Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 389 (2022) 114378
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

Multi-fidelity regression using artificial neural networks: Efficient approximation of parameter-dependent output quantities

Mengwu Guo[a,*], Andrea Manzoni[b], Maurice Amendt[c], Paolo Conti[b], Jan S. Hesthaven[c]

[a] *Department of Applied Mathematics, University of Twente, The Netherlands*
[b] *MOX – Dipartimento di Matematica, Politecnico di Milano, Italy*
[c] *Institute of Mathematics, École Polytechnique Fédérale de Lausanne, Switzerland*

Received 26 February 2021; received in revised form 20 September 2021; accepted 15 November 2021
Available online 6 December 2021

Abstract

Highly accurate numerical or physical experiments are often very time-consuming or expensive to obtain. When time or budget restrictions prohibit the generation of additional data, the amount of available samples may be too limited to provide satisfactory model results. Multi-fidelity methods deal with such problems by incorporating information from other sources, which are ideally well-correlated with the high-fidelity data, but can be obtained at a lower cost. By leveraging correlations between different data sets, multi-fidelity methods often yield superior generalization when compared to models based solely on a small amount of high-fidelity data. In the current work, we present the use of artificial neural networks applied to multi-fidelity regression problems. By elaborating a few existing approaches, we propose new neural network architectures for multi-fidelity regression. The introduced models are compared against a traditional multi-fidelity regression scheme — co-kriging. A collection of artificial benchmarks are presented to measure the performance of the analyzed models. The results show that cross-validation in combination with Bayesian optimization leads to neural network models that outperform the co-kriging scheme. Additionally, we show an application of multi-fidelity regression to an engineering problem. The propagation of a pressure wave into an acoustic horn with parametrized shape and frequency is considered, and the index of reflection intensity is approximated using the proposed multi-fidelity models. A finite element, full-order model and a reduced-order model built through the reduced basis method are adopted as the high- and low-fidelity, respectively. It is shown that the multi-fidelity neural networks return outputs that achieve a comparable accuracy to those from the expensive, full-order model, using only very few full-order evaluations combined with a larger amount of inaccurate but cheap evaluations of the reduced order model.

Keywords: Machine learning; Artificial neural network; Multi-fidelity regression; Gaussian process regression; Reduced order modeling; Parametrized PDE

1. Introduction

Artificial neural networks (ANNs) have arguably been one of the most active topics during the recent years. They have been successfully applied in a substantial number of research areas, including image recognition [1],

* Corresponding author.
E-mail addresses: m.guo@utwente.nl (M. Guo), andrea1.manzoni@polimi.it (A. Manzoni), paolo.conti@polimi.it (P. Conti), Jan.Hesthaven@epfl.ch (J.S. Hesthaven).

https://doi.org/10.1016/j.cma.2021.114378

translation [2], and fraud detection [3]. More recently, ANNs have also been widely used in the emerging area of machine learning in computational science and engineering, sometimes referred to as *scientific machine learning* [4]. The remarkable expressive power of neural networks (NNs) has made them stand out in the solution of forward and inverse problems governed by partial differential equations (PDEs) [5–7], reduced order modeling [8–10], data-driven discovery [11], multiscale analysis [12], and so on. This success can largely be explained by three major factors: computational power, flexibility and access to large data sets. The great flexibility of NNs, as well as their multi-purpose nature, is a dominant factor explaining their overwhelming success, not only in research, but also in real-world applications. Several application-dependent architectures have been developed, e.g., convolutional neural networks are often used in image related tasks, whereas recurrent neural networks have found their success in speech recognition. Nevertheless, even for a fixed structure, a neural network is still able to adapt to different situations. This feature is mainly explained by the large number of parameters and hyperparameters that can be tuned to fit data in many situations.

The ability to handle large data sets without overwhelming computational costs has made NNs a good candidate for multi-fidelity (MF) regression. MF regression exploits correlations between different data sets to provide a regression model that generalizes better than a simple regression model, which only takes a single data set into account. The MF approach can be included in the more general machine learning framework of *weakly supervised learning* [13,14], in particular in the context of *inexact supervision*, which concerns the situation where supervised information is available, but not sufficiently accurate as required. The most common setup for MF regression is a set of data sources with different fidelity levels. High-fidelity (HF) samples are usually rare due to their cost, be it computational or experimental. However, such data are accurate and are the best available knowledge about the problem. In contrast, low-fidelity (LF) data are assumed to be easy and cheap to obtain, yet might lack accuracy. Often generated numerically, the LF data set ideally expresses major trends of the problem and correlate well with the HF data. The goal of MF regression is to infer trends from LF data and use it to approximate the HF model, especially in regions where HF data are sparse.

A natural MF setting arose in the geostatistics community, who realized that *"if core data at other locations are correlated, they should be included to improve the regression"* [15]. In the context of soil porosity, precise measurements are combined with seismic data to better predict porosity in large areas. Another common MF situation arises when solving PDEs [16,17]. High order numerical schemes with a fine mesh give rise to accurate, HF data, whereas the usage of a coarse mesh, a partially converged solution, or a linearized equation can lead to LF data. In the past decade, MF methods have found applications in many areas of scientific computing, including uncertainty quantification, inference, and optimization. We refer to [18] for a comprehensive review. A widely used MF technique is co-kriging [19,20] which relies on vector-valued Gaussian processes for regression. Such a Gaussian process regression scheme presents two major benefits. Firstly, as a non-parametric regression tool, it is suitable for many different applications. Secondly, the method can be cast in the Bayesian framework, and the regression results naturally include an uncertainty estimation, which is usually desirable. Nevertheless, Gaussian process regression is not suitable for many applications as it suffers from different drawbacks, such as the curse of dimensionality. Consequently, new methods for MF regression are needed, ideally able to detect non-trivial, highly nonlinear correlations between the data sets of different fidelity levels, and should be applicable to a general class of problems.

Based on these considerations, ANNs appear to be a promising candidate to solve MF problems. In the related current work, we consider MF regression with ANNs. Several approaches have been proposed in the literature so far and successfully used NNs in a MF setting. In [21], the authors tested a deep NN structure on different artificial MF benchmarks, extended the idea to the physics-informed NN scheme and applied it to inverse problems governed by partial differential equations (PDEs). Their MF model clearly outperforms a single-fidelity regression. A different NN architecture, incorporated into a Monte Carlo sampling algorithm to estimate uncertainties in a MF setting, was introduced in [22], and reduced computational cost has been observed as compared to traditional Monte Carlo sampling. In addition, different MF strategies for training NNs were discussed in [23], NNs were used to approximate the discrepancy between the HF and LF physics-constrained NNs in [24], and deep neural networks (DNNs) were embedded into co-kriging in [25]. Moreover, a composition of Gaussian processes in a multi-layer network structure was used for MF modeling in [26], and a multi-fidelity Bayesian neural network scheme has been developed in [27] and applied to the physics-informed versions.

In this work, we propose different ANN architectures for the purpose of MF regression. Inspired by [21] and [22], we present two all-in-one models in which different fidelity levels are trained simultaneously, as well as two

multilevel models that define separate NNs for the hierarchy of fidelity levels. In addition, we utilize a strategy based on cross-validation and Bayesian optimization to automatically select the best performing NN hyperparameters. To the best of our knowledge, hyperparameter optimization (HPO) has not been investigated for MF neural networks. The goal of this work is to define a reliable strategy that consistently proposes a neural network which achieves high accuracy and shows good generalization properties. We will assess the performance of the proposed NNs on a set of manufactured benchmarks, all chosen carefully to test for the desirable properties. All models will be compared to single-fidelity regression schemes. Comparisons will also include co-kriging results to benchmark the proposed models against common practices in MF frameworks.

Additionally, we show an application of the proposed MF regression schemes to the evaluation of a quantity of interest defined as a functional of the solution to a parameter-dependent problem governed by PDEs. Such a task often occurs in the applied sciences and engineering, where multiple evaluations of PDE solutions, for different scenarios described in terms of physical or geometrical parameters, can be computationally demanding if relying on high-fidelity, full-order models (FOMs) such as detailed finite element approximations. To overcome this difficulty, low-fidelity, reduced-order models (ROMs) can be built through the reduced basis (RB) method. Despite ROMs featuring much lower-dimensional solution spaces than those of the FOMs, they are able to capture the critical physical features of the FOMs. A reduced model seeks the solutions on a low-dimensional manifold which is approximated by a linear trial subspace spanned by a set of global basis functions, built from a set of full-order snapshots. The ROM accuracy is often granted at the price of a relatively large number of basis functions involved in the reduced-order approximation. On the other hand, an efficient assembly of the ROM during the online stage may only be possible provided that an expensive hyper-reduction is performed during the offline stage. Therefore, a low dimensionality without expensive hyper-reduction can make a reduced model extremely efficient, but potentially inaccurate. Our goal, enabled by the MF neural network schemes in this work, is to provide accurate approximations to the output quantities by leveraging a relatively large number of output evaluations using very low-dimensional ROMs and a small number of evaluations using the FOM, so as to avoid the efficiency issues stemming from the ROM construction and evaluation without compromising the outcome accuracy. In particular, we apply the proposed approaches to a parametrized PDE problem, namely the propagation of a pressure wave into an acoustic horn with parametrized shape and frequency, described by the Helmholtz equation. We assess the impact of both the quality and the amount of the training data on the overall accuracy of the MF regression outcome.

Following the introduction, the concepts of ANNs and Gaussian process regression are briefly reviewed, and their underlying correlation is discussed in Section 2. Several NN structures for MF regression are introduced and discussed in Section 3, and their effectiveness is demonstrated by a series of benchmark test cases in Section 4. An application to a parametrized PDE problem is presented in Section 5, and conclusions are drawn in Section 6.

2. Artificial neural networks and Gaussian processes for regression

2.1. Artificial neural networks (ANNs)

In this section, we consider an L-hidden-layer fully-connected NN [28] with width M_l and nonlinear activation function ϕ for the lth layer, $1 \le l \le L$. At the jth neuron in the lth layer of the NN, the pre- and post-activation are denoted by z_j^l and x_j^l, respectively, $1 \le i \le M_l$. Let $\mathbf{x} = \mathbf{x}^0 \in \mathbb{R}^{d_{\text{in}}}$ denote the inputs of the network and $\mathbf{y} = \mathbf{z}^{L+1} \in \mathbb{R}^{d_{\text{out}}}$ denote the outputs. Note that we have $M_0 = d_{\text{in}}$ and $M_{L+1} = d_{\text{out}}$. Weight and bias parameters between the $(l-1)$th and lth layers are represented by W_{ij}^l and b_i^l, respectively, $1 \le l \le (L+1)$, $1 \le i \le M_l$, $1 \le j \le M_{l-1}$. Then one has

$$
\begin{aligned}
&z_i^l(\mathbf{x}) = b_i^l + \sum_{j=1}^{M_{l-1}} W_{ij}^l x_j^{l-1}(\mathbf{x}), \quad x_i^l(\mathbf{x}) = \phi(z_i^l(\mathbf{x})), \quad 1 \le i \le M_l,\ 1 \le l \le L, \quad \text{and} \\
&y_i(\mathbf{x}) = z_i^{L+1}(\mathbf{x}) = b_i^{L+1} + \sum_{j=1}^{M_L} W_{ij}^{L+1} x_j^L(\mathbf{x}), \qquad 1 \le i \le d_{\text{out}}.
\end{aligned}
\tag{1}
$$

A multivariate function $\mathbf{y} = \mathbf{f}(\mathbf{x})$ is approximated by a vector-valued network surrogate $\mathbf{f}^{\text{NN}}(\cdot; \mathbf{W}, \mathbf{b}) : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{out}}}$ to be trained on the input–output pairs $\{(\mathbf{x}^{(k)}, \mathbf{y}^{(k)})\}_{k=1}^N$. Here $\mathbf{W}$ and $\mathbf{b}$ are the vectors collecting all the weight and

bias parameters, respectively, and N is the number of data pairs. Such a training is often performed by minimizing a cost function:

$$(\mathbf{W}, \mathbf{b}) = \arg\min_{\mathbf{W},\mathbf{b}} \left\{ \frac{1}{N} \sum_{k=1}^{N} \|\mathbf{y}^{(k)} - \mathbf{f}^{\mathrm{NN}}(\mathbf{x}^{(k)}; \mathbf{W}, \mathbf{b})\|_2^2 + \lambda \|\mathbf{W}\|_2^2 \right\}, \tag{2}$$

in which the first term is the mean squared error (MSE) and the second term is a regularization term with $\lambda \geq 0$ being the penalty coefficient.

2.2. GaussIan process regression (GPR)

Single-fidelity GPR

A Gaussian process (GP) is a collection of random variables, any finite number of which obey a joint Gaussian distribution. In the GPR, the prior on the scalar-valued regression function $f : \mathbb{R}^{d_{\mathrm{in}}} \to \mathbb{R}$ is assumed to be a GP corrupted by an independent Gaussian noise term, i.e., for $(\mathbf{x}, \mathbf{x}') \in \mathbb{R}^{d_{\mathrm{in}}} \times \mathbb{R}^{d_{\mathrm{in}}}$,

$$f(\mathbf{x}) \sim \mathrm{GP}(0, \kappa(\mathbf{x}, \mathbf{x}')), \quad y = f(\mathbf{x}) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \chi^2), \tag{3}$$

where χ is the standard deviation of a Gaussian noise term ϵ, and the positive semidefinite kernel function κ gives the covariance of the prior GP.

Given N pairs of input–output training data, a prior joint Gaussian is defined for the corresponding outputs:

$$\boldsymbol{y}|\boldsymbol{X} \ \sim \ \mathcal{N}(\mathbf{0}, \mathbf{K}_y), \quad \mathbf{K}_y = \mathrm{Cov}[\boldsymbol{y}|\boldsymbol{X}] = \kappa(\boldsymbol{X}, \boldsymbol{X}) + \chi^2 \mathbf{I}_N, \tag{4}$$

where $\boldsymbol{y} = \{y^{(1)}, y^{(2)}, \ldots, y^{(N)}\}^{\mathrm{T}}$, $\boldsymbol{X} = [\ \mathbf{x}^{(1)} \mid \mathbf{x}^{(2)} \mid \cdots \mid \mathbf{x}^{(N)}\ ]$ and $\mathbf{I}_N$ is the M-dimensional unit matrix, N being the number of training samples.

From a regression model, the goal is to predict the noise-free output $f^*(\mathbf{s})$ for a new test input $\mathbf{s} \in \mathbb{R}^{d_{\mathrm{in}}}$. By the standard rules for conditioning Gaussians, the posterior predictive distribution conditioning on the training data is obtained as a new GP:

$$\begin{aligned} &f^*(\mathbf{s})|\ \mathbf{s}, \boldsymbol{X}, \boldsymbol{y} \ \sim \ \mathrm{GP}(m^*(\mathbf{s}), c^*(\mathbf{s}, \mathbf{s}')), \\ &m^*(\mathbf{s}) = \kappa(\mathbf{s}, \boldsymbol{X}) \mathbf{K}_y^{-1} \boldsymbol{y}, \quad c^*(\mathbf{s}, \mathbf{s}') = \kappa(\mathbf{s}, \mathbf{s}') - \kappa(\mathbf{s}, \boldsymbol{X}) \mathbf{K}_y^{-1} \kappa(\boldsymbol{X}, \mathbf{s}'). \end{aligned} \tag{5}$$

Multi-fidelity GPR

GPR with training data from different fidelity levels is known as co-kriging [19] or vector-valued GPR [20]. In such a regression scheme, one can use a large amount of LF data and only a limited number of HF samples to train a model of a reasonable accuracy. Since the LF evaluations are cheap, the cost of training data preparation can be reduced by controlling the number of HF evaluations. Assuming a linear correlation between the different fidelity levels, we can employ the linear model of coregionalization (LMC) [20] that expresses the prior of a hierarchy of D solution fidelities as

$$f_i(\mathbf{x}) = \sum_{j=1}^{D} a_{i,j} u_j(\mathbf{x}), \quad i = 1, 2, \ldots, D, \tag{6}$$

i.e., each level of solution f_i is written as a linear combination of D independent Gaussian processes $u_j \sim \mathrm{GP}(0, \kappa_j(\cdot, \cdot))$. In addition, the vector $\mathbf{a}_j$, $1 \leq j \leq D$, collects the weights of the corresponding GP component u_j, i.e., $\mathbf{a}_j = \{a_{1,j}, \ldots, a_{D,j}\}^{\mathrm{T}}$. This formulation leads to a matrix-valued kernel for the MF GPR model as

$$\boldsymbol{\mathcal{K}}(\mathbf{x}, \mathbf{x}') = \sum_{j=1}^{D} \mathbf{a}_j \mathbf{a}_j^{\mathrm{T}} \kappa_j(\mathbf{x}, \mathbf{x}'). \tag{7}$$

In the two-level case, the well-known form of AR(1)-cokriging [19] is a special form of the linear model defining the prior of a LF solution f_L and a HF f_H:

$$\begin{aligned} f_H(\mathbf{x}) &= \rho u_1(\mathbf{x}) + u_2(\mathbf{x}), \\ f_L(\mathbf{x}) &= u_1(\mathbf{x}), \end{aligned} \tag{8}$$

in which $\mathbf{a}_1 = \{\rho, 1\}^{\mathrm{T}}$ and $\mathbf{a}_2 = \{1, 0\}^{\mathrm{T}}$. Conditioning on the training input–output pairs from both the LF and HF, denoted by $(\boldsymbol{X}_L, \boldsymbol{y}_L)$ and $(\boldsymbol{X}_H, \boldsymbol{y}_H)$, respectively, the predictive distribution for the HF can be expressed as a posterior GP, i.e., $f_H^*(\mathbf{s})|\ \mathbf{s}, \boldsymbol{X}_H, \boldsymbol{y}_H, \boldsymbol{X}_L, \boldsymbol{y}_L \sim \mathrm{GP}$.

2.3. The link between ANNs and GPR

It can be shown that the prior of a neural network output can be seen as a set of Gaussian processes under the following probabilistic assumptions [29,30][1]: (I) All the weight parameters W_{ij}^l's are independent and identically distributed (i.i.d), as are all the bias parameters b_i^l's, and the weight and bias parameter sets are independent of each other; (II) In the lth layer, $1 \leq l \leq L+1$, $b_i^l \sim \mathcal{N}(0, \sigma_b^2)$, and W_{ij}^l's are independently drawn from any distribution with zero mean and variance σ_w^2/M_{l-1}; and (III) $M_l \to \infty$, $1 \leq l \leq L$. Here we briefly show by induction that $\{z_i^l : 1 \leq i \leq M_l\}$ are i.i.d. zero-mean Gaussian processes and $\{x_j^l : 1 \leq j \leq M_l\}$ are i.i.d. for all $2 \leq l \leq L+1$.

We consider an arbitrary set of finite locations of the input $\mathbf{x}$, denoted by $\boldsymbol{X}$. Since both $\{b_i^1 : \forall i\}$ and $\{W_{i,j}^1 : \forall i, j\}$ are i.i.d., one obtains $\{z_i^1(\boldsymbol{X}) = b_i^1 + \sum_{j=1}^{d_{\text{in}}} W_{ij}^1 x_j(\boldsymbol{X}) : \forall i\}$ are i.i.d. Thus $\{x_j^1(\boldsymbol{X}) = \phi(z_j^1(\boldsymbol{X})) : \forall j\}$ are i.i.d, naturally leading to $\{z_i^2(\boldsymbol{X}) = b_i^2 + \sum_{j=1}^{M_1} W_{ij}^2 x_j^1(\boldsymbol{X}) : \forall i\}$ being i.i.d. For each $1 \leq i \leq M_2$, we apply the multivariate central limit theorem [31] to a sequence of i.i.d. random vectors $\{\sqrt{M_1} W_{i1}^2 x_1^1(\boldsymbol{X}), \sqrt{M_1} W_{i2}^2 x_2^1(\boldsymbol{X}), \ldots\}$, all with zero mean and covariance $\sigma_w^2 \text{Cov}[x_\cdot^1(\boldsymbol{X})]$, and we obtain that $z_i^2(\boldsymbol{X}) \sim \mathcal{N}(\mathbf{0}, \sigma_b^2 \mathbf{I} + \sigma_w^2 \mathbb{E}[x_\cdot^1(\boldsymbol{X}) \otimes x_\cdot^1(\boldsymbol{X})])$ as $M_1 \to \infty$. Since the input locations $\boldsymbol{X}$ are arbitrary, each $z_i^2(\mathbf{x})$ is a Gaussian process as $z_i^2(\cdot) \sim \text{GP}(0, K^2(\cdot, \cdot))$ with its kernel K^2 defined as $K^2(\mathbf{x}, \mathbf{x}') = \sigma_b^2 \mathbf{I} + \sigma_w^2 \mathbb{E}[\phi(z_\cdot^1(\mathbf{x})) \phi(z_\cdot^1(\mathbf{x}'))]$. After the nonlinear activation in the 2nd layer, we have the i.i.d. $\{x_j^2 : \forall j\}$. Therefore the proposition holds true for $l = 2$.

Assume it holds true for l, $2 \leq l \leq L$. The proposition can be verified to be true for $l+1$, similarly to that for $l = 2$. The pre-activation in each intermediate layer follows a Gaussian process $z_\cdot^l \sim \text{GP}(0, K^l(\cdot, \cdot))$, $2 \leq l \leq L+1$, and

$$K^l(\mathbf{x}, \mathbf{x}') = \sigma_b^2 + \sigma_w^2 \mathbb{E}_{z_\cdot^{l-1} \sim \text{GP}(0, K^{l-1})}[\phi(z_\cdot^{l-1}(\mathbf{x})) \phi(z_\cdot^{l-1}(\mathbf{x}'))] \,. \tag{9}$$

The outputs $\mathbf{y}(\mathbf{x}) = \mathbf{z}^{L+1}(\mathbf{x})$ thus follow i.i.d. Gaussian process priors and the kernel function can be formed from the recursive relation (9). We refer the readers to [32] for a more general discussion.

3. Artificial neural networks for multi-fidelity regression

Currently, there have been two successful approaches [21,22] to the use of NNs in a multi-fidelity context. The neural networks for multi-fidelity regression (NNMFR) presented in these two studies show inherent differences at the architectural level of the networks. Whereas [21] uses a single NN to perform a simultaneous regression of both the HF and LF data, [22] splits them up and used two distinct ANNs, one for each fidelity level. Note that in the present work, we restrict ourselves to bi-fidelity problems, i.e., we are interested in approximating the scalar HF function $f_{\texttt{HF}}(\mathbf{x})$ by incorporating information from the scalar LF function $f_{\texttt{LF}}(\mathbf{x})$. Consequently, there will generally be only a few available observations of the HF function, whereas the LF data are abundant. In that respect, it is useful to introduce the following notation:

- $N_{\texttt{HF}}$ and $N_{\texttt{LF}}$ denote the numbers of HF and LF samples, respectively.
- The training set for the HF function is given by $\mathcal{T}_{\texttt{HF}} = \{(\mathbf{x}_{\texttt{HF}}^{(i)}, y_{\texttt{HF}}^{(i)}) : 1 \leq i \leq N_{\texttt{HF}}\}$, which satisfies the HF function $y_{\texttt{HF}} = f_{\texttt{HF}}(\mathbf{x})$.
- By replacing the subscripts HF in the above notations by LF, we obtain the LF counterparts.
- The NN approximations corresponding to the HF and LF functions are denoted by $f_{\texttt{HF}}^{\texttt{NN}}(\mathbf{x})$ and $f_{\texttt{LF}}^{\texttt{NN}}(\mathbf{x})$, respectively.

In the current section, we will review existing NNMFR approaches and propose our own strategies by elaborating and adapting existing ideas. Different from the existing NNMFR strategies, our all-in-one models consider the LF outputs as latent variables of the HF surrogate or mimic the correlation between HF and LF levels in co-kriging, and our multilevel models are formulated directly from the hierarchy of fidelity levels.

3.1. All-in-one NNMFR

As previously noted, a single NN, which performs two simultaneous regressions on the HF and LF data, is used in [21]. It essentially resembles the structure of an autoencoder, where the inputs are encoded towards the LF output

[1] The discussion here involves some modification from the work in [29,30].

y_{LF}, then decoded and encoded again for the HF output y_{HF}. The setup relies on the assumption that

$$f_{\mathrm{HF}}(\mathbf{x}) = \mathcal{F}(\mathbf{x}, y_{\mathrm{LF}}) = \alpha\mathcal{F}_l(\mathbf{x}, y_{\mathrm{LF}}) + (1-\alpha)\mathcal{F}_{nl}(\mathbf{x}, y_{\mathrm{LF}}), \quad \alpha \in [0, 1], \tag{10}$$

i.e., the unknown mapping $\mathcal{F}$ of the LF to the HF data can be decomposed into a linear and a nonlinear part, denoted by $\mathcal{F}_l$ and $\mathcal{F}_{nl}$, respectively.

In (10), the hyperparameter α determines the strength of the linear correlation, with $\alpha = 1$ corresponding to a fully linear relation between the HF and LF outputs. The determination of the value of α was not specified in [21], neither are any values indicated in the reported results.

Based on these considerations, we propose the following two modified NN architectures, see (a) and (b) in Fig. 1:

1. **"Intermediate" model**: The first NN structure is similar to the one in [21], except that the same input layer is used for HF and LF data and we omit the autoencoder resemblance by adding additional nodes to the layer containing the LF output. Furthermore, we do not impose (10) as we seek to directly model the function $\mathcal{F}(\mathbf{x}, y_{\mathrm{LF}})$. We refer to this NN as the "Intermediate" model, referring to the location of the LF output. In the numerical experiments, the LF output is situated in the 3rd hidden layer of an Intermediate network and the HF output in the last layer. The number of neurons in the first 3 layers is fixed to 64 each and the width and depth of the layers between the LF and HF outputs are determined by hyperparameter optimization. Except for the output layer, we use the hyperbolic tangent activation function.
2. **"GPmimic" model**: The second NNMFR is based on the similarity between a wide NN and a GP. Based on this correspondence, and since Gaussian processes are commonly used for MF regression, we implement an architecture, which seeks to mimic the action of a GP. Hereafter we refer to this architecture as "GPmimic". In contrast to NNs, the conventional GPR often suffers from the curse of dimensionality — the computational cost is rather expensive with high-dimensional inputs and large data sets but the accuracy can hardly be guaranteed, which is usually considered as one of the GPR's major drawbacks. Hence, the GPmimic NN indirectly enables GP-like regression even in the case of large data sets and high-dimensional inputs. The NN architecture is depicted in Fig. 1. The neurons labeled by u_1 and u_2 are defined to be analogous to two independent Gaussian processes, especially when the previous layers are wide, and they play the same roles as in the vector-valued GPR (6). By considering no nonlinear activation in the output layer, the NN outputs of the GPmimic model are given as:

$$\begin{aligned} y_{\mathrm{HF}}(\mathbf{x}) &= W_{11}u_1(\mathbf{x}) + W_{12}u_2(\mathbf{x}) + b_1 \\ y_{\mathrm{LF}}(\mathbf{x}) &= W_{21}u_1(\mathbf{x}) + W_{22}u_2(\mathbf{x}) + b_2. \end{aligned} \tag{11}$$

   By inspecting (11), we notice that the outputs are recovered as an affine transformation of the two variables $u_1(\mathbf{x})$ and $u_2(\mathbf{x})$, and the parameters W_{ij}'s define the correlations between y_{HF} and y_{LF} as in the vector-valued GPR. In principle, this setting only allows to model linear correlations between the HF and LF data.

Remember that the all-in-one architectures perform vector-valued learning of $\mathbf{f} = [f_{\mathrm{HF}}(\mathbf{x}), f_{\mathrm{LF}}(\mathbf{x})]^{\mathrm{T}}$. There are thus two different error components, one for each fidelity level. More specifically, these components are given by the training errors $\mathrm{MSE}_{\mathrm{HF}}$ and $\mathrm{MSE}_{\mathrm{LF}}$ of the HF and LF models, defined as

$$\mathrm{MSE}_{\mathrm{HF}} = \frac{1}{N_{\mathrm{HF}}}\sum_{i=1}^{N_{\mathrm{HF}}} |y_{\mathrm{HF}}^{(i)} - f_{\mathrm{HF}}^{\mathrm{NN}}(\mathbf{x}_{\mathrm{HF}}^{(i)})|^2, \quad \text{and} \quad \mathrm{MSE}_{\mathrm{LF}} = \frac{1}{N_{\mathrm{LF}}}\sum_{i=1}^{N_{\mathrm{LF}}} |y_{\mathrm{LF}}^{(i)} - f_{\mathrm{LF}}^{\mathrm{NN}}(\mathbf{x}_{\mathrm{LF}}^{(i)})|^2. \tag{12}$$

Hence, training an all-in-one network involves a multiobjective minimization problem, which is generally not easy to solve. In [21], the problem is reduced to a single optimization problem by considering a loss function given by the direct addition of the MSEs related to the two fidelity sets and a L_2-regularization term.

A more general approach in multiobjective optimization is to use a weighted sum instead, to account for objectives of different scales. Even though the data sets can be initially scaled to ensure the same order of magnitude, large discrepancies between $\mathrm{MSE}_{\mathrm{HF}}$ and $\mathrm{MSE}_{\mathrm{LF}}$ could develop during the learning process. Based on these considerations, the loss function in the Intermediate and GPmimic models is given as

$$\mathcal{L} = \alpha\mathrm{MSE}_{\mathrm{HF}} + (1-\alpha)\mathrm{MSE}_{\mathrm{LF}} + \lambda\|\mathbf{W}\|_2^2, \tag{13}$$

where $\alpha \in [0, 1]$ (unrelated to the one in (10)) acts as a scaling factor between the two fidelity levels and $\lambda > 0$ is a penalty parameter. The extreme cases $\alpha = 0$ and $\alpha = 1$ correspond to the single-fidelity regressions for the

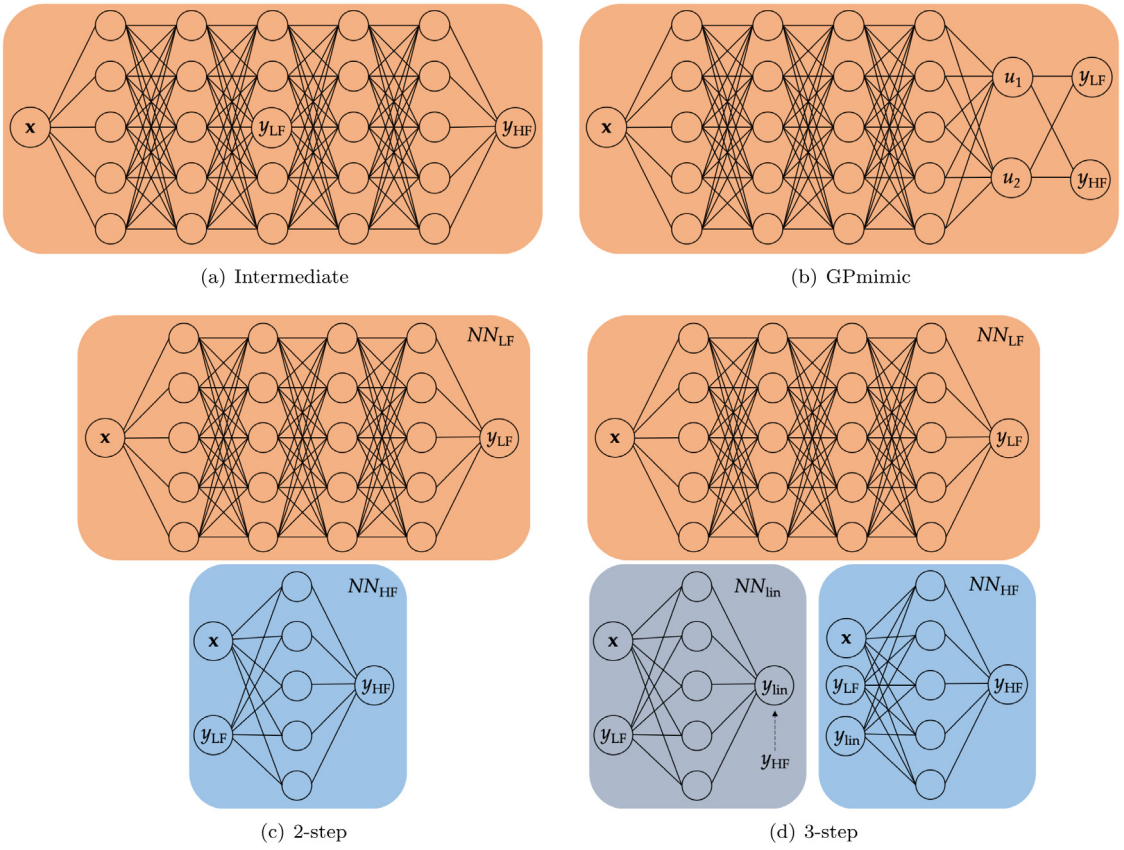


Fig. 1. Different ANNs proposed for MF regression. All-in-one models in the top row (a, b), and multilevel models in the bottom row (c, d). The "Intermediate" model in (a) considers the LF outputs at intermediate latent variables of the surrogate for HF. The "GPmimic" model in (b) employs the LMC to mimic MF GPR. The "2-step" and "3-step" models in (c) and (d) define separate NNs for the hierarchy of fidelity levels.

LF and HF data, respectively. The choice $\alpha = 0.5$ represents a balanced importance of the LF and HF levels. The value of α will be tuned by hyperparameter optimization in the numerical examples. The α values minimize the testing errors of k-fold cross-validation and are determined by Bayesian optimization.

3.2. Multilevel NNMFR

The NNMFR introduced in Section 3.1 relies on a single NN to learn the multidimensional function $\mathbf{f} = [f_{\mathrm{HF}}(\mathbf{x}), f_{\mathrm{LF}}(\mathbf{x})]^{\mathrm{T}}$. Another approach, leading to the multilevel NNMFR, is to use distinct NNs to model these functions. Such an approach has been employed in [22], and the presented method can be summarized as follows. A first NN NN_1 learns a correlation function $y_{\mathrm{HF}} = \mathcal{F}(\mathbf{x}, y_{\mathrm{LF}})$ based on the input data $\left\{(\mathbf{x}_{\mathrm{HF}}^{(i)}, y_{\mathrm{LF}}(\mathbf{x}_{\mathrm{HF}}^{(i)})) : 1 \le i \le N_{\mathrm{HF}}\right\}$ and the output data $\left\{y_{\mathrm{HF}}^{(i)} : 1 \le i \le N_{\mathrm{HF}}\right\}$. In this case, it is required that the HF input locations should form a subset of the LF input locations, i.e., $\mathcal{X}_{\mathrm{HF}} = \{\mathbf{x}_{\mathrm{HF}}^{(i)} : 1 \le i \le N_{\mathrm{HF}}\} \subseteq \mathcal{X}_{\mathrm{LF}} = \{\mathbf{x}_{\mathrm{LF}}^{(i)} : 1 \le i \le N_{\mathrm{LF}}\}$. For each available LF sample $\mathbf{x}' \in \mathcal{X}_{\mathrm{LF}} \setminus \mathcal{X}_{\mathrm{HF}}$, an approximate HF data sample $y'_{\mathrm{HF}}(\mathbf{x}')$ can be generated by the network NN_1, and these generated data, together with the existing HF data at $\mathcal{X}_{\mathrm{HF}}$, can be used as the new HF data set $\mathcal{Y}'_{\mathrm{HF}} = \{y_{\mathrm{HF}}(\mathcal{X}_{\mathrm{HF}}), y'_{\mathrm{HF}}(\mathcal{X}_{\mathrm{LF}} \setminus \mathcal{X}_{\mathrm{HF}})\}$. Finally, a second NN NN_2 is trained to model the HF function $f_{\mathrm{HF}}(\mathbf{x})$ based the input–output pairs between $\mathcal{X}_{\mathrm{LF}}$ and $\mathcal{Y}'_{\mathrm{HF}}$. Following this multilevel approach in [22], we propose two modified multi-step NNs for MF regression by adopting the following major changes:

- The LF function $f_{\mathrm{LF}}(\mathbf{x})$ is modeled by a first NN NN_{LF}. This modification is relevant if the computational cost of additional LF data is not cheap, for instance when having to solve PDEs.
- Now that we can rapidly generate new LF data using NN_{LF}, modeling the direct mapping between $\mathbf{x}$ and $y_{\mathrm{HF}} = f_{\mathrm{HF}}(\mathbf{x})$ is unnecessary. Hence our second network NN_{HF} will be analogous to NN_1 in [22].

These considerations lead to the following multilevel architectures, also see (c) and (d) in Fig. 1:

1. **"2-step" model**: A DNN NN_{LF} is trained on $\mathcal{T}_{\mathrm{LF}}$ to learn the LF function $f_{\mathrm{LF}}(\mathbf{x})$. Using the NN NN_{LF}, we can predict the values of the LF function at the training inputs $\mathcal{X}_{\mathrm{HF}}$ of the HF data, denoted by $f_{\mathrm{LF}}^{\mathrm{NN}}(\mathcal{X}_{\mathrm{HF}}) = \{f_{\mathrm{LF}}^{\mathrm{NN}}(\mathbf{x}_{\mathrm{HF}}^{(i)}) : 1 \le i \le N_{\mathrm{HF}}\}$. Then a second artificial network NN_{HF} approximates the HF function $y_{\mathrm{HF}} = \mathcal{F}(\mathbf{x}, y_{\mathrm{LF}})$ based on the input data $(\mathcal{X}_{\mathrm{HF}}, f_{\mathrm{LF}}^{\mathrm{NN}}(\mathcal{X}_{\mathrm{HF}})) = \{(\mathbf{x}_{\mathrm{HF}}^{(i)}, f_{\mathrm{LF}}^{\mathrm{NN}}(\mathbf{x}_{\mathrm{HF}}^{(i)})) : 1 \le i \le N_{\mathrm{HF}}\}$ and the available HF output data $\mathcal{Y}_{\mathrm{HF}} = y_{\mathrm{HF}}(\mathcal{X}_{\mathrm{HF}}) = \{y_{\mathrm{HF}}^{(i)} : 1 \le i \le N_{\mathrm{HF}}\}$. NN_{HF} is a shallow NN consisting of a single hidden layer. In other words, the network NN_{HF} approximates the function $y_{\mathrm{HF}} = \mathcal{F}(\mathbf{x}, y_{\mathrm{LF}})$ based on the HF and LF data at the same locations $\mathcal{X}_{\mathrm{HF}}$. However, as the HF and LF data locations, i.e., $\mathcal{X}_{\mathrm{HF}}$ and $\mathcal{X}_{\mathrm{LF}}$, are generated independently, the observations of the LF function at the HF inputs $\mathcal{X}_{\mathrm{HF}}$ have to be evaluated from NN_{LF} if not directly available at $\mathcal{X}_{\mathrm{HF}}$.
2. **"3-step" model**: This model is a modification of the 2-step model by adding an additional level of fidelity, generated by a third network NN_{lin}. NN_{lin} is equivalent to the NN_{HF} in the 2-step model in the sense that it models the correlation function $y_{\mathrm{HF}} = \mathcal{F}(\mathbf{x}, y_{\mathrm{LF}})$. However, no nonlinear activation function is used in NN_{lin}. In other words, NN_{lin} is responsible for capturing the linear correlations between the HF and LF data sets. Let $f_{\mathrm{lin}}^{\mathrm{NN}}\left(\mathcal{X}_{\mathrm{HF}}, f_{\mathrm{LF}}^{\mathrm{NN}}(\mathcal{X}_{\mathrm{HF}})\right)$ denote the set of outputs of NN_{lin} at the HF input locations $\mathcal{X}_{\mathrm{HF}}$. This set will then serve as part of the inputs for the third and final network NN_{HF}. NN_{HF} approximates the correlation function $y_{\mathrm{HF}} = \mathcal{F}'(\mathbf{x}, y_{\mathrm{LF}}, y_{\mathrm{lin}})$, y_{lin} being the output of NN_{lin} as a linear approximation of y_{HF}, and the training of NN_{HF} is based on the input–output pairs between $\left(\mathcal{X}_{\mathrm{HF}}, f_{\mathrm{LF}}^{\mathrm{NN}}(\mathcal{X}_{\mathrm{HF}}), f_{\mathrm{lin}}^{\mathrm{NN}}\left(\mathcal{X}_{\mathrm{HF}}, f_{\mathrm{LF}}^{\mathrm{NN}}(\mathcal{X}_{\mathrm{HF}})\right)\right)$ and $\mathcal{Y}_{\mathrm{HF}}$. NN_{HF} is again a shallow NN consisting of a single hidden layer. Note that the 3-step model should only present an advantage over the 2-step model when there exist strong linear correlations between the HF and LF data.

It should be clear that the proposed architectures present a few differences. The key difference between the all-in-one and the multilevel strategies is at the structural level. All-in-one NNMFR aims to approximate the two-dimensional function $\mathbf{f} = [f_{\mathrm{HF}}(\mathbf{x}), f_{\mathrm{LF}}(\mathbf{x})]^{\mathrm{T}}$ with a single NN, while multilevel NNMFR uses distinct networks to model the HF and LF functions separately. However, to incorporate information from the LF function, this is done sequentially by first modeling $f_{\mathrm{LF}}(\mathbf{x})$ and then the correlation between the two fidelity levels.

We note that all-in-one models include the additional hyperparameter α. This hyperparameter can potentially be used to incorporate prior knowledge into the regression model. Suppose for instance that we know that the LF data provide a good representation of the problem. We can then enforce the NN to accurately model the LF data by setting the value of α accordingly. Furthermore, this can be taken into account in the Intermediate model by choosing the number of neurons in the layer containing the LF output, where a smaller layer width may indicate a stronger dependency on the LF data. In the multilevel architectures, the question of how much trust is put in the LF data is left to the model and determined during the training process. An advantage of the multilevel approaches is that the LF function does not have to be approximated by a neural network. Instead, we can use other techniques which might be more accurate or less time-consuming. For instance, in the presence of discontinuities, an accurate approximation can be obtained by utilizing gradient boosting algorithms [33].

It is worth pointing out that the proposed multi-fidelity NN models can be naturally extended to the cases with more than two fidelity levels. Extra LF latent variables can be added to the intermediate layer in an *Intermediate* network, while the last two layers of a *GPmimic* network can be modified according to the LMC formulation (6) with $D > 2$. With added NN steps, the multilevel models will be able to approximate a hierarchy of more fidelity levels as well.

Finally, in the non-hierarchical cases where the available information sources present similar levels of fidelities, the GPmimic model is the only NNMFR that can be used without any adaptation. In fact, all other proposed structures rely on a hierarchy of the available data sets according to their fidelity levels.

4. Numerical results (I): benchmark test cases

In this section, we analyze the performance of the proposed ANN structures in MF regression problems. For that purpose, the models presented in Section 3 are tested on a variety of artificial benchmarks, and compared against commonly used co-kriging methods.

In all benchmarks, the hyperparameters of the neural networks, for instance the parameter α to balance the MSE terms, are optimized by k-fold cross-validation and Bayesian optimization, whereas the hyperparameters of the GPs are obtained by maximizing the marginal likelihood [34]. For the Bayesian optimization, Python package Hyperopt [35] is used for a tree-structured parzen estimator (TPE) based approach [36], in which the priors on hyperparameters can be chosen from a range of distributions for both continuous and discrete random variables. Both the single-fidelity and multi-fidelity GPRs are implemented by the Python package GPy [37]. Furthermore, GPy automatically adds a noise term for each fidelity level to account for noisy data. As artificial benchmarks give rise to noiseless observations, the standard deviation values of all the noise terms are fixed to 10^{-5} in these benchmarks. HPO results for the first three benchmarks' NNMFR can be found in Appendix A.

The numerical study in this section compares the performance of the proposed NNs against co-kriging. In each case, the qualitative difference between single- and multi-fidelity regressions is noted briefly before presenting a comparative study between NNMFR and multi-fidelity GPR. Models are evaluated using the MSE on a test set covering the whole input domain. The results will reveal whether choosing a model based on the validation error is a good strategy. The comparison between NN regression and the best GPR will not only include accuracy results but also computational time. In addition, we evaluate the following R^2 score that allows for a cross-benchmark comparison:

$$R^2 = 1 - \frac{\sum_{i=1}^{N_{\texttt{test}}} \left(y_{\texttt{HF}}^{(i)} - f_{\texttt{reg}}^{(i)} \right)^2}{\sum_{i=1}^{N_{\texttt{test}}} \left(y_{\texttt{HF}}^{(i)} - \bar{y}_{\texttt{HF}} \right)^2}, \qquad \text{where } \bar{y}_{\texttt{HF}} = \frac{1}{N_{\texttt{test}}} \sum_{i=1}^{N_{\texttt{test}}} y_{\texttt{HF}}^{(i)}, \tag{14}$$

$N_{\texttt{text}}$ is the size of the test set, and $f_{\texttt{reg}}$ denotes the prediction given by a regression model.

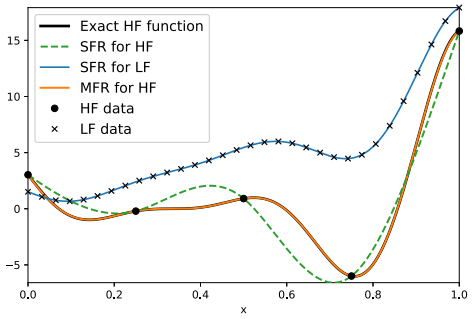
4.1. Benchmark case 1: Linear correlation

The first benchmark is a common test case for MF methods. HF and LF functions are defined over $\Omega = [0, 1]$ as

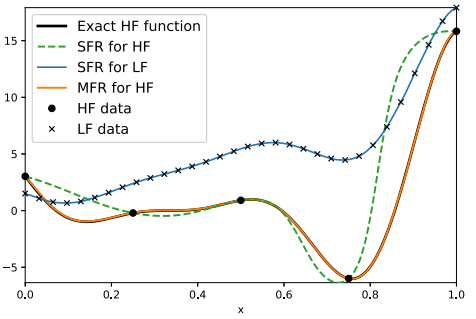
$$f_{\texttt{HF}}(x) = (6x - 2)^2 \sin(12x - 4),$$
$$f_{\texttt{LF}}(x) = 0.5 f_{\texttt{HF}}(x) + 10(x - 0.5) + 5,$$

respectively. The LF function is obtained by a linear transformation from the HF function and the input variable x. The setup serves as a good initial test for MF models. HF and LF samples are given by 5 and 32 equally spaced values in Ω, respectively. Since the number of the HF data is very limited, hyperparameters of the NNs are optimized using leave-one-out cross-validation (LOOCV). The values can be found in Table A.1. Fig. 2 shows the results of both the single-fidelity regression (SFR) and MF regression (MFR). Both the NN and the GPR fail to approximate the function based solely on the HF data; in particular, both models fail to accurately predict the function in the interval $[0, 0.8]$. It is also important to note that the kernel used for the GPR is induced by an NN structure [29]. In fact, many other kernels predict an almost constant mean function with large uncertainty intervals. Table 1 provides quantitative insight in the regression results. No validation error is available for co-kriging, as the hyperparameters are optimized by maximizing the marginal likelihood. However, since the best performing kernel has been chosen based on the MSE on the test set, it is fair to assume that the performance of GPR is slightly overestimated in general.

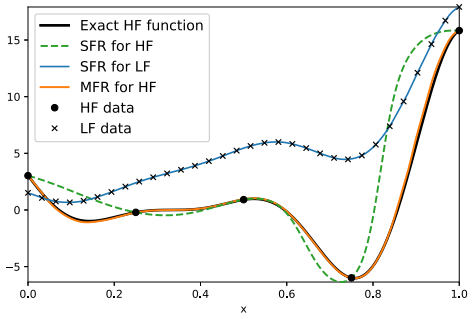
In general, except for the 2-step model, the validation errors are very conservative estimates of the test error. Co-kriging and the 3-step model outperform all other models by one or even two orders of magnitude. This can be explained by their mathematical setup that is designed to exactly model the linear correlation between the HF and LF functions. In fact, the 3rd neural network $NN_{\texttt{HF}}$ of the 3-step model is redundant in this case, as there is no nonlinear trend to catch. Consequently, in this first and simple benchmark there is nothing to be gained by using ANN, as they all perform worse than classic GPR and are also computationally more expensive.



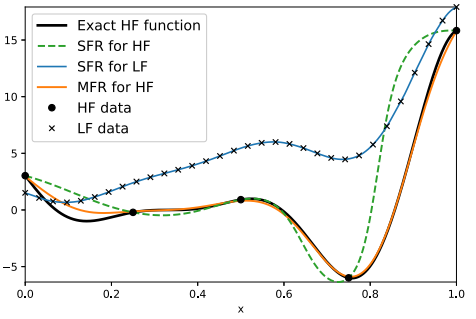
(a) SFR with GP, MFR with co-kriging.



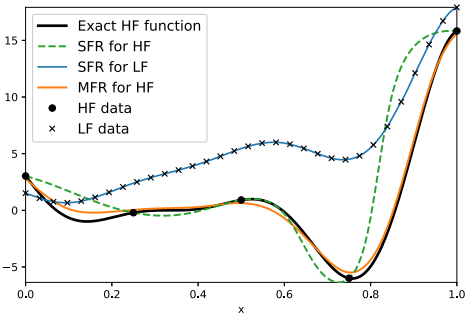
(b) SFR with NN, MFR with 3-step.



(c) SFR with NN, MFR with 2-step.



(d) SFR with NN, MFR with GPmimic.



(e) SFR with NN, MFR with Intermediate.

Fig. 2. Linear correlation: single-fidelity and MF regression results using GPs and different ANNs. There are 5 HF (circle) and 32 LF (cross) observations. Single-fidelity model based on the 5 HF data points is shown in green (dashed).

Table 1
Linear correlation: comparison of the MF regression models. Indicated times account for both HPO and final model predictions.

Model	Validation MSE	Test MSE	R^2	Elapsed time (s)
Co-kriging	–	$\mathbf{2.9 \times 10^{-4}}$	0.999	9
Intermediate	3.03×10^{1}	1.87×10^{-1}	0.990	650
GP-mimic	4.25×10^{0}	1.55×10^{-1}	0.992	752
2-step	7.81×10^{-3}	3.37×10^{-2}	0.998	104
3-step	2.23×10^{-3}	9.53×10^{-4}	0.999	401

Table 2
Discontinuous function: comparison of the MF regression models. Indicated times account for HPO and final model predictions.

Model	Validation MSE	Test MSE	R^2	Elapsed time (s)
Co-kriging	–	8.07×10^{-1}	0.983	35
Intermediate	9.14×10^{0}	2.99×10^{-1}	0.994	1587
GPmimic	1.66×10^{0}	6.92×10^{-1}	0.985	1511
2-step	3.39×10^{1}	$\mathbf{9.52 \times 10^{-2}}$	0.998	311
3-step	7.06×10^{-2}	1.45×10^{-1}	0.997	624

4.2. Benchmark case 2: Discontinuous function

This second benchmark is designed to analyze how well the proposed NN models can approximate discontinuities in an MF setting. HF and LF data are generated from the following functions:

$$f_{\mathrm{LF}}(x) = \begin{cases} 0.5(6x-2)^2 \sin(12x-4) + 10(x-0.5) - 5, & 0 \le x < 0.5 \\ 3 + 0.5(6x-2)^2 \sin(12x-4) + 10(x-0.5) - 5, & 0.5 < x \le 1 \end{cases}$$

$$f_{\mathrm{HF}}(x) = \begin{cases} 2 f_{\mathrm{LF}}(x) - 20(x-1), & 0 \le x < 0.5 \\ 4 + 2 f_{\mathrm{LF}}(x) - 20(x-1), & 0.5 < x \le 1. \end{cases}$$

8 and 32 equally spaced locations over $\Omega = [0, 1]$ are used as the HF and LF inputs, respectively. In addition, 10 equally spaced points in the interval $[0.45, 0.55]$ are added to the LF data set to allow for a better approximation of the discontinuity. In contrast to the first test case, the correlation between the two functions is only piecewise linear, and the piecewise definition results in the discontinuities of distinct amplitudes, see Fig. 3(a).

Fig. 3 shows that the regression models based merely on the HF data are not able to approximate the discontinuity at $x = 0.5$, but the regression results are significantly improved by taking the LF data into account. With R^2 scores over 0.99, both the 2-step and 3-step multilevel NN models show a good match with the exact solution. However, the discontinuity is considerably smoothened when other NN models are used, especially when employing the GPmimic model. While the co-kriging presents a discontinuity at $x = 0.5$, fluctuations are present in the interval $[0.4, 0.6]$. Despite its low test error, the Intermediate model yields a significantly higher validation error than all the other NNs (see Table 2).

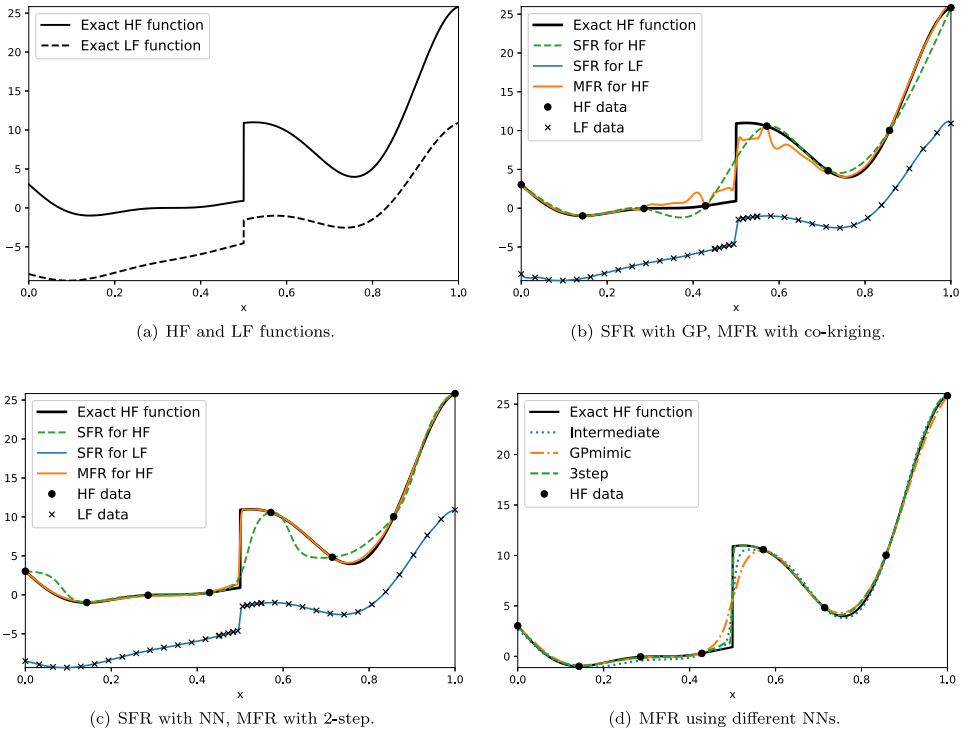
4.3. Benchmark case 3: Nonlinear correlation

The third one-dimensional benchmark will test the proposed NN models on a nonlinear correlation between HF and LF levels. The data samples are obtained from the following functions:

$$f_{\mathrm{LF}}(x) = \sin(8\pi x),$$
$$f_{\mathrm{HF}}(x) = (x - \sqrt{2}) f_{\mathrm{LF}}^2(x).$$

We use 15 equally spaced HF data points and 42 LF data points over the interval $\Omega = [0, 1]$. Fig. 4(a) shows that the frequency of the HF function differs from that of the LF function. Moreover, the amplitude of the HF function is linearly decreasing with x.

As in the previous test cases, single-fidelity regression models do not provide satisfactory accuracy. Neither co-kriging nor the GPmimic model can leverage the LF data to improve the regression results. With an R^2 score of 0.128, the GPmimic NN hardly performs better than the average function value. However, it is not surprising that both models fail in the current benchmark, as their mathematical structure is unable to detect nonlinear correlations between the two fidelity levels. Fig. 4 underlines that the difficulty of this test case lies in approximating the different local extrema of the HF function, since data points are not always available close to the local extremum. Nevertheless, both the multilevel NN models recover an R^2 score exceeding 0.99 (see Table 3).



(a) HF and LF functions.

(b) SFR with GP, MFR with co-kriging.

(c) SFR with NN, MFR with 2-step.

(d) MFR using different NNs.

Fig. 3. Discontinuous function: single- and MF regression results using GPs and different ANNs. There are 8 HF (circle) and 42 LF (cross) observations.

Table 3
Nonlinear correlation: comparison of the MF regression models. Indicated times account for HPO and final model predictions.

Model	Validation MSE	Test MSE	R^2	Elapsed time (s)
Co-kriging	–	5.94×10^{-2}	0.561	36
Intermediate	8.60×10^{-2}	5.48×10^{-3}	0.959	2135
GPmimic	1.46×10^{-1}	1.18×10^{-1}	0.128	2019
2-step	9.51×10^{-4}	8.09×10^{-4}	0.994	1072
3-step	1.38×10^{-3}	$\mathbf{4.49 \times 10^{-4}}$	0.997	1208

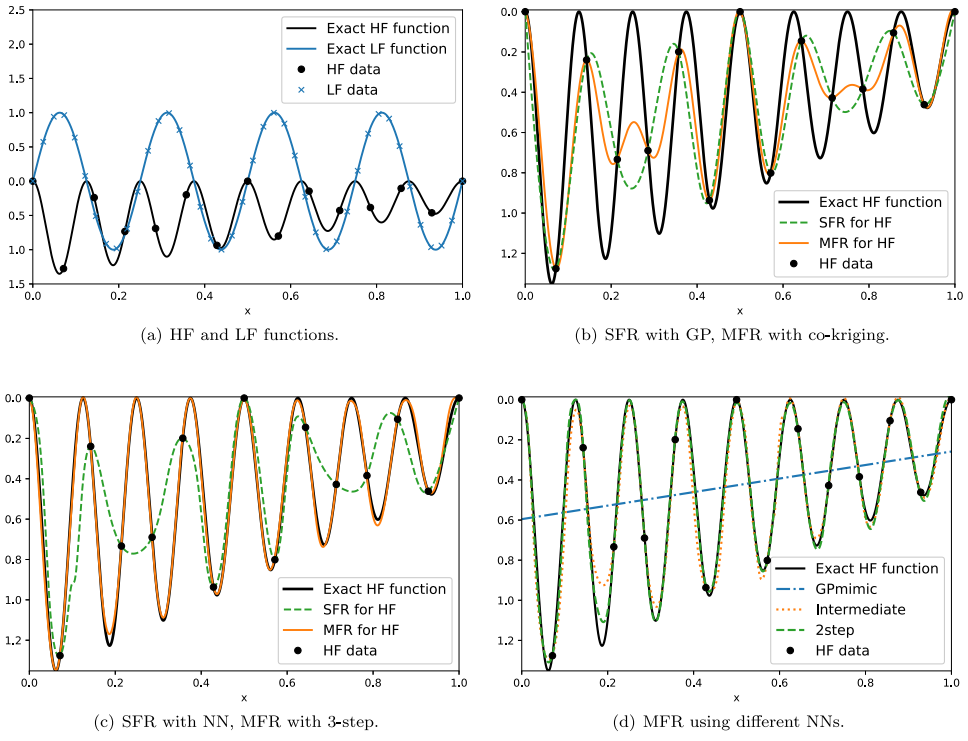
4.4. Benchmark case 4: 20-D benchmark

It is well known that GPR suffer from the curse of dimensionality, and cannot be effectively used when the number of data points is very large ($N > 10000$), especially when big data sets are needed to sufficiently cover the high-dimensional input space. The current benchmark is chosen to show that, in contrast to GPR, NNs remain a valid candidate for MF regression in the presence of high dimensionality and large data sets. The following 20-dimensional functions [21] define the MF setting in this example:

$$f_{\mathrm{HF}}(\mathbf{x}) = (x_1 - 1)^2 + \sum_{i=2}^{20} (2x_i^2 - x_{i-1})^2 \,,$$

$$f_{\mathrm{LF}}(\mathbf{x}) = 0.8 f_{\mathrm{HF}}(\mathbf{x}) - \sum_{i=2}^{20} 0.4 x_{i-1} x_i - 50 \,,$$

with $\mathbf{x} = \{x_1, x_2, \ldots, x_{20}\} \in \Omega = [-3, 3]^{20}$. HF data are sampled from f_{HF} at 5000 locations randomly chosen from a uniform distribution over Ω, while the LF samples are evaluated at 30 000 random input locations.



(a) HF and LF functions.

(b) SFR with GP, MFR with co-kriging.

(c) SFR with NN, MFR with 3-step.

(d) MFR using different NNs.

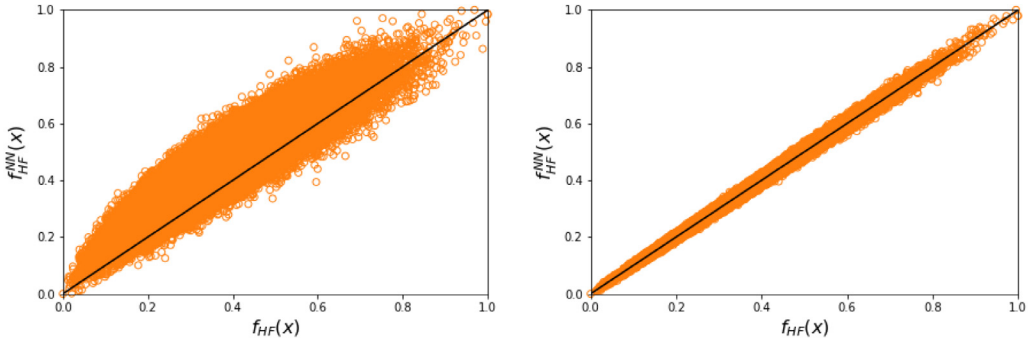
Fig. 4. Nonlinear correlation: single- and MF regression results using GPs and different ANNs. There are 15 HF (circle) and 42 LF (cross) observations. (a) has a different range for the vertical axis than (b), (c) and (d).

In this case, the co-kriging scheme has a 35k $\times$ 35k covariance matrix, which means the evaluations of posterior mean and variance will require multiple solves of linear systems of size 35k. For a fixed set of hyperparameters, there is one linear solve required to formulate the posterior mean function, but the variance at each test location will require a new linear solve. Additionally, when determining the optimal values of hyperparameters, linear solves are repeatedly needed at the updated hyperparameter values in each iterative step. On the other hand, just the storage of such a matrix can take around 10 GB of memory. One linear solve with an $N \times N$ coefficient matrix has the computational complexity of more than $O(N^2)$, $O(N^3)$ at most, $N = 35$k in this case, whereas one step of stochastic gradient descent for NN training requires $O(n_w n_b)$ computations, n_w and n_b being the number of NN parameters (weights and biases) and the size of a mini-batch, respectively. Hence the construction of a co-kriging surrogate is very likely to be more computationally intensive than NN training. Therefore, the GPR would have a high demand for both memory and computational time in this 20-D problem, suffering from the curse of dimensionality, but the reliability could still be poor. All these factors make co-kriging a lousy candidate for this problem.

Instead we utilize the proposed 3-step NN model whose training is accelerated by a GPU, and we opt for the 5-fold cross-validation to tune NN hyperparameters. Fig. 5 shows that the 3-step model can accurately predict the value of y_{HF} at one million random input locations.

5. Numerical results (II): application to parametrized PDEs

In this section we apply our MF framework to a problem arising in acoustics, namely the propagation of a pressure wave $P(\mathbf{x}, t)$ into an acoustic horn with parametrized shape, addressed in [38]. In particular, we are interested in evaluating an input–output map involving the solution of a PDE, exploiting the accuracy of the finite element method to obtain HF solutions, while relying on a reduced basis (RB) method to compute LF, but fast and inexpensive, approximations. Different from the previous section, we consider different sources of data for each fidelity level.



(a) Single-fidelity regression (b) MF regression with the 3-step model

Fig. 5. 20-D benchmark: Exact function values versus predicted values by NNs at one million random locations. All data are normalized to the range [0,1].

We consider an acoustic device, illustrated in Fig. 6 (left), comprising of a waveguide, with infinite extension to the left and a conical extremity on the right, namely the horn[2], and an internal propagating planar wave inside the waveguide: once the wave reaches the horn, a portion of its energy is converted into an outer-going wave. Under the assumptions of single-frequency and time harmonic waves, the acoustic pressure can be expressed as $P(\mathbf{x}, t) = \mathrm{Re}(p(\mathbf{x})e^{i\omega t})$, where the complex amplitude $p(\mathbf{x})$ satisfies the monochromatic steady-state Helmholtz equation with mixed Neumann–Robin boundary conditions:

$$\begin{aligned} \Delta p + k^2 p &= 0 && \text{in } \Omega \\ (ik + \frac{1}{2R})p + \nabla p \cdot \mathbf{n} &= 0 && \text{on } \Gamma_0 \\ ikp + \nabla p \cdot \mathbf{n} &= 2ikA && \text{on } \Gamma_i \\ \nabla p \cdot \mathbf{n} &= 0 && \text{on } \Gamma_h \cup \Gamma_s = \Gamma_n, \end{aligned} \tag{15}$$

where $k = \omega/c$ is the wave number, $\omega = 2\pi f$ the angular frequency and $c = 340$ cm s^{-1} the speed of sound; $\mathbf{n}$ denotes the outward-directed unit normal on the boundary of Ω. We restrict the computation to the domain Ω shown in Fig. 6, and impose on Γ_i a propagating wave with amplitude $A = 1$ while an absorbing condition on the far-field boundary Γ_o — absorbing the outer-going planar waves, and homogeneous Neumann boundary conditions on the sound-hard walls of the device Γ_h as well as on the symmetry boundary Γ_s. We take, for simplicity, the radius equal to $R = 1$, see [39] for more details.

In addition to the frequency f of the incoming wave, we parametrize, as in [38], the shape of the horn by means of radial basis functions, introducing as parameters $\boldsymbol{\mu}_g = (\mu_{g,1}, \ldots, \mu_{g,4})$ the vertical displacement of four control points located on the horn wall Γ_h, shown in Fig. 6 (right). The admissible domain configurations are defined as the diffeomorphic images $\Omega(\boldsymbol{\mu}_g)$ of the reference shape Ω through a deformation mapping $\mathbf{T}(\cdot; \boldsymbol{\mu}_g)$ obtained as linear combinations of the control points' displacements. Hence, the acoustic problem depends on five parameters, i.e., we let $\boldsymbol{\mu} = (f, \boldsymbol{\mu}_g)$ denote the parameter vector and $\mathcal{D} \subset \mathbb{R}^p$ the parameter domain.

We focus our analysis on a specific output of interest, namely the index of reflection intensity (IRI) [39] which measures the transmission efficiency of the acoustic horn and is defined as the absolute value of the average reflected wave at the sound inlet Γ_i, i.e.,

$$\mathrm{IRI}(\boldsymbol{\mu}) = J(p(\boldsymbol{\mu})) = \left| \frac{1}{|\Gamma_i|} \int_{\Gamma_i} p(\boldsymbol{\mu})\, d\Gamma - 1 \right|,$$

in which the functional J defines the quantity of interest — the IRI. In particular,

$$\mathrm{IRI}_{\mathrm{LF}}^{r}(\boldsymbol{\mu}) = J(p_r^m(\boldsymbol{\mu})), \qquad \text{and} \qquad \mathrm{IRI}_{\mathrm{HF}}(\boldsymbol{\mu}) = J(p_h(\boldsymbol{\mu})),$$

[2] For simplicity, the device extends infinitely in the direction normal to the plan and its wall consists of sound-hard material. Therefore, for the frequencies in the range under consideration, we assume that all non-planar modes in the waveguide are negligible, which allows us to reduce the problem to two space dimensions.

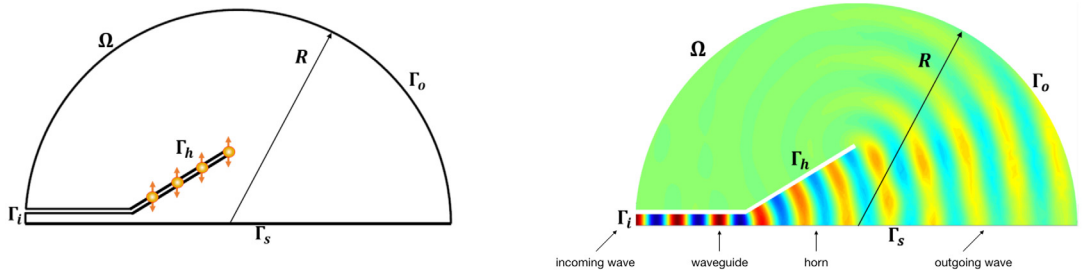


Fig. 6. Acoustic horn problem. Left: the computational domain Ω and the boundaries. Right: the control points used in RBF shape parametrization, whose vertical displacements are treated as parameters.

Table 4
Computational details in the case with $p = 1$ parameter: $\boldsymbol{\mu} = f$.

Number of parameters	1	Parameter domain $\mathcal{D}$	$[10, 1800]$
Number of finite elements	8740	Tolerance RB POD	10^{-5}
Number of FE DoFs n	4567	Number of ROM DoFs r	44
Number of HF data	from 5 to 22	Number of bases	from 5 to 22
Number of LF data	32	Sampling method	LHS

define the MF setting for this test case. We denote by $(p_r^m(\boldsymbol{\mu}))$ and $(p_h(\boldsymbol{\mu}))$ the LF and the HF solutions of the parametrized PDE, respectively, the construction of which is reported in Appendix C. Moreover, we highlight the dependence of the LF model on the dimension N of the ROM used to evaluate the PDE solution. In the following subsections, we consider two different scenarios, dealing with either 1 or 5 parameters.

5.1. Case with $p = 1$ parameter

In this first case, we compare different MF strategies, by considering the output of interest as a function of the frequency $\boldsymbol{\mu} = f$ only, letting it vary in $\mathcal{D} = [10, 1800]$. Hence, we first limit our analysis to the reference configuration of the horn, without taking geometric parameters into account. In this specific case, the linear system arising from the FE approximation of the problem (15) exhibits an affine decomposition (see Appendix C), so that any further hyper-reduction stage is not required when constructing the ROM. To build this, we first randomly sample 150 values of the frequency f and compute the corresponding HF solutions through the FOM. The FOM is approximated by $\mathbb{P}_1$ finite elements and, considering a mesh made of 8740 triangular elements, we have an HF model of dimension $n = 4567$. Then, we apply POD and extract $r = 44$ reduced basis functions by imposing a relative projection error of 10^{-5}. All computational details are summarized in Table 4. The computation of HF and LF solutions, i.e. the FOM and ROM solutions, respectively, is carried out in MATLAB, using the redbKIT library [40]. All hyperparameters obtained by HPO are reported in Appendix A.

We assess how the quality of the LF model and the amount of HF data impact the accuracy of the MF prediction. We recall that (*i*) we can improve the quality of the LF model by selecting a smaller or larger number of bases, and (*ii*) we can freely decide the parameter values for which we solve the HF model, by keeping the sampling method fixed. A comparison among LF models obtained with different dimensions r of the ROM is reported in Fig. 7. Even though the maximum number of available basis functions is 44, we choose to limit the selected basis functions in the POD-Galerkin ROM to be between 5 and 22, to deal with a potentially inaccurate (or, at least, not accurate enough) LF model. The ability to obtain accurate predictions by combining a few HF data and several evaluations of a reliable, but not sufficiently accurate, ROM, is indeed an attractive feature of the proposed framework, as this may prevent us from constructing ROMs with large dimensions and poor efficiency. Moreover, we consider the same amount of HF data, while we keep the number of LF data fixed to 32.

In the following, we test the Intermediate, 2-step, and 3-step network models. For each combination *number of HF data – number of bases*, we train the NNs, predict the output of interest and compute the goodness-of-fit indices, R^2 and MSE. Evaluated outputs with these network architectures are displayed in Fig. 8. The values of R^2

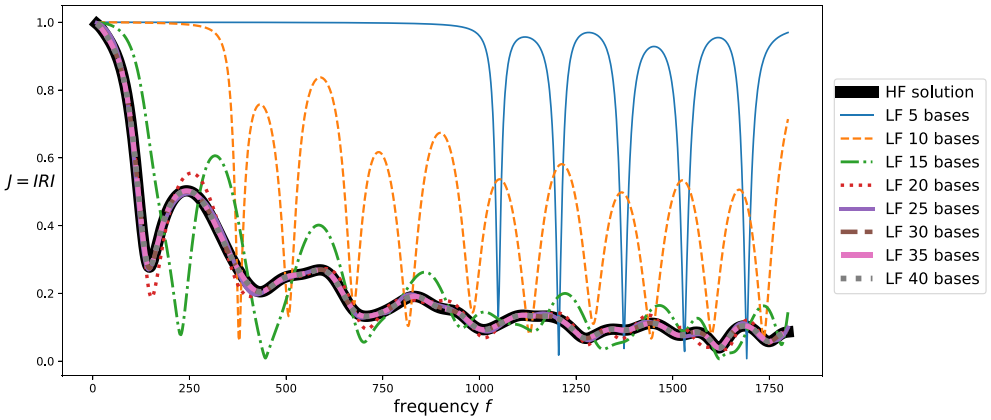


Fig. 7. Quantity of interest $J = \text{IRI}$: HF solution $f_{\text{HF}}(\boldsymbol{\mu})$ and different LF solutions $f_{\text{LF}}(\boldsymbol{\mu})$ depending on the number r of basis functions. For too small values of r, the LF model prediction is rather poor, while it is almost indistinguishable from the HF model prediction for $r \geq 25$.

and MSE are reported in Figs. 9–11, as functions of the number N of basis functions and the amount of HF data considered.

Overall, the NN architectures perform very well and provide good predictions, in terms of both MSE and R^2, provided that a sufficient number of basis functions and HF data are used. The 2-step and 3-step models produce similar results and perform better than the Intermediate model, as displayed in Fig. 8. In particular, the multilevel models are robust and efficient even in the cases where the LF models are built with few bases, while the Intermediate network has poor predictive accuracy without a sufficiently accurate LF model, even if a large amount of HF data are provided. In fact, with multilevel networks we can reach values of R^2 larger than 0.8 even considering just $r = 5$ bases, whereas the Intermediate model does not provide large values of R^2 with less than 12 bases, see Figs. 9–11. Regarding the computed outputs, we see how the peaks with larger amplitude found for $f < 1000$ are correctly described by both the 2-step and 3-step models, while smaller amplitude peaks for $f > 1000$ are better captured by the 3-step model than the 2-step. On the other hand, the Intermediate model provides a less accurate trend of the output, and only provides reliable results provided that both N and the amount of HF data are large enough.

As expected, the prediction improves as the number of HF data or the number of basis functions increases, even if these two features impact the accuracy in a slightly different way. For the sake of simplicity, we restrict to the case of a 3-step model, see Fig. 11 (top). It is observed that, for each fixed size of reduced basis r, the goodness-of-fit indices improve as the number of HF data increases until a threshold limit is reached, which is determined solely by the number of bases and cannot be overcome by adding more HF data. Conversely by increasing N, as shown in Fig. 11 (bottom), the trends of R^2 (resp. MSE), corresponding to the different numbers of HF data adopted, increase (resp. decrease) and also tend to reach values closer to each other. The amount of HF data thus becomes less and less important as the LF model improves. Hence, in this specific problem it is more efficient to have a good LF model even with a small amount of HF data, rather than lots of HF data but a poor LF model.

5.2. *Case with $p = 5$ parameters*

We finally apply our MF setting to the case where all the five parameters $\boldsymbol{\mu} = (f, \boldsymbol{\mu}_g)$, namely the frequency and the four geometric parameters, vary, in order to consider shape variations in the horn geometry as well. The parameter domain is $\mathcal{D} = [50, 1000] \times \mathcal{D}_g$, where $\mathcal{D}_g = [-0.03, 0.03]^4$.

We compute 200 FOM snapshots for 200 points in the parameter domain $\mathcal{D}$ through a Latin Hypercube sampling (LHS) design. In this case, the reduced basis is made from $r = 80$ POD modes, see Table 5. As in the case of $p = 1$ parameter, we select an appropriate range for the number of bases and HF data to assess the efficiency of the MF approach in different scenarios. We consider a larger number of LF data (500 instead of 100) than in the case of $p = 1$ parameter. We then vary the number of HF data from 5 to 45, and the number of basis functions of the

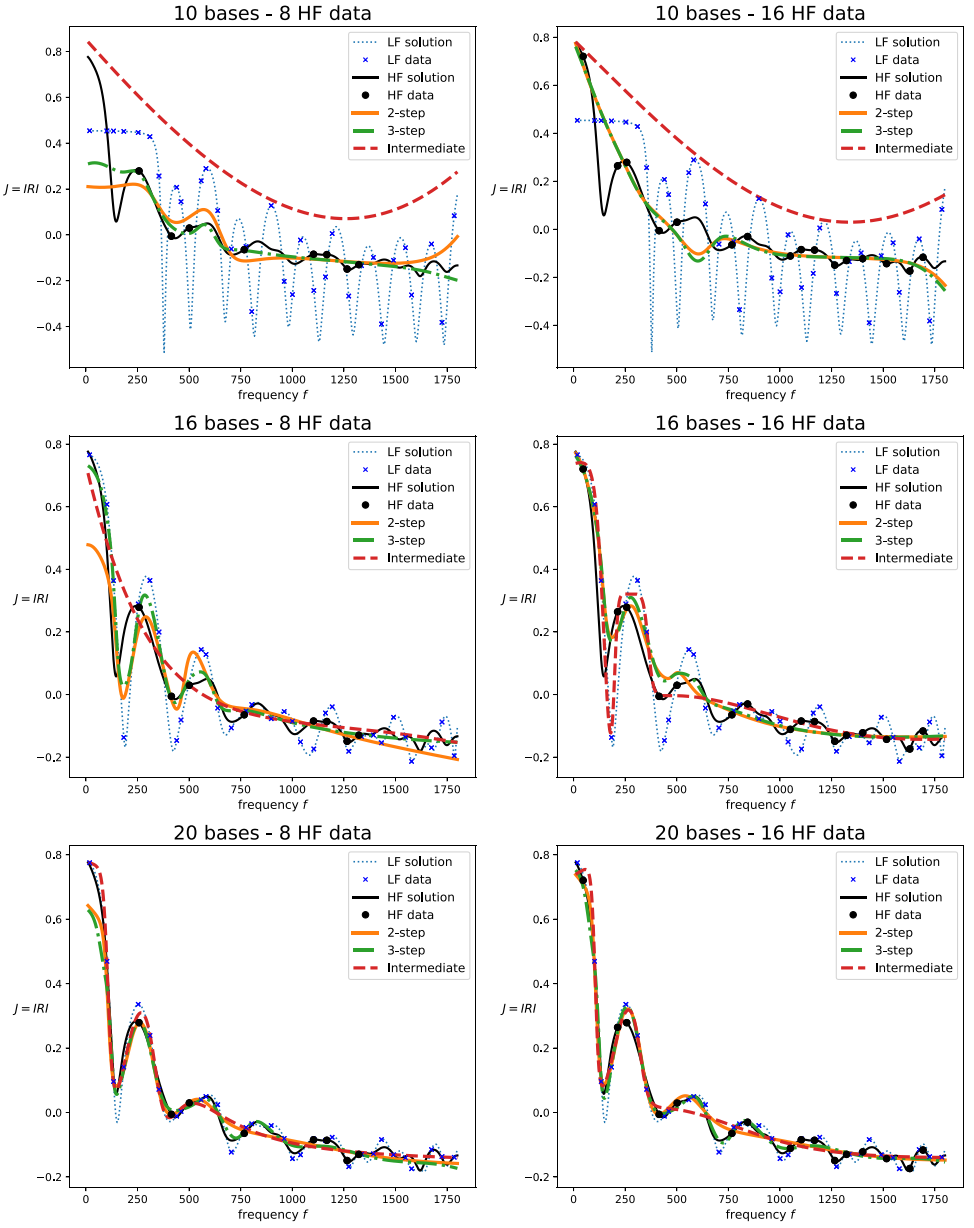


Fig. 8. Quantity of interest $J = \text{IRI}$: the HF and LF solutions as well as the MF (*Intermediate*, *2-step*, *3-step*) results considering different numbers of bases and HF training data.

LF model from 5 to 40. Results are only reported in the case of the *3-step* model for the sake of brevity, focusing on the roles of the quality and quantity of the training data rather than on the choice of the NN architecture.

Passing from 1 to 5 parameters does not worsen the prediction power of the NNMFR. Indeed, once again we obtain an accurate prediction from a small number of FOM data by exploiting a large number of ROM solutions that can be computed very quickly and inexpensively. As in the previous case, the MF prediction improves both as the number of HF data and the number of bases increase, with the latter playing a more important role in this problem. Similar to the case with $p = 1$, we display R^2 and the MSE obtained with the 3-step NN model as functions of the aforementioned factors, see Fig. 12.

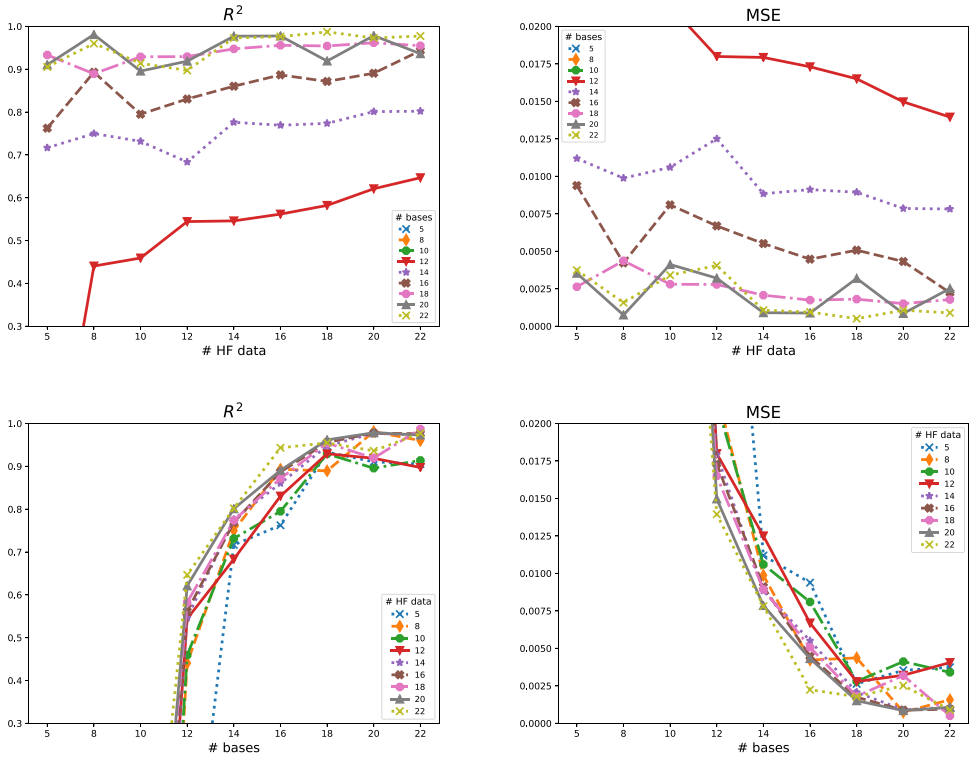


Fig. 9. Case with $p = 1$ parameter, *Intermediate* model. MSE and R^2 for different amounts of HF data and LF dimension r.

Table 5
Computational details in the case with $p = 5$ parameters: $\boldsymbol{\mu} = (f, \boldsymbol{\mu}_g)$.

Number of parameters	5	Parameter domain $\mathcal{D}$	$[50, 1000] \times [-0.03, 0.03]^4$
Number of finite elements	8740	Number of FE snapshots	200
Number of FE DoFs n	4567	Number of ROM DoFs r	80
Number of HF data	from 5 to 45	Number of bases	from 5 to 40
Number of LF data	500	Sampling method	LHS

Numerical results show that we can achieve very good results even in 5 dimensions by employing 500 LF data, and that increasing both the amount of HF data and the dimension of the LF model improves the accuracy of prediction. When considering a poor LF model, the number of HF data is fundamental for obtaining good prediction: for instance, with an LF model of dimension $r = 15$, we obtain $R^2 = 0.12768$ with 5 HF data, and $R^2 = 0.91093$ with 40 HF data. In contrast, the amount of HF data loses importance as the LF model becomes more accurate. In fact, with an LF model of dimension $r = 35$, when HF data increases from 10 to 45, R^2 only improves by 0.2%, passing from 0.98882 to 0.99107. For a fixed, small number of basis functions, e.g., $r = 5$, R^2 (resp., the MSE) continues to increase (resp., decrease) as the number of HF data increases, while both indices flatten when considering larger values of r, e.g., $r > 15$. Considering a fixed number of HF data greater than 5, it is possible to reach excellent values of the goodness-of-fit indices just by increasing the number of bases. In particular, starting from 20 bases, we already reach an R^2 greater than 0.97 and an MSE smaller than 1.2×10^{-3}, both continuing to improve as the number of bases increases. On the other hand, as r increases, the indices keep improving and the number of HF data becomes less and less relevant. Therefore, in the case with $p = 5$ parameters, we can again conclude that improving the quality of the LF model is a more efficient strategy – computationally cheaper as well – than increasing the number of HF data, to reach a certain degree of accuracy.

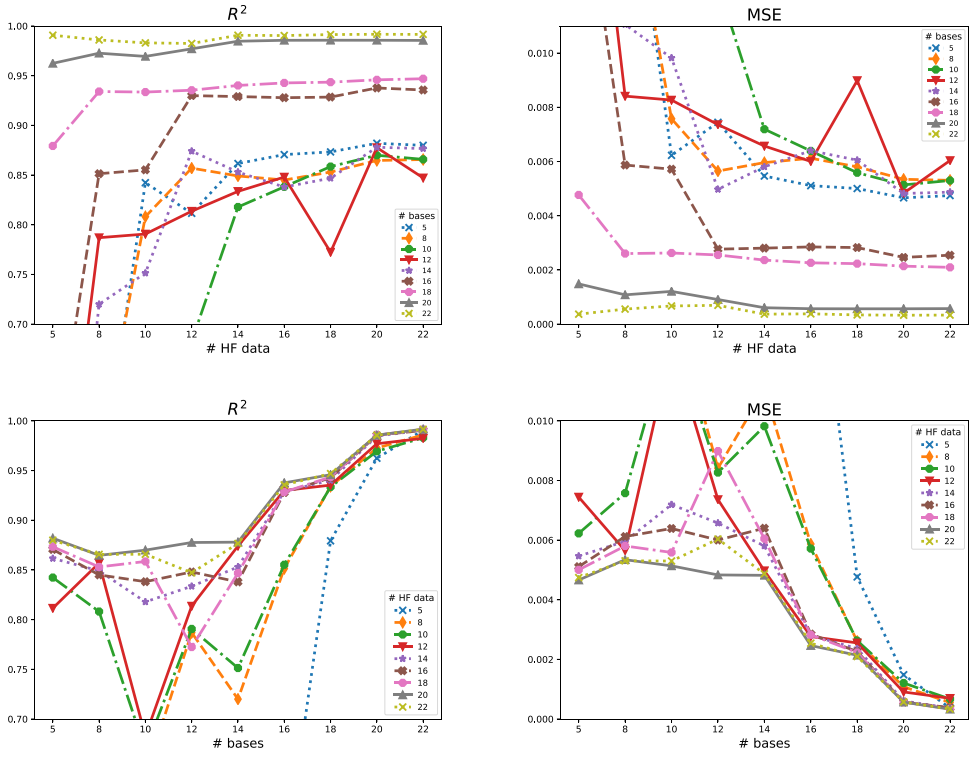


Fig. 10. Case with $p = 1$ parameter, *2-step* model. MSE and R^2 for different amounts of HF data and LF dimension r.

6. Conclusions

In this work we discuss MF regression with ANNs, for which four different architectures are presented. The proposed NN schemes are benchmarked against co-kriging on a collection of test cases of increasing complexity. We also successfully predict output quantities associated with parametrized PDEs using the multi-fidelity NN models. Observations made from different numerical examples show that MF models based on NNs can outperform co-kriging schemes. In contrast to co-kriging, NNs are able to detect nonlinear correlations between fidelity levels more effectively and are capable of dealing with large data sets. In addition, the hyperparameter selection for the proposed multi-fidelity NNs is automatized based on cross validation and Bayesian optimization, and the tuned NN models consistently yield very low prediction errors.

The proposed models have been tested on a series of manufactured benchmarks and applied to a parametrized PDE problem. In the latter, the goal is to evaluate an output functional of the PDE solution that features an oscillating input–output dependence, and the LF model is constructed through the reduced basis method while the HF model is given by detailed finite element analysis. Numerical results show that the accuracy of the predictions through MF regression is mainly driven by the reliability of the reduced order model, rather than the amount of HF data fed into the NNs.

Even though only bi-fidelity cases are discussed in this work, all the proposed NN models can be extended to more than two fidelity levels, by adding to Intermediate models more latent variables and/or outputs (for GPmimic), or by adding extra NN steps to multilevel models. A promising direction for future work is to inspect whether changing the value of the balance parameter α during the training process can improve the model accuracy. As the ratio $\mathrm{MSE}_{\mathrm{HF}}/\mathrm{MSE}_{\mathrm{LF}}$ evolves during the training process, a corresponding adaptation of α could be a reasonable improvement of the current all-in-one models.

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.

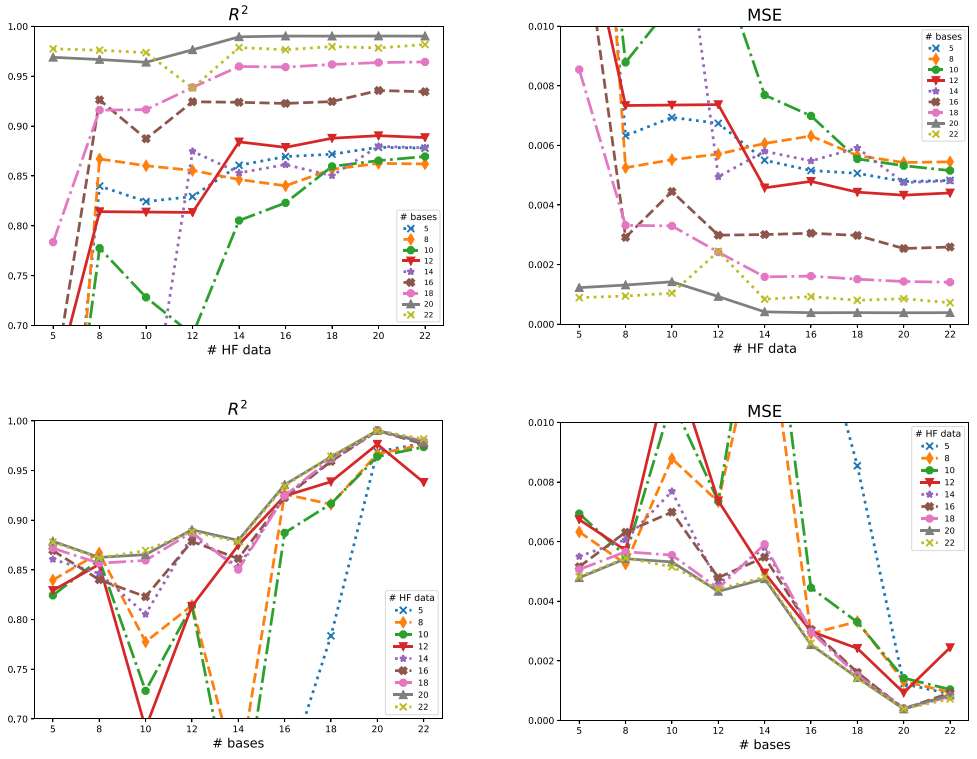


Fig. 11. Case with $p = 1$ parameter, *3-step* model. MSE and R^2 for different amounts of HF data and LF dimension r.

Acknowledgments

M. Guo is supported by Sectorplan Bèta under the focus area *Mathematics of Computational Science*. A. Manzoni acknowledges the support of Fondazione Cariplo under grant No. 2019-4608.

Appendix A. Hyperparameter summary

Tables A.1 and A.2 show the values of NN hyperparameters resulting from the HPO in different numerical examples. We recall that α is the parameter in all-in-one networks that balances the fidelity levels' contributions to training error, λ is a L^2-regularization parameter, and η is the learning rate.

In Table A.1, the 'depth $\times$ width' column shows the results of optimized architecture of the NN models. Note that the 'depth' for an Intermediate model is the number of hidden layers after the 3rd layer where the LF latent variable is located, and the 'depth' of a GPmimic model stands for the number of hidden layers before $\{u_1, u_2\}$. For 2-step and 3-step models, 'depth $\times$ width' gives out optimization results of the single-hidden-layer NN_{HF}, for which the 'width' is optimized. In Table A.2, 'width' indicates the number of nodes in each layer whose size is left to be determined through the HPO. Here the number of hidden layers of the Intermediate model is fixed to 5, and that of NN_{HF} in the multilevel models is 1.

Although NN hyperparameters should generally not be analyzed separately, the following comments about the individual hyperparameters can be made:

- In all test cases, the optimization procedure chooses $\alpha < 0.1$, which corresponds to weighting the LF data one order of magnitude more than their HF counterpart.
- The learning rates η in models belonging to the same NNMFR class (all-in-one or multilevel) are generally at the same order of magnitude.
- In most cases, the AdaMax algorithm is the best performing optimizer.

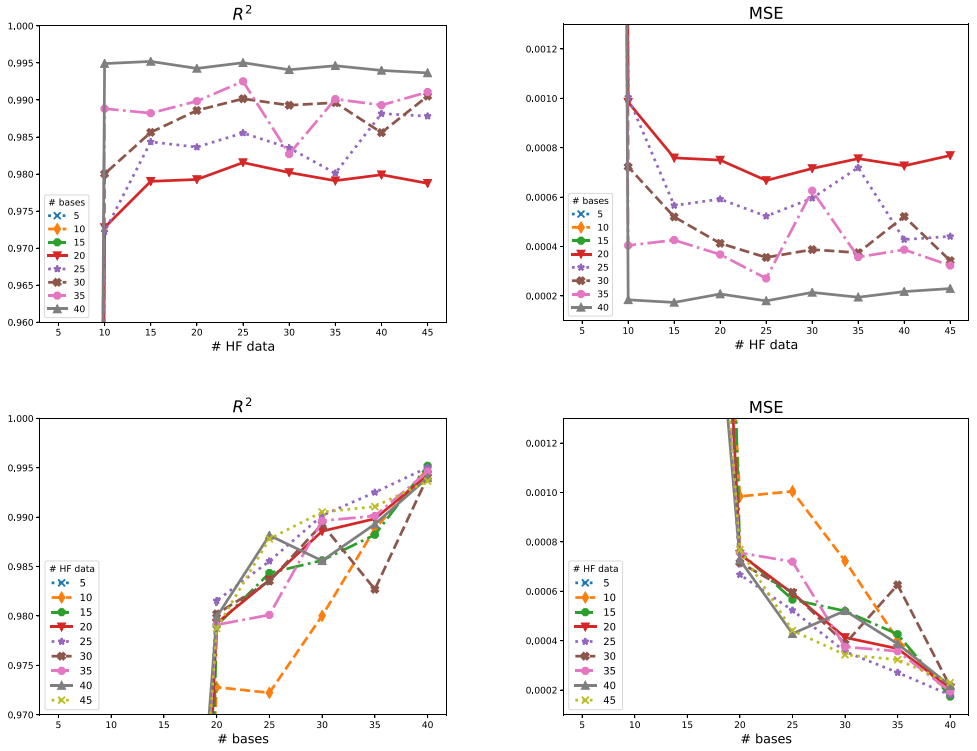


Fig. 12. Case with $p = 5$ parameters, *3-step* model. MSE and R^2 for different amounts of HF data and LF dimension r.

Table A.1
Hyperparameter values of the NNMFR used for different artificial test cases. Benchmark 1 involves linear correlation, benchmark 2 discontinuous functions, and benchmark 3 non-linear correlation.

Test	Model	Weight initializer	Depth ×width	α	λ	Optimizer	η
1	Intermediate	Glorot normal	2×59	4.59×10^{-4}	2.28×10^{-3}	Adam	6.35×10^{-3}
	GPmimic	Glorot uniform	3×33	5.01×10^{-4}	1.16×10^{-4}	AdaMax	3.90×10^{-3}
	2-step	uniform	1×34	–	1.01×10^{-4}	AdaMax	5.17×10^{-2}
	3-step	Glorot uniform	1×38	–	1.08×10^{-7}	Adam	2.06×10^{-2}
2	Intermediate	uniform	2×30	9.22×10^{-3}	1.94×10^{-3}	AdaMax	3.41×10^{-2}
	GPmimic	Glorot uniform	4×10	2.71×10^{-2}	2.17×10^{-4}	AdaMax	4.42×10^{-2}
	2-step	normal	1×60	–	1.01×10^{-3}	AdaMax	3.65×10^{-3}
	3-step	uniform	1×98	–	2.30×10^{-4}	AdaMax	1.02×10^{-3}
3	Intermediate	Glorot uniform	3×39	4.02×10^{-4}	2.85×10^{-5}	Adam	1.55×10^{-2}
	GPmimic	Glorot normal	4×22	9.74×10^{-1}	0	AdaMax	1.14×10^{-4}
	2-step	uniform	1×44	–	1.02×10^{-4}	AdaMax	4.84×10^{-3}
	3-step	Glorot uniform	1×62	–	2.35×10^{-4}	Adam	1.25×10^{-4}

- In the benchmarks, Glorot uniform weight initialization is preferred, whereas standard uniform initialization performs best when predicting the IRI in the acoustic horn problem.

Appendix B. Motivation behind the choice of the architectures

The widths and the depths of the NN models proposed in Section 3 may significantly affect their performance, which will be briefly discussed in this appendix. Here the Intermediate and 2-step models are taken as examples, and analysis will be performed on the first benchmark case. As emphasized in the body texts, we employed the HPO to find the optimal choices of hyperparameters. The validation MSE of LOOCV are computed at 100 combinations

Table A.2
Hyperparameter values of the considered NN architectures regarding the problems in Section 5; here P indicates the number of parameters considered.

P	Model	Weight initializer	α	λ	Optimizer	Width	η	Elapsed time (s)
1	Inter.	Uniform	3.19×10^{-2}	2.65×10^{-3}	Adam	114	4.27×10^{-4}	1426
1	2-step	Uniform	–	1.21×10^{-4}	Adamax	18	1.76×10^{-3}	997
1	3-step	Glorot unif.	–	2.01×10^{-4}	Adam	6	1.42×10^{-3}	995
5	3-step	Uniform	–	7.33×10^{-2}	Adam	48	7.57×10^{-4}	1089

of 'depth × width'. Specifically, for the 2-step model we tune the depth and width for both NN_{LF} and NN_{HF}, while for the Intermediate model we optimize the number of layers between the LF and HF outputs and the corresponding number of nodes. We consider a range from 1 to 5 for the depth, and from 4 to 120 for the width, and the results are shown in Fig. 13. Note that the optimal hyperparameter values shown in Fig. 13 are slightly different from those in Table A.1, as more hyperparameters on a refined grid are considered for optimization and NN_{HF}'s depth is fixed in Table A.1.

Overall, the NN architectures may suffer from limited approximation capacity if an insufficient number of layers or nodes are employed, while a saturation of the goodness-of-fit may be incurred if too many layers or nodes are adopted. The minimum of validation MSE is achieved at the optimal choice of 'depth × width'. In the present work, we sometimes observe that the optimization of width makes more difference than depth, e.g., in the acoustic horn problem. In such a case we fix the depth and leave the width to be optimized. Moreover, the depth is chosen to be sufficiently expressive for fitting the LF and HF functions, while over-fitting is dealt with through L_2-regularization.

Appendix C. HF and LF models for the acoustic horn problem

To derive the HF model related to the test case of Section 5, we employ the Galerkin-finite element method. We write the weak formulation of problem (15): given $\boldsymbol{\mu} \in \mathcal{D}$, find $p(\boldsymbol{\mu}) \in V$ s.t.

$$a(p(\boldsymbol{\mu}, u; \boldsymbol{\mu})) = f(u; \boldsymbol{\mu}) \qquad \forall u \in V \tag{16}$$

where $V = H^1(\Omega(\boldsymbol{\mu}_g)) = \{q \in L^2(\Omega(\boldsymbol{\mu}_g)) : \partial q/\partial x_j \in L^2(\Omega(\boldsymbol{\mu}_g)),\ j \in \{1, 2\}\}$ and the bilinear form $a(\cdot, \cdot, \boldsymbol{\mu}) : V \times V \to \mathbb{C}$ and the linear form $f(\cdot; \boldsymbol{\mu}) : V \to \mathbb{C}$ are defined, respectively, by

$$2a(p, u; \boldsymbol{\mu}) = \int_{\Omega(\boldsymbol{\mu}_g)} \{\nabla p \cdot \nabla \bar{u} - k^2 p \bar{u}\}\, d\Omega + ik \int_{\Gamma_o \cup \Gamma_i} p \bar{u}\, d\Gamma + \frac{1}{2R} \int_{\Gamma_o} p \bar{u}\, d\Gamma \tag{17}$$

$$f(u; \boldsymbol{\mu}) = 2ikA \int_{\Gamma_i} \bar{u}\, d\Gamma \tag{18}$$

Next, we introduce a conforming triangulation $\mathcal{T}_h = \{\Delta_k\}_{k=1}^{n_e}$ of the domain Ω and seek a HF approximation $p_h(\boldsymbol{\mu}) \in V_h$ as a globally continuous, piecewise linear, function belonging to a finite-dimensional space $V_h \subset V$. In our case, V_h is spanned by a set of basis functions $\{\phi_i\}_{i=1}^n$ for the space V_h consisting of a set of n piecewise polynomial nodal basis functions on $\mathcal{T}_h$. The Galerkin-finite element approximation of (16) thus results in the following n dimensional linear system:

$$\mathbf{A}(\boldsymbol{\mu})\mathbf{p}_h(\boldsymbol{\mu}) = \mathbf{f}(\boldsymbol{\mu}) \tag{19}$$

where $\mathbf{A}_{ij}(\boldsymbol{\mu}) = a(\phi_j, \phi_i; \boldsymbol{\mu})$, $\mathbf{f}_i(\mu) = f(\phi_i; \boldsymbol{\mu})$, for $\leq i, j \leq n$ and $\mathbf{p}_h$ is the vector of coefficients $\{p_i\}_{i=1}^n$ such that the projection of p onto V_h is $p_h(\mathbf{x}) = \sum_{i=1}^n p_i \phi_i(\mathbf{x})$. This latter formula allows us to state a one-to-one correspondence between the finite element functions $p_h(\boldsymbol{\mu}) \in V_h$ and their discrete counterparts $\mathbf{p}_h(\boldsymbol{\mu}) \in \mathbb{R}^n$.

To derive the LF model, we employ the reduced basis (RB) method [41,42], which is briefly recalled here. The RB method is a projection-based reduced order modeling technique, addressing the repeated solution of parametrized PDEs, which allows to dramatically reduce the dimension of the discrete problems arising from numerical approximation. The strategy adopted in RB methods consists in the projection of the HF problem upon a subspace made of specially selected basis functions, built from a set of HF solutions corresponding to suitably chosen parameters (or snapshots), e.g., through proper orthogonal decomposition (POD). Later, a (Petrov-)Galerkin projection onto the RB space is employed to generate the ROM.

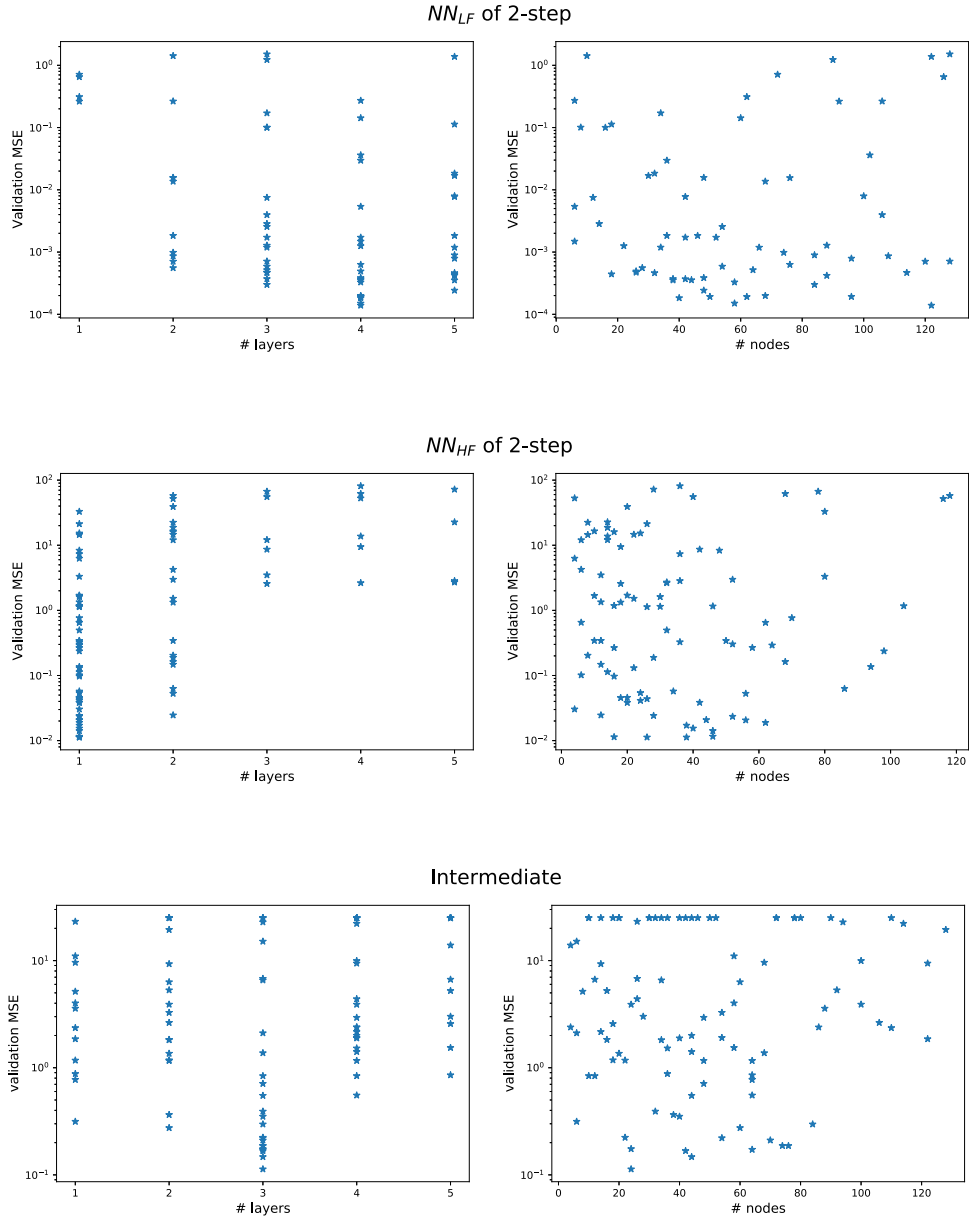


Fig. 13. Validation MSE as function of width and depth for each of the 100 combinations of hyperparameters. Straying from the optimal depth and width values results in a degradation of performance, indicated by an increase in validation error.

Starting from the FOM (19), i.e. find $\mathbf{p}_h(\boldsymbol{\mu})$ such that $\mathbf{A}(\mu)\mathbf{p}_h(\boldsymbol{\mu}) = \mathbf{f}(\mu)$, the idea of a projection-based ROM is to approximate $\mathbf{p}_h(\mu) \approx \mathbf{V}\mathbf{p}_r(\mu)$ as a linear combination of basis functions, for a vector of unknown reduced degrees of freedom $\mathbf{p}_r(\mu)$ of reduced dimension $r \ll n$. This latter is sought by imposing that

$$\mathbf{W}^{\mathrm{T}}(\mathbf{A}(\boldsymbol{\mu})\mathbf{V}\mathbf{p}_N(\boldsymbol{\mu}) - \mathbf{f}(\boldsymbol{\mu})) = \mathbf{0},$$

a condition which enforces the orthogonality of the residual to a subspace spanned by a suitable test basis $\mathbf{W} \in \mathbb{R}^{n\times r}$. The ROM reads: find $\mathbf{p}_r \in \mathbb{R}^r$ such that

$$\mathbf{A}_r(\boldsymbol{\mu})\mathbf{p}_r(\boldsymbol{\mu}) = \mathbf{f}_r(\mu) \tag{20}$$

where $\mathbf{A}_r(\boldsymbol{\mu}) = \mathbf{W}^{\mathrm{T}}\mathbf{A}(\boldsymbol{\mu})\mathbf{V}$ and $\mathbf{f}_r(\boldsymbol{\mu}) = \mathbf{W}^{\mathrm{T}}\mathbf{f}(\boldsymbol{\mu})$. A Galerkin projection results if $\mathbf{W} = \mathbf{V}$. As for the HF approximation, the RB approximation reflects a one-to-one correspondence between the function $p_r^h(\boldsymbol{\mu}) \in V_h$ and its finite-dimensional counterpart $\mathbf{V}\mathbf{p}_r(\boldsymbol{\mu}) \in \mathbb{R}^n$.

Although the dimension of the RB problem (20) is very small as compared to the FOM (19), the assembling of the former system still depends in general on the dimension n of the HF system for any $\boldsymbol{\mu} \in \mathcal{D}$. A convenient situation arises when the HF arrays in (19) can be written as

$$\mathbf{A}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_A} \Theta_q^a(\boldsymbol{\mu})\mathbf{A}_q, \qquad \mathbf{f}(\boldsymbol{\mu}) = \sum_{q=1}^{Q_f} \Theta_q^f(\boldsymbol{\mu})\mathbf{f}_q.$$

By virtue of this property — which we refer to as affine parametric dependence of $\mathbf{A}(\boldsymbol{\mu})$ and $\mathbf{f}(\boldsymbol{\mu})$, the assembling of the system (20) during the online stage can be made efficient, since the arrays $\mathbf{W}^{\mathrm{T}}\mathbf{A}_q\mathbf{V}$, $q = 1, \ldots, Q_A$ and $\mathbf{W}^{\mathrm{T}}\mathbf{f}_q$, $q = 1, \ldots, Q_F$, can be pre-computed and stored during a possibly expensive offline stage.

Since in the Helmholtz problem with $p > 1$ parameters we deal with parametrized shape deformations, the FOM arrays $\mathbf{A}(\boldsymbol{\mu})$ and $\mathbf{f}(\boldsymbol{\mu})$ are nonaffine functions of $\boldsymbol{\mu}$, so we employ hyper-reduction through the discrete empirical interpolation method (DEIM) [43] and its matrix version (MDEIM) [44,45] to compute approximate affine decompositions as [38]

$$\mathbf{f}(\boldsymbol{\mu}) \approx \mathbf{f}_m(\boldsymbol{\mu}) = \sum_{k=1}^{M_{\mathrm{f}}} \theta_k^{\mathrm{f}}(\boldsymbol{\mu})\mathbf{f}_k\,, \qquad \mathbf{A}(\boldsymbol{\mu}) \approx \mathbf{A}_m(\boldsymbol{\mu}) = \sum_{k=1}^{M_A} \Theta_k^a(\boldsymbol{\mu})\mathbf{A}_k,$$

where $\mathbf{f}_k$, $k = 1, \ldots, M_{\mathrm{f}}$, and $\mathbf{A}_k$, $k = 1, \ldots, M_A$ are precomputable vectors and matrices, respectively, and independent of $\boldsymbol{\mu}$. In this way, we can approximate the ROM arrays as

$$\mathbf{f}_r(\boldsymbol{\mu}) \approx \mathbf{f}_r^m(\boldsymbol{\mu}) = \sum_{k=1}^{M_{\mathrm{f}}} \theta_k^{\mathrm{f}}(\boldsymbol{\mu})\mathbf{f}_r^k\,, \qquad \mathbf{A}_r(\boldsymbol{\mu}) \approx \mathbf{A}_r^m(\boldsymbol{\mu}) = \sum_{k=1}^{M_{\mathrm{a}}} \theta_k^{\mathrm{a}}(\boldsymbol{\mu})\mathbf{A}_r^k,$$

where $\mathbf{f}_r^k = \mathbf{W}^{\mathrm{T}}\mathbf{f}_k \in \mathbb{R}^N$, $k = 1, \ldots, M_{\mathrm{f}}$, and $\mathbf{A}_r^k = \mathbf{W}^{\mathrm{T}}\mathbf{A}^k\mathbf{V} \in \mathbb{R}^{N\times N}$, $k = 1, \ldots, M_A$. Taking advantage of hyper-reduction, we recover the following hyper reduced order model: find $\mathbf{p}_r^m(\boldsymbol{\mu}) \in \mathbb{R}^r$ s.t.

$$\mathbf{A}_r^m(\boldsymbol{\mu})\mathbf{p}_r^m(\boldsymbol{\mu}) = \mathbf{f}_r^m(\boldsymbol{\mu}). \tag{21}$$

Due to its small dimension, the solution of the system (21) can be very fast and computationally inexpensive, allowing us to generate many instances of the output of interest, which can be used as LF training data. Finally, we denote by $p_r^m(\boldsymbol{\mu}) \in V_h$ the finite element approximation of the problem corresponding to the vector $\mathbf{V}\mathbf{p}_r^m(\boldsymbol{\mu}) \in \mathbb{R}^n$.

References

[1] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (1998) 2278–2324.

[2] M. Chen, Y. Li, R. Li, Research on neural machine translation model, J. Phys. Conf. Ser. 1237 (2019) 052020.

[3] S. Ghosh, D.L. Reilly, Credit card fraud detection with a neural-network, in: 1994 Proceedings of the Twenty-Seventh Hawaii International Conference on System Sciences, Vol. 3, 1994, pp. 621–630.

[4] N. Baker, F. Alexander, T. Bremer, A. Hagberg, Y. Kevrekidis, H. Najm, M. Parashar, A. Patra, J. Sethian, S. Wild, et al., Workshop Report on Basic Research Needs for Scientific Machine Learning: core Technologies for Artificial Intelligence, Technical Report, USDOE Office of Science (SC), Washington, DC (United States), 2019.

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

[6] E. Weinan, J. Han, A. Jentzen, Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations, Commun. Math. Stat. 5 (4) (2017) 349–380.

[7] J. Sirignano, K. Spiliopoulos, Dgm: A deep learning algorithm for solving partial differential equations, J. Comput. Phys. 375 (2018) 1339–1364.

[8] J.S. Hesthaven, S. Ubbiali, Non-intrusive reduced order modeling of nonlinear problems using neural networks, J. Comput. Phys. 363 (2018) 55–78.

[9] K. Lee, K.T. Carlberg, Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders, J. Comput. Phys. 404 (2020) 108973.

[10] S. Fresca, L. Dedè, A. Manzoni, A comprehensive deep learning-based approach to reduced order modeling of nonlinear time-dependent parametrized PDEs, 2020, arXiv:2001.04001.

[11] D. Ray, J.S. Hesthaven, An artificial neural network as a troubled-cell indicator, J. Comput. Phys. 367 (2018) 166–191.

[12] F. Regazzoni, L. Dedè, A. Quarteroni, Machine learning of multiscale active force generation models for the efficient simulation of cardiac electromechanics, Comput. Methods Appl. Mech. Engrg. 370 (2020) 113268.
[13] Y. Wang, Q. Yao, J.T. Kwok, L.M. Ni, Generalizing from a few examples: A survey on few-shot learning, ACM Comput. Surv. 53 (3) (2020) 1–34.
[14] Z.-H. Zhou, A brief introduction to weakly supervised learning, Natl. Sci. Rev. 5 (1) (2018) 44–53.
[15] A.G. Journel, Fundamentals of Geostatistics in Five Lessons, in: Short Courses in Geology, vol. 8, American Geophysical Union (AGU), 1989, pp. 10–15.
[16] M. Raissi, P. Perdikaris, G.E. Karniadakis, Inferring solutions of differential equations using noisy multi-fidelity data, J. Comput. Phys. 335 (2017) 736–746.
[17] M. Kast, M. Guo, J.S. Hesthaven, A non-intrusive multifidelity method for the reduced order modeling of nonlinear problems, Comput. Methods Appl. Mech. Engrg. 364 (2020) 112947.
[18] B. Peherstorfer, K. Willcox, M. Gunzburger, Survey of multifidelity methods in uncertainty propagation, inference, and optimization, SIAM Rev. 60 (3) (2018) 550–591.
[19] A. O'Hagan, M.C. Kennedy, Predicting the output from a complex computer code when fast approximations are available, Biometrika 87 (1) (2000) 1–13.
[20] M.A. Álvarez, L. Rosasco, N.D. Lawrence, Kernels for vector-valued functions: A review, Found. Trends® Mach. Learn. 4 (3) (2012) 195–266.
[21] X. Meng, G.E. Karniadakis, A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse PDE problems, J. Comput. Phys. 401 (2019).
[22] M. Motamed, A multi-fidelity neural network surrogate sampling method for uncertainty quantification, Int. J. Uncert. Quantif. 10 (4) (2020) 315–332.
[23] R.C. Aydin, F.A. Braeu, C.J. Cyron, General multi-fidelity framework for training artificial neural networks with computational models, Front. Mater. 6 (2019) 61.
[24] D. Liu, Y. Wang, Multi-fidelity physics-constrained neural network and its application in materials modeling, J. Mech. Des. 141 (12) (2019).
[25] M. Raissi, G. Karniadakis, Deep multi-fidelity Gaussian processes, 2016, arXiv:1604.07484.
[26] K. Cutajar, M. Pullin, A. Damianou, N. Lawrence, J. González, Deep gaussian processes for multi-fidelity modeling, 2019, arXiv:1903.07320.
[27] X. Meng, H. Babaee, G.E. Karniadakis, Multi-fidelity Bayesian neural networks: Algorithms and applications, J. Comput. Phys. 438 (2021) 110361.
[28] G. Strang, Linear Algebra and Learning from Data, Wellesley-Cambridge Press, 2019.
[29] J. Lee, Y. Bahri, R. Novak, S.S. Schoenholz, J. Pennington, J. Sohl-Dickstein, Deep neural networks as gaussian processes, 2017, ArXiv Preprint No. 1711.00165.
[30] R.M. Neal, Bayesian Learning for Neural Networks, Vol. 118, Springer Science & Business Media, 2012.
[31] R. Durrett, Probability: Theory and Examples, Vol. 49, Cambridge University Press, 2019.
[32] M. Guo, A brief note on understanding neural networks as Gaussian processes, 2021, arXiv:2107.11892.
[33] J. Friedman, Greedy function approximation: A gradient boosting machine, Ann. Statist. 29 (5) (2001) 1189–1232.
[34] C. Williams, C. Rasmussen, Gaussian Processes for Machine Learning, the MIT Press, 2006.
[35] J. Bergstra, D. Yamins, D.D. Cox, Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures, in: Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28, in: ICML'13, JMLR.org, 2013, pp. 115–123.
[36] J.S. Bergstra, R. Bardenet, Y. Bengio, B. Kégl, Algorithms for hyper-parameter optimization, in: Advances in Neural Information Processing Systems, 2011, pp. 2546–2554.
[37] GPy, GPY: A Gaussian process framework in python, since 2012, http://github.com/SheffieldML/GPy.
[38] F. Negri, A. Manzoni, D. Amsallem, Efficient model reduction of parametrized systems by matrix discrete empirical interpolation, J. Comput. Phys. 303 (2015) 431–454.
[39] E. Bängtsson, D. Noreland, M. Berggren, Shape optimization of an acoustic horn, Comput. Methods Appl. Mech. Engrg. 192 (11–12) (2003) 1533–1571.
[40] F. Negri, RedbKIT version 2.2, 2016, http://redbkit.github.io/redbKIT/.
[41] J.S. Hesthaven, G. Rozza, B. Stamm, Certified Reduced Basis Methods for Parametrized Partial Differential Equations, Springer International Publishing, 2016.
[42] A. Quarteroni, A. Manzoni, F. Negri, Reduced Basis Methods for Partial Differential Equations. An Introduction, Springer International Publishing, 2016.
[43] S. Chaturantabut, D.C. Sorensen, Nonlinear model reduction via discrete empirical interpolation, SIAM J. Sci. Comput. 32 (5) (2010) 2737–2764.
[44] K.T. Carlberg, R. Tuminaro, P. Boggs, Preserving Lagrangian structure in nonlinear model reduction with application to structural dynamics, SIAM J. Sci. Comput. 37 (2) (2015) B153–B184.
[45] P. Benner, S. Gugercin, K. Willcox, A survey of projection-based model reduction methods for parametric dynamical systems, SIAM Rev. 57 (4) (2015) 483–531.