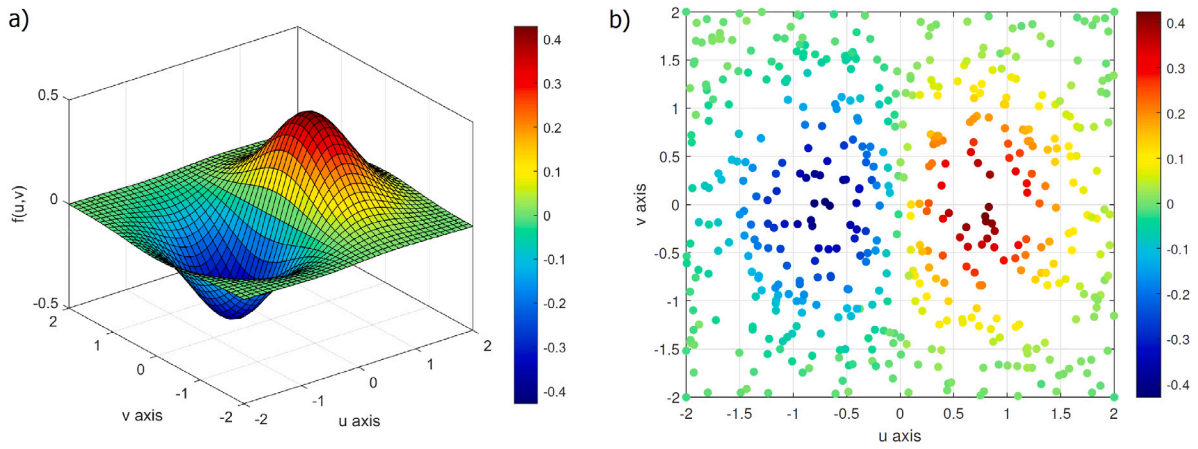


Fig. 1. Mathematical function $f(u,v) = ue^{-(u^2+v^2)}$ shown on a rectangular regular grid of $u = -2 : 0.1 : 2$ and $v = -2 : 0.1 : 2$ [a]. 500 data points simulated randomly using uniform distribution for the mathematical function $f(u,v)$ scattered irregularly over region $-2 \le u \le 2$ and $-2 \le v \le 2$ [b].

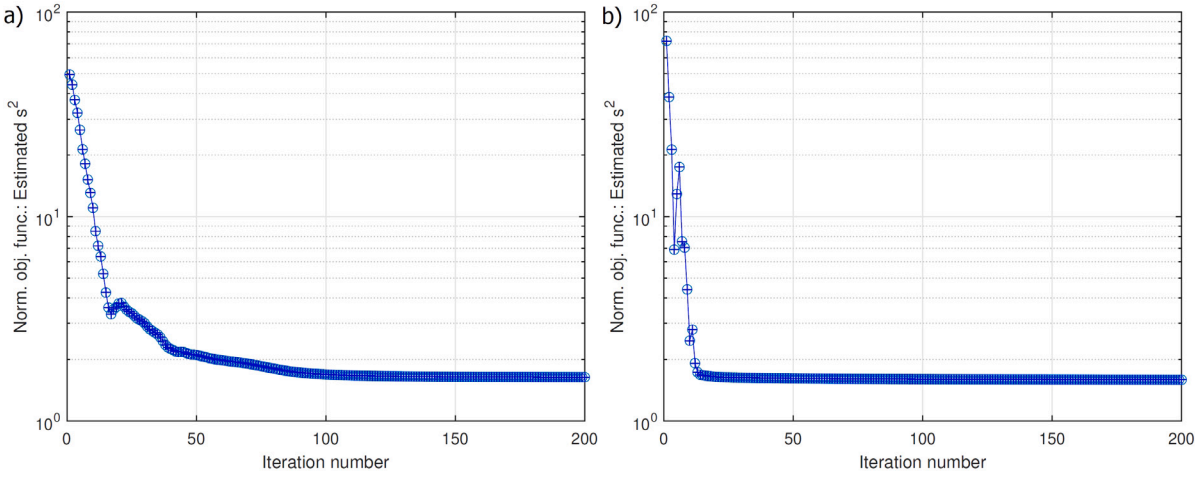


Fig. 2. Normalized objective function, estimated variance factor $\hat{s}^2 = \hat{e}^T Q_y^{-1} \hat{e}/(m-n)$, over different iterations for Case 1, with $n = 20$; SD method (a) and GN method (b).

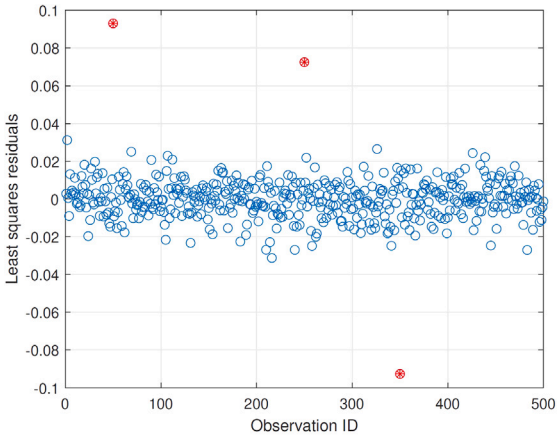


Fig. 3. Least squares residuals of the trained linear model $y = Ax + e$. Indicated are outlying observations # 50, 250, and 350 (red circles).

methods. Fig. 2 shows the reduction of the (normalized) objective function in different iterations, which is indeed the estimated variance factor $\hat{s}^2 = \hat{e}^T Q_y^{-1} \hat{e}/(m-n)$ and is expected to be one. The normalized values make the comparison easier when dealing with different values of $n = 5,\ 10$, and 20. A clear reduction is observed in both sub-frames, indicating that the network is learning through iterations of both SD and GN. Although SD shows a steady reduction through iterations, GN converges quickly, which is known to have a faster convergence rate (less iterations) compared to SD (Teunissen, 1990); here the GN normal matrix $J_w^T Q_{\mathrm{vec}(Y)}^{-1} J_w$ is of size $(kn = 60) \times (kn = 60)$, but the computational burden can drastically increase for larger values of k and n. For both methods, the estimated variance factor does not reach the preassigned value of 1, which can have two reasons: (1) The model is under-parametrized indicating that the number of unknowns ($n = 20$), or the number of features $k = 3$ are not sufficient to fully capture the non-linearity of $f(u,v)$. (2) The three outliers introduced do not allow the estimated variance factor $\hat{s}^2$ to become smaller. We further elaborate on this.

Detection in DIA: We now apply the hypothesis testing using OMT, referred to as 'detection' in the DIA procedure in Section 6.2. For Cases 1 and 2, the normalized objective functions of models, for $n = 5,\ 10$ and 20, are presented in Table 2. In general, introducing more parameters, either by having larger n's or by having more features (Case 2 versus Case 1) reduces the estimated variances. In the 'detection' step, an OMT $\hat{s}^2 = T/(m-n) = \hat{e}^T Q_y^{-1} \hat{e}/(m-n)$ has been performed to test the validity of the model and observations at a confidence level of 99%. All cases indicate that the null hypothesis is rejected, expressing that

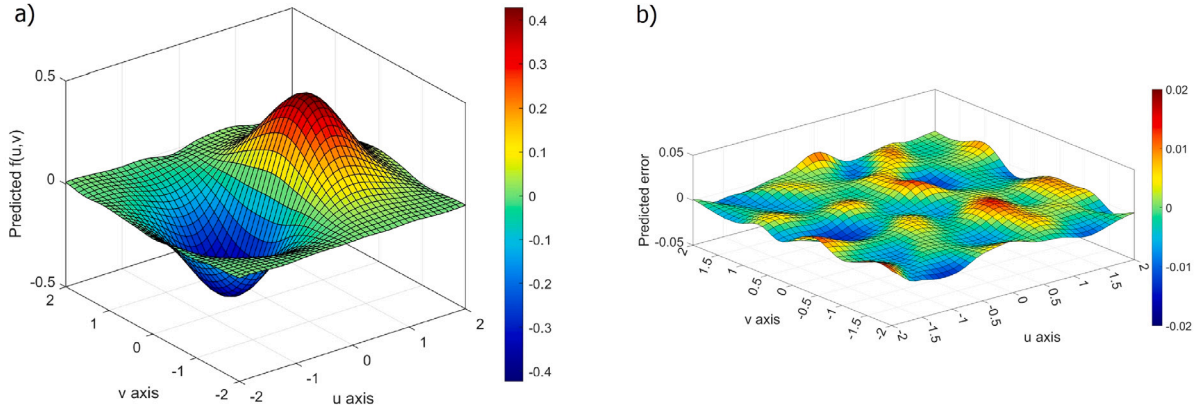


Fig. 4. Function values predicted on a regular grid of $41 \times 41 = 1681$ points on $u = -2 : 0.1 : 2$ and $v = -2 : 0.1 : 2$ [a]. Predicted errors, difference between predicted and true values [b].

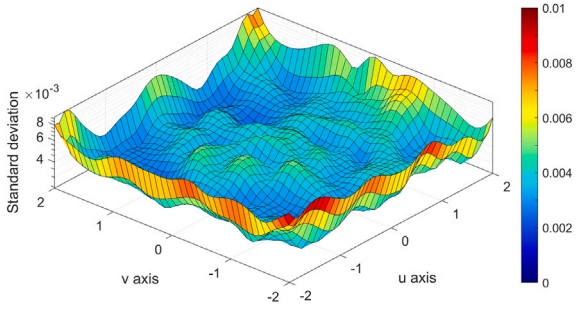


Fig. 5. Standard deviation of predicted function values on a regular grid of $41 \times 41 = 1681$ points on $u = -2 : 0.1 : 2$ and $v = -2 : 0.1 : 2$.

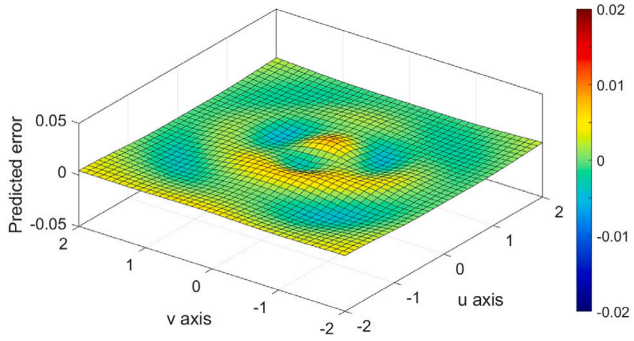


Fig. 6. Prediction error (difference between the true and predicted values) for the double-layer model.

the observations do not fit the model. This is mainly due to the under-parametrization of the model or the introduced outlying observations, which are identified in the identification step.

Identification in DIA: We now apply the hypothesis testing using the w-test statistic, referred to as 'identification' in the DIA procedure (see Section 6.2). Case 2 (and $n = 20$) is used to identify the outlying observations. The least squares residuals, presented in Fig. 3, clearly show there are three outlying observations. The w-test statistics in Eq. (12) are as follows: $w_{50} = 10.07$, $w_{250} = 8.22$, and $w_{350} = -10.39$, which are all rejected at the 99% confidence level of the standard normal distribution, with the critical value of ± 2.58. Therefore, these observations should be excluded from the dataset.

Adaptation in DIA: When outliers are confidently identified, they should be compensated for in the functional model (here we just remove them). When the original observations are used (those without outliers), the OMT will accept the null hypothesis (see Table 3), which includes Case 1, with $n = 20$ and Case 2 with $n = 5,\ 10$, and 20. This indicates that the observations fit the model well, and hence no under-parametrization occurs in these cases. This is further confirmed by the w-test values, which are $w_{50} = 1.23$, $w_{250} = -0.84$, and $w_{350} = -2.01$ and they are all accepted at the above confidence level. For Case 1, with $n = 5$ and $n = 10$, the OMT is still rejected, indicating under-parametrization due to an insufficient number of neurons in the hidden layer (small n's).

Table 3
Overall model test (OMT) on test statistics $\hat{s}^2 = T/(m-n) = \hat{e}^T Q_y^{-1} \hat{e}/(m-n)$ for Cases 1 and 2 (TS1 and TS2) versus critical values (CV) under three scenarios with $n = 5,\ 10$, and 20; outlying observations 50, 250 and 350 were removed.

n	CV	$\hat{s}^2$ = TS1	Result	$\hat{s}^2$ = TS2	Result
5	1.154	16.340	reject	1.034	**accept**
10	1.155	1.723	reject	1.002	**accept**
20	1.156	1.070	**accept**	0.984	**accept**

Prediction: Having the weight matrix W and the unknown vector x estimated, the function values can be predicted on a regular grid of $41 \times 41 = 1681$ points on $u = -2 : 0.1 : 2$ and $v = -2 : 0.1 : 2$ (the same grid as in Fig. 1 [a]). The feature matrix D_t and hence the design matrix $A_t = \mathrm{A}(D_t W)$ can be used to predict the function values. The results are shown in Fig. 4 [a]. Their predicted errors, differences with those provided in Fig. 1 [b] as the true values, are presented in Fig. 4 [bottom]. Their root mean square error (RMSE) is 0.005. This is partly due to the uncertainties in $\hat{x}$ (resulted from observation errors $\sigma = 0.01$) and partly due to mismodeling of the function $f(u, v)$.

Uncertainty of prediction: DL methods typically require test data to evaluate the performance of the trained model. LSBDL, however, goes further by providing additional measures, such as the covariance matrix of predictions, which results in the prediction uncertainty. The uncertainties in observations Y contribute to the uncertainty of the estimates $\hat{x}$ and $\hat{W}$ and hence to the prediction results. We use Eq. (74) to estimate the standard deviation of the predicted function values shown in Fig. 4 [a]; the results are presented in Fig. 5. The average standard deviation is around 4×10^{-3}, which closely aligns with the previously mentioned RMSE of 5×10^{-3}. The slight difference can be attributed to the approximation errors, an inherent characteristic of all approximation methods (Elbrächter et al., 2021; Lu et al., 2021), and a challenge that future research in this field will need to address.

Double-layer model: In addition to our primary focus on a single-layer network, we also investigated the performance of a double-layer network model using Eq. (96). The hyper-parameters were set to $\alpha = 0.01$,

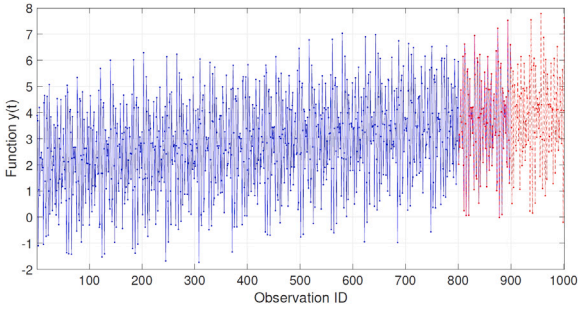


Fig. 7. Time series data set contaminated with normally distributed noise with $\sigma = 0.01$. Solid blue line shows training data and dashed red line shows testing data.

$\kappa = 10^{-6}$, $\mu = 0.9$ and $s = 0.5$. For the single-layer model, the number of unknown weights was 60, calculated as 6 input neurons multiplied by 10 hidden neurons. In contrast, the double-layer model had only 24 unknown weights, with 6 input neurons connecting to 2 hidden neurons in the first layer (W_1 of size 6×2), and 2 hidden neurons connecting to 6 hidden neurons in the second layer (W_2 of size 2×6). Although this reduction in the number of parameters can potentially lead to under-parametrization and a worse fit, the added depth of the network resulted in a significant improvement in performance. When evaluated over 10 independent runs, the RMSE of the double-layer model decreased from 4.6×10^{-3} (observed in the single-layer model) to 2.5×10^{-3}. Fig. 6 illustrates the prediction error for the double-layer model (cf. Fig. 4 [b]). These findings follow the principle that exponential functions, such as the one given in Eq. (110), introduce a high degree of non-linearity (Strogatz, 2018). Our results indicate the potential advantages of employing deeper network architectures, which highlights the enhanced capability of the double-layer model to capture and learn complex patterns within the data more effectively than the single-layer model.

9.2. Time series forecasting

As an extrapolation problem, time series prediction is considered to be a challenging type of predictive problem. The complexity originates from the sequence dependence among the input variables. In general, parametric and non-parametric methods are commonly used in time series analysis. The parametric methods include dynamic time series prediction methods in which the functional and stochastic effects can be captured in the best linear unbiased prediction (BLUP) framework (Teunissen, 2001). They assume that the underlying stationary stochastic process has a certain structure such as an auto-regression or a moving average with a small number of parameters. The task is then to estimate the parameters of the model describing its stochastic process.

By contrast, a nonparametric method is a mathematical tool that does not make any assumptions on the underlying model and hence on its corresponding parameters. As a data-driven method, the underlying model is determined/trained without assuming any structure on its particular application. The non-parametric methods used are singular spectrum analysis (SSA) (Elsner and Tsonis, 2013), spectral and wavelet analysis (Torrence and Compo, 1998), and recurrent neural networks (RNN) such as the long-short term memory (LSTM) method (Hochreiter and Schmidhuber, 1997).

Time series simulation: The LSBDL method is used to forecast events in time series. Let us simulate a time series based on the following model:

$$\begin{aligned} y(t) &= y_0 + r\,t + \textstyle\sum_{i=1}^{3} \sin(\omega_i t + \phi_i) \\ &+ \big(1 + 2\sin(\omega_4 t)\big)\sin(\omega_5 t) \end{aligned} \tag{111}$$

where $y_0 = 2$ is the intercept, $r = 0.02$ is the rate, $\omega_1 = 10$, $\omega_2 = 13$, $\omega_3 = 16$, $\omega_4 = 6$ and $\omega_5 = 20$ are the frequencies and $\phi_1 = 1$, $\phi_2 = 2$ and $\phi_3 = 3$ are the phases. The time series is simulated over a time span of $T = [0, 100]$, with an interval of 0.1, so in total there are $m = 1001$ observations. The observations are contaminated with normally distributed random noise with a standard deviation of $\sigma = 0.01$. Three observations, namely observations 100, 400 and 700, are intentionally contaminated with outliers of 0.1, −0.1 and −0.1, respectively. Fig. 7 shows a typical example of the simulated data set.

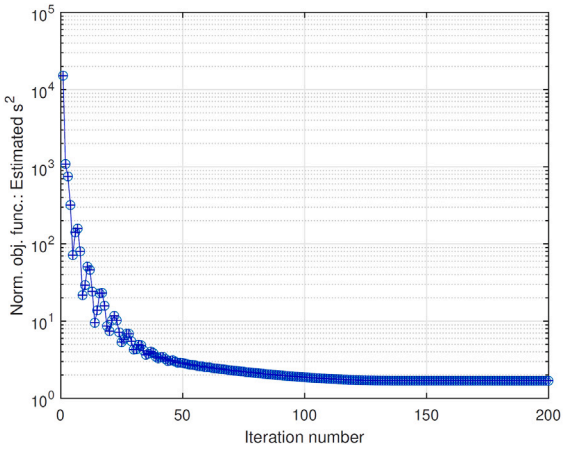


Fig. 8. Normalized objective function, estimated variance factor $\hat{s}^2 = \hat{e}^T Q_y^{-1} \hat{e}/(m-n)$, over different iterations of training time series data (SD method).

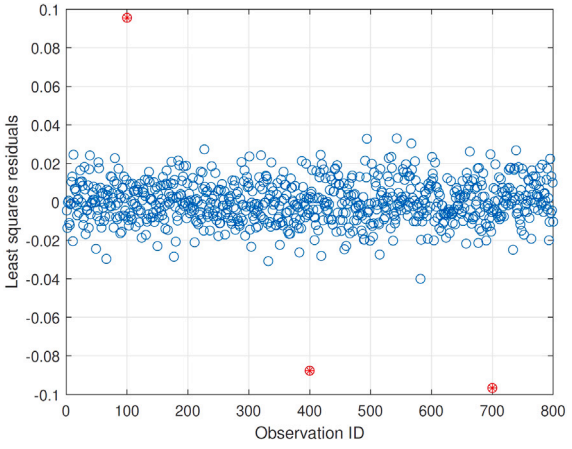


Fig. 9. Least squares residuals of the trained linear model $y = \mathbb{A}(DW)x = Ax + e$ of time series data. Indicated are outliers in observations # 100, 400, and 700 (red circles).

Input and output: We aim to predict future events merely based on historical data, so without using Eq. (111). Given the time series values at $T = [t_1, t_2, \ldots, t_p]$ a common exercise in time series analysis is to forecast q future values at $[t_{p+1}, \ldots, t_{p+q}]$ based on the historical data (Elsworth and Güttel, 2020). We set $p = 100$ and use $q = 1$ in our forecasting. The above simulated data are divided into training and test data sets (observations y_1 to y_{900} for training and observations y_{801} to y_{1001} for test). For the training data set, feature matrix D and observations y are of the form

$$D = \begin{bmatrix} y_1 & \cdots & y_{100} \\ y_2 & \cdots & y_{101} \\ \vdots & \ddots & \vdots \\ y_{800} & \cdots & y_{899} \end{bmatrix}, \quad y = \begin{bmatrix} y_{101} \\ y_{102} \\ \vdots \\ y_{900} \end{bmatrix} \tag{112}$$

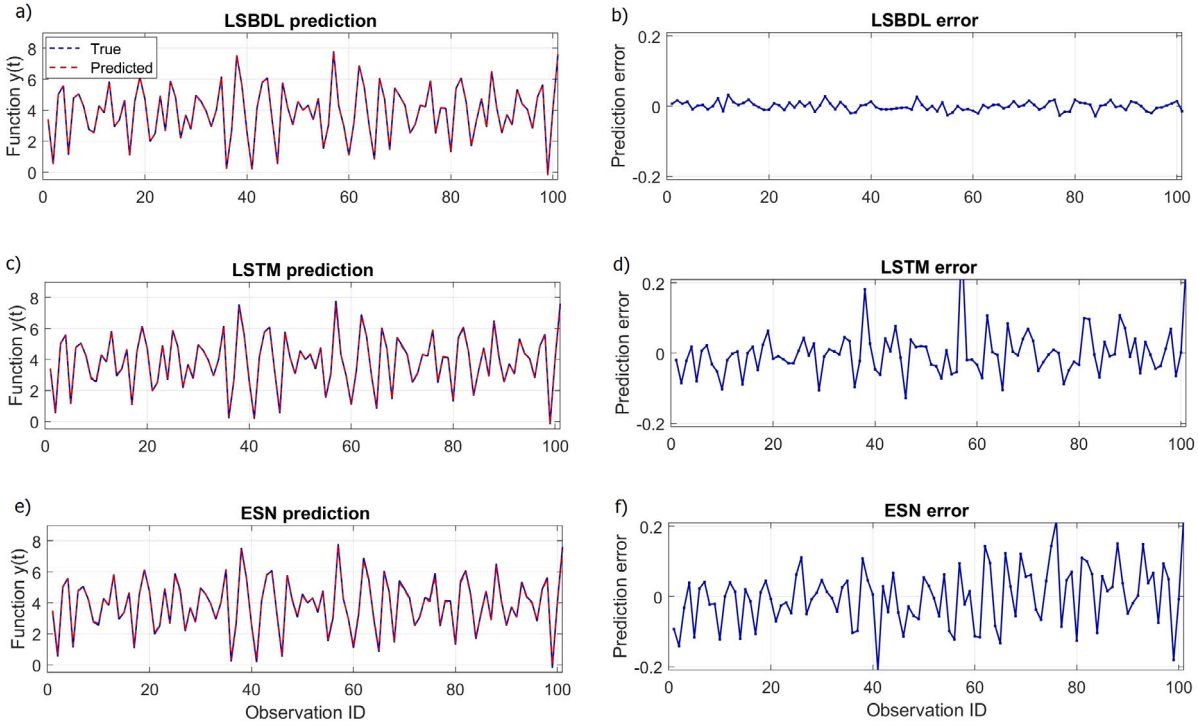


Fig. 10. Time series of original test data (dashed blue line) versus and predicted data (dashed red line) [left]. Difference between the true and predicted values (prediction error) [right]; LSBDL (top), LSTM (middle), and ESN (bottom).

In a similar manner, D_t and y_t can be constructed from observations y_{801} to y_{1001} for the test data set.

Establishing model: The feature matrix is augmented to include the bias term as $D \leftarrow [D : u]$, where u is the summation vector containing all ones. We then have $k = 101$. As indicated before, we use the $m = 800$ data points to train the network. The convolution of DW is activated using a ReLU function to train the design matrix $A = \mathrm{A}(D\ W)$. The weight matrix W is of size $101 \times (n = 5)$ with their initial entries generated randomly using the uniform distribution between -1 and 1. We set $Q_y = \sigma^2 I_m$, where $\sigma = 0.01$. The observations (including the three outliers) are used to train the network. The hyper-parameters are set to $\alpha = 0.001$, $\kappa = 10^{-6}$, $\mu = 0.9$ and $s = 0.5$.

Application of DIA: Fig. 8 shows the reduction of the (normalized) objective function in different training iterations, which is indeed the estimated variance factor $s^2 = \hat{e}^T Q_y^{-1} \hat{e}/(m-n)$; it is mathematically expected to be one, and therefore we can apply the DIA procedure in the hypothesis testing. In the presence of the outliers, an OMT has also been performed to test the validity of the model and observations at a confidence level of 99%. The test statistic is $T/(m-n) = 1.68$, which is larger than the critical value of $\chi^2(m-n, 0.01)/(m-n) = 1.12$, indicating that the null hypothesis is rejected; the observations do not fit the model ('detection' step). This is due to the presence of the three outliers. In the 'identification' step the three outliers can be detected. Fig. 9 shows the least squares residuals. The outliers at the positions 100, 400 and 700 can clearly be marked and hence removed by the w-test statistic. We may then remove the outlying observations and repeat the analysis in the 'adaptation' step.

Prediction: Having the weight matrix W and the unknown vector x estimated, the trained network is finally used to predict future events. Based on the test data y_{801} to y_{1001} we obtain D_t and y_t and then predict $\hat{y}_t = A_t \hat{x} = \mathrm{A}(D_t W)\hat{x}$. The results are provided in Fig. 10 [a]. Their predicted errors ($\hat{e}_p = y_t - \hat{y}_t$), differences with those provided in Fig. 7 as the true values, are presented in the same plot [b]. Their root mean squares error (RMSE) is 0.014. This is partly due to the observation errors ($\sigma = 0.01$) and partly due to mismodeling of the trained network.

Architecture of ESN and LSTM: We compare the LSBDL results with those obtained by two types of recurrent neural network (RNN), namely an echo state network (ESN) (Lukosevicius and Jaeger, 2009; Viehweg et al., 2023) and long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997; Greff et al., 2016), using the same training and test data. Both methods were tuned for their optimal architectures and hyper-parameters. (i) In the ESN, a reservoir of randomly initialized neurons captures the dynamics of the input time series, while only the readout weights are trained. Key parameters include the input size of 100 (number of past time steps used for prediction), so identical to that of LSBDL. The reservoir size, which influences the network's capacity to model complex patterns, was set to 8000 neurons. Sparsity, controlling the density of connections within the reservoir, was set at 0.1 to balance computational efficiency with modeling capacity. The spectral radius of the reservoir weight matrix was adjusted to ensure the echo state property, usually kept below 1, with a common value around 0.95 to maintain stability (Jaeger, 2001). Regularization, applied during the training of readout weights, was set at a small value of 10^{-8}. (ii) The LSTM architecture was designed with an input size corresponding to the number of past samples, and a hidden layer consisting of 100 neurons to capture the temporal dependencies in the data. The network includes a sequence input layer, followed by an LSTM layer, and a fully connected layer to map the hidden states to the output. The final regression layer was used to facilitate continuous value prediction. For training, we employed the Adam optimizer with a maximum of 200 epochs and 1200 iterations. The training options were configured with an initial learning rate of 0.001, and a piecewise learning rate schedule. The learning rate was set to decrease by a factor of 0.2 every 125 epochs to ensure stable convergence. The network was trained on the training data and subsequently used to make predictions on the test data.

Comparison of metrics: The performance evaluation of the three methods, LSBDL, LSTM and ESN, demonstrates a clear superiority of LSBDL in both accuracy and computational efficiency. (i) The RMSE for LSBDL

is 0.014, significantly lower than the RMSEs for LSTM (0.066) and ESN (0.081). Given the range of the original time series (∼0 to 8), all methods provide acceptable performance; however, LSBDL outperforms the other two methods by a substantial margin in terms of precision (Fig. 10). (ii) We also evaluated the computational burden of the three methods on a computer with an Intel Core i7-10610U CPU@1.80 GHz, 2.3 GHz, 4 cores, 16 GB RAM. The computational time required for LSBDL is remarkably low at 0.20 s, compared to 60.3 s for LSTM and 34.4 s for ESN. This highlights LSBDL's efficiency and effectiveness, which makes it also an appropriate alternative for time series analysis applications.

9.3. Downscaling of GRACE data

The third example demonstrates the application of LSBDL to a real-world geoscience problem. The application of AI methods for downscaling the groundwater storage anomaly (GWSA) is an important research topic, particularly in regions with complex hydrological and climatic dynamics (Shokri et al., 2018; Chen et al., 2019b). Many basins, suffering from significant water loss due to a combination of climatic factors and human activities, demand high-resolution data for effective water resource management. The launch of the gravity recovery and climate experiment (GRACE) satellite mission in 2002 provided important large-scale groundwater data, but its coarse resolution limits its utility for smaller basins. To overcome this, downscaling methods, especially those based on AI techniques, have been used.

To downscale the GWSA to higher resolutions, AI techniques, including SVR, MLP, and RF, have been employed to enhance the spatial resolution of GRACE data. Among them, RF stands out as the most effective method for GWSA downscaling. RF is a supervised learning algorithm that uses an ensemble learning method to improve prediction accuracy. It can handle large datasets with numerous input variables without overfitting and can effectively identify the importance and interactions of variables (Breiman, 2001).

Previous studies have demonstrated the superiority of RF over SVR and MLP in various hydrological applications. For example, research has been conducted to successfully apply RF to downscale GRACE data over the Indus basin, and hence to achieve higher resolution (Ali et al., 2021). RF has also been used to improve the spatial resolution of groundwater storage data for the Northern High Plains aquifer (Rahaman et al., 2019). These examples underscore RF's capability to capture complex nonlinear relationships inherent in hydrological processes, which makes it a reliable tool for downscaling GWSA. In a recent work, we also investigated the performance of the above three methods for downscaling the GWSA extracted from GRACE (Sabzehee et al., 2023). The methods, tuned for their optimal performance, predicted GWSA as a function of hydro-climatic variables such as precipitation, evapotranspiration, land surface temperature, normalized difference vegetation index (NDVI), time and geographical coordinates. RF was shown to have the best performance over SVR and MLP.

To evaluate and select the optimal model, the performance of RF, SVR and MLP was assessed in a recent publication using three distinct metrics (Sabzehee et al., 2023): RMSE (already introduced), Pearson's correlation coefficient (R), and Nash–Sutcliffe efficiency (NSE). R measures the linear relationship between predicted and observed values, with values ranging from −1 to 1. Values closer to 1 indicate strong positive correlations, which suggests that the model predictions align well with the actual observations. NSE assesses the predictive power of the models, with values ranging from $-\infty$ to 1 (Nash and Sutcliffe, 1970). An NSE value of 1 corresponds to a perfect match between predicted and observed data, while values less than 0 indicate that the model performs worse than the mean of the observed data.

We investigate the performance of LSBDL using the same metrics. The hyper-parameters were set to $n = 500$, $\kappa = 10^{-6}$, $\alpha = 5 \times 10^{-5}$, $\mu = 0.95$, and $s = 0.5$. We used a dataset consisting of $m = 31640$ samples from Sabzehee et al. (2023). To investigate the performance of the methods, we split the data into training and test sets, with 70% of the samples used for training and 30% for testing. For each run, the training samples were selected randomly to ensure variability and robustness in the evaluation. To make the results reliable, we employed a cross-validation approach by carrying out the procedure on 10 independent runs. The performance metrics were computed based on these 10 runs to provide an assessment of each method's performance. The computational burden of the models was evaluated on a computer with an Intel Core i7-10610U CPU@1.80 GHz, 2.3 GHz, 4 cores, 16 GB RAM.

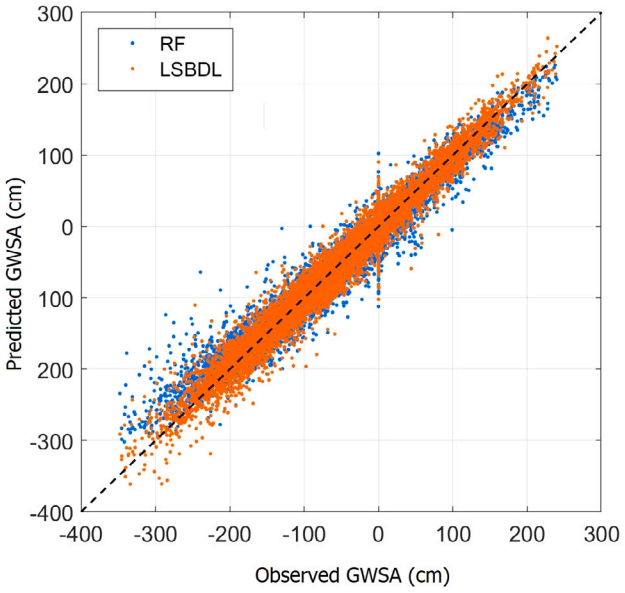


Fig. 11. Predicted versus observed GWSA to investigate the performance of RF (blue) and LSBDL (red) on the test data.

The results are presented in Table 4. They indicate that LSBDL significantly outperforms the SVR, MLP, and RF models as it achieves higher Pearson correlation coefficients (R), larger NSE values, and lower RMSE values. This is also graphically demonstrated in Fig. 11 from a typical run, which compares LSBDL with the competing method RF. However, LSBDL is not the fastest method. SVR, while being the fastest, has the worst performance in terms of accuracy metrics. MLP offers a balance between accuracy and running time, performing better than SVR but worse than RF and LSBDL in terms of accuracy, with a moderate running time. RF, despite its longer running time, has good performance in accuracy metrics, being second best in RMSE, R, and NSE.

Prior to the above analysis we took a step to pre-analyze the data for detecting possible outliers. This is performed using the identification step in the DIA procedure, which applies hypothesis testing using the w-test statistic (see Section 6.2). We used all data (so both training and test data together) to train the LSBDL using the set-ups explained earlier. To check the consistency of the results, we run the method twice (two independent runs, each run consists of 10 sub-runs). The w-test statistic values were then computed for these two runs. Fig. 12 shows the results, which highlight two key observations: (i) The results from the two independent runs demonstrate a high degree of consistency. This indicates that the LSBDL model produces stable and reliable results when applied multiple times under the same conditions. (ii) The figure indicates that some observations are rejected at the 99% confidence level of the standard normal distribution (corresponding to critical values of ±2.58). These outliers were excluded from the dataset in the above analysis to enhance the accuracy of the final results.

It is also noted that we applied the classical DIA theory (Teunissen, 2000a). Future work can also consider an extended DIA method, which

Table 4
Metrics RMSE, R, and NSE used to evaluate the performance of SVR, MLP, and RF. The table also includes the running time of each method.

Metric	SVR	MLP	RF	LSBDL
RMSE	35.76	24.74	20.38	18.15
R	0.938	0.969	0.980	0.984
NSE	0.874	0.939	0.959	0.967
Running time (s)	22.5	46.2	85.5	37.7

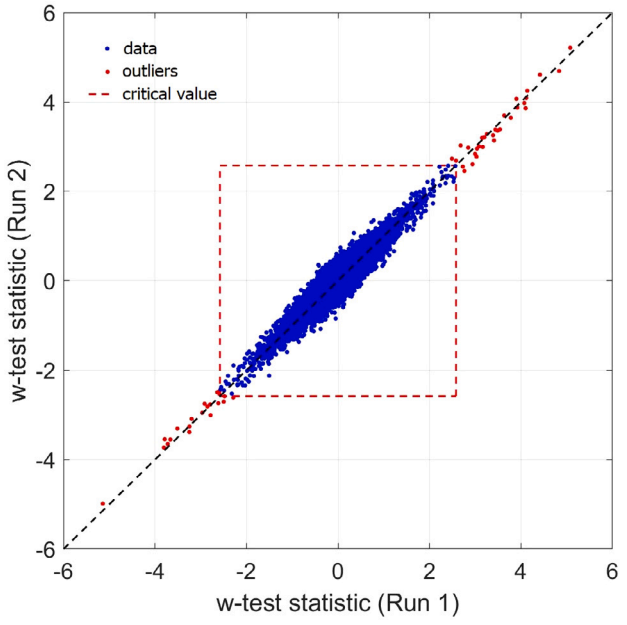


Fig. 12. The w-test values of GWSA modeling using LSBDL obtained for two independent runs; outliers indicated as red dots.

integrates both estimation and testing in the same framework (Teunissen, 2018).

10. Concluding remarks

10.1. Summary

This contribution presented the least-squares-based deep learning (LSBDL) method. LSBDL applies the least squares theory of linear models to the deep learning (DL) problem. A linear model usually links the observations y (dependent variables) to a set of explanatory variables (independent variables) through an unknown linear relation. A linear model of observation equation $y = Ax + e$ is a matrix representation of the above relation, where A, the given design matrix, consists of the explanatory variables. There are applications for which the relation between the observations and explanatory variables is not known a priori. This is the case in many engineering and (geo)science problems. Data-driven techniques such as those provided by artificial intelligence (AI) are widely used to establish such a (non)linear relation between input (explanatory variables) and output (observations). The link is realized by means of convolutions and nonlinear activation functions to capture the unknown relation. Weights and biases are both learnable parameters inside the DL network.

We investigated the possibility of applying the least squares theory of linear models to DL. This turned out to be feasible. A design matrix is trained to relate the input and output variables in an iterative procedure. The objective function is minimized using optimization techniques such as the gradient-based descent methods. Two methods were proposed: (1) Steepest descent (SD) method, (2) Gauss–Newton (GN) method. Analytical expressions, formulated by the estimates $\hat{X}$ and $\hat{E}$, were derived to compute the gradient of the objective function or the Jacobian matrix and hence to iteratively improve the initial weights and biases. The performance of the method was investigated on two basic illustrative examples, with synthetic data sets, providing promising results.

10.2. Highlights of LSBDL

The advantages and strengths of LSBDL are highlighted as follows:

- LSBDL was formulated using two gradient-based methods: Steepest descent (SD) and Gauss–Newton (GN) (Teunissen, 1990). Both formulations leverage the capability of deep learning (DL) methods to capture complex non-linearity and the appropriate statistical inference of the least squares theory.
- In the current structure of DL methods, both the output and hidden layer(s) are updated using gradient descent methods and back-propagation equations (Bishop and Bishop, 2024). In contrast, our approach follows a different procedure where the output layer with variables X does not require training in LSBDL. This is because the global minimizer for X is already known from the linear least squares theory, as described in Eq. (17).
- The contribution of the least squares residuals in training depends on observation weights expressed in Q^{-1} and Σ^{-1}. For example, if the precision of an observation improves by a factor of 2, its residual is multiplied by a factor of $2^2 = 4$, thereby having a more significant contribution in the training process. The current literature does not take this issue into consideration, see Buduma et al. (2022). The method can thus be used when dealing with inconsistent and heterogeneous data.
- By integrating prior knowledge and physics-based constraints into the least squares formulation, LSBDL enhances the interpretability and explainability of DL models (XAI). This integration allows the model to leverage well-established physical principles and to provide a clear understanding of the underlying processes, which thereby addresses one of the major challenges in traditional DL: the black-box problem (Von Eschenbach, 2021).
- Unlike available DL methods that rely on testing data to evaluate performance, LSBDL intrinsically determines quality control measures such as the covariance matrix and the precision of predicted outcomes, see Teunissen (2000b). This built-in mechanism provides insights into the confidence and variability of the predictions, thereby enhancing XAI. LSBDL offers quality assessment during training, which sets it apart from standard DL methods, often lacking such quality control features.
- LSBDL can employ available reliability and hypothesis testing theory to identify and remove model mis-specifications and outlying observations using the DIA theory (Baarda, 1968; Teunissen, 2000a, 2018). This is particularly useful for out-of-distribution (OOD) detection, as OOD data often manifest as outliers or anomalies in the input (Cui and Wang, 2022).

10.3. Outlook

LSBDL offers new opportunities in geoscience and aviation. The applications cover a variety of problems in which the ordinary linear model theory cannot directly be applied. Such applications range from current challenges in global navigation satellite systems (GNSS), the gravity recovery and climate experiment (GRACE), aviation acoustic and operations, multi-sensor integrated unmanned ship navigation, atmospheric monitoring and forecasting, multi-frequency multi-beam echo-sounders (MF-MBES) to satellite-based Lidar, SAR and multi-spectral imagery. Different aspects of LSBDL were addressed in this contribution. Local feature extractors and a multilayer DL formulation were provided. What kinds of local features, or how many layers to

use in the LSBDL are still relevant challenges to be addressed. It is also expected that LSBDL will be applied to various scientific and engineering applications. A wide range of applications in tomography, data assimilation and data fusion of multiple sensors is expected. For example, merging inconsistent, heterogeneous and statistically correlated data is one of the challenges in data fusion methods. LSBDL can take the covariance matrix of observations into consideration to train networks with heterogeneous information.

CRediT authorship contribution statement

Alireza Amiri-Simkooei: Writing – original draft, Validation, Methodology, Investigation, Formal analysis, Conceptualization. **Christian Tiberius:** Writing – review & editing, Validation, Methodology. **Roderik Lindenbergh:** Writing – review & editing, Visualization, Validation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The first two examples use simulated data for which the set-up is provided in the paper. The data of the third example and the code used during this study are available from the corresponding author upon request.

Acknowledgments

We are indebted to our colleague Peter Teunissen from the Department of Geoscience and Remote Sensing, Delft University of Technology for his invaluable feedback and guidance. His critical insights have significantly improved the quality of this contribution, which is greatly acknowledged. We would also like to express our gratitude to Filippo Gatti from Université Paris-Saclay, and four anonymous reviewers, for their valuable feedback and constructive comments that significantly improved the presentation of this research.

Appendix A. Proof of Theorem 1

This appendix provides the partial derivatives for two expressions needed for the derivations. The expressions can contain a couple of matrix operations like transposes, multiplications, inverses, traces and activation functions. To calculate the derivatives we follow the following convention.

For the sake of brevity, the summation $\sum$ will be disregarded. This is based on the 'summation' convention, which states that whenever there arises an expression in which an index occurs twice on the same side of any equation, or term within an equation, it is understood to represent a summation on the repeated index. A repeated index is called a 'summation index', while an unrepeated index is a 'free index'. For two matrices S and T, the *trace* of S can be denoted as $\mathrm{tr}(S) = s_{ii}$ (summation over i), while the matrix itself is symbolized as $S = s_{ij}$, its transpose is denoted as $S^T = s_{ij}^T = s_{ji}$, and the multiplication of S and T is denoted as $ST = s_{ik}t_{kj}$, where i and j are free indices and k is the summation index.

A.1. Proof of part I

Expression in Eq. (21) is proved. We aim to obtain the partial derivative of the following equation with respect to W:

$$\phi = \mathrm{tr}(B\mathtt{A}(DW)C) = \mathrm{tr}(BAC) \tag{A.1}$$

where B, D, W and C are arbitrary matrices with appropriate sizes, and $\mathtt{A}$ is a given activation function, to be applied in an element-wise manner to all entries of DW. The elements of the matrix W are assumed to be the function variables, for which the following partial derivatives are to be calculated: $\frac{\partial\phi}{\partial W}$. This is indeed the partial derivative of a scalar $\phi = \mathrm{tr}(.)$ with respect to the matrix W, which is a matrix of the same size. It reads

$$\frac{\partial\phi}{\partial W} = \frac{\partial(B\mathtt{A}(DW)C)_{ii}}{\partial(W)_{uv}} = \frac{\partial b_{ij}a_{jk}c_{ki}}{\partial w_{uv}} \tag{A.2}$$

where $a_{jk} = \mathtt{a}(d_{jl}w_{lk}) = A = \mathtt{A}(DW)$ is the design matrix. The above equation is then reformulated as

$$\frac{\partial\phi}{\partial W} = \frac{\partial b_{ij}\mathtt{a}(d_{jl}w_{lk})c_{ki}}{\partial w_{uv}} \tag{A.3}$$

or, taking the terms b_{ij} and c_{ki} out of the derivative, as

$$\frac{\partial\phi}{\partial W} = b_{ij}c_{ki}\frac{\partial\mathtt{a}(d_{jl}w_{lk})}{\partial w_{uv}} \tag{A.4}$$

Using the chain rule in partial derivative, we obtain

$$\frac{\partial\phi}{\partial W} = b_{ij}c_{ki}\frac{\partial\mathtt{a}(d_{jl}w_{lk})}{\partial d_{jl}w_{lk}}\frac{\partial d_{jl}w_{lk}}{\partial w_{uv}} \tag{A.5}$$

or, with $\mathtt{a}'(d_{jl}w_{lk}) = \frac{\partial\mathtt{a}(d_{jl}w_{lk})}{\partial d_{jl}w_{lk}}$, as

$$\frac{\partial\phi}{\partial W} = d_{jl}b_{ij}c_{ki}\mathtt{a}'(d_{jl}w_{lk})\frac{\partial w_{lk}}{\partial w_{uv}} \tag{A.6}$$

The partial derivative of w_{lk} with respect to w_{uv} is the product $\delta_{lu}\delta_{kv}$, where δ is the Kronecker delta, which is $\delta = 1$ if the indices are equal, and $\delta = 0$ otherwise. This, with $a'_{jk} = \mathtt{a}'(d_{jl}w_{lk})$, will then give

$$\frac{\partial\phi}{\partial W} = d_{jl}b_{ij}c_{ki}a'_{jk}\delta_{lu}\delta_{kv} \tag{A.7}$$

or

$$\frac{\partial\phi}{\partial W} = d_{ju}b_{ij}c_{vi}a'_{jv} = d_{uj}^T\left(b_{ji}^Tc_{iv}^T \odot a'_{jv}\right) \tag{A.8}$$

This finally provides the partial derivative in matrix notation as

$$\frac{\partial\phi}{\partial W} = D^T(B^TC^T \odot A') \tag{A.9}$$

which completes the proof of the first part. The function $\phi = \mathrm{tr}(.)$ can in fact be in different forms as

$$\phi = \mathrm{tr}(B\mathtt{A}(DW)C) = \mathrm{tr}(\mathtt{A}(DW)CB) = \mathrm{tr}(CB\mathtt{A}(DW)) \tag{A.10}$$

of which Eq. (A.9) is its partial derivative.

A.2. Proof of part II

Expression in Eq. (22) is proved. We aim to obtain the partial derivative of the following equation with respect to W:

$$\phi = \mathrm{tr}\left(E(B\mathtt{A}(DW))^{-1}C\right) = \mathrm{tr}\left(E(BA)^{-1}C\right) \tag{A.11}$$

where $A = \mathtt{A}(DW)$ is the activation function/matrix. We need to compute the partial derivative of ϕ with respect to matrix W. It reads

$$\frac{\partial\phi}{\partial W} = \frac{\partial\phi}{\partial w_{uv}} \tag{A.12}$$

where u and v are the free indices of W. For the sake of brevity we assume $H = (B\mathtt{A}(DW))^{-1}$. It is known that the differentiation, $\mathtt{d}(.)$, of matrix H is

$$\mathtt{d}(H) = -H\,\mathtt{d}(B\mathtt{A}(DW))H \tag{A.13}$$

Eqs. (A.11) and (A.12) with $H = (B\mathtt{A}(DW))^{-1}$ gives

$$\frac{\partial \phi}{\partial W} = \frac{\partial \mathrm{tr}(EHC)}{\partial w_{uv}} \tag{A.14}$$

or

$$\frac{\partial \phi}{\partial W} = \frac{\partial (e_{ij} h_{jk} c_{ki})}{\partial w_{uv}} = e_{ij} c_{ki} \frac{\partial h_{jk}}{\partial w_{uv}} \tag{A.15}$$

This, with Eq. (A.13), rewritten in the form of $\mathtt{d}(h_{jk}) = \mathtt{d}(h_{jl} b_{lm} a_{mn} h_{nk}) = h_{jl} h_{nk} b_{lm} \mathtt{d}(a_{mn})$, gives

$$\frac{\partial \phi}{\partial W} = -e_{ij} c_{ki} \frac{h_{jl} h_{nk} b_{lm} \partial (a_{mn})}{\partial w_{uv}} \tag{A.16}$$

Further, we have $\mathtt{d}(a_{mn}) = \mathtt{d}(\mathtt{a}(d_{mo} w_{on}))$, which yields

$$\frac{\partial \phi}{\partial W} = -e_{ij} c_{ki} h_{jl} h_{nk} b_{lm} \frac{\partial (\mathtt{a}(d_{mo} w_{on}))}{\partial w_{uv}} \tag{A.17}$$

Using the chain rule in the partial derivatives, we obtain

$$\frac{\partial \phi}{\partial W} = -e_{ij} c_{ki} h_{jl} h_{nk} b_{lm} \frac{\partial \mathtt{a}(d_{mo} w_{on})}{\partial d_{mo} w_{on}} \frac{\partial d_{mo} w_{on}}{\partial w_{uv}} \tag{A.18}$$

The partial derivative of w_{on} with respect to w_{uv} is the product $\delta_{ou} \delta_{nv}$

$$\frac{\partial \phi}{\partial W} = -e_{ij} c_{ki} h_{jl} h_{nk} b_{lm} a'_{mn} d_{mo} \delta_{ou} \delta_{nv} \tag{A.19}$$

where $a'_{mn} = \mathtt{a}'(d_{mo} w_{on})$ is the derivative of a_{mn}. The above equation simplifies to

$$\frac{\partial \phi}{\partial W} = -e_{ij} c_{ki} h_{jl} h_{vk} b_{lm} a'_{mv} d_{mu} \tag{A.20}$$

Rearranging the matrices products yields

$$\frac{\partial \phi}{\partial W} = -d^T_{mu} \left(b^T_{lm} h^T_{jl} e^T_{ij} c^T_{ki} h^T_{vk} \odot a'_{mv} \right) \tag{A.21}$$

which finally provides the partial derivative in the matrix notation as

$$\frac{\partial \phi}{\partial W} = -D^T \left(B^T H^T E^T C^T H^T \odot A' \right) \tag{A.22}$$

or, with $H = (BA)^{-1}$, as

$$\frac{\partial \phi}{\partial W} = -D^T \left(B^T (A^T B^T)^{-1} E^T C^T (A^T B^T)^{-1} \odot A' \right) \tag{A.23}$$

where $A = \mathtt{A}(DW)$. This completes the proof of the second part.

Appendix B. Proof of Theorem 2

B.1. Proof of part I

We derive the Jacobian matrix in Eq. (46). Let $h = BAz = B\mathtt{A}(DW)z$ be an m-vector, with the given matrices B, A, D and W of appropriate sizes. The partial derivative of h with respect to $w = \mathrm{vec}(W)$ is an $m \times kn$ Jacobian matrix as follows:

$$J = \frac{\partial h}{\partial w} = \frac{\partial B\mathtt{A}(DW)z}{\partial \mathrm{vec}(W)} \tag{B.1}$$

where $W = [w_1, \ldots, W_n]$ is a $k \times n$ matrix of running variables. Vector h can further be developed as

$$\begin{aligned} h &= [B\mathtt{a}(Dw_1), \ldots, B\mathtt{a}(Dw_n)]z \\ &= z_1 b_i^T \mathtt{a}(Dw_1), \ldots, z_n b_i^T \mathtt{a}(Dw_n) \end{aligned} \tag{B.2}$$

where b_i^T, for $i = 1, \ldots, m$, is the i^{th} row of B. The Jacobian matrix is then

$$\begin{aligned} J &= \left[\partial_{w_1} h, \ldots, \partial_{w_n} h \right] \\ &= \left[\partial_{w_1} z_1 b_i^T \mathtt{a}(Dw_1), \ldots, \partial_{w_n} z_n b_i^T \mathtt{a}(Dw_n) \right] \\ &= \left[z_1 b_i^T \partial_{w_1} \mathtt{a}(Dw_1), \ldots, z_n b_i^T \partial_{w_n} \mathtt{a}(Dw_n) \right] \end{aligned} \tag{B.3}$$

where the partial derivatives $\partial_{w_j} \mathtt{a}(Dw_j) = \frac{\partial \mathtt{a}(Dw_j)}{\partial w_j}$ are given as (note $w_j = [w_{1j}, \ldots, w_{kj}]^T$)

$$\partial_{w_j} \mathtt{a}(Dw_j) = \left[\partial w_{1j} \mathtt{a}(Dw_j), \ldots, \partial w_{kj} \mathtt{a}(Dw_j) \right] \tag{B.4}$$

for $j = 1, \ldots, n$. The above equation gives (we have $D = [d_1, \ldots, d_k]$)

$$\partial_{w_j} \mathtt{a}(Dw_j) = \left[\mathtt{a}'(Dw_j) \odot d_1, \ldots, \mathtt{a}'(Dw_j) \odot d_k \right] \tag{B.5}$$

or

$$\partial_{w_j} \mathtt{a}(Dw_j) = (\mathtt{a}'(Dw_j) u_k^T) \odot D = (a'_j u_k^T) \odot D \tag{B.6}$$

where u_k is a summation vector of size k. This, with Eq. (B.3), gives

$$\begin{aligned} J &= \left[z_1 b_i^T (a'_1 u_k^T) \odot D, \ldots, z_n b_i^T (a'_n u_k^T) \odot D \right] \\ &= b_i^T \left[z_1 (a'_1 u_k^T) \odot D, \ldots, z_n (a'_n u_k^T) \odot D \right] \\ &= b_i^T \left[(u_m z^T \otimes u_k^T) \odot (A' \otimes u_k^T) \odot (u_n^T \otimes D) \right] \end{aligned} \tag{B.7}$$

where u_m and u_n are the summation vectors of size m and n, respectively. The above equation, with $B = b_i^T$ for $i = 1, \ldots, m$, gives

$$J = B \left((u_m z^T \otimes u_k^T) \odot M \right) \tag{B.8}$$

where $M = (A' \otimes u_k^T) \odot (u_n^T \otimes D)$ is an $m \times kn$ matrix.

B.2. Proof of part II

We derive the Jacobian matrix in Eq. (47). Let $h = CA^T z = C\mathtt{A}(W^T D^T)z$ be an m-vector, with the given matrices C, A, D and W of appropriate sizes. The partial derivative of h with respect to $w = \mathrm{vec}(W)$ is an $m \times kn$ Jacobian matrix as follows:

$$J = \frac{\partial h}{\partial w} = \frac{\partial C\mathtt{A}(W^T D^T)z}{\partial \mathrm{vec}(W)} \tag{B.9}$$

where $W = [w_1, \ldots, W_n]$ is a $k \times n$ matrix of running variables. Vector h can further be developed as

$$h = C\mathtt{A}(W^T D^T)z = c_i^T \begin{bmatrix} \mathtt{a}(w_1^T D^T)z \\ \vdots \\ \mathtt{a}(w_n^T D^T)z \end{bmatrix} \tag{B.10}$$

where c_i^T is the i^{th} row of C, for $i = 1, \ldots, m$. We then have

$$h = c_{i1} \mathtt{a}(w_1^T D^T)z + \cdots + c_{in} \mathtt{a}(w_n^T D^T)z \tag{B.11}$$

The Jacobian matrix is

$$\begin{aligned} J &= \left[\partial_{w_1} h, \ldots, \partial_{w_n} h \right] \\ &= \left[\partial_{w_1} c_{i1} \mathtt{a}(w_1^T D^T)z, \ldots, \partial_{w_n} c_{in} \mathtt{a}(w_n^T D^T)z \right] \\ &= \left[c_{i1} \partial_{w_1} z^T \mathtt{a}(Dw_1), \ldots, c_{in} \partial_{w_n} z^T \mathtt{a}(Dw_n) \right] \\ &= \left[c_{i1} z^T \partial_{w_1} \mathtt{a}(Dw_1), \ldots, c_{in} z^T \partial_{w_n} \mathtt{a}(Dw_n) \right] \end{aligned} \tag{B.12}$$

because $\mathtt{a}(w_j^T D^T)z = z^T \mathtt{a}(Dw_j)$ is a scalar. The partial derivatives are given as (see above)

$$\partial_{w_j} \mathtt{a}(Dw_j) = (\mathtt{a}'(Dw_j) u_k^T) \odot D = (a'_j u_k^T) \odot D \tag{B.13}$$

where u_k is the summation vector of size k. With Eq. (B.12), this gives

$$J = \left[c_{i1} z^T (a'_j u_k^T \odot D), \ldots, c_{in} z^T (a'_n u_k^T \odot D) \right] \tag{B.14}$$

The above equation, with $C = c_i^T$ for $i = 1, \ldots, m$, gives

$$J = (C \otimes u_k^T) \odot (u_m z^T M) \tag{B.15}$$

where $M = (A' \otimes u_k^T) \odot (u_n^T \otimes D)$ is an $m \times kn$ matrix.

Appendix C. Entries of L and D

The block-triangular decomposition of N in Eq. (87) is of the form $N = LDL^T$, where L, a lower block-triangular unit matrix and D, a block-diagonal matrix, are given in Eq. (88). The sub-matrices L_{ij} and D_{jj} are as follows:

$$L_{ij} = N_{ij|J} N_{jj|J}^{-1} \tag{C.1}$$

where $j < i$ and $J = \{1, \dots, j-1\}$ and

$$N_{ij|J} = N_{ij} - \sum_{k=1}^{j-1} N_{ik|J} N_{kk}^{-1} N_{ik|J}^T \tag{C.2}$$

Further, the matrices $D_{jj} = N_{jj|J}$ are given as

$$D_{jj} = N_{jj|J} = N_{jj} - \sum_{k=1}^{j-1} N_{jk|J} N_{kk|J}^{-1} N_{jk|J}^T \tag{C.3}$$

Proof. Substitute the lower block-triangular unit matrix L and the block-diagonal matrix D into LDL^T to show that it equals N in Eq. (87).

For example, the sub-matrices D_{jj} are of the form

$$\begin{aligned} D_{11} &= N_{11} \\ D_{22} &= N_{22} - N_{21} N_{11}^{-1} N_{21}^T = N_{22|1} \\ D_{33} &= N_{33} - \textstyle\sum_{j=1}^{2} N_{3j|J} N_{jj|J}^{-1} N_{3j|J}^T = N_{33|J} \end{aligned} \tag{C.4}$$

where $J = \{\}$ for $j = 1$, and $J = \{1\}$ for $j = 2$. Accordingly, the sub-matrices L_{ij} are

$$\begin{aligned} L_{21} &= N_{21} N_{11}^{-1} \\ L_{31} &= N_{31} N_{11}^{-1} \\ L_{32} &= (N_{32} - N_{31} N_{11}^{-1} N_{31}^T) N_{22|1}^{-1} = N_{32|1} N_{22|1}^{-1} \\ L_{42} &= (N_{42} - N_{41} N_{11}^{-1} N_{41}^T) N_{22|1}^{-1} = N_{42|1} N_{22|1}^{-1} \\ L_{43} &= (N_{43} - \textstyle\sum_{j=1}^{2} N_{4j|J} N_{jj|J}^{-1} N_{4j|J}^T) N_{33|J}^{-1} = N_{43|J}^{-1} N_{33|J}^{-1} \end{aligned} \tag{C.5}$$

where $J = \{\}$ for $j = 1$, and $J = \{1\}$ for $j = 2$.

Appendix D. Proof of Theorem 3

D.1. Proof of part I

To derive the first part of Theorem 3 (Eq. (101)), just for notational convenience, let us assume $\mathbf{S} = \mathtt{A}_1$, $\mathtt{A} = \mathtt{A}_2$, $U = W_1$, and $W = W_2$, Eq. (99) reads

$$\phi_1 = \mathrm{tr}(B\mathtt{A}(\mathbf{S}(DU)W)C) = \mathrm{tr}(BAC) \tag{D.1}$$

It is therefore required to calculate $\frac{\partial \phi_1}{\partial U}$. This reads

$$\frac{\partial \phi_1}{\partial U} = \frac{\partial b_{ij} a_{jk} c_{ki}}{\partial u_{uv}} = b_{ij} c_{ki} \frac{\partial a_{jk}}{\partial u_{uv}} = b_{ij} c_{ki} \frac{\partial \mathtt{a}(s_{jl} w_{lk})}{\partial u_{uv}} \tag{D.2}$$

Using the chain rule in partial derivatives yields

$$\frac{\partial \phi_1}{\partial U} = b_{ij} c_{ki} \frac{\partial \mathtt{a}(s_{jl} w_{lk})}{\partial s_{jl} w_{lk}} \frac{\partial s_{jl} w_{lk}}{\partial u_{uv}} \tag{D.3}$$

or

$$\frac{\partial \phi_1}{\partial U} = b_{ij} c_{ki} w_{lk} a'_{jk} \frac{\partial s_{jl}}{\partial u_{uv}} \tag{D.4}$$

where $a'_{jk} = \mathtt{a}'(s_{jl} w_{lk})$ is the derivative of $\mathtt{a}(s_{jl} w_{lk})$. The above equation, with $s_{jl} = \mathbf{s}(d_{jm} u_{ml})$, gives

$$\frac{\partial \phi_1}{\partial U} = b_{ij} c_{ki} w_{lk} a'_{jk} \frac{\partial \mathbf{s}(d_{jm} u_{ml})}{\partial u_{uv}} \tag{D.5}$$

or, with the chain rule, as

$$\frac{\partial \phi_1}{\partial U} = b_{ij} c_{ki} w_{lk} a'_{jk} \frac{\partial \mathbf{s}(d_{jm} u_{ml})}{\partial d_{jm} u_{ml}} \frac{\partial d_{jm} u_{ml}}{\partial u_{uv}} \tag{D.6}$$

or, with $s'_{jl} = \frac{\partial \mathbf{s}(d_{jm} u_{ml})}{\partial d_{jm} u_{ml}}$, as

$$\frac{\partial \phi_1}{\partial U} = b_{ij} c_{ki} w_{lk} d_{jm} a'_{jk} s'_{jl} \frac{\partial u_{ml}}{\partial u_{uv}} \tag{D.7}$$

or

$$\frac{\partial \phi_1}{\partial U} = b_{ij} c_{ki} w_{lk} d_{jm} a'_{jk} s'_{jl} \delta_{mu} \delta_{lv} = b_{ij} c_{ki} w_{vk} d_{ju} a'_{jk} s'_{jv} \tag{D.8}$$

Rearranging the indices from u to v gives

$$\frac{\partial \phi_1}{\partial U} = d_{uj}^T b_{ji}^T c_{ik}^T a'_{jk} w_{kv}^T s'_{jv} \tag{D.9}$$

or in a clearer notation as

$$\frac{\partial \phi_1}{\partial U} = d_{uj}^T \left((b_{ji}^T c_{ik}^T \odot a'_{jk}) w_{kv}^T \otimes s'_{jv} \right) \tag{D.10}$$

or in matrix notation as

$$\frac{\partial \phi_1}{\partial U} = D^T \left((B^T C^T \odot A') W^T \odot S' \right) \tag{D.11}$$

Back substitution from $S = A_1$, $A = A_2$, $U = W_1$, and $W = W_2$ proves the theorem as

$$\frac{\partial \phi_1}{\partial U} = D^T \left((B^T C^T \odot A'_1) W_1^T \odot A'_2 \right) \tag{D.12}$$

which completes the proof.

D.2. Proof of part II

To derive the second part of Theorem 2 (Eq. (102)), we again assume $\mathbf{S} = \mathtt{A}_1$, $\mathtt{A} = \mathtt{A}_2$, $U = W_1$, and $W = W_2$. Eq. (100) will then read

$$\phi_2 = \mathrm{tr}\left(E(B\mathtt{A}(\mathbf{S}(DU)W))^{-1} C \right) = \mathrm{tr}\left(E(BA)^{-1} C \right) \tag{D.13}$$

where $A = \mathtt{A}(\mathbf{S}(DU)W)$ is the activation function. For the sake of brevity we assume $H = N^{-1} = (BA)^{-1} = (B\mathtt{A}(\mathbf{S}(DU)W))^{-1}$, where $N = BA = H^{-1}$. The differentiation, $\mathtt{d}(.)$, of matrix H is

$$\mathtt{d}(H) = d(BA)^{-1} = -H\, \mathtt{d}(N) H \tag{D.14}$$

It is then required to compute the partial derivative of $\phi_2 = \mathrm{tr}(EHC)$ with respect to matrix U as

$$\frac{\partial \phi_2}{\partial U} = \frac{\partial \phi}{\partial u_{uv}} = \frac{\partial e_{ij} h_{jk} c_{ki}}{\partial u_{uv}} = e_{ij} c_{ki} \frac{\partial h_{jk}}{\partial u_{uv}} \tag{D.15}$$

where u and v are the free indices of U. The above equation with Eq. (D.14) reads

$$\frac{\partial \phi_2}{\partial U} = -e_{ij} c_{ki} h_{jl} h_{mk} \frac{\partial n_{lm}}{\partial u_{uv}} = -e_{ij} c_{ki} h_{jl} h_{mk} b_{lo} \frac{\partial a_{om}}{\partial u_{uv}} \tag{D.16}$$

The term $\frac{\partial a_{om}}{\partial u_{uv}}$ can further be developed as

$$\frac{\partial a_{om}}{\partial u_{uv}} = \frac{\partial \mathtt{a}(s_{op} w_{pm})}{\partial u_{uv}} = \frac{\partial \mathtt{a}(s_{op} w_{pm})}{\partial s_{op} w_{pm}} \frac{\partial s_{op} w_{pm}}{\partial u_{uv}} \tag{D.17}$$

or

$$\frac{\partial a_{om}}{\partial u_{uv}} = a'_{om} w_{pm} \frac{\partial s_{op}}{\partial u_{uv}} = a'_{om} w_{pm} \frac{\partial \mathbf{s}(d_{oq} u_{qp})}{\partial u_{uv}} \tag{D.18}$$

or

$$\frac{\partial a_{om}}{\partial u_{uv}} = a'_{om} w_{pm} s'_{op} d_{oq} \frac{\partial u_{qp}}{\partial u_{uv}} \tag{D.19}$$

This with Eq. (D.16) gives

$$\frac{\partial \phi_2}{\partial U} = -e_{ij} c_{ki} h_{jl} h_{mk} b_{lo} a'_{om} w_{pm} s'_{op} d_{oq} \delta_{qu} \delta_{pv} \tag{D.20}$$

or

$$\frac{\partial \phi_2}{\partial U} = -e_{ij} c_{ki} h_{jl} h_{mk} b_{lo} a'_{om} w_{vm} s'_{ov} d_{ou} \tag{D.21}$$

or

$$\frac{\partial \phi_2}{\partial U} = -d^T_{uo} b^T_{ol} h^T_{lj} e^T_{ji} c^T_{ik} h^T_{km} a'_{om} w^T_{mv} s'_{ov} \tag{D.22}$$

or

$$\frac{\partial \phi_2}{\partial U} = -D^T \left((B^T H^T E^T C^T H^T \odot A') W^T \odot S' \right) \tag{D.23}$$

The above equation, with $S = A_1$, $A = A_2$, $U = W_1$, and $W = W_2$, yields

$$\frac{\partial \phi_2}{\partial U} = -D^T \left((B^T H^T E^T C^T H^T \odot A'_1) W_1^T \odot A'_2 \right) \tag{D.24}$$

which completes the proof.

References

Ahmad, M.W., Reynolds, J., Rezgui, Y., 2018. Predictive modelling for solar thermal energy systems: A comparison of support vector regression, random forest, extra trees and regression trees. J. Clean. Prod. 203, 810–821.

Albawi, S., Mohammed, T.A., Al-Zawi, S., 2017. Understanding of a convolutional neural network. In: 2017 International Conference on Engineering and Technology. ICET, IEEE, pp. 1–6.

Ali, S., Liu, D., Fu, Q., Cheema, M.J.M., Pham, Q.B., Rahaman, M.M., Dang, T.D., Anh, D.T., 2021. Improving the resolution of GRACE data for spatio-temporal groundwater storage assessment. Remote Sens. 13 (17), 3513.

Alla, H., Moumoun, L., Balouki, Y., 2021. A multilayer perceptron neural network with selective-data training for flight arrival delay prediction. Sci. Program. 2021 (1), 5558918.

Amiri-Simkooei, A., 2007. Least-Squares Variance Component Estimation: Theory and GPS Applications (Ph.D. thesis). Delft University of Technology, Publication on Geodesy, 64, Netherlands Geodetic Commission.

Amiri-Simkooei, A., 2009. Noise in multivariate GPS position time-series. J. Geod. 83 (2), 175–187.

Amiri-Simkooei, A., 2013. On the nature of GPS draconitic year periodic pattern in multivariate position time series. J. Geophys. Res.: Solid Earth 118 (5), 2500–2511.

Amiri-Simkooei, A., 2019. Unified least-squares formulation of a linear model with hard constraints. J. Surv. Eng. 145 (4), 04019012.

Amiri-Simkooei, A.R., Hosseini-Asl, M., Safari, A., 2018. Least squares 2D bi-cubic spline approximation: Theory and applications. Measurement 127, 366–378.

Amiri-Simkooei, A., Zangeneh-Nejad, F., Asgari, J., 2016. On the covariance matrix of weighted total least-squares estimates. J. Surv. Eng. 142 (3), 04015014.

Araya-Polo, M., Jennings, J., Adler, A., Dahlke, T., 2018. Deep-learning tomography. Leading Edge 37 (1), 58–66.

Arrieta, A.B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R., et al., 2020. Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. Inf. Fusion 58, 82–115.

Asgarimehr, M., Arnold, C., Weigel, T., Ruf, C., Wickert, J., 2022. GNSS reflectometry global ocean wind speed using deep learning: Development and assessment of CyGNSSnet. Remote Sens. Environ. 269, 112801.

Awad, M., Khanna, R., 2015. Support vector regression. Efficient Learn. Mach. 67–80.

Baarda, W., 1968. A Testing Procedure for Use in Geodetic Networks. Tech. rep., Netherlands Geodetic Commission, Publ. on Geodesy, New Series, Vol. 2(5), Delft.

Bagnall, A., Lines, J., Bostrom, A., Large, J., Keogh, E., 2017. The great time series classification bake off: A review and experimental evaluation of recent algorithmic advances. Data Min. Knowl. Discov. 31, 606–660.

Bai, T., Luo, J., Zhao, J., Wen, B., Wang, Q., 2021. Recent advances in adversarial training for adversarial robustness. arXiv preprint arXiv:2102.01356.

Bakhshi, A., Chalup, S., 2021. Multimodal emotion recognition based on speech and physiological signals using deep neural networks. In: International Conference on Pattern Recognition. Springer, pp. 289–300.

Baldi, P., Sadowski, P.J., 2013. Understanding dropout. In: Advances in Neural Information Processing Systems, vol. 26.

Barata, J.C.A., Hussein, M.S., 2012. The Moore–Penrose pseudoinverse: A tutorial review of the theory. Braz. J. Phys. 42 (1), 146–165.

Baydin, A.G., Pearlmutter, B.A., Radul, A.A., Siskind, J.M., 2018. Automatic differentiation in machine learning: A survey. J. Mach. Learn. Res. 18 (153), 1–43.

Belgiu, M., Drăguţ, L., 2016. Random forest in remote sensing: A review of applications and future directions. ISPRS J. Photogramm. Remote Sens. 114, 24–31.

Belhadi, A., Djenouri, Y., Djenouri, D., Michalak, T., Lin, J.C.-W., 2020. Deep learning versus traditional solutions for group trajectory outliers. IEEE Trans. Cybern.

Ben-Israel, A., Greville, T.N., 2003. Generalized Inverses: Theory and Applications, vol. 15, Springer Science & Business Media.

Bercovier, M., Jacobi, A., 1994. Minimization, constraints and composite Bézier curves. Comput. Aided Geometric Des. 11 (5), 533–563.

Berend, D., Xie, X., Ma, L., Zhou, L., Liu, Y., Xu, C., Zhao, J., 2020. Cats are not fish: Deep learning testing calls for out-of-distribution awareness. In: Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering. pp. 1041–1052.

Bishop, C.M., Bishop, H., 2024. Deep Learning: Foundations and Concepts. Springer Nature.

Bjorck, N., Gomes, C.P., Selman, B., Weinberger, K.Q., 2018. Understanding batch normalization. In: Advances in Neural Information Processing Systems, vol. 31.

Botev, A., 2020. The Gauss-NEwton Matrix for Deep Learning Models and Its Applications (Ph.D. thesis). UCL (University College London).

Botev, A., Ritter, H., Barber, D., 2017. Practical Gauss-Newton optimisation for deep learning. In: International Conference on Machine Learning. PMLR, pp. 557–565.

Breiman, L., 2001. Random forests. Mach. Learn. 45 (1), 5–32.

Buduma, N., Buduma, N., Papa, J., 2022. Fundamentals of Deep Learning. O'Reilly Media, Inc..

Buscema, M., 1998. Back propagation neural networks. Substance Use Misuse 33 (2), 233–270.

Cai, Z., Chen, J., Liu, M., Liu, X., 2020. Deep least-squares methods: An unsupervised learning-based numerical method for solving elliptic PDEs. J. Comput. Phys. 420, 109707.

Carr, J.C., Beatson, R.K., Cherrie, J.B., Mitchell, T.J., Fright, W.R., McCallum, B.C., Evans, T.R., 2001. Reconstruction and representation of 3D objects with radial basis functions. In: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques. pp. 67–76.

Casella, G., Berger, R.L., 2021. Statistical Inference. Cengage Learning.

Cha, Y.-J., Mostafavi, A., Benipal, S.S., 2023. DNoiseNet: Deep learning-based feedback active noise control in various noisy environments. Eng. Appl. Artif. Intell. 121, 105971.

Chang, J., Song, D., 2023. Detection of sugar content in food based on the electrochemical method with the assistance of partial least square method and deep learning. J. Food Meas. Charact. 17 (5), 4864–4869.

Chen, C., He, C., Hu, C., Pei, H., Jiao, L., 2019a. A deep neural network based on an attention mechanism for SAR ship detection in multiscale and complex scenarios. IEEE Access 7, 104848–104863.

Chen, L., He, Q., Liu, K., Li, J., Jing, C., 2019b. Downscaling of GRACE-derived groundwater storage based on the random forest model. Remote Sens. 11 (24), 2979.

Chen, Z., Jin, M., Deng, Y., Wang, J.-S., Huang, H., Deng, X., Huang, C.-M., 2019c. Improvement of a deep learning algorithm for total electron content maps: Image completion. J. Geophys. Res. Space Phys. 124 (1), 790–800.

Chen, X.-W., Lin, X., 2014. Big data deep learning: Challenges and perspectives. IEEE Access 2, 514–525.

Chen, J., Zeng, G.-Q., Zhou, W., Du, W., Lu, K.-D., 2018. Wind speed forecasting using nonlinear-learning ensemble of deep learning time series prediction and extremal optimization. Energy Convers. Manag. 165, 681–695.

Chou, Y.-L., Moreira, C., Bruza, P., Ouyang, C., Jorge, J., 2022. Counterfactuals and causability in explainable artificial intelligence: Theory, algorithms, and applications. Inf. Fusion 81, 59–83.

Costarelli, D., Spigler, R., et al., 2013. Constructive approximation by superposition of sigmoidal functions. Anal. Theory Appl. 29 (2), 169–196.

Criminisi, A., Shotton, J., Konukoglu, E., et al., 2012. Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. Found. Trends Comput. Graphics Vis. 7 (2–3), 81–227.

Cui, P., Wang, J., 2022. Out-of-distribution (OOD) detection based on deep learning: A review. Electronics 11 (21), 3500.

Cuomo, S., Di Cola, V.S., Giampaolo, F., Rozza, G., Raissi, M., Piccialli, F., 2022. Scientific machine learning through physics–informed neural networks: Where we are and what is next. J. Sci. Comput. 92 (3), 88.

Cybenko, G., 1989. Approximation by superpositions of a sigmoidal function. Math. Control Signals Systems 2 (4), 303–314.

Dai, M., 2024. A hybrid machine learning-based model for predicting flight delay through aviation big data. Sci. Rep. 14 (1), 4603.

Daly, C., Baba, W., Bergsma, E., Thoumyre, G., Almar, R., Garlan, T., 2022. The new era of regional coastal bathymetry from space: A showcase for West Africa using optical Sentinel-2 imagery. Remote Sens. Environ. 278, 113084.

E-Silva, L.C., Murça, M.C.R., 2023. A data analytics framework for anomaly detection in flight operations. J. Air Transp. Manag. 110, 102409.

Elbrächter, D., Perekrestenko, D., Grohs, P., Bölcskei, H., 2021. Deep neural network approximation theory. IEEE Trans. Inform. Theory 67 (5), 2581–2623.

Elsner, J.B., Tsonis, A.A., 2013. Singular Spectrum Analysis: A New Tool in Time Series Analysis. Springer Science & Business Media.

Elsworth, S., Güttel, S., 2020. Time series forecasting using LSTM networks: a symbolic approach. arXiv preprint arXiv:2003.05672.

Elthakeb, A.T., Pilligundla, P., Mireshghallah, F., Yazdanbakhsh, A., Esmaeilzadeh, H., 2018. Releq: A reinforcement learning approach for deep quantization of neural networks. arXiv preprint arXiv:1811.01704.

Emmert-Streib, F., Yang, Z., Feng, H., Tripathi, S., Dehmer, M., 2020. An introductory review of deep learning for prediction models with big data. Front. Artif. Intell. 3, 4.

Ezugwu, E., Fadare, D., Bonney, J., Da Silva, R., Sales, W., 2005. Modelling the correlation between cutting and process parameters in high-speed machining of inconel 718 alloy using an artificial neural network. Int. J. Mach. Tools Manuf. 45 (12–13), 1375–1385.

Feng, H., Zhou, Y., Zeng, W., Ding, C., 2023. Review on metrics and prediction methods of civil aviation noise. Int. J. Aeronaut. Space Sci. 24 (5), 1199–1213.

Foroumandi, E., Nourani, V., Huang, J.J., Moradkhani, H., 2022. Drought monitoring by downscaling GRACE-derived terrestrial water storage anomalies: A deep learning approach. J. Hydrol. 128838.

Fracastoro, G., Magli, E., Poggi, G., Scarpa, G., Valsesia, D., Verdoliva, L., 2021. Deep learning methods for synthetic aperture radar image despeckling: An overview of trends and perspectives. IEEE Geosci. Remote Sens. Mag. 9 (2), 29–51.

Gao, W., Li, Z., Chen, Q., Jiang, W., Feng, Y., 2022. Modelling and prediction of GNSS time series using GBDT, LSTM and SVM machine learning approaches. J. Geodesy 96 (10), 1–17.

Garbin, C., Zhu, X., Marques, O., 2020. Dropout vs. batch normalization: An empirical study of their impact to deep learning. Multimedia Tools Appl. 79 (19), 12777–12815.

Gege, P., 2004. The water color simulator WASI: An integrating software tool for analysis and simulation of optical in situ spectra. Comput. Geosci. 30 (5), 523–532.

Gilks, W.R., Richardson, S., Spiegelhalter, D., 1995. Markov Chain Monte Carlo in Practice. CRC Press.

Glorot, X., Bengio, Y., 2010. Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, JMLR Workshop and Conference Proceedings. pp. 249–256.

Golub, G.H., Pereyra, V., 1973. The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate. SIAM J. Numer. Anal. 10 (2), 413–432.

Gratton, S., Lawless, A.S., Nichols, N.K., 2007. Approximate Gauss–Newton methods for nonlinear least squares problems. SIAM J. Optim. 18 (1), 106–132.

Greff, K., Srivastava, R.K., Koutník, J., Steunebrink, B.R., Schmidhuber, J., 2016. LSTM: A search space odyssey. IEEE Trans. Neural Netw. Learn. Syst. 28 (10), 2222–2232.

Greville, T., 1959. The pseudoinverse of a rectangular or singular matrix and its application to the solution of systems of linear equations. SIAM Rev. 1 (1), 38–43.

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J., et al., 2018. Recent advances in convolutional neural networks. Pattern Recognit. 77, 354–377.

Hansen, P.C., 1999. The L-Curve and Its Use in the Numerical Treatment of Inverse Problems. Tech. Rep., Department of Mathematical Modelling, Technical University of Denmark.

Hansen, P., 2001. The L-Curve and Its Use in the Numerical Treatment of Inverse Problems; Invited Chapter in P. Johnston (Ed.), Computational Inverse Problems in Electrocardiology, Computational Inverse Problems in Electrocardiology. WIT Press, Southampton, 2001, pp. 119–142.

Hartley, H.O., 1961. The modified Gauss-Newton method for the fitting of non-linear regression functions by least squares. Technometrics 3 (2), 269–280.

Hastie, T., Tibshirani, R., Friedman, J.H., Friedman, J.H., 2009. The Elements of Statistical Learning: Data Mining, Inference, and Prediction, vol. 2, Springer.

He, Y., Qin, H., 2004. Surface reconstruction with triangular B-splines. In: Geometric Modeling and Processing, 2004. Proceedings. IEEE, pp. 279–287.

He, K., Zhang, X., Ren, S., Sun, J., 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1026–1034.

He, K., Zhang, X., Ren, S., Sun, J., 2016a. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–778.

He, K., Zhang, X., Ren, S., Sun, J., 2016b. Identity mappings in deep residual networks. In: European Conference on Computer Vision. Springer, pp. 630–645.

Helgo, M., 2023. Deep learning and machine learning algorithms for enhanced aircraft maintenance and flight data analysis. J. Robot. Spectrum 1, 90–99.

Hernández-García, A., König, P., 2018. Data augmentation instead of explicit regularization. arXiv preprint arXiv:1806.03852.

Hinton, G.E., Osindero, S., Teh, Y.-W., 2006. A fast learning algorithm for deep belief nets. Neural Comput. 18 (7), 1527–1554.

Ho, T.K., 1995. Random decision forests. In: Proceedings of 3rd International Conference on Document Analysis and Recognition, vol. 1, IEEE, pp. 278–282.

Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Comput. 9 (8), 1735–1780.

Hornik, K., Stinchcombe, M., White, H., 1989. Multilayer feedforward networks are universal approximators. Neural Netw. 2 (5), 359–366.

Hsu, L.-T., 2017. GNSS multipath detection using a machine learning approach. In: 2017 IEEE 20th International Conference on Intelligent Transportation Systems. ITSC, IEEE, pp. 1–6.

Hu, J., Niu, H., Carrasco, J., Lennox, B., Arvin, F., 2020. Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning. IEEE Trans. Veh. Technol. 69 (12), 14413–14423.

Ibrahim, A., Mirjalili, S., El-Said, M., Ghoneim, S.S., Al-Harthi, M.M., Ibrahim, T.F., El-Kenawy, E.-S.M., 2021. Wind speed ensemble forecasting based on deep learning using adaptive dynamic optimization algorithm. IEEE Access 9, 125787–125804.

Ioffe, S., Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning. PMLR, pp. 448–456.

Jaeger, H., 2001. The Echo State Approach to Analysing and Training Recurrent Neural Networks-With an Erratum Note, vol. 148, (no. 34), German National Research Center for Information Technology GMD Technical Report, Bonn, Germany, p. 13.

Jin, B., Li, X., Quan, Q., Zhou, Z., 2024. Conductivity imaging from internal measurements with mixed least-squares deep neural networks. SIAM J. Imaging Sci. 17 (1), 147–187.

Jones, G.L., Qin, Q., 2022. Markov chain Monte Carlo in practice. Annu. Rev. Stat. Appl. 9, 557–578.

Jordan, M.I., Mitchell, T.M., 2015. Machine learning: Trends, perspectives, and prospects. Science 349 (6245), 255–260.

Jyolsna, P., Kambhammettu, B., Gorugantula, S., 2021. Application of random forest and multi-linear regression methods in downscaling GRACE derived groundwater storage changes. Hydrol. Sci. J. 66 (5), 874–887.

Karaoğlu, U., Mbah, O., Zeeshan, Q., 2023. Applications of machine learning in aircraft maintenance. J. Eng. Manag. Syst. Eng. 2 (1), 76–95.

Karniadakis, G.E., Kevrekidis, I.G., Lu, L., Perdikaris, P., Wang, S., Yang, L., 2021. Physics-informed machine learning. Nat. Rev. Phys. 3 (6), 422–440.

Kawaguchi, K., 2016. Deep learning without poor local minima. In: Advances in Neural Information Processing Systems, vol. 29.

Kim, H., Lakshmi, V., 2018. Use of cyclone global navigation satellite system (CYGNSS) observations for estimation of soil moisture. Geophys. Res. Lett. 45 (16), 8272–8282.

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

Koch, K.R., 1999. Parameter Estimation and Hypothesis Testing in Linear Models. Springer Verlag, Berlin.

Kong, C., Lucey, S., 2017. Take it in your stride: Do we need striding in CNNs?. arXiv preprint arXiv:1712.02502.

Kruse, R., Borgelt, C., Braune, C., Mostaghim, S., Steinbrecher, M., Klawonn, F., Moewes, C., 2011. Computational Intelligence: A Methodological Introduction. Springer.

Kumar, S., Ningombam, D., 2018. Short-term forecasting of stock prices using long short term memory. In: 2018 International Conference on Information Technology. ICIT, IEEE, pp. 182–186.

Kurzidem, I., Saad, A., Schleiss, P., 2020. A systematic approach to analyzing perception architectures in autonomous vehicles. In: International Symposium on Model-Based Safety and Assessment. Springer, pp. 149–162.

Lee, Y., Han, D., Ahn, M.-H., Im, J., Lee, S.J., 2019. Retrieval of total precipitable water from himawari-8 AHI data: A comparison of random forest, extreme gradient boosting, and deep neural network. Remote Sensing 11 (15), 1741.

Lehmann, F., Gatti, F., Bertin, M., Clouteau, D., 2023. Fourier neural operator surrogate model to predict 3D seismic waves propagation. arXiv preprint arXiv:2304.10242.

Lehmann, F., Gatti, F., Bertin, M., Clouteau, D., 2024. Synthetic ground motions in heterogeneous geologies: the HEMEW-3D dataset for scientific machine learning. Earth Syst. Sci. Data Discuss. 2024b, 1–26.

Lehmann, F., Gatti, F., Bertin, M., Clouteau, D., 2024a. 3D elastic wave propagation with a factorized Fourier neural operator (F-FNO). Comput. Methods Appl. Mech. Engrg. 420, 116718.

Li, G., Jung, J.J., 2023. Deep learning for anomaly detection in multivariate time series: Approaches, applications, and challenges. Inf. Fusion 91, 93–102.

Li, M., Zhang, T., Chen, Y., Smola, A.J., 2014. Efficient mini-batch training for stochastic optimization. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 661–670.

Liaw, A., Wiener, M., et al., 2002. Classification and regression by random forest. R News 2 (3), 18–22.

Lu, J., Shen, Z., Yang, H., Zhang, S., 2021. Deep network approximation for smooth functions. SIAM J. Math. Anal. 53 (5), 5465–5506.

Lukosevicius, M., Jaeger, H., 2009. Reservoir computing approaches to recurrent neural network training. Comput. Sci. Rev. 3 (3), 127–149.

Mao, X., Li, Q., Xie, H., Lau, R.Y., Wang, Z., Paul Smolley, S., 2017. Least squares generative adversarial networks. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2794–2802.

Matsubara, Y., Levorato, M., Restuccia, F., 2022. Split computing and early exiting for deep learning applications: Survey and research challenges. ACM Comput. Surv. 55 (5), 1–30.

Meng, T., Jing, X., Yan, Z., Pedrycz, W., 2020. A survey on machine learning for data fusion. Inf. Fusion 57, 115–129.

Menke, W., 2015. Review of the generalized least squares method. Surv. Geophys. 36 (1), 1–25.

Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E., 1953. Equation of state calculations by fast computing machines. J. Chem. Phys. 21 (6), 1087–1092.

Muir, J.B., 2022. Model Parameterization and Model Selection in Geophysical Inverse Problems: Designing Inverse Problems that Respect a Priori Geophysical Knowledge. California Institute of Technology.

Nabi, M., Senyurek, V., Gurbuz, A.C., Kurum, M., 2022. Deep learning-based soil moisture retrieval in CONUS using CYGNSS delay–doppler maps. IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 15, 6867–6881.

Najafabadi, M.M., Villanustre, F., Khoshgoftaar, T.M., Seliya, N., Wald, R., Muharemagic, E., 2015. Deep learning applications and challenges in big data analytics. J. Big Data 2 (1), 1–21.

Nash, J.E., Sutcliffe, J.V., 1970. River flow forecasting through conceptual models part I: A discussion of principles. J. Hydrol. 10 (3), 282–290.

Natras, R., Soja, B., Schmidt, M., 2022. Ensemble machine learning of random forest, AdaBoost and XGBoost for vertical total electron content forecasting. Remote Sens. 14 (15), 3547.

Nelson, D.M., Pereira, A.C., De Oliveira, R.A., 2017. Stock market's price movement prediction with LSTM neural networks. In: 2017 International Joint Conference on Neural Networks. IJCNN, Ieee, pp. 1419–1426.

Penrose, R., 1955. A generalized inverse for matrices. In: Mathematical Proceedings of the Cambridge Philosophical Society, vol. 51, Cambridge University Press, pp. 406–413.

Pottmann, H., Leopoldseder, S., 2003. A concept for parametric surface fitting which avoids the parametrization problem. Comput. Aided Geom. Design 20 (6), 343–362.

Puranik, T.G., Rodriguez, N., Mavris, D.N., 2020. Towards online prediction of safety-critical landing metrics in aviation using supervised machine learning. Transp. Res. C 120, 102819.

Qian, N., 1999. On the momentum term in gradient descent learning algorithms. Neural Netw. 12 (1), 145–151.

Qiu, S., Zhao, H., Jiang, N., Wang, Z., Liu, L., An, Y., Zhao, H., Miao, X., Liu, R., Fortino, G., 2022. Multi-sensor information fusion based on machine learning for real applications in human activity recognition: State-of-the-art and research challenges. Inf. Fusion 80, 241–265.

Rahaman, M.M., Thakur, B., Kalra, A., Li, R., Maheshwari, P., 2019. Estimating high-resolution groundwater storage from GRACE: A random forest approach. Environments 6 (6), 63.

Raissi, M., Perdikaris, P., Karniadakis, G.E., 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J. Comput. Phys. 378, 686–707.

Ramachandran, P., Zoph, B., Le, Q.V., 2017. Searching for activation functions. arXiv preprint arXiv:1710.05941.

Razin, M.-R.G., Voosoghi, B., 2022. Modeling of precipitable water vapor from GPS observations using machine learning and tomography methods. Adv. Space Res. 69 (7), 2671–2681.

Rodriguez-Galiano, V., Sanchez-Castillo, M., Chica-Olmo, M., Chica-Rivas, M., 2015. Machine learning predictive models for mineral prospectivity: An evaluation of neural networks, random forest, regression trees and support vector machines. Ore Geol. Rev. 71, 804–818.

Ruhe, A., 1979. Accelerated Gauss-Newton algorithms for nonlinear least squares problems. BIT Numer. Math. 19 (3), 356–367.

Rumelhart, D.E., Hinton, G.E., Williams, R.J., 1985. Learning Internal Representations By Error Propagation. Tech. Rep., California Univ San Diego La Jolla Inst for Cognitive Science.

Rumelhart, D.E., Hinton, G.E., Williams, R.J., 1986. Learning representations by back-propagating errors. Nature 323 (6088), 533–536.

Sabzehee, F., Amiri-Simkooei, A., Iran-Pour, S., Vishwakarma, B.D., Kerachian, R., 2023. Enhancing spatial resolution of GRACE-derived groundwater storage anomalies in Urmia catchment using machine learning downscaling methods. J. Environ. Manag. 330, 117180.

Salcedo-Sanz, S., Ghamisi, P., Piles, M., Werner, M., Cuadra, L., Moreno-Martínez, A., Izquierdo-Verdiguier, E., Muñoz-Marí, J., Mosavi, A., Camps-Valls, G., 2020. Machine learning information fusion in earth observation: A comprehensive review of methods, applications and data sources. Inf. Fusion 63, 256–272.

Santos, D., Fernández-Fernández, S., Abreu, T., Silva, P.A., Baptista, P., 2022. Retrieval of nearshore bathymetry from sentinel-1 SAR data in high energetic wave coasts: The Portuguese case study. Remote Sens. Appl.: Soc. Environ. 25, 100674. http://dx.doi.org/10.1016/j.rsase.2021.100674.

Santurkar, S., Tsipras, D., Ilyas, A., Madry, A., 2018. How does batch normalization help optimization?. In: Advances in Neural Information Processing Systems, vol. 31.

Schwegmann, C.P., Kleynhans, W., Salmon, B.P., Mdakane, L.W., Meyer, R.G., 2016. Very deep learning for ship discrimination in synthetic aperture radar imagery. In: 2016 IEEE International Geoscience and Remote Sensing Symposium. IGARSS, IEEE, pp. 104–107.

Selmic, R.R., Lewis, F.L., 2002. Neural-network approximation of piecewise continuous functions: Application to friction compensation. IEEE Trans. Neural Netw. 13 (3), 745–751.

Senyurek, V., Lei, F., Boyd, D., Kurum, M., Gurbuz, A.C., Moorhead, R., 2020. Machine learning-based CYGNSS soil moisture estimates over ISMN sites in CONUS. Remote Sens. 12 (7), 1168.

Shahvandi, M.K., Soja, B., 2022. Inclusion of data uncertainty in machine learning and its application in geodetic data science, with case studies for the prediction of earth orientation parameters and GNSS station coordinate time series. Adv. Space Res. 70 (3), 563–575.

Shokri, A., Walker, J.P., van Dijk, A.I., Pauwels, V.R., 2018. Performance of different ensemble kalman filter structures to assimilate GRACE terrestrial water storage estimates into a high-resolution hydrological model: A synthetic study. Water Resour. Res. 54 (11), 8931–8951.

Singh, P., Singh, S., Paprzycki, M., 2024. Optimised context encoder-based fusion approach with deep learning and nonlinear least square method for pan-sharpening. Int. J. Bio-Inspired Comput. 23 (1), 53–67.

Smola, A.J., Schölkopf, B., 2004. A tutorial on support vector regression. Stat. Comput. 14 (3), 199–222.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., 2014. Dropout: A simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. 15 (1), 1929–1958.

Stracuzzi, D.J., Darling, M.C., Chen, M.G., Peterson, M.G., 2018. Data-driven uncertainty quantification for multisensor analytics. In: Ground/Air Multisensor Interoperability, Integration, and Networking for Persistent ISR IX, vol. 10635, SPIE, pp. 155–167.

Strang, G., 1988. Linear Algebra and Its Application, second ed. Harcourt Brace Jovanovich Publishers, San Diego, USA.

Strogatz, S.H., 2018. Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering. CRC Press.

Sutskever, I., Martens, J., Dahl, G., Hinton, G., 2013. On the importance of initialization and momentum in deep learning. In: International Conference on Machine Learning. PMLR, pp. 1139–1147.

Suzuki, T., Amano, Y., 2021. NLOS multipath classification of GNSS signal correlation output using machine learning. Sensors 21 (7), 2503.

Tarantola, A., 2005. Inverse Problem Theory and Methods for Model Parameter Estimation. SIAM.

Teunissen, P.J.G., 1985. Generalized Inverses, Adjustment, the Datum Problem and S-Transformations. Tech. Rep., In: Optimization and Design of Geodetic Networks, EW Grafarend and F Sanso (Eds).

Teunissen, P.J., 1988a. The non-linear 2D symmetric helmert transformation: An exact non-linear least-squares solution. Bull. Geod. 62 (1), 1–16.

Teunissen, P.J.G., 1988b. Towards a least-squares framework for adjusting and testing of both functional and stochastic model, internal research memo, Geodetic Computing Centre, Delft, a reprint of original 1988 report is also available in 2004, no. 26.

Teunissen, P.J.G., 1990. Nonlinear least-squares. Manuscr. Geod. 15 (3), 137–150.

Teunissen, P.J.G., 2000a. Testing Theory: An Introduction. Delft University Press, Website: http://www.vssd.nl series on Mathematical Geodesy and Positioning.

Teunissen, P.J.G., 2000b. Adjustment Theory: An Introduction. Delft University Press, Website http://www.vssd.nl, series on Mathematical Geodesy and Positioning.

Teunissen, P.J.G., 2001. Dynamic Data Processing: Recursive Least Squares. Delft University Press, Website http://www.vssd.nl, series on Mathematical Geodesy and Positioning.

Teunissen, P.J.G., 2018. Distributional theory for the dia method. J. Geod. 92 (1), 59–80.

Teunissen, P.J.G., Amiri-Simkooei, A.R., 2008. Least-squares variance component estimation. J. Geod. 82 (2), 65–82. http://dx.doi.org/10.1007/s00190-007-0157-x.

Teunissen, P., Knickmeyer, E., 1988. Nonlinearity and least squares. CISM J. 42 (4), 321–330.

Thiyagalingam, J., Shankar, M., Fox, G., Hey, T., 2022. Scientific machine learning benchmarks. Nat. Rev. Phys. 4 (6), 413–420.

Thompson, N.C., Greenewald, K., Lee, K., Manso, G.F., 2020. The computational limits of deep learning. arXiv preprint arXiv:2007.05558.

Tikhonov, A.N., 1963. Solution of incorrectly formulated problems and the regularization method. Soviet Math. 4, 1035–1038.

Torrence, C., Compo, G.P., 1998. A practical guide to wavelet analysis. Bull. Am. Meteorol. Soc. 79 (1), 61–78.

Torres, J.F., Hadjout, D., Sebaa, A., Martínez-Álvarez, F., Troncoso, A., 2021. Deep learning for time series forecasting: a survey. Big Data 9 (1), 3–21.

Viehweg, J., Worthmann, K., Mäder, P., 2023. Parameterizing echo state networks for multi-step time series prediction. Neurocomputing 522, 214–228.

Vilone, G., Longo, L., 2021. Notions of explainability and evaluation approaches for explainable artificial intelligence. Inf. Fusion 76, 89–106.

Von Eschenbach, W.J., 2021. Transparency and the black box problem: Why we do not trust AI. Philos. Technol. 34 (4), 1607–1622.

Wan, L., Zeiler, M., Zhang, S., Le Cun, Y., Fergus, R., 2013. Regularization of neural networks using dropconnect. In: International Conference on Machine Learning. PMLR, pp. 1058–1066.

Wang, J., Li, J., Wang, X., Wang, J., Huang, M., 2021a. Air quality prediction using CT-LSTM. Neural Comput. Appl. 33, 4779–4792.

Wang, X., Liu, X., Chen, L., Hu, H., 2021b. Deep-learning damped least squares method for inverse kinematics of redundant robots. Measurement 171, 108821.

Wang, W., Ma, J., 2024. A review: Applications of machine learning and deep learning in aerospace engineering and aero-engine engineering. Adv. Eng. Innov. 6, 54–72.

Wang, W., Pottmann, H., Liu, Y., 2006. Fitting B-spline curves to point clouds by curvature-based squared distance minimization. ACM Trans. Graphics (ToG) 25 (2), 214–238.

Wang, J., Yang, Z., Jin, L., Deng, J., Chen, F., 2011. Parallel and adaptive surface reconstruction based on implicit PHT-splines. Comput. Aided Geom. Design 28 (8), 463–474.

Wang, L., Zhao, Y., 2017. Unscented transformation with scaled symmetric sampling strategy for precision estimation of total least squares. Studia Geophys. Et Geodaetica 61 (3), 385–411.

Watson-Parris, D., 2021. Machine learning for weather and climate are worlds apart. Phil. Trans. R. Soc. A 379 (2194), 20200098.
Williams, P., 2017. Assessing collaborative learning: Big data, analytics and university futures. Assess. Eval. Higher Educ. 42 (6), 978–989.
Wu, J., Poloczek, M., Wilson, A.G., Frazier, P., 2017. Bayesian optimization with gradients. In: Advances in Neural Information Processing Systems, vol. 30.
Xu, P., Roosta, F., Mahoney, M.W., 2020. Newton-type methods for non-convex optimization under inexact Hessian information. Math. Program. 184 (1), 35–70.
Zeilfelder, F., 2002. Scattered data fitting with bivariate splines. In: Tutorials on Multiresolution in Geometric Modelling. Springer, pp. 243–286.
Zeng, W., Zhang, D., Fang, Y., Wu, J., Huang, J., 2018. Comparison of partial least square regression, support vector machine, and deep-learning techniques for estimating soil salinity from hyperspectral data. J. Appl. Remote Sens. 12 (2), 022204–022204.
Zhang, Z., 2018. Improved adam optimizer for deep neural networks. In: 2018 IEEE/ACM 26th International Symposium on Quality of Service. IWQoS, IEEE, pp. 1–2.
Zhang, C., Bengio, S., Hardt, M., Mozer, M.C., Singer, Y., 2019a. Identity crisis: Memorization and generalization under extreme overparameterization. arXiv preprint arXiv:1902.04698.
Zhang, Y., Le, J., Liao, X., Zheng, F., Li, Y., 2019b. A novel combination forecasting model for wind power integrating least square support vector machine, deep belief network, singular spectrum analysis and locality-sensitive hashing. Energy 168, 558–572.
Zhang, N., Tang, S., Huang, Z., 2024. Short-term regional ionospheric TEC forecast using a hybrid deep learning neural network. Adv. Space Res. 73 (7), 3772–3781.
Zhang, B., Yao, Y., 2021. Precipitable water vapor fusion based on a generalized regression neural network. J. Geod. 95 (3), 1–14.
Zhao, X., Fu, S., Tian, Y., Zhao, K., 2022. Asymmetric and robust loss function driven least squares support vector machine. Knowl.-Based Syst. 258, 109990.