Available online at www.sciencedirect.com

ScienceDirect

Comput. Methods Appl. Mech. Engrg. 415 (2023) 116207

Computer methods in applied mechanics and engineering

www.elsevier.com/locate/cma

Probabilistic neural data fusion for learning from an arbitrary number of multi-fidelity data sets

Carlos Mora[a,1], Jonathan Tammer Eweis-Labolle[a,1], Tyler Johnson[a], Likith Gadde[b], Ramin Bostanabad[a,*]

[a] Department of Mechanical and Aerospace Engineering, University of California, Irvine, United States of America
[b] Northwood High School, Irvine, United States of America

Received 29 January 2023; received in revised form 5 June 2023; accepted 21 June 2023
Available online xxxx

Abstract

In many applications in engineering and sciences analysts have simultaneous access to multiple data sources. In such cases, the overall cost of acquiring information can be reduced via data fusion or multi-fidelity (MF) modeling where one leverages inexpensive low-fidelity (LF) sources to reduce the reliance on expensive high-fidelity (HF) data. In this paper, we employ neural networks (NNs) for data fusion in scenarios where data is very scarce and obtained from an arbitrary number of sources with varying levels of fidelity and cost. We introduce a unique NN architecture that converts MF modeling into a nonlinear manifold learning problem. Our NN architecture inversely learns non-trivial (e.g., non-additive and non-hierarchical) biases of the LF sources in an interpretable and visualizable manifold where each data source is encoded via a low-dimensional distribution. This probabilistic manifold quantifies model form uncertainties such that LF sources with small bias are encoded close to the HF source. Additionally, we endow the output of our NN with a parametric distribution not only to quantify aleatoric uncertainties, but also to reformulate the network's loss function based on strictly proper scoring rules which improve robustness and accuracy on unseen HF data. Through a set of analytic and engineering examples, we demonstrate that our approach provides a high predictive power while quantifying various sources of uncertainty. Our codes and examples can be accessed via GitLab.

Keywords: Multi-fidelity modeling; Uncertainty quantification; Bayesian neural networks; Inverse problems; Manifold learning; Data fusion

1. Introduction

In an increasing number of applications in engineering and sciences analysts have simultaneous access to multiple sources of information. For instance, material properties can be estimated via multiple techniques such as (in decreasing order of cost and accuracy/fidelity) experiments, direct numerical simulations (DNS), a host of physics-based reduced order models (ROMs), or analytical methods [1–5]. In such applications, the overall cost of gathering information about the system of interest can be reduced via *multi-fidelity* (MF) *modeling* or *data fusion* where

* Corresponding author.
E-mail address: Raminb@uci.edu (R. Bostanabad).
[1] Equal contribution.

https://doi.org/10.1016/j.cma.2023.116207
0045-7825/© 2023 Elsevier B.V. All rights reserved.

one leverages inexpensive low-fidelity (LF) sources to reduce the reliance on expensive high-fidelity (HF) data sources. In this paper, we employ neural networks (NNs) for MF modeling in scenarios where data is scarce and obtained from multiple sources with varying levels of fidelity and cost (i.e., data is unbalanced since more samples are available from cheaper sources). In particular, our contributions are as follows (1) we introduce a unique NN architecture that not only facilitates data fusion, but also quantifies and visualizes the discrepancies/similarities between all data sources, and (2) we illustrate that a Bayesian treatment, besides alleviating overfitting and providing a probabilistic surrogate (i.e., an emulator), provides the means to develop a novel loss function (based on proper scoring rules) that improves the performance and robustness of the resulting MF NN emulator.

Over the past few decades, many techniques have been developed for building MF surrogates which are used in outer-loop applications such as design optimization [6,7], calibration of computer models [8], or Bayesian optimization [9]. The main motivation behind these techniques is to leverage the correlations between LF and HF data sources (and the fact that sampling from the former is typically cheaper) to improve the predictive performance of the surrogate while reducing the overall data acquisition costs. Early works in this field focused primarily on hierarchically linking bi-fidelity data. For instance, in space mapping [10–12] or multi-level [13–15] techniques the inputs of the LF data are mapped following formulations such as $\boldsymbol{x}^l = F(\boldsymbol{x}^h)$ where $\boldsymbol{x}^l$ and $\boldsymbol{x}^h$ are the inputs of LF and HF sources, respectively. In this equation, $F(\cdot)$ is a transformation function whose predefined functional form is calibrated such that $y^l(F(\boldsymbol{x}^h))$ approximates $y^h(\boldsymbol{x}^h)$ as closely as possible. These techniques are useful in applications where higher fidelity data are obtained by successively refining the discretization in simulations [13,14], e.g., by refining the mesh when modeling the flow around an airfoil or estimating the fracture toughness of a microstructure. The main disadvantages of space mapping techniques are that (1) they rely on iterative and time-consuming analysis for choosing a near-optimal functional form for $F(\cdot)$, (2) they cannot jointly fuse more than two data sources at a time, (3) they quantify similarity/discrepancy between the sources based on pre-defined functions whose space may not include the true discrepancy, and (4) they do not quantify some uncertainty sources (such as lack of data) and are rarely formulated within a Bayesian setting that leverages prior information.

A well-known hierarchical bi-fidelity modeling framework is that of Kennedy and O'Hagan (KOH) [16] who assume that the discrepancy between the LF and HF sources is additive (multiplicative terms have also been explored [17]) and that both sources as well as the discrepancy between them can be modeled via Gaussian processes (GPs). Upon this modeling assumption, KOH find the joint posterior of GPs' hyperparameters via either fully [18,19] or modular Bayesian inference [20–23]. While KOH's approach considers multiple uncertainties and has been successfully applied to a broad range of applications [24–26], it has three main limitations: (1) it only accommodates two data sources at a time, (2) it places a priori independence assumption between the GPs, and (3) it does not provide a low-dimensional, visualizable, and interpretable metric that quantifies the correlations between the data sources.

Recent works have acknowledged the limitations of hierarchical methods and devised new methodologies to address them. For instance, MF modeling can be achieved via a recursive scheme [27] where a bi-fidelity method is repeatedly applied from the lowest to the highest fidelities. However, such recursive schemes inherit the limitations of bi-fidelity methods, cannot *jointly* fuse multi-source data sets, and are sensitive to the ordering (i.e., the relative accuracy of all sources must be known a priori).

As another example, [28,29] present MF networks (MFNets): an approach based on directed acyclic graphs that builds a MF surrogate using an arbitrary number of data sources. MFNets accommodate noisy data and are trained via gradient-based minimization of a nonlinear least squares objective. Although MFNets tackle issues such as learning from non-hierarchically related data sources, they either rely on some prior knowledge on a set of latent variables that explain the relations between sources [29] or on iterative approaches for finding the optimal graph structure [28]. Additionally, MFNets cannot provide a learnt metric that quantifies the (potential) bias of LF sources with respect to the HF source and also they do not explicitly separate epistemic and aleatoric uncertainties.

Other notable works that have studied the limitations of hierarchical techniques include [30–32] which are focused on identifying (and correcting) non-additive discrepancies between LF and HF sources. However, the proposed solution in these works is intrusive and relies on some rather strong modeling assumptions that largely limit the applications. These limitations arise because the formulation of the discrepancy is learned via an embedded operator whose functional form and interaction with the LF source are constructed a priori.

We have recently developed a GP-based approach [33] that addresses the above issues by converting MF modeling into a manifold learning problem where the relations between the sources are automatically quantified via an appropriately learnt distance measure. The conversion is achieved via latent map Gaussian processes [33] (LMGPs, see Section 2.1) which enable GPs to handle categorical variables and, correspondingly, data fusion: by augmenting the inputs via a categorical variable (which indicates the source of a data point) and then concatenating all the data sets, LMGPs can simultaneously learn from an arbitrary number of information sources. We have shown [33] that LMGP-based MF modeling consistently outperforms KOH' approach and can also handle calibration problems.

Following the success of LMGPs in data fusion, in this work we examine the potentials of NNs in matching (and, hopefully, improving) LMGPs' efficiency in MF modeling. Our current studies are motivated by the facts that (1) when viewed as (probabilistic or deterministic) graphical models [34], NNs provide unique opportunities to use MF data sets to uncover complex hidden relations between the corresponding sources, (2) the recent hardware and software advancements have drastically accelerated architecture design and training of NNs, and (3) NNs scale to higher dimensions and big data significantly better than GPs.

Over the past few years some NN-based approaches have been developed for MF modeling [35–38]. However, most of these works design the network architecture primarily based on hierarchical methods and consequently inherit their limitations. For instance, [35] builds two sequentially connected deterministic networks based on KOH's method where the first and second NNs are tasked to emulate the LF and HF sources, respectively. In addition to sharing the limitations of KOH's method, such a sequential bi-fidelity NN requires that the two parts of the network are trained separately (unless the LF and HF training data are available at the same inputs) and also relies on manual tuning of the architecture and loss function. Similarly, [36] build a hierarchical bi-fidelity method that allows all-at-once training of the two sequentially connected NNs (regardless of data location) but their approach is also inspired by that of KOH and requires iterative fine tuning of the architecture and loss function. It has been argued [37] that such sequentially trained NNs bridge MF modeling with transfer learning where the knowledge gained from the LF data is used in building the NN module that surrogates the HF source.

Non-sequential NNs are rarely used for MF modeling (esp. with > 2 sources) due to the fact that searching for the optimum architecture (and effectively training it with small data) is a difficult task. We address this challenge by drawing inspiration from LMGPs where we design the architecture such that any number of MF data sets can be simultaneously fused and the overall discrepancies between sources are quantified with visualizable metrics. We also illustrate that making specific parts of the network probabilistic, in addition to being superior to both deterministic and all-probabilistic NNs, enables us to infuse a proper scoring rule [39] into the loss function and, in turn, improve the performance of the MF emulator. The particular rule that we adopt is the negatively oriented interval score which is frequently used in testing the quality of probabilistic predictions but, to the best of our knowledge, has never been used in the training stage of a probabilistic NN. In summary, our major contributions are as follows:

- We introduce a unique NN architecture for MF modeling that can fuse an arbitrary number of data sets and quantify both epistemic and aleatoric uncertainties.
- We inversely learn the accuracy of the LF sources (with respect to the HF source) and visualize the learned relations in an interpretable manifold.
- We show that a probabilistic setting allows us to develop a novel loss function (based on proper scoring rules) that improves the performance of the emulator.
- We validate the performance of our approach on analytical and real-world examples and show that it performs on par with the state of the art while providing improved scalability to high dimensions and big data.

The rest of this paper is organized as follows. We review the relevant technical background in Section 2 and then introduce our approach in Section 3. We test the performance of our approach on a host of analytical problems and real-world data sets in Section 4 and conclude the paper in Section 5.

2. Technical preliminaries

In this section we first review LMGPs which are extensions of GPs that handle categorical inputs and, thus, can readily fuse any number of data sets. Then, we provide some background on Bayesian neural networks (BNNs) which form the foundation of our neural data fusion framework.

2.1. Latent map Gaussian processes (LMGPs)

Let us denote the output and inputs in the training data by $y \in \mathcal{Y} \equiv \mathbb{R}$ and $\boldsymbol{x} = [x_1, x_2, \ldots, x_{dx}]^T \in \mathbb{R}^{dx}$, respectively, with an individual training point $i = 1, \ldots, n$ denoted by the pair $(y^{(i)}, \boldsymbol{x}^{(i)})$. Assume the training data is a realization from a constant-mean[2] GP and that the following relation holds:

$$y(\boldsymbol{x}) = m + \xi(\boldsymbol{x}) \tag{1}$$

where m is the unknown constant mean and $\xi(\boldsymbol{x})$ is a zero-mean GP whose covariance function or kernel is:

$$\operatorname{cov}(\xi(\boldsymbol{x}), \xi(\boldsymbol{x}')) = c(\boldsymbol{x}, \boldsymbol{x}') = s^2 r(\boldsymbol{x}, \boldsymbol{x}') \tag{2}$$

where s^2 is the variance of the process and $r(\cdot, \cdot)$ is a parametric correlation function such as the Gaussian:

$$r(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\sum_{i=1}^{dx} 10^{\omega_i}(x_i - x_i')^2\right) = \exp\left(-\left(\boldsymbol{x} - \boldsymbol{x}'\right)^T 10^{\boldsymbol{\Omega}} \left(\boldsymbol{x} - \boldsymbol{x}'\right)\right) \tag{3}$$

where $\boldsymbol{\omega} = [\omega_1, \ldots, \omega_{dx}]^T$ are the roughness or scale parameters and $\boldsymbol{\Omega} = diag(\boldsymbol{\omega})$.

The training process and prediction formulas for a GP depend on the choice of the correlation function, which relies on a weighted Cartesian distance metric between any two inputs, see Eq. (3). As we recently motivated in [40], to directly use GPs for mixed-variable modeling we reformulate $r(\cdot, \cdot)$ as detailed below such that it can handle categorical (qualitative) inputs.

Let us denote the categorical inputs by $\boldsymbol{t} = [t_1, \ldots, t_{dt}]^T$ where the total number of distinct levels for qualitative variable t_i is τ_i. To handle mixed inputs, LMGP learns a parametric function that maps categorical variables to some points in a quantitative manifold or latent space.[3] These points (and hence the mapping function) can be incorporated into any standard correlation function, such as the Gaussian, which is reformulated as follows for mixed inputs:

$$r\left((\boldsymbol{x}, \boldsymbol{t}), (\boldsymbol{x}', \boldsymbol{t}')\right) = \exp\left(-\left\|\boldsymbol{z}(\boldsymbol{t}) - \boldsymbol{z}(\boldsymbol{t}')\right\|_2^2 - \left(\boldsymbol{x} - \boldsymbol{x}'\right)^T 10^{\boldsymbol{\Omega}} \left(\boldsymbol{x} - \boldsymbol{x}'\right)\right) \tag{4}$$

or, equivalently,

$$r\left((\boldsymbol{x}, \boldsymbol{t}), (\boldsymbol{x}', \boldsymbol{t}')\right) = \exp\left(-\sum_{i=1}^{dx} 10^{\omega_i}(x_i - x_i')^2\right) \times \exp\left(-\sum_{i=1}^{dz}(z_i(\boldsymbol{t}) - z_i(\boldsymbol{t}'))^2\right) \tag{5}$$

where $\|\cdot\|_2$ denotes the Euclidean 2-norm and $\boldsymbol{z}(\boldsymbol{t}) = [z_1(\boldsymbol{t}), \ldots, z_{dz}(\boldsymbol{t})]_{1\times dz}$ is the to-be-learned latent space point corresponding to the particular combination of categorical variables denoted by $\boldsymbol{t}$. To find these points in the latent space, LMGP assigns a unique vector (i.e., a prior representation) to each combination of categorical variables. Then, it uses matrix multiplication[4] to map each of these vectors to a point in a latent space of dimension dz:

$$\boldsymbol{z}(\boldsymbol{t}) = \boldsymbol{\zeta}(\boldsymbol{t})\boldsymbol{A} \tag{6}$$

where $\boldsymbol{\zeta}(\boldsymbol{t})$ is the $1 \times \sum_{i=1}^{dt} \tau_i$ unique prior vector representation of $\boldsymbol{t}$ and $\boldsymbol{A}$ is a $\sum_{i=1}^{dt} \tau_i \times dz$ matrix that maps $\boldsymbol{\zeta}(\boldsymbol{t})$ to $\boldsymbol{z}(\boldsymbol{t})$. In this paper, we use $dz = 2$ since it simplifies visualization and has been shown to provide sufficient flexibility for learning the latent relations [40]. We construct $\boldsymbol{\zeta}$ via a form of one-hot encoding where we first construct the $1 \times \tau_i$ vector $\boldsymbol{v}^i = \left[v_1^i, v_2^i, \ldots, v_{\tau_i}^i\right]$ for each categorical variable t_i such that $v_j^i = 1$ when t_i is at level $k = j$ and $v_j^i = 0$ when t_i is at level $k \neq j$ for $k \in 1, 2, \ldots, \tau_i$. Then, we set $\boldsymbol{\zeta}(\boldsymbol{t}) = [\boldsymbol{v}^1, \boldsymbol{v}^2, \ldots, \boldsymbol{v}^{d_t}]$. For example, for the two categorical variables t_1 and t_2 with 2 and 3 levels, $\boldsymbol{\zeta}(\boldsymbol{t}) = [0, 1, 0, 1, 0]$ encodes the combination where both variables are at level 2.

To train an LMGP, we use maximum likelihood estimation (MLE) to jointly estimate all of its parameters:

$$\left[\hat{m}, \hat{s}^2, \hat{\boldsymbol{\omega}}, \hat{\boldsymbol{A}}\right] = \underset{m, s^2, \boldsymbol{\omega}, \boldsymbol{A}}{\operatorname{argmax}} \quad \left|2\pi s^2 \boldsymbol{R}\right|^{-\frac{1}{2}} \times \exp\left(-\frac{1}{2}(\boldsymbol{y} - \boldsymbol{1}m)^T (s^2\boldsymbol{R})^{-1}(\boldsymbol{y} - \boldsymbol{1}m)\right) \tag{7}$$

[2] GPs (and LMGPs) can also be formulated by using a linear combination of basis functions in place of the constant mean. This formulation relies on prior knowledge of the functional form of the output and can improve performance in extrapolation, see [33].

[3] Multiple mapping functions can also be used to build multiple manifolds. We leverage this in Section 4 where we build two manifolds for data fusion problems with categorical or mixed inputs.

[4] More complex transformations based on, e.g., NNs, may also be used, although we do not do so in this paper.

where $|\cdot|$ denotes the determinant operator, $\boldsymbol{y} = [y_1, \ldots, y_n]^T$ is the $n \times 1$ vector of outputs in the training data, $\boldsymbol{R}$ is the $n \times n$ correlation matrix with the (i, j)th element $R_{ij} = r\left((\boldsymbol{x}^{(i)}, \boldsymbol{t}^{(i)}), (\boldsymbol{x}^{(j)}, \boldsymbol{t}^{(j)})\right)$ for $i, j = 1, \ldots, n$, and $\mathbf{1}$ is a $n \times 1$ vector of ones.

After estimating the hyperparameters, we use the conditional distribution formulas to predict the response distribution at the arbitrary point $\boldsymbol{p}^* = (\boldsymbol{x}^*, \boldsymbol{t}^*)$. The mean and variance of this normal distribution are:

$$\mathbb{E}\left[y\left(\boldsymbol{p}^*\right)\right] = \hat{m} + \boldsymbol{r}^T\left(\boldsymbol{p}^*\right)\boldsymbol{R}^{-1}\left(\boldsymbol{y} - \mathbf{1}\hat{m}\right) \tag{8}$$

$$\operatorname{cov}\left(y(\boldsymbol{p}^*), y(\boldsymbol{p}^{'})\right) = \hat{s}^2 r(\boldsymbol{p}^*, \boldsymbol{p}^{'}) = \hat{s}^2\left\{1 - \boldsymbol{r}^T(\boldsymbol{p}^*)\boldsymbol{R}^{-1}\boldsymbol{r}(\boldsymbol{p}^{'}) + g(\boldsymbol{p}^*)(\mathbf{1}^T\boldsymbol{R}^{-1}\mathbf{1})^{-1}g(\boldsymbol{p}^{'})\right\} \tag{9}$$

where $\mathbb{E}$ denotes expectation, $\boldsymbol{r}\left(\boldsymbol{p}^*\right)$ is an $(n \times 1)$ vector with the ith element $r\left(\boldsymbol{p}^{(i)}, \boldsymbol{p}^*\right)$, and $g\left(\boldsymbol{p}^*\right) = 1 - \mathbf{1}^T\boldsymbol{R}^{-1}\boldsymbol{r}\left(\boldsymbol{p}^*\right)$.

To perform data fusion via LMGP, we re-frame multi-fidelity modeling as a manifold learning problem. Assume that we have ds data sources whose inputs and outputs are denoted by $\boldsymbol{x}^{s_i}, y^{s_i}$, respectively, with $i = 1, \ldots, ds$. We first pre-process the data by appending the inputs with a single categorical variable t^s with ds levels (hereafter referred to as the source index variable) that distinguishes the data sources. Specifically, we add t^s at level i for source s_i, i.e., $\boldsymbol{x}^{s_i} \rightarrow \left[\boldsymbol{x}^{s_i}, \boldsymbol{i}_{n^{s_i}\times 1}\right]$, where $\boldsymbol{i}_{n^{s_i}\times 1}$ is an $n^{s_i} \times 1$ vector of i's and n^{s_i} is the number of data points for source s_i. We then combine the data for all sources into one unified data set and fit an LMGP directly to it, i.e., we fit LMGP to *all* of the data from *all* sources at once.

The fitted LMGP can provide predictions for any desired data source based on the level used for t^s and as such is an emulator for all of the data sources. Additionally, since the data sources are distinguished via a categorical variable, LMGP learns the correlations between them via a visualizable latent representation and uses these correlations to improve its predictions [33]. In the case that the raw inputs contain categorical variables $\boldsymbol{t}^c$, we use separate mappings for t^s and $\boldsymbol{t}^c$, i.e., we assign unique priors $\boldsymbol{\zeta}(t^s)$ and $\boldsymbol{\zeta}(\boldsymbol{t}^c)$ which LMGP uses to find mapping matrices $\boldsymbol{A}^s$ and $\boldsymbol{A}^c$. The latent points corresponding to each mapping are then $\boldsymbol{z}^s$ and $\boldsymbol{z}^c$, respectively.

Note that the correlation function in Eq. (5) depends directly on the Euclidean distance between a pair of latent points. This means that relative distances in the latent space directly correspond to correlations, e.g., if a pair of data sources y^{s_1} and y^{s_2} have corresponding latent points with a distance Δ in the latent space then this directly implies by Eq. (5) that LMGP has found those two sources to have a correlation of $\exp(-\Delta^2)$.

2.2. Bayesian neural networks

Feedforward neural networks (FFNNs) are one of the most common models used in deep learning and their main goal is to learn the underlying function $f(\boldsymbol{x})$ that maps the inputs $\boldsymbol{x}$ to the target y [41]. To this end, an FFNN defines the mapping $\hat{f}(\boldsymbol{x}; \boldsymbol{\theta})$ whose parameters $\boldsymbol{\theta}$ are estimated such that $\hat{y} = \hat{f}(\boldsymbol{x}; \boldsymbol{\theta})$ best approximates $f(\boldsymbol{x})$. NN-based approaches for MF emulation can provide attractive advantages since they are universal function approximators [42] and can handle high-dimensional inputs and large data sets. In this subsection, we first briefly describe the working principle of FFNNs and then motivate the use of BNNs and Bayes by backprop [43].

FFNNs propagate information from the inputs $\boldsymbol{x}$ to the output y through intermediate computations that define $\hat{f}$. They are traditionally built via a succession of L layers where $L-2$ hidden layers are placed between the input and output layers. The output of layer k is denoted by z_k and is obtained as follows:

$$z_1 = \boldsymbol{x}, \tag{10}$$

$$z_k = \phi_k\left(\boldsymbol{W}_k z_{k-1} + \boldsymbol{b}_k\right) \quad \forall\, k \in [2, L-1], \tag{11}$$

$$\hat{y} = \phi_L\left(\boldsymbol{W}_L z_{L-1} + b_L\right) \tag{12}$$

where ϕ is the (typically non-linear) activation function. The parameters $\boldsymbol{\theta}_k = (\boldsymbol{W}_k, \boldsymbol{b}_k)$, where $\boldsymbol{W}_k$ and $\boldsymbol{b}_k$ are the weight matrices and bias vectors, respectively, correspond to the connections between the $(k-1)^{th}$ and kth layer. For brevity, we denote the parameters of the entire network by $\boldsymbol{\theta}$.

Since $\hat{f}(\boldsymbol{x}^{(i)}; \boldsymbol{\theta})$ approximates $y^{(i)}$, we can write:

$$y^{(i)} = f(\boldsymbol{x}^{(i)}) + \epsilon \approx \hat{f}(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}) + \epsilon \tag{13}$$

where $\epsilon \sim \mathcal{N}(0, \sigma^2)$ represents noise. Eq. (13) indicates that $P(y^{(i)}|\boldsymbol{x}^{(i)}) \sim \mathcal{N}(f(\boldsymbol{x}^{(i)}), \sigma^2)$ or $\mathbb{E}[y^{(i)}|\boldsymbol{x}^{(i)}] = f(\boldsymbol{x}^{(i)}) \approx \hat{f}(\boldsymbol{x}^{(i)}; \boldsymbol{\theta})$. Therefore, FFNNs can be seen as a statistical model with parameters $\boldsymbol{\theta}$ that aim to learn the expected conditional distribution $\mathbb{E}[y^{(i)}|\boldsymbol{x}^{(i)}]$. To that end, the conditional probability $P\left(\mathcal{D}|\boldsymbol{\theta}\right)$ is written as:

$$P(\mathcal{D}|\boldsymbol{\theta}) = \prod_{i=1}^{n} P(y^{(i)}|\boldsymbol{x}^{(i)}, \boldsymbol{\theta})P(\boldsymbol{x}^{(i)}) = \prod_{i=1}^{n} \mathcal{N}(y^{(i)}; \hat{y}^{(i)}, \sigma^2)P(\boldsymbol{x}^{(i)})$$
$$= \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2}\left(y^{(i)} - \hat{y}^{(i)}\right)^2\right)P(\boldsymbol{x}^{(i)}) \tag{14}$$

Since the likelihood function $L(\boldsymbol{\theta}) \equiv P(\mathcal{D}|\boldsymbol{\theta})$, the parameters $\boldsymbol{\theta}$ can be estimated by maximizing $L(\boldsymbol{\theta})$ (the dependence on $\boldsymbol{x}$ is dropped for brevity):

$$\boldsymbol{\theta}_{MLE} = \arg\max_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) \equiv \arg\min_{\boldsymbol{\theta}} \; -\log P(\mathcal{D}|\boldsymbol{\theta}) = \arg\min_{\boldsymbol{\theta}} \frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - \hat{f}(\boldsymbol{x}^{(i)}; \boldsymbol{\theta})\right)^2 \tag{15}$$

which is equivalent to minimizing the mean squared error (MSE) of the predictions $\hat{y} = \hat{f}(\boldsymbol{x}; \boldsymbol{\theta})$ with respect to the targets y. Eq. (15) can be updated via Bayes rule to consider prior knowledge on $\boldsymbol{\theta}$ in the optimization. These maximum a posteriori (MAP) estimates are obtained via:

$$\boldsymbol{\theta}_{MAP} = \arg\max_{\boldsymbol{\theta}} P\left(\boldsymbol{\theta}|\mathcal{D}\right) = \arg\max_{\boldsymbol{\theta}} \log P\left(\boldsymbol{\theta}|\mathcal{D}\right) = \arg\max_{\boldsymbol{\theta}} \left(\log P\left(\mathcal{D}|\boldsymbol{\theta}\right) + \log P(\boldsymbol{\theta})\right) \tag{16}$$

where the first term recovers MSE as in Eq. (15) and the second term depends on the prior distribution assigned to the parameters. Eq. (16) illustrates that Gaussian and Laplacian priors are equivalent to $L2$ and $L1$ regularization, respectively [41,43].

FFNNs are likely to overfit in scenarios where data is scarce. Additionally, they cannot directly quantify prediction uncertainty and are often overconfident in extrapolation [44]. BNNs are developed to address these issues [45,46]. In BNNs, the weights are endowed with probability distributions (rather than single point estimates) which naturally results in probabilistic predictions and can dramatically reduce overfitting via parameter regularization and model averaging.

Predictions via a BNN requires sampling from the posterior distribution of the parameters, i.e., $P\left(\boldsymbol{\theta}|\mathcal{D}\right)$, which does not have a closed form and is highly complex. Over the past few years, various techniques have been developed to obtain samples from $P\left(\boldsymbol{\theta}|\mathcal{D}\right)$ (or an approximation thereof). The most popular techniques are based on either Markov Chain Monte Carlo (MCMC) [47] or variational inference (VI) [48] which, unlike MCMC, learns an approximation of the posterior distribution.

Although MCMC methods are arguably the best techniques for sampling from the *exact* posterior, their lack of scalability makes them inefficient for BNNs of any practical size [49]. Hence, we employ Bayes by backprop [43] which is a variational method that approximates $P\left(\boldsymbol{\theta}|\mathcal{D}\right)$ with the parameterized distribution $q(\boldsymbol{\theta}|\boldsymbol{\varphi})$. The parameters $\boldsymbol{\varphi}$ are learned by minimizing the Kullback–Leibler (KL) divergence between the true and approximated posteriors:

$$\mathrm{KL}[q(\boldsymbol{\theta}|\boldsymbol{\varphi})\|P(\boldsymbol{\theta}|\mathcal{D})] = \int q(\boldsymbol{\theta}|\boldsymbol{\varphi})\log\left(\frac{q(\boldsymbol{\theta}|\boldsymbol{\varphi})}{P(\boldsymbol{\theta}|\mathcal{D})}\right)d\boldsymbol{\theta} = \int q(\boldsymbol{\theta}|\boldsymbol{\varphi})\log\left(\frac{q(\boldsymbol{\theta}|\boldsymbol{\varphi})P(\mathcal{D})}{P(\mathcal{D}|\boldsymbol{\theta})P(\boldsymbol{\theta})}\right)d\boldsymbol{\theta}$$
$$= \int q(\boldsymbol{\theta}|\boldsymbol{\varphi})\log P(\mathcal{D})d\boldsymbol{\theta} + \int q(\boldsymbol{\theta}|\boldsymbol{\varphi})\log\left(\frac{q(\boldsymbol{\theta}|\boldsymbol{\varphi})}{P(\boldsymbol{\theta})}\right)d\boldsymbol{\theta} - \int q(\boldsymbol{\theta}|\boldsymbol{\varphi})\log P(\mathcal{D}|\boldsymbol{\theta})d\boldsymbol{\theta}$$
$$= \log P(\mathcal{D}) + \mathrm{KL}[q(\boldsymbol{\theta} \parallel \boldsymbol{\varphi})|P(\boldsymbol{\theta})] - \mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{\varphi})}[\log P(\mathcal{D}|\boldsymbol{\theta})] \tag{17}$$

where Bayes rule is applied to $P(\boldsymbol{\theta}|\mathcal{D})$ in the first line. Then, the parameters $\boldsymbol{\varphi}$ are estimated by minimizing Eq. (17):

$$\boldsymbol{\varphi}^* = \underset{\boldsymbol{\varphi}}{\operatorname{argmin}}\,\mathrm{KL}[q(\boldsymbol{\theta}|\boldsymbol{\varphi})\|P(\boldsymbol{\theta}|\mathcal{D})] = \underset{\boldsymbol{\varphi}}{\operatorname{argmin}}\,\mathrm{KL}[q(\boldsymbol{\theta}|\boldsymbol{\varphi})\|P(\boldsymbol{\theta})] - \mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{\varphi})}[\log P(\mathcal{D}|\boldsymbol{\theta})] \tag{18}$$

where the term $\log P(\mathcal{D})$ is excluded as it is constant. Eq. (18) aims to minimize the sum of two terms. The second term corresponds to the expectation of the negative log-likelihood while the first term acts as a regularizer and corresponds to the KL divergence between the approximated posterior and the prior.

3. Probabilistic neural data fusion

Designing a multi-fidelity NN that leverages an ensemble of LF data sets to better learn an HF source is a very challenging task because of the following major reasons:

1. The relations among the data sources can be unknown. For instance, in the Rational example (see Table 4 in Appendix A) there are three LF sources whose biases are *not* additive. Additionally, these LF sources are not hierarchically ordered in the sense that the second LF source is more accurate than the first one.
2. There are typically (but not always) more LF data available since LF sources are generally cheaper compared to the HF source. Learning from such an unbalanced MF data is quite difficult especially in the presence of scarce HF data (as an example, see the sample sizes for the engineering applications described in Appendix B).
3. NNs can be built in many ways and, as shown in Section 4, their performance heavily depends on their architecture and training mechanism. Building an optimum[5] NN with small, unbalanced, and MF data is even more difficult since the sensitivity to the architecture and training mechanism considerably increases.

We propose to address the above challenges by converting MF modeling to a manifold[6] learning problem which is then solved via an NN. We design the architecture, loss function, and training mechanism of this NN with a particular focus on uncertainty sources that include data scarcity (especially HF samples), noise with unknown variance (which can affect any of the data sources), non-trivial biases of LF sources, and data imbalances.

As schematically demonstrated in Fig. 1, we convert MF modeling to manifold learning by augmenting the input space with the categorical variable t^s whose levels (e.g., $\{'1', '2', \dots\}$ or $\{a, b, \dots\}$) indicate the source that generates a sample. We then map this *source indicator* variable to a low-dimensional manifold via a BNN (see Block 1 in Fig. 1). If the original input space has the categorical variables $\boldsymbol{t}^c$, we similarly map them to a manifold (but this time we use a deterministic NN, see Block 2 in Fig. 1). Afterwards, we combine the latent variables of these two manifolds with the quantitative inputs $\boldsymbol{x}$ via a deterministic NN, see Block 3 in Fig. 1. As opposed to the other two blocks, we require Block 3 to produce a normal probability distribution in order to capture aleatoric uncertainties. Finally, we train the entire network on the entire[7] data using our custom loss function that noticeably improves the prediction intervals.

In the following subsections, we elaborate on our rationale for designing a multi-block architecture and a custom loss function in Section 3.1 and Section 3.2, respectively. Then, we provide some details on the training and inference stages in Section 3.3.

3.1. Multi-block architecture

Each block of our network is designed to address particular challenges associated with MF modeling. Specifically, the BNN of Block 1 maps a quantitative prior representation $\boldsymbol{\zeta}(t^s)$ of the source indicator variable t^s to a continuous manifold $\boldsymbol{z}^s$. We design $\boldsymbol{\zeta}(t^s)$ by one-hot encoding t^s to merely inform the network about the source that generates a sample.[8] We build $\boldsymbol{z}^s$ based on a categorical variable because it forces the manifold to uncover the relations between sources (i.e., the levels of t^s). These relations are represented as distances in $\boldsymbol{z}^s$ where sources that produce similar data are encoded with close-by points (see Section 4 for multiple examples). This distance learning is in sharp contrast to existing approaches since (1) it does not assume there is any hierarchy between the data sources, (2) it is scalable to an arbitrary number of data sets, (3) it enables training the entire network via all available samples, (4) it is visualizable and interpretable which helps in identifying anomalous data sources, and (5) it does not assume any specific form (e.g., additive, multiplicative, etc.) for the biases of LF sources.

Block 1 is the only part of our network where the weights and biases are endowed with probability distributions. We make this choice to better learn model form errors and more accurately quantify the epistemic uncertainties due to lack of data and source-wise discrepancies. We note that, while the outputs of Block 1 do *not* parameterize a probability distribution, they are probabilistic by nature since they are obtained by propagating the deterministic vector $\boldsymbol{\zeta}(t^s)$ through some probabilistic hidden layers.

Block 2 is an FFNN that maps the quantitative prior representation $\boldsymbol{\zeta}(\boldsymbol{t}^c)$ of the categorical inputs $\boldsymbol{t}^c$ to the manifold $\boldsymbol{z}^c$ (Block 2 is omitted if the original inputs are purely quantitative). Similar to Block 1, we design $\boldsymbol{\zeta}(\boldsymbol{t}^c)$

[5] We measure optimality in terms of NN's error in predicting unseen data from the HF source.
[6] A manifold or a latent-space is a compact representation of a high-dimensional object such as an image.
[7] By entire, we mean the combined data sets from all sources.
[8] If there is some prior knowledge about the relation among the sources, $\boldsymbol{\zeta}(t^s)$ can be designed to reflect it. We do not pursue designing such informative priors in this work.

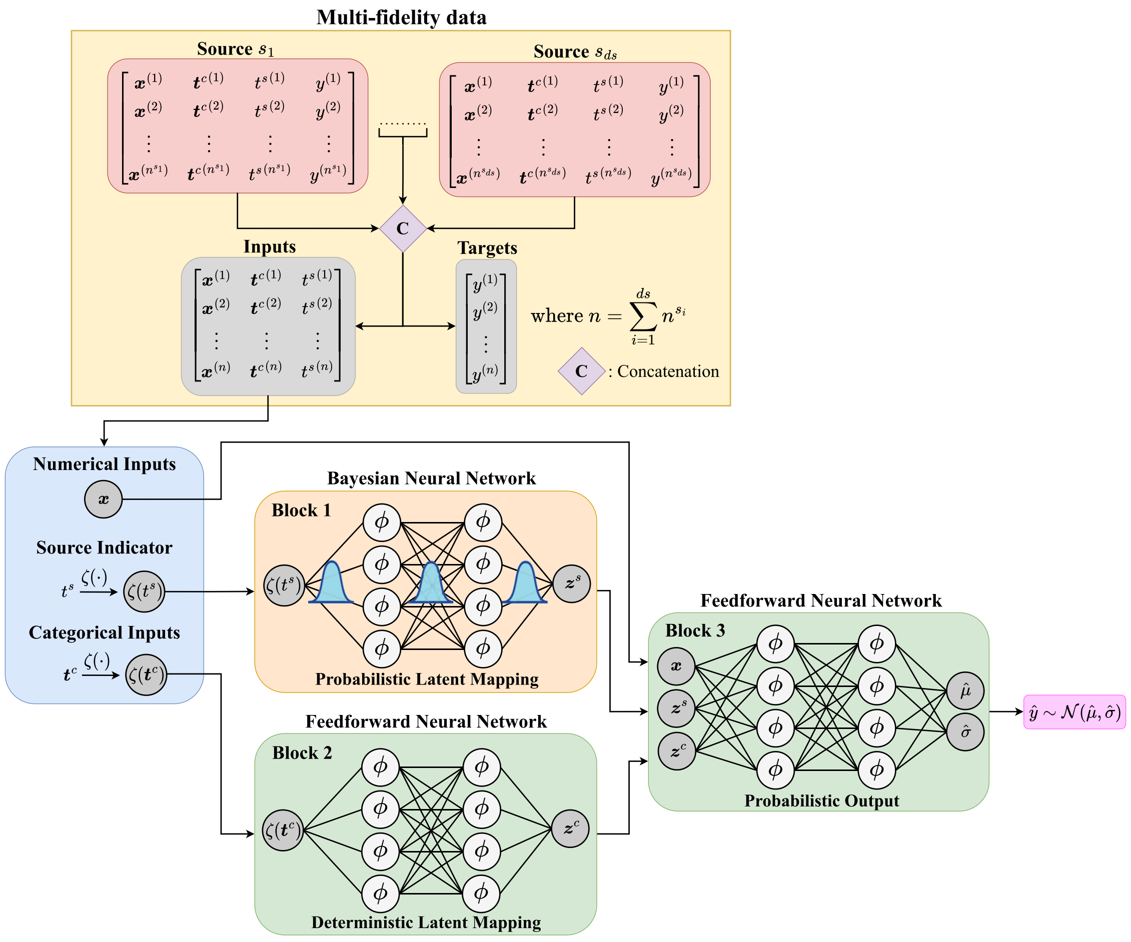
Multi-fidelity data
Source s_1
Source s_{ds}
C
Inputs
Targets
where $n = \sum_{i=1}^{ds} n^{s_i}$
C : Concatenation
Numerical Inputs
Source Indicator
Categorical Inputs
Bayesian Neural Network
Block 1
Probabilistic Latent Mapping
Feedforward Neural Network
Block 2
Deterministic Latent Mapping
Feedforward Neural Network
Block 3
Probabilistic Output
$\hat{y} \sim \mathcal{N}(\hat{\mu}, \hat{\sigma})$

Fig. 1. Probabilistic neural data fusion (Pro-NDF): The proposed architecture allows to combine an arbitrary number of sources by appending a source indicator variable to the data sets and then concatenating them. Pro-NDF consists of three blocks that perform separate tasks related to MF modeling: (1) Block 1 is a BNN that maps a quantitative prior representation of the source indicator $\boldsymbol{\zeta}(t^s)$ to a continuous manifold, (2) Block 2 is an FFNN that maps a quantitative prior representation of the categorical inputs $\boldsymbol{\zeta}(\boldsymbol{t}^c)$ to a continuous manifold, and (3) Block 3 is an FFNN with a probabilistic output that maps the numerical inputs and the latent variables to a parametric distribution.

via one-hot encoding and use deterministic outputs. However, unlike Block 1 we use a deterministic FFNN in Block 2 to map $\boldsymbol{\zeta}(\boldsymbol{t}^c)$ into $\boldsymbol{z}^c$. We make this decision to reduce the number of parameters and also because the meaning (and hence effects) of categorical inputs across different sources is typically the same.[9]

We set the manifold dimension to 2 for both Block 1 and Block 2, i.e., $dz^s = dz^c = 2$. While higher dimensions provide more learning capacities, our results in Section 4 and those reported elsewhere [50–55] indicate that low-dimensional manifolds are quite powerful in learning highly complex relations. For instance, [56] shows that a single latent variable can encode *smiling* in images of human faces which is a high-dimensional and complex feature in the original data space. Additionally, our choice simplifies the visualization of the manifolds and reduces the chances of overfitting since we are primarily interested in scarce data applications.

Block 3 is also an FFNN that maps the numerical inputs and the latent variables in both manifolds to a parametric distribution which represents the output. Block 3 has deterministic weights and biases since source-wise uncertainties are propagated to it via Block 1. However, we equip Block 3 with a probabilistic output [57] because it: (1) quantifies aleatoric uncertainties that are inherent to the data sets,[10] and (2) enables designing a multi-task loss that considers

[9] Due to severe discrepancies such as large model form errors, the effects of a categorical variable on the response may be quite different across the sources.

[10] The predicted variance also includes epistemic uncertainties that are propagated from Block 1, see Section 3.3.

the quality of the prediction intervals (detailed in Section 3.2). Additionally, Block 3 is responsible for learning the behavior for all data sources simultaneously, which allows it to leverage correlations between sources to augment predictions through a process akin to weight sharing.

3.2. Uncertainty-aware loss

NNs typically provide overconfident predictions especially when they are trained on small and unbalanced data. As explained in Section 3.1, we aim to address this issue by making Block 1 and the network's final output probabilistic. However, for these measures to work, we must develop an effective optimization[11] scheme where the loss function appropriately rewards prediction intervals (PIs) that are sufficiently wide (but not too wide) to cover unseen data (especially HF data). To design such a loss function, we draw inspiration from strictly proper scoring rules [39] and augment Eq. (18) with the negatively oriented interval score. Our loss is defined as:

$$\mathcal{L} = \mathcal{L}_{NLL} + \alpha_1 \mathcal{L}_{KL} + \alpha_2 \mathcal{L}_{IS} + \alpha_3 \mathcal{L}_2 \tag{19}$$

where $\mathcal{L}_{NLL}$ refers to the negative log-likelihood, $\mathcal{L}_{KL}$ is the KL divergence between the prior and the variational posterior distributions on the parameters (only applicable for the BNN from Block 1), $\mathcal{L}_{IS}$ denotes the interval score term, and $\mathcal{L}_2$ is $L2$ regularization (only applicable for deterministic NNs, i.e., Block 2 and 3). α_1, α_2 and α_3 are hyperparameters that, respectively, determine the relative strengths of $\mathcal{L}_{KL}$, $\mathcal{L}_{IS}$ and $\mathcal{L}_2$ compared to $\mathcal{L}_{NLL}$. The four terms in Eq. (19) are calculated as:

$$\mathcal{L}_{NLL} = -\frac{1}{N}\sum_{i=1}^{N} \log \mathcal{N}(y^{(i)}; \hat{\mu}^{(i)}, (\hat{\sigma}^{(i)})^2) \tag{20}$$

$$\mathcal{L}_{KL} = \mathrm{KL}[q(\boldsymbol{\theta}|\boldsymbol{\varphi}) \| P(\boldsymbol{\theta})] \tag{21}$$

$$\mathcal{L}_{IS} = \frac{1}{N}\sum_{i=1}^{N}[(\hat{u}^{(i)} - \hat{l}^{(i)}) + \frac{2}{\gamma}(\hat{l}^{(i)} - y^{(i)})\mathbb{1}\{y^{(i)} < \hat{l}^{(i)}\} + \frac{2}{\gamma}(y^{(i)} - \hat{u}^{(i)})\mathbb{1}\{y^{(i)} > \hat{u}^{(i)}\}] \tag{22}$$

$$\mathcal{L}_2 = |\boldsymbol{\theta}|^2 \tag{23}$$

where $\mathcal{L}_{KL}$ is computed via a Monte Carlo approximation, N is the batch size, and $\mathbb{1}\{\cdot\}$ denotes the indicator function that returns 1 if the event in brackets is true and 0 otherwise. The three terms of Eq. (19) compose a multi-task loss where: (1) the likelihood term $\mathcal{L}_{NLL}$ penalizes the model if the predicted distribution does not match the target distribution, (2) the KL divergence term $\mathcal{L}_{KL}$ favors variational posteriors that are similar to the assumed prior as per Eq. (18), and (3) the interval score term $\mathcal{L}_{IS}$ rewards narrow PIs while penalizing the model for each observation $y^{(i)}$ that lies outside the $(1-\gamma) \times 100\%$ prediction interval that spans the range $[\hat{l}^{(i)}, \hat{u}^{(i)}]$ where $\hat{l}^{(i)} = \hat{\mu}^{(i)} - 1.96\hat{\sigma}^{(i)}$ and $\hat{u}^{(i)} = \hat{\mu}^{(i)} + 1.96\hat{\sigma}^{(i)}$. In this paper, we use $\gamma = 5\%$, thus implying that $\mathcal{L}_{IS}$ is minimized by a distribution whose 95% PI is as tight as possible while containing all the training data.

3.3. Training and prediction

In BNNs, the variational posterior of $\boldsymbol{\theta}$ is typically defined layer-wise as a multivariate Gaussian with mean $\boldsymbol{\mu} \in \mathbb{R}^{c_k}$ and covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{c_k \times c_k}$, i.e., $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where c_k is the total number of connections between two consecutive layers. Estimating the full covariance matrix requires learning $\mathcal{O}(c_k^2)$ parameters and is thus computationally prohibitive in most applications [49]. To reduce the costs, some simplifications have been adopted in the literature, such as learning diagonal or block diagonal [58] covariance matrices. However, our approach does not suffer from this computational issue since the only Bayesian part of our network is Block 1 (see Fig. 1) whose size is typically very small (we use one hidden layer with 5 neurons for all the studies in Section 4). Hence, we estimate a dense covariance matrix between any two layers of Block 1 to improve its uncertainty quantification capacity. As for the prior, we use a zero mean Gaussian distribution with diagonal covariance matrix which makes the KL term equivalent to $L2$ regularization with a rate defined by the standard deviation of the prior distribution [59]. Thus, the standard deviation is a hyperparameter that needs to be tuned specifically to each problem.

[11] Recall that we use Bayes by backprop which takes a variational approach towards finding the posteriors, see Section 2.2.

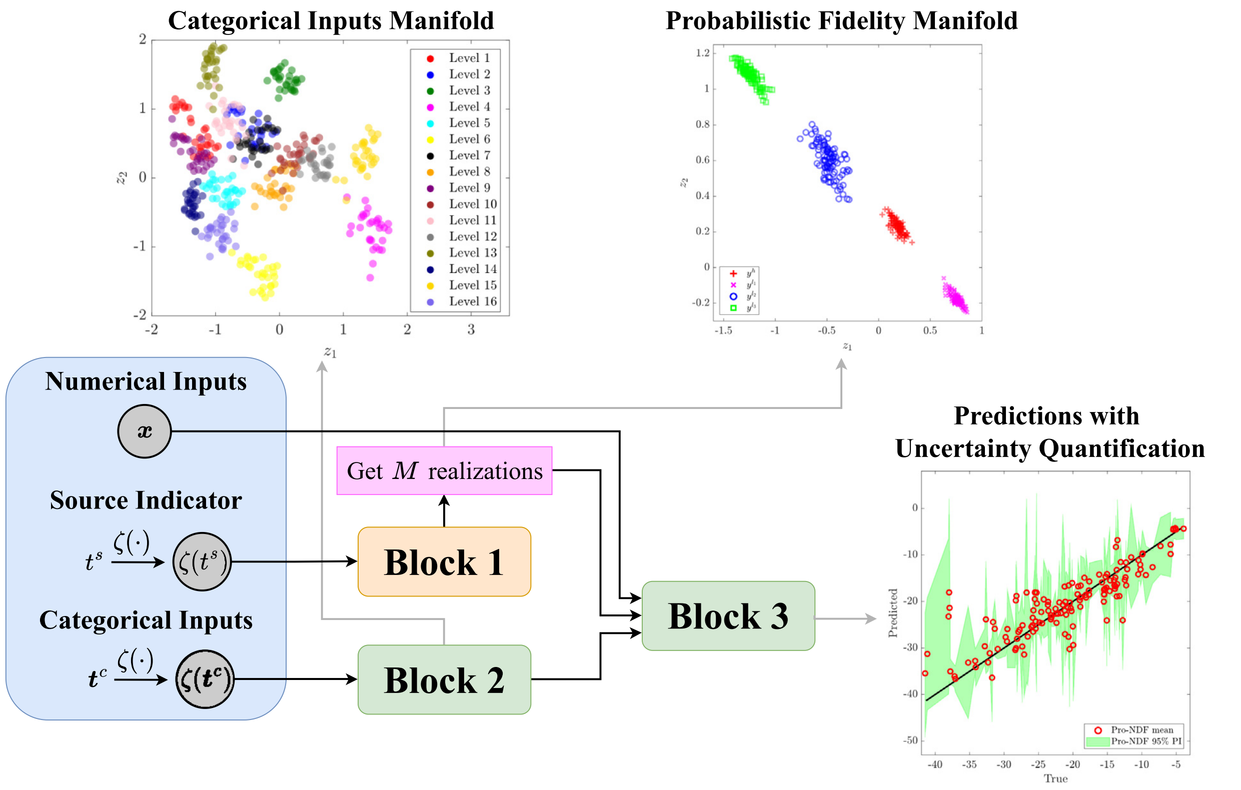

Fig. 2. Outputs of Pro-NDF : We visualize the outputs of Pro-NDF after it is trained on the MF data of the HOIP data set, which does not have any numerical inputs (see Appendix B for more details). To provide probabilistic predictions that quantify both epistemic as well as aleatoric uncertainties, Pro-NDF learns a probabilistic fidelity manifold (where sources with similar behavior are encoded with close-by distributions) and a deterministic manifold for the categorical inputs.

BNNs represent their weights and biases by parameterized distributions which in our case are multivariate normal with dense covariance matrices. In a forward pass during either training or prediction, we take individual samples from these distributions and assign them to the weights and biases. In this way, instead of explicitly obtaining the true posterior distribution of the output of Block 1 (i.e., $\boldsymbol{z}^s$, see Fig. 1), we obtain an empirical distribution in the $\boldsymbol{z}^s$ manifold by taking a number of forward passes, see Fig. 2. We refer to these forward passes as *realizations* and as explained below we use different number of passes in training versus prediction.

To obtain the response (in training or testing) at the input $\boldsymbol{u}$ using Pro-NDF, which contains both a BNN component and a probabilistic output, we use ensemble prediction formulas [38]:

$$\hat{\mu}(\boldsymbol{u}) = \frac{1}{M}\sum_{j=1}^{M}\hat{\mu}_{\boldsymbol{\theta}_j}(\boldsymbol{u}) \tag{24}$$

$$\hat{\sigma}(\boldsymbol{u}) = \frac{1}{M}\sum_{j=1}^{M}\left(\hat{\sigma}^2_{\boldsymbol{\theta}_j}(\boldsymbol{u}) + \hat{\mu}^2_{\boldsymbol{\theta}_j}(\boldsymbol{u})\right) - \hat{\mu}^2(\boldsymbol{u}) \tag{25}$$

where $\hat{\mu}_{\boldsymbol{\theta}_j}(\boldsymbol{u})$ and $\hat{\sigma}_{\boldsymbol{\theta}_j}(\boldsymbol{u})$ are, respectively, the mean and standard deviation of the output distribution in the jth realization and $\boldsymbol{\theta}_j$ are the associated network parameters. For predictions with a fitted NN, we use $M = 1000$ since it provides a higher accuracy in quantifying the uncertainty associated with learning the fidelity manifold (i.e., $\boldsymbol{z}^s$). While training the network, we use $M = 200$ to reduce the computational costs.

The performance of an NN is highly sensitive to its architecture and hyperparameters if the training data is small, unbalanced, and multi-fidelity. To reduce this sensitivity and leverage the low costs of training a single NN on small data, we perform automated hyperparameter tuning.[12] To this end, we use RayTune [60] and Hyperopt to find the optimum hyperparameters and architecture by minimizing the five-fold cross-validation errors on predicting the high-fidelity data.

For our approach specifically, we apply the above tuning strategy to the architecture of Block 3, the learning rate of the Adam optimizer, α_1, α_2 and α_3 in Eq. (19), the prior standard deviation of weight matrices in Block 1, and the batch size. We fix the architectures of Block 1 and Block 2 to one hidden layer with 5 neurons and the dimension of both manifolds to 2. The activation function for all the neurons of Block 1 and 3 is hyperbolic tangent, whereas for Block 2 it is the sigmoid function. In addition to one-hot encoding the categorical inputs, we scale (normalize) the numerical inputs and outputs to the range $[0, 1]$ based on the range of the training data to assist the network in training. The outputs are un-scaled to the original space after prediction. For more information and full details on implementation, please see our GitLab repository.

4. Results and discussions

In this section, we validate our approach on multiple analytic and real-world MF problems (detailed in Appendices A and B) and compare its performance against LMGP, multi-fidelity deep Gaussian processes (MF-DGP) [61], a single-fidelity GP (SF-GP) fit on the HF data, and two other existing NN-based approaches which are based on simple feedforward networks or sequential multi-fidelity (SMF) networks which are described in Appendix C. The hyperparameters of all the NN-based approaches are tuned as described in Section 3.3. We refer the reader to our GitLab repository for specific details on implementation, estimated hyperparameters, and training/test data. For LMGP, SF-GP, and MF-DGP, none of its architectural parameters (such as the kernel type, mean function, latent map, etc.) are tuned.

We first conduct an ablation study in Section 4.1 to quantify the impacts of our designed architecture, loss function, and probabilistic elements. Then, we test the performance of the six MF approaches on the analytic and real-world problems in Section 4.2 and Section 4.3, respectively. Finally, in Section 4.4 we carry out convergence studies on three bi-fidelity examples to test the performance of our approach against existing modeling techniques over a range of data set sizes. In each problem, the goal is to model the HF source as accurately as possible, i.e., to obtain the lowest mean prediction error while maximizing the number of training/test samples that fall in the 95% PI. To this end, we use normalized root mean squared-error (NRMSE) and normalized mean negatively oriented interval score (NIS)[13] as our evaluation metrics. Note that the FFNN and SMF approaches are not probabilistic, i.e., they provide point estimates rather than PIs and therefore they are only evaluated based on NRMSE.

All the experiments of this section are conducted on an NVIDIA GeForce RTX 3060 with 64 GB of RAM using NVIDIA CUDA 11.2 and cuDNN 8.4.1. We use Python 3.8.10 with Tensorflow 2.6.0, Tensorflow Probability 0.14.0, and Keras 2.6.0. We report the training time for fitting Pro-NDF and the other MF approaches in Appendix D.1.

4.1. Ablation study

To evaluate the impact of the key components of Pro-NDF, we perform an ablation study on the Rational and DNS-ROM problems which are detailed in Appendices A and B, respectively. Namely, we analyze the impact of:

1. Using a BNN rather than a deterministic FFNN in Block 1 for probabilistically learning the relations between the data sources.
2. Considering $\mathcal{L}_{IS}$ in the loss function of Eq. (19).
3. Fitting the model to the parameters of a distribution instead of a scalar, i.e., using a probabilistic output.
4. Leveraging the fidelity map to detect the least accurate LF source and, in turn, assessing whether this source helps emulating the HF source.

Regarding the third item above we note that we no longer use NIS in the loss once the probabilistic output is removed. However, we still calculate the NIS after training based on the empirical distribution of the fidelity manifold which is produced by the multiple realizations of the BNN component.

[12] We use this approach for all NN-based data fusion approaches (including ours) in Section 4.

[13] When used as evaluation metrics, the NRMSE and NIS are equivalent to Eq. (A.1) and Eq. (22), respectively, calculated over the test data and scaled by the standard deviation of the high-fidelity training data.

Table 1
Results of the ablation study: We evaluate the effect of removing individual components of Pro-NDF from it by reporting the NRMSE and NIS on unseen HF data. All models are trained as discussed in Section 3 (e.g., all models benefit from automatic hyperparameter tuning). For both NRMSE and NIS, lower numbers indicate better performance. The ticks indicate whether a component is used. The acronyms and symbols are defined as: HF: high-fidelity, LF1: low-fidelity 1, LF2: low-fidelity 2, LF3: low-fidelity 3, $\mathcal{L}_{IS}$: negatively oriented interval score term in the loss function of Eq. (19), PB1: probabilistic Block 1, PO: probabilistic output.

Problem	Model version	Input data				Components			NRMSE	NIS
		HF	LF1	LF2	LF3	$\mathcal{L}_{IS}$	PB1	PO		
Rational	Base	✓	✓	✓	✓	✓	✓	✓	**0.084**	**0.464**
	V1	✓	✓	✓	✓	✗	✓	✓	0.119	0.536
	V2	✓	✓	✓	✓	✓	✗	✓	0.110	0.488
	V3	✓	✓	✓	✓	✗	✓	✗	0.092	0.830
	V4	✓	✓	✓	✗	✓	✓	✓	0.156	2.021
DNS-ROM	Base	✓	✓	✓	✓	✓	✓	✓	0.101	**0.520**
	V1	✓	✓	✓	✓	✗	✓	✓	0.119	0.632
	V2	✓	✓	✓	✓	✓	✗	✓	0.155	1.186
	V3	✓	✓	✓	✓	✗	✓	✗	0.140	4.379
	V4	✓	✓	✓	✗	✓	✓	✓	**0.100**	0.564

We summarize the results of the ablation study on the two examples in Table 1. For both problems, we observe that using all components minimizes the test NRMSE and NIS. Notably, both of our model's probabilistic components significantly increase the performance: the probabilistic output enables Pro-NDF to not only capture aleatoric uncertainty, but also leverage NIS in its loss function. Additionally, using a BNN improves Pro-NDF's HF emulation capabilities by preventing overfitting in scarce data regions (since Block 1 is regularized) and by partially disentangling epistemic and aleatoric uncertainties which yields better PIs.

We observe that without a probabilistic output, the NIS (and hence the uncertainty quantification accuracy) drops quite significantly (compare V3 to V1 and the base in either of the problems) since the model can no longer account for aleatoric uncertainties. By comparing V1 to the base model in either of the problems in Table 1 we see that for a model with a probabilistic output the optimal performance is obtained when $\mathcal{L}_{IS}$ is used in the loss. That is, leveraging the NIS in training improves both mean prediction and uncertainty quantification (measured via NRMSE and NIS, respectively).

In both problems, evaluating V1 through V3 against one another indicates that there is a trade-off between NRMSE and NIS. That is, versions that perform well in terms of NRMSE, do not generally provide the smallest NIS. However, when all of these components are included in Pro-NDF (see the base model in Table 1 for either of the problems), both NRMSE and NIS are reduced. This improvement is due to the fact that the priors and $\mathcal{L}_{IS}$ effectively regularize the model whose learning capacity is substantially increased by the probabilistic natures of Block 1 and the output.

The probabilistic fidelity manifold (i.e., output of Block 1) provides an intuitive and visualizable tools to learn the similarity/discrepancy among the sources. Hence, once we fit the base model in each problem, we analyze the learned fidelity manifold to determine the LF source that has the least similarity to the HF source, see Figs. 4(a) and 6(a). Based on the distances in the fidelity manifold of each problem, we conclude that the third LF source is the least correlated one with the HF source in both cases. We exclude this source and its data from MF modeling and refit the base model to the rest of the data, see version V4 for both problems.

One of the major outputs of Pro-NDF is the learned fidelity manifold which indicates which LF source has the highest discrepancy compared to the HF source. Hence, after training a Pro-NDF and inversely identifying the least accurate LF source, we can build another Pro-NDF while excluding the data from this source. In the Rational problem, omitting the lowest-fidelity source results in much worse NRMSE and NIS. We explain this observation by noting that this problem has an extremely small number of HF samples and therefore it is important to judiciously use all available data in training. However, in the DNS-ROM problem version V4 achieves the best NRMSE while Pro-NDF with all components achieves the best NIS and second best NRMSE (compare base to V4 in Table 1). We explain this trend by noting that the size of the training data in the DNS-ROM problem is significantly higher than that in the Rational problem. Therefore, omitting a highly inaccurate data source slightly improves mean prediction accuracy for the HF source in the DNS-ROM problem since the input–output relationships learned by Block 3 for

Table 2
Results on the analytic examples for different models: We test the performance of Pro-NDF against LMGP, SF-GP, MF-DGP, and two existing NN-based technologies for the Rational, Wing-weight and Borehole examples detailed in Table 4. The training procedure for Pro-NDF, LMGP, FFNN and SMF is discussed in Section 3, Section 2.1, Appendix C.1 and, Appendix C.2 respectively. We report the NRMSE and NIS on unseen HF data. We provide MF-DGP with additional information that specifies the relative accuracy of the LF sources.

Model	Rational		Wing-weight		Borehole	
	NRMSE	NIS	NRMSE	NIS	NRMSE	NIS
Pro-NDF	**0.084**	0.464	0.140	0.675	0.080	0.400
LMGP	0.085	**0.413**	**0.112**	**0.540**	**0.074**	**0.365**
MF-DGP	0.095	0.885	0.197	1.056	0.453	2.936
SF-GP	0.111	0.889	0.327	2.114	6.423	3.43
FFNN	0.112	–	0.146	–	0.096	–
SMF	0.612	–	0.371	–	0.273	–

the different sources are more similar. Omitting data from this source also increases the ratio of HF data available in the unified data set which helps in learning the HF behavior. However, using all data sources provides Pro-NDF with more information which improves the uncertainty quantification capability and hence a smaller error on NIS.

4.2. Analytic problems

In this section, we validate our approach against LMGP, SF-GP, MF-DGP, and two existing NN-based technologies for the Rational, Wing-weight, and Borehole examples detailed in Appendix A. These examples cover a wider range of input dimensionality, number of sources, and model form errors (e.g., additive and nonlinear biases). Similar to the previous section, we use NRMSE and NIS on HF test data as the performance metrics. The input space of these three examples does not have categorical features and hence both Pro-NDF and LMGP learn a single manifold. We visualize the fidelity manifolds learned by Pro-NDF and LMGP to examine these models' ability in inversely learning the relationships among the data sources (note that the LF sources are *not* ordered based on their accuracy). We highlight that MF-DGPs require knowledge of the relative fidelity of the sources and hence we provide them with this additional information. We also remark that, unlike Pro-NDF, the fidelity manifold of LMGP is not probabilistic and hence each data source is encoded with a single point in the manifold.

The results for each approach on each problem are summarized in Table 2 and demonstrate that the probabilistic approaches, i.e., LMGP and Pro-NDF, significantly outperform the deterministic approaches and MF-DGP (which is probabilistic) in all problems. The FFNN approach performs significantly worse than LMGP and sometimes approaches the performance of Pro-NDF in NRMSE, while the SMF approach shows poor performance for all problems. We explain SMF's poor performance by noting that, as explained in Appendix C, hierarchical MF techniques such as SMF heavily rely on the knowledge of fidelity levels to process the data sources sequentially in the order of increasing accuracy. Since we assume in the problem setup that we only know which source has the highest fidelity and do not know the relative fidelity levels of the LF sources, the LF sources are ordered sub-optimally in the SMF approach which leads to a very poor prediction accuracy. The FFNN approach, by contrast, does not rely on the knowledge of fidelity levels and as such performs better than SMF. However, its performance lags behind that of LMGP and Pro-NDF because the architecture is not designed with MF problems in mind.

LMGP, which is considered as our gold standard for MF problems with small data, outperforms Pro-NDF in both NRMSE and NIS for the Wing-weight and Borehole problems, and in NIS for the Rational problem. The Rational problem is simultaneously the most data deficient and least complex of the problems examined in this paper: as shown in Table 4, there are 4 data sources with only one being especially inaccurate, the input and output are both $1D$, and there are only 5 training samples provided for the HF source. Pro-NDF and LMGP are well suited to tackle this problem as they both perform well for low-dimensional problems with simple underlying functional forms and well-correlated sources, and as such they have similar performance. Fig. 3(a, b) reveals that LMGP captures all of the training points in a narrower 95% PI compared to Pro-NDF which explains LMGP's lower NIS in Table 2. However, Pro-NDF shows a better performance for this problem in terms of mean prediction accuracy and it also has a higher degree of agreement with the true function in extrapolation while LMGP reverts to its mean. We therefore conclude that both methods perform on par on the Rational problem.

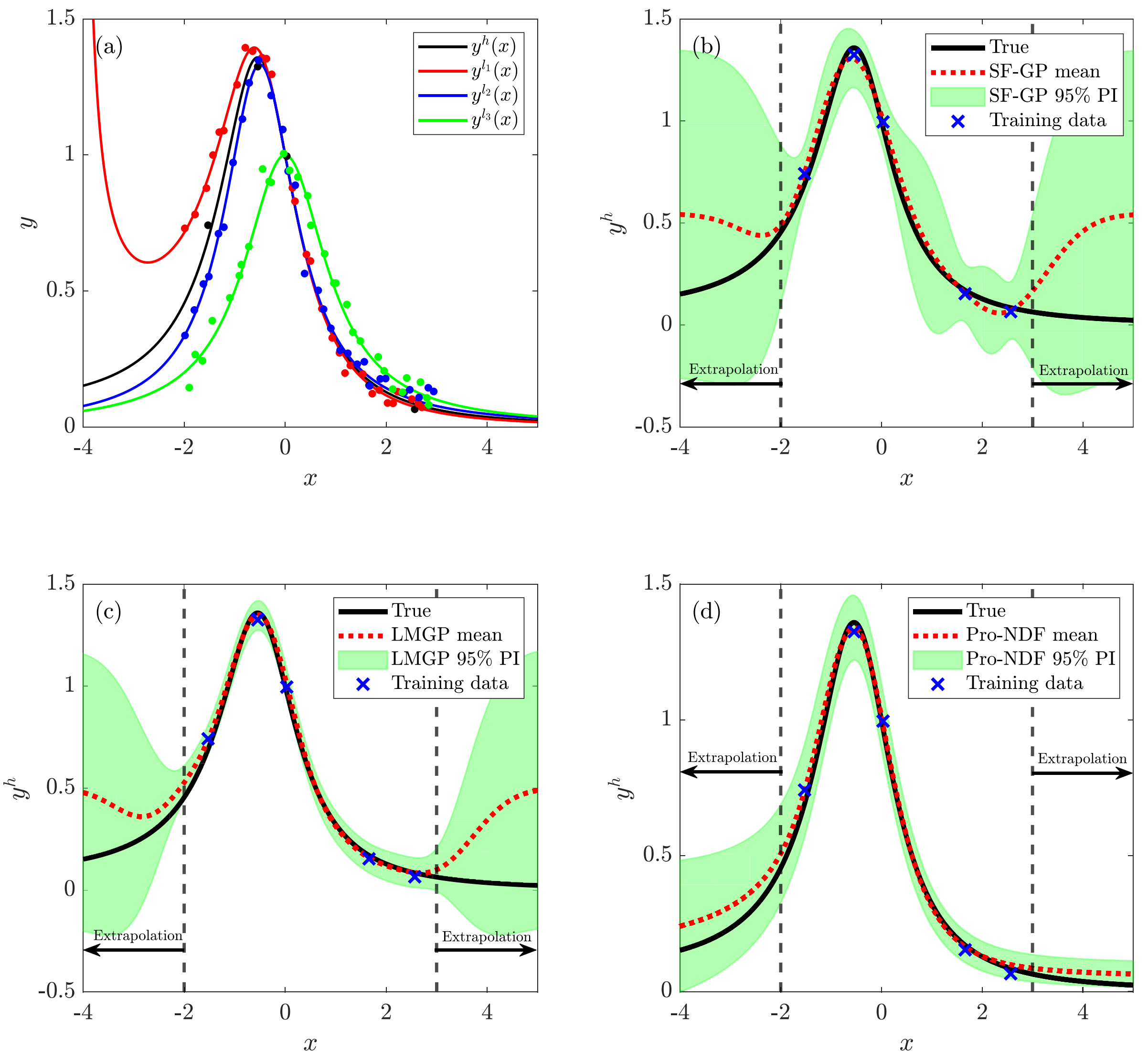

Fig. 3. High-fidelity emulation on the Rational problem: (a) Four data sources and the corresponding training data, (b) SF-GP fit only to the HF data, (c) LMGP fit to all available data and (d) Pro-NDF fit to all available data. Pro-NDF and LMGP approaches clearly outperform SF-GP in terms of mean accuracy and give a narrower PI. They both produce similar results in terms of mean prediction in interpolation. However, LMGP has a narrower 95% PI and reverts to its mean in extrapolation.

The learned fidelity manifold of Pro-NDF for the Rational problem is shown in Fig. 4(a) which indicates that the network has inversely learned the true relationship between the data sources as y^{l_1} and y^{l_2} are encoded close to y^h while y^{l_3} is quite far from y^h. These relative distances are proportional to the accuracy of the LF sources with respect to the HF source (in the interpolation region) which are reported in Table 4. The fidelity manifold also shows a high spread in the distributions of the realizations for individual sources which indicates either a poor fit to the data or a lack of training samples. In this case, we attribute this spread to lack of data since the performance in NIS and NRMSE is quite good.

The Wing-weight and Borehole problems are both high-dimensional problems with relatively complex underlying functional forms and small amounts of data. LMGP is very well suited to tackle this type of problem [33] because the number of its hyperparameters scales much better than NN-based approaches such as Pro-NDF. Accordingly, we observe that LMGP achieves lower NRMSE and NIS for both examples.

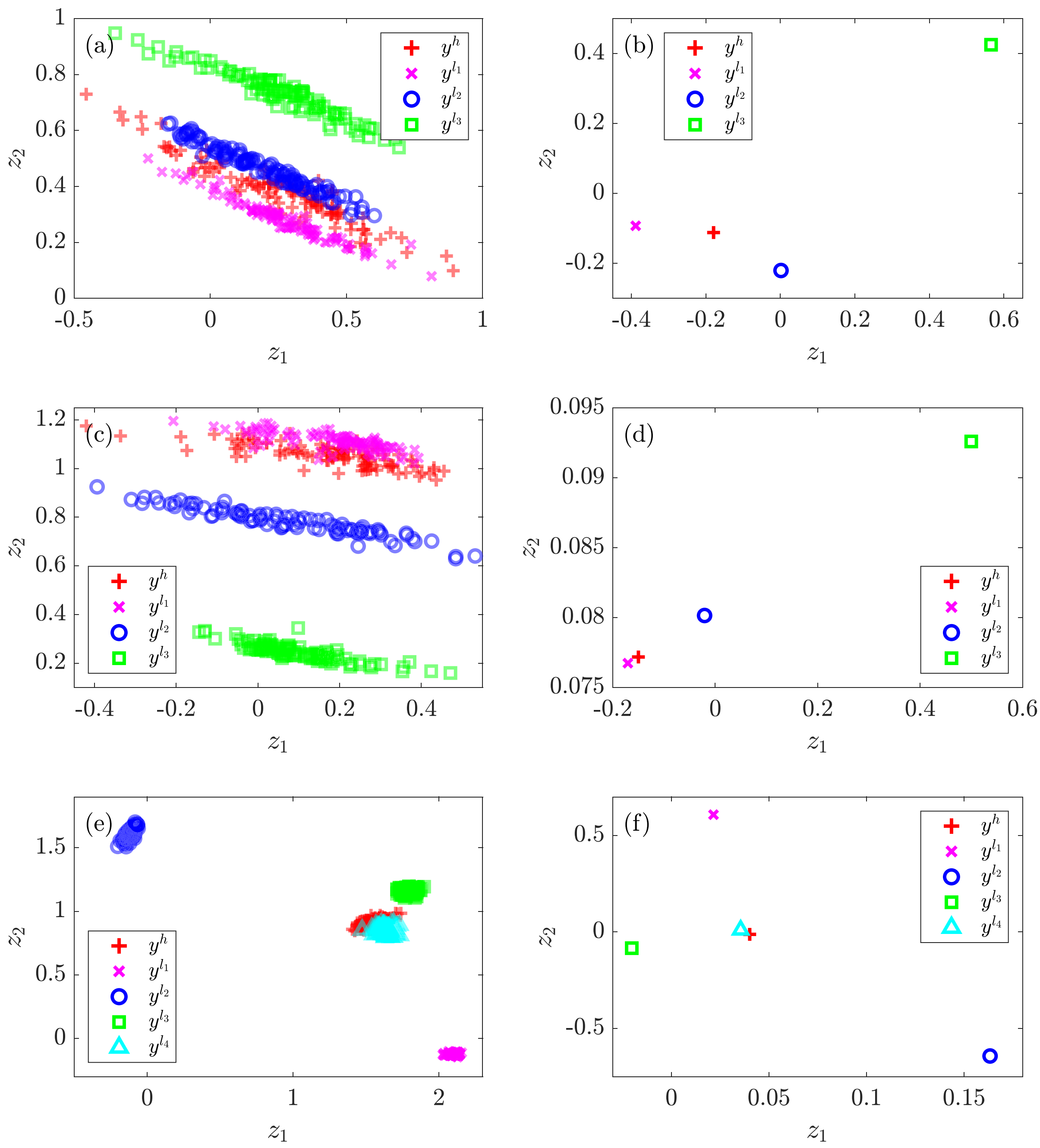

Fig. 4. Pro-NDF and LMGP fidelity manifolds for the analytic problems: (a) Pro-NDF for Rational problem, (b) LMGP for Rational problem, (c) Pro-NDF for Wing-weight problem, (d) LMGP for Wing-weight problem, (e) Pro-NDF for Borehole problem, and (f) LMGP for Borehole problem.

Comparing the performance of Pro-NDF across the two high-dimensional problems, we observe that it performs much better on the Borehole problem. Examining the fidelity manifold learned by Pro-NDF and LMGP for the Wing-weight problem, see Figs. 4(c) and 4(d), respectively, we see that both approaches accurately determine the relationship between the sources as they agree with the NRMSEs reported in Table 4. Specifically, y^{l_1} is closer to y^h than y^{l_2}, which in turn is closer than y^{l_3}. Notably, both LMGP and Pro-NDF have the same relative ordering and

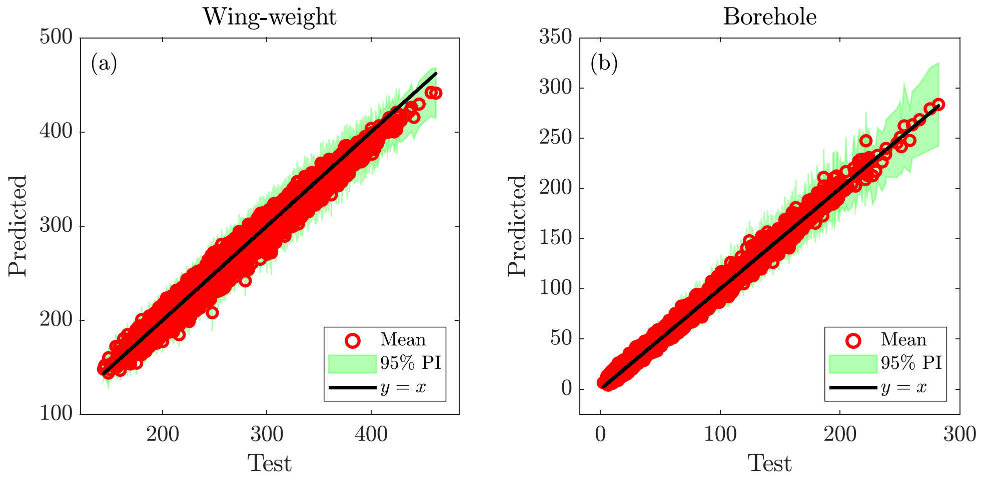

Fig. 5. Predictions vs noisy test outputs: We compare the predictions of Pro-NDF on 10,000 random HF samples for the two high-dimensional analytical problems. The noisy test data are within the 95% PIs which indicates that Pro-NDF is achieving a high performance in uncertainty quantification.

positioning of the sources, i.e., (1) the mean position of all sources lies on an axis, and (2) y^{l_1} is in the opposite direction relative to y^h from y^{l_2} and y^{l_3}. This reinforces our earlier assertion in Section 3: the positions of the sources in the fidelity manifold learned by Pro-NDF reflect *correlations* between the data sources. However, the relative distances between the LF sources in the latent space found by LMGP more accurately represents the true relationships between the sources because the position for y^{l_3} is much more distant from y^h than encoded positions of the other sources.

In Fig. 4(a) we observe a large spread in the realizations (i.e., the posterior distributions in the fidelity manifold are quite wide) which partially explains the poor[14] performance in this problem. We attribute this performance level to the relative accuracy of the data sources since only one source, y^{l_1}, is at all accurate with respect to y^h while the other LF sources are quite inaccurate. LMGP's performance is not inherently hampered by including poorly correlated sources in the data fusion problem [33] since its performance, upon successful optimization, is at worse on par with fitting separate GPs to each source. By contrast, since Pro-NDF's Block 3 is responsible for learning the relations between all sources and uses weight sharing, including especially inaccurate sources leads to relatively poor performance as shown in 4.1.

The Borehole problem has five total sources where two LF sources (y^{l_3} and y^{l_4}) are accurate while two LF sources (y^{l_1} and y^{l_2}) are quite inaccurate with respect to y^h. Since there are more total data available compared to the Wing-weight problem due to the additional data source, and since there are more high-accuracy LF sources, we observe that the spread of the realizations in the fidelity manifold of Pro-NDF is much smaller than in the Wing-weight, see Fig. 4(e). This narrow spread indicates that a good fit has been achieved. We again observe that the relative directions and distances of the LF sources from y^h are nearly identical in the manifolds of Pro-NDF and LMGP, see Fig. 4(f), and that both methods have correctly identified the relationships between the sources, see Table 4. Based on these observations, it is no surprise that Pro-NDF achieves very good performance and nearly matches LMGP in terms of NRMSE and IS. For a better comprehension of the accuracy of Pro-NDF in both high-dimensional problems, we show the HF predictions at 10,000 random test inputs in Fig. 5.

Lastly, we note that the performance of MF-DGPs, especially in uncertainty quantification (as measured by NIS), is much worse than both LMGP and Pro-NDF. This is an especially interesting result since we provide MF-DGP with the correct ordering of the data sources since it processes the data sequentially (e.g., for the Rational problem, MF-DGP knows that y^{l_2} is the most accurate LF source which is then followed by y^{l_1} and y^{l_3}). We attribute the deficiencies of MF-DGP to its sequential architecture which prevents the model from using all available data in emulating all the sources (e.g., as opposed to our approach, HF data does not directly contribute to learning any of the LF sources).

[14] Poor with respect to LMGP. The performance of Pro-NDF is still much better than the other four approaches.

Table 3
Results on the real-world examples for different models: We test the performance of Pro-NDF against LMGP, SF-GP, MF-DGP, and two existing NN-based technologies for the DNS-ROM and HOIP data sets detailed in Appendix B. We report the NRMSE and NIS on unseen HF data. We provide MF-DGP with additional information that specifies the relative accuracy of the LF sources. MF-DGP and vanilla GPs cannot handle categorical variables and hence for the HOIP example we do not apply MF-DGP and use LMGP (which can handle categorical inputs) instead of vanilla GPs for the SF-GP method.

Model	DNS-ROM		HOIP	
	NRMSE	NIS	NRMSE	NIS
Pro-NDF	**0.101**	**0.520**	**0.470**	**1.855**
LMGP	0.105	0.706	0.473	2.510
MF-DGP	0.140	0.791	–	–
SF-GP	0.123	0.653	0.779	5.923
FFNN	0.113	–	0.580	–
SMF	0.130	–	0.663	–

4.3. Real-world problems

In this section, we validate our approach against LMGP, SF-GP, MF-DGP, and two existing NN-based technologies on two engineering applications which are detailed in Appendix B. We again use NRMSE and NIS on HF test data as our performance metrics and examine the manifolds learned by Pro-NDF and LMGP. In both of these applications, the input space has categorical features (so Pro-NDF and LMGP each build two manifolds) and we do not know the underlying relationships between the data sources.

The results for each approach on each problem are summarized in Table 3 and demonstrate that the probabilistic approaches again significantly outperform the deterministic ones as well as MF-DGP which is provided with the additional information on the relative accuracy of each data source (this information is not provided to any of the other approaches). The FFNN approach performs nearly as well as LMGP and Pro-NDF in the DNS-ROM problem, but lags behind Pro-NDF and LMGP in the HOIP problem in terms of NRMSE. The SMF approach shows poor performance for both problems for the same reasons provided in Section 4.2. Notably, Pro-NDF outperforms LMGP for both problems in terms of both metrics which we partially explain by noting that there are much more data are available in these real-world problems compared to the analytical examples of Appendix A. Being an NN-based approach, Pro-NDF scales very well with additional data while the performance of LMGP has diminishing returns and eventually plateaus (recall that the latent map and kernel of LMGP are *not* tuned which contribute to this plateauing performance).

As shown in Fig. 6(a-b), the fidelity manifolds learned by Pro-NDF and LMGP for the DNS-ROM problem are nearly analogous as the relative distances are quite similar. However, LMGP finds all sources to be on the diagonal axis while Pro-NDF learns a more nuanced relationship between the sources, which may contribute to its superior performance. We also observe that the spreads in the individual realizations for each point are fairly tight, which indicates that Pro-NDF is able to learn the relations between the sources reasonably well and, accordingly, provide good performance in terms of NRMSE and NIS. Like in Section 4.2, we show the HF predictions at test inputs in Fig. 7 in order to facilitate the comprehension of the accuracy of Pro-NDF in these two problems.

The HOIP problem has three categorical inputs with 10, 3, and 16 levels and as such Pro-NDF uses two separate latent transformations (one for the data source and the other for the three categorical variables) that correspond to Blocks 1 and 2 in Fig. 1. The learned categorical manifolds for Pro-NDF and LMGP are shown in Figs. 8 and 9 where the z^c is visualized four times as the combinations of the categorical variables are color-coded based on the levels of each of the three categorical variables and based on the average value of the output.[15] Since there are no numerical features, the combined inputs are $\boldsymbol{u} = [\boldsymbol{\zeta}(t^s), \boldsymbol{\zeta}(\boldsymbol{t}^c)]$ and $\boldsymbol{v} = [z^s, z^c]$ are the inputs to Block 3 of Pro-NDF. Recall that we only use a BNN in Block 1 and as such we show only one realization for the manifold for Pro-NDF that encodes the categorical variables.

Pro-NDF outperforms LMGP in terms of NRMSE by a small margin and NIS by a significant margin for this problem which we attribute to the size of the data sets. Pro-NDF is able to leverage these additional data much more

[15] This average is obtained using the entire data set including both the training and test data.

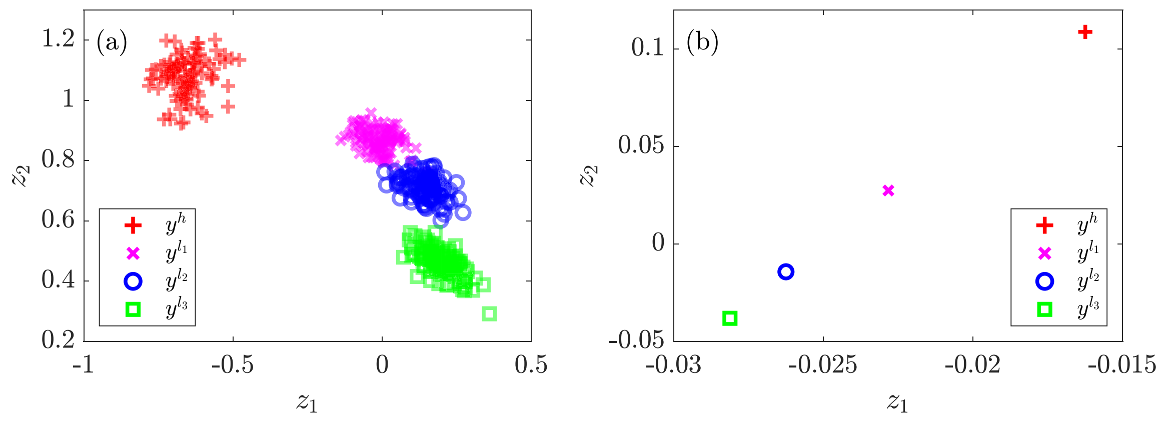
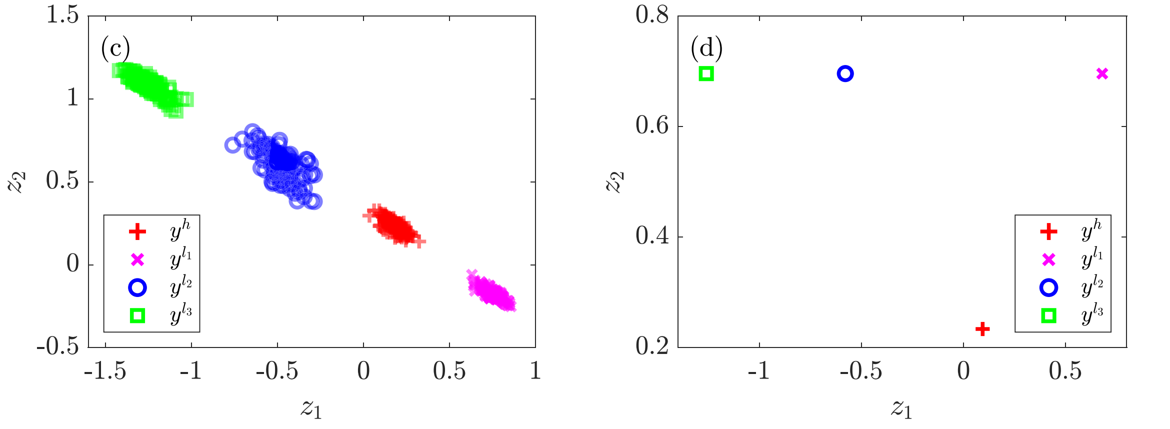

Fig. 6. Pro-NDF and LMGP fidelity manifolds for the real-world problems: (a) Pro-NDF for DNS-ROM data set, (b) LMGP for DNS-ROM data set, (c) Pro-NDF for HOIP data set, (d) LMGP for HOIP data set.

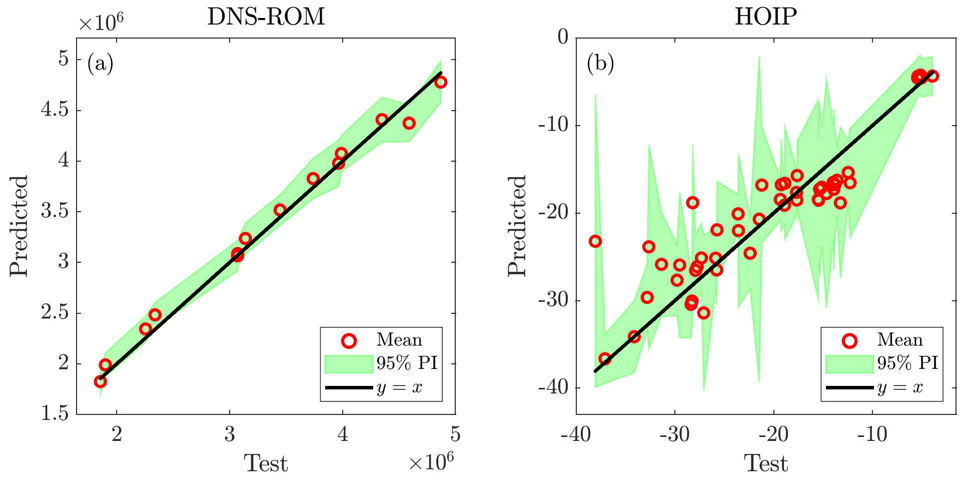

Fig. 7. Predictions vs noisy test outputs: We compare the predictions of Pro-NDF on test data for the two high-dimensional engineering problems. The noisy test data are within the 95% PIs which indicates that Pro-NDF is achieving a high performance in uncertainty quantification.

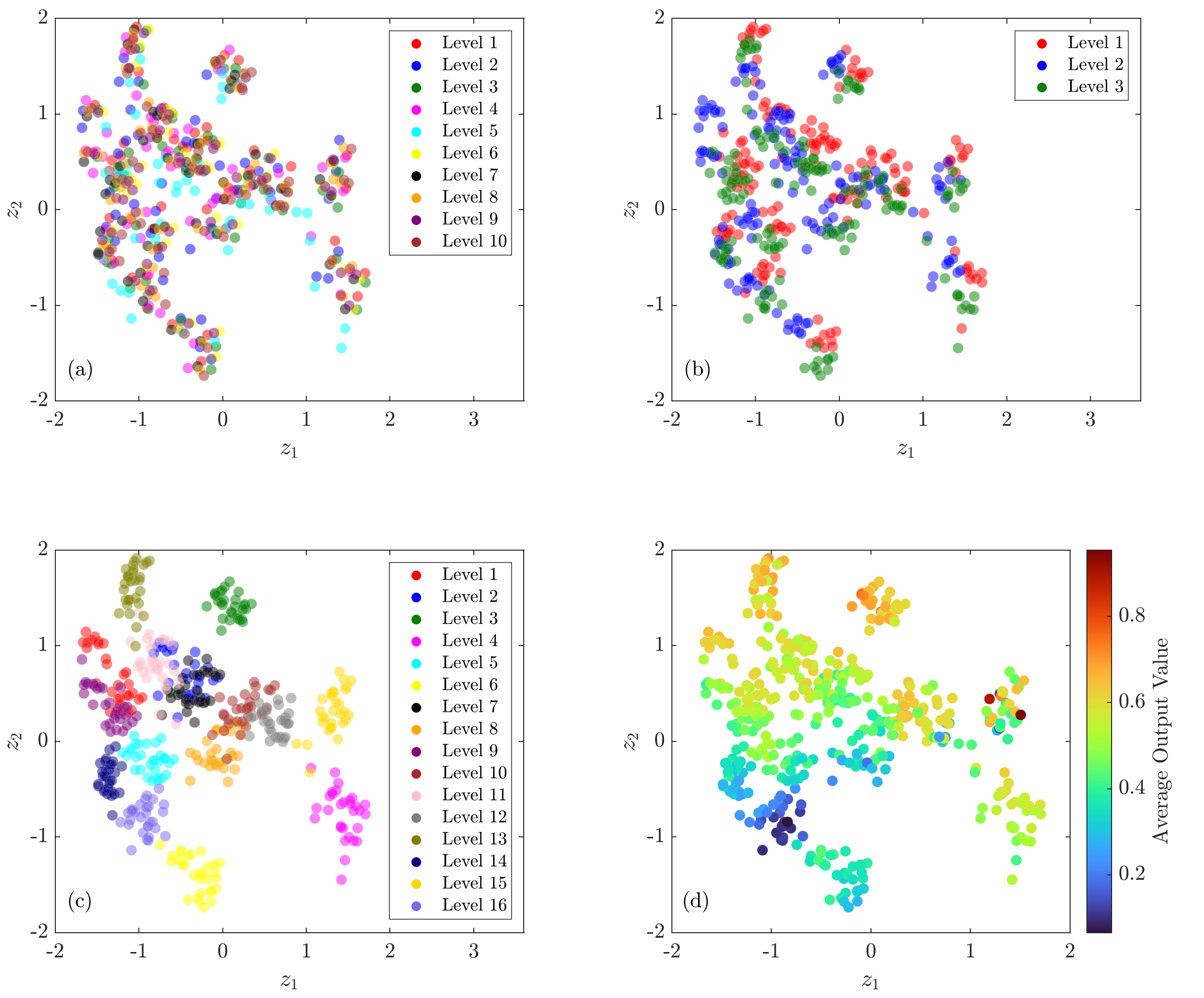

Fig. 8. Pro-NDF categorical manifold for the HOIP problem: The combination of the categorical variables' levels are color-coded based on: (a) the levels of t_1^c, (b) the levels of t_2^c, (c) the levels of t_3^c, (d) the average output value. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

readily than LMGP which only uses simple mapping functions to handle categorical variables t^s and $\boldsymbol{t}^c$. Pro-NDF also finds fairly tight spreads in the probabilistic fidelity manifold, see Fig. 6(c); indicating that it has high certainty in its outputs and that we should expect good performance. We note that all sources are found to be roughly on one axis and roughly spaced evenly from each other, which may indicate that Pro-NDF has failed to learn the more nuanced relationships between the sources. Equally likely, however, is that the relationships between the sources are simple enough to be represented in this way; since we do not know the underlying functional forms for this problem, we cannot give a definitive answer.

We can also glean some information about the relationships between the categorical variable levels and their impact on the output by examining the corresponding manifolds in Figs. 8 and 9. Fig. 8(c) shows that Pro-NDF finds distinct clusters for all 16 levels of t_3^c which indicates that distinguishing between the levels of t_3^c is important to learning the output. Similarly, the levels of t_2^c are distinguishable in Fig. 8(b) as t_2^c affects the response value. By contrast, Fig. 8(a) shows no apparent trend between the 10 levels of t_1^c which implies that t_1^c has little effect on the output as Pro-NDF does not learn to distinguish the levels from each other. By contrast, the manifold found by LMGP, shown in Fig. 9 shows much less distinct clustering for each of the three categorical variables, which may help explain why it achieves a lower NIS than Pro-NDF. Finally, we examine whether the latent positions for the categorical combinations are influenced by the average output value in Figs. 8(d) and 9(d). The manifold for

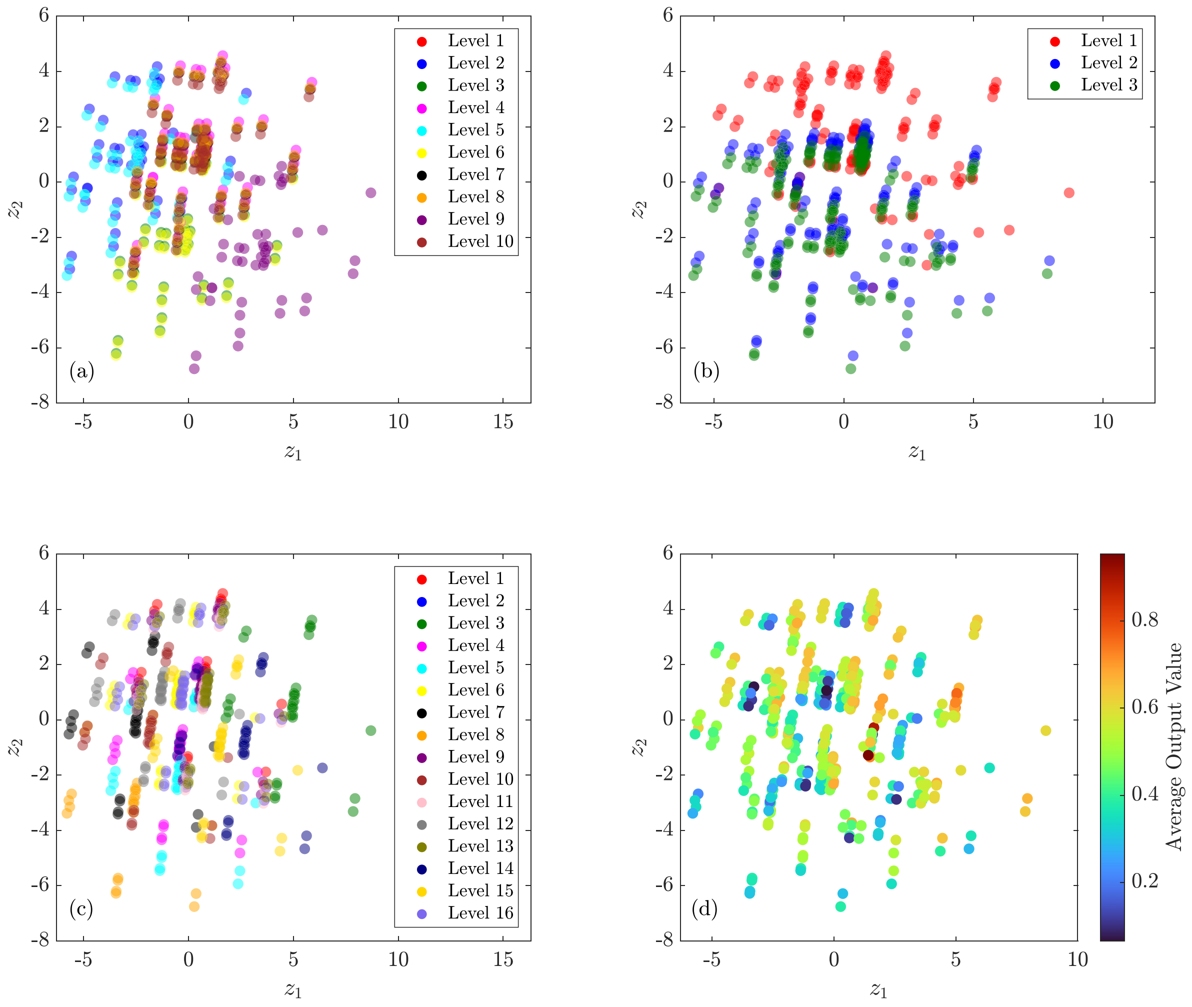

Fig. 9. LMGP categorical manifold for the HOIP problem: The combination of the categorical variables' levels are color-coded based on: (a) the levels of t_1^c, (b) the levels of t_2^c, (c) the levels of t_3^c, (d) the average output value. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Pro-NDF shows a clear trend of the average output value increasing as the latent points move from the bottom-left of the space to the top-right, while for LMGP there is no obvious trend. Based on these manifolds, Pro-NDF shows superior ability to discern relationships between the categorical combinations and between levels of categorical variables.

Lastly, we observe MF-DGP provides the worst performance in the DNS-ROM example even though it is provided with the correct ordering of the data sources. We believe MF-DGP does not perform well because it sequentially stacks a set of GPs that (1) prevents bi-directional information flow between any two sources where the data from source i informs the model about source j only if source j is believed to be more accurate than it, and (2) if two sources are not immediately related by two sequential GPs, their relations is only indirectly learnt via intermediate GPs.

4.4. Convergence study

In this section, we perform a convergence study to assess the performance of Pro-NDF against three other methods on three analytical examples while varying the sizes of the data sets. We compare our approach to LMGP, MF-DGP, and multi-fidelity neural networks (MFNN) [36]. All the examples in this subsection are bi-fidelity since

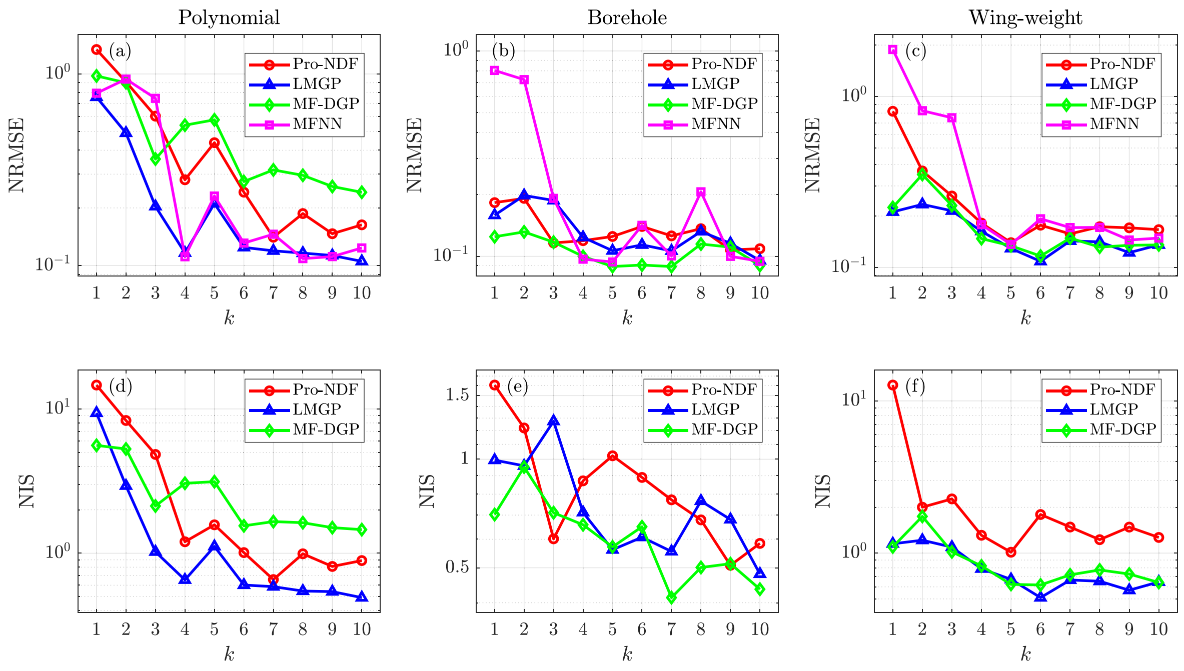

Fig. 10. Convergence study: We compare four data fusion methods in terms of NRMSE and NIS across three examples as the sizes of the data sets increase. The Polynomial (a, d), Borehole (b, e) and Wing-weight (c, f) problems are detailed in Table 4. The size of the initial data sets (at $k = 1$) for the three problems are $n_1^h = 5$ and $n_1^l = 20$. These sizes increase as $n_k^h = k \times n_1^h$ and $n_k^l = k \times n_1^l$.

MFNN can only fuse two data sources (for this reason MFNN is not used in Sections 4.2 and 4.3). The three problems studied here are listed in Table 4 and include the Polynomial function of [62] and the adapted versions of the Wing-weight and Borehole examples where we only use two data sources ($y^h(\boldsymbol{x})$ and $y^{l_2}(\boldsymbol{x})$ for the Wing-weight, and $y^h(\boldsymbol{x})$ and $y^{l_3}(\boldsymbol{x})$ for the Borehole).

We show the results of the convergence study in Fig. 10 in terms of NRMSE and NIS for different data set sizes. The size of the initial data sets (at $k = 1$) for the three problems are $n_1^h = 5$ and $n_1^l = 20$. The sizes of the subsequent data sets are increased via a multiplicative factor k, i.e., $n_k^h = k \times n_1^h$ and $n_k^l = k \times n_1^l$. The results show that all methods, including Pro-NDF, provide higher accuracy as the number of samples for both sources increase. We also observe that for high-dimensional problems (Wing-weight and Borehole), the GP-based approaches are more accurate when data sets are small (e.g., when $k \leq 4$).

In these bi-fidelity examples, both MFNN and MF-DGP naturally leverage the hierarchy between the LF and HF sources since both of them are structured such that the information from the LF source is directly propagated to the part of the model that surrogates the HF source. However, LMGP and Pro-NDF do not leverage this information in that they both aim to emulate both sources as accurately as possible using the cross-correlation between the LF and HF data sets. As we demonstrate in Sections 4.2 and 4.3 such hierarchical approaches provide poor performance in applications where there are more than two data sources.

Despite not being designed for bi-fidelity problems, Pro-NDF achieves a comparable performance to the other state-of-the-art methods in these examples. Rather, Pro-NDF is better suited for MF problems with more than two sources, which we demonstrate by comparing the results in Fig. 10 with the ones from previous subsections. For instance, in Fig. 10 we can see that MF-DGP is able to outperform Pro-NDF in some data sets for the Borehole and Wing-weight examples. However, when using all the available data (not just two sources of data), we observe in Sections 4.2 and 4.3 that Pro-NDF significantly outperforms MF-DGP for all the examples (in fact, in both Borehole and Wing-weight examples, the performance of MF-DGP drops once it is provided with LF data from two additional sources, compare Fig. 10 to Table 4).

5. Conclusion

In this paper, we introduce Pro-NDF for data fusion under uncertainty. Pro-NDF is based on a multi-block NN where each block is designed to take on specific tasks for MF modeling problems that arise in typical engineering applications. One of these blocks is probabilistic whose visualizable output can be used to detect LF sources with large model form errors. The final output of Pro-NDF is also probabilistic which enables to not only quantify aleatoric uncertainties, but also leverage strictly proper scoring rules during training.

We validate each of the key components of Pro-NDF by performing an ablation study on an analytic and a real-world example. We also demonstrate that on multi-source problems Pro-NDF outperforms other NN-based data fusion approaches by a large margin. Moreover, Pro-NDF performs on par to LMGP in low-dimensional cases with small data sets and slightly lags behind LMGP (a competing GP-based approach) in high-dimensional examples with very small data sets. However, as the size of the training data increases, Pro-NDF scales better than LMGP and provides smaller errors. We obeserve based on the reported NIS that Pro-NDF performs comparably to LMGP at avoiding overconfidence, which we attribute to our novel loss function. In these studies, we test the performance on unseen HF data but note that Pro-NDF builds an MF emulator that probabilistically surrogates all the data sources simultaneously.

Our convergence study indicates (1) the performance of Pro-NDF, similar to other MF modeling techniques, improves as the data set sizes increase, and (2) the advantages of our approach are more pronounced in problems with more than two data sources as we can use data on any source to better learn another source.

A particularly useful output of Pro-NDF is its learnt fidelity manifold which encodes source-wise similarities/discrepancies. While the learnt distances in this manifold do not directly link correlation between the sources, we observe that the fidelity manifold of Pro-NDF and LMGP look quite similar in our studies. Since the fidelity manifold of LMGP is embedded in its kernel and hence indicates the correlations, we believe the fidelity of Pro-NDF also estimates a scaled version of correlation. An added benefit of Pro-NDF's fidelity manifold is that it is probabilistic where wide distributions can indicate if Pro-NDF is able to learn the relation between the data sources. Reducing this uncertainty via domain knowledge (especially qualitative information in engineering applications) is a future direction that we plan to investigate.

The performance of any data fusion approach (including ours) can drop if there are one or more very inaccurate LF sources. With Pro-NDF, the learned fidelity manifold can be used to identify-discard such sources and then retrain Pro-NDF anew. This process can be repeated until all LF sources are encoded close to the HF source in the fidelity manifold. This iterative approach is, however, quite inefficient so we plan to develop an automated mechanism that perhaps leverages the fidelity manifold to adjust the loss function and, in turn, prevent Pro-NDF from learning the highly inaccurate LF sources.

We also note that in Sections 4.2 and 4.3, we utilized very small HF data sets in all the examples to assess the performance of the MF emulators under the most challenging conditions (i.e., highly unbalanced data sets which have many LF samples but only a few HF samples). From our observations, we have noticed that the performance of Pro-NDF and the other data fusion approaches can become sensitive to the quality of the training data in such extreme scenarios. However, the results shown in Section 4 suggest a common statistically significant trend: Pro-NDF consistently outperforms other multi-fidelity NN methods and SF-GP, while achieving a similar performance to LMGP (with large data, Pro-NDF can provide higher accuracy).

Finally, we stress the fact that the fidelity manifold currently provided by Pro-NDF is only a function of the source indicator, thereby providing a global correlation measure between the sources. However, for future work, we aim to extend Pro-NDF to include a fidelity manifold that is also dependent on the inputs. This extended approach would enable the learning of a local correlation measure, conditioned on the inputs, thus allowing for the detection of regions in the feature space where the correlation between sources is either high or limited. Another possible direction involves changing the BNN blocks in the architecture to another probabilistic or stochastic model. For example, dropout regularization acts as an approximation of Bayesian inference and has a similar interpretation as an ensemble [63] while providing better computational tractability than BNNs.

Declaration of competing interest

The authors of this research paper declare no conflicting interests.

Data availability

Data is uploaded on GitLab and the link is provided.

Acknowledgments

We appreciate the support from National Science Foundation, United States of America (award numbers 2045322 and 2103708) and the Early Career Faculty grant from NASA's Space Technology Research Grants Program, United States of America (award number 80NSSC21K1809). We also would like to thank the anonymous reviewers for their constructive feedback.

Appendices

We provide the formulations of the analytic problems in Appendix A, the background and details of the real-world problems in Appendix B, and the methodology and details of the FFNN and SMF methods in Appendix C.

Appendix A. Table of analytic examples

Table 4 details the analytic functions used for the examples covered in Section 4. For each multi-fidelity problem, we calculate the accuracy of each LF source with respect to the HF source via normalized root mean squared error (NRMSE):

$$\text{NRMSE} = \sqrt{\frac{(\boldsymbol{y}^l - \boldsymbol{y}^h)^T(\boldsymbol{y}^l - \boldsymbol{y}^h)}{10000 \times \text{var}(\boldsymbol{y}^h)}} \tag{A.1}$$

where $\boldsymbol{y}^l$ and $\boldsymbol{y}^h$ are 10000×1 arrays of outputs sampled randomly via Sobol sequence from the LF and HF sources, respectively. We use the same sample locations and outputs as our test data when evaluating MSE and IS in Sections 4.1 and 4.2.

Appendix B. Background on real-world examples

The DNS-ROM data set is extracted from [2] where the goal is to accelerate multiscale damage simulations of cast aluminum alloys by replacing direction numerical simulations (DNS) that are done at the microscale via reduced-order models (ROMs). These ROMs provide computational acceleration by solving a reduced-order representation of the governing equations and, depending on how much the governing equations are simplified (to achieve speedups), provide different levels of accuracy with respect to the DNS. A data point in the DNS-ROM data set contains the toughness of a microstructure as a function of four microstructural descriptors (pore volume fraction, number of pores, pore aspect ratio, average nearest neighbor distance among the pores) and two material properties which affect the damage behavior of the microstructure (evolutionary rate parameter and critical effective plastic strain). Hence, this data set has 1 output (toughness) and 6 inputs. The response of each sample is obtained via either DNS (HF source) or one of three ROMs (LF sources) which differ in terms of their accuracy and cost (i.e., there is a total of four sources). The data set has $n^h = 70, n^{l_1} = 110, n^{l_2} = 170, n^{l_3} = 250$ samples.

The HOIP problem deals with the composition of HOIP crystals, which are a relatively recently developed class of materials with desirable photovoltaic properties for applications in solar cells [64]. The quantity of interest is the inter-molecular binding energy between a HOIP and solvent pair used in a solar cell, which is used as an indicator of the cell's photovoltaic conversion efficiency. HOIPs are perovskite materials and therefore their crystals have an ABX_3 configuration, i.e., they are composed of three compounds corresponding to sites A (organic or inorganic cation), B (inorganic metal cation), and X_3 (halide combination). There are a number of choices for each site and any combination of these choices is feasible, so the possible HOIP compositions comprise a very large combinatorial space. The data set we use in this paper has three categorical inputs with $l_1 = 10$, $l_2 = 3$, and $l_3 = 16$ levels which

Table 4
Table of analytic functions: The analytic examples have different input dimensionality, number of sources, and forms of model error. n denotes the number of samples, σ^2 is the variance of the noise, and NRMSE is the normalized root mean squared error of an LF source with respect to an HF source, see Eq. (A.1).

Name	Source ID	Formulation	n	σ^2	NRMSE
Rational	$y^h(x)$	$\frac{1}{0.1x^3+x^2+x+1}$	5	0.001	-
	$y^{l_1}(x)$	$\frac{1}{0.2x^3+x^2+x+1}$	30	0.001	0.23
	$y^{l_2}(x)$	$\frac{1}{0\times x^3+x^2+x+1}$	30	0.001	0.15
	$y^{l_3}(x)$	$\frac{1}{0\times x^3+x^2+0\times x+1}$	30	0.001	0.73
Polynomial[a]	$y^h(\boldsymbol{x})$	$4x_1^2-2.1x_1^4+\frac{x_1^6}{3} - 4x_2^2+4x_2^4+x_1x_2$	5	1	-
	$y^l(\boldsymbol{x})$	$y^h\left(0.7x_1, 0.7x_2\right)+x_1x_2-65$	20	1	5.77
Wing-weight	$y^h(\boldsymbol{x})$	$0.036S_\omega^{0.758}W_{f\omega}^{0.0035}\left(\frac{A}{\cos^2(\Lambda)}\right)^{0.6}q^{0.006}\times \lambda^{0.04}\left(\frac{100t_c}{\cos(\Lambda)}\right)^{-0.3}\left(N_zW_{dg}\right)^{0.49}+S_\omega W_p$	15	25	-
	$y^{l_1}(\boldsymbol{x})$	$0.036S_\omega^{0.758}W_{f\omega}^{0.0035}\left(\frac{A}{\cos^2(\Lambda)}\right)^{0.6}q^{0.006}\times \lambda^{0.04}\left(\frac{100t_c}{\cos(\Lambda)}\right)^{-0.3}\left(N_zW_{dg}\right)^{0.49}+1\times W_p$	50	25	0.20
	$y^{l_2}(\boldsymbol{x})$	$0.036S_\omega^{0.8}W_{f\omega}^{0.0035}\left(\frac{A}{\cos^2(\Lambda)}\right)^{0.6}q^{0.006}\times \lambda^{0.04}\left(\frac{100t_c}{\cos(\Lambda)}\right)^{-0.3}\left(N_zW_{dg}\right)^{0.49}+1\times W_p$	50	25	1.14
	$y^{l_3}(\boldsymbol{x})$	$0.036S_\omega^{0.9}W_{f\omega}^{0.0035}\left(\frac{A}{\cos^2(\Lambda)}\right)^{0.6}q^{0.006}\times \lambda^{0.04}\left(\frac{100t_c}{\cos(\Lambda)}\right)^{-0.3}\left(N_zW_{dg}\right)^{0.49}+0\times W_p$	50	25	5.75
Borehole	$y^h(\boldsymbol{x})$	$\frac{2\pi T_u(H_u-H_l)}{\ln\left(\frac{r}{r_w}\right)\left(1+\frac{2LT_u}{\ln\left(\frac{r}{r_w}\right)r_w^2k_w}+\frac{T_u}{T_l}\right)}$	15	6.25	-
	$y^{l_1}(\boldsymbol{x})$	$\frac{2\pi T_u(H_u-0.8H_l)}{\ln\left(\frac{r}{r_w}\right)\left(1+\frac{1LT_u}{\ln\left(\frac{r}{r_w}\right)r_w^2k_w}+\frac{T_u}{T_l}\right)}$	50	6.25	3.67
	$y^{l_2}(\boldsymbol{x})$	$\frac{2\pi T_u(H_u-3H_l)}{\ln\left(\frac{r}{r_w}\right)\left(1+\frac{8LT_u}{\ln\left(\frac{r}{r_w}\right)r_w^2k_w}+0.75\frac{T_u}{T_l}\right)}$	50	6.25	3.73
	$y^{l_3}(\boldsymbol{x})$	$\frac{2\pi T_u(1.1H_u-H_l)}{\ln\left(\frac{4r}{r_w}\right)\left(1+\frac{3LT_u}{\ln\left(\frac{r}{r_w}\right)r_w^2k_w}+\frac{T_u}{T_l}\right)}$	50	6.25	0.38
	$y^{l_4}(\boldsymbol{x})$	$\frac{2\pi T_u(1.05H_u-H_l)}{\ln\left(\frac{2r}{r_w}\right)\left(1+\frac{2LT_u}{\ln\left(\frac{r}{r_w}\right)r_w^2k_w}+\frac{T_u}{T_l}\right)}$	50	6.25	0.19

[a]Extracted from [62].

correspond to 10 possible X_3 halide compositions,[16] 3 possible choices for the organic/inorganic A site cation,[17] and 16 possible choices of solvent,[18] respectively. The B site cation is held constant as lead, as other choices are exceedingly rare [64]. The output is the inter-molecular binding energy between the input HOIP and solvent. There are one HF and three LF data sets with unknown levels of fidelity and $n^h = 480$, $n^{l_1} = 480$, $n^{l_2} = 179$, $n^{l_3} = 240$ samples. We use 90% of the available samples for each source for training and 10% for testing. We refer the reader to [64] for further details on HOIPs.

[16] All possible three-element combinations of chlorine, bromine, and iodine.
[17] Methylammonium, formamidinium, or cesium.
[18] Acetone, methanol, chloroform, etc.

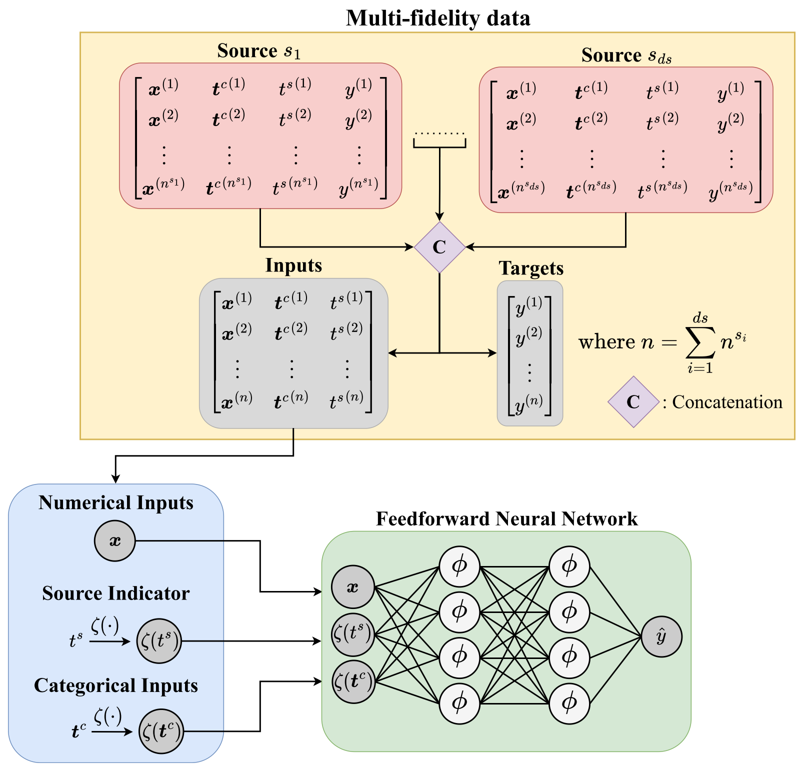

Fig. 11. FFNN for multi-fidelity modeling: As Pro-NDF, this approach allows to use an arbitrary number of sources by appending a source indicator variable to each data set and concatenating them. The FFNN maps the numerical inputs $\boldsymbol{x}$, a priori representation of the source indicator $\boldsymbol{\zeta}(t^s)$, and categorical inputs $\boldsymbol{\zeta}(\boldsymbol{t}^c)$ to the output.

Appendix C. Other multi-fidelity NN-based approaches

C.1. Feedforward neural networks

As depicted in Fig. 11, for MF modeling via an FFNN we simply feed the numerical inputs $\boldsymbol{x}$, the prior representation of the source indicator $\boldsymbol{\zeta}(t^s)$ and the prior representation of the categorical inputs $\boldsymbol{\zeta}(\boldsymbol{t}^c)$ into the FFNN to produce the output. This approach has two clear disadvantages with respect to Pro-NDF: (1) it does not provide a tool such as the fidelity manifold of Pro-NDF that provides a direct visualization of the correlation between the data sources, and (2) it has a fully deterministic setting which does not enable uncertainty quantification and thus using a loss function based on proper scoring rules. In particular, we use the following loss function for training the FFNN:

$$\mathcal{L} = \mathcal{L}_{MSE} + \beta \mathcal{L}_2 \tag{C.2}$$

where $\mathcal{L}_{MSE}$ is the mean squared error of the predictions and $\mathcal{L}_2$ is $L2$ regularization:

$$\mathcal{L}_{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - \hat{y}^{(i)})^2 \tag{C.3}$$

$$\mathcal{L}_2 = |\boldsymbol{\theta}|^2 \tag{C.4}$$

We employ Adam as the optimizer and use RayTune [60] and Hyperopt with five-fold cross-validation to find the optimum architecture and hyperparameters which include the learning rate, regularization parameter β, and batch size N. For further details on implementation, please see our GitLab repository.

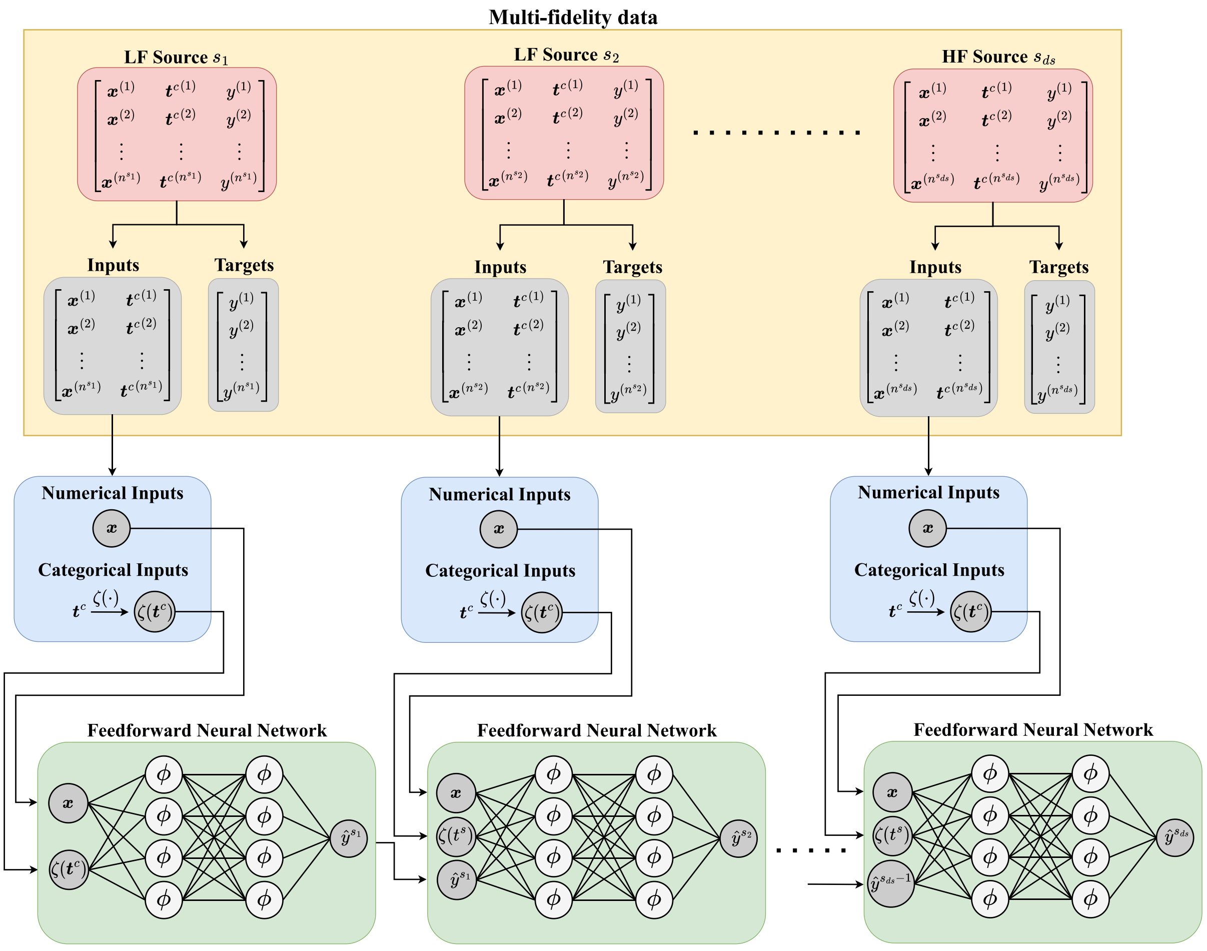

Fig. 12. Sequential Multi-fidelity (SMF) Networks: SMF is a hierarchical approach that relies on sequentially training a model (e.g., an FFNN) for each data source in an ascending order based on the fidelities. The inputs of a model are augmented with the outputs of the previous one until reaching the model of the HF source.

C.2. Sequential multi-fidelity networks

SMF is our extension of the approach detailed in [35] to more than two sources of data. Unlike the other methods presented in this paper, multi-fidelity modeling via SMF requires training a separate surrogate for each data source. As depicted in Fig. 12, individual FFNNs are trained for each source in the sequence that ends with the HF source. After a surrogate is trained for a data source, its outputs are used to augment the inputs of the next model in the sequence and hence the resulting input–output relationships are:

$$\begin{aligned}
\hat{y}^{s_1} &= \hat{f}^{s_1}\left(\boldsymbol{u}^{s_1}\right) \\
\hat{y}^{s_2} &= \hat{f}^{s_2}\left(\boldsymbol{u}^{s_2}, \hat{y}^{s_1}(\boldsymbol{u}^{s_2})\right) \\
&\cdots \\
\hat{y}^{s_{ds}} &= \hat{f}^{s_{ds}}\left(\boldsymbol{u}^{s_{ds}}, \hat{y}^{s_{ds-1}}(\boldsymbol{u}^{s_{ds}})\right)
\end{aligned} \tag{C.5}$$

where $\hat{y}^{s_i}$ is the output of the FFNN, $\hat{f}^{s_i}$ is the mapping defined by the FFNN, $\boldsymbol{u}^{s_i}$ is the combined numeric and categorical input $\boldsymbol{u} = [\boldsymbol{x}, \boldsymbol{\zeta}(\boldsymbol{t}^c)]$, and i denotes the data source with $i = ds$ being the HF source. Each individual FFNN employs the same loss function and optimizer as in the FFNN method presented in Appendix C.1.

Like MF-DGP, the SMF approach is highly sensitive to the ordering of the data sources in the sequence. In the case that the fidelity levels are known, they are assigned in the order of increasing fidelity, i.e., source 1 is the

Table 5
Training time for different models on each example (in seconds).

	Rational	Wing-weight	Borehole	DNS-ROM	HOIP
Pro-NDF	195.80	65.99	91.32	35.96	32.43
LMGP	3.35	49.40	133.88	127.53	112.95
MF-DGP[a,b]	304.12	983.16	2312.01	3533.04	–
SF-GP	6.43	5.80	4.88	5.10	12.76
FFNN	75.42	418.39	71.11	183.94	387.55
SMF	96.76	108.99	476.85	165.05	935.42

[a]Simulations for MF-DGP were conducted on a workstation equipped with an 11th Gen Intel Core i7-11700K CPU, with a clock speed of 3.60 GHz, 8 cores and 16 threads.
[b]MF-DGP cannot handle categorical variables and hence is not applied to the HOIP example.

least accurate LF source while source $ds-1$ is the most accurate. With this ordering, the SMF approach leverages the entire data set to achieve good HF prediction accuracy by minimizing the complexity of the mapping learned by each successive FFNN. However, in the case that the fidelities are not known, the order of the LF sources is assigned randomly. In this case, the mappings of the successive FFNNs no longer monotonically approaches that of the HF function, and the SMF approach is unable to properly leverage the additional LF data. In this paper, we assume that the fidelity levels are unknown and therefore assign the data source ordering randomly when using SMF.

Similar to the FFNN approach, the SMF approach does not provide a latent mapping and is entirely deterministic. Like all hierarchical approaches, it also requires knowledge of fidelity levels for good performance. These factors lead to a marked disadvantage in the context of the problems examined in this paper, and we therefore expect the SMF method to perform poorly.

We use RayTune and Hyperopt with five-fold cross-validation to find the optimum architecture and hyperparameters for *each* FFNN in the SMF method. Namely, we tune the learning rate, regularization parameter β, and batch size N. We also tune an additional parameter that determines whether to use the numeric and categorical inputs $\boldsymbol{u}$ in the final FFNN, since the mapping may be simple enough to learn from just the previous FFNN outputs in the case that the last LF source is highly accurate. For further details on implementation, please see our GitLab reposi tory.

Appendix D. Additional details on simulations

D.1. Computational time

The required training time for each MF approach and problem assessed in Sections 4.2 and 4.3 is shown in Table 5.

References

[1] Ghanshyam Pilania, James E. Gubernatis, Turab Lookman, Multi-fidelity machine learning models for accurate bandgap predictions of solids, Comput. Mater. Sci. 129 (2017) 156–163.

[2] Shiguang Deng, Carlos Mora, Diran Apelian, Ramin Bostanabad, Data-driven calibration of multifidelity multiscale fracture models via latent map Gaussian process, J. Mech. Des. 145 (1) (2023) 011705.

[3] Xiaotong Liu, Pierre-Paul De Breuck, Linghui Wang, Gian-Marco Rignanese, A simple denoising approach to exploit multi-fidelity data for machine learning materials properties, 2022, arXiv preprint arXiv:2204.10430.

[4] Shiguang Deng, Diran Apelian, Ramin Bostanabad, Adaptive spatiotemporal dimension reduction in concurrent multiscale damage analysis, Comput. Mech. (ISSN: 1432-0924) (2023) http://dx.doi.org/10.1007/s00466-023-02299-7.

[5] Zahra Zanjani Foumani, Mehdi Shishehbor, Amin Yousefpour, Ramin Bostanabad, Multi-fidelity cost-aware Bayesian optimization, Comput. Methods Appl. Mech. Engrg. (ISSN: 0045-7825) 407 (2023) 115937, http://dx.doi.org/10.1016/j.cma.2023.115937, URL https://www.sciencedirect.com/science/article/pii/S0045782523000609.

[6] Souvik Chakraborty, Tanmoy Chatterjee, Rajib Chowdhury, Sondipon Adhikari, A surrogate based multi-fidelity approach for robust design optimization, Appl. Math. Model. 47 (2017) 726–744.

[7] Péter Zénó Korondi, Mariapia Marchi, Lucia Parussini, Carlo Poloni, Multi-fidelity design optimisation strategy under uncertainty with limited computational budget, Opt. Eng. 22 (2) (2021) 1039–1064.
[8] Ghina N. Absi, Sankaran Mahadevan, Multi-fidelity approach to dynamics model calibration, Mech. Syst. Signal Process. 68 (2016) 189–206.
[9] Sanaz Zanjani Foumani, Mehdi Shishehbor, Amin Yousefpour, Ramin Bostanabad, Multi-fidelity cost-aware Bayesian optimization, 2022, Available At SSRN 4268166.
[10] Siyu Tao, Daniel W Apley, Wei Chen, Andrea Garbo, David J Pate, Brian J German, Input mapping for model calibration with application to wing aerodynamics, AIAA J. 57 (7) (2019) 2734–2745.
[11] Slawomir Koziel, Qingsha S. Cheng, John W. Bandler, Space mapping, IEEE Microw. Mag. 9 (6) (2008) 105–122.
[12] John W. Bandler, Radoslaw M. Biernacki, Shao Hua Chen, Piotr A. Grobelny, Ronald H. Hemmers, Space mapping technique for electromagnetic optimization, IEEE Trans. Microw. Theory Tech. 42 (12) (1994) 2536–2544.
[13] Anand Amrit, Leifur Leifsson, Slawomir Koziel, Fast multi-objective aerodynamic optimization using sequential domain patching and multifidelity models, J. Aircr. 57 (3) (2020) 388–398.
[14] Slawomir Koziel, Leifur Leifsson, Multi-level CFD-based airfoil shape optimization with automated low-fidelity model selection, Procedia Comput. Sci. 18 (2013) 889–898.
[15] Leifur Leifsson, Slawomir Koziel, Aerodynamic shape optimization by variable-fidelity computational fluid dynamics models: a review of recent progress, J. Comput. Sci. 10 (2015) 45–54.
[16] Marc C. Kennedy, Anthony O'Hagan, Bayesian calibration of computer models, J. R. Stat. Soc. Ser. B Stat. Methodol. 63 (3) (2001) 425–464.
[17] John McFarland, Sankaran Mahadevan, Multivariate significance testing and model calibration under uncertainty, Comput. Methods Appl. Mech. Eng. 197 (29-32) (2008) 2467–2479.
[18] Matthew Plumlee, Bayesian calibration of inexact computer models, J. Amer. Statist. Assoc. 112 (519) (2017) 1274–1285.
[19] Dave Higdon, Marc Kennedy, James C. Cavendish, John A. Cafeo, Robert D. Ryne, Combining field data and computer simulations for calibration and prediction, SIAM J. Sci. Comput. 26 (2) (2004) 448–466.
[20] Daniel W. Apley, Jun Liu, Wei Chen, Understanding the effects of model uncertainty in robust design with computer experiments, 2006.
[21] Maria J. Bayarri, James O. Berger, Rui Paulo, Jerry Sacks, John A. Cafeo, James Cavendish, Chin-Hsu Lin, Jian Tu, A framework for validation of computer models, Technometrics 49 (2) (2007) 138–154.
[22] Paul D. Arendt, Daniel W. Apley, Wei Chen, David Lamb, David Gorsich, Improving identifiability in model calibration using multiple responses, 2012.
[23] Paul D. Arendt, Daniel W. Apley, Wei Chen, Quantification of model uncertainty: Calibration, model discrepancy, and identifiability, 2012.
[24] David A. Stainforth, Tolu Aina, Carl Christensen, Mat Collins, Nick Faull, Dave J. Frame, Jamie A. Kettleborough, S. Knight, A. Martin, J.M. Murphy, et al., Uncertainty in predictions of the climate response to rising levels of greenhouse gases, Nature 433 (7024) (2005) 403–406.
[25] Weizhao Zhang, Ramin Bostanabad, Biao Liang, Xuming Su, Danielle Zeng, Miguel A. Bessa, Yanchao Wang, Wei Chen, Jian Cao, A numerical Bayesian-calibrated characterization method for multiscale prepreg preforming simulations with tension-shear coupling, Compos. Sci. Technol. 170 (2019) 15–24.
[26] Robert B. Gramacy, Derek Bingham, James Paul Holloway, Michael J. Grosskopf, Carolyn C. Kuranz, Erica Rutter, Matt Trantham, R Paul Drake, Calibrating a large computer experiment simulating radiative shock hydrodynamics, Ann. Appl. Stat. 9 (3) (2015) 1141–1168.
[27] Lluis Jofre, Gianluca Geraci, Hillary Fairbanks, Alireza Doostan, Gianluca Iaccarino, Multi-fidelity uncertainty quantification of irradiated particle-laden turbulence, 2018, arXiv preprint arXiv:1801.06062.
[28] Alex A. Gorodetsky, John D. Jakeman, Gianluca Geraci, MFNets: data efficient all-at-once learning of multifidelity surrogates as directed networks of information sources, Comput. Mech. (ISSN: 1432-0924) 68 (4) (2021) 741–758, http://dx.doi.org/10.1007/s00466-021-02042-0.
[29] Alex A. Gorodetsky, John D. Jakeman, Gianluca Geraci, Michael S. Eldred, MFNets: multi-fidelity data-driven networks for Bayesian learning and prediction, Int. J. Uncertain. Quantif. 10 (6) (2020).
[30] Rebecca E. Morrison, Todd A. Oliver, Robert D. Moser, Representing model inadequacy: A stochastic operator approach, SIAM/ASA J. Uncertain. Quantif. 6 (2) (2018) 457–496.
[31] Rebecca E. Morrison, Embedded discrepancy operators in reduced models of interacting species, 2019, arXiv preprint arXiv:1910.08191.
[32] Teresa Portone, Damon McDougall, Robert D. Moser, A stochastic operator approach to model inadequacy with applications to contaminant transport, 2017, arXiv preprint arXiv:1702.07779.
[33] Jonathan Tammer Eweis-Labolle, Nicholas Oune, Ramin Bostanabad, Data fusion with latent map Gaussian processes, J. Mech. Des. 144 (9) (2022) 091703.
[34] Ian Goodfellow, Yoshua Bengio, Aaron Courville, Deep Learning, MIT Press, ISBN: 0262337371, 2016.
[35] Liang Yan, Tao Zhou, An adaptive surrogate modeling based on deep neural networks for large-scale Bayesian inverse problems, 2019.
[36] Xuhui Meng, George Em Karniadakis, A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse PDE problems, J. Comput. Phys. 401 (2020) 109020.
[37] Subhayan De, Jolene Britton, Matthew Reynolds, Ryan Skinner, Kenneth Jansen, Alireza Doostan, On transfer learning of neural networks using bi-fidelity data for uncertainty propagation, Int. J. Uncertain. Quantif. 10 (6) (2020).
[38] Suraj Pawar, Omer San, Prakash Vedula, Adil Rasheed, Trond Kvamsdal, Multi-fidelity information fusion with concatenated neural networks, Sci. Rep. (2022).

[39] Tilmann Gneiting, Adrian E. Raftery, Strictly proper scoring rules, prediction, and estimation, J. Amer. Statist. Assoc. 102 (477) (2007) 359–378.

[40] Nicholas Oune, Ramin Bostanabad, Latent map Gaussian processes for mixed variable metamodeling, Comput. Methods Appl. Mech. Engrg. 387 (2021) 114128.

[41] Yann LeCun, Yoshua Bengio, Geoffrey Hinton, Deep learning, Nature 521 (7553) (2015) 436–444, URL https://www.nature.com/articles/nature14539.

[42] Kurt Hornik, Maxwell Stinchcombe, Halbert White, Multilayer feedforward networks are universal approximators, Neural Netw. 2 (5) (1989) 359–366.

[43] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, Daan Wierstra, Weight uncertainty in neural network, in: International Conference on Machine Learning, PMLR, 2015, pp. 1613–1622.

[44] Chuan Guo, Geoff Pleiss, Yu Sun, Kilian Q. Weinberger, On calibration of modern neural networks, in: International Conference on Machine Learning, PMLR, 2017, pp. 1321–1330.

[45] John Mitros, Brian Mac Namee, On the validity of Bayesian neural networks for uncertainty estimation, 2019, arXiv preprint arXiv:1912.01530.

[46] Agustinus Kristiadi, Matthias Hein, Philipp Hennig, Being Bayesian, even just a bit, fixes overconfidence in relu networks, in: International Conference on Machine Learning, PMLR, 2020, pp. 5436–5446.

[47] W. Keith Hastings, Monte Carlo sampling methods using Markov chains and their applications, Oxford University Press, 1970.

[48] David M. Blei, Alp Kucukelbir, Jon D. McAuliffe, Variational inference: A review for statisticians, J. Amer. Statist. Assoc. 112 (518) (2017) 859–877.

[49] Laurent Valentin Jospin, Hamid Laga, Farid Boussaid, Wray Buntine, Mohammed Bennamoun, Hands-on bayesian neural networks—a tutorial for deep learning users, IEEE Comput. Intell. Mag. 17 (2) (2022) 29–48.

[50] S.T. Roweis, L.K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (5500) (2000) 2323–2326, http://dx.doi.org/10.1126/science.290.5500.2323, ISSN: 0036-8075 (Print) 0036-8075 (Linking).

[51] D.L. Donoho, C. Grimes, Hessian eigenmaps: locally linear embedding techniques for high-dimensional data, Proc. Natl. Acad. Sci. USA 100 (10) (2003) 5591–5596, http://dx.doi.org/10.1073/pnas.1031596100, ISSN: 0027-8424 (Print) 0027-8424 (Linking) URL https://www.ncbi.nlm.nih.gov/pubmed/16576753.

[52] J.B. Tenenbaum, V. de Silva, J.C. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (5500) (2000) 2319–2323, http://dx.doi.org/10.1126/science.290.5500.2319, ISSN: 0036-8075 (Print) 0036-8075 (Linking).

[53] Ashutosh Saxena, Abhinav Gupta, Amitabha Mukerjee, Non-linear dimensionality reduction by locally linear isomaps, in: Neural Information Processing, Springer, pp. 1038–1043.

[54] Ronald R. Coifman, Stéphane Lafon, Diffusion maps, Appl. Comput. Harmon. Anal. (ISSN: 10635203) 21 (1) (2006) 5–30, http://dx.doi.org/10.1016/j.acha.2006.04.006, URL http://www.sciencedirect.com/science/article/pii/S1063520306000546.

[55] N. Lawrence, Probabilistic non-linear principal component analysis with Gaussian process latent variable models, J. Mach. Learn. Res. (ISSN: 1532-4435) 6 (Nov) (2005) 1783–1816, <Go to ISI>://WOS:000236330700002.

[56] Francois Chollet, Deep Learning with Python, Manning Publications Co, ISBN: 1617294438, 2017.

[57] Alexander Amini, Wilko Schwarting, Ava Soleimany, Daniela Rus, Deep evidential regression, in: Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS '20, Curran Associates Inc., Red Hook, NY, USA, ISBN: 9781713829546, 2020, URL https://proceedings.neurips.cc/paper/2020/hash/aab085461de182608ee9f607f3f7d18f-Abstract.html.

[58] Hippolyt Ritter, Aleksandar Botev, David Barber, A scalable laplace approximation for neural networks, in: 6th International Conference on Learning Representations, ICLR 2018-Conference Track Proceedings, 6, International Conference on Representation Learning, 2018.

[59] Meire Fortunato, Charles Blundell, Oriol Vinyals, Revisiting Bayes by backprop, 2018.

[60] Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E Gonzalez, Ion Stoica, Tune: A research platform for distributed model selection and training, 2018, arXiv preprint arXiv:1807.05118.

[61] Kurt Cutajar, Mark Pullin, Andreas Damianou, Neil Lawrence, Javier González, Deep gaussian processes for multi-fidelity modeling, 2019, arXiv preprint arXiv:1903.07320.

[62] Lixue Liu, Xueguan Song, Chao Zhang, Dacheng Tao, GAN-MDF: An enabling method for multifidelity data fusion, IEEE Internet Things J. 9 (15) (2022) 13405–13415.

[63] Yarin Gal, Zoubin Ghahramani, Dropout as a Bayesian approximation: Representing model uncertainty in deep learning, in: Proceedings of the 33rd International Conference on International Conference on Machine Learning - Vol. 48, ICML '16, JMLR.org, 2016, pp. 1050–1059.

[64] Henry C. Herbol, Weici Hu, Peter Frazier, Paulette Clancy, Matthias Poloczek, Efficient search of compositional space for hybrid organic–inorganic perovskites via bayesian optimization, Npj Comput. Mater. 4 (1) (2018) http://dx.doi.org/10.1038/s41524-018-0106-7.