Delft University of Technology

Single-to-multi-fidelity history-dependent learning with uncertainty quantification and disentanglement
Application to data-driven constitutive modeling

Yi, Jiaxiang; Ferreira, Bernardo P.; Bessa, Miguel A.

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Single-to-multi-fidelity history-dependent learning with uncertainty quantification and disentanglement: Application to data-driven constitutive modeling [*]

Jiaxiang Yi [a], Bernardo P. Ferreira [b], Miguel A. Bessa [b,*]

[a] *Faculty of Mechanical Engineering, Delft University of Technology, Mekelweg 2, Delft, 2628 CD, The Netherlands*
[b] *School of Engineering, Brown University, 184 Hope St., Providence, RI 02912, USA*

ARTICLE INFO

ABSTRACT

Data-driven learning is generalized to consider history-dependent multi-fidelity data, while quantifying epistemic uncertainty and disentangling it from data noise (aleatoric uncertainty). This generalization is hierarchical and adapts to different learning scenarios: from training the simplest single-fidelity deterministic neural networks up to the proposed multi-fidelity variance estimation Bayesian recurrent neural networks. The proposed methodology is demonstrated by applying it to different data-driven constitutive modeling scenarios for history-dependent plasticity of elastoplastic biphasic materials that include multiple fidelities with and without aleatoric uncertainty (noise). The method accurately predicts the response and quantifies model error while also discovering the noise distribution (when present). The versatility and generality of the proposed method open opportunities for future real-world applications in diverse scientific and engineering domains; especially, the most challenging cases involving design and analysis under uncertainty.

1. Introduction

Data-driven learning focuses on the use of machine learning methods to learn non-trivial models from data. Although these models can include Physics constraints [1,2], in general, they require large quantities of data [3]. Unfortunately, creating large datasets with **high-fidelity (HF)** data can be challenging, as it involves high-throughput or a vast number of parallel low-throughput experiments or simulations. By definition, HF datasets require costly simulations or experiments that are time-consuming or resource-intensive to conduct.[1] Conversely, **low-fidelity (LF)** datasets can be generated faster but have higher error and can be stochastic (noisy). In practice, **multi-fidelity (MF)** datasets may be necessary to balance the advantages and disadvantages of each fidelity.

Without loss of generality, the accuracy-efficiency paradox addressed by MF modeling will be illustrated in this article by focusing on data-driven constitutive modeling of materials. Although the idea of using neural networks as material models can be traced back to the early 90s [4], it remained largely dormant for more than two decades. However, the advent of high-performance automatic differentiation libraries and high-throughput data generation strategies, such as simulations of **representative volume elements**

---

[*] The code developed for this article is available: https://github.com/bessagroup/MF-VeBRNN
[*] Corresponding author.
  E-mail address: miguel_bessa@brown.edu (M.A. Bessa).
[1] This article associates "cost" to time and other resources (computational or experimental) that are needed to run a computer simulation or a physical experiment, such that a sample can be collected into a dataset. High cost implies that it is time-consuming or resource-intensive to generate a large dataset.
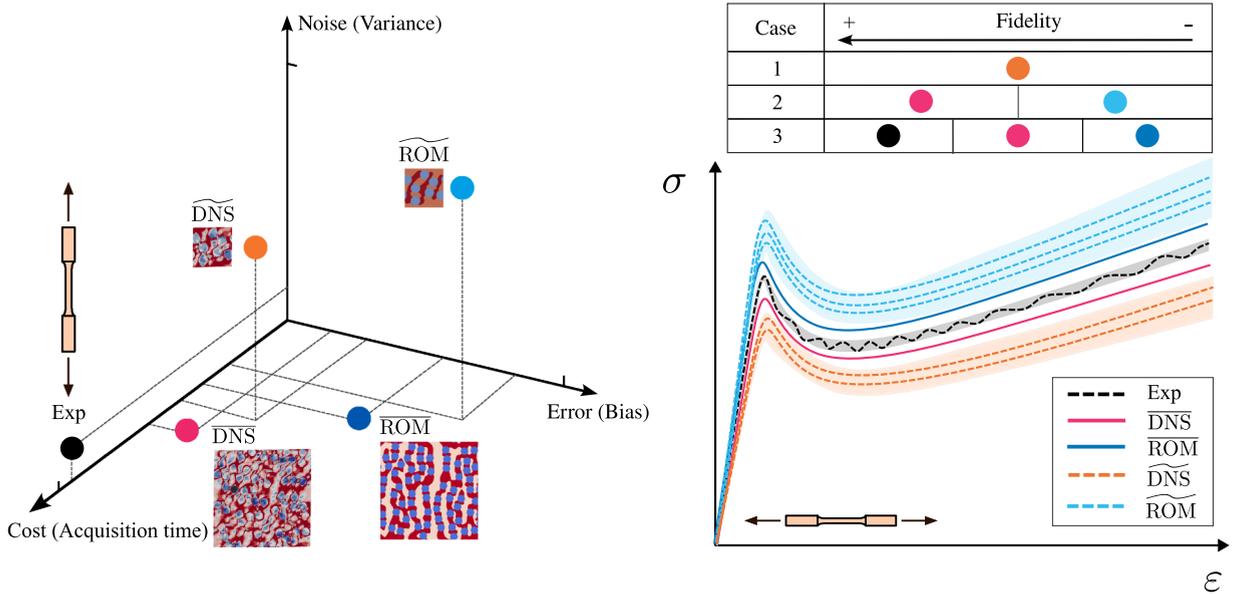
Fig. 1. Schematic summarizing different data generation methods for discovering the constitutive behavior of materials. Each method and corresponding dataset from which it originates is organized according to cost, noise, and accuracy. A method or a dataset is considered to have **high-fidelity (HF)** if it has lower error than another method, which is considered to have **low-fidelity (LF)**. Experiments are usually the most expensive and slowest method to acquire data (high cost), but provide the lowest error, despite the presence of some noise (aleatoric uncertainty). A **deterministic direct numerical simulation** ($\overline{\mathrm{DNS}}$) is obtained in this work by **finite element analysis (FEA)** of a **representative volume element (RVE)**, i.e., a sufficiently large material domain that generates noiseless data. A **stochastic DNS** ($\widetilde{\mathrm{DNS}}$) is obtained by FEA of a **stochastic volume element (SVE)**, i.e., a smaller material domain that is faster to simulate but generates noisy data. A **deterministic reduced order model** ($\overline{\mathrm{ROM}}$) is obtained by **self-consistent clustering analysis (SCA)** of an RVE (noiseless data with lower-fidelity than FEA but faster to acquire). And lastly, a **stochastic ROM** ($\widetilde{\mathrm{ROM}}$) is obtained by SCA of an SVE (noisy data with the lowest fidelity and fastest acquisition time).

(RVEs)[2] of materials, were shown to create sufficiently large datasets for training accurate models [5]. This can be achieved by finite element analyses of thousands of RVEs when they are composed of elastic or hyperelastic materials [6], but it becomes more challenging when they undergo history-dependent phenomena such as plasticity because the simulations are slower [5]. A viable pathway is to consider faster simulation methods, e.g., the **self-consistent clustering analysis (SCA)** [7], at the expense of lower accuracy (lower fidelity). Once a sufficiently large dataset is available, recurrent neural networks have been found to learn the complete history-dependent constitutive behavior of materials [8]. Since then, several researchers followed data-driven strategies to learn constitutive models for a wide range of materials, e.g., Wu et al. [9], He and Chen [10], Eghbalian et al. [11], Kalina et al. [12], Jadoon et al. [13], including their use to accelerate topology optimization [14].

Fig. 1 exemplifies five different methods of generating datasets with different fidelities, and organizes them according to three characteristics: error, cost, and noise. As indicated in the figure, data can be collected from physical experiments but also from synthetic experiments via deterministic (noiseless) or stochastic (noisy) computer simulations[3]. In addition, computer simulations can be divided into two types: **direct numerical simulations (DNSs)** and **reduced order models (ROMs)**.

Usually, the most accurate but least efficient method of data collection is to perform a real experiment of the material (indicated by a black circle in Fig. 1). Data collected from experiments has low error, high cost, and low noise. Noise is illustrated as the shaded black average stress–strain response in the right part of Fig. 1, arising from measurement uncertainties (sensor signals and data acquisition), manufacturing and preparation variations, intrinsic material heterogeneity, and setup-related imperfections. Experimental data is usually scarce and may not be available in sufficient quantity to train a neural network constitutive law [15].

In addition to conducting real experiments, performing a deterministic DNS (denoted as $\overline{\mathrm{DNS}}$) of a material is the best way to collect high-fidelity data. Deterministic DNSs have low error, high cost, and no noise (no aleatoric uncertainty), and they aim to simulate the experimental response as closely as possible. Fig. 1 illustrates this type of simulation in magenta. A deterministic DNS is typically obtained by finite element analysis (FEA) of an RVE.

A stochastic DNS (denoted as $\widetilde{\mathrm{DNS}}$) of a material provides a simple way to reduce cost when compared to a deterministic DNS because it considers a smaller material domain that is no longer representative. Such material domains are usually called **stochastic**

---

[2] Recall that an RVE is a sufficiently large domain of a material that is representative of that material's mechanical behavior, i.e., it leads to the same response even after randomizing the microstructure [5].

[3] Stochastic computer simulations can simulate a noisy response, e.g., by randomizing the microstructure in a material domain while keeping the same descriptors of geometry, constituent properties and boundary conditions.

volume elements (SVEs) because randomization of the microstructure leads to different responses (indicated in orange in Fig. 1). Therefore, a stochastic DNS has noise and typically higher error than a deterministic DNS, but leads to faster data collection.

Typically, DNSs are not sufficiently efficient to create large enough databases when the material undergoes history-dependent phenomena like plasticity deformation, even when considering SVEs instead of RVEs. In that case, **reduced-order models (ROMs)** must be considered. In this article, we choose the ROM to be the previously mentioned SCA method [7], as it can predict the elastoplastic responses of RVEs and SVEs faster than the finite element method used as DNS. The SCA method and its adaptive formulation [16] recently became open-source [17], so these (and all other) results obtained in this article can be fully replicated. By definition, a deterministic ROM (denoted as $\overline{\text{ROM}}$) is noiseless and leads to faster predictions than the corresponding deterministic DNS (less cost), but its predictions have a higher error. A stochastic ROM ($\widetilde{\text{ROM}}$) leads to the fastest predictions because it uses a fast simulation method (like SCA) for estimating the behavior of a small material domain, but introduces noise and is associated with the highest error.

Summary of contribution. This article generalizes data-driven learning to all types of datasets exemplified in Fig. 1. This requires the ability to learn deterministically or probabilistically from datasets that are noiseless or noisy, single- or multi-fidelity, history-dependent or independent. Importantly, the ability to estimate both aleatoric and epistemic uncertainties[4] is also considered because data-driven models are difficult to interpret, and their trustworthiness improves with appropriate uncertainty quantification. Disentangling aleatoric uncertainty from epistemic uncertainty not only facilitates the updating of constitutive models based on experimental data [15], but also supports the design of robust materials by optimizing mechanical performance while accounting for data variation arising from unknown or uncontrolled manufacturing factors. This capability further enables scalable active learning and Bayesian optimization [18] with more informative acquisition functions increasing efficiency, although this is not explored herein.

The proposed generalization requires merging three separate machine learning contributions: (1) scalable **Bayesian recurrent neural networks (BRNNs)**, (2) efficient uncertainty disentanglement, and (3) multi-fidelity neural network architectures trainable on large datasets. We have recently contributed to these fields separately [19,20] with the aim of merging them into a general learning paradigm, as presented herein.

Remark 1. The importance of uncertainty quantification and/or disentanglement depends on the application.

- Estimation of aleatoric uncertainty is relevant for risk-aware applications. In these cases, it is common to design based on the lower bound of the 95 % confidence interval for a quantity of interest (e.g., fracture toughness). For example, for a quantity of interest following a Gaussian distribution, the aim becomes predicting the mean minus two standard deviations of the response. This is only possible if the aleatoric uncertainty is estimated accurately.
- Estimation of epistemic uncertainty is needed when quantifying model error is desired. Active learning and Bayesian optimization are the most important scenarios, but sometimes the analyst may simply want to know the error of a trained model (rigorously).
- Disentangling uncertainties is needed when **both** types of uncertainty are required, i.e., when considering risk-aware scenarios and conducting active learning or Bayesian optimization. For example, using Bayesian optimization to design a recycled material such that it does not fail with 95 % certainty (recycled materials usually have high aleatoric uncertainty due to the unknown origin of their components).
- Neither aleatoric nor epistemic uncertainties are necessary if the only goal is to accurately predict the mean response without quantifying the error of this prediction.[5]

Related work on uncertainty quantification. Uncertainty quantification has become an important topic in machine learning [21–23], as it facilitates decision-making and design optimization [24–26]. Initially, most solutions relied on Gaussian process regression [27], and assumed homoscedastic aleatoric uncertainty (constant noise). Even when considering heteroscedastic noise [28], the main bottleneck of Gaussian processes is their lack of scalability due to the "curse of dimensionality". In principle, Bayesian neural networks (BNNs) do not have this limitation but their adoption in Engineering practice has been slow because Bayesian inference is challenging [29,30]. Various approximate Bayesian inference methods have been proposed, such as Markov chain Monte Carlo (MCMC) sampling [31], Variational Inference (VI) [32], MC-Dropout [33], and Deep Ensembles [34]. This has enabled the recent application of BNNs to cases without history-dependency, such as estimating material properties [35–38] and learning hyperelastic constitutive models [39]. However, enabling uncertainty quantification of history-dependent or path-dependent phenomena, such as material plasticity, requires BRNNs.

Related work on Bayesian recurrent neural networks and uncertainty disentanglement. Literature on BRNNs is scarce, but the interested reader is referred to [40–42] for early work on the topic. Two recent methodologies have improved their scalability [20,43]. In this work, we use the method we developed – **Variance estimation Bayesian recurrent neural networks (VeBRNNs)** [20] – because it is also able to separate epistemic and aleatoric uncertainties. VeBRNNs are scalable because inference is performed by **preconditioned stochastic gradient Langevin dynamics (pSGLD)** [44], and they effectively disentangle uncertainties due to the cooperative training

---

[4] There are two types of uncertainties: epistemic (or model) uncertainty, and aleatoric uncertainty (or noise). Epistemic uncertainty quantifies the error of model predictions; therefore, it reduces as more data is used to train the model (for infinite data, there is no model error). Aleatoric uncertainty refers to the noise that is inherent to the data; therefore, a measurement of a given system under the same input conditions does not lead to the same output when there is aleatoric uncertainty (noise). Aleatoric uncertainty is irreducible because even if we had infinite data, we could not eliminate noise as it results from unknown factors that affect the measurements.

[5] Note, however, that estimating aleatoric uncertainty can still be beneficial because it regularizes the model for noisy data.

(a) RVE vs SVE   (b) Strain inputs   (c) Stress outputs
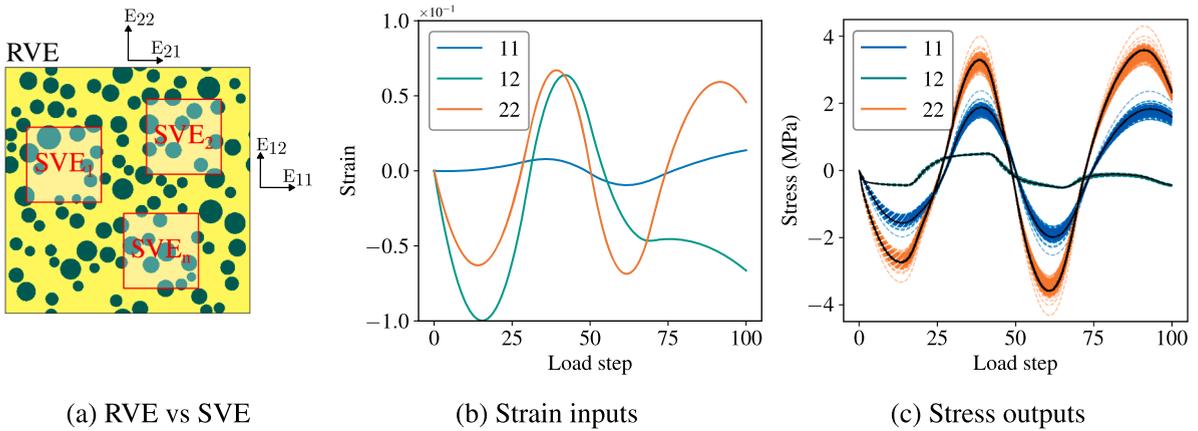
Fig. 2. RVE and SVE strain-stress path data: (a) illustration of biphasic material RVE and three SVEs; (b) example of random polynomial strain paths for the three strain components; and (c) comparison between the RVE (deterministic) homogenized stress response shown in solid lines, and the stochastic stress responses of 100 SVEs shown in dashed lines.

of a variance estimation network [45] and a BRNN. To the best of our knowledge, this method is the first that is capable of learning history-dependent constitutive behavior, estimating noise (aleatoric uncertainty), and predicting model (epistemic) uncertainty for small or large single-fidelity datasets.

Related work on multi-fidelity machine learning. Reliable data-driven constitutive laws require a substantial amount of HF data [5], but gathering it is often time-consuming and impractical. As previously referred, this is particularly relevant when involving complex experiments or expensive simulations [46]. Effective MF approaches can mitigate this challenge by leveraging the useful information embedded in LF data, thereby reducing the dependence on HF data [47,48]. MF machine learning has been successfully applied in various domains, such as solving expensive partial differential equations [49], optimization [50,51], and material design [25]. Initially, MF machine learning was limited to data scarce scenarios and low-dimensional problems by considering MF Gaussian process regression [27,52,53]. However, the literature has adopted deterministic methods when dealing with large datasets, e.g., using MF recurrent neural networks [54], or continual learning of subnetworks [55] for transferring knowledge without forgetting. The seamless integration of MF Bayesian machine learning for history-dependent problems remains unexplored. Moreover, existing studies do not explicitly address stochastic data in LF datasets, and its repercussions on the HF predictions.

2. Proposed datasets: Four different single- and multi-fidelity learning scenarios

We created four different scenarios based on Fig. 1 with the aim of illustrating the generality of the proposed work and to motivate future investigations on this topic by the research community. The datasets for all problems are obtained from simulations that capture history-dependent behavior of material volume elements (RVEs or SVEs for deterministic or stochastic data, respectively). The biphasic material volume elements (Fig. 2(a)) are subjected to synthetic random polynomial strain paths (Fig. 2(b)), from which the stress response is obtained (Fig. 2(c)). The biphasic material consists of:

- a matrix phase (yellow area of Fig. 2(a)) characterized by a von Mises elastoplastic model with isotropic strain hardening with elastic modulus $E_{\text{matrix}} = 100\,\text{MPa}$, Poisson's ratio $v_{\text{matrix}} = 0.3$, and a von Mises hardening law defined as $\sigma_y = 0.5 + 0.5(\bar{\epsilon}^p)^{0.4}$, where $\bar{\epsilon}^p$ denotes the accumulated plastic strain;
- and elastic particles (green circles of Fig. 2(a)) characterized by a linear elastic isotropic model with $E_{\text{fiber}} = 15\,\text{MPa}$ and $v_{\text{fiber}} = 0.19$.

As seen in Fig. 2(a), an SVE can be viewed as a subdomain of an RVE (although we still assume periodic microstructures for both). Different SVE realizations obtained by randomizing the microstructure (Fig. 2(a)) lead to different average stress responses (dashed lines in Fig. 2(c)), even when subjecting the SVEs to the same input deformation path (Fig. 2(b)).[6] In contrast, randomization of the microstructure of an RVE still leads to the same homogenized stress response (see the solid lines in Fig. 2(c), embedded in the dashed lines corresponding to different realizations of the SVE).

Remark 2. In this article, the datasets are obtained from computer simulations. Therefore, the aleatoric uncertainty of the proposed datasets arises primarily from the stochastic homogenization of SVEs, as illustrated in Fig. 2(c). However, experimental data contains aleatoric uncertainty that can arise from different sources (e.g., data acquisition, intrinsic material heterogeneity, manufacturing variations).

---

[6] Readers unfamiliar with homogenization theory are referred to [5,8], where it is explained how to convert an average strain state into a periodic boundary value problem of an RVE or SVE, and then calculate the corresponding average stress state.
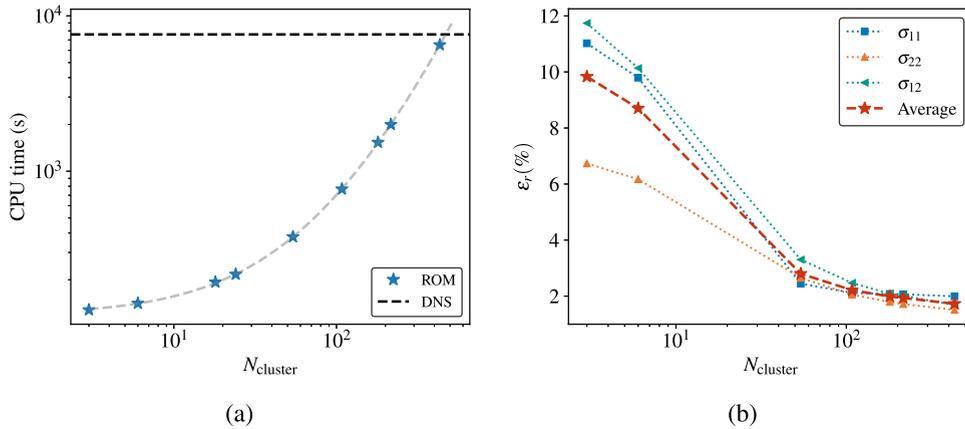
Fig. 3. Performance comparison between FEA-based DNS and SCA in the computational homogenization of biphasic microstructure RVE. The accuracy and computational cost of the SCA method depend on the number of clusters.

Without loss of generality, all learning problems in this article involve sequence-to-sequence regression where the input sequence is the average strain history applied to the material volume element and the output prediction is the average stress response for that path. Therefore, a nonlinear and history-dependent constitutive model is obtained by training a neural network on strain-stress paths such that:

$$\boldsymbol{\sigma}_{:} = f(\boldsymbol{\varepsilon}_{:}, \boldsymbol{\theta}), \tag{1}$$

where $\boldsymbol{\varepsilon}_{:}$ is the strain path, $\boldsymbol{\theta}$ is a set of hidden variables (e.g., weights and biases of a neural network f), and $\boldsymbol{\sigma}_{:}$ is the predicted stress path.

In the inference stage, the model recurrently predicts a stress state $\boldsymbol{\sigma}$ for a given input strain state $\boldsymbol{\varepsilon}$ (see [8]). Additional details on the generation of input strain paths are found in Appendix C.1, and on the origins of aleatoric uncertainty in this context are in Fig. 2 and Appendix C.3.

We also recall that in addition to considering an RVE or an SVE, we can also choose different simulation methods (FEA or SCA). We ensure that the FEA simulations have a sufficiently fine mesh to be considered DNSs, but we use different discretization levels for the SCA to achieve different accuracy-efficiency trade-offs (creating different fidelities from these ROMs). In SCA, the number of clusters is denoted by N_{cluster}, and it controls the trade-off between accuracy and computational efficiency compared to the DNS (FEA is employed as DNS). A comparison between accuracy and efficiency between $\overline{\text{ROM}}$ and $\overline{\text{DNS}}$ is presented in Fig. 3. The computational cost of SCA scales quadratically with N_{cluster}, so a larger number of clusters improves accuracy but increases computational expense. For instance, if we use only three clusters, we obtain a large stress prediction error ϵ_r of approximately 10 %, yet the corresponding CPU time is very low (almost negligible). Conversely, employing $N_{\text{cluster}} = 512$ achieves an average ϵ_r of around 2 %, i.e., slightly less accurate than the DNS obtained by FEA but already having comparable computational cost (that is why we choose FEA as DNS).

In summary, we generate different datasets by considering RVEs or SVEs, and simulating them with the FEA or SCA method. These four choices are used to create six different single-fidelity datasets, each of which with different cost, accuracy and noise, as presented in Table 1. Importantly, we introduce the concept of cost ratio to properly evaluate the computational cost of generating a single random strain-stress path for each dataset when compared to the most time consuming simulation, $\overline{\text{DNS}}$, which takes approximately 8000 s to obtain a path in a platform equipped with an AMD Ryzen™ 9 7945HX processor (1 core used), 16 GB RAM. The table also includes the number of paths used for training, for **in-distribution (ID)** testing, and for **out-of-distribution (OOD)** testing. The ID testing paths were considered using an applied strain range of $(\varepsilon_{ij,\text{min}}, \varepsilon_{ij,\text{max}}) = (-0.1, 0.1), \{i, j\} = 1, 2$. The OOD testing paths were obtained considering a wider strain range of $(\varepsilon_{ij,\text{min}}, \varepsilon_{ij,\text{max}}) = (-0.125, 0.125), \{i, j\} = 1, 2$. This means that the OOD paths include 25 % extrapolation beyond the training domains. Also note that for the testing paths of stochastic cases (SVEs), we repeat the simulations 100 times (indicated as ×100 in Table 1) – this is necessary to have a meaningful estimation of the ground truth aleatoric uncertainty associated to that fidelity, so that we can then compare with the predictions obtained by our models.

We created four different scenarios inspired by real-world applications and by considering data generated by one or two of the six fidelities shown in Table 1:

- **Dataset 1 (stochastic single fidelity):** $\widetilde{\text{DNS}}$ – The only single fidelity dataset considered in this article is created to mimic the *stochastic* experimental case. Therefore, we obtain this dataset by FEA of an SVE [7], leading to a noisy DNS dataset (recall the tilde on top indicates the data has noise).
- **Dataset 2 (deterministic LF and deterministic HF):** $\overline{\text{ROM}} + \overline{\text{DNS}}$ – A common scenario in the deterministic MF machine learning literature, where both fidelities are noiseless. HF data is obtained by FEA of an RVE, while LF data is obtained by SCA of the same RVE (recall that the bar on top indicates the data is noiseless).

---

[7] An SVE does not imply a fixed microstructure across the dataset; rather, each strain path corresponds to a random microstructure realization.

Table 1

Data collected with six different fidelity levels according to the choice of material volume element (RVE vs. SVE), simulation method (DNS vs. ROM) and discretization level (number of clusters used for ROM). The bar on top indicates that data is noiseless (obtained from an RVE). The tilde on top indicates that data is noisy (obtained from an SVE). ID means in-distribution, and OOD means out-of-distribution. Cost ratio indicates the approximate time it takes to simulate one stress–strain path compared to a deterministic DNS. Data types are ordered according to computational cost (decreasing cost from left to right).

Fidelity	$\overline{\text{DNS}}$	$\widetilde{\text{DNS}}$	$\overline{\text{ROM}}(N_{\text{cluster}} = 18)$	$\overline{\text{ROM}}(N_{\text{cluster}} = 3)$	$\widetilde{\text{ROM}}(N_{\text{cluster}} = 18)$	$\widetilde{\text{ROM}}(N_{\text{cluster}} = 3)$
Cost ratio (c)	1.0	1/20	1/36	1/60	1/120	1/200
Volume element	RVE	SVE	RVE	RVE	SVE	SVE
Simulation method	FEA	FEA	SCA	SCA	SCA	SCA
Noise (aleatoric uncert.)	No	Yes	No	No	Yes	Yes
# Training paths	1000	2000	2000	2000	2000	2000
# ID testing paths	100	100 (×100)	n/a	n/a	n/a	n/a
# OOD testing paths	100	100 (×100)	n/a	n/a	100 (×100)	100 (×100)

- **Dataset 3 (stochastic LF and stochastic HF):** $\widetilde{\text{ROM}} + \widetilde{\text{DNS}}$ – MF dataset where both fidelities are noisy. HF data is the same as in Dataset 1 (DNS of an SVE), while LF data is obtained by SCA of an SVE.
- **Dataset 4 (stochastic LF and deterministic HF):** $\widetilde{\text{ROM}} + \overline{\text{DNS}}$ – MF dataset where the HF is noiseless and the LF is noisy. This is also a common scenario, as HF data tends to have less (or negligible) noise when compared to LF data.

3. Proposed methodology

We aim to generalize data-driven learning to both single- and multi-fidelity problems, with or without history-dependency, and with the ability to disentangle uncertainties (if needed). Therefore, we created a flowchart (Fig. 4) that highlights the modular nature of the framework, hoping to facilitate navigation among possible choices. In the remainder of the article, we focus on the most general choices, and omit the simpler cases for brevity (those already exist in the literature, as we saw in the Introduction).

3.1. Neural network architecture choice

Fig. 5 presents a flowchart for choosing a neural network architecture that learns from data of a given fidelity. The most general architecture, VeBRNN, can be simplified into all other networks. As mentioned above, VeBRNNs are capable of single-fidelity history-dependent predictions with uncertainty quantification and disentanglement. This architecture is trained cooperatively between a variance estimation network and a Bayesian recurrent neural network, enabling multidimensional predictions of (1) mean response, (2) variance of aleatoric uncertainty (noise), and (3) variance of epistemic (model) uncertainty. The method is explained in detail in [20], but we summarize it here for completeness.

Assuming stress observations at each pseudo-time t (or load step) follow a Gaussian distribution, we can formulate the problem as follows,

$$f_j(\mathbf{x}_t) \sim \mathcal{N}\left(y_j(\mathbf{x}_t), s_j^2(\mathbf{x}_t)\right) \quad \text{where} \quad t = 1, \dots, T, \tag{2}$$

where $y_j(\mathbf{x})$ is the ground truth mean of the output component j (note that $\mathbf{y} \equiv \boldsymbol{\sigma}$ and that $j = 1, 2, 3$ because the stress has three components[8]), and where $s_j^2(\mathbf{x})$ is the variance of the aleatoric uncertainty of the appropriate output component (i.e., the covariance matrix of the Gaussian is diagonal). If $s_j^2(\mathbf{x})$ depends on the input $\mathbf{x}$ (strain components $\boldsymbol{\epsilon}$), it is referred as heteroscedastic aleatoric uncertainty; otherwise, it is homoscedastic. Fig. 2 and Fig. C3 clearly show that SVEs have heteroscedastic aleatoric uncertainty (noise).

Algorithm 1 contains the pseudo-code for the cooperative training of the VeBRNN. The algorithm involves training a deterministic RNN in Step 1, then a variance estimation RNN for step 2, and a BRNN in Step 3.[9] If no aleatoric uncertainty estimation is needed, then Step 2 can be skipped. If no epistemic uncertainty is required, then step 3 can be skipped (and the architecture is simply a deterministic RNN – Step 1). Appendix D describes the 3 steps, but a complete description is also provided in the original publication [20]. Section 4.1 presents results obtained from training a VeBRNN on Dataset 1 (obtained by DNS of an SVE).

We note that Step 1 and Step 2 involve deterministic training of neural networks, but the respective loss functions are different. Step 1 results from minimizing the conventional mean squared error of the average response (standard training in regression tasks). Step 2 results from training another RNN to learn the standard deviation of the aleatoric uncertainty after the mean response has been determined in Step 1. However, the loss function in Step 2 is different from Step 1, as we derived it from assuming a Gamma distribution of the square of the residual, which can be demonstrated to lead to the variance of the aleatoric uncertainty [20]. The sequential training of Step 1 and Step 2 was shown to stabilize training of the variance network (Step 2), addressing an important knowledge gap in the literature [20].

---

[8] From Section 2, $\boldsymbol{\sigma}$ has three components for two-dimensional problems under plane strain conditions: σ_{11}, σ_{12}, and σ_{22}.

[9] If the data is history-independent, each network of the 3 steps does not need recurrent units. So, the mean estimation would be with a feedforward neural network (FNN), variance estimation also with an FNN, and final prediction with epistemic uncertainty with a BNN.
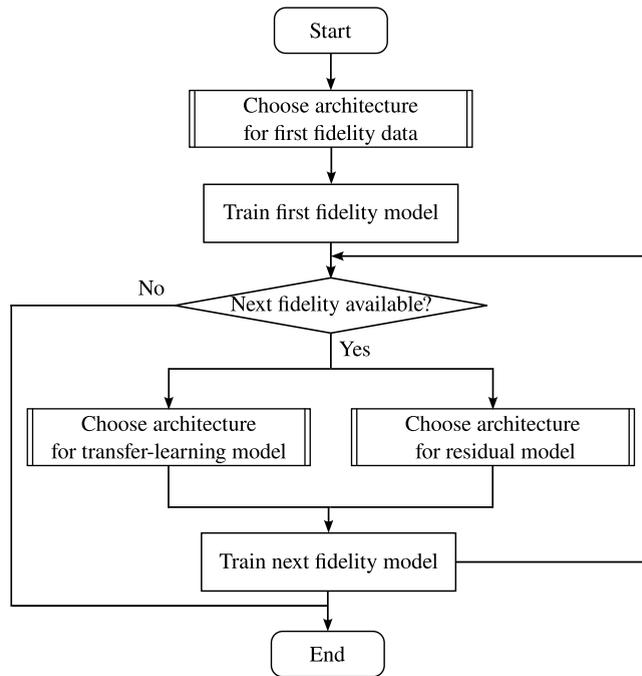
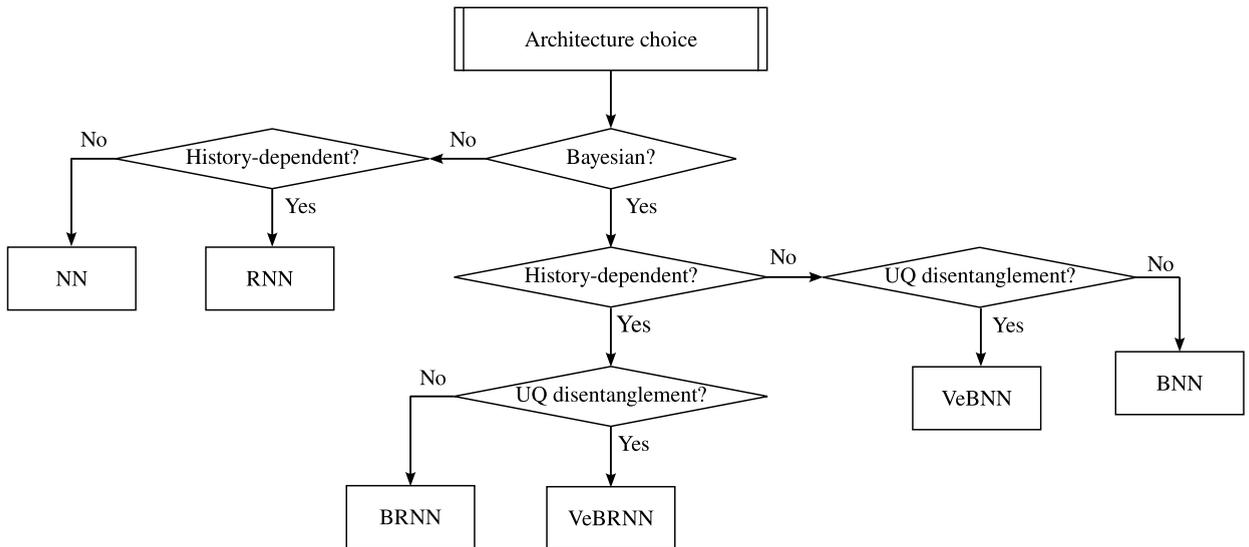**Fig. 4.** Flowchart of the proposed practical multi-fidelity framework.

Fig. 5. Flowchart for choosing a neural network architecture that learns from data of given fidelity. UQ is an abbreviation for uncertainty quantification.

In addition, Step 3 concerns "training" a BRNN, which is not done by minimizing a loss function but by sampling the posterior distribution – a process designated by Bayesian inference (Appendix A). This step allows to update the mean response prediction (from Step 1, and for the aleatoric uncertainty predicted in Step 2). More importantly, it also allows to estimate the uncertainty associated to the mean prediction, i.e., it determines the epistemic or model uncertainty. There are many strategies to perform Bayesian inference, and we thoroughly investigated them for different regression problems in [20]. We demonstrated that pSGLD is robust and scalable, and recommend its choice for regression problems involving smooth and unimodal probability distribution functions, such as Gaussian and Gamma distributions.

Algorithm 1: Cooperative training of VeBRNN

Data: Mean RNN network $f(\mathbf{x}; \theta)$, variance RNN network $s^2(\mathbf{x}; \phi)$, training data $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$ number of iterations K

Result: Optimal parameters for variance and Bayesian mean RNN networks

Step 1: Mean network training:

    Deterministic training: find mean estimation by minimizing the MSE loss Eq. D.5 using *Adam* optimizer.

for $i = 1$ **to** K **do**

    Step 2: Variance network training (Aleatoric uncertainty):

        Deterministic training: find data variance estimation by minimizing Eq. D.6 for fixed mean from Step 1 or Step 3 using *Adam* optimizer.

    Step 3: Bayesian network training (Epistemic uncertainty):

        Bayesian inference: obtain posterior distribution estimation from Eq. D.10 to update mean and epistemic variance estimates for fixed aleatoric variance from Step 2 using *pSGLD* sampler illustrated in Appendix A.1.2.

        Compute the log marginal likelihood of `LMglk[i]`

end

Identify the optimal models by $i^* = \arg\max_i$ `LMglk[i]`.

3.2. Multi-fidelity framework

In a recent article, we proposed a unification of the MF machine learning literature into a practical MF framework [19]. We postulated that MF machine learning can be expressed through the general formulation,

$$f^h(\mathbf{x}) = g(f^l(\mathbf{x}), \mathbf{x}) + r(\mathbf{x}), \tag{3}$$

where $f^h(\mathbf{x})$ is the HF model, $f^l(\mathbf{x})$ is the LF model, $g(\mathbf{x})$ is a transfer learning model, and $r(\mathbf{x})$ is a residual model. We demonstrate the capability of handling problems of different dimensionality and dataset sizes by selecting suitable machine learning models. In the original work, we adopted a deterministic feedforward neural network for the LF model, a linear transfer learning model (linear regression), and a Bayesian neural network for the residual model. This was demonstrated to be effective in finding the homogenized properties of a biphasic microstructure material (without history-dependency).

To the best of our knowledge, RNNs have not yet been considered in a MF setting. Therefore, in this work, we investigate how to incorporate architectures with recurrent units in the above-mentioned MF framework. As we have to consider inputs that arrive as sequences (contain history), the MF sequence dataset $\mathcal{D}$ consist of HF inputs $\mathbf{X}_t^h \in \mathbb{R}^{N^h \times d^{in}}$ and outputs $\mathbf{Y}_t^h \in \mathbb{R}^{N^h \times d^{out}}$, as well as LF inputs $\mathbf{X}_t^l \in \mathbb{R}^{N^l \times d^{in}}$ and outputs $\mathbf{Y}_t^l \in \mathbb{R}^{N^l \times d^{out}}$. Here, N^h and N^l are the number of HF and LF sequences, respectively, while d^{in} and d^{out} represent input and output dimensions, and t indicates time (or pseudo-time) increment in the sequence. Accordingly, we rewrite Eq. (3) to explicitly account for time-series data as

$$f^h(\mathbf{x}_t^h) = g(f^l(\mathbf{x}_t^h), \mathbf{x}_t^h) + r(\mathbf{x}_t^h); \quad t = 1, \dots, T, \tag{4}$$

where $f^h(\mathbf{x})$, $f^l(\mathbf{x})$, $g(\mathbf{x})$, and $r(\mathbf{x})$ are models for the HF, LF, transfer learning, and residual, respectively, including recurrent units.[10]

Eq. (4) looks deceptively simple, but it accepts many architectural choices to relate LF and HF data. For example, the literature often considers a linear transfer learning model g (effectively not doing transfer learning), or uses a neural network as transfer learning without considering an additional residual model r. We carefully investigated different possibilities and included in Appendix E four solutions for comparison. For brevity, the main text only includes the best solution – illustrated in Fig. 6.

The MF model in Fig. 6 can be regarded as a particular realization of Fig. 4. The LF model is trained first (independently) using an RNN for the LF data, then the linear transfer learning model $g(\mathbf{x})$ is trained, and finally another RNN is trained to learn the residual $r(\mathbf{x})$ using HF data. Therefore, there are three models that need to be defined in a MF prediction: $f^l(\mathbf{x})$, $g(\mathbf{x})$, and $r(\mathbf{x})$. These models can be deterministic or Bayesian, depending on whether uncertainty quantification is desired or not. Also note that Fig. 6 should be interpreted along two dimensions: one corresponding to the time increment (from left to right), and the other to the fidelity level (from bottom to top). This structure enables the model to capture both the temporal evolution of material responses and transfer useful information across different fidelity levels, facilitating accurate predictions.

Interestingly, we found that transferring the hidden states from the LF model to the HF model was more effective than directly transferring the LF outputs (i.e., the stresses that are decoded at the last layer of the LF RNN, denoted as RNN^l in Fig. 6). This is reflected in Fig. 6 by noticing that the input to the residual model, which is also an RNN (denoted as RNN^r), is the hidden state of the LF model $h^l(\mathbf{x}_t^h)$ together with the input HF data $\mathbf{x}^h$, instead of using the decoded output $\mathbf{y}^l$ of the RNN^l. While this may seem like a subtle difference, we found it to be significant – see Appendix E. Moreover, we also found that using a linear transfer learning model (i.e., $g(\mathbf{x})$ is simply an identity map in Fig. 6) together with the referred RNN for the residual model (or a Bayesian version like a VeBRNN) is marginally better than using an RNN transfer learning model and removing the residual model – see Appendix E.2.1 and E.2.2. We did not find any benefit in using expressive models for both transfer learning and residual learning. Nevertheless, we

---

[10] We consider two fidelity levels (HF and LF) for the convenience of illustration. The framework can naturally be extended to accommodate problems involving more than two fidelity levels. This is shown in Fig. 4 by recurrently adding new fidelities.
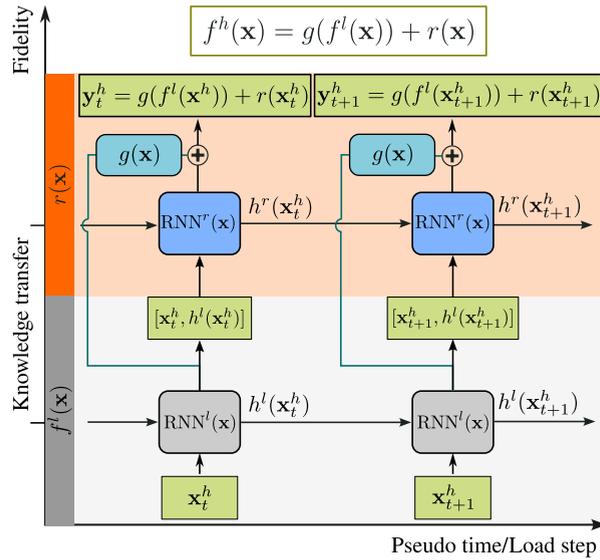
Fig. 6. Schematic of the proposed multi-fidelity recurrent neural network architectures for predicting uncertainty-aware history-dependent plasticity.

Table 2
Summary of architectural choices for single- and multi-fidelity datasets. See Fig. 4 for understanding single- and multi-fidelity training with neural network architectures chosen according to Fig. 5. Notation: "LF model" + "HF model".

Model	Comment	LF predictions			HF predictions		
		Mean	Aleatoric	Epistemic	Mean	Aleatoric	Epistemic
RNN	Deterministic (single-fidelity)				✓		
VeBRNN	Bayesian (single-fidelity)				✓	✓	✓
RNN + RNN	Deterministic LF & HF	✓			✓		
VeBRNN + RNN	Bayesian LF, deterministic HF	✓	✓	✓	✓		
RNN + VeBRNN	Deterministic LF, Bayesian HF	✓			✓	✓	✓
VeBRNN + VeBRNN	Bayesian LF & HF	✓	✓	✓	✓	✓	✓

do not claim that Fig. 6 represents the best MF model for all possible datasets. Appendix E.2.1 and E.2.2 simply shows that it is the most effective solution for the four datasets that we consider in this article, and that we discuss in the next sections.

Remark 3. For simplicity, this subsection and Fig. 6 refer to RNNs. However, as shown in the next sections, any other architecture with recurrent units can be used. Namely, the VeBRNN architecture that we described previously (see Fig. 4 and Algorithm 1). The VeBRNN architecture,

- when applied only to the LF model (replacing RNN^l by $VeBRNN^l$), enables uncertainty quantification and disentanglement for LF predictions.
- when applied only to the residual learning model (replacing RNN^r by $VeBRNN^r$), enables uncertainty quantification and disentanglement for HF predictions.
- when applied to both models, enables uncertainty quantification and disentanglement for both fidelities.

3.3. MF framework with different neural network architectures

We expect that a fully Bayesian MF model using VeBRNNs is applicable to a wide range of scenarios, but we also ablate that model to consider progressively simpler architectural choices. We believe that the four above-mentioned datasets (Section 2) illustrate different scenarios encountered in Engineering practice. Table 2 summarizes the different models, from simpler (top) to more complex (bottom) – confront with Figs. 4 and 5.

4. Results

We analyze the proposed framework (Fig. 6) on four datasets (Section 2), and compare the performance of different model architectures (Table 2) while considering the total computational cost of data generation as,

$$T_c = N_{\text{train}}^h c^h + N_{\text{train}}^l c^l, \tag{5}$$

Table 3

Performance metrics considered in this work. The last column indicates the desired trend of the metric (e.g., relative error should decrease, PICP should tend to $1 - \alpha = 95\,\%$, and test log-likelihood should increase).

Metric	Abbreviation	Description	Trend
Relative error ([55])	ϵ_r	Relative error between mean prediction and ground truth.	↓
Test Log-Likelihood ([56])	TLL	Test log-likelihood of ground truth mean under the predicted epistemic distribution.	↑
Wasserstein Distance ([57])	WA	Wasserstein distance between predicted aleatoric and ground truth distributions.	↓
Prediction Interval Coverage Probability ([58])	PICP	Ratio of ground truth mean within the predicted epistemic interval under given confidence level.	$\rightarrow (1 - \alpha)$
Mean Prediction Interval Width ([58])	MPIW	Mean width of predicted epistemic uncertainty interval.	↓

Table 4

Experimental setup for dataset 1.

Data fidelities	Model	Evaluation metrics	Results
$\widetilde{\text{DNS}}$	RNN	ϵ_r	Fig. 7
$\widetilde{\text{DNS}}$	VeBRNN	ϵ_r, TLL, WA, PICP, MPIW	Fig. 7

where N_{train}^h and N_{train}^l are the number of training paths for HF and LF, respectively; c^h and c^l are the corresponding cost ratios of HF and LF, which can be found in Table 1.

Recall that performance evaluation for Bayesian models involves more than one metric, as opposed to evaluating deterministic models. We use five accuracy metrics summarized in 3. A more detailed description of these metrics is provided in Appendix B. Note that the ground truth mean in the proposed dataset is obtained from $\widetilde{\text{DNS}}$ (noiseless FEA simulations of an RVE subjected to different strain paths), which is the highest fidelity data that we have available. The confidence intervals are all defined for $\alpha = 0.05$, i.e. we use 95 % confidence intervals for all experiments. Hyperparameters used for every model are listed in Appendix G. Additional results are presented in Appendix F, and we will refer to this appendix when appropriate.

4.1. Results for dataset 1: $\widetilde{DNS}$ (stochastic single fidelity problem from FEA of an SVE)

This first dataset includes aleatoric uncertainty and only considers one fidelity ($\widetilde{\text{DNS}}$). It is introduced to demonstrate the ability of the VeBRNN to predict mean, aleatoric uncertainty variance, and epistemic uncertainty variance. This contrasts with the widely used RNNs that only predict the mean response, as they are deterministic models. A summary of the setup of experiments is given by Table 4.

As indicated in Table 1, this dataset has up to 2000 training paths (all of which are considered HF, as there is no LF data). In the results below, we consider experiments with an increasing number of training paths from 100 to 2000. Accordingly, T_c increases from 5 to 100, following the relation $T_c = N_{\text{train}}^h \times \frac{1}{20} + 0$. The corresponding results are shown in Fig. 7. Note that we also have access to the noiseless data ($\widetilde{\text{DNS}}$) that we can use to precisely assess the error of this dataset for the relative error ϵ_r and Epistemic TLL.

Remark 4. The single-fidelity case trivializes the flowchart in Fig. 4, reducing to training a single RNN (or VeBRNN).

As shown in Fig. 7, the relative error ϵ_r obtained on ID testing paths decreases as the number of training paths increases for both the deterministic model and the proposed VeBRNN method, as expected. The VeBRNN has comparable ϵ_r to the RNN, achieving marginally better mean predictions when $T_c > 20$. The remaining metrics are only relevant to the VeBRNN, as they quantify the quality of uncertainty estimates and their separation (the RNN is deterministic). The Epistemic TLL (Fig. 7(c)) improves with more training paths, indicating that the VeBRNN reduces model uncertainty with more training data (as expected). The Wasserstein distance (Fig. 7(b)) to the aleatoric uncertainty distribution also reduces, dropping sharply from $T_c = 5$ to $T_c = 40$ and then plateauing, indicating that VeBRNN rapidly and effectively captures data noise (aleatoric uncertainty) and separates it from the epistemic uncertainty. Based on the well-captured aleatoric uncertainty, we see that the PICP achieves the desired test coverage of 0.95 for the ID testing data – this indicates that 95 % of testing points are within the estimated confidence interval by the VeBRNN. Evidently, in the extrapolation regime, i.e., for OOD testing data, the PICP is worse. Equivalently, the tightening of the epistemic uncertainty confidence interval with increasing data is clear from Fig. 7(e) (decreasing MPIW value) – this indicates that the model becomes more confident about its prediction with increasing training data. For the OOD test paths, the performance generally worsens for all performance metrics, as expected. Nevertheless, we observe that MPIW is larger than that for ID test paths, reflecting that the VeBRNN predicts larger epistemic uncertainty for the extrapolation regime (as it should). Interestingly, the ϵ_r of deterministic training is slightly smaller than

(a) Mean estimate metric (↓)   (b) Aleatoric uncertainty metric (↓)

(c) Epistemic uncertainty metric (↑)   (d) Epistemic uncert. metric (→ 0.95)   (e) Epistemic uncertainty metric (↓)
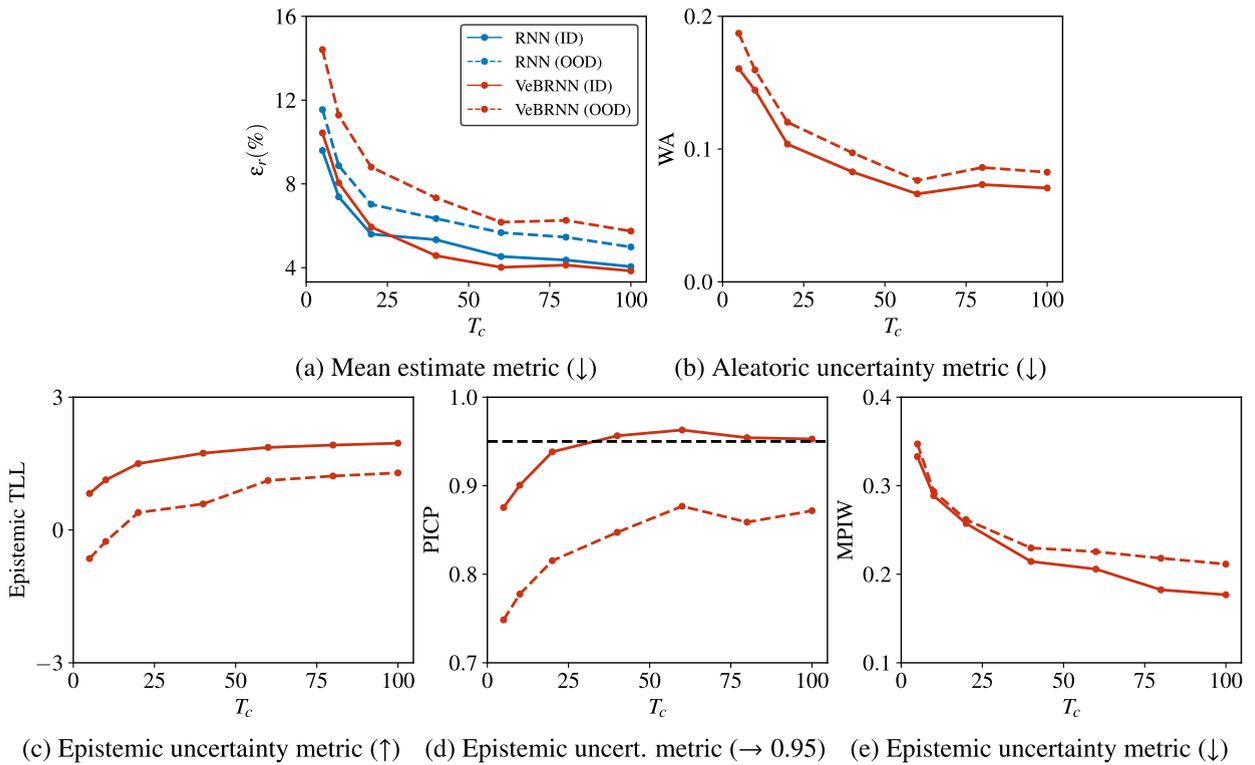
Fig. 7. Comparison of test performance metrics on Dataset 1 (single-fidelity problem) based on $\widetilde{\text{DNS}}$ training paths. In this figure, $T_c = 1$ corresponds to 20 training paths, and the horizontal dashed line in (d) indicates the desired confidence level 0.95. Testing data is considered both in-distribution (ID), and out-of-distribution (OOD). (a) Mean estimate metric ε_r (↓); (b) Aleatoric uncertainty metric WA (↓); (c) Epistemic uncertainty metric TLL (↑); (d) Epistemic uncertainty metric PICP (→ 0.95); (e) Epistemic uncertainty metric MPIW (↓).

for the VeBRNN for OOD test paths. Bayesian models have a tendency to revert to the non-trivial prior in the OOD region, as there is no information to create a good quality prediction when there is no data.

Fig. 8 exemplifies the predictions of VeBRNN for a randomly selected ID testing path (for all 3 components), and considering different training dataset sizes. The quality of uncertainty quantification and disentanglement achieved by the cooperative training of VeBRNNs is evident. The last row of the figure, corresponding to results obtained for all 2000 training paths, shows that the predicted aleatoric distribution (shaded areas) closely overlaps with the ground truth distribution (black dotted lines) generated through 100 repeated simulations based on $\widetilde{\text{DNS}}$. Note the nontrivial nature of these predictions, since the model captures the larger variation of σ_{11} and σ_{22} compared to σ_{12}. Fig. 8(f) presents the predicted epistemic uncertainty: the band is narrow and yet it covers all three stress components (recall that this path is unseen). The epistemic uncertainty is smaller where the input strain is smaller and increases when the input strain magnitude grows, as expected. For comparison, the prediction from deterministic RNNs is given in Fig. F.7, where it can be seen that σ_{12} exhibits significant overfitting, emphasizing the advantages of Bayesian modeling in reliable uncertainty quantification and avoiding overconfident predictions.

From top to bottom, Fig. 8 indicates the influence of different training set sizes on the model's predictive performance. The results reinforce that fewer training points decrease performance for all accuracy metrics, as also shown in Fig. 7. Concerning the uncertainty estimates, few training points lead to less confident predictions for both uncertainties, with wider shaded areas in both aleatoric and epistemic uncertainties. It is worth mentioning that the aleatoric uncertainty predictions for all stress components remain stable from $T_c = 40$ (Fig. 8c to $T_c = 100$ (Fig. 8e) while the predicted epistemic uncertainties (Fig. 8(d) and (f)) decrease, as expected.

4.2. Results for dataset 2: $\overline{\text{ROM}} + \overline{\text{DNS}}$ (MF problem with deterministic LF & HF data)

This MF dataset does not have aleatoric uncertainty (noise); therefore, we can consider RNNs for both fidelities. However, we can also consider VeBRNNs (which reduce to BRNNs because they find zero aleatoric uncertainty), in case we want to predict epistemic uncertainty. We include both cases in this section. Unless otherwise stated, all the available LF training paths indicated in Table 1 are used, and models are assessed for an increasing number of HF training paths. Detailed experiment setup is summarized in Table 5.

4.2.1. MF model with RNN + RNN architectures

A MF model with fully deterministic architectures is trained using *Adam* [59]. Fig. 9 presents the results, including a comparison with a single-fidelity RNN model trained only on HF data ($\overline{\text{DNS}}$). Up to a certain computational cost, the MF model (RNN + RNN)
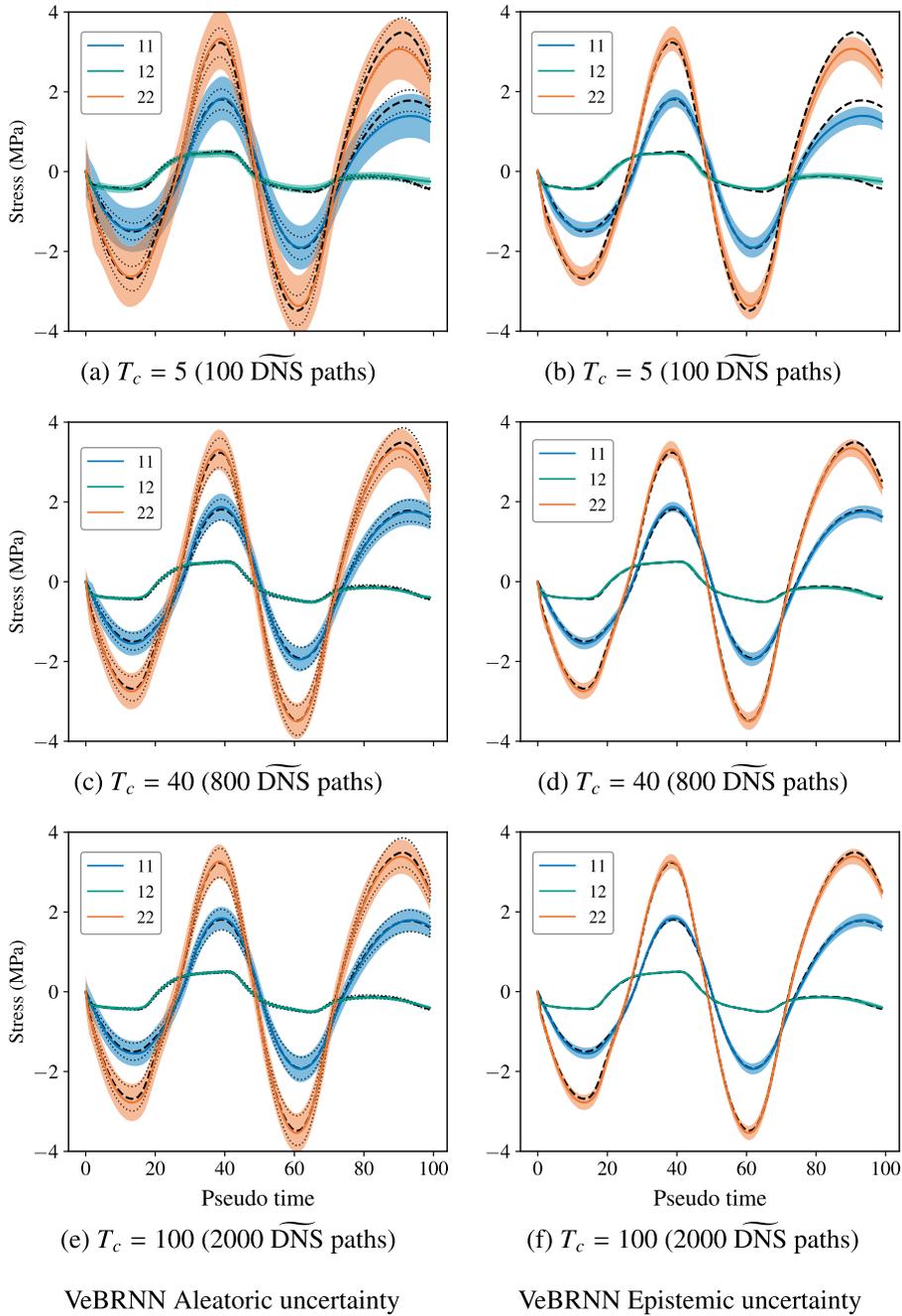
(a) $T_c = 5$ (100 $\widetilde{\text{DNS}}$ paths)

(b) $T_c = 5$ (100 $\widetilde{\text{DNS}}$ paths)

(c) $T_c = 40$ (800 $\widetilde{\text{DNS}}$ paths)

(d) $T_c = 40$ (800 $\widetilde{\text{DNS}}$ paths)

(e) $T_c = 100$ (2000 $\widetilde{\text{DNS}}$ paths)

(f) $T_c = 100$ (2000 $\widetilde{\text{DNS}}$ paths)

VeBRNN Aleatoric uncertainty   VeBRNN Epistemic uncertainty

Fig. 8. Predictions of VeBRNN on an ID test path after training with 2000 paths ($T_c = 100$) from Dataset 1 ($\widetilde{\text{DNS}}$). The left column shows the comparison between 0.95 confidence intervals of the predicted aleatoric uncertainty (colored shaded areas) and the ground truth distributions (black dotted shaded areas); the right column gives the predicted epistemic uncertainty distributions. The ground truth mean is shown as black dashed lines in both columns. (a) $T_c = 5$ (100 $\widetilde{\text{DNS}}$ paths); (b) $T_c = 5$ (100 $\widetilde{\text{DNS}}$ paths); (c) $T_c = 40$ (800 $\widetilde{\text{DNS}}$ paths); (d) $T_c = 40$ (800 $\widetilde{\text{DNS}}$ paths); (e) $T_c = 100$ (2000 $\widetilde{\text{DNS}}$ paths) VeBRNN Aleatoric uncertainty; (f) $T_c = 100$ (2000 $\widetilde{\text{DNS}}$ paths) VeBRNN Epistemic uncertainty.

Table 5
Experimental setup for dataset 2.

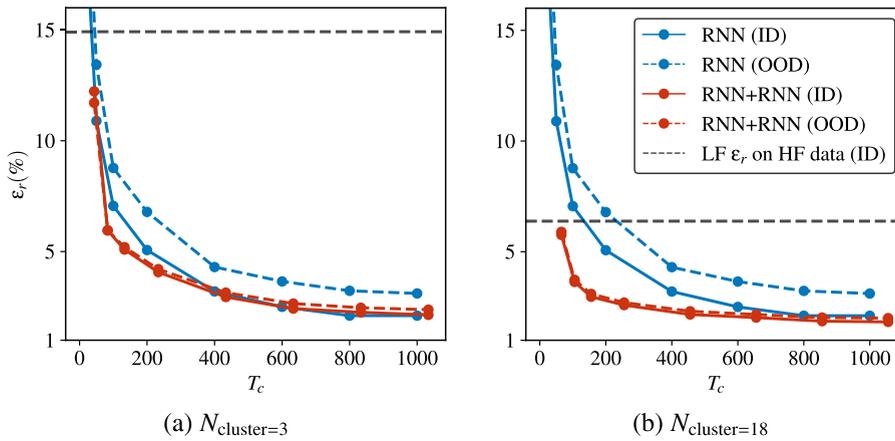| Data fidelities	Model	Evaluation metrics	Results
$\overline{\text{DNS}}$	RNN	ϵ_r	Fig. 9
$\overline{\text{ROM}} + \overline{\text{DNS}}$	RNN + RNN	ϵ_r	Fig. 9
$\overline{\text{DNS}}$	VeBRNN	ϵ_r, TLL, WA, PICP, MPIW	Fig. 10
$\overline{\text{ROM}} + \overline{\text{DNS}}$	RNN + VeBRNN	ϵ_r, TLL, WA, PICP, MPIW	Fig. 10

Fig. 9. Results obtained for the fully deterministic MF model (RNN + RNN) trained on Dataset 2 ($\overline{\text{ROM}} + \overline{\text{DNS}}$) compared with a single-fidelity RNN trained on data obtained from $\overline{\text{DNS}}$. (a) results of LF data with $N_{\text{cluster}} = 3$; (b) results of LF data with $N_{\text{cluster}} = 18$. For the single-fidelity RNN, T_c corresponds to the following number of training paths: [10, 50, 100, 200, 400, 600, 800, 1000]. For RNN + RNN model, T_c has offsets of 33.3 and 55.5 for $N_{\text{cluster}} = 3$ and $N_{\text{cluster}} = 18$, respectively, because we use all the LF data and only change the HF data size. The dark gray dashed lines on both figures indicate the relative error ϵ_r of LF models on the HF data.

is more accurate than the single-fidelity model, especially for OOD test paths. The MF model advantage is more pronounced when the LF model is better (using a higher number of clusters for the SCA method – Fig. 9b). For a limit case of only 10 HF training paths, the RNN + RNN model achieves significantly lower ϵ_r than the single RNN (note that the relative error ϵ_r is even smaller than the one obtained by the LF model – shown by the horizontal dark gray bar). These findings confirm that the RNN + RNN model is particularly effective in data-scarce HF scenarios – encouraging for future real-world Engineering applications. Importantly, the RNN + RNN model significantly decreases the ϵ_r gap between ID and OOD test paths – a key benefit of leveraging computationally inexpensive LF data to explore a larger domain and enhancing HF prediction accuracy and generalization.

4.2.2. MF model with RNN + VeBRNN architectures

Considering the VeBRNN (which will end up reducing to a BRNN) for the HF data allows to quantify epistemic uncertainty (and to estimate zero aleatoric uncertainty). Results are shown in Fig. 10. Note that now we compare with a single-fidelity model with the VeBRNN, instead of the RNN, so that we can assess if there is an advantage in predicting epistemic uncertainty by using MF data. The relative error evolution with increasing training dataset sizes is consistent with the previous observations for the other models, with a clear advantage for the MF model compared to the SF one. As in Section 4.1, the PICP and MPIW reveal that the RNN + VeBRNN model estimates a wide epistemic confidence interval when using only 10 HF training paths, corresponding to a high PICP. As the number of HF training paths increases, the MPIW drops rapidly, initially accompanied by a decrease in PICP, followed by a steady recovery to the target confidence level of 95 %. Regarding the Wasserstein distance, it is calculated with a ground truth of zero variance (as there is no noise), and it can be seen to reduce to a small value rapidly, demonstrating that the RNN + VeBRNN finds that the dataset has negligible noise.[11]

Fig. 11 presents the RNN + VeBRNN prediction for an ID test path for a model trained with a total computational cost of $T_c = 1000 + 2000 \times \frac{1}{36} = 1055.5$. The LF model was obtained by SCA of an RVE using $N_{\text{cluster}} = 18$. Since both fidelity levels are obtained from RVE simulations, the dataset is noiseless. As a result, the predicted aleatoric uncertainty of the RNN + VeBRNN is near zero in Fig. 11(a). Moreover, the predicted epistemic uncertainty is very tight but still covers the ground truth, as it should. The results demonstrate the advantage of considering a Bayesian model, even for noiseless datasets. They also show that MF models can effectively learn from datasets gathered using fewer computational resources.

4.3. Results for dataset 3: $\widetilde{\text{ROM}} + \widetilde{\text{DNS}}$ (MF problem with stochastic LF & HF data)

This is the most challenging dataset, as both fidelities are stochastic (noisy). Therefore, we have selected to use five different MF modeling strategies from Table 2, and the details are listed in Table 6.

Fig. 12 presents the VeBRNN + VeBRNN model when trained with the LF model using $N_{\text{cluster}} = 18$. We also share the results for an RNN + RNN model (fully deterministic) in Appendix F.2, where the LF model uses $N_{\text{cluster}} = 3$ for the SCA method. Furthermore, we also considered a MF model using RNN + VeBRNN, but the predictions are similar to the VeBRNN + VeBRNN model (see Appendix F.2 for the RNN + VeBRNN model results).

---

[11] If the analyst already knows that the dataset is noiseless for both fidelities, then the VeBRNN can simply be replaced by a BRNN (as there is no need to find the noise). We wanted to demonstrate that the method is general, and can be used even when one is not certain about the presence or absence of aleatoric uncertainty.
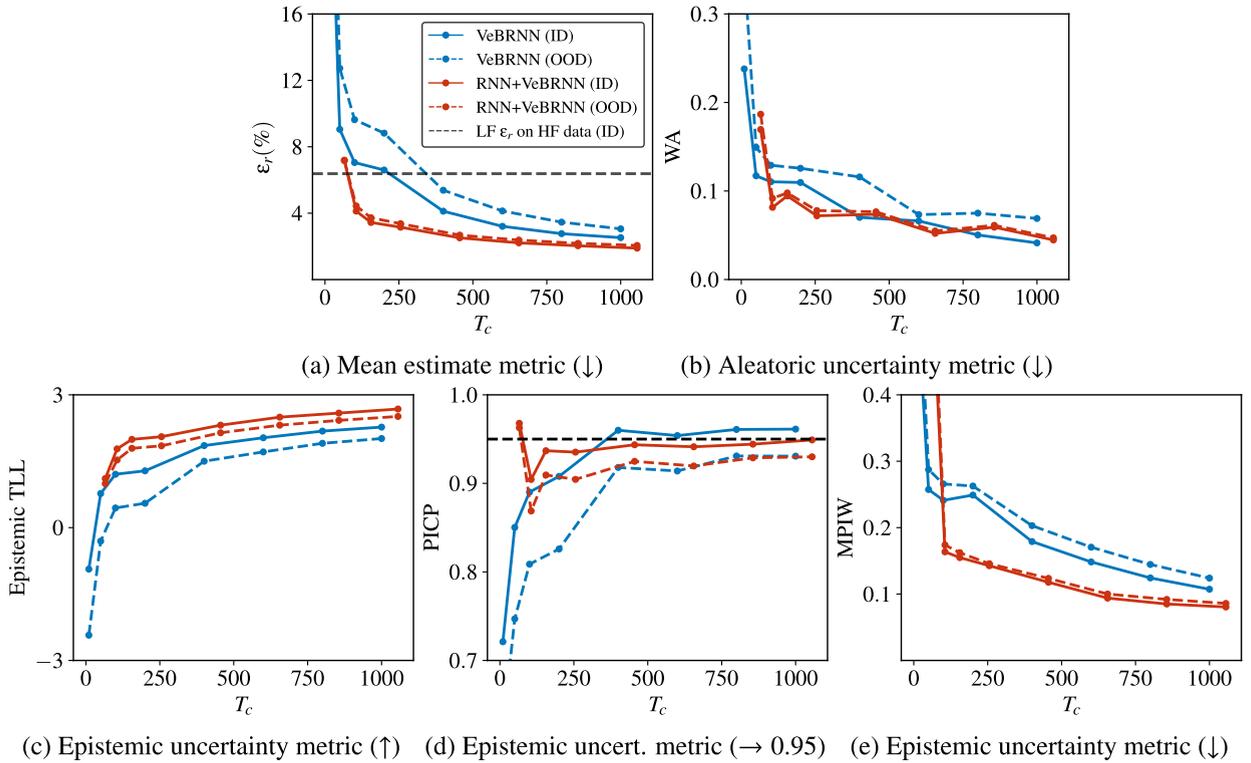
(a) Mean estimate metric (↓)   (b) Aleatoric uncertainty metric (↓)

(c) Epistemic uncertainty metric (↑)   (d) Epistemic uncert. metric (→ 0.95)   (e) Epistemic uncertainty metric (↓)

Fig. 10. Results of RNN + VeBRNN model trained on Dataset 2 ($\overline{\text{ROM}} + \overline{\text{DNS}}$), compared to single fidelity VeBRNN trained only on HF data ($\overline{\text{DNS}}$). The LF data was obtained by SCA of an RVE using $N_{\text{cluster}} = 18$. The horizontal dashed lines in (d) indicate the desired 95 % confidence interval. (a) Mean estimate metric ε_r (lower is better, as indicated by ↓); (b) Aleatoric uncertainty metric WA (↓); (c) Epistemic uncertainty metric TLL (↑); (d) Epistemic uncertainty metric PICP (→ 0.95); (e) Epistemic uncertainty metric MPIW (↓).

(a) RNN+VeBRNN Aleatoric uncertainty   (b) RNN+VeBRNN Epistemic uncertainty
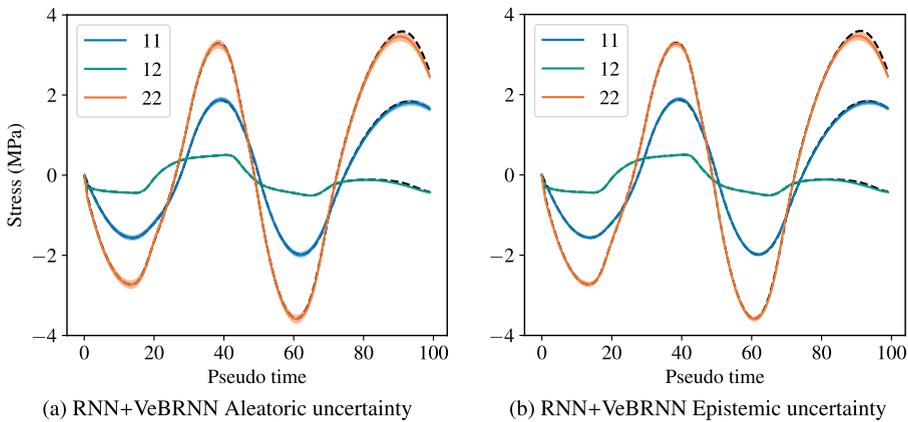
Fig. 11. Prediction of RNN + VeBRNN for a training dataset ($\overline{\text{ROM}} + \overline{\text{DNS}}$) with $T_c = 1000 + 2000 \times \frac{1}{36} = 1055.5$, where the LF model was obtained by SCA of an RVE using $N_{\text{cluster}} = 18$. (a) predicted aleatoric distributions (ground-truth is Dirac delta, i.e., zero aleatoric uncertainty), (b) predicted epistemic distributions.

Table 6
Experimental setup for dataset 3.

Data fidelities	Model	Evaluation metrics	Results
$\overline{\text{DNS}}$	VeBRNN	ϵ_r, TLL, WA, PICP, MPIW	Fig. 12
$\overline{\text{ROM}} + \overline{\text{DNS}}$	VeBRNN + VeBRNN	ϵ_r, TLL, WA, PICP, MPIW	Fig. 12
$\overline{\text{DNS}}$	RNN	ϵ_r	Fig. F.8
$\overline{\text{ROM}} + \overline{\text{DNS}}$	RNN + RNN	ϵ_r	Fig. F.8
$\overline{\text{ROM}} + \overline{\text{DNS}}$	RNN + VeBRNN	ϵ_r, TLL, WA, PICP, MPIW	Fig. F.9

(a) Mean estimate metric (↓) (b) Aleatoric uncertainty metric (↓)

(c) Epistemic uncertainty metric (↑) (d) Epistemic uncert. metric (→ 0.95) (e) Epistemic uncertainty metric (↓)
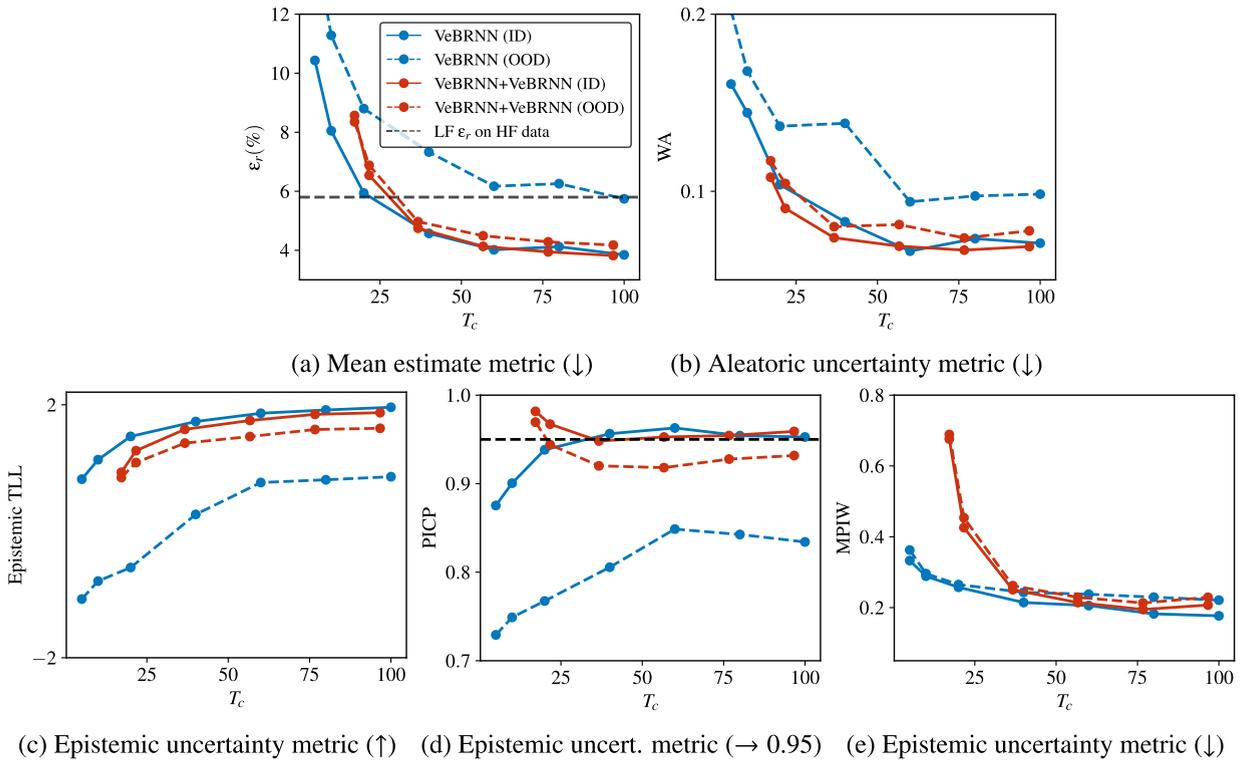
Fig. 12. Results for the VeBRNN + VeBRNN model, when compared with a single-fidelity VeBRNN trained on HF data. The LF data is obtained by SCA of an SVE with $N_{\text{cluster}} = 18$. The number of HF training paths $(\widetilde{\text{DNS}})$ used in MF training is $[10, 100, 400, 800, 1200, 1600]$, and the number of LF paths is fixed at 2000. This corresponds to an initial total computational cost of $T_c = \frac{10}{20} + \frac{2000}{120} = 17.16$ for the MF dataset. (a) Mean estimate metric ε_r (↓); (b) Aleatoric uncertainty metric WA (↓); (c) Epistemic uncertainty metric TLL (↑); (d) Epistemic uncertainty metric PICP (→ 0.95); (e) Epistemic uncertainty metric MPIW (↓).

Fig. 12 shows that the MF model has similar performance to the single-fidelity model trained on HF data when considering the same total computational cost for ID test paths. This result is not surprising when noticing that the cost ratio between HF and LF data acquisition is only 6:1, but the LF is significantly less accurate and both fidelities are noisy. However, for OOD test paths the MF model is significantly better than the single-fidelity model, as observed by comparing the dashed lines in Fig. 12.

Remark 5. In principle, it is not possible to guarantee *a priori* if a given MF dataset will lead to a MF model that performs better than the corresponding single-fidelity model for the same total budget. However, using active learning, it is possible to automate the data acquisition process, allowing to choose between fidelities on-the-fly. Bayesian models like VeBRNNs are a crucial development towards this because active learning becomes possible with accurate epistemic uncertainty estimation. Therefore, we find active learning and Bayesian optimization as an interesting future direction to pursue.

4.4. Results for dataset 4: $\widetilde{ROM} + \overline{DNS}$ (MF problem with stochastic LF & deterministic HF)

Fig. 13 presents the results obtained for Dataset 4 using a VeBRNN + RNN model (Bayesian LF model and deterministic HF model). The results include a comparison with the single-fidelity RNN trained only on HF data, as listed in Table 7. The results are similar to Fig. 9, where a clear advantage is observed when the LF model has a higher accuracy (obtained with the ROM with $N_{\text{cluster}} = 18$, instead of $N_{\text{cluster}} = 3$). Note that in this example the VeBRNN is used to train the LF model (because the LF data is noisy, and we want to determine the aleatoric uncertainty – see Table 2). With all the 2000 training paths used to train the LF model, the VeBRNN achieves $\varepsilon_r = 3.87\,\%$, Epistemic TLL = 1.78, WA = 0.074, PICP = 0.949, and MPIW = 0.214. The predictions for a test path are shown in Fig. 14, where excellent predictions for all mean, aleatoric, and epistemic uncertainties are observed. In this case, the estimation of aleatoric uncertainty also reflects the quality of LF data.

Fig. 15 shows the HF prediction from the MF model (VeBRNN + RNN). We note that the LF predictions that were transferred to the HF model, shown as colored dashed lines, already provide accurate estimates for σ_{11} and σ_{22}, but not for σ_{12}. The MF model effectively narrows this gap and predicts the HF accurately (colored solid lines). We find it interesting that a Bayesian model on the LF improves a deterministic model at the HF, although this might not be a common model construction (typically, the analyst intends to predict epistemic uncertainty at the HF and may not be interested in estimating noise at the LF). In addition, there is a potential

Table 7

Experimental setup for dataset 4.

Data fidelities	Model	Evaluation metrics	Results
$\overline{DNS}$	RNN	ϵ_r	Fig. 13
$\widetilde{ROM} + \overline{DNS}$	VeRNN + RNN	ϵ_r	Fig. 13

(a) $N_{\text{cluster}} = 3$ (b) $N_{\text{cluster}} = 18$

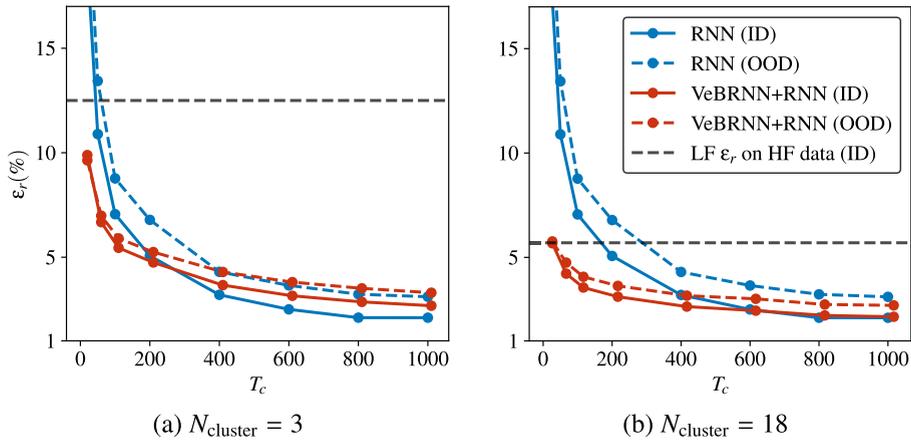**Fig. 13.** Results for VeBRNN + RNN model trained on Dataset 4 ($\widetilde{ROM} + \overline{DNS}$) compared to single-fidelity RNN trained on HF data. (a) LF model trained on data obtained by SCA of an SVE with $N_{\text{cluster}} = 3$; (b) LF model trained on data obtained by SCA of an SVE with $N_{\text{cluster}} = 18$.
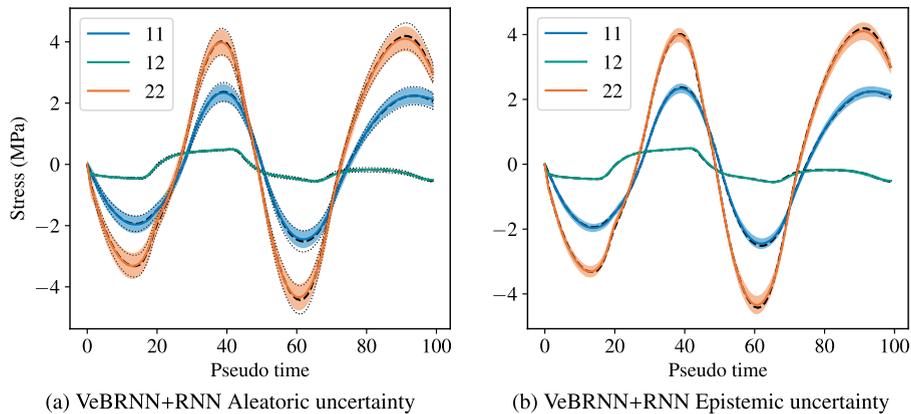

(a) VeBRNN+RNN Aleatoric uncertainty   (b) VeBRNN+RNN Epistemic uncertainty

Fig. 14. Predictions of the LF model (VeBRNN) for training data obtained by SCA of an RVE with $N_{\text{cluster}} = 18$ and considering all 2000 training paths. The ground truth is obtained by repeating the SVE simulation 100 times. (a) LF predicted aleatoric uncertainty; (b) LF predicted epistemic uncertainty.

risk of overfitting to the HF data, particularly in the prediction of σ_{12}. As discussed in earlier sections, considering a Bayesian model at the HF mitigates this risk.

5. Discussion

There are three key characteristics that control whether MF models are more accurate than their single-fidelity counterparts: (1) the cost ratio between acquiring HF and LF data; (2) how much error (bias) is introduced by the LF data; and (3) how much aleatoric uncertainty (noise) is present in the LF or HF data. This can be demonstrated by considering Datasets 3 and 4, evaluating performance for different computational budgets, and using different HF and LF data ratios.

Starting with Dataset 3 ($\widetilde{ROM} + \overline{DNS}$), recall that the MF model's performance was not significantly different from a model trained only on HF data for a given total cost when using 2000 training paths for the LF data and testing with ID paths (Section 4.3). However, Fig. 16 shows that if we change the percentage of LF data used in training while keeping the total cost at $T_c = 15$, then the MF model outperforms both the HF and LF models for most cases (blue markers compared to red and orange markers, respectively). However, these improvements are modest because (1) the accuracy of the HF and LF models is similar, and (2) the cost ratio between the
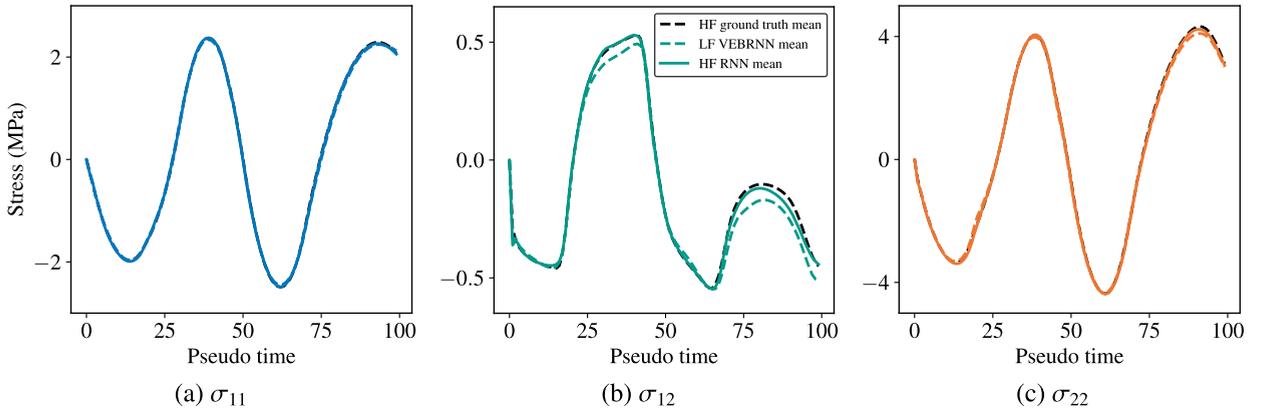
Fig. 15. Prediction of VeBRNN + RNN model for Dataset 4 ($\widetilde{\text{ROM}} + \overline{\text{DNS}}$) on a random HF test path using 2000 LF training paths (obtained by SCA of an SVE with $N_{\text{cluster}} = 18$) and using 1000 HF training paths (obtained by FEA of an RVE). The black dashed line is the HF ground truth (from FEA of an RVE), the colored dashed line represents the LF mean prediction on the HF data, and the solid colored line is the final mean prediction by the MF model. (a) σ_{11}; (b) σ_{12}; (c) σ_{22}.
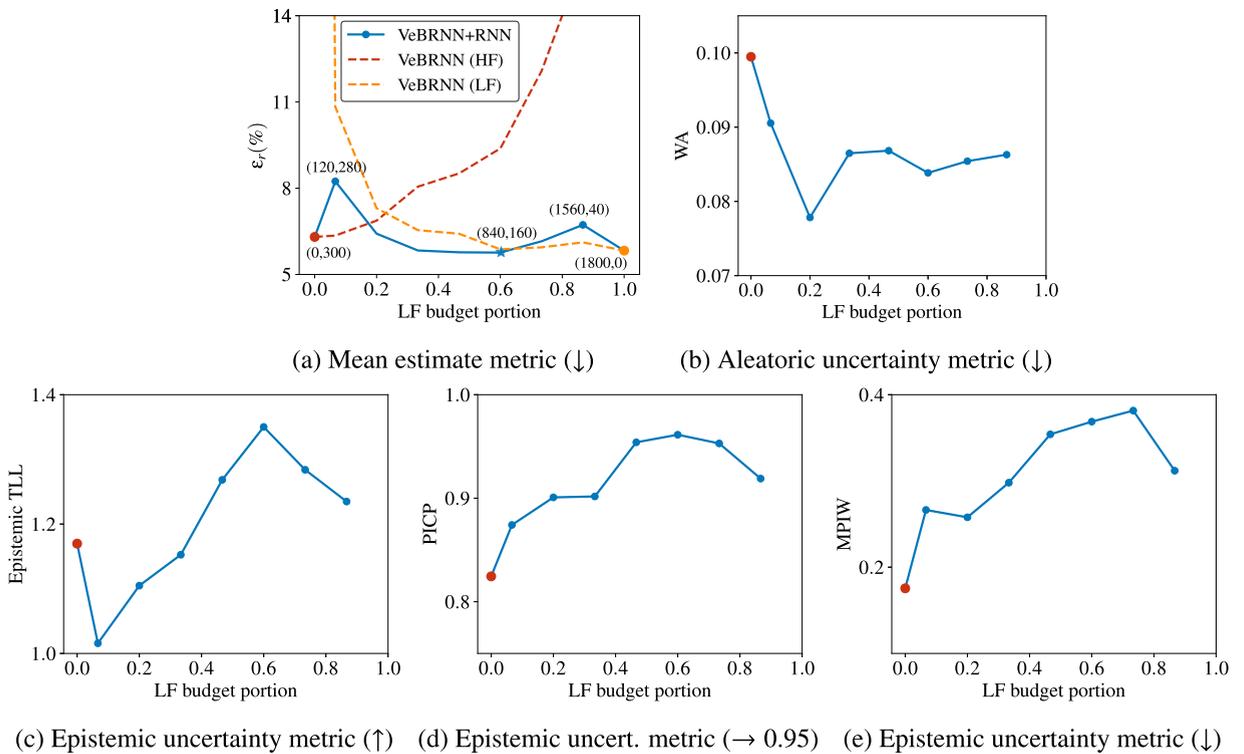

Fig. 16. Results for Dataset 3 ($\widetilde{\text{ROM}} + \overline{\text{DNS}}$ with $N_{\text{cluster}} = 18$) when considering different budget percentages of LF data acquisition for a total budget cost of $T_c = 15$. The performance metrics of the VeBRNN + VeBRNN model are compared with the two single-fidelity models obtained when allocating the entire budget to HF data (red marker) or LF data (orange marker). The red and orange dashed lines in (a) represent the relative errors of using only the HF or LF data from the MF model train setup counterpart shown in the parentheses. Note that we do not include the orange marker for metrics characterizing uncertainty (b–e), as there is no HF prediction in that case. (a) Mean estimate metric ε_r ($\downarrow$); (b) Aleatoric uncertainty metric WA ($\downarrow$); (c) Epistemic uncertainty metric TLL ($\uparrow$); (d) Epistemic uncertainty metric PICP ($\rightarrow 0.95$); (e) Epistemic uncertainty metric MPIW ($\downarrow$). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)
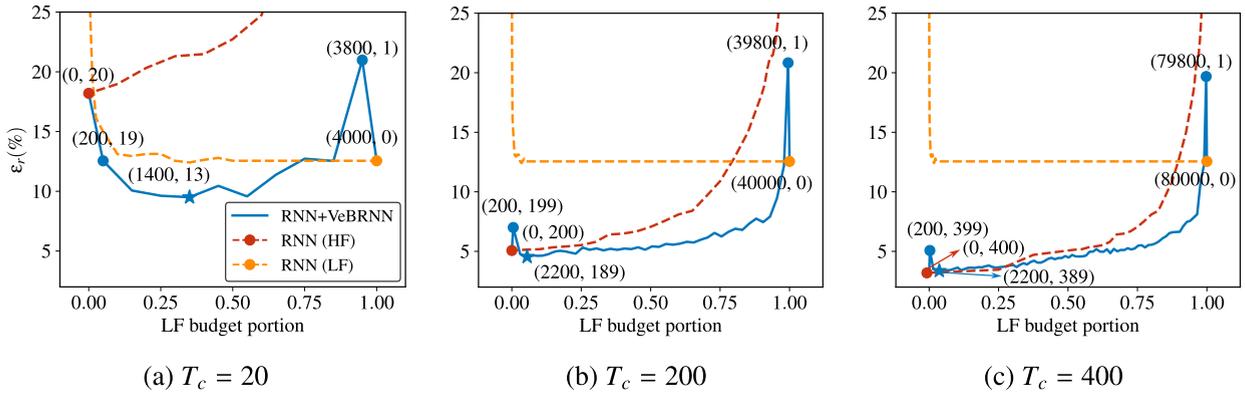
(a) $T_c = 20$ (b) $T_c = 200$ (c) $T_c = 400$

Fig. 17. Results for Dataset 4 ($\widetilde{\mathrm{ROM}} + \overline{\mathrm{DNS}}$ with $N_{\mathrm{cluster}} = 18$) when considering different budget percentages of LF data acquisition for a given total budget cost: (a) $T_c = 20$, (b) $T_c = 200$, and (c) $T_c = 400$. Each point is labeled with the number of training paths used at the LF and HF, respectively. The red and orange dashed lines represent the relative errors of using only the HF or LF data from the MF model train setup counterpart shown in the parentheses. Confront these results with Fig. 13(b), obtained using 2000 LF training paths and a different number of HF training paths leading to the corresponding total cost in that figure. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

acquisition of the HF and LF data only differs by a factor of 6 (see Table 1). Therefore, if the HF and LF data have similar acquisition cost, and similar accuracy gain for each additional total cost made available, then the MF model loses usefulness.

A similar observation arises from considering Dataset 4 ($\widetilde{\mathrm{ROM}} + \overline{\mathrm{DNS}}$). Section 4.4 showed that if the LF data is obtained by a reasonably accurate ROM (SCA method discretized by $N_{\mathrm{cluster}} = 18$ clusters), then the MF model is better for a large range of total cost values. This is no longer the case if the ROM is less accurate (SCA discretized by $N_{\mathrm{cluster}} = 3$ clusters). Fig. 17 complements this observation by showing model error as a function of the percentage of LF budget used for a given total cost. In this figure, we see that for low total cost ($T_c = 20$) there is a wide range where the MF model is better than the single-fidelity counterparts (the best MF model is highlighted by a star marker, and curiously it is close to the case considering 2000 training paths for the LF in Section 4.4). Unsurprisingly, if the total budget is high ($T_c = 200$ or $T_c = 400$), then there is already enough budget to effectively learn by only using HF data. This explains why the best MF model for $T_c = 200$ is only marginally better than the HF model, and why none of the MF models is better than the HF model for $T_c = 400$.

We also note that in all cases (Figs. 16 and 17) we see that if we use very few samples for HF data (or conversely for LF data), then the MF model is worse than the corresponding single-fidelity model (notice the spikes close to the extremes of the plots). This is logical because if there are very few samples of a given fidelity, then the model obtained for that fidelity has a large error (dashed lines). In that case, the information transferred from that fidelity within the MF model is not sufficiently useful (in fact, it disrupts model performance). Interestingly, we see this behavior quickly dissipating, i.e., even a small number of points (but not extremely small) becomes beneficial. Our investigations point to the heuristic conclusion that LF data acquisition should be allocated between 30 % and 70 % of the total budget, although this depends on the three key factors mentioned above.

6. Conclusions and future work

We demonstrate that variance estimation Bayesian recurrent neural networks (VeBRNNs) can be used within a multi-fidelity (MF) framework to learn nonlinear, history-dependent phenomena while quantifying and disentangling uncertainties. Our contribution is modular, since the most general MF model that uses VeBRNNs in each fidelity can be simplified (or ablated) to the simplest single-fidelity model that uses a deterministic feedforward neural network model (Fig. 4 explains how to reduce a MF model with an arbitrary number of fidelities to a single-fidelity model; Fig. 5 shows how to progressively choose more complex neural network models up to VeBRNNs).

Furthermore, we highlight that VeBRNNs are trained by a cooperative strategy that also simplifies into standard deterministic training of a neural network when only performing Step 1 of Algorithm 1. If aleatoric uncertainty does not need to be estimated, we can skip Step 2 of Algorithm 1. If epistemic uncertainty is also not needed, we can skip Step 3 of Algorithm 1. Therefore, the presented methodology seamlessly generalizes conventional data-driven learning from the simplest deterministic cases to the most complete Bayesian cases presented to date. We expect this work will open new avenues in data-driven design and analysis, far beyond constitutive modeling.

Without loss of generality, the examples in this article focused on data-driven learning of constitutive models from data generated by simulations of material volume elements of history-dependent plasticity of elastoplastic biphasic materials with or without aleatoric uncertainty (data noise). Four different scenarios were presented: one single-fidelity case with noisy data, and three bi-fidelity cases considering noisy and noiseless data for different fidelities. The proposed method can quantify and separate epistemic and aleatoric uncertainty at every fidelity (when using VeBRNNs). In other words, the models simultaneously estimate the quality of the mean response (epistemic uncertainty) and the inherent noise present in the data (aleatoric uncertainty).

The proposed method is envisioned to be applied across a wide range of Science and Engineering problems, including but not limited to realistic microstructure reconstruction and inverse design of advanced materials. However, we believe that further generalizing this work to include active learning and applying it to Bayesian optimization is also important. This will enable on-the-fly data acquisition, selecting the fidelity from which the next data point should be acquired (for learning or for optimization).

CRediT authorship contribution statement

Jiaxiang Yi: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization; **Bernardo P. Ferreira:** Writing – review & editing, Visualization, Software, Methodology, Investigation, Formal analysis; **Miguel A. Bessa:** Writing – review & editing, Writing – original draft, Visualization, Supervision, Resources, Project administration, Investigation, Funding acquisition, Formal analysis, Conceptualization.

Data availability

Code and datasets for replicating this work: https://github.com/bessagroup/MF-VeBRNN

Declaration of competing interest

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

Acknowledgment

Appendix A. Bayesian inference methods

We have mentioned that MCMC and VI are two fundamental approaches for approximating the PPD (Eq. (D.8)). In the following subsections, we briefly introduce those approaches in Appendices A.1 and A.2, respectively.

A.1. Markov chain Monte Carlo sampling

MCMC is a class of algorithms that can be used to sample from a probability distribution directly [60]. The most straightforward approach is the random walk Metropolis-Hasting algorithm [31], while it suffers from high rejection rates for high-dimensional problems. To address this limitation and accelerate the mixing rate of MCMC, the Hamiltonian Monte Carlo (HMC) was developed by Neal et al. [31]. HMC leverages the concept of Hamiltonian mechanics by incorporating the **gradient** of the target probability density to guide the sampling trajectory. Therefore, the mixing rate can be improved tremendously compared with the random walk Metropolis-Hasting algorithm.

A.1.1. Hamiltonian Monte Carlo

The schematic of the Hamiltonian mechanics system is shown in Fig. A1, where the total energy is:

$$\mathcal{H}(\theta, \boldsymbol{v}) = \varepsilon(\theta) + \mathcal{K}(\boldsymbol{v}). \tag{A.1}$$

Generally, the potential energy with parameter θ is usually set to be the QoI, specifically, the negative log PPD shown in Eq. (D.8):

$$\varepsilon(\theta) = -\log \tilde{p}(\theta), \tag{A.2}$$

and the kinetic energy to be:

$$\mathcal{K}(\boldsymbol{v}) = \frac{1}{2} \boldsymbol{v}^T \Sigma^{-1} \boldsymbol{v}, \tag{A.3}$$

where Σ is the mass matrix and $\boldsymbol{v}$ is the velocity.

As illustrated in Fig. A1, the bowl-shaped surface represents the target PDF landscape, where the blue dot points are the expected sample points. The HMC can be interpreted as going over the whole landscape in light of the position updating rule of the Hamiltonian mechanics system. To further improve the accuracy of the sampling process, the leapfrog technique is adopted [31], which is a symplectic integration operator that can preserve the energy of the system. Overall, the algorithm is summarized in Algorithm 2.

As shown in Algorithm 2, HMC still involves an acceptance–rejection step. Although it significantly improves over the random-walk Metropolis-Hastings algorithm, this step can still hinder its efficiency in high-dimensional problems, such as obtaining the posterior distribution of BNNs in Eq. (D.8).
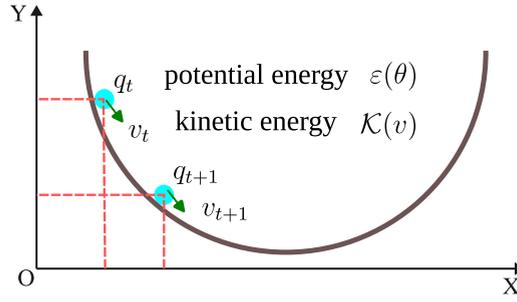
Fig. A1. Schematic of Hamiltonian mechanics system.

Algorithm 2: Hamiltonian Monte Carlo Algorithm

Data: target $p(\theta)$, proposed $q(\theta'|\theta) \sim \mathcal{N}(\theta'|\theta, \eta^2 I)$, burn-in iteration N_b, total iteration N_t, step size η, leapfrog steps L

Result: Collection of $\Theta = \{\theta^1, \theta^2, ... \theta^{N_a}\}$

initialize θ^0

for $t = 0, 1, 2, ..., T$ **do**

    Sample random momentum $\boldsymbol{v}_{t-1} \sim N(\mathbf{0}, \boldsymbol{\Sigma})$

    Set $(\theta'_0, \boldsymbol{v}'_0) = (\theta_{t-1}, \boldsymbol{v}_{t-1})$

    Half step for momentum: $\boldsymbol{v}'_{\frac{1}{2}} = \boldsymbol{v}'_t - \frac{\eta}{2}\nabla\mathcal{f}(\theta'_0)$

    for $l = 1 : L - 1$ **do**

        $\theta'_l = \theta'_{l-1} + \eta\boldsymbol{\Sigma}^{-1}\boldsymbol{v}'_{l-\frac{1}{2}}$

        $\boldsymbol{v}'_{l+\frac{1}{2}} = \boldsymbol{v}'_{l-\frac{1}{2}} - \eta\nabla\mathcal{f}(\theta'_l)$

    end

    Full step for location: $\theta'_L = \theta'_{L-\frac{1}{2}} + \eta\boldsymbol{\Sigma}^{-1}\boldsymbol{v}'_{L-\frac{1}{2}}$

    Half step for momentum: $\boldsymbol{v}'_L = \boldsymbol{v}'_{L-\frac{1}{2}} - \frac{\eta}{2}\nabla\mathcal{f}(\theta'_L)$

    Compute $\alpha = min(1, \exp\left[-\mathcal{H}(\theta_L, \boldsymbol{v}_L) + \mathcal{H}(\theta_{t-1}, \boldsymbol{v}_{t-1})\right])$

    Set $\theta_t = \theta_L$ with probability α, otherwise $\theta_t = \theta_{t-1}$

end

A.1.2. Stochastic gradient Langevin dynamics

To address the scalability bottleneck of HMC, Welling et al. [61] developed a novel Bayesian inference approach called Stochastic Gradient Langevin Dynamics (SGLD), leveraging Stochastic Gradient Descent (SGD) [62] and Langevin Monte Carlo (LMC) [60]. As mentioned in Algorithm 2, the *leapfrog* step is essential to the accuracy of HMC, where a larger L is expected for a higher acceptance rate, while requiring more posterior evaluations. LMC can be regarded as a special case of HMC for $L = 1$ whose procedure is listed in Algorithm 3.

Algorithm 3: Langevin Monte Carlo Algorithm

Data: target $p(\theta)$, proposed $q(\theta'|\theta) \sim \mathcal{N}(\theta'|\theta, \eta^2 I)$, burn-in iteration N_b, total iteration N_t, step size η

Result: Collection of $\Theta = \{\theta^1, \theta^2, ... \theta^N\}$

initialize θ^0

for $t = 1, 2, ..., T$ **do**

    Sample random momentum $\boldsymbol{v}_{t-1} \sim N(\mathbf{0}, \boldsymbol{\Sigma})$

    Update the location: $\theta' = \theta_{t-1} - \frac{\eta^2}{2}\boldsymbol{\Sigma}^{-1}\nabla\varepsilon(\theta_{t-1}) + \eta\boldsymbol{\Sigma}^{-1}\boldsymbol{v}_{t-1}$

    Update the velocity: $\boldsymbol{v}' = \boldsymbol{v}_{t-1} - \frac{\eta}{2}\nabla\mathcal{f}(\theta_{t-1}) - \frac{\eta}{2}\nabla\mathcal{f}(\theta')$

    Compute $\alpha = min(1, \exp\left[-\mathcal{H}(\theta', \boldsymbol{v}') + \mathcal{H}(\theta_{t-1}, \boldsymbol{v}_{t-1})\right])$

    Set $\theta_t = \theta'$ with probability α, otherwise $\theta_t = \theta_{t-1}$

end

The position updating rule of LMC includes an additional term, $\frac{\eta^2}{2}\boldsymbol{\Sigma}^{-1}\nabla\varepsilon(\theta_{t-1})$, compared with that of Algorithm 2. This extra term follows the Langevin diffusion process. Also, there is a simplification to eliminate the MH acceptance step, making the algorithm more efficient in practice:

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \frac{\eta^2}{2} \boldsymbol{\Sigma}^{-1} \nabla \varepsilon(\boldsymbol{\theta}_{t-1}) + \eta \boldsymbol{\Sigma}^{-1} \boldsymbol{v}_{t-1} \tag{A.4}$$

$$= \boldsymbol{\theta}_{t-1} - \frac{\eta^2}{2} \boldsymbol{\Sigma}^{-1} \nabla \varepsilon(\boldsymbol{\theta}_{t-1}) + \eta \sqrt{\boldsymbol{\Sigma}^{-1}} \boldsymbol{\epsilon}_{t-1}, \tag{A.5}$$

where $\boldsymbol{v}_{t-1} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ and $\boldsymbol{\epsilon}_{t-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. It is noted that the above updating is similar to the SGD by adding noise to it ($\eta \sqrt{\boldsymbol{\Sigma}^{-1}} \boldsymbol{\epsilon}_{t-1}$). When $\boldsymbol{\Sigma} = \mathbf{I}$ and $\eta = \sqrt{2}$, the updating process follows the Langevin diffusion written

$$d\boldsymbol{\theta}_t = -\nabla \varepsilon(\boldsymbol{\theta}_t) dt + \sqrt{2} d\boldsymbol{B}_t, \tag{A.6}$$

where $\boldsymbol{B}_t$ is a d-dimensional Brownian motion.

Now, let's revisit the SGD with a loss function [62]:

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{n=1}^{N} \mathcal{L}_n(\boldsymbol{\theta}). \tag{A.7}$$

By taking a mini-batch, we can reformulate the mini-batch loss function and calculate the corresponding gradient as

$$\nabla_B \mathcal{L}(\boldsymbol{\theta}) = \frac{1}{B} \sum_{n=1}^{B} \mathcal{L}_n(\boldsymbol{\theta}). \tag{A.8}$$

There is a difference (diffusion term) between the actual gradient and the mini-batch one, expressed as

$$\boldsymbol{v}_t = \sqrt{\eta} \big(\nabla \mathcal{L}(\boldsymbol{\theta}) - \nabla_B \mathcal{L}(\boldsymbol{\theta}) \big). \tag{A.9}$$

To this end, SGD with mini-batch can be rewritten as

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \nabla_B \mathcal{L}(\boldsymbol{\theta}_t) = \boldsymbol{\theta}_t - \eta \nabla \mathcal{L}(\boldsymbol{\theta}_t) + \sqrt{\eta} \boldsymbol{v}_t, \tag{A.10}$$

where $\boldsymbol{v}_t$ has a zero mean.

Combining Eqs. (A.4) and (A.10), the updating process of SGLD is obtained by substituting the $p(\boldsymbol{\theta}|\boldsymbol{X}) \propto p(\boldsymbol{\theta}) \prod_{n=1}^{N} p(y_i|\boldsymbol{\theta})$ to the loss function $\mathcal{L}(\boldsymbol{\theta})$ [61]:

$$\Delta \boldsymbol{\theta}_t = \frac{\epsilon_t}{2} \Big(\nabla \log p(\boldsymbol{\theta}_t) + \frac{N}{n} \log p(y_i|\boldsymbol{\theta}_t) \Big) + \boldsymbol{\eta}_t, \tag{A.11}$$

where $\boldsymbol{\eta}_t \sim \mathcal{N}(\mathbf{0}, \epsilon_t)$.

However, SGLD assumes that all parameters $\boldsymbol{\theta}$ have the same step size, leading to slow convergence or even divergence in the cases where the components of $\boldsymbol{\theta}$ have different curvatures. Thus, a refined version of SGLD, called preconditioned SGLD (pSGLD) [44], was proposed to address this issue. In pSGLD [44], the update rule incorporates a user-defined preconditioning matrix $G(\boldsymbol{\theta}_t)$, which adjusts the gradient updates and the noise term adaptively:

$$\Delta \boldsymbol{\theta}_t = \frac{\epsilon_t}{2} \Big[G(\boldsymbol{\theta}_t) \Big(\nabla \log p(\boldsymbol{\theta}_t) + \frac{N}{n} \log p(y_i|\boldsymbol{\theta}_t) \Big) + \Gamma(\boldsymbol{\theta}_t) \Big] + \boldsymbol{\eta}_t G(\boldsymbol{\theta}_t), \tag{A.12}$$

where $\Gamma_i = \sum_j \frac{\partial G_{i,j}(\boldsymbol{\theta})}{\partial \theta_j}$ describes how the preconditioner changes with respect to $\boldsymbol{\theta}$.

A.2. Variational inference (VI)

VI is another Bayesian inference method that approximates the PPD by a proposed distribution from a parametric family [32]. The objective of VI is to minimize the Kullback-Leibler divergence between the true posterior distribution $p(\boldsymbol{\theta}|D)$ and the proposed distribution $q(\boldsymbol{\theta})$ as illustrated in Fig. A2.

In practice, ψ is regarded as the variational parameter from a parametric family Ω; therefore, the optimal ψ^* can be obtained by minimizing the KL divergence as follows:

$$\psi^* = \arg\min_{\psi} D_{\mathbb{KL}} \big(q_\psi(\boldsymbol{\theta}) || p_\theta(\boldsymbol{\theta}|D) \big) \tag{A.13}$$

$$= \arg\min_{\psi} \mathbb{E}_{q_\psi(\boldsymbol{\theta})} \big[\log q_\psi(\boldsymbol{\theta}) - \log p_\theta(\boldsymbol{\theta}|D) \big] \tag{A.14}$$

$$= \arg\min_{\psi} \mathbb{E}_{q_\psi(\boldsymbol{\theta})} \left[\log q_\psi(\boldsymbol{\theta}) - \log \left(\frac{p_\theta(D|\boldsymbol{\theta}) p_\theta(\boldsymbol{\theta})}{p_\theta(D)} \right) \right] \tag{A.15}$$

$$= \arg\min_{\psi} \mathbb{E}_{q_\psi(\boldsymbol{\theta})} \big[\log q_\psi(\boldsymbol{\theta}) - \log p_\theta(D|\boldsymbol{\theta}) - \log p_\theta(\boldsymbol{\theta}) + \log p_\theta(D) \big] \tag{A.16}$$

$$= \arg\min_{\psi} \mathbb{E}_{q_\psi(\boldsymbol{\theta})} \big[\log q_\psi(\boldsymbol{\theta}) - \log p_\theta(D|\boldsymbol{\theta}) - \log p_\theta(\boldsymbol{\theta}) \big] + \log p_\theta(D). \tag{A.17}$$

According to Eq. (A.13), the optimization problem can be decomposed into two terms: the first term is the negative evidence lower bound (ELBO), and the second term is the log marginal evidence. The log marginal evidence is a constant term that does not depend on ψ; therefore, we only need to optimize the first term alone:
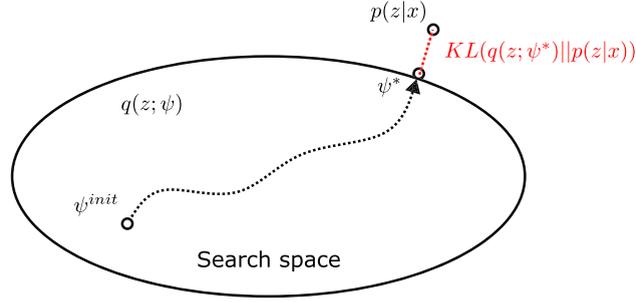
Fig. A2. Schematic of VI [60]: The ellipse represents the search space of the proposed distribution, $p(\theta|D)$ is the true posterior distribution, the KL divergence is the distance between the two distributions defined as $D_{\mathbb{KL}}\big(q(z)||p_\theta(z|x)\big)$, and the goal is to minimize it.

$$\psi^* = \arg\min_\psi \mathbb{E}_{q_\psi(\theta)}\Big[\log q_\psi(\theta) - \log p_\theta(D|\theta) - \log p_\theta(\theta)\Big] \tag{A.18}$$

$$= \arg\min_\psi \mathbb{E}_{q_\psi(\theta)}\Big[-\log p_\theta(D|\theta)\Big] + D_{\mathbb{KL}}(q_\psi(\theta)||p_\theta(\theta)), \tag{A.19}$$

where the first term is the negative log likelihood of the given dataset, and the second term is the KL divergence between the proposed distribution and the prior distribution. The optimization problem can be solved by any optimization algorithm, such as SGD [62] or Adam [59].

Appendix B. Performance metrics

To evaluate the performance of different methods, we adopt the following metrics evaluated based on the test dataset:

- **Relative error (ϵ_r)**

$$\epsilon_r = \frac{1}{N_{\text{test}}T}\sum_{i=1}^{N_{\text{test}}}\sum_{t=1}^{T}\left(\frac{\|\bar{\mathbf{y}}_{i,t} - \hat{\mathbf{y}}_{i,t}\|_F}{\|\bar{\mathbf{y}}_{i,t}\|_F}\right)\cdot 100\,\%, \tag{B.1}$$

where N_{test} is the number of HF testing points, $\|\cdot\|_F$ is a Frobenius norm, $\bar{\mathbf{y}}_{i,t}$ and $\hat{\mathbf{y}}_{i,t}$ are the mean values of predicted and ground truth of ith test sample and t^{th} time step, correspondingly.

- **Test Log-Likelihood (TLL)**

$$\text{TLL} = \frac{1}{N_{\text{test}}T}\sum_{i=1}^{N_{\text{test}}}\sum_{t=1}^{T}\log p(\bar{\mathbf{y}}_{i,t}|\theta). \tag{B.2}$$

- **Wasserstein distance (WA)**

$$\text{WA} = \frac{1}{N_{\text{test}}T}\sum_{i=1}^{N_{\text{test}}}\sum_{t=1}^{T}W_2\big(p(\bar{\mathbf{y}}_{i,t}|\theta), q(\bar{\mathbf{y}}_{i,t}|\theta)\big), \tag{B.3}$$

where $p(\bar{\mathbf{y}}_{i,t}|\theta)$ and $q(\bar{\mathbf{y}}_{i,t}|\theta)$ are the predicted and ground truth distributions, respectively. If Gaussian assumptions are applied, we can rewrite Eq. (B.3) into:

$$\text{WA} = \frac{1}{N_{\text{test}}T}\sum_{i=1}^{N_{\text{test}}}\sum_{t=1}^{T}\sqrt{(\bar{\mathbf{y}}_{i,t} - \hat{\mathbf{y}}_{i,t})^2 + (\bar{\mathbf{s}}_{i,t} - \hat{\mathbf{s}}_{i,t})^2}. \tag{B.4}$$

- **Prediction Interval Coverage Probability (PICP)**

$$\text{PICP} = \frac{1}{N_{\text{test}}T}\sum_{i=1}^{N_{\text{test}}}\sum_{t=1}^{T}\mathbb{1}\Big[\mathbf{y}_{i,t} \in \Big[\hat{\mathbf{y}}_{i,t}^{\text{lower}}, \hat{\mathbf{y}}_{i,t}^{\text{upper}}\Big]\Big]. \tag{B.5}$$

- **Mean Prediction Interval Width (MPIW)**

$$\text{MPIW} = \frac{1}{N_{\text{test}}T}\sum_{i=1}^{N_{\text{test}}}\sum_{t=1}^{T}\Big(\hat{\mathbf{y}}_{i,t}^{\text{upper}} - \hat{\mathbf{y}}_{i,t}^{\text{lower}}\Big), \tag{B.6}$$

where $\hat{\mathbf{y}}_{i,t}^{\text{lower}}$ and $\hat{\mathbf{y}}_{i,t}^{\text{upper}}$ are the predicted lower and upper bounds based on the epistemic uncertainty with a confidence level of 0.95.

Appendix C. Additional information about the datasets

C.1. Random polynomial strain-stress paths

We adopt the random polynomial strain path developed in our previous work [8,55], whose generation process is briefly given by Fig. C1.

As illustrated in Fig. C1, the control points are randomly sampled within the range $[\varepsilon_{ij,\min}, \varepsilon_{ij,\max}]$ for each strain component ε_{ij}, where $i, j = 1, 2$. A quadratic curve is then fitted through these points to construct the strain path for each component (blue curve). This continuous path is subsequently discretized into T load steps. In this work, we use 6 control points and set the total number of load steps to $T = 100$. Given the prescribed strain paths, the corresponding history-dependent stress responses can be computed using any solver, where an example is shown in Fig. 2.

C.2. Convergence analysis to determine the size of an RVE

By definition, a Representative Volume Element (RVE) should have negligible aleatoric uncertainty when randomizing the microstructure. As shown in Fig. C2, when reducing the size of the material volume element (Fig. 2(a)), the response exhibits aleatoric uncertainty (noise), i.e., we have a Stochastic Volume Element (SVE) instead of an RVE.

We conduct a convergence analysis based on finite element analyses of different volume element sizes. The results are summarized in Fig. C2, where we can see that a volume element with a length of 0.1 mm can be considered an RVE. Furthermore, we can also see from Fig. C2 that the stress variation decreases as the size of the microstructure unit cell increases, highlighting the fact that aleatoric
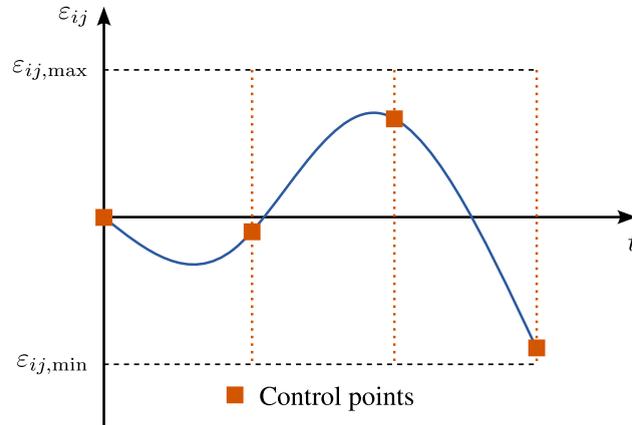

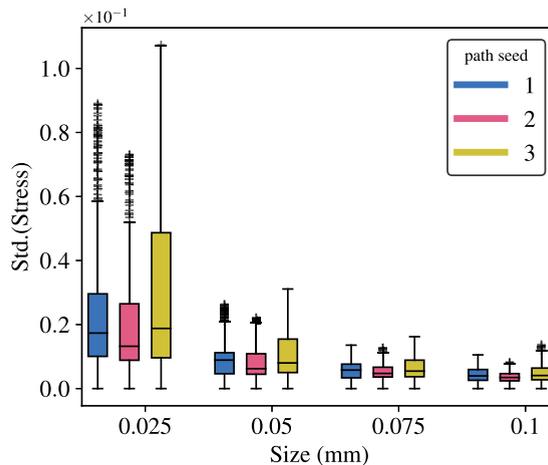**Fig. C1.** Generation of the random polynomial strain paths.

Fig. C2. Convergence analysis of the size of the microstructure volume element and the origin of aleatoric uncertainty (noise). Different colors represent different seeds for strain path generation. The size of the volume element varies from 0.025 mm to 0.10 mm, and we repeat the simulation 100 times to get the statistics.
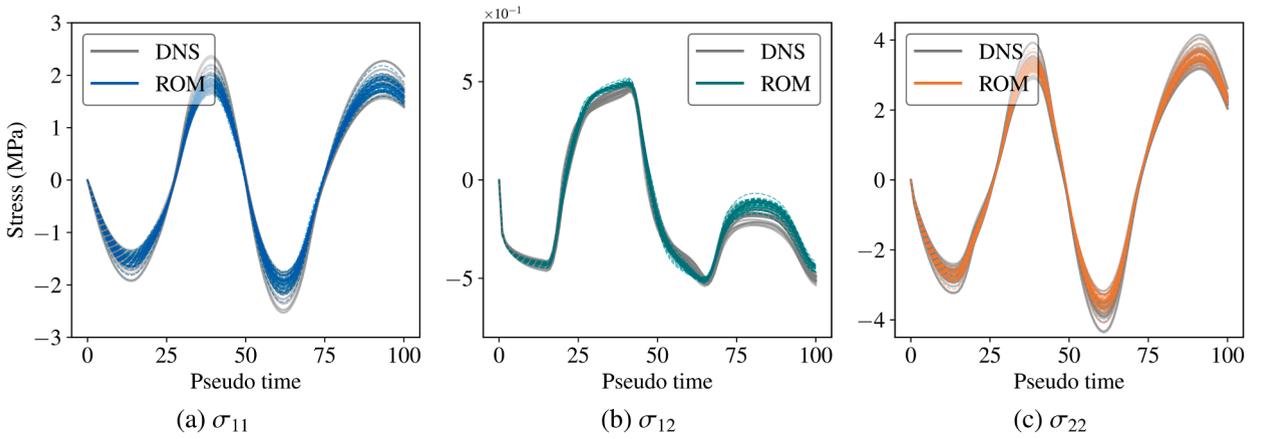
Fig. C3. Aleatoric uncertainty of **SCA-SVE** with $N_{\text{cluster}} = 18$ in comparison to **DNS-SVE**. From left to right we see the stress components $\sigma_{11}, \sigma_{12}, \sigma_{22}$. Colored lines indicate responses obtained by DNS (FEA), and gray lines by ROM (SCA). (a) σ_{11}; (b) σ_{12}; (c) σ_{22}.
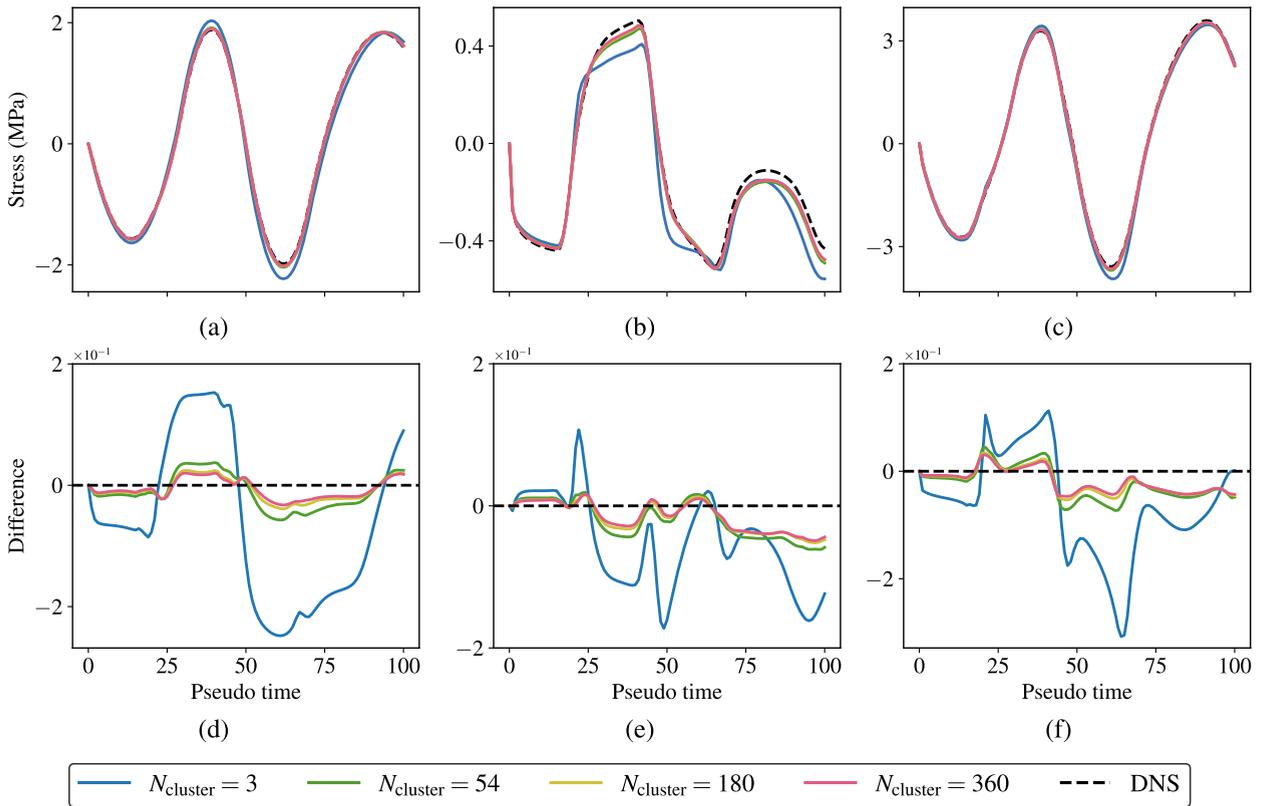

Fig. C4. Bias error introduced by **SCA-RVE** when compared to **DNS-RVE** and considering different numbers of clusters for the SCA method. The columns from left to right refer to the different stress components: σ_{11}, σ_{12}, and σ_{22}, respectively.

uncertainty originated because of inadequate statistical information. In other words, the aleatoric uncertainty will disappear when RVE is utilized; then it becomes a deterministic dataset as listed in Section 2.

C.3. Aleatoric uncertainty

C.3.1. Note on the aleatoric uncertainty of $\widetilde{DNS}$ (data obtained from DNS of an RVE)

As discussed in the main text, RVEs lead to deterministic responses because they contain sufficient morphological and topological information to produce consistent homogenized stress responses [63]. However, aleatoric uncertainty or noise arises from randomizing the microstructure of SVEs because, by definition, they are a small domain of material that is not representative, leading to variations

in the homogenized stress responses. Nevertheless, the mean response across multiple SVEs converges to that of the corresponding RVE. These variations correspond to aleatoric uncertainty in our dataset.

Specifically, the particles – occupying a 30 % volume fraction – are modeled as circles with radii sampled from a normal distribution $\mathcal{N}(0.003, 0.001^2)$. Based on a convergence analysis presented in Appendix C.2, we set the RVE size to 0.1 mm with a realization shown in Fig. 2(a). Finite Element Analysis (FEA) is adopted as direct numerical simulation (DNS) [64], where the RVE is discretized into 400×400 elements. The homogenized stress response undergoing a prescribed random polynomial strain path (Fig. 2(b)) is depicted as the solid lines in Fig. 2(c).

On the other hand, the SVE is defined with a domain size of 0.025 mm as illustrated by the red window in Fig. 2a. By moving this window within the RVE domain, multiple SVE realizations can be obtained, although we ensure that the microstructure remains periodic. In this case, the local volume fraction varies depending on the window's position; thus, we model the SVE volume fraction using a normal distribution $\mathcal{N}(0.3, 0.03^2)$ to reflect this spatial variability. In Fig. 2(c), we observe noticeable variability in the stress components σ_{11} and σ_{22}, while the variation in shear stress σ_{12} is significantly smaller. Additionally, the stochastic SVE responses exhibit heteroscedastic aleatoric uncertainty (noise is different for different input values), both across different stress components and along the strain load steps.

C.3.2. Note on the aleatoric uncertainty of $\widetilde{ROM}$ (data obtained from SCA of an SVE)

Fig. C3 compares the noisy results obtained from $\widetilde{DNS}$ (i.e., FEA of an SVE) with the noisy results obtained from $\widetilde{ROM}$ (i.e., SCA of an SVE). This allows for visualizing the differences in aleatoric uncertainty when data is obtained from different simulation methods (FEA vs. SCA). For consistency, note that the stress responses shown in Fig. C3 are obtained for the same strain path shown in Fig. 2(a).

Fig. C3 shows that the $\widetilde{ROM}$ has higher aleatoric uncertainty than the $\widetilde{DNS}$. This is reasonable because the aleatoric uncertainty in the ROM arises not only from the microstructure variations (the primary source), but also from the discretization level of the ROM, i.e., the number of clusters used in the SCA method (secondary source of aleatoric uncertainty) [7,65]. However, in this study, we focus on the total aleatoric uncertainty and do not further decompose its individual contributions.

C.4. Bias error introduced by $\overline{ROM}$ compared to $\overline{DNS}$ (data acquired by SCA of an RVE vs DNS of an RVE)

Conducting simulations with a fast reduced order model (ROM) when compared to a slow direct numerical simulation is a dominant factor, as illustrated in Fig. 1. This acceleration is achieved by the self-consistent clustering analysis (SCA) method [7,16,17], a type of reduced order model (ROM) that we developed previously. We have shown the accuracy and computational time comparison in Fig. 3, and the quadratic scaling of SCA with the discretization level (number of clusters N_{cluster}).

In this section, we further analyze the differences between $\overline{DNS}$ and $\overline{ROM}$ when varying the N_{cluster} for the SCA method – see Fig. C4. Intuitively, the difference reduces rapidly when the number of clusters increases. The difference is large for $N_{\text{cluster}} = 3$, but it reduces significantly for $N_{\text{cluster}} = 54$. After that, the improvement of the SCA with more clusters is limited, despite the dramatic increase in computational cost (Fig. 3). Therefore, we employ $N_{\text{cluster}} = 3$ and $N_{\text{cluster}} = 18$, serving as lower quality but faster LF data, and as higher quality LF data but relatively slower, respectively. The approximation error exhibited by the SCA method with respect to the DNS method originates from the limited number of clusters. Given that these methods are solely employed to generate the data in our application, once the number of clusters is determined, this discrepancy is fixed, and we therefore refer to it as a *bias error*. From the perspective of the solver alone, such an approximation error could be broadly regarded as epistemic uncertainty; yet, this should not be confused with the epistemic uncertainty of the machine learning model, which in this paper specifically denotes the uncertainty arising from the machine learning model itself.

Appendix D. Cooperative training of VeBRNNs

Training Bayesian models to provide good estimates of aleatoric and epistemic uncertainties is not trivial. Although BNNs are reported to be capable of modeling both aleatoric and epistemic uncertainties, state-of-the-art approaches either treat the noise as a fixed hyperparameter under a homoscedastic assumption [21], or embed it directly into the likelihood for end-to-end Bayesian inference [33]. We recently proposed a more efficient strategy to quantify aleatoric uncertainty [20]. The idea consists of a three-step cooperative training strategy, in which we first train a deterministic neural network for the mean only, then train another variance estimation (Ve) network deterministically for the aleatoric uncertainty, and finally execute Bayesian inference to update the mean and epistemic uncertainty. Furthermore, the second and third steps can be iterated if necessary. The cooperative training strategy for uncertainty disentanglement is summarized in the main text – see Algorithm 1.

D.1. Step 1: Mean network training

The first step of the cooperative training strategy involves training a mean network that follows the same procedure of conventional RNN training by minimizing the mean squared error (MSE) loss between the RNN outputs and the learning objective. As noted in Remark 3, the specific inputs and outputs of the RNN may vary depending on the architecture, as summarized in Table E1. For clarity, we denote the input as **x** and the output as **y**. Without any loss of generality, we demonstrate the proposed framework by adopting a GRU model [8,55] due to its simplicity and effectiveness, which is written as:

$$\mathbf{z}_t = \text{sigmoid}(\mathbf{W}_{xz}\mathbf{x}_t + \mathbf{b}_{xz} + \mathbf{W}_{hz}\mathbf{h}_{t-1} + \mathbf{b}_{hz}), \tag{D.1}$$

$$\mathbf{r}_t = \text{sigmoid}(\mathbf{W}_{xr}\mathbf{x}_t + \mathbf{b}_{xr} + \mathbf{W}_{hr}\mathbf{h}_{t-1} + \mathbf{b}_{hr}), \tag{D.2}$$

$$\tilde{\mathbf{h}}_t = \tanh\left(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{b}_{xh} + \mathbf{r}_t \odot \left(\mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{b}_{hh}\right)\right), \tag{D.3}$$

$$\mathbf{h}_t = \mathbf{z}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{z}_t) \odot \tilde{\mathbf{h}}_t, \tag{D.4}$$

where $\mathbf{h}_0 = \mathbf{0}$, $\mathbf{x}_t$ is the input vector at the time (or pseudo-time) step t, $\mathbf{h}_t$ is the output in the hidden state corresponding to that time step, and the final decoded prediction of the quantity of interest is $f(\mathbf{x}_t) = \mathbf{W}_{ho}\mathbf{h}_t + \mathbf{b}_{ho}$. With this step, we use *Adam* [59] to optimize the MSE loss between the decoded $f(\mathbf{x}_t)$ and $\mathbf{y}_t$, finding the point estimate of the learnable parameters $\theta = [W, b]$. Thus, the loss function can be given explicitly as:

$$\mathcal{L}_{\text{MSE}}(\theta) = \frac{1}{NT} \sum_{n=1}^{N} \sum_{t=1}^{T} \left(\mathbf{y}_{n,t} - f(\mathbf{x}_{n,t}; \theta)\right)^2, \tag{D.5}$$

where N is the number of training paths, T is the total pseudo time step.

D.2. Step 2: Variance network training for aleatoric uncertainty estimation

Once the mean is obtained from the first step, a variance network (also a GRU) is trained with the mean prediction fixed, as described in Algorithm 1. A crucial detail that facilitates stable training of this variance estimation network lies in its output representation. Specifically, the network does not directly predict the variance $s^2(\mathbf{x})$ as in Eq. (2). Instead, it is trained to model a Gamma distribution, enabling a more stable and expressive representation of aleatoric uncertainty. This follows from our previous proof that the square residual $r_t = \left(f(\mathbf{x}_t; \theta) - \mathbf{y}_t\right)^2$ follows a Gamma distribution since $\mathbf{y}_t$ is Gaussian. The corresponding Negative log-likelihood loss to train the variance network is defined by:

$$\mathcal{L}_{\text{Gamma}}(\phi) = \sum_{n=1}^{N} \sum_{t=1}^{T} \left[\alpha(\mathbf{x}_{n,t}; \phi) \log \frac{\lambda(\mathbf{x}_{n,t}; \phi)}{\Gamma(\alpha(\mathbf{x}_{n,t}; \phi))} - \left(\alpha(\mathbf{x}_{n,t}; \phi) - 1\right) \log r_{n,t} + \frac{\lambda(\mathbf{x}_{n,t}; \phi)}{r_{n,t}} \right], \tag{D.6}$$

where $\alpha(\mathbf{x}_t; \phi) > 0$ and $\lambda(\mathbf{x}_t; \phi) > 0$ are the shape and rate parameters of the Gamma distribution. After training the variance network, the expected value of the Gamma distribution becomes the desired heteroscedastic variance, which leads to:

$$s^2(\mathbf{x}_{1:T}; \phi) = \frac{\alpha(\mathbf{x}_{1:T}; \phi)}{\lambda(\mathbf{x}_{1:T}; \phi)}. \tag{D.7}$$

D.3. Step 3: Bayesian recurrent neural network training for updated mean and epistemic uncertainty

After completing **Step 2** training for the aleatoric variance, we aim to update the mean obtained from **Step 1** as well as get the epistemic uncertainty through Bayesian inference. The distinction between deterministic (**Step 1**) and Bayesian treatments is that the latter places the parameters in probabilistic distributions, whereas the deterministic approach treats them as real numbers.[12] The posterior distribution of the parameters can be obtained using Bayes' rule, expressed by [66],

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})}; \quad p(\mathcal{D}) = \int p(\mathcal{D}|\theta)p(\theta)d\theta, \tag{D.8}$$

where $p(\mathcal{D}|\theta)$ denotes the observation likelihood, $p(\theta)$ is the prior of the neural parameters, and $p(\mathcal{D})$ is the marginal likelihood. Following common practice in BNNs, we adopt a unit Gaussian distribution prior for neural parameters; moreover, $p(\mathcal{D}|\theta)$ is assumed to be independent and identically distributed (i.i.d.) Gaussian distribution:

$$p(\mathcal{D}|\theta) = \prod_{i=n}^{N} \prod_{t=1}^{T} \mathcal{N}\left(\mathbf{y}_{i,t}; f(\mathbf{x}_{n,t}; \theta), s^2(\mathbf{x}_{n,t}; \phi)\right), \tag{D.9}$$

where $s^2(\mathbf{x}; \phi)$ is the aleatoric (data) uncertainty (from **step 2**). Obtaining the posterior distribution in Eq. (D.8) is difficult because of the intractable high-dimensional probability integral. Thus, approximate methods such as MCMC [60] or VI [32,60] are required; we provide details in the Appendix A. After obtaining the posterior distribution of the parameters in Eq. (D.8), the predicted posterior distribution for any unknown point $\mathbf{x}'_{1:T}$ can be computed as

$$p(\mathbf{y}'_{1:T} \mid \mathbf{x}'_{1:T}, \mathcal{D}) = \int p(\mathbf{x}'_{1:T} \mid \mathbf{x}'_{1:T}, \theta)p(\theta \mid \mathcal{D})d\theta. \tag{D.10}$$

The predicted mean can be approximately obtained as

$$\mathbb{E}(\mathbf{y}'_{1:T} \mid \mathbf{x}'_{1:T}, \mathcal{D}) = \mathbb{E}_{p(\theta|D)}\left[f(\mathbf{x}_{1:T})\right], \tag{D.11}$$

and predicted variance as

---

[12] Theoretically, deterministic training can be regarded as finding the mode of PPD, refers to Maximum A Posterior (MAP); or assume all neural parameters following delta distributions
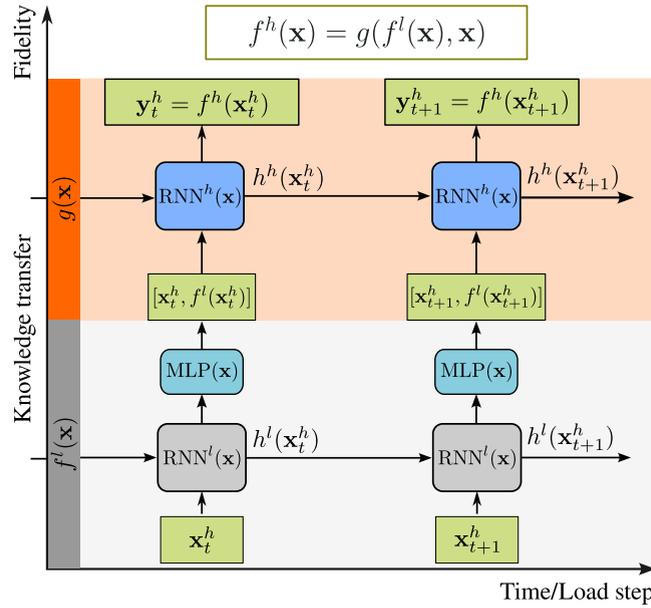
Fig. E1. Schematic of the proposed multi-fidelity recurrent neural network architecture: *MF-Nest-Output*. This configuration follows the formulation $f^h(\mathbf{x}) = g(f^l(\mathbf{x}), \mathbf{x})$, where $g(\cdot)$ is a non-linear transfer learning model–specifically, another RNN. This architecture does not include an explicit residual model. Meanwhile, the transfer model takes as input the decoded stress prediction from the low-fidelity RNN, concatenated with the high-fidelity strain inputs.

$$
\begin{aligned}
\mathrm{Var}(\mathbf{y}'_{1:T} \mid \mathbf{x}'_{1:T}, \mathcal{D}) &= \underbrace{\mathbb{E}_{p(\theta|D)}\left[s^2(\mathbf{x}_{1:T}; \boldsymbol{\phi})\right]}_{\text{Aleatoric Uncertainty}} + \underbrace{\mathrm{Var}_{p(\theta|D)}\left[f(\mathbf{x}_{1:T})\right]}_{\text{Epistemic Uncertainty}} \\
&= \underbrace{s^2(\mathbf{x}_{1:T}; \boldsymbol{\phi})}_{\text{Aleatoric Uncertainty}} + \underbrace{\mathrm{Var}_{p(\theta|D)}\left[f(\mathbf{x}_{1:T})\right]}_{\text{Epistemic Uncertainty}} .
\end{aligned}
\tag{D.12}
$$

Appendix E. Alternative multi-fidelity models and comprehensive investigations

E.1. Alternative multi-fidelity models

We described a multi-fidelity approach where an RNN was first trained on the LF data, followed by a linear transfer learning model, and another RNN was used to learn the residual between the high-fidelity response and the transfer-learned prediction. Experiments in Section 4 demonstrated its effectiveness across various datasets. However, we also clarified that multi-fidelity architecture is one particular architecture drawn from Fig. 4. Herein, we provide three other variants that can be obtained following Eq. (3), showing in Figs. E1–E3.

As shown in Figs. E1–E3, the key difference between the configurations using $f^h(\mathbf{x}) = g(f^l(\mathbf{x}), \mathbf{x})$ and $f^h(\mathbf{x}) = g(f^l(\mathbf{x})) + r(\mathbf{x})$ lies in the knowledge transfer. Specifically, for the configurations following $f^h(\mathbf{x}) = g(f^l(\mathbf{x}), \mathbf{x})$ (*MF-Nest-Output* and *MF-Nest-Hidden*), the transfer learning model $g(\cdot)$ is implemented as another RNN, and no residual model is used. In contrast, the configurations following $f^h(\mathbf{x}) = g(f^l(\mathbf{x})) + r(\mathbf{x})$ (*MF-Residual-Original* and *MF-Residual-Hidden*, as shown in Fig. 6) adopt a linear transfer learning model[13] and additionally employ a residual model–another RNN–to learn the discrepancy between the high-fidelity and low-fidelity predictions.

Another subtle distinction exists within each configuration regarding the inputs and outputs for the knowledge transfer module. For the configurations of *MF-Nest-Output* and *MF-Nest-Hidden*, the input to the transfer learning model can be formed by concatenating $\mathbf{x}^h$ with the decoded prediction of LF at $\mathbf{x}^h$ (i.e., the LF predicted stress) or with the hidden state at $\mathbf{x}^h$ of the LF model directly. Similarly, *MF-Residual-original* and *MF-Residual-Hidden* can select to concatenate the additional hidden state at $\mathbf{x}^h$ of the LF model or use $\mathbf{x}^h$ only. Based on these variations, we summarize the uniqueness of the inputs and outputs for transfer learning of each architecture in Table E1.

---

[13] We adopt the identity function as the transfer learning model in this work.
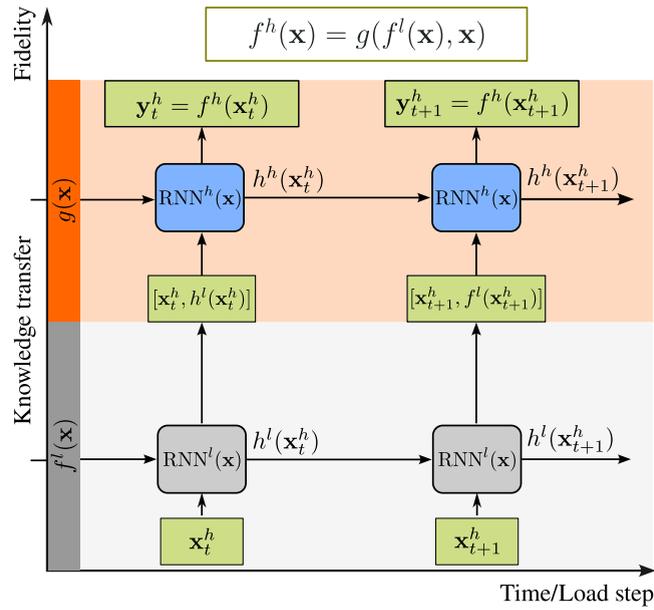
Fig. E2. Schematic of the proposed multi-fidelity recurrent neural network architectures: *MF-Nest-Hidden*. This configuration follows the formulation $f^h(\mathbf{x}) = g(f^l(\mathbf{x}), \mathbf{x})$, where $g(\cdot)$ is also another RNN and without a residual model. Instead, the transfer model takes as input the hidden state from the low-fidelity RNN, concatenated with the high-fidelity strain inputs.
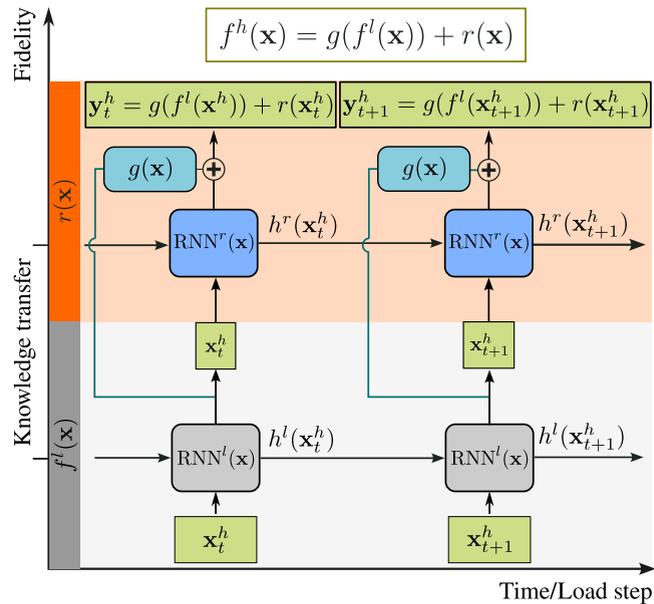

Fig. E3. Schematic of the proposed multi-fidelity recurrent neural network architecture: *MF-Residual-Original*. This configuration adopts the formulation $f^h(\mathbf{x}) = g(f^l(\mathbf{x})) + r(\mathbf{x})$, where $g(\cdot)$ is a linear transfer learning model and $r(\mathbf{x})$ is a residual model implemented as another RNN. Unlike the configuration shown in Fig. 6, this variant trains the residual model using only the original high-fidelity strain input.

Table E1
Summary of inputs and outputs for different multi-fidelity model variants from Fig. 4.

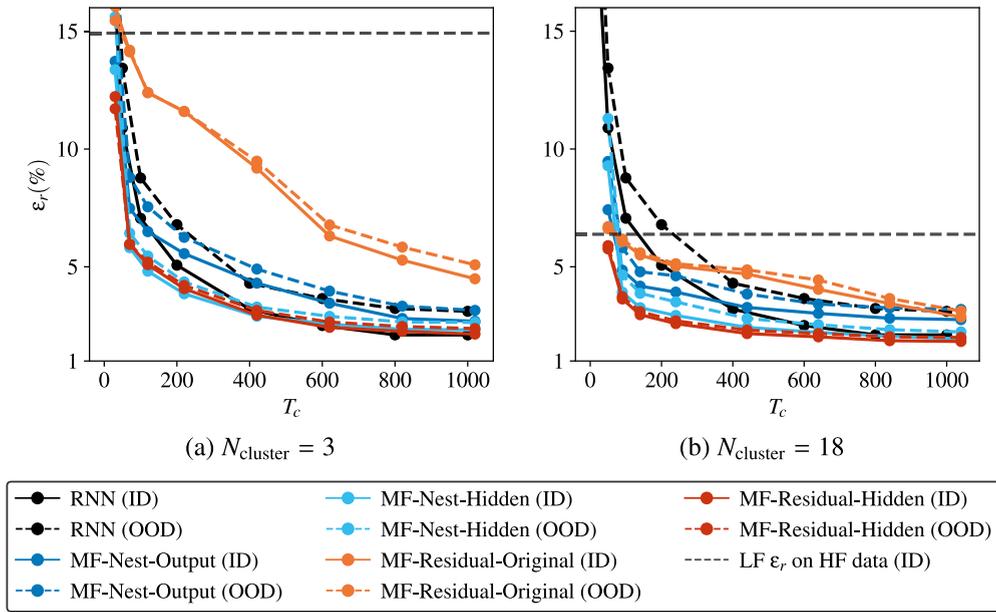| Variation	Input	Output / Learning target	Formula
MF-Nest-Output	$[\mathbf{x}^h, f^l(\mathbf{x}^h)]$	$\mathbf{y}^h$	$f^h(\mathbf{x}) = g(f^l(\mathbf{x}), \mathbf{x})$
MF-Nest-Hidden	$[\mathbf{x}^h, h^l(\mathbf{x}^h)]$	$\mathbf{y}^h$	$f^h(\mathbf{x}) = g(f^l(\mathbf{x}), \mathbf{x})$
MF-Residual-Original	$\mathbf{x}^h$	$\mathbf{y}^h - g(f^l(\mathbf{x}^h))$	$f^h(\mathbf{x}) = g(f^l(\mathbf{x})) + r(\mathbf{x})$
MF-Residual-Hidden	$[\mathbf{x}^h, h^l(\mathbf{x}^h)]$	$\mathbf{y}^h - g(f^l(\mathbf{x}^h))$	$f^h(\mathbf{x}) = g(f^l(\mathbf{x})) + r(\mathbf{x})$

Fig. E4. Comparison of different knowledge transfer configurations for **Dataset 2** with RNN + RNN training. The single-fidelity RNN training based on $\overline{\text{DNS}}$ results are shown with black lines for comparison. Meanwhile, the results of RNN + RNN training have offsets of 33.3 and 55.5 for $N_{\text{cluster}} = 3$ and $N_{\text{cluster}} = 18$ in the x-axis, considering the computational cost of low-fidelity data. (a) $N_{\text{cluster}} = 3$; (b) $N_{\text{cluster}} = 18$.

E.2. Comprehensive investigation on multi-fidelity architectures

We proposed four distinct multi-fidelity models in this paper, namely *MF-Nest-Hidden*, *MF-Nest-Output*, *MF-Residual-Hidden*, and *MF-Residual-Original*, with detailed input and learning objective summarized in Table E1. In this section, we conduct a comprehensive investigation to compare the performance of these configurations using **Dataset 2**. Specifically, an RNN + RNN architecture is first trained in Appendix E.2.1, and followed by an RNN + VeBRNN architecture trained in Appendix E.2.2.

E.2.1. RNN + RNN architecture training

We follow the same setup as Fig. 9 that uses all 2000 low-fidelity training paths to train the low-fidelity RNN model first. Then, we gradually increase the number of high-fidelity training paths from 10 to 1000. The results are presented in Fig. E4.

Different colors in Fig. E4 represent different MF model configurations proposed according to Eq. (4) and Fig. 4, where it is evident that a better LF model improves the performance of MF modeling. Among them, *MF-Residual-Hidden* is outperforming other MF model configurations across all experiments (achieving the lowest ϵ_r across all T_c) shown in Fig. E4. Conversely, *MF-Residual-Original* has the worst performance, showing an apparent gap to other MF model configurations. As for the other two MF model configurations–*MF-Nest-Hidden* and *MF-Nest-Output*–which rely on nonlinear transfer learning without a residual model, struggle to give robust predictions in cases where the number of high-fidelity training points is limited.

Interestingly, we observe that the MF model configurations using hidden-state-based knowledge transfer–*MF-Nest-Hidden* and *MF-Residual-Hidden*–consistently outperform their output-state-based counterparts–*MF-Nest-Output* and *MF-Residual-Original*, respectively. This is a noteworthy finding, as it suggests that directly transferring the hidden states from the low-fidelity model to the high-fidelity model is more effective than transferring decoded outputs (i.e., stresses). This observation is counterintuitive to previous studies in data-driven constitutive modeling, which often argue that hidden state representations are elusive and lack interpretability, making them unsuitable for transfer learning. Our results indicate that transferring hidden states bypasses the need to decode low-fidelity outputs and re-encode them into the HF representation space, as shown in Fig. E1. Furthermore, hidden states in Fig. E2 and Fig. 6 typically provide a high-dimensional representation of the underlying history-dependent material behavior, and thus may encode richer and more informative features than the observable outputs alone.

We also note that having a linear knowledge transfer, such as *MF-Residual-Original* and *MF-Residual-Hidden*, improves the OOD performance. Specifically, the OOD ϵ_r of *MF-Residual-Original* and *MF-Residual-Hidden* are almost the same as the ID ϵ_r, while *MF-Nest-Output* and *MF-Nest-Hidden* have a larger discrepancy between ID and OOD ϵ_r. This observation is consistent with our previous findings based on feedforward neural networks [19].

E.2.2. RNN + VeBRNN architecture training

In this section, we continue to test the four multi-fidelity model configurations, using deterministic training for the low-fidelity data and cooperative training for the high-fidelity data, specifically the RNN + VeBRNN architecture. The results trained based on different low-fidelity data of $\overline{\text{ROM}}$, i.e., $N\text{cluster} = 3$ and $N\text{cluster} = 18$, are shown in Figs. E5 and E6, respectively.

(a) Mean estimate metric ($\downarrow$) (b) Aleatoric uncertainty metric ($\downarrow$)

(c) Epistemic uncertainty metric ($\uparrow$) (d) Epistemic uncert. metric ($\rightarrow 0.95$) (e) Epistemic uncertainty metric ($\downarrow$)
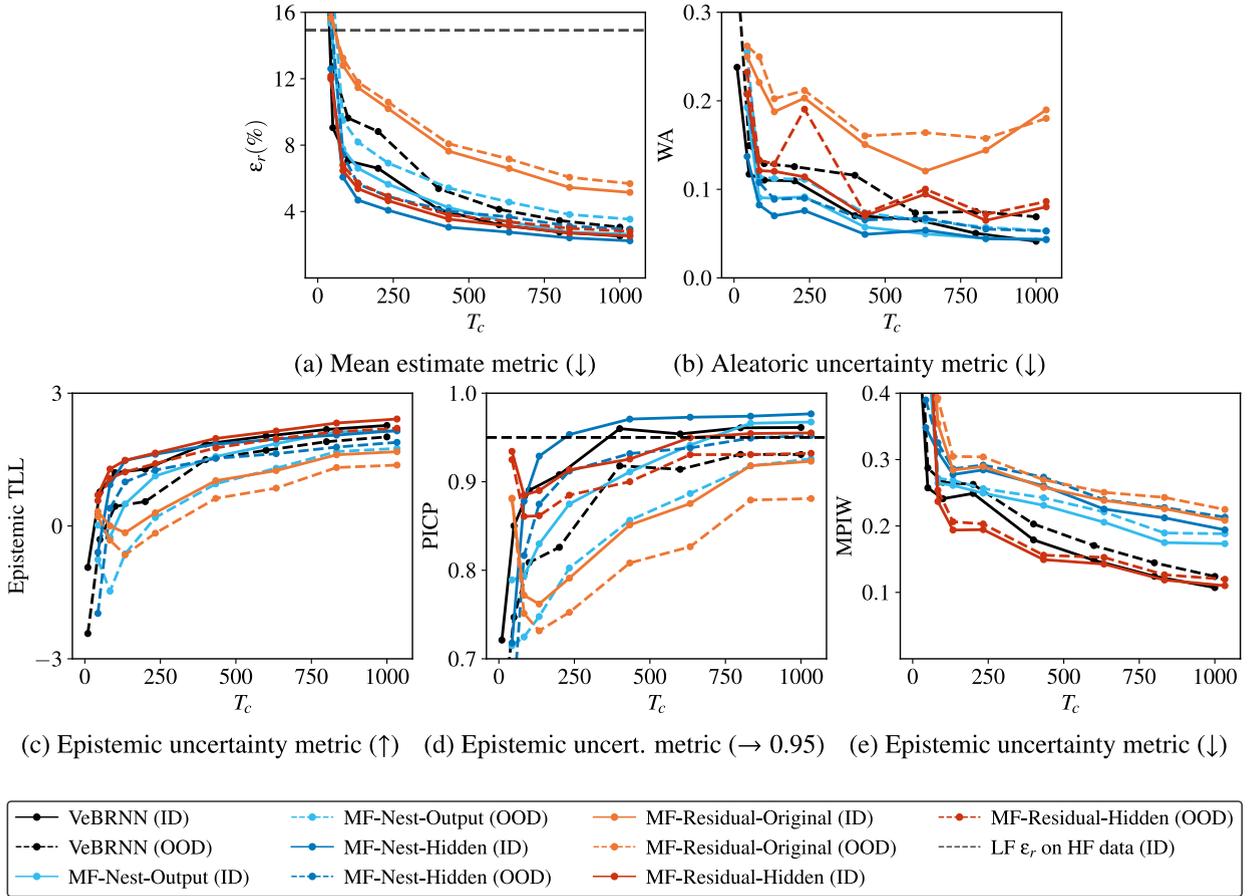
Fig. E5. Comparison of different multi-fidelity model configurations proposed in this paper for **Dataset 2** with RNN + VeBRNN architecture under low-fidelity of $N_{\text{cluster}} = 3$. (a) Mean estimate metric ε_r ($\downarrow$); (b) Aleatoric uncertainty metric WA ($\downarrow$); (c) Epistemic uncertainty metric TLL ($\uparrow$); (d) Epistemic uncertainty metric PICP ($\rightarrow 0.95$); (e) Epistemic uncertainty metric MPIW ($\downarrow$).

According to Figs. E5 and E6, the conclusion that the multi-fidelity model performs better with better low-fidelity data resources remains the same for all configurations. Furthermore, employing a cooperative training for the high-fidelity model does not alter the conclusion that *MF-Residual-Hidden* remains the best configuration. Moreover, results on Epistemic TLL and PICP provide strong evidence that *MF-Residual-Hidden* yields more consistent predictions across both ID and OOD data. It is particularly encouraging that the use of a transfer learning based on hidden states gives tighter or similar confidence intervals as shown Figs. E5(e) and E6(e), while leading the PICP close to the preset confidence level of 0.95. Concerning the Wasserstein distance, *MF-Nest-Hidden* is slightly worse than *MF-Nest-Output* and *MF-Nest-Hidden*; however, the prediction in Fig. E5 shows that the *MF-Nest-Hidden* already converged to a reasonable level for this deterministic dataset.

In summary, the results in Appendix E.2.1 and E.2.2 demonstrate that *MF-Residual-Hidden* is the best multi-fidelity model configuration among all four configurations proposed in this section. Therefore, we recommend using *MF-Residual-Hidden* for multi-fidelity data-driven plasticity law.
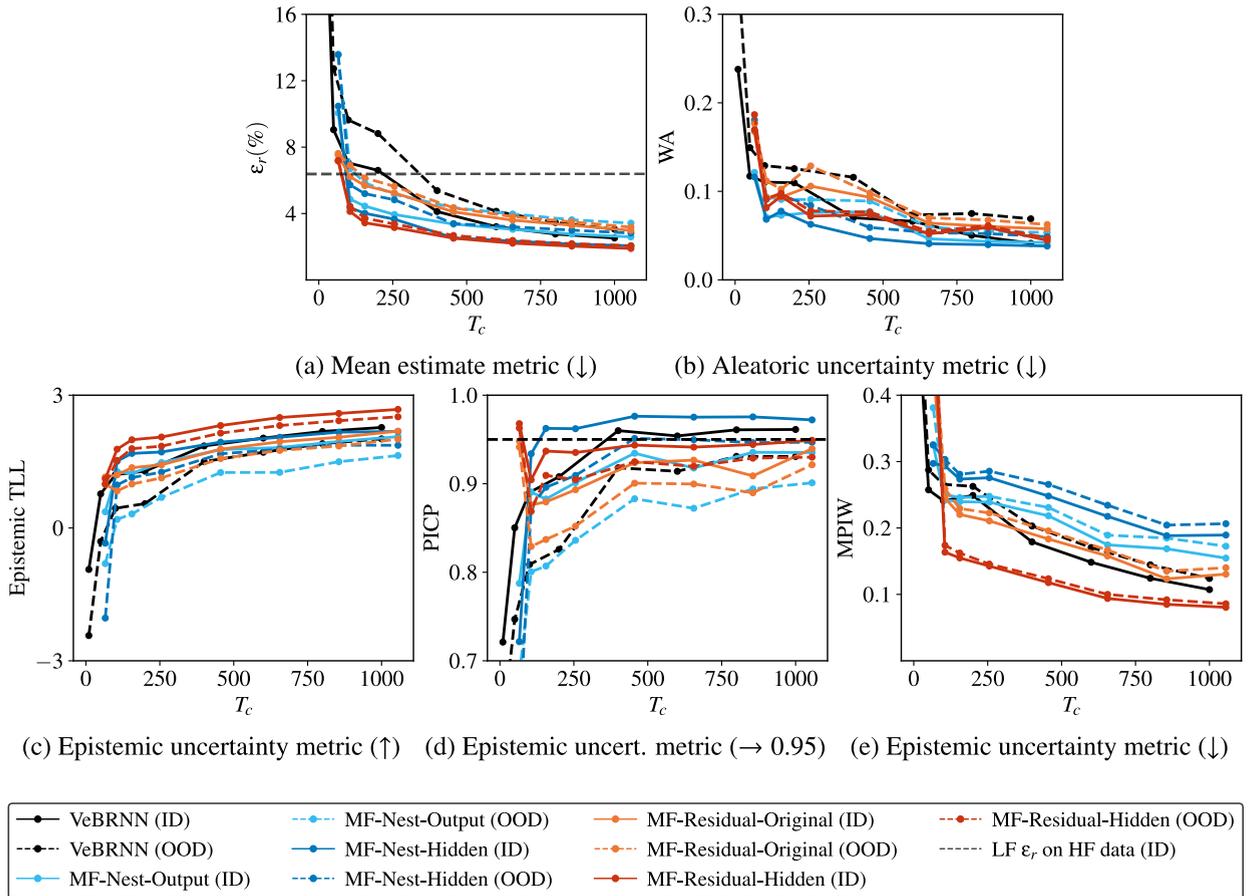
(a) Mean estimate metric ($\downarrow$)　　(b) Aleatoric uncertainty metric ($\downarrow$)

(c) Epistemic uncertainty metric ($\uparrow$)　(d) Epistemic uncert. metric ($\rightarrow 0.95$)　(e) Epistemic uncertainty metric ($\downarrow$)

VeBRNN (ID)	MF-Nest-Output (OOD)	MF-Residual-Original (ID)	MF-Residual-Hidden (OOD)
VeBRNN (OOD)	MF-Nest-Hidden (ID)	MF-Residual-Original (OOD)	LF ε_r on HF data (ID)
MF-Nest-Output (ID)	MF-Nest-Hidden (OOD)	MF-Residual-Hidden (ID)	

Fig. E6. Comparison of different multi-fidelity model configurations proposed in this paper for **Dataset 2** with RNN + VeBRNN architecture under low-fidelity of $N_{\text{cluster}} = 18$. (a) Mean estimate metric ε_r ($\downarrow$); (b) Aleatoric uncertainty metric WA ($\downarrow$); (c) Epistemic uncertainty metric TLL ($\uparrow$); (d) Epistemic uncertainty metric PICP ($\rightarrow 0.95$); (e) Epistemic uncertainty metric MPIW ($\downarrow$).

Appendix F. Additional experimental results

F.1. Predictions of RNN on the single-fidelity problem

The predictions of RNN with 2000 training paths obtained by $\widetilde{\text{(DNS)}}$ are given in Fig. F.7. RNN only predicts a mean estimation for each stress component, and it is clear to see the overfitting in the prediction of σ_{12}. In addition, it loses the essential information on uncertainty quantification, making it difficult to assess the overall quality of material properties.

F.2. Additional training results for dataset 3

Section 4.3 presents the VeBRNN + VeBRNN model for Dataset 3, where both fidelity levels are noisy (see also Fig. C3 for visualizing the noise from both fidelities). In this section, we present the results for the same dataset but considering two additional MF models: (1) a fully deterministic training MF model composed of RNN + RNN shown in Fig. F.8; and (2) a RNN + VeBRNN model where the HF prediction is Bayesian but the LF is deterministic (Fig. F.9). For this more challenging scenario, using a lower-quality LF model (i.e., $N_{\text{cluster}} = 3$) does not lead to a better MF model when compared to the single-fidelity case of an RNN trained solely on HF data. Even with a better LF model (i.e., $N_{\text{cluster}} = 18$), the benefits remain small, observing only marginal improvements within the range of $T_c = 20$ to $T_c = 80$.
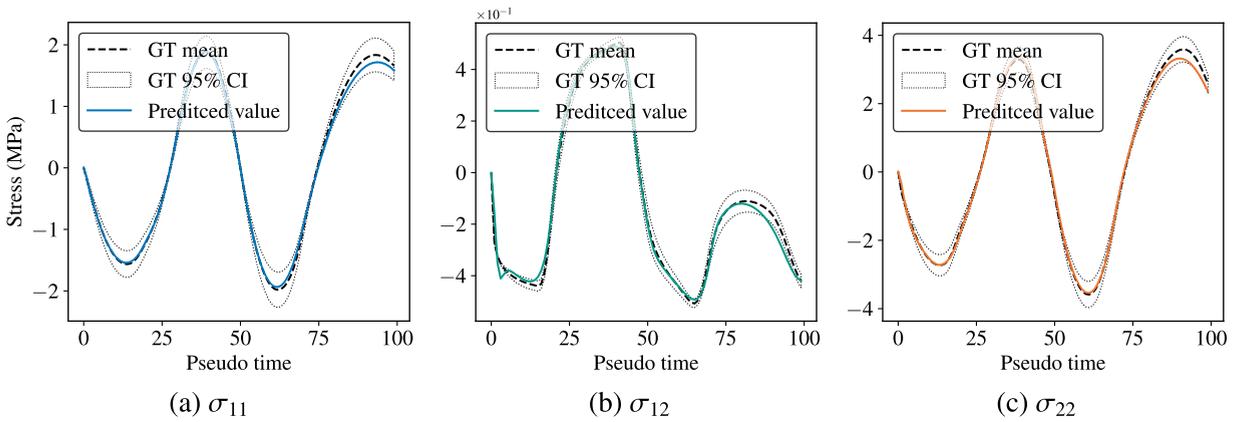

(a) σ_{11} (b) σ_{12} (c) σ_{22}

Fig. F.7. Predictions of the RNN with 2000 training paths ($T_c = 2000$) on the ID test path. (a) σ_{11}; (b) σ_{12}; (c) σ_{22}.

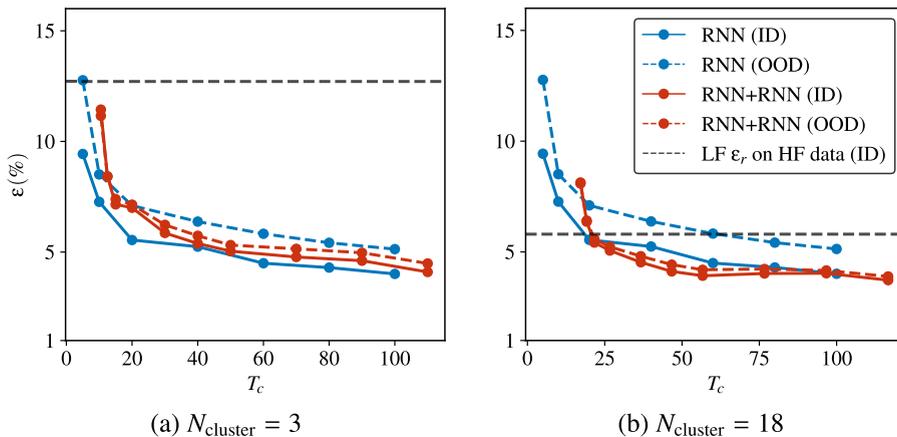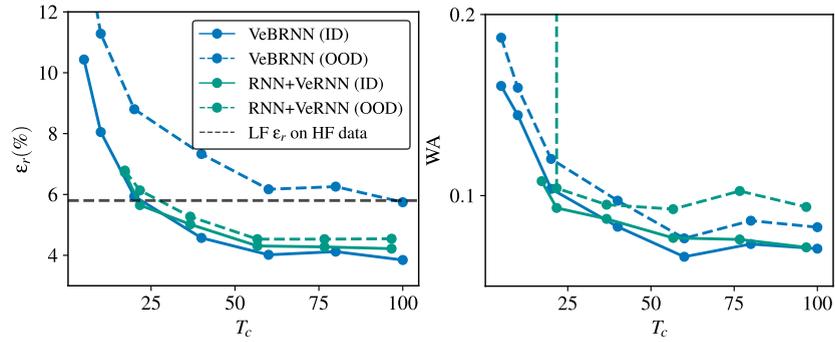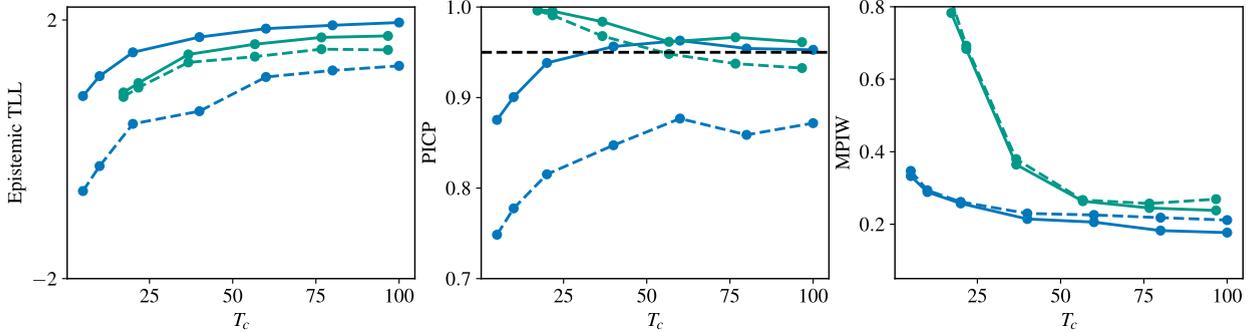(a) $N_{\text{cluster}} = 3$ (b) $N_{\text{cluster}} = 18$

Fig. F.8. Results for RNN + RNN model trained on Dataset 3 ($\widetilde{\text{ROM}} + \widetilde{\text{DNS}}$) compared to single-fidelity RNN trained on HF data. (a) LF model trained on data obtained by SCA of an SVE with $N_{\text{cluster}} = 3$; (b) LF model trained on data obtained by SCA of an SVE with $N_{\text{cluster}} = 18$.

(a) Mean estimate metric (↓)   (b) Aleatoric uncertainty metric (↓)

(c) Epistemic uncertainty metric (↑)   (d) Epistemic uncert. metric (→ 0.95)   (e) Epistemic uncertainty metric (↓)

Fig. F.9. Results for the RNN + VeBRNN model, when compared with a single-fidelity VeBRNN trained on HF data. The LF data is obtained by SCA of an SVE with $N_{\text{cluster}} = 18$. (a) Mean estimate metric ε_r (↓); (b) Aleatoric uncertainty metric WA (↓); (c) Epistemic uncertainty metric TLL (↑); (d) Epistemic uncertainty metric PICP (→ 0.95); (e) Epistemic uncertainty metric MPIW (↓).

Appendix G. Hyperparameter settings

For the hyperparameter settings, we draw on insights from prior studies that address similar problems in a deterministic setting [8, 55], and define our configurations as follows:

- **Architectures:**
  - **Mean network:** A two-layer GRU architecture with 128 hidden units per layer.
  - **Variance network:** A smaller GRU network with one layer and 8 hidden units.
- **Optimizer / Inference:**
  - We use *Adam* with a learning rate of 0.001 for 1000 epochs during deterministic training and the warm-up phase of the cooperative training strategy.
  - For variance network training:
    * Datasets with SVEs: learning rate 0.01, 4000 epochs.
    * Datasets with RVEs: learning rate 0.01, 10,000 epochs.
  - For posterior sampling via pSGLD, the learning rate is set to be 0.001; and we use a burn-in of 50 epochs and draw 100 posterior samples every 10 epochs, totaling 1050 epochs.
- **Training / Validation:**
  - For deterministic training, the dataset is split into 80 %/20 % for training and validation, respectively.
  - For cooperative training, no validation split is needed. We fix the iteration count to $K = 2$ for all experiments.

References

[1] A.I.E. Lagaris, D.I. Likas, Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, IEEE Trans. Neural Netw. 9 (5) (1998) 987–1000.
[2] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707.
[3] Y. Lecun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.
[4] J. Ghaboussi, J.H. Garrett, X. Wu, Knowledge-based modeling of material behavior with neural networks, J. Eng. Mech. 117 (1) (1991) 132–153.
[5] M.A. Bessa, R. Bostanabad, Z. Liu, A. Hu, D.W. Apley, C. Brinson, W. Chen, W.K. Liu, A framework for data-driven analysis of materials under uncertainty: countering the curse of dimensionality, Comput. Methods Appl. Mech. Eng. 320 (2017) 633–667.

[6] J.B. Le, Q.-C. Yvonnet, He, Computational homogenization of nonlinear elastic materials using neural networks, Int. J. Numer. Methods Eng. 104 (12) (2015) 1061–1084.

[7] Z. Liu, M.A. Bessa, W.K. Liu, Self-consistent clustering analysis: an efficient multi-scale scheme for inelastic heterogeneous materials, Comput. Methods Appl. Mech. Eng. 306 (2016) 319–341.

[8] M. Mozaffar, R. Bostanabad, W. Chen, K. Ehmann, J. Cao, M.A. Bessa, Deep learning predicts path-dependent plasticity, Proc. Nat. Acad. Sci. 116 (52) (2019) 26414–26420.

[9] L. Wu, D. Van, N. Nguyen, L.G. Kilingar, Noels, A recurrent neural network-accelerated multi-scale model for elasto-plastic heterogeneous materials subjected to random cyclic and non-proportional loading paths, Comput. Methods Appl. Mech. Eng. 369 (2020) 113234.

[10] X. He, J.-S. Chen, Thermodynamically consistent machine-learned internal state variable approach for data-driven modeling of path-dependent materials, Comput. Methods Appl. Mech. Eng. 402 (2022) 115348. A Special Issue in Honor of the Lifetime Achievements of J.

[11] M. Eghbalian, M. Pouragha, R. Wan, A physics-informed deep neural network for surrogate modeling in classical elasto-plasticity, Comput. Geotech. 159 (2023) 105472.

[12] K.A. Kalina, J. Brummund, W. Sun, M. Kästner, Neural networks meet anisotropic hyperelasticity: a framework based on generalized structure tensors and isotropic tensor functions, Comput. Methods Appl. Mech. Eng. 437 (2025) 117725.

[13] K.A.A. Jadoon, A. Meyer, J.N. Fuhg, Automated model discovery of finite strain elastoplasticity from uniaxial experiments, Comput. Methods Appl. Mech. Eng. 435 (2025) 117653.

[14] H. Vijayakumaran, J.B. Russ, G.H. Paulino, M.A. Bessa, Consistent machine learning for topology optimization with microstructure-dependent neural network material models, J. Mech. Phys. Solids 196 (2025) 106015.

[15] P. Bernardo, M.A. Ferreira, Bessa, Automatically differentiable model updating (adimu): conventional, hybrid, and neural network material model discovery including history-dependency, Technical Report, arXiv preprint, 2025.

[16] B.P. Ferreira, F.M.A. Pires, M.A. Bessa, Adaptivity for clustering-based reduced-order modeling of localized history-dependent phenomena, Comput. Methods Appl. Mech. Eng. 393 (2022) 114726.

[17] P. Bernardo, F.M. Ferreira, M.A.A. Pires, Bessa, Crate: a python package to perform fast material simulations, J. Open Source Softw. 8 (87) (2023) 5594.

[18] Y. Gal, R. Islam, Z. Ghahramani, Deep Bayesian active learning with image data, Proceedings of the 34th International Conference on Machine Learning 70 (PMLR) (2017) 6–11.

[19] J. Yi, J. Cheng, M.A. Bessa, Practical Multi-Fidelity Machine Learning: Fusion of Deterministic and Bayesian Models, 2024.

[20] J. Yi, M.A. Bessa, Cooperative Bayesian and Variance Networks Disentangle Aleatoric and Epistemic Uncertainties, 2025.

[21] G. Andrew, P. Wilson, Izmailov, Bayesian Deep Learning and a Probabilistic Perspective of Generalization, 33, Curran Associates, Inc, 2020.

[22] M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U.R. Acharya, V. Makarenkov, S. Nahavandi, A review of uncertainty quantification in deep learning: techniques, applications and challenges, Inf. Fusion 76 (2021) 243–297.

[23] A.F. Psaros, X. Meng, Z. Zou, L. Guo, G.E. Karniadakis, Uncertainty quantification in scientific machine learning: methods, metrics, and comparisons, J. Comput. Phys. 477 (2023) 111902.

[24] M.A. Bessa, P. Glowacki, M. Houlder, Bayesian machine learning in metamaterial design: fragile becomes supercompressible, Adv. Mater. 31 (48) (2019) 1904845.

[25] D. Shin, A. Cupertino, H.J. Matthijs, P.G.D. Jong, M.A. Steeneken, R.A. Bessa, Norte, Spiderweb nanomechanical resonators via bayesian optimization: inspired by nature and guided by machine learning, p. 2106248.

[26] Y. Wang, D.L. Mcdowell, Uncertainty quantification in materials modeling, in: Y. Wang, D.L. Mcdowell (Eds.), Uncertainty Quantification in Multiscale Materials Modeling, Elsevier Series in Mechanics of Advanced Materials, Woodhead Publishing, 2020, pp. 1–40.

[27] K.I. Christopher, C.E. Williams, Rasmussen, Gaussian Processes for Machine Learning, 2, MIT press, Cambridge, MA, Cambridge, MA, 2006.

[28] O. Ozbayram, A. Olivier, L. Graham-Brady, Heteroscedastic gaussian process regression for material structure-property relationship modeling, Comput. Methods Appl. Mech. Eng. 431 (2024) 117326.

[29] M. Radford, Neal, MCMC using Hamiltonian dynamics. Handbook of Markov chain Monte Carlo, 2, 2011.

[30] T. Papamarkou, M. Skoularidou, K. Palla, L. Aitchison, J. Arbel, D. Dunson, M. Filippone, P.V. Fortuin, Hennig, José Miguel Hernández-Lobato, et al. Position: Bayesian deep learning is needed in the age of large-scale ai, Technical Report, arXiv preprint, 2024.

[31] M. Radford, Neal, Bayesian Learning for Neural Networks, 118, Springer Science & Business Media, 2012.

[32] D.M. Blei, A. Kucukelbir, J.D. Mcauliffe, Variational inference: a review for statisticians, J. Am. Stat. Assoc. 112 (518) (2017) 859–877.

[33] Y. Gal, Z. Ghahramani, Dropout as a Bayesian approximation: representing model uncertainty in deep learning, Proceedings of The 33rd International Conference on Machine Learning 48 (PMLR) (2016) 1050–1059.

[34] A.B. Lakshminarayanan, C. Pritzel, Blundell, Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles, 30, Curran Associates, Inc, 2017.

[35] A. Olivier, M.D. Shields, L. Graham-Brady, Bayesian neural networks for uncertainty quantification in data-driven materials modeling, Comput. Methods Appl. Mech. Eng. 386 (2021) 114079.

[36] X. Zhang, K. Garikipati, Label-free learning of elliptic partial differential equation solvers with generalizability across boundary value problems, Comput. Methods Appl. Mech. Eng. 417 (2023) 116214.

[37] W. Bridgman, X. Zhang, G. Teichert, M. Khalil, K. Garikipati, R. Jones, A heteroencoder architecture for prediction of failure locations in porous metals using variational inference, Comput. Methods Appl. Mech. Eng. 398 (2022) 115236.

[38] G.D. Pasparakis, L. Graham-Brady, M.D. Shields, Bayesian neural networks for predicting uncertainty in full-field material response, 2024.

[39] K. Linka, G.A. Holzapfel, E. Kuhl, Discovering uncertainty: Bayesian constitutive artificial neural networks, Comput. Methods Appl. Mech. Eng. 433 (2025) 117517.

[40] P. Sotirios, Chatzis, Sparse bayesian recurrent neural networks, in: Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2015, Porto, Portugal, Springer, 2015, pp. 359–372. Proceedings, Part II 15.

[41] M. Fortunato, C. Blundell, O. Vinyals, Bayesian Recurrent Neural Networks, 2017.

[42] Z. Gan, C. Li, C. Chen, Y. Pu, Q. Su, L. Carin, Scalable Bayesian Learning of Recurrent Neural Networks for Language Modeling, 2017.

[43] D. Coscia, M. Welling, N. Demo, G. Rozza, Barnn, A bayesian autoregressive and recurrent neural network, Technical Report, arXiv preprint, 2025.

[44] C. Li, C. Chen, D. Carlson, L. Carin, Preconditioned Stochastic Gradient Langevin Dynamics for Deep Neural Networks, 2015.

[45] A. David, A.S. Nix, Weigend, Estimating the mean and variance of the target probability distribution, 1 of *1994 ieee international conference on neural networks (ICNN'94)*, IEEE, 1994.

[46] C. Fare, P. Fenner, M. Benatan, A. Varsi, E.O. Pyzer-Knapp, A multi-fidelity machine learning approach to high throughput materials screening, Comput. Mater. 8 (1) (2022) 257.

[47] C. Park, R.T. Haftka, N.H. Kim, Remarks on multi-fidelity surrogates, Struct. Multidiscip. Optim. 55 (3) (2017) 1029–1050.

[48] H. Liu, J. Cai, Y.-S. Ong, Remarks on multi-output Gaussian process regression, Knowl. Based Syst. 144 (2018) 102–121.

[49] X. Meng, G.E. Karniadakis, A composite neural network that learns from multi-fidelity data: application to function approximation and inverse pde problems, J. Comput. Phys. 401 (2020) 109020.

[50] S. Li, W. Xing, R. Kirby, S. Zhe, Multi-Fidelity Bayesian Optimization via Deep Neural Networks, 33, Curran Associates, Inc, 2020.

[51] J. Cheng, Q. Lin, J. Yi, An enhanced variable-fidelity optimization approach for constrained optimization problems and its parallelization, Struct. Multidiscip. Optim. 65 (7) (2022) 188.

[52] I.J. Alexander, A. Forrester, A.J. Sóbester, Keane, Multi-fidelity optimization via surrogate modelling, Proc. Royal Soc. A: Math. Phys. Eng. Sci. 463 (2007) 3251–3269.

[53] X. Meng, H. Babaee, G.E. Karniadakis, Multi-fidelity bayesian neural networks: algorithms and applications, J. Comput. Phys. 438 (2021) 110361.

[54] P. Conti, M. Guo, A. Manzoni, J.S. Hesthaven, Multi-fidelity surrogate modeling using long short-term memory networks, Comput. Methods Appl. Mech. Eng. 404 (2023) 115811.

[55] A. Dekhovich, O.T. Turan, J. Yi, M.A. Bessa, Cooperative data-driven modeling, Comput. Methods Appl. Mech. Eng. 417 (2023) 116432.

[56] D. Sameer, S. Ghosh, T.D. Nguyen, T. Broderick, Are you using test loglikelihood correctly?, 2024.

[57] L.V. Kantorovich, Mathematical methods of organizing and planning production, Manage. Sci. 6 (4) (1960) 366–422.

[58] L. Sluijterman, E. Cator, T. Heskes, How to evaluate uncertainty estimates in machine learning for regression?, Neural Netw. 173 (2024) 106203.

[59] P. Diederik, J. Kingma, Ba, Adam: A Method for Stochastic Optimization, 2014.

[60] Murphy, Probabilistic Machine Learning: Advanced Topics, 2022.

[61] M. Welling, Y.W. Teh, Bayesian learning via stochastic gradient langevin dynamics, in: Proceedings of the 28th International Conference on Machine Learning (ICML-11), the 28th international conference on machine learning (ICML-11), 2011, pp. 681–688.

[62] S. Ruder, An Overview of Gradient Descent Optimization Algorithms, Technical Report, arXiv preprint, 2016.

[63] R. Hill, Elastic properties of reinforced solids: some theoretical principles, J. Mech. Phys. Solids 11 (5) (1963) 357–372.

[64] J. Yi, M.A. Bessa, Rvesimulator: an automated representative volume element simulator for data-driven material discovery, in: AI for Accelerated Materials Design-NeurIPS, 2023.

[65] P. Bernardo, F.M. Ferreira, M.A.A. Pires, Bessa, Adaptive clustering-based reduced-order modeling framework: Fast and accurate modeling of localized history-dependent phenomena, 2021.

[66] L.V. Jospin, H. Laga, F. Boussaid, W. Buntine, M. Bennamoun, Hands-on bayesian neural networks-a tutorial for deep learning users, IEEE Comput. Intell. Mag. 17 (2) (2022) 29–48.