Multi-Fidelity Bayesian machine learning with uncertainty disentanglement for material modeling and design

Yi, J.

DOI
10.4233/uuid:831ab16c-ccfe-486b-8e3a-fb3b5efbd3f2

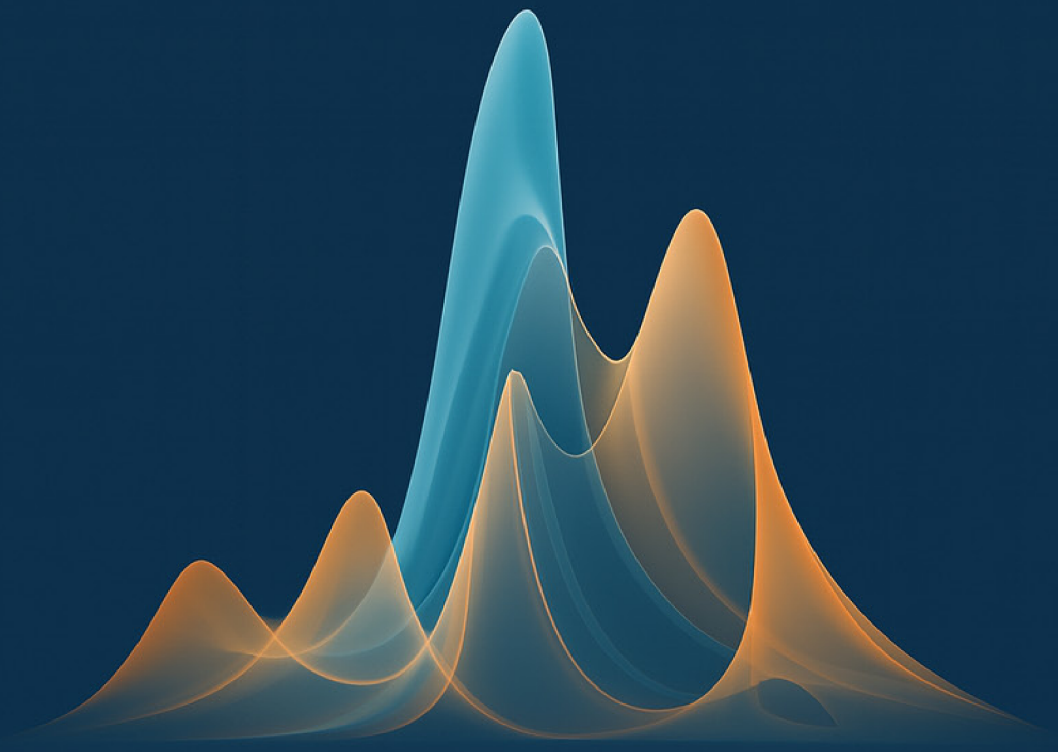
Publication date
2026

Document Version
Final published version

Citation (APA)
Yi, J. (2026). *Multi-Fidelity Bayesian machine learning with uncertainty disentanglement for material modeling and design*. [Dissertation (TU Delft), Delft University of Technology]. Delft University of Technology. https://doi.org/10.4233/uuid:831ab16c-ccfe-486b-8e3a-fb3b5efbd3f2

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

MULTI-FIDELITY BAYESIAN MACHINE LEARNING WITH UNCERTAINTY DISENTANGLEMENT FOR MATERIAL MODELING AND DESIGN



Jiaxiang YI

MULTI-FIDELITY BAYESIAN MACHINE LEARNING WITH UNCERTAINTY DISENTANGLEMENT FOR MATERIAL MODELING AND DESIGN

MULTI-FIDELITY BAYESIAN MACHINE LEARNING WITH UNCERTAINTY DISENTANGLEMENT FOR MATERIAL MODELING AND DESIGN

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology,
by the authority of the Rector Magnificus, Prof. dr. ir. T.H.J.J. van der Hagen,
chair of the Board for Doctorates,
to be defended publicly on
Wednesday 7 January 2026 at 17:30

by

Jiaxiang YI

Master of Science in Naval Architecture and Ocean Structure,
Huazhong University of Science and Technology, Wuhan, China,
born in Jiangxi, China.

This dissertation has been approved by the promotors.

Composition of the doctoral committee:

Rector Magnificus,	Chairperson
Dr. ir. M.H.F. Sluiter	Delft University of Technology, *promotor*
Dr. M.A. Bessa	Brown University, United States, *promotor*
Dr. B. Caglar	Delft University of Technology, *copromotor*

Independent members:

Prof. dr. ir. L.J. Sluijs	Delft University of Technology
Dr. D. Zarouchas	Delft University of Technology
Dr. Y. Li	Imperial College London, United Kingdom
Prof. dr. K. Garikipati	University of Southern California, United States
Prof. dr. J. Dik	Delft University of Technology, reserve member

This research was carried out in the Faculty of Mechanical Engineering, Delft University of Technology.
This project was financially supported by China Scholarship Council(CSC).

Keywords: Bayesian deep learning, Uncertainty quantification and disentanglement, Multi-fidelity data-driven modeling, Constitutive modeling, Uncertainty-aware inverse design.

Printed by: Proefschriftenprinten.nl

Front & Back: Jiaxiang Yi designed using Midjourney and ChatGPT.

ISBN 978-90-836415-3-9

An electronic version of this dissertation is available at
http://repository.tudelft.nl/.

CONTENTS

SUMMARY

Machine learning delivers strong predictive performance in scientific and engineering tasks when high-fidelity data are abundant. Yet, real-world models seldom quantify aleatoric (data) and epistemic (model) uncertainties, leading to overfitting on noisy inputs. In addition, collecting adequate high-fidelity data is often expensive or infeasible, whereas low-fidelity data are more accessible but less reliable. To address these challenges, this thesis proposes a general multi-fidelity Bayesian learning framework that enables trustworthy uncertainty disentanglement, and extends its application to constitutive modeling and design of recycled composite materials.

The thesis begins with an introduction to regression from a Bayesian perspective, establishing the connection between deterministic and probabilistic treatments, and covering models from linear to kernel and deep neural network regressions. This chapter provides the theoretical foundation for the remainder of the thesis by formalizing inference techniques, uncertainty quantification strategies, and model evaluation metrics.

Two core machine learning methods are developed to tackle the challenges of uncertainty estimation and multi-fidelity data fusion. First, a cooperative training scheme is proposed that combines a variance estimation neural network with a Bayesian mean neural network, enabling explicit disentanglement of aleatoric and epistemic uncertainties while improving mean prediction. The approach demonstrates strong scalability and generality across tasks and network architectures. Second, a practical multi-fidelity Bayesian learning framework is introduced, which fuses low- and high-fidelity data via a deterministic model, a transfer-learning module, and a Bayesian residual learner. This architecture balances expressiveness and computational efficiency, yielding robust predictions in both data-scarce and data-rich regimes.

To support data-driven mechanics, the proposed framework is extended to generalized constitutive modeling of history-dependent materials. A hierarchical learning scheme is developed that spans from single-fidelity deterministic networks to multi-fidelity Bayesian recurrent neural networks. This addresses two major limitations in data-driven modeling: the reliance on large, clean datasets and the lack of interpretability in neural network predictions.

Finally, the methodology is applied to the sustainable design of recycled composite polymers, where uncertainty arises from microstructural variability and the use of compatibilizers. Leveraging the cooperative training framework, both aleatoric and epistemic uncertainties are quantified, and a novel polymer design is optimized to achieve better expected performance and lower data variation.

Together, this thesis presents a unified and scalable framework for Bayesian learning under uncertainty and multi-fidelity conditions, with broad applicability in scientific computing, materials modeling, and sustainable engineering. The thesis concludes by summarizing key findings, discussing current limitations, and outlining future research directions.

SAMENVATTING

Machine learning levert sterke voorspellende prestaties bij wetenschappelijke en technische taken wanneer er voldoende hoog-fideliteitsdata beschikbaar zijn. In de praktijk kwantificeren modellen echter zelden de aleatorische (data) en epistemische (model) onzekerheden, waardoor ze overfitten op rumoerige invoer. Bovendien is het verzamelen van hoog-fideliteitsdata vaak duur of onhaalbaar, terwijl laag-fideliteitsdata weliswaar ruimer voorhanden maar minder betrouwbaar zijn. Om deze uitdagingen het hoofd te bieden, stelt dit proefschrift een algemeen multifideliteits Bayesian leerraamwerk voor dat betrouwbare ontkoppeling van onzekerheden mogelijk maakt en wordt toegepast op constitutief modelleren en het ontwerp van gerecyclede composietmaterialen.

Het proefschrift begint met een inleiding tot regressie vanuit een Bayesian perspectief, waarin de verbinding wordt gelegd tussen deterministische en probabilistische benaderingen en modellen worden besproken van lineaire tot kernel- en deep-neural-network-regressies. Dit hoofdstuk vormt het theoretische fundament voor de rest van het proefschrift door inferentietechnieken, strategieën voor onzekerheidskwantificatie en evaluatiematen te formaliseren.

Vervolgens worden twee kernmethoden ontwikkeld voor onzekerheidsschatting en multifideliteits datafusie. Ten eerste wordt een coöperatief trainingsschema gepresenteerd dat een variantienetwerk combineert met een Bayesian meannetwerk, waardoor aleatorische en epistemische onzekerheden expliciet worden ontkoppeld en de gemiddelde voorspelling verbetert. De methode blijkt schaalbaar en algemeen toepasbaar voor uiteenlopende taken en netwerken. Ten tweede wordt een praktisch multifideliteits Bayesian-raamwerk geïntroduceerd dat laag- en hoog-fideliteitsdata samenvoegt via een deterministisch basismodel, een transfer-learning-module en een Bayesian residueleerder. Deze architectuur balanceert expressiviteit en rekenefficiëntie en levert robuuste voorspellingen in zowel data-schaarse als data-rijke scenario's.

Voor data-gedreven mechanica wordt het raamwerk uitgebreid naar gegeneraliseerd constitutief modelleren van geschiedenisafhankelijke materialen. Een hiërarchisch leerschema bestrijkt daarbij het spectrum van enkel-fideliteits deterministische netwerken tot multifideliteits Bayesian recurrente neurale netwerken. Zo worden twee grote beperkingen van data-gedreven modellering aangepakt: de noodzaak van grote, schone datasets en het gebrek aan interpretatiemogelijkheden van neurale netwerken.

Ten slotte wordt de methodologie toegepast op het duurzame ontwerp van gerecyclede composietpolymeren, waarbij onzekerheid voortkomt uit microstructurele variabiliteit en het gebruik van compatibilisatoren. Door gebruik te maken van het coöperatieve trainingskader worden zowel aleatorische als epistemische onzekerheden gekwantificeerd, en wordt een nieuw polymeerontwerp geoptimaliseerd om betere verwachte prestaties en een lagere datavariatie te bereiken.

Al met al presenteert dit proefschrift een verenigd en schaalbaar raamwerk voor Bayesian leren onder onzekerheid en multifideliteitscondities, met brede toepasbaar-

heid in wetenschappelijke computing, materiaalmodellering en duurzame engineering. Het besluit met een samenvatting van de belangrijkste bevindingen, bespreekt huidige beperkingen en schetst richtingen voor toekomstig onderzoek.

1

INTRODUCTION

This research focuses on three primary topics: (1) uncertainty quantification and disentanglement in data-driven modeling approaches; (2) multi-fidelity modeling that integrates data from sources with varying fidelity levels; (3) their application to complex material modeling and design problems, such as the ones considering plasticity or failure. This chapter provides an overview of the research background and its associated challenges, followed by a discussion of the existing knowledge gaps and an outline of the thesis structure.

1

1.1. MOTIVATION

Machine learning (ML) methods have become ubiquitous in addressing scientific problems, particularly with the recent advances in large language models (LLMs) [1]. ML methods can be categorized into three types, which are supervised learning, unsupervised learning, and reinforcement learning, according to the different nature of tasks [2, 3]. Specifically, supervised learning aims to learn the mapping relations between the inputs and labeled outputs, such as regression and classification problems. In contrast, unsupervised learning often works with unlabeled data, which seeks to discover its hidden structures or patterns, like clustering and representation learning. Reinforcement learning is based on game theory [4], which learns optimal actions with environmental interactions, usually applied to control problems and decision-making.

In this dissertation, I primarily focus on supervised learning, with a particular emphasis on regression problems. In this regard, various neural architectures have been developed for different data types. For instance, Deep neural networks (DNNs) architectures like Convolutional neural networks (CNNs) are effective for images [6], while Recurrent neural networks (RNNs) are designed for time-series data [7]. Recently, diffusion models [8], Transformers[9], and other advanced architectures have been developed to further enhance the performance of ML on complex and larger-scale tasks.

Generally, when ML methods are employed to assist in the design and modeling of scientific problems, the approach is referred to as data-driven modeling. In this context, a unified framework for data-driven design and analysis of materials was proposed in [10]. This framework consists of three general steps: (1) the design of experiments to define the input space; (2) computational analyses to evaluate the corresponding outputs, thereby constructing a database; and (3) the application of ML methods to this database for surrogate modeling, optimization, or other downstream tasks. Thereby, data-driven modeling has emerged as the *de facto* paradigm for tackling complex, nonlinear, and computationally intensive problems in material modeling and design [11, 12], leveraging the expressive power of machine learning models by the universal approximation theorem [13].

Despite the success of data-driven modeling, several important challenges remain overlooked, particularly in uncertainty quantification and the fusion of data sources with varying cost, accuracy, and noise. In other words, prevalent ML models are often trained deterministically on large datasets, which may lead to overfitting and a lack of interpretability. Moreover, scientific problems, for instance, material design and analysis, inherently have data from different sources such as experiments, simulations, or empirical models. These challenges motivate the development of novel ML methods from a Bayesian perspective aiming for stable uncertainty quantification and scalable adaptation to multi-fidelity datasets.

1.2. KNOWLEDGE GAPS AND RESEARCH QUESTIONS

Although ML methods have been widely applied to data-driven modeling, they are often trained in a deterministic manner. Unfortunately, deterministic models can be prone to overfitting, particularly when in the presence of noisy data [14], and they cannot quantify uncertainty [15–17]. Uncertainties typically arise from two sources: (1) *Aleatoric*

uncertainty, also known as data uncertainty, refers to the inherent variability in the data generation process. This type of uncertainty is irreducible, even with an infinite amount of data. (2) *Epistemic uncertainty*, in contrast, stems from a lack of knowledge about the model parameters or structure. It is therefore reducible and can be mitigated by incorporating more data or adopting better modeling strategies [18, 19]. In the literature, there are numerous methods for quantifying uncertainty, but there is no consensus on how to quantify and disentangle both types of uncertainties. Deterministic models such as mean and variance estimation networks [20] can predict both mean and aleatoric uncertainty, despite some training difficulties that are investigated and solved in this thesis. Probabilistic models like Bayesian neural networks [17, 21–23] treat parameters as random variables and are capable of estimating epistemic uncertainty. However, as I will also discuss later, inference for Bayesian neural networks is not trivial. Therefore, creating a scalable model that is capable of estimating mean, aleatoric uncertainty and epistemic uncertainty simultaneously remains a challenge.

Knowledge Gap 1

To the best of my knowledge, a unified and scalable method that can disentangle *aleatoric* and *epistemic* uncertainties is lacking in the machine learning literature.

Research question 1

Considering the limitations of existing methods that separately estimate *aleatoric* or *epistemic* uncertainty, is it possible to create a unifying method that simultaneously quantifies both types of uncertainty?

Another important need that arises in data-driven design and analysis is related to the quality and availability of data. Advances in experimental techniques, high-throughput simulations, and automated data processing are facilitating data acquisition, leading to larger datasets that include varying degrees of fidelity [10, 24]. This follows from the inevitable trade-off of achieving faster (slower) data acquisition when considering lower (higher) fidelity data. Therefore, practical datasets tend to have various levels of fidelity. Low-fidelity data are inexpensive to generate but often contain bias and considerable noise. In contrast, high-fidelity data are more accurate and less noisy, but also more costly to obtain [25]. Conventional data-driven methods are typically restricted to using single-fidelity datasets and lack the capability to integrate information across fidelities. Recently, important efforts have been made to consider multi-fidelity data-driven approaches [25, 26]. These methods exploit low-fidelity data to capture the general trend or underlying structure of the target function, while leveraging high-fidelity data to refine the predictions and improve accuracy [27–29].

Knowledge Gap 2

Existing Bayesian multi-fidelity methods are limited to low-dimensional or small-scale problems, while current scalable approaches rely on deterministic training and cannot quantify uncertainty. A method that is both scalable and uncertainty-aware for multi-fidelity machine learning has remained illusive.

Research question 2

Can a Bayesian machine learning method be developed to become scalable and integrate multi-fidelity data while leading to reliable predictions that include uncertainty estimation?

In the field of data-driven mechanics, particularly for material modeling, the use of advanced deterministic ML methods has become increasingly prevalent [10, 12]. The majority of existing studies focus on relatively simple constitutive behaviors, such as elasticity [30] or hyperelasticity [31], but more complex phenomena such as plasticity or failure has also been achieved [34]. The latter phenomena are inherently nonlinear and history-dependent, making the stress response depend not only on the strain input of the current state, but also on the previous deformation path. As a result, the learning task becomes a sequence-to-sequence regression problem amenable to models with recurrent units like RNNs [34–36]. Until now, and regardless of the problem of interest being history-dependent or independent, the literature has focused almost exclusively on deterministic models that lack uncertainty quantification. Yet, uncertainty-aware models raise trustworthiness in the predictions that are made, and they can help avoid overfitting when data is noisy.

Knowledge Gap 3

Mechanics of materials (and many other) problems often require recurrent neural network architectures (sequence-to-sequence modeling), while involving multi-fidelity data and aleatoric uncertainty. Furthermore, model predictions need to be uncertainty-aware to improve trustworthiness, i.e., epistemic uncertainty needs to be estimated. However, this would imply the development and training of Bayesian recurrent neural networks that learn history-dependency, disentangle *aleatoric* and *epistemic* uncertainties, and that are capable of integrating multi-fidelity data. This has not been demonstrated in the literature.

Research Question 3

How can a Bayesian recurrent learning network be extended to model the history-dependent behavior of materials, while enabling multi-fidelity integration and disentanglement of *aleatoric* and *epistemic* uncertainties?

Beyond learning material behavior, one of the ultimate goals in data-driven mechanics

is to design or optimize materials with superior performance. This is a natural extension once a forward model has been accurately trained, and is commonly referred to as inverse design [37, 38]. However, existing approaches are typically formulated in a deterministic manner, where uncertainties are not accounted for during the optimization process [39]. In other words, they aim to maximize or minimize target properties without considering variability in material responses or model approximation errors [39, 40]. As a result, the generated designs may be overconfident or suboptimal when deployed in real-world scenarios [41, 42].

Knowledge Gap 4

Robust material design implies the ability to design under uncertain material behavior. Existing deterministic design methods are limited in this setting. A Bayesian inverse design framework is missing in this context, and it would benefit from the ability to disentangle *aleatoric* and *epistemic* uncertainties.

Research Question 4

Can uncertainty-aware inverse design methods be developed such that robust material design becomes possible in practice?

1.3. OUTLINE

As discussed in this dissertation, a new method called variance estimation Bayesian neural networks is proposed, enabling the disentanglement of aleatoric and epistemic uncertainties (addressing **RQ1**). This contribution is followed by a unified multi-fidelity machine learning framework that handles datasets with different fidelities and quantifies uncertainties at each desired fidelity (addressing **RQ2**). Building upon these two contributions, I apply them to model history-dependent plasticity material behavior, generalizing training from single- to multi-fidelity settings and including uncertainty quantification disentanglement (addressing **RQ3**). Finally, the developed methods are applied to uncertainty-aware inverse material design of sustainable polymer composites, with a particular focus on failure-related properties (**RQ4**). The outline of the dissertation is as follows:

- Chapter 2 introduces the foundations of Bayesian machine learning and highlights its distinctions and connections to deterministic approaches for regression problems, ranging from simple linear models to deep neural networks. It is intended to serve an educational purpose for readers who may be less familiar with probabilistic machine learning.

- Chapter 3 presents a novel cooperative training algorithm for variance estimation and Bayesian neural networks. The method is easy to implement and adaptable to various neural architectures. Most importantly, it enables the disentanglement of *aleatoric* and *epistemic* uncertainties, advancing the frontier of Bayesian machine

learning and laying a theoretical foundation for subsequent applications in material modeling and design.

- Chapter 4 proposes a practical multi-fidelity machine learning framework, which unifies the topic with a general formula. In this framework, a deterministic model is employed for the low-fidelity data to exploit its computational efficiency, while a Bayesian model is adopted for the high-fidelity data to enable accurate prediction with uncertainty quantification.
- Chapter 5 generalizes the methods developed in Chapter 3 and Chapter 4 to history-dependent material modeling. In particular, a Bayesian recurrent neural network is formulated to address sequence-to-sequence regression tasks arising in path-dependent constitutive laws, such as plasticity. This extension supports both single- and multi-fidelity learning with disentangled uncertainty quantification.
- Chapter 6 applies the cooperative training strategy from Chapter 3 to learn the failure behavior of sustainable polymer composite materials. Furthermore, an uncertainty-aware inverse design strategy is followed to discover a sustainable composite with improved performance while reducing the intrinsic material uncertainty.
- Chapter 7 concludes the dissertation by summarizing the main findings and contributions; then outlines potential directions for future research.
- In Appendix A, an open-source platform is introduced for automated computational analysis of materials across different fidelity levels. This tool not only serves as the primary data generator for the case studies presented in this dissertation, but also provides a foundation for future research in data-driven material modeling.

Overall, this dissertation aims to bridge the gap between Bayesian learning and complex material design and analysis tasks by developing novel ML methodologies that are theoretically sound and practically applicable.

2

Bayesian Perspectives on Regression: From Linear Models to Deep Neural Networks

This chapter introduces the foundations of machine learning techniques for regression tasks, ranging from the simplest linear models to deep neural networks. For each model with increasing complexity, I first present the deterministic formulation and then introduce its corresponding Bayesian counterpart, in order to highlight the key differences and underlying connections between the two paradigms. Then, inference methods are introduced, which are essential components of Bayesian neural networks.

2.1. INTRODUCTION

This chapter introduces machine learning from a Bayesian perspective, providing a fundamental background for the subsequent chapters. In short, Bayesian machine learning is based on Bayes' rule [2]:

$$p(\theta|y=\mathscr{D}_y) = \frac{p(y=\mathscr{D}_y|\theta)\,p(\theta)}{p(y=\mathscr{D}_y)} \tag{2.1}$$

- $p(\theta)$: prior distribution, representing prior knowledge about the problem.
- $p(y=\mathscr{D}_y|\theta)$: likelihood function, indicating how well θ explain the data.
- $p(y=\mathscr{D}_y)$: marginal likelihood, obtained by integrating over θ.
- $p(\theta|y=\mathscr{D}_y)$: posterior, our updated belief about θ.

I first introduce parametric linear models, focusing on Bayesian linear regression, which yields closed-form posteriors for model parameters. The discussion then extends to nonparametric kernel-based models, particularly Gaussian Process Regression (GPR), which defines priors in function space and also allows closed-form inference with uncertainty quantification. Finally, I consider Bayesian Neural Networks (BNNs), which are highly expressive but require approximate inference due to intractable posteriors. Across these models, I emphasize the connections and distinctions between Bayesian models and their frequentist (deterministic) counterparts, highlighting how the Bayesian perspective introduces principled mechanisms for uncertainty quantification. The chapter concludes with an overview of common posterior approximation methods, including variational inference and Markov chain Monte Carlo.

2.2. LINEAR REGRESSION: FROM FREQUENTIST TO BAYESIAN

Consider a one-dimensional analytical problem with inputs and targets $\mathscr{D} = \{x_i, y_i\}_{i=1}^N$. The regression problem can be formulated as:

$$y = f(x) + \varepsilon, \tag{2.2}$$

where $f(x)$ denotes the underlying noiseless ground truth function (expected mean), and $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ is the corresponding noise (data noise, aleatoric uncertainty).

2.2.1. LINEAR REGRESSION

Linear regression assumes the prediction has the following format:

$$\hat{y} = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(x), \tag{2.3}$$

where $\boldsymbol{\theta} := \boldsymbol{w}$, $\boldsymbol{w} \in \mathbb{R}^D$ is a column-vector that contains the weights, and $\phi(x) \in \mathbb{R}^D$ is the input feature mapping column-vector: $\boldsymbol{\phi}(x) = \left[1, x, x^2, ..., x^{D-1}\right]^{\mathrm{T}}$, where D is the order of the polynomial (*hyperparameter*). Under a deterministic setup, the target of linear

regression is to find a set of parameters that best fits the data by minimizing the sum of mean squared errors (MSE) and regularization loss[1] :

$$\mathscr{L}_2(\boldsymbol{w}) = \sum_{i=1}^{N} \left(y_i - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(x_i)\right)^2 + \lambda \boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}, \tag{2.4}$$

where the first term is the MSE, and the second term is the regularization term to avoid overfitting. $\lambda > 0$ is the penalty term (*hyperparameter*). Equation (2.4) can be further written in matrix form as

$$\mathscr{L}_2(\boldsymbol{w}) = \left\|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{w}\right\|_2^2 + \lambda \left\|\boldsymbol{w}\right\|_2^2, \tag{2.5}$$

where $\boldsymbol{y} \in \mathbb{R}^N$ is the vector of target values, and $\boldsymbol{\Phi} \in \mathbb{R}^{N\times D}$. By taking the derivative of $\boldsymbol{w}$

$$\frac{\partial \mathscr{L}_2}{\partial \boldsymbol{w}} = 2\left(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} + \lambda \boldsymbol{I}_N\right)\boldsymbol{w} - 2\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y}, \tag{2.6}$$

Setting it to zero, the closed-form solution of $\boldsymbol{w}$ is

$$\boldsymbol{w}^* = \left(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi} + \lambda \boldsymbol{I}_N\right)^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y}. \tag{2.7}$$

2.2.2. BAYESIAN LINEAR REGRESSION

Bayesian linear regression aims to model the uncertainty over the parameters $\boldsymbol{w}$ by imposing a prior distribution. Usually, the likelihood is assumed to be a Gaussian distribution. For each observation (x_i, y_i), the likelihood is:

$$p(y_i \mid \boldsymbol{w}, x_i) = \mathscr{N}(y_i \mid \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(x_i), \sigma^2), \tag{2.8}$$

where the mean $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(x_i)$ follows the same expression as Equation (2.3), which exhibits the expected output given x_i and current weights, the second term is the variance of the Gaussian distribution, which represents the assumed observation noise level [2]. Thus, for the entire dataset, the likelihood can be expressed in matrix form as

$$p(\boldsymbol{y} \mid \boldsymbol{w}, \boldsymbol{\Phi}) = \mathscr{N}(\boldsymbol{y} \mid \boldsymbol{\Phi}\boldsymbol{w}, \sigma^2 \boldsymbol{I}_N), \tag{2.9}$$

Assuming the weight prior to be another Gaussian distribution, expressed as

$$p(\boldsymbol{w}) = \mathscr{N}(\boldsymbol{w} \mid \hat{\boldsymbol{w}}, \hat{\boldsymbol{\Sigma}}), \tag{2.10}$$

The posterior can be obtained by Equation (2.1), written by

$$p(\boldsymbol{w} \mid \mathscr{D}, \sigma^2) \propto \mathscr{N}(\boldsymbol{w}|\hat{\boldsymbol{w}}, \hat{\boldsymbol{\Sigma}})\mathscr{N}(\boldsymbol{y}|\boldsymbol{\Phi}\boldsymbol{w}, \sigma^2 \boldsymbol{I}_N) = \mathscr{N}(\boldsymbol{w} \mid \tilde{\boldsymbol{w}}, \tilde{\boldsymbol{\Sigma}}), \tag{2.11}$$

where

$$\tilde{\boldsymbol{w}} = \tilde{\boldsymbol{\Sigma}}\left(\hat{\boldsymbol{\Sigma}}^{-1}\hat{\boldsymbol{w}} + \frac{1}{\sigma^2}\boldsymbol{\Phi}^{\top}\boldsymbol{y}\right), \quad \tilde{\boldsymbol{\Sigma}} = \left(\hat{\boldsymbol{\Sigma}}^{-1} + \frac{1}{\sigma^2}\boldsymbol{\Phi}^{\top}\boldsymbol{\Phi}\right)^{-1}. \tag{2.12}$$

[1] Note that the presented loss function in Equation (2.4) also refers to ridge regression [2] since the regularization term is included.

[2] The noise is considered to be known in this section for the completeness of the derivation

The posterior predictive distribution is also a Gaussian distribution, obtained by:

$$\begin{aligned} p(y \mid \mathbf{x}, \mathcal{D}, \sigma^2) &= \int \mathcal{N}(y \mid \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}), \sigma^2) \mathcal{N}(\boldsymbol{w} \mid \tilde{\boldsymbol{w}}, \tilde{\boldsymbol{\Sigma}}) \, \mathrm{d}\boldsymbol{w} \\ &= \mathcal{N}\left(y \mid \tilde{\boldsymbol{w}}^\top \boldsymbol{\phi}(\boldsymbol{x}), \sigma^2 + \boldsymbol{\phi}(\boldsymbol{x})^\top \tilde{\boldsymbol{\Sigma}} \boldsymbol{\phi}(\boldsymbol{x})\right) \end{aligned} \tag{2.13}$$

where $\tilde{\boldsymbol{w}}^\top \boldsymbol{\phi}(\boldsymbol{x})$ is the predicted mean, $\boldsymbol{\phi}(\boldsymbol{x})^\top \tilde{\boldsymbol{\Sigma}} \boldsymbol{\phi}(\boldsymbol{x})$ is the epistemic uncertainty of the model parameter, and σ^2 is the known aleatoric uncertainty.

Remark 2.2.1. *In practice, it is challenging to specify an informative prior. A common choice is to assume a Gaussian prior with zero mean and covariance $\hat{\Sigma} = \tau^2 \boldsymbol{I}_N$. Under this setting, the posterior mean coincides with the solution in Equation (2.7) when setting $\lambda = \frac{\sigma^2}{\tau^2}$. Hence, the linear regression model introduced in Section 2.2.1 can be interpreted as a point estimate of the Bayesian linear regression.*

Roles of different components of Bayesian linear regression According to Bayes' rule, the prior distribution and likelihood are the primary factors influencing the performance of Bayesian linear regression. The prior distribution is the knowledge about the problem on the model parameters; however, it is difficult to obtain exact information with a pure data-driven modeling process. As a result, it is a common practice to set it to be a unit Gaussian distribution. As for the likelihood, it is a function of θ that maps the inputs to the observations. Furthermore, the likelihood is dominated by data once the function is defined.

A simple linear model with two parameters taken from [43], $y = w_1 x + w_0$ where $w_1 = 0.5$ and $w_0 = -0.3$, is adopted to illustrate how the data influence model parameter posterior distribution and predictive posterior distribution. The observation contains white noise generated by a Gaussian distribution with zero mean and standard deviation of 0.2. In this example, the data noise (aleatoric uncertainty) is known, and hence the $\sigma^2 = 0.2^2$ in Equation (2.9). As discussed previously, the prior distributions are two independent Gaussian distributions, leading to:

$$p(\boldsymbol{w}) = \mathcal{N}\left(\boldsymbol{w} \middle| \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right), \tag{2.14}$$

Figure 2.1 presents the outcomes of Bayesian learning under varying data sizes. The first column depicts the likelihood function, the second column shows the prior or posterior distributions over the weights, and the third column visualizes the predictive distribution of the output variable y in the data space. This figure highlights several key characteristics of Bayesian machine learning:

- The first row corresponds to the case where no data are available, and thus the prior distribution constitutes the sole source of information. The right panel depicts six predictive samples drawn from the prior in the data space. The black dashed curve denotes the ground truth, while the red curves represent realizations from the prior. As expected, in the absence of observation data, the unit Gaussian prior yields predictions that broadly explore the function space but exhibit substantial deviation from the true underlying process.

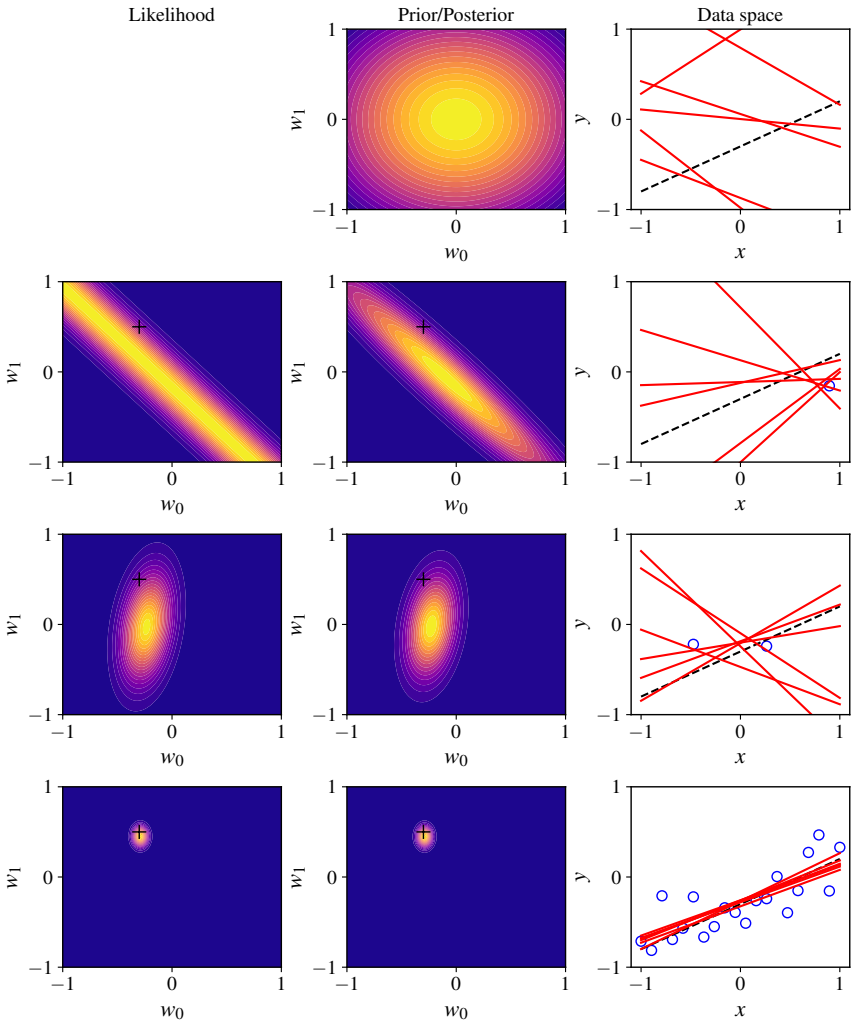
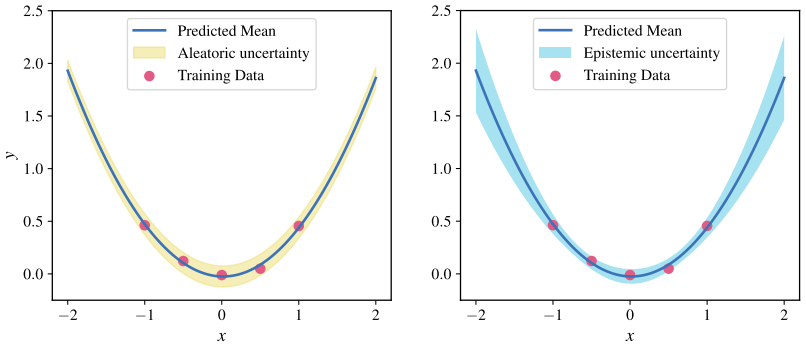


Figure 2.1: Illustration of Bayesian linear regression with a simple model $y = w_1 x + w_0$ replicated from [43].

- The second row illustrates the case with a single observed data point, represented by the blue circle in the data space. Incorporating this observation, the likelihood function imposes a soft constraint, encouraging the regression line to pass near the data point. By combining the likelihood with the prior distribution (shown in the middle panel of the first row), the resulting posterior distribution is obtained, as shown in the middle subfigure of the second row. It is evident that the posterior remains Gaussian. The corresponding predictions, displayed in the rightmost panel, fit the observed data point closely while maintaining high uncertainty in regions lacking data, thereby retaining exploratory behavior.

- The third row illustrates the scenario with two data points. The inclusion of two observations results in a more concentrated posterior distribution. Consequently, the predictive mean (right panel) aligns more closely with the ground truth, as the model is now constrained to fit both data points.

- The fourth row presents the case with 20 data points. A sharply peaked likelihood is observed, leading to a correspondingly sharp posterior distribution. The predictions in the right panel closely follow the ground truth, indicating high model confidence and significantly reduced epistemic uncertainty. In the limit of infinitely many data points, the posterior would converge to a delta function centered at the ground truth.



(a) Given aleatoric uncertainty

(b) Predicted epistemic uncertainty

Figure 2.2: Illustration of Bayesian Linear Regression. (a) Predicted mean with given aleatoric uncertainty; (b) Predicted mean and epistemic uncertainty.

Predictive distribution In practice, the posterior distribution over model parameters is often of less interest than the predictive posterior distribution, as the latter directly reflects uncertainty in the predicted observation y. Although Figure 2.1 briefly illustrates the relationship between the parameter posterior and the predictive distribution in data space, it is more common to report the results derived from Equation (2.13) directly. Therefore, I present an additional example with the ground truth function $y = 0.5x^2 + \varepsilon$, where $\varepsilon \sim \mathcal{N}(0, 0.05^2)$, as shown in Figure 2.2. Figure 2.2a shows the predicted mean along with the given aleatoric uncertainty, while Figure 2.2b depicts the predicted mean together with the epistemic uncertainty. Bayesian linear regression provides not only point predictions (mean) but also quantifies uncertainty. Notably, the epistemic uncertainty is low in regions where training data are available, and increases in regions with no data, reflecting the model's lack of knowledge in those areas.

Remark 2.2.2. *As illustrated in this section, the aleatoric uncertainty σ^2 is assumed to be known and input-independent, which is commonly referred to as homoscedastic data uncertainty. In Bayesian models, it is often treated as a hyperparameter.*

2.3. KERNEL-BASED REGRESSION: KRR AND GPR

Although a linear regression model is capable of capturing epistemic uncertainty, which is an informative indicator of model performance, it struggles with multi-dimensional and highly nonlinear problems. In theory, this limitation can be addressed by increasing the polynomial order to infinity; however, this approach introduces practical challenges such as overfitting to noise and a dramatic increase in the size of the feature matrix $\boldsymbol{\Phi}$. To overcome this bottleneck, kernel tricks are developed, which also refer to *Kernel Ridge Regression* (KRR). The Bayesian formulation of Kernel Ridge Regression leads to *Gaussian Process Regression* (GPR) [44].

2.3.1. KERNEL RIDGE REGRESSION

Kernel ridge regression is a simple generalization of Equation (2.3) from linear regression, assuming the dataset $\mathscr{D} = \{\mathbf{x} \in \mathscr{R}^{N\times d}, \boldsymbol{y} \in \mathscr{R}^{N\times 1}\}$, defined by

$$\hat{y} = \boldsymbol{w}^{\mathrm{T}} k(\mathbf{x}, \mathbf{x}). \tag{2.15}$$

where $\boldsymbol{w}$ are the model parameters and $k(\mathbf{x}, \mathbf{x})$ is a N-dimensional kernel feature vector. Similarly, the loss function of Kernel ridge regression with regularization on weights is given by

$$\mathscr{L}_2(\boldsymbol{w}) = \left\| \boldsymbol{K}\boldsymbol{w} - \boldsymbol{y} \right\|^2 + \lambda \boldsymbol{w}^{\top} \boldsymbol{K} \boldsymbol{w} \tag{2.16}$$

where $\boldsymbol{K} \in \mathscr{R}^{N\times N}$ is the covariance (Gram) matrix, which is defined by

$$\boldsymbol{K} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & k(\mathbf{x}_1, \mathbf{x}_2) & \cdots & k(\mathbf{x}_1, \mathbf{x}_N) \\ k(\mathbf{x}_2, \mathbf{x}_1) & k(\mathbf{x}_2, \mathbf{x}_2) & \cdots & k(\mathbf{x}_2, \mathbf{x}_N) \\ \vdots & \vdots & \ddots & \vdots \\ k(\mathbf{x}_N, \mathbf{x}_1) & k(\mathbf{x}_N, \mathbf{x}_2) & \cdots & k(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix} \tag{2.17}$$

By taking derivative of Equation (2.16) and setting it to zero, the close-form solution of $\boldsymbol{w}$ can be obtained as

$$\boldsymbol{w}^* = (\boldsymbol{K} + \lambda \boldsymbol{I}_N)^{-1} \boldsymbol{y} \tag{2.18}$$

WHAT IS KERNEL?

In the literature, various types of kernel functions have been proposed, such as the Radial Basis Function (RBF) kernel, polynomial kernel, sigmoid kernel, and so on [44]. In this section, a specific kernel is introduced, which results from the inner product of polynomial basis functions of infinite order – the RBF kernel. RBF kernel is defined by

$$k(\mathbf{x}^i, \mathbf{x}^j) = \exp\left(-\sum_{c=1}^{d} \theta_l \left(x_c^i - x_c^j \right)^2 \right) \tag{2.19}$$

where i, j represent the index of the training points, θ is a kernel parameter to control the relevance or influence of the c-th input feature.

Proof that the RBF kernel arises from an inner product over infinite polynomial basis functions.

Proof. The mathematical notation for a kernel (disregarding hyperparameters) can be rewritten as:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{1}{2}\|\mathbf{x}_i - \mathbf{x}_j\|^2\right) \tag{2.20}$$

It can be expanded as follows:

$$\begin{aligned} k(\mathbf{x}_i, \mathbf{x}_j) &= \exp\left(-\frac{1}{2}\|\mathbf{x}_i - \mathbf{x}_j\|^2\right) \\ &= \exp\left(-\frac{1}{2}\left(\langle \mathbf{x}_i, \mathbf{x}_i\rangle - 2\langle \mathbf{x}_i, \mathbf{x}_j\rangle + \langle \mathbf{x}_j, \mathbf{x}_j\rangle\right)\right) \\ &= \exp\left(-\frac{1}{2}\|\mathbf{x}_i\|^2\right)\exp\left(\langle \mathbf{x}_i, \mathbf{x}_j\rangle\right)\exp\left(-\frac{1}{2}\|\mathbf{x}_j\|^2\right) \\ &= \underbrace{\exp\left[-\frac{1}{2}\|\mathbf{x}_i\|^2 - \frac{1}{2}\|\mathbf{x}_j\|^2\right]}_{\text{Constant}}\exp\left(\langle \mathbf{x}_i, \mathbf{x}_j\rangle\right) \\ &= C \cdot \exp\left(\langle \mathbf{x}_i, \mathbf{x}_j\rangle\right) \qquad \text{(where } C \text{ is a constant w.r.t. } \mathbf{x}_i, \mathbf{x}_j\text{)} \end{aligned} \tag{2.21}$$

Now, using the Taylor expansion of the exponential function:

$$\exp\left(\langle \mathbf{x}_i, \mathbf{x}_j\rangle\right) = \sum_{m=0}^{\infty} \frac{1}{m!}\langle \mathbf{x}_i, \mathbf{x}_j\rangle^m \tag{2.22}$$

Hence,

$$k(\mathbf{x}_i, \mathbf{x}_j) = C \sum_{m=0}^{\infty} \frac{1}{m!}\langle \mathbf{x}_i, \mathbf{x}_j\rangle^m \tag{2.23}$$

□

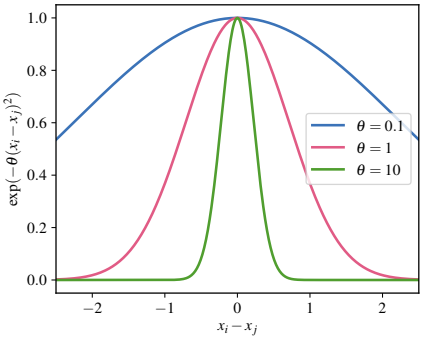


Figure 2.3: Illustration of kernel value between two random points x_i and x_j with different θ values [27].

Therefore, the RBF kernel is equivalent to an inner product in an *infinite-dimensional feature space* constructed by polynomial basis functions of all degrees. A one-dimensional illustration of the kernel value of two random points with different θ is shown in Figure 2.3. A larger values of θ lead to sharper peaks in the kernel function, indicating that only inputs very close to each other contribute significantly to the prediction. In contrast, smaller values of θ result in broader kernels, meaning the model considers more distant points as relevant, thereby producing smoother and more global generalizations.

2.3.2. GAUSSIAN PROCESS REGRESSION

GPR (also known as Kriging when the data is noiseless) can be viewed as the generalized form of Bayesian linear regression with embedding kernel tricks. In other words, it is kernel ridge regression with a Bayesian treatment. First of all, the posterior predictive distribution of Bayesian linear regression can be rewritten as follows:

$$p(y \mid \mathbf{x}, \mathscr{D}, \sigma^2) = \mathcal{N}\left(y \mid \tilde{\boldsymbol{w}}^\top \boldsymbol{\phi}(\boldsymbol{x}),\ \sigma^2 + \boldsymbol{\phi}(\boldsymbol{x})^\top \tilde{\boldsymbol{\Sigma}} \boldsymbol{\phi}(\boldsymbol{x})\right), \tag{2.24}$$

where $\tilde{\boldsymbol{w}}$ and $\tilde{\boldsymbol{\Sigma}}$ are given by Equation (2.12), which is rewritten here for convenience:

$$\tilde{\boldsymbol{w}} = \tilde{\boldsymbol{\Sigma}}\left(\hat{\boldsymbol{\Sigma}}^{-1}\hat{\boldsymbol{w}} + \frac{1}{\sigma^2}\boldsymbol{\Phi}^\top \boldsymbol{y}\right), \quad \tilde{\boldsymbol{\Sigma}} = \left(\hat{\boldsymbol{\Sigma}}^{-1} + \frac{1}{\sigma^2}\boldsymbol{\Phi}^\top \boldsymbol{\Phi}\right)^{-1}. \tag{2.25}$$

Selecting a proper polynomial order in Bayesian linear regression often requires manual tuning and may be insufficient for capturing highly nonlinear patterns. In contrast, as discussed in the previous section, kernel functions provide a powerful alternative by implicitly representing multi-dimensional feature spaces without explicitly constructing the basis functions. Assuming a Gaussian prior over the weights, $p(\boldsymbol{w}) = \mathcal{N}(\mathbf{0}, \hat{\boldsymbol{\Sigma}})$, the kernel trick allows the prior covariance of function values to be expressed as:

$$\boldsymbol{K} = \boldsymbol{\Phi}\hat{\boldsymbol{\Sigma}}\boldsymbol{\Phi}^\top, \quad k(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x})^\top \hat{\boldsymbol{\Sigma}} \boldsymbol{\phi}(\mathbf{x}'), \tag{2.26}$$

where $k(\mathbf{x}, \mathbf{x}')$ is the kernel function, $\boldsymbol{K} \in \mathbb{R}^{N\times N}$ is the kernel matrix over the training set.

Thus, using the Woodbury identity, the posterior covariance matrix of the weights in Bayesian linear regression can be rewritten as:

$$\tilde{\boldsymbol{\Sigma}} = \hat{\boldsymbol{\Sigma}} - \hat{\boldsymbol{\Sigma}}\boldsymbol{\Phi}^\top \left(\boldsymbol{K} + \sigma^2 \boldsymbol{I}\right)^{-1} \boldsymbol{\Phi}\hat{\boldsymbol{\Sigma}}. \tag{2.27}$$

By substituting Equation (2.27) into $\tilde{\boldsymbol{w}}$, and subsequently the predictive distribution of GPR for an unknown point $\mathbf{x}_*$ becomes:

$$p(y \mid \mathbf{x}, \mathscr{D}, \sigma^2) \sim \mathcal{N}\left(\boldsymbol{k}_*^\top (\boldsymbol{K} + \sigma^2 \boldsymbol{I}_N)^{-1} \boldsymbol{y},\ \underbrace{k(\mathbf{x}_*, \mathbf{x}_*) - \boldsymbol{k}_*^\top (\boldsymbol{K} + \sigma^2 \boldsymbol{I}_N)^{-1} \boldsymbol{k}_*}_{\text{epistemic}} + \underbrace{\sigma^2}_{\text{aleatoric}}\right) \tag{2.28}$$

where $\boldsymbol{K} \in \mathbb{R}^{N\times N}$ is the covariance matrix computed over the training inputs using a specified kernel function, and $\boldsymbol{k}_* \in \mathbb{R}^N$ is the covariance vector between the test input $\mathbf{x}_*$ and each training input.

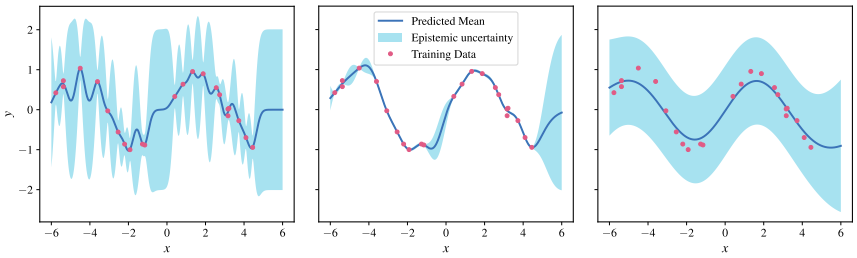
The previous derivation adopts a weight-space perspective that connects Bayesian linear regression with kernel ridge regression. Alternatively, GPR can also be derived from a function-space perspective, which assumes that the observations follow a multivariate Gaussian distribution. Instead of introducing explicit weight parameters, the function-space formulation directly models the covariance between outputs using a kernel function [44]. This approach leads to the same predictive distribution as shown in Equation (2.28). As a result, it is common to denote a GPR model compactly as:

$$f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x})), \tag{2.29}$$

which defines a Gaussian process prior with zero mean and covariance function $k(\mathbf{x}, \mathbf{x})$.

Remark 2.3.1. *Since the GPR model described in Equation (2.29) does not rely on a fixed set of parameters to explicitly define the function $f(\mathbf{x})$, but instead represents it through the kernel function evaluated over all training inputs, it is typically referred to as a* non-parametric *model. The only parameters involved are the kernel parameters $\boldsymbol{\theta}$ (also known as the hyperparameters), which control the smoothness of the covariance function.*

Hyperparameter influence on the performance of GPR A one-dimensional example with varying kernel hyperparameters is illustrated in Figure 2.4, where it is evident that these hyperparameters have a substantial impact on both the predictive mean and the associated epistemic uncertainty. In detail, a large value of θ leads to overfitting, while a small value results in underfitting. This highlights the critical role of kernel design and parameter tuning in the performance of GPR models. The procedure for estimating these hyperparameters will be discussed in detail in Section 2.3.2.



(a) $\theta = 10$ (b) $\theta = 1.0$ (c) $\theta = 10$

Figure 2.4: Illustration of GPR fitting with RBF kernel with different kernel parameters

GAUSSIAN PROCESS REGRESSION WITH EXPLICIT BASIS FUNCTION

Although it is common to adopt a zero-mean function due to GPR's strong empirical performance in many applications, it may limit the model's expressiveness or require more training data to achieve satisfactory results, especially when the true underlying function deviates significantly from zero [44]. To address this, explicitly modeling the mean function can improve both predictive performance and interpretability, where GPR with an explicit mean function can be defined as:

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x})), \tag{2.30}$$

More explicitly, it can be written as:

$$f(\mathbf{x}) = \boldsymbol{h}(\mathbf{x})^T \boldsymbol{\beta} + \delta(\mathbf{x}), \tag{2.31}$$

where $m(\mathbf{x}) = \boldsymbol{h}(\mathbf{x})^T \boldsymbol{\beta}$ is the mean function. $\boldsymbol{h}(\mathbf{x})$ is a set of basis function, i.e. $\boldsymbol{h}(\mathbf{x}) = [1, \mathbf{x}, \mathbf{x}^2, ...]^T$, with the length is set to be M; $\delta(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}))$ represents a zero-mean GPR as shown previously in Equation (2.29) that models the residuals that not captured by the explicit basis. The coefficient $\boldsymbol{\beta}$ is used to fit the mean trend and minimize the discrepancy between $\boldsymbol{h}(\mathbf{x})^T \boldsymbol{\beta}$ and $\boldsymbol{y}$. Both the kernel hyperparameters within the covariance function and $\boldsymbol{\beta}$ are determined by maximum likelihood estimation, which will be introduced in Section 2.3.2.

The predictive posterior distribution of the generalized GPR, accounting for the contribution of the explicit basis functions, can then be expressed as:

$$\begin{aligned} p(y \mid \mathbf{x}_*, \mathcal{D}, \sigma^2) \sim \mathcal{N}\Big(& \boldsymbol{h}(\mathbf{x}_*)^\top \hat{\boldsymbol{\beta}} + \boldsymbol{k}_*^\top (\boldsymbol{K} + \sigma^2 \boldsymbol{I}_N)^{-1} \left(\boldsymbol{y} - \boldsymbol{H}\hat{\boldsymbol{\beta}}\right), \\ & \underbrace{k(\mathbf{x}_*, \mathbf{x}_*) - \boldsymbol{k}_*^\top (\boldsymbol{K} + \sigma^2 \boldsymbol{I}_N)^{-1} \boldsymbol{k}_* + \boldsymbol{r}^\top \left(\boldsymbol{H}^\top (\boldsymbol{K} + \sigma^2 \boldsymbol{I}_N)^{-1} \boldsymbol{H}\right)^{-1} \boldsymbol{r}}_{\text{epistemic}} + \underbrace{\sigma^2}_{\text{aleatoric}} \Big), \end{aligned} \tag{2.32}$$

where $\boldsymbol{r} = \boldsymbol{h}(\mathbf{x}_*) - \boldsymbol{H}^\top (\boldsymbol{K} + \sigma^2 \boldsymbol{I}_N)^{-1} \boldsymbol{k}_*$, and $\boldsymbol{H} \in \mathbb{R}^{N \times M}$ is the design matrix formed by stacking $\boldsymbol{h}(\mathbf{x}_i)^\top$ row-wise for all training inputs.

The predictive mean contains two components: the contribution from the explicit basis function model $\boldsymbol{h}(\mathbf{x}_*)^\top \hat{\boldsymbol{\beta}}$, and the residual correction from the zero-mean Gaussian process. In the predictive variance, the first two terms correspond to the standard GPR variance, while the third term accounts for the additional uncertainty due to the estimation of the mean function [3].

PARAMETERS ESTIMATION

For the convenience of derivation, let $\boldsymbol{C} = \boldsymbol{K} + \sigma^2 \boldsymbol{I}_N$. Assuming $\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{H}\boldsymbol{\beta}, \boldsymbol{C})$, the marginal likelihood is

$$L(\boldsymbol{y} \mid \boldsymbol{\beta}, \boldsymbol{\theta}) = \frac{1}{\sqrt{(2\pi)^N |\boldsymbol{C}|}} \exp\left(-\frac{1}{2}(\boldsymbol{y} - \boldsymbol{H}\boldsymbol{\beta})^\top \boldsymbol{C}^{-1} (\boldsymbol{y} - \boldsymbol{H}\boldsymbol{\beta})\right). \tag{2.33}$$

Then, the log marginal likelihood can be expressed by:

$$\ell(\boldsymbol{\beta}, \boldsymbol{\theta}) = -\frac{1}{2}\left[\ln |\boldsymbol{C}| + (\boldsymbol{y} - \boldsymbol{H}\boldsymbol{\beta})^\top \boldsymbol{C}^{-1} (\boldsymbol{y} - \boldsymbol{H}\boldsymbol{\beta}) + N \ln(2\pi)\right]. \tag{2.34}$$

By taking derivatives of Equation (2.34) and setting it to zero ($\partial \ell / \partial \boldsymbol{\beta} = \boldsymbol{0}$), we obtain maximum likelihood estimation for $\boldsymbol{\beta}$.

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{H}^\top \boldsymbol{C}^{-1} \boldsymbol{H})^{-1} \boldsymbol{H}^\top \boldsymbol{C}^{-1} \boldsymbol{y}. \tag{2.35}$$

[3] GPR can also be applied to noiseless problems by setting $\sigma^2 = 0$, in which case the predictive distribution reflects only epistemic uncertainty and interpolates the training data exactly.

Substituting $\hat{\boldsymbol{\beta}}$ back into Equation (2.34) and constant terms removed to give the concentrated ln-likelihood function:

$$\ell(\boldsymbol{\theta}) = -\frac{1}{2}\Big[\ln|\boldsymbol{C}| + (\boldsymbol{y} - \boldsymbol{H}\hat{\boldsymbol{\beta}})^\top \boldsymbol{C}^{-1}(\boldsymbol{y} - \boldsymbol{H}\hat{\boldsymbol{\beta}})\Big]. \tag{2.36}$$

Remark 2.3.2. *The parameter estimation procedure also refers to type II maximum likelihood estimation since it maximizes the marginal likelihood of the observed data.*

The value of Equation (2.36) depends on the kernel hyperparameters $\boldsymbol{\theta}$ as well as σ^2 (if present), and its maximization generally does not admit a closed-form solution. Therefore, heuristic or gradient-based numerical optimization methods are typically employed to obtain the optimal values of $\boldsymbol{\theta}$ and σ^2 . It should be noted that the computational complexity of GPR scales cubically with the number of training samples, i.e., $\mathcal{O}(N^3)$, due to the need to invert the covariance matrix $\boldsymbol{C}$. As a result, despite its elegant theoretical formulation, GPR suffers from the so-called *curse of dimensionality*, which limits its applicability to large datasets.

Remark 2.3.3. *It is worth mentioning that GPR can predict homoscedastic noise by directly minimizing the concentrated log-likelihood in Equation* (2.36)*. However, when the data noise is input-dependent, i.e., $\sigma^2(\mathbf{x})$, this assumption no longer holds, and more advanced models or approximation strategies are required.*

2.4. NEURAL NETWORK REGRESSION: DNN AND BNN

To address *curse of dimensionality* of GPR, neural networks have emerged as a scalable alternative, renowned for their capacity to approximate any nonlinear continuous function given sufficient data [13]. In the literature, DNNs and BNNs are typically distinguished by their treatment of parameters: DNNs use deterministic weights, whereas BNNs place probability distributions over the weights to quantify uncertainty [45]. A schematic comparison of the two paradigms is illustrated in Figure 2.5.

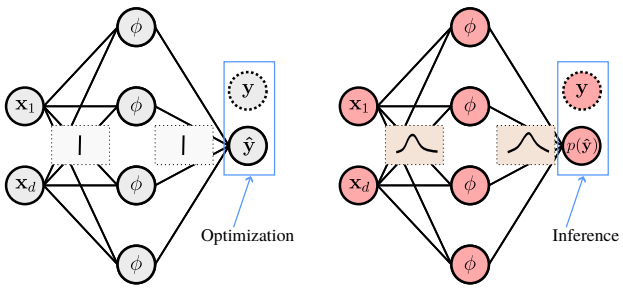


Figure 2.5: Schematics of DNN and BNN

As shown in Figure 2.5, both DNN and BNN have the same architecture: they take input $\mathbf{x}$, passes it through one or more hidden layers where nonlinear transformation $\phi(\cdot)$ is applied, and the predictions $\hat{\boldsymbol{y}}$ are outputted. Specifically, DNN (see Figure 2.5a) treats the parameters $\boldsymbol{\theta}$ including weights $\boldsymbol{W}$ and biases $\boldsymbol{b}$ as deterministic real quantities.

These parameters are typically learned by minimizing a loss function using optimization algorithms like *Adam* [46] and Stochastic Gradient Descent (SGD) [47]. On the other hand, BNN (see Figure 2.5b) defines parameters $\boldsymbol{\theta}$ as random variables with prior distributions. This probabilistic formulation enables the network to capture *epistemic uncertainty*, i.e., the uncertainty about the model parameters.

2.4.1. DEEP NEURAL NETWORK

According to Figure 2.5a, the general formulation of a DNN for approximating the input–output relationship can be written as:

$$\begin{aligned} \boldsymbol{a}^{(0)} &= \mathbf{x}, \\ \boldsymbol{a}^{(l)} &= \phi_l\left(\boldsymbol{W}^{(l)}\boldsymbol{a}^{(l-1)} + \boldsymbol{b}^{(l)}\right), \quad \text{for } l = 1,\dots,L-1, \\ f(\mathbf{x}) &= \boldsymbol{W}^{(L)}\boldsymbol{a}^{(L-1)} + \boldsymbol{b}^{(L)}, \end{aligned} \tag{2.37}$$

where L denotes the total number of layers, and $\phi_l(\cdot)$ represents the activation function applied element-wise at layer l. DNN is a composition of affine transformation and nonlinear activation. The non-linearity in terms of weights and biases parameters is introduced by the activation functions ϕ_l, which enable the network to approximate complex functions [13] [4].

Once the neural architecture (i.e., the number of layers L, the width D of each layer, and the choice of activation functions) is determined, the next step is to optimize the neural parameters, namely the weights and biases. That is to say, the goal is to minimize the discrepancy between the predicted output and the observation. A commonly used loss in regression problems is the MSE, which is the same as that of linear regression and KRR, defined as:

$$\mathscr{L}_2 = \frac{1}{N}\sum_{i=1}^{N}\left(f(\mathbf{x}_i) - y_i\right)^2 + \lambda\sum_{l=1}^{L}\left\|\boldsymbol{W}^{(l)}\right\|_F^2. \tag{2.38}$$

Note that the computation of $f(\mathbf{x})$ corresponds to the *forward-propagation* process in a DNN. Unlike linear regression or kernel-based methods, it is generally intractable to obtain an analytical solution for the neural parameters, due to the high-dimensional parameter space and the presence of nonlinear activation functions. Therefore, gradient-based optimization methods are employed, which update the neural parameters with the following formula:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta\nabla_{\boldsymbol{w}}\mathscr{L}_2, \tag{2.39}$$

η is the learning rate, and $\nabla_{\boldsymbol{w}}\mathscr{L}_2$ is the gradient with respect to $\boldsymbol{w}$. The gradient is usually computed via *automatic differentiation* [3], typically implemented through the *back-propagation* algorithm [5].

2.4.2. BAYESIAN NEURAL NETWORK

BNNs share the same neural architecture as DNNs, as illustrated in Figure 2.5b. Again, BNNs place probability distributions over the weights and biases instead of performing

[4] Usually, L and D are two hyperparameters of neural network

point estimation as in DNNs. This probabilistic formulation allows BNNs to capture epistemic uncertainty in the predictions. Accordingly, we can modify the Bayes' rule for BNNs [21], given by:

$$p(\boldsymbol{w} \mid \mathscr{D}) \propto p(\mathscr{D} \mid \boldsymbol{w})\, p(\boldsymbol{w}) \tag{2.40}$$

assuming a Gaussian likelihood $\mathcal{N}(f(\mathbf{x}), \sigma^2)$ and independent Gaussian priors over the weights and biases, i.e., $\mathcal{N}(0, \sigma^2 \boldsymbol{I}_N)$.

Compared to the setup of Bayesian linear regression (Equation (2.13)) and GPR (Equation (2.24)), retain the same probabilistic structure but replace the likelihood mean with a nonlinear function $f(\mathbf{x})$ defined by a DNN. The prediction of BNNs can be obtained by:

$$p(\boldsymbol{y}_* \mid \mathbf{x}_*, \mathscr{D}) = \int p(\boldsymbol{y}_* \mid \mathbf{x}_*, \boldsymbol{w})\, p(\boldsymbol{w} \mid \mathscr{D})\, \mathrm{d}\boldsymbol{w} \tag{2.41}$$

where $\mathbf{x}_*$ is the location of a point where the prediction is desired.

The neural network architecture used to describe the input-output relation of BNNs makes the closed-form solution for the posterior distribution impossible. To this end, inference methods are developed to approximate the posterior predictive distribution (see Section 2.5).

2.4.3. MODELS SUMMARY

Table 2.1: Comparison of Bayesian linear regression, Gaussian process regression, and Bayesian neural network in terms of model structure, inference, uncertainty modeling, and computational aspects.

Methods	BLR	GPR	BNNs
Model type	Parametric	Non-parametric	Parametric
Likelihood	$\mathcal{N}(\boldsymbol{y} \mid \boldsymbol{\Phi}\boldsymbol{w}, \sigma^2 \boldsymbol{I}_N)$	$\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{H}\boldsymbol{\beta}, \boldsymbol{C})$	$\mathcal{N}(\boldsymbol{y} \mid f_{\boldsymbol{\theta}}(\mathbf{x}), \sigma^2)$
Prior	Over weights	Functional prior	Over weights
Posterior	Closed-form	Closed-form	Intractable
PPD	Gaussian, closed-form	Gaussian, closed-form	Approximate
Uncertainty	Epistemic + Aleatoric	Epistemic + Aleatoric	Epistemic + Aleatoric
Scalability	$\mathcal{O}(D^3)$ w.r.t. features	$\mathcal{O}(N^3)$ w.r.t. data	Scalable; complexity depends on inference method
Application	Linear regression	Data-scarce problem	Large-scale problems

Table 2.1 compares the three models considering several essential aspects for Bayesian models. The choice of likelihood function across the models reflects their expressive capacity. Despite these structural differences, all three models ultimately produce the same types of outputs: predictive mean, aleatoric, and epistemic uncertainties. Among them, BNNs exhibit the highest expressivity, as they employ neural networks to parameterize the likelihood mean, allowing them to capture highly complex input–output relationships. However, this expressivity comes at the cost of analytical tractability, as the posterior distribution over parameters cannot be obtained in closed form. In contrast, GPR provides a closed-form posterior predictive distribution, which makes it particularly effective in data-scarce settings. Nevertheless, its computational complexity scales cubically with

the number of data points, limiting its applicability in high-dimensional or large-scale problems.

2.5. INFERENCE METHODS

The process of computing the posterior distribution over these variables based on observed data following Bayes' rule, is referred to as *inference*. This subsection begins with a simple closed-form inference example based on a one-dimensional linear regression problem, serving as a warm-up for understanding Bayes' rule in practice. Subsequently, two major categories of approximate inference techniques are introduced: Markov Chain Monte Carlo sampling and variational inference.

2.5.1. CLOSED-FORM INFERENCE

Consider a simplified linear regression problem in which only the slope parameter θ is treated as unknown, while the intercept b is assumed to be known, yielding the following linear model:

$$y = \theta x + b, \tag{2.42}$$

Assuming Gaussian observation distribution ($p(y \mid \theta) = \mathcal{N}(y \mid \theta x + b, \sigma^2)$) and Gaussian prior ($p(\theta) = \mathcal{N}(\theta \mid \mu_\theta, \sigma_\theta^2)$), and according to Bayes' rule (Equation (2.1)), we can obtain:

$$\begin{aligned} p(y \mid \theta)\, p(\theta) &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y-(\theta x+b))^2}{2\sigma^2}\right) \cdot \frac{1}{\sqrt{2\pi\sigma_\theta^2}} \exp\left(-\frac{(\theta-\mu_\theta)^2}{2\sigma_\theta^2}\right) \\ &= C_* \cdot \frac{1}{\sqrt{2\pi\sigma_*^2}} \exp\left(-\frac{(\theta-\mu_*)^2}{2\sigma_*^2}\right), \end{aligned} \tag{2.43}$$

where:

$$\mu_* = \sigma_*^2\left(\frac{\mu_\theta}{\sigma_\theta^2} + \frac{x(y-b)}{\sigma^2}\right),\ \sigma_*^2 = \left(\frac{1}{\sigma_\theta^2} + \frac{x^2}{\sigma^2}\right)^{-1},\ C_* = \frac{1}{|x|\sqrt{2\pi\left(\sigma_\theta^2 + \frac{\sigma^2}{x^2}\right)}} \exp\left[-\frac{\left(\mu_\theta - \frac{y-b}{x}\right)^2}{2\left(\sigma_\theta^2 + \frac{\sigma^2}{x^2}\right)}\right].$$

The numerator of Bayes' rule for the univariate case is the product of the likelihood and the prior, which yields a Gaussian distribution scaled by a constant C^*, as shown in Equation (2.43). The evidence distribution can then be calculated by integrating over all possible values of θ:

$$p(y) = \int p(y|\theta)\, p(\theta)\, d\theta = \int C^* \cdot \frac{1}{\sqrt{2\pi\sigma_*^2}} \exp\left(-\frac{(\theta-\mu_*)^2}{2\sigma_*^2}\right) d\theta = C^*. \tag{2.44}$$

The evidence obtained in Equation (2.44) is the same normalization constant C^*. By substituting both Equation (2.44) and Equation (2.43) into Bayes' rule, the posterior

distribution for the univariate problem can be derived as:

$$p(\theta \mid y) = \frac{p(y \mid \theta)p(\theta)}{p(y)} = \frac{C^* \cdot \frac{1}{\sqrt{2\pi\sigma_*^2}} \exp\left(-\frac{(\theta-\mu_*)^2}{2\sigma_*^2}\right)}{C^*} = \frac{1}{\sqrt{2\pi\sigma_*^2}} \exp\left(-\frac{(\theta-\mu_*)^2}{2\sigma_*^2}\right) \tag{2.45}$$

Therefore, the posterior distribution $p(\theta \mid y)$ is a univariate Gaussian distribution with updated mean μ_* and variance σ_*^2, expressed as:

$$p(\theta \mid y) = \mathcal{N}(\theta \mid \mu_*, \sigma_*^2), \tag{2.46}$$

where μ_* is the predicted mean that is a weighted average between the prior mean μ_θ and the transformation of the observation $(y-b)/x$, and the posterior variance σ_*^2 relates to all factors of data x, prior variance σ_θ^2, and σ^2.

The derivation of the univariate case illustrates a fundamental property of Bayesian learning: the posterior distribution balances prior beliefs and data-driven likelihood in a consistent manner. Specifically, in the case where the prior variance $\sigma_\theta^2 \to \infty$, the posterior mean converges to:

$$\mu_* \to \frac{y-b}{x}, \quad \text{and} \quad \sigma_*^2 \to \frac{\sigma^2}{x^2}. \tag{2.47}$$

Conversely, when the likelihood becomes highly uncertain (i.e., $\sigma^2 \to \infty$), the posterior distribution reverts to the prior:

$$\mu_* \to \mu_\theta, \quad \text{and} \quad \sigma_*^2 \to \sigma_\theta^2. \tag{2.48}$$

2.5.2. MARKOV CHAIN MONTE CARLO METHODS

For most Bayesian machine learning models, the marginal distribution cannot be computed analytically due to the complexity of the integrals involved. As a result, approximate inference methods are required to estimate the posterior distribution. The most straightforward method is the Monte Carlo Simulation (MCS) [48], which draws random samples from the posterior distribution directly to approximate expectations. Unfortunately, MCS is often infeasible for BNNs since the evidence is usually intractable, especially in high-dimensional spaces. To resolve this bottleneck, the Markov Chain Monte Carlo (MCMC) method was developed. In short, MCMC builds a stationary Markov chain on the state space Θ with the target distribution of interest, which is usually the $p(\boldsymbol{\theta}|\mathscr{D})$. MCMC starts with generating random samples sequentially, where each state $\boldsymbol{\theta}$ is proportional to $p(\boldsymbol{\theta}|\mathscr{D})$. Provided enough correlated samples $\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \cdots$, it approaches the true posterior distribution $p(\boldsymbol{\theta}|\mathscr{D})$.

RANDOM WALK MARKOV CHAIN MONTE CARLO

The most straightforward implementation of MCMC is the Random Walk Metropolis-Hastings (RWMH) algorithm (Algorithm 1), which requires a proposal distribution to generate candidate samples. Starting from an initial point, the algorithm iteratively proposes new samples and accepts them with a probability that ensures the chain converges to the desired posterior.

Algorithm 1: Random Walk Metropolis-Hastings (RWMH) Algorithm

Data: target $\tilde{p}(\boldsymbol{\theta})$, proposal distribution $q(\boldsymbol{\theta}'|\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}'|\boldsymbol{\theta}, \tau^2\boldsymbol{I})$, total iterations s, step size τ
Result: Sampled collection $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_s\}$
Initialize $\boldsymbol{\theta}^0$
for $s = 0, 1, 2, \ldots, s-1$ **do**
  Sample proposal $\boldsymbol{\theta}' \sim q(\boldsymbol{\theta}'|\boldsymbol{\theta}_s)$
  Compute acceptance ratio:
    $\alpha = \frac{\tilde{p}(\boldsymbol{\theta}')q(\boldsymbol{\theta}_s|\boldsymbol{\theta}')}{\tilde{p}(\boldsymbol{\theta}_s)q(\boldsymbol{\theta}'|\boldsymbol{\theta}_s)}$
  Compute acceptance probability $A = \min(1, \alpha)$
  Draw $u \sim \mathcal{U}(0, 1)$
  if $u \leq A$ **then**
    Accept: $\boldsymbol{\theta}_{s+1} = \boldsymbol{\theta}'$
  else
    Reject: $\boldsymbol{\theta}_{s+1} = \boldsymbol{\theta}_s$
  end
  Store: $\boldsymbol{\Theta} = \boldsymbol{\Theta} \cup \{\boldsymbol{\theta}_{s+1}\}$
end

According to Algorithm 1, the proposal distribution is a predefined distribution that governs the generation of new candidate samples. The step size τ determines the scale or breadth of the proposal, influencing how far the algorithm explores the parameter space in each iteration. The algorithm proceeds for a total of s iterations. While RWMH method is theoretically guaranteed to converge to the true posterior given infinite computational resources, it is well known for its inefficiency in high-dimensional settings, often suffering from high rejection rates and poor mixing behavior.

HAMILTONIAN MONTE CARLO

To accelerate the mixing rate of MCMC methods, the Hamiltonian Monte Carlo (HMC) algorithm was introduced by Neal [21]. HMC leverages principles from Hamiltonian mechanics to guide the sampling process, utilizing the *gradient* of the target probability density function to inform proposals. The Hamiltonian mechanics system (Figure 2.6) is assumed to be an idealized frictionless system, which leads to total energy conservation. Specifically, the total energy can be expressed by the summation of potential energy and kinetic energy, defined by:

$$\mathcal{H}(\boldsymbol{\theta}, \boldsymbol{v}) = \mathcal{E}(\boldsymbol{\theta}) + \mathcal{K}(\boldsymbol{v}), \tag{2.49}$$

where $\mathcal{H}(\boldsymbol{\theta}, \boldsymbol{v})$ is total energy of the Hamiltonian system with potential energy $\mathcal{E}(\boldsymbol{\theta})$ and kinetic energy $\mathcal{K}(\boldsymbol{v})$. In the Bayesian inference setting, the potential energy is usually set to be logarithm target, i.e., the possibly unnormalized distribution:

$$\mathcal{E}(\boldsymbol{\theta}) = -\log \tilde{p}(\boldsymbol{\theta}), \tag{2.50}$$

and the kinetic energy is assumed to be:

$$\mathcal{K}(\boldsymbol{v}) = \frac{1}{2}\boldsymbol{v}^T \boldsymbol{\Sigma}^{-1}\boldsymbol{v}, \tag{2.51}$$

where $\boldsymbol{\Sigma}$ is a positive mass matrix, known as the inverse mass matrix.

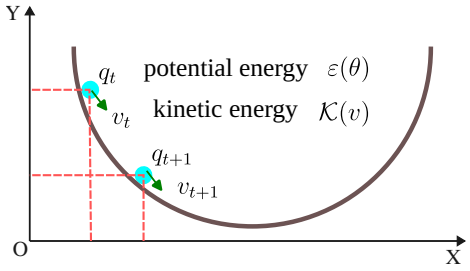


Figure 2.6: Schematic of Hamiltonian mechanics system

The bowl-shaped surface (Figure 2.6) represents the target, with the blue dots indicating the desired sample locations. The sampling process can be interpreted as traversing this landscape according to the position-update dynamics governed by Hamiltonian mechanics. How to integrate the Hamilton's equation (Equation (2.49)) effectively poses a bottleneck. The Euler's method ([49]) with the position and velocity updating rule in time can be written as:

$$\begin{aligned}
\boldsymbol{v}_{t+1} &= \boldsymbol{v}_t + \eta\frac{\mathrm{d}\boldsymbol{v}(\boldsymbol{\theta}_t, \boldsymbol{v}_t)}{\mathrm{d}t} = \boldsymbol{v}_t - \eta\frac{\partial\mathscr{E}(\boldsymbol{\theta}_t)}{\partial\boldsymbol{\theta}} \\
\boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t + \eta\frac{\mathrm{d}\boldsymbol{\theta}(\boldsymbol{\theta}_t, \boldsymbol{v}_t)}{\mathrm{d}t} = \boldsymbol{\theta}_t - \eta\frac{\partial\mathcal{K}(\boldsymbol{v}_t)}{\partial\boldsymbol{v}}
\end{aligned} \tag{2.52}$$

A modified Euler's method to improve the accuracy is proposed where the position $\boldsymbol{\theta}_{t+1}$ is updated based on the gradient of kinetic energy at $t+1$ instead of t in the Equation (2.52).

$$\begin{aligned}
\boldsymbol{v}_{t+1} &= \boldsymbol{v}_t + \eta\frac{\mathrm{d}\boldsymbol{v}(\boldsymbol{\theta}_t, \boldsymbol{v}_t)}{\mathrm{d}t} = \boldsymbol{v}_t - \eta\frac{\partial\mathscr{E}(\boldsymbol{\theta}_t)}{\partial\boldsymbol{\theta}} \\
\boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t + \eta\frac{\mathrm{d}\boldsymbol{\theta}(\boldsymbol{\theta}_t, \boldsymbol{v}_{t+1})}{\mathrm{d}t} = \boldsymbol{\theta}_t - \eta\frac{\partial\mathcal{K}(\boldsymbol{v}_{t+1})}{\partial\boldsymbol{v}}
\end{aligned} \tag{2.53}$$

Combining the advantages of both Equation (2.52) and Equation (2.53), the leapfrog method is adopted as the final update rule, which is defined as:

$$\boldsymbol{v}_{t+\frac{1}{2}} = \boldsymbol{v}_t - \frac{\eta}{2}\frac{\partial\mathscr{E}(\boldsymbol{\theta}_t)}{\partial\boldsymbol{\theta}}, \quad \boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \eta\frac{\partial\mathcal{K}(\boldsymbol{v}_{t+\frac{1}{2}})}{\partial\boldsymbol{v}}, \quad \boldsymbol{v}_{t+1} = \boldsymbol{v}_{t+\frac{1}{2}} - \frac{\eta}{2}\frac{\partial\mathscr{E}(\boldsymbol{\theta}_{t+1})}{\partial\boldsymbol{\theta}} \tag{2.54}$$

According to Equation (2.54), the leapfrog integration scheme utilizes a half-step update at time $t+\frac{1}{2}$ to improve the position update. Essentially, the leapfrog method is a

symplectic integrator commonly used to enhance the accuracy of Hamiltonian dynamics simulation. That is to say, the leapfrog scheme achieves higher accuracy with a larger number of repeated integration steps. In this context, the number of leapfrog steps L is treated as a hyperparameter in HMC[5]. Overall, the detailed procedure of HMC is summarized in Algorithm 2. The HMC algorithm still involves an accept/reject step. Although the proposals generated by simulating Hamiltonian dynamics significantly improve the acceptance ratio compared to random-walk methods, this procedure remains a limiting factor for large-scale problems. Furthermore, each HMC sample requires L *leapfrog* steps, which can substantially increase the overall computational cost.

Algorithm 2: Hamiltonian Monte Carlo Algorithm

Data: Target density $p(\boldsymbol{\theta})$, step size η, mass matrix $\boldsymbol{\Sigma}$, desired samples size S, burn-in iteration N_b, leapfrog steps L
Result: Samples $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_S\}$
Initialize $\boldsymbol{\theta}^0$
$\boldsymbol{\Theta} \leftarrow \emptyset$
for $t = 0$ **to** $S + N_b - 1$ **do**
  Sample momentum $\boldsymbol{\nu}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$
  Set $(\boldsymbol{\theta}'_0, \boldsymbol{\nu}'_0) = (\boldsymbol{\theta}_t, \boldsymbol{\nu}_t)$
  Half step for momentum: $\boldsymbol{\nu}'_{1/2} = \boldsymbol{\nu}'_0 - \frac{\eta}{2} \nabla \mathscr{E}(\boldsymbol{\theta}'_0)$
  for $l = 1$ **to** $L - 1$ **do**
    $\boldsymbol{\theta}'_l = \boldsymbol{\theta}'_{l-1} + \eta \boldsymbol{\Sigma}^{-1} \boldsymbol{\nu}'_{l-1/2}$
    $\boldsymbol{\nu}'_{l+1/2} = \boldsymbol{\nu}'_{l-1/2} - \eta \nabla \mathscr{E}(\boldsymbol{\theta}'_l)$
  end
  Full step for position: $\boldsymbol{\theta}'_L = \boldsymbol{\theta}'_{L-1} + \eta \boldsymbol{\Sigma}^{-1} \boldsymbol{\nu}'_{L-1/2}$
  Half step for momentum: $\boldsymbol{\nu}'_L = \boldsymbol{\nu}'_{L-1/2} - \frac{\eta}{2} \nabla \mathscr{E}(\boldsymbol{\theta}'_L)$
  Compute acceptance ratio:

$$\alpha = \min\Big(1, \exp\big[-\mathscr{H}(\boldsymbol{\theta}'_L, \boldsymbol{\nu}'_L) + \mathscr{H}(\boldsymbol{\theta}_t, \boldsymbol{\nu}_t)\big]\Big)$$

  Sample $u \sim \mathscr{U}(0, 1)$
  if $u \leq \alpha$ **then**
    Accept: $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}'_L$
  else
    Reject: $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t$
  end
  if $t > N_b$ **then**
    Append: $\boldsymbol{\Theta} \leftarrow \boldsymbol{\Theta} \cup \{\boldsymbol{\theta}_t\}$
  end
end

[5] Each leapfrog step requires evaluating the posterior probability density function.

STOCHASTIC GRADIENT LANGEVIN DYNAMICS

Welling et al.[50] proposed a novel inference approach called Stochastic Gradient Langevin Dynamics (SGLD), which combines the advantages of SGD and Langevin Monte Carlo (LMC) [51] to further enhance HMC, especially for large scale problems, like BNNs. The LMC updating step is introduced as a special case of HMC with $L = 1$, which given by:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \frac{\eta^2}{2}\boldsymbol{\Sigma}^{-1}\nabla\mathscr{E}(\boldsymbol{\theta}_t) + \eta\,\boldsymbol{\Sigma}^{-1}\boldsymbol{v}_t = \boldsymbol{\theta}_t - \frac{\eta^2}{2}\boldsymbol{\Sigma}^{-1}\nabla\mathscr{E}(\boldsymbol{\theta}_t) + \eta\sqrt{\boldsymbol{\Sigma}^{-1}}\,\boldsymbol{\epsilon}_t \tag{2.55}$$

where $\boldsymbol{v}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ and $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. It is worth noting that the above update resembles SGD with an added noise term, $\eta\sqrt{\boldsymbol{\Sigma}^{-1}}\boldsymbol{\epsilon}_t$. When $\boldsymbol{\Sigma} = \boldsymbol{I}$ and $\eta = \sqrt{2}$, the update process reduces to a discretized Langevin diffusion, given by:

$$\mathrm{d}\boldsymbol{\theta}_t = -\nabla\mathscr{E}(\boldsymbol{\theta}_t)\,\mathrm{d}t + \sqrt{2}\,\mathrm{d}\boldsymbol{B}_t \tag{2.56}$$

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

Remark 2.5.1. *The LMC update step eliminates the* leapfrog *integration step by introducing an additional term* $\frac{\eta^2}{2}\boldsymbol{\Sigma}^{-1}\nabla\mathscr{E}(\boldsymbol{\theta}_t)$. *Building upon the LMC updating step, the Metropolis-Hastings acceptance step can also be removed with a simple modification, further enhancing the method's scalability for large-scale problems.*

Using SGD for a loss function with the form of MSE involves the gradient with respect to the parameters $\boldsymbol{\theta}$, which can be formulated as:

$$\nabla_{\boldsymbol{\theta}}\mathscr{L}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{n=1}^{N}\nabla\mathscr{L}_n(\boldsymbol{\theta}), \tag{2.57}$$

Practically, it is common to use a mini-batch where the gradient is given by:

$$\nabla^B_{\boldsymbol{\theta}}\mathscr{L}(\boldsymbol{\theta}) = \frac{1}{B}\sum_{n=1}^{B}\nabla_{\boldsymbol{\theta}}\mathscr{L}_n(\boldsymbol{\theta}), \tag{2.58}$$

where B is the size of the mini-batch. It provides an unbiased estimate of the true gradient over the entire dataset. However, a diffusion term is associated with both the mini-batch gradient and the full-batch gradient. Specifically, according to Equation (2.57) and Equation (2.58), the mini-batch is randomly sampled from the full batch; thus, the expectation of the mini-batch can be calculated by:

$$\mathbb{E}_{\mathscr{B}}\left[\nabla^B_{\boldsymbol{\theta}}\mathscr{L}(\boldsymbol{\theta}) - \nabla_{\boldsymbol{\theta}}\mathscr{L}(\boldsymbol{\theta})\right] = 0, \tag{2.59}$$

which confirms that the mini-batch estimator is unbiased. Furthermore, the variance (covariance matrix) of this difference, also referred to as the diffusion term:

$$\mathrm{Cov}_{\mathscr{B}}\left[\nabla^B_{\boldsymbol{\theta}}\mathscr{L}(\boldsymbol{\theta}) - \nabla_{\boldsymbol{\theta}}\mathscr{L}(\boldsymbol{\theta})\right] = \mathbb{E}_{\mathscr{B}}\left[\left(\nabla^B_{\boldsymbol{\theta}}\mathscr{L}(\boldsymbol{\theta}) - \nabla_{\boldsymbol{\theta}}\mathscr{L}(\boldsymbol{\theta})\right)\left(\nabla^B_{\boldsymbol{\theta}}\mathscr{L}(\boldsymbol{\theta}) - \nabla_{\boldsymbol{\theta}}\mathscr{L}(\boldsymbol{\theta})\right)^\top\right]. \tag{2.60}$$

Similarly, the difference in the gradient estimation between mini-batch and full-batch for time step t can be defined by:

$$\boldsymbol{v}_t = \sqrt{\eta}\left(\nabla \mathscr{L}(\boldsymbol{\theta}_t) - \nabla_B \mathscr{L}(\boldsymbol{\theta}_t)\right) \tag{2.61}$$

As a result, the updating step of SGD can be explicitly written as:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \nabla^B \mathscr{L}(\boldsymbol{\theta}_t) = \boldsymbol{\theta}_t - \eta \nabla \mathscr{L}(\boldsymbol{\theta}_t) + \sqrt{\eta}\boldsymbol{v}_t, \tag{2.62}$$

which has an identical expression as Equation (2.55).

To this end, the SGLD proposed by Welling et al. [50], a specific case of LMC, aiming for the posterior inference for BNNs could be formulated naturally. Since the posterior distribution of BNNs is proportional to the product of the prior and the likelihood, i.e., $p(\boldsymbol{\theta} \mid \mathscr{D}) \propto p(\boldsymbol{\theta}) \prod_{n=1}^{N} p(y_n \mid \boldsymbol{\theta})$, the SGLD update rule for sampling from the posterior can be formulated as:

$$\Delta\boldsymbol{\theta}_t = \frac{\epsilon_t}{2}\left(\nabla \log p(\boldsymbol{\theta}_t) + \frac{N}{n}\log p(y_i|\boldsymbol{\theta}_t)\right) + \eta_t \mathbf{I} \tag{2.63}$$

where $\eta_t \sim \mathscr{N}(\mathbf{0}, \epsilon_t)$. Accordingly, the SGLD procedures are summarized in Algorithm 3.

Algorithm 3: Stochastic Gradient Langevin Dynamics (SGLD)

Data: Target posterior $p(\boldsymbol{\theta}|\mathscr{D})$, step size schedule ϵ_t, total samples S, burn-in iterations N_b
Result: Samples $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_S\}$
Initialize $\boldsymbol{\theta}^0$
$\boldsymbol{\Theta} \leftarrow \emptyset$
for $t = 0$ **to** $S + N_b - 1$ **do**
  Sample mini-batch $\mathscr{B}_t = \{(x_i, y_i)\}_{i=1}^{n}$
  Sample noise $\boldsymbol{\eta}_t \sim \mathscr{N}(\mathbf{0}, \epsilon_t \mathbf{I})$
  Update the position of $\boldsymbol{\theta}$:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \frac{\epsilon_t}{2}\left(\nabla \log p(\boldsymbol{\theta}_t) + \frac{N}{n}\log p(y_i|\boldsymbol{\theta}_t)\right) + \boldsymbol{\eta}_t$$

  if $t > N_b$ **then**
    Append: $\boldsymbol{\Theta} \leftarrow \boldsymbol{\Theta} \cup \{\boldsymbol{\theta}_{t+1}\}$
  end
end

PRECONDITIONED STOCHASTIC GRADIENT LANGEVIN DYNAMICS

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) [52], was proposed to address this issue. In pSGLD, the update rule incorporates a user-defined preconditioning matrix $G(\boldsymbol{\theta}_t)$ leveraging the same preconditioner employed in RMSprop [52], which adjusts the gradient updates and the noise term adaptively:

$$\Delta\boldsymbol{\theta}_t = \frac{\epsilon_t}{2}\left[G(\boldsymbol{\theta}_t)\left(\nabla \log p(\boldsymbol{\theta}_t) + \frac{N}{n}\log p(y_i|\boldsymbol{\theta}_t)\right) + \Gamma(\boldsymbol{\theta}_t)\right] + \eta_t G(\boldsymbol{\theta}_t) \tag{2.64}$$

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

$$G(\boldsymbol{\theta}_{t+1}) = \text{diag}\left(\mathbf{1} \oslash \left(\lambda \mathbf{1} + \sqrt{V(\boldsymbol{\theta}_{t+1})}\right)\right) \tag{2.65}$$

$$V(\boldsymbol{\theta}_{t+1}) = \alpha V(\boldsymbol{\theta}_t) + (1-\alpha)\,\bar{g}(\boldsymbol{\theta}_t; \mathscr{B}_t) \odot \bar{g}(\boldsymbol{\theta}_t; \mathscr{B}_t) \tag{2.66}$$

where $\oslash$ and $\odot$ are the element-wise matrix division and multiplication, respectively; and the mean gradient $\bar{g}(\boldsymbol{\theta}_t; \mathscr{B}_t)$ is given by:

$$\bar{g}(\boldsymbol{\theta}_t; \mathscr{B}_t) = \frac{1}{n}\sum_{i=1}^{n} \nabla_{\boldsymbol{\theta}} \log p(y_i \mid \boldsymbol{\theta}_t), \tag{2.67}$$

Details of pSGLD are given by Algorithm 4, where it is highlighted that two additional hyperparameters are introduced: λ and α, which control the curvature and balance the historical and current gradients, respectively. By default, λ and α are set to be 10^{-5} and 0.99, respectively.

Algorithm 4: Preconditioned Stochastic Gradient Langevin Dynamics (pSGLD)

Data: Target posterior $p(\boldsymbol{\theta}|\mathscr{D})$, step size schedule ϵ_t, total samples S, burn-in iterations N_b, decay rate α, stability constant λ
Result: Samples $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_S\}$
Initialize $\boldsymbol{\theta}^0$, $V^0 = \mathbf{0}$
$\boldsymbol{\Theta} \leftarrow \emptyset$
for $t = 0$ **to** $S + N_b - 1$ **do**
  Sample mini-batch $\mathscr{B}_t = \{(x_i, y_i)\}_{i=1}^n$
  Calculate $\bar{g}_t = \frac{1}{n}\sum_{i=1}^n \nabla_{\boldsymbol{\theta}} \log p(y_i \mid \boldsymbol{\theta}_t)$
  Update the preconditioner

$$V(\boldsymbol{\theta}_{t+1}) = \alpha V(\boldsymbol{\theta}_t) + (1-\alpha)\,\bar{g}(\boldsymbol{\theta}_t; \mathscr{B}_t) \odot \bar{g}(\boldsymbol{\theta}_t; \mathscr{B}_t)$$

$$G(\boldsymbol{\theta}_{t+1}) = \text{diag}\left(\mathbf{1} \oslash \left(\lambda \mathbf{1} + \sqrt{V(\boldsymbol{\theta}_{t+1})}\right)\right)$$

  Compute $\Gamma(\boldsymbol{\theta}_t)_i = \sum_j \frac{\partial G_{i,j}^{t+1}}{\partial \theta_j}$
  Sample noise $\boldsymbol{\eta}_t \sim \mathscr{N}(\mathbf{0}, \epsilon_t \mathbf{I})$
  Update the position of $\boldsymbol{\theta}$:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \frac{\epsilon_t}{2}\left[G^{t+1}\left(\nabla \log p(\boldsymbol{\theta}_t) + \frac{N}{n}\bar{g}_t\right) + \Gamma(\boldsymbol{\theta}_t)\right] + \boldsymbol{\eta}_t G^{t+1}$$

  if $t \geq N_b$ **then**
    Append: $\boldsymbol{\Theta} \leftarrow \boldsymbol{\Theta} \cup \{\boldsymbol{\theta}_{t+1}\}$
  end
end

SHORT SUMMARY

As illustrated in this section, MCMC samplers start with a random initialization, then they try to integrate the probability integral sequentially with given update rules. A schematic shown in Figure 2.7, where the shaded area indicates a typical set, which is the region that covers the high probabilities of the posterior distribution. Usually, the MCMC samplers have three stages: the first stage is to search for the typical set; the second stage is the most effective phase, exploring the typical set rapidly; and in the third stage, where the sampler may tend to converge to a single mode.

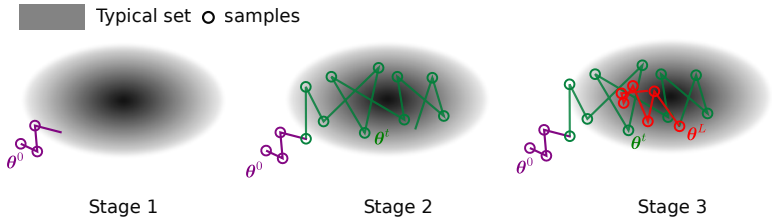


Figure 2.7: Schematic of MCMC samples for Bayesian inference

2.5.3. VARIATIONAL INFERENCE

VI approximates the PPD by a proposed distribution from a parametric family [53]. The objective of VI is to minimize the Kullback-Leibler divergence between the true posterior distribution $p(\boldsymbol{\theta}|\mathscr{D})$ and the proposed distribution $q(\boldsymbol{\theta})$ as illustrated in Figure 2.8.

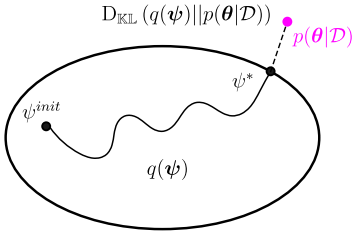


Figure 2.8: Schematic of VI [45]: The ellipse represents the search space of the proposed distribution, $p(\boldsymbol{\theta}|\mathscr{D})$ is the true posterior distribution, the KL divergence is the distance between the two distributions defined as $\mathrm{D}_{\mathbb{KL}}\left(q(z)||p_\theta(z|x)\right)$, and the goal is to minimize it.

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:

$$
\begin{aligned}
\psi^* &= \arg\min_{\psi} \mathrm{D}_{\mathbb{KL}}\left(q_\psi(\boldsymbol{\theta})||p_\theta(\boldsymbol{\theta}|\mathscr{D})\right) \\
&= \arg\min_{\psi} \mathbb{E}_{q_\psi(\boldsymbol{\theta})}\left[\log q_\psi(\boldsymbol{\theta}) - \log p_\theta(\boldsymbol{\theta}|\mathscr{D})\right] \\
&= \arg\min_{\psi} \mathbb{E}_{q_\psi(\boldsymbol{\theta})}\left[\log q_\psi(\boldsymbol{\theta}) - \log\left(\frac{p_\theta(\mathscr{D}|\boldsymbol{\theta})\,p_\theta(\boldsymbol{\theta})}{p_\theta(\mathscr{D})}\right)\right] \\
&= \arg\min_{\psi} \mathbb{E}_{q_\psi(\boldsymbol{\theta})}\left[\log q_\psi(\boldsymbol{\theta}) - \log p_\theta(\mathscr{D}|\boldsymbol{\theta}) - \log p_\theta(\boldsymbol{\theta})\right] + \log p_\theta(\mathscr{D})
\end{aligned}
\tag{2.68}
$$

According to Equation (2.68), 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:

$$\begin{aligned}\psi^* &= \arg\min_{\psi} \mathbb{E}_{q_\psi(\boldsymbol{\theta})}\left[\log q_\psi(\boldsymbol{\theta}) - \log p_\theta(\mathscr{D}|\boldsymbol{\theta}) - \log p_\theta(\boldsymbol{\theta})\right] \\ &= \arg\min_{\psi} \mathbb{E}_{q_\psi(\boldsymbol{\theta})}\left[-\log p_\theta(\mathscr{D}|\boldsymbol{\theta})\right] + \mathrm{D}_{\mathbb{KL}}(q_\psi(\boldsymbol{\theta})||p_\theta(\boldsymbol{\theta}))\end{aligned} \tag{2.69}$$

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.

2.5.4. EXAMPLE ILLUSTRATIONS

Gaussian mixture distribution The first example is to illustrate the advantages and bottlenecks several inference methods, including HMC [49], SGLD [50], pSGLD [52], and VI [45], based on a two-dimensional Gaussian mixture distribution, which is given by:

$$p(\boldsymbol{\theta}) \propto \sum_{i=1}^{3} \frac{1}{\sqrt{(2\pi)^2|\Sigma_i|}} \exp\left(-\frac{1}{2}(\boldsymbol{\theta}-\boldsymbol{\mu}_i)^\top \Sigma_i^{-1}(\boldsymbol{\theta}-\boldsymbol{\mu}_i)\right), \tag{2.70}$$

where

$$\boldsymbol{\mu}_1 = \begin{bmatrix}4\\2\end{bmatrix}, \quad \boldsymbol{\mu}_2 = \begin{bmatrix}3\\-2\end{bmatrix}, \quad \boldsymbol{\mu}_3 = \begin{bmatrix}-4\\0\end{bmatrix}, \tag{2.71}$$

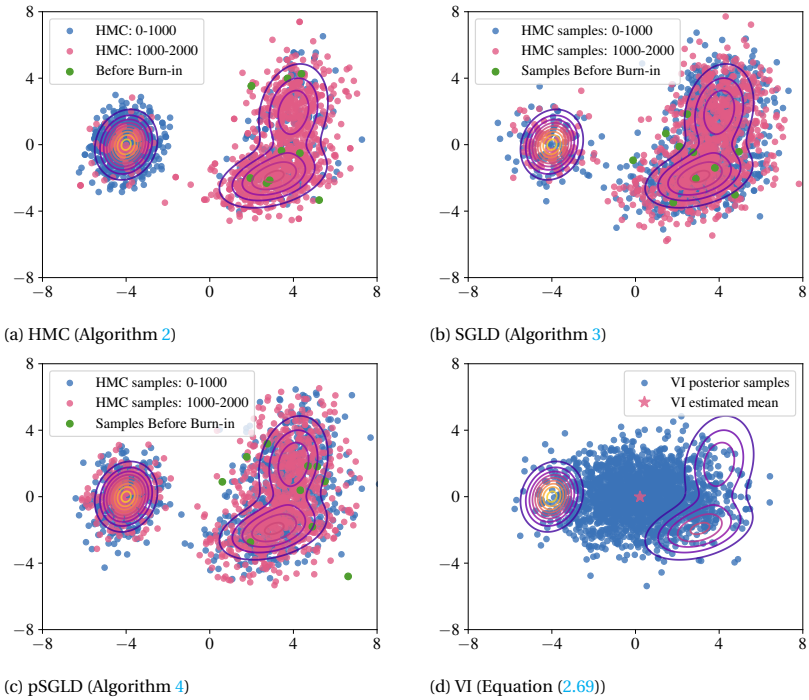
$$\Sigma_1 = \begin{bmatrix}1 & 1/3\\1/3 & 3\end{bmatrix}, \quad \Sigma_2 = \begin{bmatrix}2 & 1/2\\1/2 & 1\end{bmatrix}, \quad \Sigma_3 = \begin{bmatrix}1/2 & 1/10\\1/10 & 1\end{bmatrix}. \tag{2.72}$$

The MCMC-based sampling methods are primarily controlled by several essential parameters: step size, leap-frog steps, burn-in iterations, and the number of posterior samples. It is worth noting that, in theory, SGLD and pSGLD can set the leap-frog step to 1. However, the original papers [50, 52] reported that using a larger leap-frog step can improve the mixing rate (convergence speed). Therefore, we set the leap-frog step to 20 for all MCMC methods. Furthermore, the step size, burn-in iterations, and number of posterior samples are set to 1.0, 100, and 2000, respectively. With this setting for those MCMC methods, it will cost $100 + 20 \times 2000 = 40010$ PDF evaluations of Equation (2.70). Therefore, I also generate the same amount of true posterior samples for optimizing the ELBO (Equation (2.69)); then, use *Adam* to optimize it for 2000 epochs with a learning rate of 0.01. The comparison results of the four approximate inference methods, averaged over 10 random runs, are summarized in Table 2.2, where the KL divergence is adopted as the evaluation metric. Moreover, the wall time is also recorded to illustrate on their efficiency concerning computational time. A visualization of representative results from a single run of each method are plotted in Figure 2.9.

As shown in both Table 2.2 and Figure 2.9, the MCMC-based sampling methods exhibit significantly better performance than VI due to the multi-modal nature of the problem. The approximated samples from all three MCMC-based methods are clearly concentrated in high-PDF regions. In Figure 2.9a, Figure 2.9b, and Figure 2.9c, the green

Method	KL value (↓)	Acceptance ratio	Wall time (s)
HMC	0.3389 (0.3112)	0.8668 (0.0337)	17.13 (0.06)
SGLD	0.2187 (0.0404)	1.0000 (0.0000)	16.76 (0.00)
pSGLD	0.0695 (0.0294)	1.0000 (0.0000)	18.77 (0.00)
VI	2.7872 (0.0650)	–	3.00 (0.10)

Table 2.2: Comparison results of different inference methods on the two-dimensional Gaussian mixture distribution over 10 random runs. For each metric, the mean and standard deviation (shown in parentheses) are reported.



(a) HMC (Algorithm 2)

(b) SGLD (Algorithm 3)

(c) pSGLD (Algorithm 4)

(d) VI (Equation (2.69))

Figure 2.9: Representative posterior approximation from a single run of each inference method. The contour illustrates the PDF defined in Equation (2.70). For the MCMC-based sampling methods, the green points represent the first 10 samples obtained before the burn-in phase, while the blue and pink points correspond to the first and second halves of the posterior samples, respectively. In the case of VI, a parametric Gaussian distribution is learned, where the pink star denotes the approximated mean and the blue points are samples drawn from the estimated distribution.

points correspond to *stage 1* in Figure 2.7, during which the sampler explores the PDF landscape. Once this exploratory phase is complete, the samplers efficiently generate samples in high-PDF areas.

The transition from *stage 2* to *stage 3* is less straightforward to identify. Therefore, I present the first and second halves of the approximated samples, which reveal a tendency for the later-half samples (pink points) to be closer to the modes. Among the MCMC-

based methods, pSGLD achieves the most accurate and robust results, albeit slightly slower due to the computation of preconditioning in Equation (2.64). Although HMC provides a theoretical guarantee of convergence to the ground truth under the assumption of unlimited posterior samples, its performance in this example is less ideal: several runs capture only one or two of the three modes, as indicated by the large standard deviation of the KL divergence. The accept–reject procedure is another bottleneck for HMC. A low acceptance rate implies the sampler becomes stuck, whereas an excessively high acceptance rate suggests insufficient exploration of the PDF. For this two-dimensional problem, the acceptance rate is approximately 0.8668, but this value would drop substantially as the dimensionality increases.

In contrast, VI approximates the true posterior using a parametric distribution. However, as shown in Figure 2.9d, it struggles with multi-modal problems: the estimated mean is located between the three modes, with a variance broad enough to cover all high-PDF areas. Despite this limitation, VI is extremely fast compared to MCMC-based methods, making it a promising option for rapid approximation.

Posterior of Bayesian neural networks In this section, I demonstrate the performance of the four inference methods on high-dimensional problems. Specifically, I employ a BNN with two hidden layers of 256 neurons each, activated by the *tanh* function, to address a one-dimensional regression problem defined as:

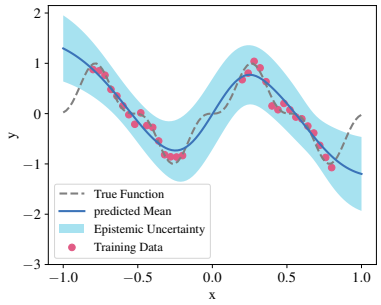
$$y = \sin^3(6x) + \varepsilon, \quad \varepsilon \sim \mathcal{N}\left(0, \sigma_{\mathrm{a}}^2\right), \tag{2.73}$$

where $\sigma_{\mathrm{a}} = 0.1^2$ is known, representing homoscedastic data noise.

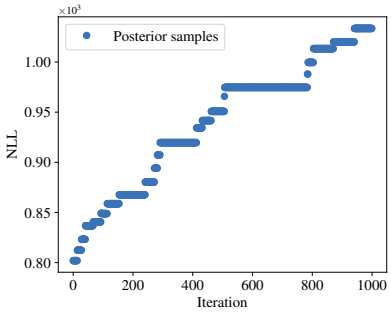
The fitting performance of each inference method is illustrated in Figure 2.10. A total of 32 training points are used, with half uniformly distributed in $[-0.8, -0.2]$ and the other half in $[0.2, 0.8]$. Given the chosen neural architecture, the BNN contains 66561 neural parameters treated as random variables. For MCMC-based sampling methods, the step size is set to be 0.001, the burn-in epoch is set to be 500, and 1000 posterior samples are used with a leap-frog step of 20. These methods end up with 20500 epochs. As for VI, *Adam* is used to optimize the ELBO for 20000 epochs considering similar budget with MCMC-based methods.

The first row presents the results of HMC: Figure 2.10a shows the predicted mean (blue solid line) compared with the ground truth mean (gray dashed line), along with the corresponding epistemic uncertainty confidence interval, while Figure 2.10b illustrates the NLL of the posterior samples. The acceptance ratio is very low with a value of 0.02, which indicates HMC gets stuck for this problem, as also indicated in Figure 2.10b by the clear plateaus for the NLL. As a result, the prediction obtained with HMC in Figure 2.10a leads to poor estimates of the epistemic uncertainty, as it predicts almost the same confidence interval everywhere and the predicted mean has large discrepancy compared with the ground truth mean.

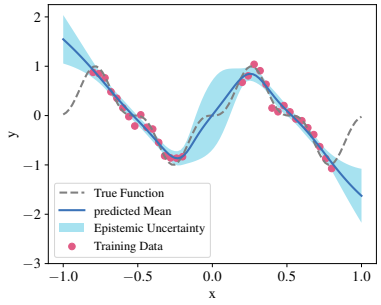
The second row presents the results of SGLD, which shows improved performance compared to HMC, with the predicted mean being closer to the ground truth as shown in Figure 2.10e. More importantly, the predicted epistemic uncertainty is more informative: it is smaller in regions containing training points and expands in regions without training data. Since SGLD omits the accept/reject step, the sampling process becomes



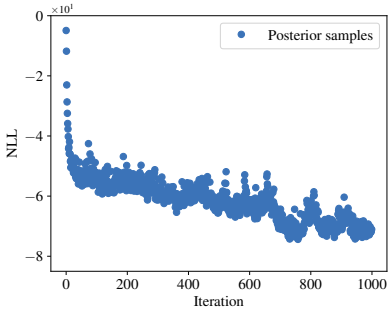
(a) HMC (Acceptance ratio=0.02)



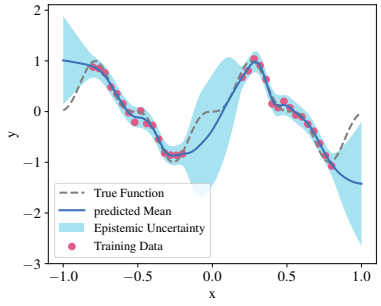
(b) NLL of posterior samples with HMC



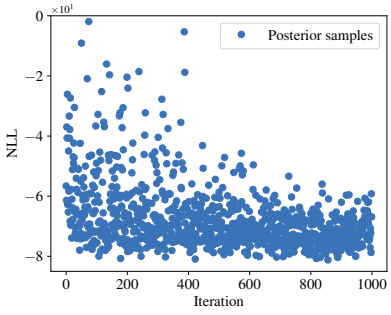
(c) SGLD (Algorithm 3)



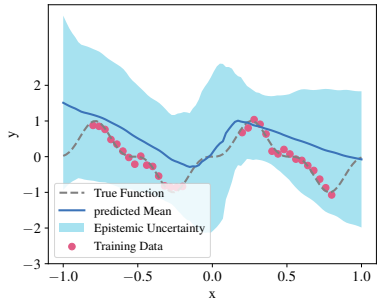
(d) NLL of posterior samples with SGLD



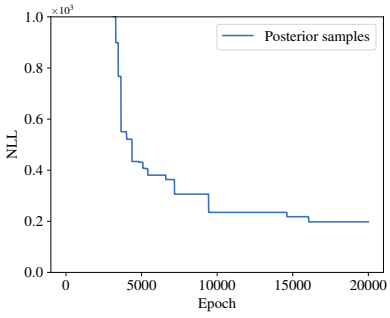
(e) pSGLD (Algorithm 4)



(f) NLL of posterior samples with pSGLD



(g) VI (Equation (2.69))



(h) NLL historical records with VI

Figure 2.10: Performance comparison of HMC, SGLD, pSGLD, and VI for a one-dimensional BNN regression task.

computationally more efficient, and the plateaus observed in HMC disappear, as shown in Figure 2.10d.

The third row presents the results of pSGLD, which show a further improvement in prediction compared to SGLD (Figure 2.10e vs.Figure 2.10c). In particular, the predicted mean almost overlaps with the ground truth mean, even with only 32 training points. Moreover, the predicted epistemic uncertainty is more informative, exhibiting a clear distinction between sampled and unsampled regions. On the other hand, the corresponding NLL values are lower and more widely spread, indicating better integration compared to the other two MCMC-based sampling methods.

The final row shows the performance of VI, which is less satisfactory compared to the MCMC-based methods. Specifically, the final prediction converges to a case with large epistemic uncertainty. Due to the overconfident estimation of epistemic uncertainty, the predicted mean deviates from the ground truth, as shown in Figure 2.10g. Figure 2.10h presents the historical NLL over the optimization epochs, where the final result after 2000 epochs converges to an NLL value of around 200 — much higher than the averages observed in Figure 2.10d and Figure 2.10f. This performance gap arises because VI assumes that each neural parameter follows an independent Gaussian distribution, resulting in 66,561 such distributions. Consequently, optimizing the ELBO with only 32 training points becomes challenging, making it difficult to achieve a satisfactory result.

2.6. CONCLUSION

This chapter lays the foundation of regression from a Bayesian perspective, covering models ranging from simple linear models to complex neural networks. A concise summary of these models is provided to offer a fair basis for model selection in subsequent applications. Furthermore, inference methods are briefly introduced, as they are essential for Bayesian neural networks to obtain the predictive posterior distribution. Finally, several examples are presented to compare and highlight the advantages and disadvantages of different inference methods.

3

Cooperative variance estimation and Bayesian neural networks disentangle aleatoric and epistemic uncertainties

Real-world data contains aleatoric uncertainty – irreducible noise arising from imperfect measurements or from incomplete knowledge about the data generation process. Mean variance estimation (MVE) networks can learn this type of uncertainty but require ad-hoc regularization strategies to avoid overfitting and are unable to predict epistemic uncertainty (model uncertainty). Conversely, Bayesian neural networks predict epistemic uncertainty but are notoriously difficult to train due to the approximate nature of Bayesian inference. We propose to cooperatively train a variance network with a Bayesian neural network and empirically demonstrate that the resulting model disentangles aleatoric and epistemic uncertainties while improving the mean estimation. We demonstrate the effectiveness and scalability of this method across a diverse range of datasets, including a time-dependent heteroscedastic regression dataset we created where the aleatoric uncertainty is known. The proposed method is straightforward to implement, robust, and adaptable to various model architectures.

This chapter is based on the manuscript: **Yi, J.**, & Bessa, M. A. (2025). *Cooperative Bayesian and Variance Networks Disentangle Aleatoric and Epistemic Uncertainties.* arXiv preprint arXiv:2505.02743. Under Review
https://arxiv.org/abs/2505.02743

3.1. INTRODUCTION

Non-probabilistic neural networks that only estimate the mean (expected value) tend to be overconfident and vulnerable to adversarial attacks [54, 55]. Quantifying aleatoric (or data) uncertainty alleviates these issues by characterizing noise [56, 57]. Estimating epistemic (or model) uncertainty enables active learning and risk-sensitive decision-making [17, 58]. Consequently, except in cases with negligible or homoscedastic data noise, simultaneously predicting aleatoric and epistemic uncertainties is essential for a wide range of safety-critical applications [16]. In such cases, an outcome with good mean performance but large aleatoric uncertainty may be unacceptable. Therefore, the principle of reducing epistemic uncertainty behind active learning or decision-making needs to be balanced by the respective prediction of aleatoric uncertainty. This is particularly important when the aleatoric uncertainty is heteroscedastic (input-dependent) due to imperfect measurements, environmental variability, and other factors [19].

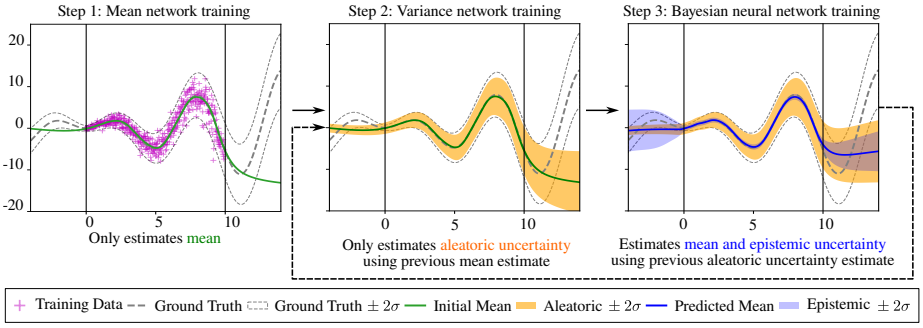


Figure 3.1: Illustration of the proposed cooperative training strategy leading to Variance estimation Bayesian Neural Networks (VeBNNs). The left figure shows the unseen ground truth mean as well as the respective training data. The method starts by training the mean network to only estimate the mean (green solid line in Step 1). Then, without updating the mean estimate, a variance network is trained to only predict aleatoric uncertainty (orange credible interval in Step 2). Subsequently, considering this aleatoric uncertainty estimation, a Bayesian neural network is trained to obtain an updated mean and corresponding epistemic uncertainty (solid blue line for the new mean, and shaded blue credible interval for the epistemic uncertainty in Step 3). If needed, the method can iterate between steps 3 and 2 to improve the disentanglement of uncertainties. Note the disentanglement of uncertainties together with the improvement of the mean estimation away from the data support ($x < 0$ and $x > 10$ in Step 3).

We propose a cooperative learning strategy for uncertainty disentanglement based on sequential training of (1) a mean network, (2) a variance network, and (3) a Bayesian neural network. Figure 3.1 illustrates the method for one-dimensional heteroscedastic regression, briefly describing it at the figure caption.

3.2. RELATED WORK

3.2.1. ALEATORIC UNCERTAINTY

Aleatoric uncertainty can be estimated by parametric or nonparametric models. The latter do not explicitly define the likelihood function and instead focus on learning to sample from the data distribution [59]. Their ability to estimate nontrivial aleatoric uncertainty comes at the cost of training difficulties and sampling inefficiencies [60,

61]. Therefore, parametric models are more common. They assume a parameterized observation distribution, usually a Gaussian as in Mean Variance Estimation (MVE) networks [62], and learn the corresponding parameters by minimizing the Negative Log-Likelihood (NLL) loss with associated regularization. Similar parametric models have been developed, replacing the Gaussian distribution with other distributions [63]. However, training these models can be challenging. MVE networks have been observed to lead to good mean but overconfident variance estimations in regions within the data support, and exhibit generalization issues outside these regions [56]. As analyzed by different authors [56, 57, 64, 65], these issues result from the loss gradients having very different magnitudes when calculated with respect to the mean or the aleatoric variance, as seen in Equation (3.3), which creates imbalances when minimizing the NLL.

Different solutions have been proposed to improve the estimation of aleatoric uncertainty, including a Bayesian treatment of variance [66], calibration by distribution matching via maximum mean discrepancy loss [67], or modifying the loss function to include an additional balance hyperparameter [57]. However, [65] demonstrated that training an MVE network that simultaneously predicts mean and variance leads to significantly worse predictions for both estimations compared to separately training, even when considering the above-mentioned modified losses (we independently reached the same conclusion while conducting our work; see Appendix B.1). Nevertheless, we note that parametric deterministic models are insufficient if they only estimate mean and aleatoric uncertainty, without accounting for epistemic uncertainty. This is also visible in Figure 3.1 (Step 2), as the model has an overconfident mean outside the data support (see $x > 10$). Furthermore, the inability to estimate epistemic uncertainty is problematic in itself, as discussed in Section 3.1.

3.2.2. EPISTEMIC UNCERTAINTY

Epistemic uncertainty is usually estimated by probabilistic models that impose a prior distribution on the model parameters [15]. Instead of finding point estimates as in deterministic models, they predict a posterior predictive distribution (PPD). Unfortunately, accurate and computationally tractable determination of the PPD is challenging in most cases, except for a few models like Gaussian processes, where inference can be done exactly under strict assumptions and with limited data scalability [44]. Bayesian neural networks (BNNs) are more scalable, but the accuracy and scalability are strongly dependent on the type of Bayesian inference [68–70].

Bayesian inference is commonly done by Markov Chain Monte Carlo (MCMC) sampling methods [21, 50, 52] or VI methods that are faster to train but less accurate [71, 72]. Avoiding formal Bayesian inference is also possible by adopting ensemble methods such as Monte Carlo (MC) Dropout [23] and deep ensembles [22, 73]. Although not strictly Bayesian, MC-Dropout can be interpreted as a Variational Bayesian approximation that has additional scalability but even lower accuracy (no free lunch).

The practical difficulties of training BNNs have limited their widespread use. Therefore, they are often trained by disregarding aleatoric uncertainty or assuming homoscedastic noise, which can be set as a hyperparameter but is impractical for complex problems [15]. Instead, the end-to-end training (also referred to as *second order* uncertainty quantification) that trains an MVE network by MC-Dropout [17] or deep ensembling [74]

is reported to approximately disentangle aleatoric and epistemic uncertainties for heteroscedastic regression and classification. However, the essential challenges faced when training MVE networks are not solved by ensembling them, and the lack of accuracy associated with MC-Dropout raises questions about their ability to make high-quality predictions and truly disentangle epistemic and aleatoric uncertainties [75, 76]. This has motivated other authors to propose different solutions. A non-Bayesian solution to separate uncertainties was proposed by introducing a high-order evidential distribution, i.e., considering priors over the likelihood function instead of over network weights [77]. Another proposal has been to use a natural reparameterization combined with an approximate Laplace expansion to estimate epistemic uncertainty [64]. Still, a recent investigation [76] suggests that no current method achieves reliable uncertainty disentanglement.

3.3. METHODOLOGY

3.3.1. PRELIMINARIES

Consider a dataset $\mathscr{D} = \{\mathbf{x}_n, y_n\}_{n=1}^N$ with $i.i.d.$ data points, where $\mathbf{x}_n \in \mathbb{R}^d$ represents the input features and $y_n \in \mathbb{R}$ the corresponding outputs[1]. The heteroscedastic regression problem can be formulated as:

$$y = f(\mathbf{x}) + s(\mathbf{x}) \tag{3.1}$$

where $f(\mathbf{x})$ denotes the underlying noiseless ground truth (expected mean), and $s(\mathbf{x})$ is the corresponding heteroscedastic noise (aleatoric uncertainty). If the noise is Gaussian, then $s(\mathbf{x}) \sim \mathscr{N}(0, \varepsilon^2(\mathbf{x}))$ where $\varepsilon^2(\mathbf{x})$ represents its ground truth input-dependent variance.

3.3.2. CHALLENGES OF ENSEMBLING TRAINING OF MVES

The NLL loss resulting from a Gaussian observation distribution with heteroscedastic aleatoric uncertainty is:

$$\mathscr{L}_1(\boldsymbol{\theta}) = \sum_{n=1}^{N} \left[\frac{(y_n - \mu(\mathbf{x}_n;\boldsymbol{\theta}))^2}{2\sigma_a^2(\mathbf{x}_n;\boldsymbol{\phi})} + \frac{\log(\sigma_a^2(\mathbf{x}_n;\boldsymbol{\phi}))}{2} \right] \tag{3.2}$$

where $\sigma_a^2(\mathbf{x};\boldsymbol{\phi}) > 0$ is the parameterized aleatoric variance, and $\mu(\mathbf{x};\boldsymbol{\theta})$ the mean. The derivatives with respect to $\mu(\mathbf{x};\boldsymbol{\theta})$ and $\sigma_a^2(\mathbf{x};\boldsymbol{\phi})$ become:

$$\nabla_\mu \mathscr{L}_1 = \sum_{n=1}^{N} \left(\frac{\mu(\mathbf{x}_n;\boldsymbol{\theta}) - y_n}{\sigma_a^2(\mathbf{x}_n;\boldsymbol{\phi})} \right), \nabla_{\sigma_a^2} \mathscr{L}_1 = \sum_{n=1}^{N} \left(\frac{\sigma_a^2(\mathbf{x}_n;\boldsymbol{\phi}) - (y_n - \mu(\mathbf{x}_n;\boldsymbol{\theta}))^2}{\left(\sigma_a^2(\mathbf{x}_n;\boldsymbol{\phi})\right)^2} \right). \tag{3.3}$$

According to eq. (3.3), the gradient with respect to $\sigma_a^2(\mathbf{x};\boldsymbol{\phi})$ has a high-order term in the denominator that makes the optimization more challenging and causes an imbalance when minimizing the loss. As a result, the end-to-end training with ensembling MVE networks would include the following unexpected cases: (1) if the MVE network overfits

[1] The equations become simpler when writing for one-dimensional outputs, but this article includes examples with multi-dimensional outputs $\mathbf{y}_n \in \mathbb{R}^m$, as shown later.

the response, i.e., $(y_n - \mu(\mathbf{x}_n;\boldsymbol{\theta}))^2 \to 0$, $\sigma_a^2(\mathbf{x}_n;\boldsymbol{\phi}) \to 0$, the loss function $\mathscr{L}_1$ may become undefined; (2) if the MVE network leads to large variance estimates when quantifying aleatoric uncertainty, then $\sigma_a^2(\mathbf{x}_n;\boldsymbol{\phi}) \to \infty$ and the gradients with respect to $\sigma_a^2(\mathbf{x}_n;\boldsymbol{\phi})$ vanish. The loss function may also contain multiple saddle points that translate into low quality point estimates for the parameters and slow convergence, as training may become trapped in degenerate regions. While balancing the loss is possible, as proposed in [57] by the β-NLL loss, this introduces additional hyperparameters and still does not address the difficulty in conducting Bayesian inference when trying to simultaneously predict aleatoric and epistemic uncertainties [76].

3.3.3. PROPOSED VEBNN

We propose a sequential training strategy (see algorithm 5) that starts with training a mean network, then a variance estimation network that predicts aleatoric uncertainty, followed by a BNN that updates both mean and epistemic uncertainty for the previously determined aleatoric uncertainty. By separating the roles of each network and ensuring their cooperative training, we empirically demonstrate that the resulting BNN can learn and improve all three estimates. Importantly, training each network separately is easier, and we show that inference of the BNN is also facilitated due to the presence of a good estimate of aleatoric uncertainty (that is not being learned at that stage).

Algorithm 5: Cooperative VeBNN Training

Data: Mean network $\mu(\mathbf{x};\boldsymbol{\theta})$, variance network $\sigma_a^2(\mathbf{x};\boldsymbol{\phi})$, training data $\mathscr{D} = \{\mathbf{x}, \mathbf{y}\}$, total iterations K

Result: Optimized parameters $\boldsymbol{\Theta}^*$ and $\boldsymbol{\phi}^*$

Step 1: Mean network training: Train deterministic mean network $\mu(\mathbf{x};\boldsymbol{\theta})$ by minimizing Equation (3.2)

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

  Step 2: Variance network training (Aleatoric uncertainty) Minimize Equation (3.6) using fixed mean from **Step 1** or **Step 3**

  Step 3: Bayesian network training (Epistemic uncertainty) Sample posterior from Equation (3.9) to determine mean and epistemic variance estimates for fixed aleatoric variance from Equation (3.8); and compute the log marginal likelihood: $\texttt{LMglk}[i] = \log \mathbb{E}_{p(\boldsymbol{\theta}|\mathscr{D})}\left[p(\boldsymbol{y} \mid \boldsymbol{\theta}^{(i)}, \boldsymbol{\phi}^{(i)})\right]$

end

Identify the optimal parameters for $\boldsymbol{\Theta}^*$ and $\boldsymbol{\phi}^*$ by $i^* = \arg\max_i \texttt{LMglk}[i]$

The proposed Algorithm 5 starts by not considering aleatoric uncertainty, i.e. $\sigma_a^2(\mathbf{x};\boldsymbol{\phi}) =$ constant, and conventionally trains the mean network by finding the maximum a posteriori (MAP) estimate of the parameters $\boldsymbol{\theta}$ of Equation (3.2), hence determining only the mean.

3.3.4. VARIANCE ESTIMATION NETWORK TRAINING

Once the mean is obtained, we then train the variance estimation (Ve) network for this fixed mean. There is, however, an important detail that facilitates training of this network (Step 2 in Algorithm 5). The VE network does not directly output $\sigma_a^2(\mathbf{x};\boldsymbol{\phi})$, and so its

parameters are not directly determined by minimizing Equation (3.2) and keeping the mean fixed. Instead, the variance network outputs the residual $r = \big(\mu(\mathbf{x};\boldsymbol{\theta}) - y\big)^2$, which follows a Gamma distribution because y is Gaussian.

Assumption 3.3.1. *$\big(\mu(\mathbf{x};\boldsymbol{\theta}) - f(\mathbf{x})\big)^2$ is finite and tends to 0 when $N \to \infty$. This follows from assuming unbiased or consistent estimations for the ground truth $f(\mathbf{x})$, regardless of noise.*

Lemma 3.3.1. *Under Assumption 3.3.1 and assuming that y follows a Gaussian distribution, the squared residual $r = (\mu(x;\theta) - y)^2$ follows a Gamma distribution.*

Proof. Since $y \mid \mathbf{x} \sim \mathcal{N}(f(\mathbf{x}), \varepsilon^2(\mathbf{x}))$, the residual $\mu(\mathbf{x};\boldsymbol{\theta}) - y \sim \mathcal{N}(0, \varepsilon^2(\mathbf{x}))$. By standardizing, it is easy to obtain:

$$Z = \frac{\mu(\mathbf{x};\boldsymbol{\theta}) - y}{\varepsilon(\mathbf{x})} \sim \mathcal{N}(0,1), \tag{3.4}$$

Thus, the squared residual can be obtained $r = \big(\mu(\mathbf{x};\boldsymbol{\theta}) - y\big)^2 = \varepsilon^2(\mathbf{x}) Z^2$. Since $Z \sim \mathcal{N}(0,1)$, we have $Z^2 \sim \chi^2(1)$, a specific case of Gamma distribution $Z^2 \sim \text{Gamma}\left(\frac{1}{2}, \frac{1}{2}\right)$. Moreover, a Gamma random variable $W \sim \text{Gamma}(\alpha, \lambda)$ scaled by a constant $c > 0$ gives $cW \sim \text{Gamma}\left(\alpha, \frac{\lambda}{c}\right)$, then:

$$r \sim \text{Gamma}\left(\frac{1}{2}, \frac{1}{2\varepsilon^2(\mathbf{x})}\right) \tag{3.5}$$

Showing that the mean of the Gamma distribution becomes $\frac{\alpha}{\lambda} = \varepsilon^2(\mathbf{x})$, i.e., the aleatoric variance.

□

To this end, we propose the Gamma likelihood to model the squared residual r, leading to the corresponding NLL loss to train the variance network:

$$\mathscr{L}_2(\boldsymbol{\phi}) = \sum_{n=1}^{N} \Big[\log\Gamma\big(\alpha(\mathbf{x}_n;\boldsymbol{\phi})\big) - \alpha(\mathbf{x}_n;\boldsymbol{\phi}) \log\lambda(\mathbf{x}_n;\boldsymbol{\phi}) - \big(\alpha(\mathbf{x}_n;\boldsymbol{\phi}) - 1\big) \log r_n + \lambda(\mathbf{x}_n;\boldsymbol{\phi})\, r_n \Big]. \tag{3.6}$$

where r are the above-mentioned residuals, and with the shape and rate parameters of the Gamma distribution, $\alpha(\mathbf{x};\boldsymbol{\phi}) > 0$ and $\lambda(\mathbf{x};\boldsymbol{\phi}) > 0$, being the outputs of the network. These parameters are estimated via MAP, and their gradients with respect to α and λ contain no high-order terms, as shown below:

$$\nabla_{\alpha(\mathbf{x})}\mathscr{L}_2 = \sum_{n=1}^{N} \Big[\psi\big(\alpha(\mathbf{x}_n)\big) - \log\lambda(\mathbf{x}_n) - \log r_n \Big], \quad \nabla_{\lambda(\mathbf{x})}\mathscr{L}_2 = \sum_{n=1}^{N} \Big[-\frac{\alpha(\mathbf{x}_n)}{\lambda(\mathbf{x}_n)} + r_n \Big]. \tag{3.7}$$

The expected value of the Gamma distribution becomes the desired heteroscedastic variance:

$$\sigma_a^2(\mathbf{x};\boldsymbol{\phi}) = \frac{\alpha(\mathbf{x};\boldsymbol{\phi})}{\lambda(\mathbf{x};\boldsymbol{\phi})} \tag{3.8}$$

3.3.5. BAYESIAN NEURAL NETWORK TRAINING

Having determined the mean and aleatoric variance estimates, we then train a BNN with a warm-start for the mean, and assuming fixed aleatoric variance that is obtained from Equation (3.8). In principle, the same network architecture can be used for the BNN and the mean network[2]. The posterior of the BNN is determined by the Bayes rule, and it is proportional to the product of likelihood and prior:

$$p(\boldsymbol{\theta} \mid \mathscr{D}) \propto p(\mathscr{D} \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta}) \tag{3.9}$$

where $p(\boldsymbol{\theta})$ is the prior over the neural network parameters, and $p(\mathscr{D} \mid \boldsymbol{\theta})$ is the likelihood for the observations. In this chapter, I consider a Gaussian prior with zero mean and unit variance, and the likelihood is given by Equation (3.2), i.e. it arises from a Gaussian observation distribution with heteroscedastic noise. The logarithm of the posterior becomes:

$$\log p(\boldsymbol{\theta} \mid \mathscr{D}) = \sum_{n=1}^{N} \left[\log \frac{1}{\sqrt{2\pi\sigma_a^2(\mathbf{x}_n;\boldsymbol{\phi})}} - \frac{\big(\mathbf{y}_n - \mu(\mathbf{x}_n;\boldsymbol{\theta})\big)^2}{2\sigma_a^2(\mathbf{x}_n;\boldsymbol{\phi})} \right] + m \log \frac{1}{\sqrt{2\pi/\kappa}} - \frac{\kappa}{2}\|\boldsymbol{\theta}\|^2 \tag{3.10}$$

where κ is the precision of the prior distribution and m is length of $\boldsymbol{\theta}$. Note that Equation (3.10) is the same as Equation (3.2), but now we explicitly include the prior terms. Section 2.5 summarizes common Bayesian inference strategies, including the MCMC-based methods that sample directly from Equation (3.10), and VI methods that minimize the evidence lower bound (ELBO) (Section 2.5.2 and Section 2.5.3, respectively). From experience, preconditioned Stochastic Gradient Langevin Dynamics (pSGLD) [52] is expected to be a good choice for this BNN because the aleatoric uncertainty is fixed, and the likelihood and prior are both Gaussian, leading to a Gaussian posterior. Importantly, since the aleatoric variance $\sigma_a^2(\mathbf{x}_n;\boldsymbol{\phi})$ is fixed from step 2 and is regarded as constant, it converges to the correct mode $\mu(\mathbf{x}_n;\boldsymbol{\theta})$, and has better and faster convergence (no gradient issue).

From the PPD of the BNN, we then estimate the mean, aleatoric, and epistemic variances (disentangled) for any unseen point $\mathbf{x}'$ based on Bayesian model averaging:

$$\mathcal{N}\left(\mathbf{y}' \,\middle|\, \underbrace{\mathbb{E}_{p(\boldsymbol{\theta}|\mathscr{D})}\big[\mu(\mathbf{x}';\boldsymbol{\theta})\big]}_{\text{Predictive Mean}}, \underbrace{\sigma_a^2(\mathbf{x}';\boldsymbol{\phi})}_{\text{Aleatoric}} + \underbrace{\mathbb{V}_{p(\boldsymbol{\theta}|\mathscr{D})}\big[\mu(\mathbf{x}';\boldsymbol{\theta})\big]}_{\text{Epistemic}}\right) \tag{3.11}$$

3.4. EXPERIMENTS

Datasets We consider four distinct sets of datasets (a total of 18 datasets): (1) the previously discussed one-dimensional illustrative example [56] (2) UCI regression datasets [78]; (3) large-scale image regression datasets [79]; and (4) our own dataset obtained from computer simulations of materials undergoing history-dependent deformations (material plasticity law discovery dataset) with known ground-truth of aleatoric uncertainty. Readers interested in more details about the dataset are referred to Appendix B.5.

[2] Estimating epistemic uncertainty with BNNs can require a network with wider hidden layers when compared to a deterministic network that only estimates the mean. So, choosing a smaller mean network in Step 1 is also possible. However, we like the simplicity of not introducing a new network.

Baselines We compare against representative methods: (1) **ME (MSE)**: standard Mean Estimation network; (2) **MVE (β-NLL)**: MVE using the β-NLL loss [57], which is capable of simultaneous prediction of mean and aleatoric uncertainty. Concerning methods that aim at disentangling uncertainties, we considered (3) **Evidential**: Deep Evidential regression [77]; (4) **MVE (Ensembles)**: ensembling [22] MVE networks; (5) **MVE (MC-Dropout)**: MC-Dropout [17] training for MVE networks. (6) **BNN-End-to-End**: end-to-end training of BNN with different inference methods, i.e., **pSGLD** [52] and Bayes By Backpropagation (**BBB**) [72], indicated in a parenthesis for every Figure and Table. (7) **BNN-Homo**: BNN training with homoscedastic noise assumption. Finally, our proposed **VeBNN** method is also trained with all the above-mentioned inference methods.

Details about accuracy metrics, additional results, and training are presented in Appendix B.2, Appendix B.3, and Appendix B.4, respectively.

3.4.1. ILLUSTRATIVE EXAMPLE: ONE-DIMENSIONAL DATASET

Comparison of different inference methods The predictions for the one-dimensional example by our method with pSGLD inference – VeBNN (pSGLD) – are shown in Figure 3.1. A direct comparison with MVE (β-NLL) assuming the best value for the β hyperparameter that we found, $\beta = 0.5$, is shown Figure 3.2. It is clear that our strategy of separately training the mean and aleatoric variance leads to better results, and avoids the need for an extra hyperparameter (β).

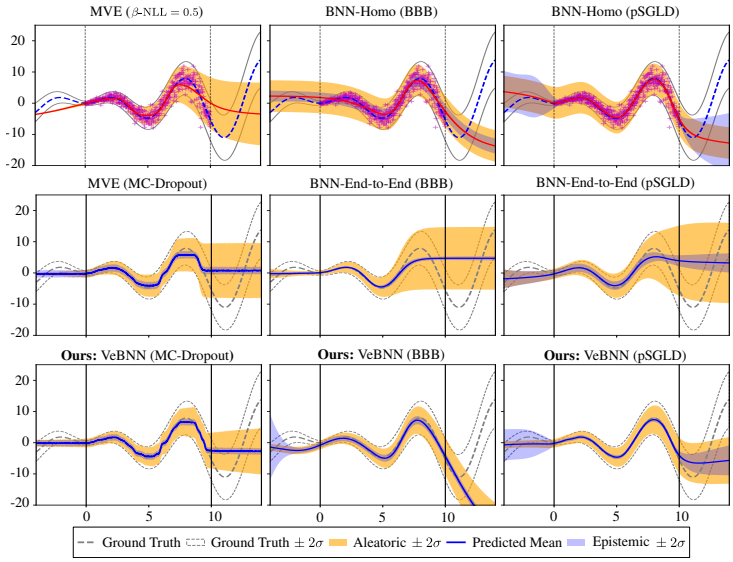


Figure 3.2: Heteroscedastic regression by our method (bottom) compared to existing BNN training with homoscedastic assumption (top) and End-to-End training methods (middle) for each inference type.

Figure 3.2 also shows the same example when compared to other Bayesian methods with homoscedastic and heteroscedastic assumptions that are capable of predicting both uncertainties. We see a consistent improvement in the predictions with the cooperative training strategy we proposed, independently of the inference method that

is chosen. Specifically, BNN training with homoscedastic assumption lead to inferior results on the aleatoric uncertainty estimation, since a constant variance σ_a^2 is not adequate. Furthermore, the End-to-End training strategy starts to overestimate aleatoric uncertainty approximately after $x > 8$, regardless of the Bayesian inference method. This effect becomes more pronounced in the extrapolation region ($x > 10$), which reflects the difficulties that arise from the joint training process. There is a clear tendency to overestimate the aleatoric uncertainty by the End-to-End strategy that also affects the mean estimation. In contrast, the proposed strategy improves predictions according to all metrics. As expected, Bayesian inference with pSGLD is the best.

Heteroscedastic noise We show the regression results for increasing the number of training data points from 5 to 500 based on the best inference method pSGLD. The accuracy metrics obtained by running each case 5 times for randomly sampled training sets are shown in Figure 3.3 and we also visualize the fitting performance of selected methods under an arbitrary run in Figure 3.4.

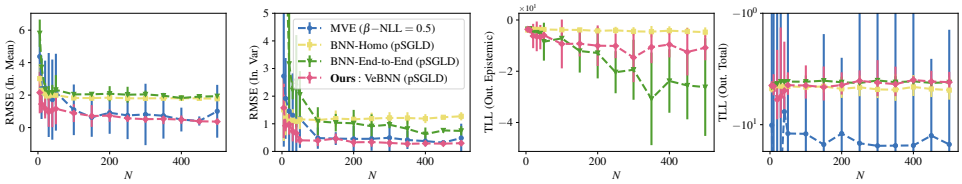


Figure 3.3: RMSE and TLL convergence curves under 5 different seeds for heteroscedastic data generation. The first figure shows the RMSE between the data values and predictive mean within the interpolation region; the second figure shows the RMSE between the noise standard deviation and the predictive one in the interpolation region since the ground truth of aleatoric uncertainty is known; and the third and the fourth figures show the Epistemic TLL and Total TLL values for extrapolation test points.

The optimal value for the hyperparameter β was 0.5. The experimental results reveal that MVE with $\beta = 0.5$ converges to the same mean but slower than the proposed VeBNN (pSGLD). Meanwhile, BNN-Homo (pSGLD) performs well when predicting the mean within the interpolation region, comparable to the proposed VeBNN (pSGLD). However, it fails to predict the aleatoric uncertainty with a single σ_a^2 tuned by grid search. BNN-End-to-End (pSGLD) has difficulty in converging to either mean or aleatoric uncertainty. In contrast, The proposed strategy successfully combines the strengths of MVE and BNN to effectively model aleatoric uncertainty, mean predictions, and epistemic uncertainty. The results highlight the effectiveness of cooperative training instead of End-to-End training, especially in the case of data-scarce scenario. Notably, the proposed method maintains a stable TLL value in the extrapolation region, outperforming alternative approaches in this critical area.

Homoscedastic noise We execute a similar experiment to test homoscedastic noise cases where the data size changes from 5 to 200. Figure 3.5 and Figure 3.6 highlight similar conclusions as Section 3.4.1. Again, the MVE training experiences great difficulty when the training data is scarce. BNN-End-to-End (pSGLD) gives really large uncertainties in

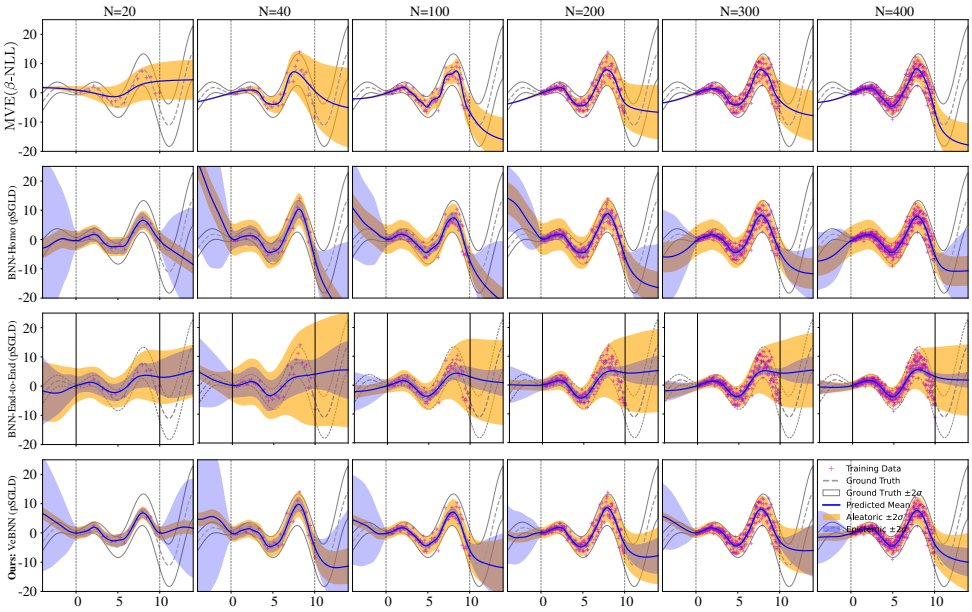


Figure 3.4: Predictions of MVE (β-NLL) with $\beta = 0.5$, BNN-Homo (pSGLD), BNN-End-to-End (pSGLD) and **Ours:** VeBNN (pSGLD) with different data points under heteroscedastic data noise. In this plot, the first to fourth rows represent the prediction of MVE (β-NLL = 0.5), BNN-Homo (pSGLD), BNN-End-to-End (pSGLD), and **Ours:** VeBNN (pSGLD) with different training data points under the same seed for data generation.

the cases of $N < 30$, suggesting that it tends to give overconfident uncertainty prediction instead of converging to the mean. BNN-Homo (pSGLD) performs slightly better than the proposed cooperative learning strategy in fitting the mean under the data-scarce regime because it has the correct noise assumption. However, the proposed cooperative learning strategy has comparable performance even in the case where 5 points are used for training, which is encouraging.

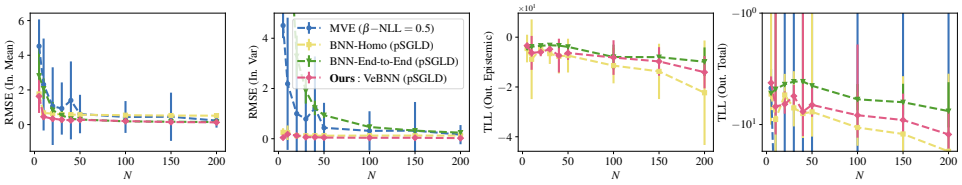


Figure 3.5: RMSE and TLL convergence curves under 5 different seeds for homoscedastic data generation.

3.4.2. REAL WORLD DATASETS WITH UNKNOWN ALEATORIC UNCERTAINTY

An important challenge in assessing the performance of methods that disentangle uncertainties is that existing datasets do not provide the ground-truth aleatoric uncertainty. In this section, we consider 16 different datasets that have been used in different papers on the topic, despite having unknown aleatoric uncertainty. Therefore, we caution that

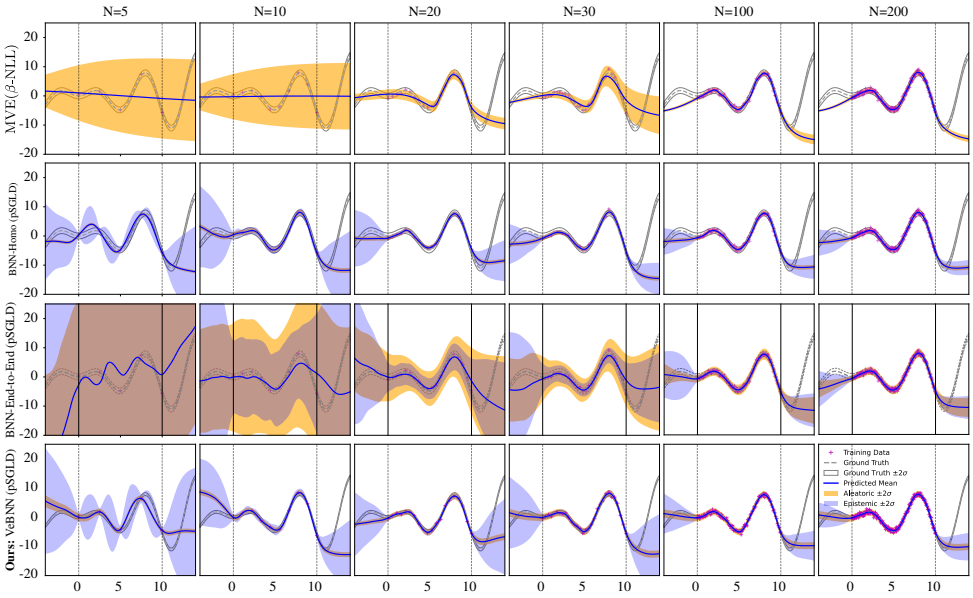


Figure 3.6: Predictions of MVE (β-NLL = 0.5), BNN-Homo (pSGLD) and **Ours:** BNN-VE(pSGLD) with different data points under homoscedastic data noise.

the datasets of this section can only be used to assess the quality of the mean and total uncertainty estimations.

UCI regression Datasets The results for the UCI regression datasets are summarized in Table 3.1 and Table 3.2 according to the RMSE and TLL, respectively. The mean estimates (RMSE) are broadly similar, as expected, although they improve with our proposed strategy for every inference method used when compared to training a single network. We also note that the mean estimates are better for VeBNN (pSGLD) than the deterministic mean network ME (MSE). The improvements in terms of TLL are more noteworthy, where the best performance is also for VeBNN (pSGLD).

Image regression datasets We considered 8 large-scale image regression datasets under real-world distribution shifts [79] that evaluate the mean and total uncertainty estimations. These problems were solved using a ResNet 34 network architecture [80]. As stated in [79], the purpose of this dataset is to evaluate the uncertainty quantification capabilities of deep learning models under distribution shift. Therefore, We first report results only for the problems involving distribution shift in Figure 3.7, while the complete results across all problems are summarized in Table 3.3. MVE (Ensembles) consistently serves as the strongest baseline across all problems, aligning with findings from [79]. As shown in Table 3.3, *Cells* and *ChairAngle* are the two baseline tasks without any distribution shift between training and test sets. In these cases, the test coverage reaches the target of 90%, as expected. However, our proposed VeBNN (pSGLD) achieves smaller MAE and narrower interval lengths on the validation set.

Table 3.1: RMSE (↓) results on UCI Regression Datasets. The best performance for each dataset is underlined and bold, and the second best is in bold.

Methods	Carbon (10721,7,1)	Concrete (1030,8,1)	Energy (768,8,2)	Boston (506,13,1)
ME (MSE)	0.0069 (0.0028)	**4.59 (0.86)**	**0.74 (0.07)**	**3.56 (0.81)**
MVE (β_{NLL}=1.0)	0.0070 (0.0029)	5.38 (0.84)	0.78 (0.08)	3.60 (0.91)
MVE (Ensemble)	**0.0066 (0.0029)**	5.11 (0.67)	0.79 (0.12)	4.79 (0.94)
Evidential	0.0068 (0.0029)	5.96 (0.61)	2.27 (0.46)	4.01 (1.02)
MVE (MC-Dropout)	0.0159 (0.0050)	5.43 (0.70)	1.66 (0.41)	3.85 (1.01)
Ours: VeBNN (MC-Dropout)	0.0105 (0.0022)	5.05 (0.81)	1.29 (0.12)	**3.08 (0.73)**
BNN-Homo (BBB)	0.0070 (0.0035)	5.69 (0.74)	1.35 (0.13)	3.81 (0.95)
BNN-End-to-End (BBB)	0.1307 (0.0384)	58.13 (47.72)	54.53 (40.59)	573.39 (238.16)
Ours: VeBNN (BBB)	0.0069 (0.0027)	5.43 (0.69)	1.20 (0.15)	**3.56 (0.83)**
BNN-Homo (pSGLD)	0.0068 (0.0028)	4.91 (0.68)	1.04 (0.10)	**3.56 (0.87)**
BNN-End-to-End (pSGLD)	**0.0066 (0.0029)**	5.58 (0.62)	2.08 (0.37)	3.81 (0.93)
Ours: VeBNN (pSGLD)	**0.0065 (0.0030)**	**4.65 (0.97)**	**0.70 (0.08)**	3.57 (0.95)
Methods	Power (9568,4,1)	Superconduct (21263,81,1)	Wine-Red (1599,11,1)	Yacht (308,6,1)
ME (MSE)	3.86 (0.15)	**11.70 (0.55)**	0.65 (0.05)	**0.67 (0.24)**
MVE (β_{NLL}=1.0)	3.90 (0.13)	12.52 (0.77)	**0.63 (0.05)**	0.83 (0.38)
MVE (Ensemble)	3.86 (0.14)	**11.82 (0.34)**	0.79 (0.09)	0.96 (0.35)
Evidential	3.92 (0.14)	13.93 (0.44)	0.64 (0.06)	3.26 (2.21)
MVE (MC-Dropout)	4.06 (0.12)	13.08 (0.92)	0.64 (0.06)	0.94 (0.31)
Ours: VeBNN (MC-Dropout)	4.00 (0.13)	12.28 (0.50)	**0.63 (0.05)**	0.78 (0.28)
BNN-Homo (BBB)	4.08 (0.13)	13.27 (0.39)	0.64 (0.05)	2.20 (0.53)
BNN-End-to-End (BBB)	8.82 (3.27)	404.30 (414.21)	2.11 (1.56)	261.08 (148.66)
Ours: VeBNN (BBB)	3.99 (0.13)	13.65 (0.46)	**0.63 (0.05)**	1.09 (0.26)
BNN-Homo (pSGLD)	**3.79 (0.14)**	**11.20 (0.39)**	**0.62 (0.05)**	1.54 (0.46)
BNN-End-to-End (pSGLD)	3.83 (0.15)	13.65 (0.31)	0.64 (0.06)	1.56 (0.76)
Ours: VeBNN (pSGLD)	**3.77 (0.14)**	12.04 (0.43)	**0.62 (0.05)**	**0.70 (0.29)**

For the shifted tasks *Cells-Gap* and *Cells-Tails*, VeBNN (pSGLD) attains test coverage closer to the 90% target compared to other methods, while also achieving lower validation MAE and shorter interval lengths. For the remaining tasks, VeBNN (pSGLD) performs comparably to MVE (Ensembles). Specifically, in *ChairAngle-Gap*, VeBNN (pSGLD) yields slightly higher test coverage but also slightly larger validation MAE and interval length. In *ChairAngle-Tail*, VeBNN (pSGLD) achieves significantly better test coverage with only a minor increase in validation MAE and interval length. For the last two problems, VeBNN (pSGLD) maintains comparable test coverage while producing tighter prediction intervals. It is also worth noting that MC-Dropout fails to maintain adequate coverage across these image regression datasets. In contrast, VeBNN (MC-Dropout) generally outperforms MVE (MC-Dropout), with the exception of the *SkinLesion* task on the validation set.

Overall, the conclusions are similar to those obtained from the UCI regression datasets. Our cooperative training strategy improves performance when compared to training a single network, whether considering an MVE network with MC-Dropout or a BNN

Table 3.2: TLL (↑) results on UCI Regression Datasets.

Methods	Carbon (10721,7,1)	Concrete (1030,8,1)	Energy (768,8,2)	Boston (506,13,1)
MVE (β_{NLL}=1.0)	3.79 (0.25)	-3.58 (1.74)	-1.14 (0.26)	-2.93 (0.66)
MVE (Ensemble)	-10.41 (49.67)	-3.45 (0.55)	**-0.93 (0.34)**	-4.39 (1.44)
Evidential	2.05 (0.31)	-3.60 (0.48)	-2.39 (0.43)	-3.57 (0.60)
MVE (MC-Dropout)	2.06 (0.12)	-2.99 (0.12)	-1.63 (1.63)	**-2.52 (0.22)**
Ours: VeBNN (MC-Dropout)	2.49 (0.03)	**-2.95 (0.15)**	-1.43 (0.06)	-2.63 (0.41)
BNN-Homo (BBB)	1.15 (0.03)	-3.17 (0.13)	-2.04 (0.15)	-2.72 (0.21)
BNN-End-to-End (BBB)	-1.99 (0.00)	-6.36 (0.35)	-6.03 (0.42)	-7.99 (0.28)
Ours: VeBNN (BBB)	**3.90 (0.33)**	-3.10 (0.25)	-1.06 (0.11)	-3.17 (1.01)
BNN-Homo (pSGLD)	1.46 (0.00)	-3.03 (0.09)	-2.15 (0.01)	-2.62 (0.18)
BNN-End-to-End (pSGLD)	0.20 (5.96)	-3.03 (0.35)	-1.16 (0.51)	**-2.57 (0.26)**
Ours: VeBNN (pSGLD)	**3.95 (0.42)**	**-2.91 (0.25)**	**-0.83 (0.08)**	-2.92 (0.59)
Methods	Power (9568,4,1)	Superconduct (21263,81,1)	Wine-Red (1599,11,1)	Yacht (308,6,1)
MVE (β_{NLL}=1.0)	-2.82 (0.12)	-3.81 (0.22)	-0.97 (0.12)	-1.98 (1.27)
MVE (Ensemble)	-2.77 (0.08)	-3.43 (0.04)	-1.72 (0.31)	-1.04 (1.01)
Evidential	-3.60 (0.65)	-4.02 (0.39)	-1.46 (0.54)	-2.93 (0.54)
MVE (MC-Dropout)	-2.82 (0.03)	-3.51 (0.44)	-1.00 (0.24)	**-0.61 (0.23)**
Ours: VeBNN (MC-Dropout)	-2.80 (0.03)	**-3.39 (0.12)**	-1.01 (0.12)	-1.24 (0.08)
BNN-Homo (BBB)	-2.89 (0.02)	-4.04 (0.04)	-0.98 (0.09)	-2.75 (0.05)
BNN-End-to-End (BBB)	-5.72 (1.00)	-7.48 (1.89)	-3.12 (0.49)	-7.57 (0.31)
Ours: VeBNN (BBB)	**-2.79 (0.04)**	-3.57 (0.10)	**-0.93 (0.08)**	-1.45 (0.18)
BNN-Homo (pSGLD)	-2.86 (0.02)	-3.83 (0.03)	-0.95 (0.13)	-2.64 (0.03)
BNN-End-to-End (pSGLD)	**-2.75 (0.06)**	-3.48 (0.54)	**-0.94 (0.13)**	**-0.64 (0.28)**
Ours: VeBNN (pSGLD)	**-2.74 (0.04)**	**-3.40 (0.09)**	**-0.93 (0.09)**	-1.29 (0.09)

trained end-to-end with pSGLD – see Table 3.3. As before, the best results are obtained from the proposed VeBNN (pSGLD) method. Compared with the strongest baseline, MVE (Ensembles), we observe similar performance in estimating the mean and total uncertainty when considering all metrics in Table 3.3. This is very encouraging because training is faster for VeBNN (pSGLD) when compared to MVE (Ensembles), as the former collects samples in the same training procedure, while ensembling requires collecting one sample per training of a single MVE (i.e., training restarts many times).

Although these datasets cannot be used to evaluate the quality of aleatoric uncertainty estimation, we noticed that the proposed VeBNN (pSGLD) method tends to have higher epistemic uncertainty in extrapolation regions (datasets ending with "-TAIL"), while the predicted aleatoric uncertainty remains stable even in these regions. This provides some indication that aleatoric uncertainty might be disentangled more effectively than with existing methods – see Figure 3.8.

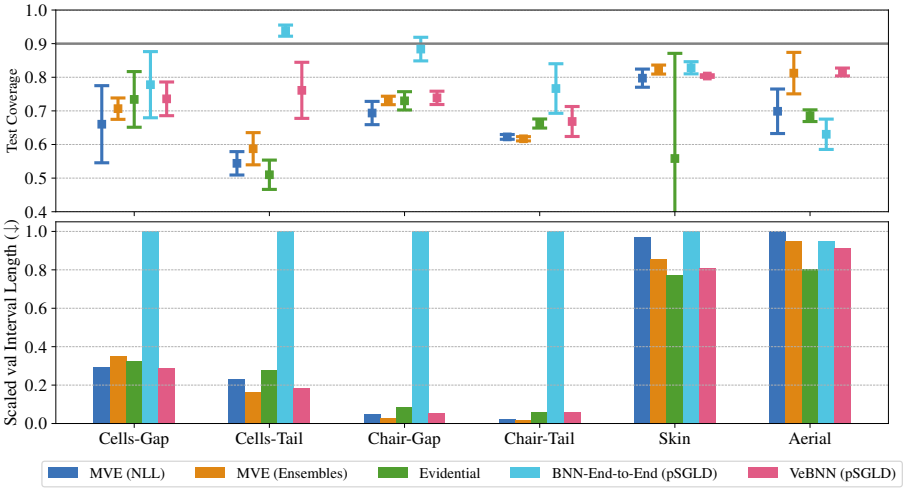


Figure 3.7: Accuracy metrics of select methods for six problems with distribution shift test dataset. The upper figure illustrates the test coverage, where values approaching 0.9 indicate better calibration. The lower figure presents the scaled interval length, normalized by the maximum value across all methods; smaller values signify tighter and more precise prediction intervals.

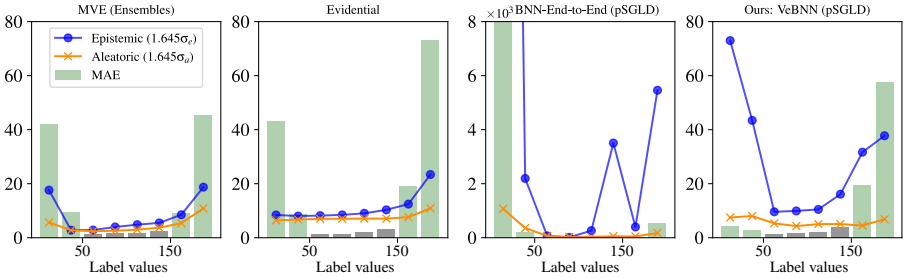


Figure 3.8: Uncertainty predictions versus Absolute Error for the *Cells-Tail* problem. The central four gray bars represent the Absolute Error of test points from the in-distribution data. The two bars on each side correspond to the Absolute Error of test points under distribution shift relative to the training dataset.

3.4.3. NEW DATASET WITH KNOWN ALEATORIC UNCERTAINTY: MATERIAL PLASTICITY LAW DISCOVERY

This section considers a dataset created by us that was generated by computer simulations of materials under mechanical deformation. For the purposes of this chapter, the underlying Physics solver and physical meaning of the inputs and outputs is not relevant. Instead, we simply highlight that because the data is generated from computer simulations of materials with stochastic microstructure variations, we can generate as many observations for the same material as we want – effectively allowing us to probe the ground-truth aleatoric uncertainty for any input point (or sequence). In addition, the outputs are history-dependent, i.e., this is a sequence-to-sequence regression problem

Table 3.3: Complete results for the image regression datasets. The first two columns are the MAE and interval length for the validation dataset, and the third and fourth columns are the MAE and test coverage results of the test dataset. For each problem, we highlight the best method in **green** but we also highlight in **yellow** methods with similar performance. Note that all metrics must be considered in the method evaluation: the objective is to have test coverage as close to 0.90 as possible (for a low validation interval length), and to have low MAE (for validation and testing sets). Test coverage is the most important metric, but only when MAE is low.

PROBLEM	METHOD	VAL MAE (↓)	VAL INTERVAL LENGTH (↓)	TEST MAE (↓)	TEST COVERAGE (0.90)
CELLS	MVE (NLL)	3.6170± 1.1362	14.5492 ± 4.4493	3.3858 ± 1.1927	0.9028 ± 0.0059
	MVE (ENSEMBLES)	2.7876 ± 1.1695	17.1285 ± 4.3337	**2.8788 ± 0.4288**	**0.9006 ± 0.0027**
	EVIDENTIAL	**2.1816 ± 0.5478**	**12.6658 ± 2.1219**	12.6657 ± 0.5186	0.8865 ± 0.0026
	MVE (MC-DROPOUT)	15.8976 ± 1.0837	93.5508 ± 4.2047	50.6258 ± 0.2451	0.8979 ± 0.0165
	OURS: VEBNN (MC-DROPOUT)	10.5583 ± 1.2076	47.8594 ± 5.1651	10.8850 ± 1.5156	0.8948 ± 0.0148
	BNN-END-TO-END (PSGLD)	9.4833 ± 3.4854	69.2175 ± 34.6855	10.8901 ± 4.2301	**0.9019 ± 0.0035**
	OURS: VEBNN (PSGLD)	**2.1714 ± 0.1681**	**11.4478 ± 2.9070**	**2.2307 ± 0.1593**	0.8959 ± 0.0056
CELLS-GAP	MVE (NLL)	3.5309 ± 1.0619	**15.6396 ± 6.2345**	6.5164 ± 1.2826	0.6603 ± 0.1147
	MVE (ENSEMBLES)	3.4612 ± 0.9543	18.7070± 4.1329	**5.3576 ± 0.2753**	0.7066 ± 0.0318
	EVIDENTIAL	**3.2381 ± 1.7424**	17.2311 ± 1.8113	6.2183± 1.4849	0.7341 ± 0.0828
	MVE (MC-DROPOUT)	15.5046 ± 2.3557	99.6555 ± 14.3327	52.3158 ± 0.3042	**0.7372 ± 0.0764**
	OURS: VEBNN (MC-DROPOUT)	11.3806 ± 4.5054	56.0703 ± 16.8834	17.3838 ± 1.0463	0.7181 ± 0.1721
	BNN-END-TO-END (PSGLD)	5.9391 ± 1.8787	53.0860 ± 34.1252	8.0690 ± 3.6231	**0.7779 ± 0.0984**
	OURS: VEBNN (PSGLD)	**2.4714 ± 0.5216**	**15.2334 ± 5.8290**	**5.5104 ± 1.9681**	0.7358 ± 0.0501
CELLS-TAIL	MVE (NLL)	4.0545 ±1.3315	15.4321 ± 4.9880	**14.6865 ± 2.2122**	0.5440 ± 0.0348
	MVE(ENSEMBLES)	**2.4069 ± 0.5805**	**10.9696± 1.7005**	**14.1253 ± 0.5800**	0.5874 ± 0.0479
	EVIDENTIAL	**2.0857 ± 0.1780**	18.4345 ± 0.8247	18.4345 ± 3.4760	0.5100 ± 0.0436
	MVE (MC-DROPOUT)	14.5111 ± 1.9255	81.3095 ± 4.3530	50.5127 ± 0.0804	0.6942 ± 0.0217
	OURS: VEBNN (MC-DROPOUT)	8.3800 ± 1.1060	37.4281 ± 3.6516	20.9490 ± 1.9950	0.5974 ± 0.0380
	BNN-END-TO-END (PSGLD)	8.7096 ± 4.2490	66.0935 ± 35.1367	407.6956 ± 601.3301	**0.9387± 0.0166**
	OURS: VEBNN (PSGLD)	2.4510 ± 0.7056	**12.2763 ± 3.5848**	41.4620 ± 37.2828	**0.7611 ± 0.0834**
CHAIRANGLE	MVE (NLL)	0.3767 ± 0.1719	1.3776 ± 0.38225	0.4805 ± 0.0274	0.9016 ± 0.0031
	MVE (ENSEMBLES)	0.3618 ± 0.1653	**1.2044 ± 0.4424**	0.4051 ± 0.0799	0.9104 ± 0.0032
	EVIDENTIAL	**0.3413 ± 0.1762**	2.6932 ± 0.3732	**0.3350 ± 0.1756**	**0.9066 ± 0.0032**
	MVE (MC-DROPOUT)	9.7782 ± 0.6145	54.9429 ±4.3036	12.5577 ± 0.2768	0.6379 ± 0.0386
	OURS: VEBNN (MC-DROPOUT)	10.1604 ± 1.1318	44.9722 ± 4.0769	21.5712 ± 0.7730	0.5752 ± 4.5598
	BNN-END-TO-END (PSGLD)	11.0020 ± 4.2520	84.5294 ± 29.1052	12.0116 ± 4.9162	0.9072 ± 0.0052
	OURS: VEBNN (PSGLD)	**0.2918 ± 0.0823**	**1.1524 ± 0.2098**	**0.2895 ± 0.0838**	**0.9060 ± 0.0046**
CHAIRANGLE-GAP	MVE (NLL)	**0.4545 ± 0.2802**	**2.0921 ± 0.9338**	**1.4182 ± 0.2737**	0.6936± 0.0346
	MVE(ENSEMBLES)	**0.2264 ± 0.0677**	**1.3059 ± 0.1362**	**1.1909 ± 0.0607**	0.7310 ± 0.0126
	EVIDENTIAL	0.4683 ± 0.1803	3.8559 ± 0.5422	1.7800 ± 0.2434	0.7299 ± 0.0272
	MVE (MC-DROPOUT)	15.5046 ± 2.3557	99.6555 ± 14.3327	15.4490 ± 1.7975	**0.7372 ± 0.0764**
	OURS: VEBNN (MC-DROPOUT)	14.2239 ± 1.8959	63.0351 ± 6.7923	22.1338 ± 2.2755	0.7331 ± 0.0463
	BNN-END-TO-END (PSGLD)	7.0322 ± 0.9985	44.2601 ± 1.2242	7.4614 ± 1.5881	0.8838 ± 0.0351
	OURS: VEBNN (PSGLD)	0.5123 ± 0.1273	2.3708 ± 0.7181	1.6514 ± 0.2616	**0.7387 ± 0.0198**
CHAIRANGLE-TAIL	MVE (NLL)	**0.2412 ± 0.0917**	**1.0941 ± 0.3829**	3.1580 ± 0.2162	0.6226 ± 0.0071
	MVE(ENSEMBLES)	**0.1337 ± 0.0190**	**0.7691 ±0.0814**	**1.4892 ± 0.0454**	0.6170 ± 0.0063
	EVIDENTIAL	0.3828 ± 0.2221	2.5303 ± 1.0144	**2.8544 ± 0.2497**	0.6624 ± 0.0135
	MVE (MC-DROPOUT)	14.5111 ± 1.9255	81.3095 ± 4.3530	12.6523 ± 0.3463	**0.6942 ± 0.0217**
	OURS: VEBNN (MC-DROPOUT)	8.3647 ± 1.3729	37.2592 ± 5.1038	20.9121 ± 1.2496	0.5090 ± 0.0686
	BNN-END-TO-END (PSGLD)	6.7768 ± 3.0065	43.6364 ± 1.5572	10.0926 ± 2.3311	**0.7665 ± 0.0737**
	OURS: VEBNN (PSGLD)	0.4755 ± 0.2692	2.5667 ± 1.5456	3.7509 ± 0.1950	0.6684 ± 0.0446
SKINLESION	MVE (NLL)	105.4170 ± 1.1518	535.1390 ± 215.4460	262.9090 ± 12.6078	0.7973 ± 0.0270
	MVE (ENSEMBLES)	**100.6390 ± 0.4642**	472.1400 ± 85.2654	**241.3947 ± 2.4395**	**0.8229 ± 0.0134**
	EVIDENTIAL	**99.6063 ± 6.9670**	**425.1200 ± 45.0808**	260.6220 ± 11.7347	0.5583 ± 0.3128
	MVE (MC-DROPOUT)	258.0571 ± 17.4759	1356.7668 ± 118.0702	645.0634 ± 5.2036	0.7328 ± 0.0290
	OURS: VEBNN (MC-DROPOUT)	305.9113 ± 35.1769	1350.9520 ± 188.2830	464.4328 ± 25.0852	0.7841 ±0.0399
	BNN-END-TO-END (PSGLD)	127.9469 ± 5.2489	550.7750 ± 25.6849	267.6061 ± 4.5646	**0.8282 ± 0.0181**
	OURS: VEBNN (PSGLD)	105.5894 ± 4.1459	**446.7374 ± 9.2086**	**260.3773 ± 9.0958**	0.8036 ± 0.0038
AERIALBUILDING	MVE (NLL)	217.8770 ± 1.7249	929.5620 ± 47.6606	330.8905 ± 32.7377	0.6988 ± 0.0662
	MVE (ENSEMBLES)	208.4870 ± 1.0358	885.3490± 27.8584	**295.1917 ± 9.6539**	**0.8123 ± 0.0617**
	EVIDENTIAL	**180.5486 ± 2.4021**	**747.9998 ± 6.5119**	354.8147 ± 12.1219	0.6857 ± 0.0174
	MVE (MC-DROPOUT)	302.0488 ± 6.7523	1455.8063 ± 51.6707	497.3040 ± 8.2818	0.7156 ± 0.0346
	OURS: VEBNN (MC-DROPOUT)	296.4448 ± 9.5291	1295.8913 ± 57.9997	537.5555 ± 38.6030	0.6815 ± 0.0617
	BNN-END-TO-END (PSGLD)	228.8089 ± 3.4239	885.3624 ± 17.3684	**228.8089 ± 109.3649**	0.6304 ± 0.0451
	OURS: VEBNN (PSGLD)	**205.4323 ± 2.6103**	**847.0608 ± 23.6498**	347.9002 ± 25.0661	**0.8157 ± 0.0117**

that should be learned considering architectures with recurrent units, such as a Gated Recurrent Unit (GRU) architecture [81, 82]. The dataset is made available as open-source, in the hope of facilitating future comparisons with new methods.

Compare to baselines the results for this dataset are shown in Figure 3.9. Note that results using BBB inference are not available due to a lack of convergence. The reader is also referred to Figure B.5 in appendix B.5 that shows a typical ground truth response with one input sequence (material deformation path) and corresponding output sequence (stochastic material response along that path) where each path results from 100 points obtained from a Physics simulator.

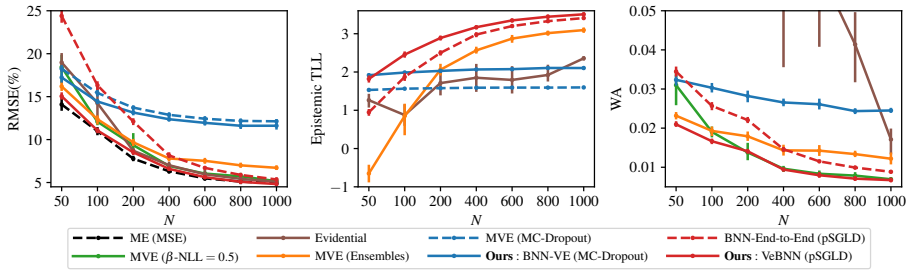


Figure 3.9: Accuracy metrics (RMSE ↓, Epistemic TLL ↑, and WA ↓) obtained for the plasticity law discovery dataset considering a training set with different number N of training sequences (history-dependent paths), where each training sequence has 100 points – an example of a typical input and heteroscedastic output path is shown in Figure B.5 of Appendix B.5. The Wasserstein distance (WA) represents the closeness of the estimated aleatoric uncertainty distribution to the ground truth distribution. Note that the MVE (MC-Dropout) and Evidential methods have large values of WA, and part of their curves is cut off. All metrics result from repeating the training of each method 5 times by resampling points randomly from the training datasets.

Figure 3.9 shows the accuracy metrics obtained with different training sequences taken from the training dataset where N increases from 50 to 1000. As expected, more training data improves accuracy for all methods, but it is clear that the proposed VeBNN (pSGLD) method achieves better predictions for all metrics. Figure 3.9 also demonstrates the performance improvements obtained from the proposed cooperative training strategy (solid lines) compared to others (dashed lines) for both inference types: MC-Dropout and pSGLD. Note that the proposed VeBNN (pSGLD) has comparable performance with the ME (MSE) when estimating the mean (RMSE metric), and similar Wasserstein distance to the MVE (β-NLL = 0.5) network, the best method among MVEs, when estimating the aleatoric uncertainty.

Figure 3.10 shows the predictions of the proposed VeBNN (pSGLD) and its correct identification of the disentangled data uncertainties, which improves as the training data increases (upper column to bottom column). As expected, the proposed VeBNN (pSGLD) outperforms BNN-End-to-End (pSGLD) for the same number of training sequences, and the estimated epistemic uncertainty decreases with increasing training data. The MVE (Ensembles) performs well when considering the larger training dataset (800 training sequences), but its prediction is significantly worse than VeBNN (pSGLD) for 50 training sequences (this is also seen in Figure 3.9). On the contrary, Evidential has significant difficulties with this problem. Predictions for other methods are given in Figure B.2, in which the proposed cooperative training strategy also considerably improves predictions when using MC-Dropout as inference.

We also observe in Figure 3.9 that the aleatoric uncertainty estimated by the VeBNN (pSGLD) converges for $N > 400$, since WA remains stable. Furthermore, the VeBNN also

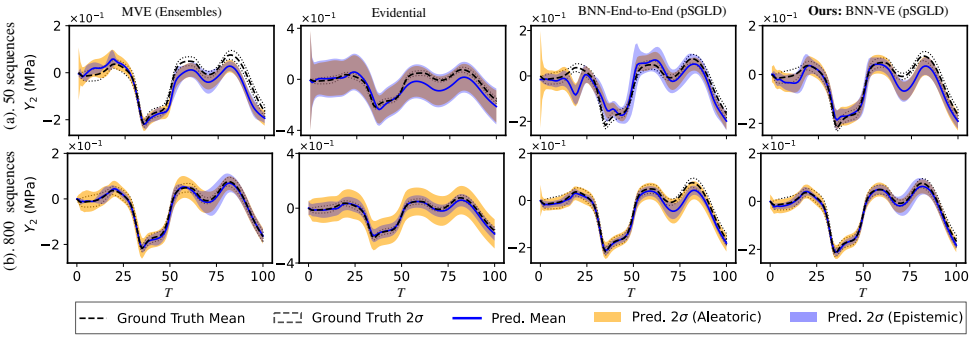


Figure 3.10: Predictions of different methods with estimation both aleatoric and epistemic uncertainties on plasticity law discovery dataset. We randomly pick one test point from the dataset and show the entire third component $\mathbf{y}_2$ of 100-time steps under the 50 and 800 training sequences respectively.

correctly disentangles the uncertainties, as shown in Figure 3.10. Unfortunately, we have not found other datasets reporting the ground-truth aleatoric uncertainty like the dataset we created herein. Nevertheless, we find these results very encouraging, and we hope our dataset will motivate future developments in the field.

Ablation study for Gamma Loss and number of iterations K An ablation study is conducted for this dataset (plasticity law discovery) because the ground-truth aleatoric uncertainty is known. The results are presented in Figure 3.11. The proposed training strategy significantly improves upon the state-of-the-art even when training the networks only once (i.e., $K = 1$). Interestingly, we also noticed that training converged for every dataset with only one additional iteration ($K = 2$). To further explore this, we show the trajectory of all essential components of VeBNN with respect to the iteration K from 1 to 5 for 800 training sequences in Figure 3.12. These results confirm that K is merely the iteration count until convergence rather than a tunable hyperparameter.

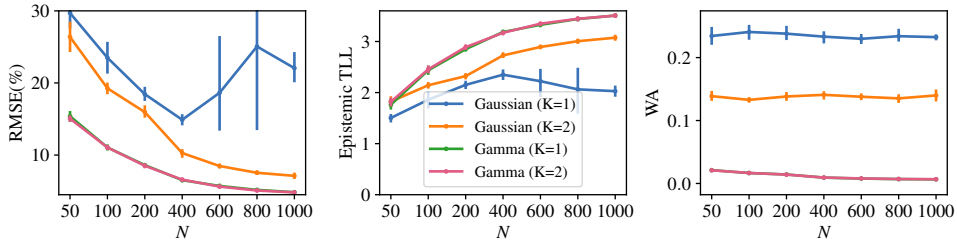


Figure 3.11: Ablation study for the Gamma loss and iteration parameter K using VeBNN (pSGLD). Curves labeled *Gamma* and *Gaussian* correspond to Step 2 training with the proposed Gamma loss (Equation (3.6)) and the original Gaussian NLL loss (Equation (3.2)), respectively, while fixing $\mu(\mathbf{x};\boldsymbol{\theta})$ from Step 1

Finding $\sigma_a^2(\mathbf{x};\boldsymbol{\phi})$ with a Gaussian NLL loss in Step 2 leads to worst performance due to the high-order term in the gradient calculation (Equation (3.3)) that can lead to $\sigma_a^2(\mathbf{x}) \to \infty$ and a degenerate local optimum for which the gradient tends to zero. In contrast, the proposed Gamma NLL loss addresses this and yields a stable optimization process.

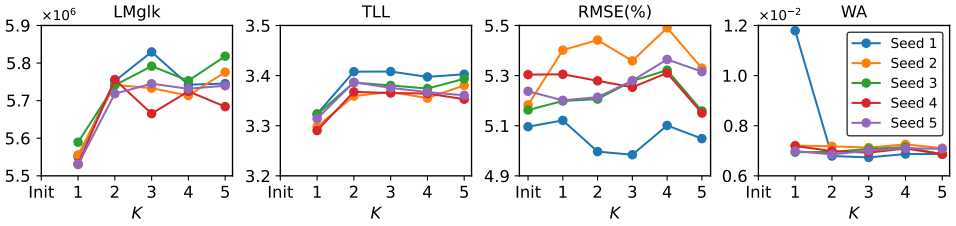


Figure 3.12: Trajectory of all essential components with respect to the iteration K for 800 training sequences in the plasticity law discovery problem. In the figures, "Init" represents the initialization of training the mean network only as described in Section 3.3.2, and different curves are realizations under different seeds to restart the procedure, as well as using new samples of training sequences in the dataset.

Ablation study for the regularization of the Gaussian likelihood of Step 3 Figure 3.13 compares VeBNN (pSGLD) with BNNs trained under different likelihood regularization schemes. While β-NLL achieves good mean and aleatoric performance (Figure 3.9), it fails to disentangle uncertainties in a Bayesian setting (Figure 3.13) and is sensitive to the choice of β. This is consistent with prior work on posterior tempering [69], which shows that likelihood scaling does not necessarily improve inference. In contrast, MSE and Homo-NLL yield poor uncertainty estimates, as they rely on unrealistic unit-variance or constant-variance assumptions in heteroscedastic regression. Finally, VeBNN (pSGLD) performs Bayesian inference using the variance learned in Step 2, achieving better mean and epistemic uncertainty—even under the original NLL—due to the correct identification of aleatoric uncertainty.

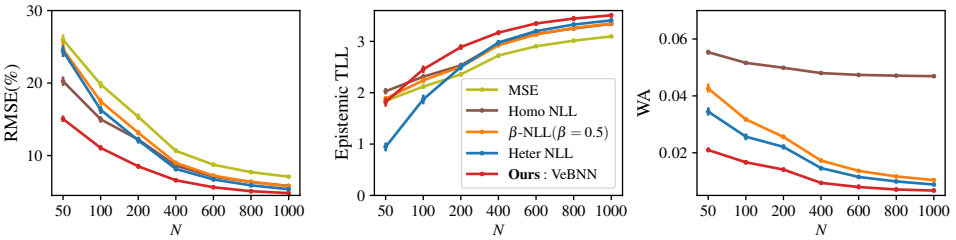


Figure 3.13: Comparison with BNN end-to-end (pSGLD) training under different likelihood regularization schemes: MSE denotes unit variance; Homo (NLL) assumes a constant variance determined by grid search; β-NLL uses the best β found via grid search; Heter-NLL corresponds to the original likelihood.

3.5. DISCUSSION

Size of variance estimation network The variance estimation network used for estimating aleatoric uncertainty is trained separately from the BNN. However, we believe that this is an advantage, as the architecture required to learn aleatoric uncertainty separately is simpler than when training for everything at once. We considered 8 different variance network configurations and observed robust estimations of aleatoric uncertainty, as shown in Figure 3.14.

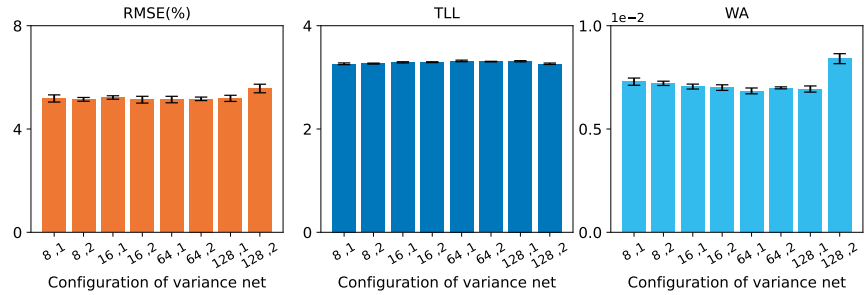


Figure 3.14: Barplot of all performance metrics with different variance network configurations under 800 training points for plasticity law discovery problem.

Computational cost As shown in Experiments, VeBNN converges quickly: the mean network is trained once, while the variance network and Bayesian inference are performed twice. The mean network quickly finds a posterior mode, facilitating MCMC sampling without a long burn-in phase. For the best inference method (pSGLD), the number of training epochs is comparable to ME and MVE training. In contrast, considering homoscedastic noise and treating it as a hyperparameter typically requires many full training runs and performs worse. Compared to Deep Ensembles, which yield only one posterior sample per run, our method is more efficient. End-to-End training has similar efficiency, but our method consistently achieves better accuracy. Since deterministic ME and VE training is inexpensive relative to BNN training, the additional cost of our strategy is negligible and benefits from the warm start of the mean.

3.5.1. LIMITATIONS

Although the proposed method facilitates Bayesian inference for BNNs, training BNNs remains more challenging than training deterministic networks. When the posterior is multimodal, non-smooth, or only partially known, inference of epistemic uncertainty under a fixed Gaussian prior–likelihood pair can become unreliable. Recent advances in Imprecise Probabilistic Machine Learning (IPML), such as Credal Bayesian Deep Learning (CBDL) [74, 83] provide broader epistemic coverage but at significantly higher computational cost. Our framework may serve as a lightweight foundation that can be extended with IPML techniques in future work. Importantly, our results show that Bayesian inference is easier when aleatoric uncertainty is determined separately, as proposed herein.

3.6. CONCLUSION

We propose a novel Bayesian heteroscedastic regression strategy based on cooperative training of deterministic and Bayesian neural networks that disentangles epistemic and aleatoric uncertainties. The proposed method is efficient, robust, and straightforward to implement because it is simpler to train each network in isolation, while ensuring complementarity in their training. We believe the method is scalable and applicable to real-world problems involving active learning and Bayesian optimization considering data-scarce and data-rich scenarios, unlike Gaussian process regression.

4

PRACTICAL MULTI-FIDELITY MACHINE LEARNING: FUSION OF DETERMINISTIC AND BAYESIAN MODELS

Multi-fidelity machine learning methods address the accuracy-efficiency trade-off by integrating scarce, resource-intensive high-fidelity data with abundant but less accurate low-fidelity data. In this chapter, a practical multi-fidelity strategy for problems spanning low- and high-dimensional domains is proposed, integrating a non-probabilistic regression model for the low-fidelity with a Bayesian model for the high-fidelity. The models are trained in a staggered scheme, where the low-fidelity model is transfer-learned to the high-fidelity data and a Bayesian model is trained to learn the residual between the data and the transfer-learned model. This three-model strategy – deterministic low-fidelity, transfer-learning, and Bayesian residual – leads to a prediction that includes uncertainty quantification for noisy and noiseless multi-fidelity data. The strategy is general and unifies the topic, highlighting the expressivity trade-off between the transfer-learning and Bayesian models (a complex transfer-learning model leads to a simpler Bayesian model, and vice versa).

This chapter is based on the manuscript: **Yi, J.**, Cheng, J., & Bessa, M. A. (2024). *Practical Multi-Fidelity Machine Learning: Fusion of Deterministic and Bayesian Models. arXiv preprint* arXiv:2407.15110. Under review
https://arxiv.org/abs/2407.15110

4.1. INTRODUCTION

Expressive deterministic (non-probabilistic) machine learning methods, such as Deep Neural networks (DNNs), can approximate any nonlinear continuous function, providing enough data [13]. However, they can be prone to over-fitting, especially in the presence of noisy data [14], and are not able to predict uncertainty. Bayesian machine learning (BML) methods do not have these limitations [84], although their training requires additional computational effort. In this article, we present a practical Multi-fidelity (MF) strategy that combines non-probabilistic Low-fidelity (LF) surrogates and Bayesian residual surrogates. We demonstrate the simplicity of the proposed MF strategy and consider two common regression scenarios encountered in engineering practice: (1) low-dimensional problems amenable to models with few hyperparameters and easy training; and (2) high-dimensional problems that require more expressive models and involve additional training effort.

MF data arises naturally in most engineering applications. High-fidelity (HF) data is often associated with costly and time-consuming experiments, or with accurate yet computationally expensive simulations. In contrast, LF data is usually obtained by efficient strategies (experimental [85], analytical [86], or computational [87, 88]) at the expense of accuracy. Therefore, LF data is typically acquired faster, often by several orders of magnitude. Using both datasets can be advantageous to train better models as long as the loss in fidelity is compensated by the presence of more LF data.

The challenges associated with developing MF regression models arise from the need to define one regression model (a surrogate) per fidelity, and in the subsequent definition of the interaction between these models [28, 89]. Without loss of generality, we focus on MF regression models with only two levels of fidelity. We start by introducing MF regression in a general form and by reviewing common choices for surrogate models. We continue by reviewing common MF strategies employing the same type of surrogate model across fidelity levels. Finally, we propose two MF strategies using non-probabilistic LF regression models together with Bayesian residual models.

Summary of contributions. We introduce a practical framework for MF regression and demonstrate its versatility. We hypothesize that most MF regression cases in practice can be covered by considering simple LR transfer-learning models and with one of the following two options for LF and HF models: 1) using KRR and GPR, leading to the KRR-LR-GPR model; or 2) using DNN and BNN, leading to the DNN-LR-BNN model. The KRR-LR-GPR model has few parameters and hyperparameters, facilitating training but limiting scalability (limits on data dimension and size, as well as training and inference time). The DNN-LR-BNN model is more flexible and more challenging to train but is applicable to low- and high-dimensional data with few scalability limits. The simpler model (KRR-LR-GPR) should be used when possible. In limited cases exhibiting complex correlations between LF and HF data, both transfer-learning and residual learning can be achieved simultaneously by a BNN – denoted as DNN-BNN models [90]. However, DNN-BNN models are more difficult to train and are expected to have narrower applicability.

4.2. METHODOLOGY AND RELATED WORK

For simplicity of notation, we start by considering datasets with d-dimensional inputs $\mathbf{x}$ but one-dimensional output y. Then, the LF dataset contains $\mathbf{x}_n^l$ input and y_n^l output points, where $n = 1,...,N^l$ are the N^l points in this dataset (the superscript l refers to LF). The HF dataset is equivalently defined by $\mathbf{x}_n^h$ and y_n^h, where $n = 1,...,N^h$ are the N^h HF points in this dataset. We propose to define an MF regression model $f^h(\mathbf{x})$ as a combination of three models: (1) the LF model $f^l(\mathbf{x})$ that is trained on LF data $\{\mathbf{x}^l, y^l\}$; (2) a transfer-learning model $g(\mathbf{x})$ that transforms the LF model to the HF data; and (3) a residual model $r(\mathbf{x})$ (if necessary) that captures the difference between transfer-learned LF model and the HF data. Therefore, a general description can be formulated as:

$$f^h(\mathbf{x}) = g\left(f^l(\mathbf{x}), \mathbf{x}\right) + r(\mathbf{x}) \tag{4.1}$$

In the data-scarce literature [25, 26], MF models use GPRs for both $f^l(\mathbf{x})$ and $r(\mathbf{x})$ and implicitly assume a linear transfer-learning model $g(\mathbf{x}) := g(f^l(\mathbf{x})) = f^l(\mathbf{x})\rho$, such that:

$$f^h(\mathbf{x}) = f^l(\mathbf{x})\rho + r(\mathbf{x}) \tag{4.2}$$

where ρ is a single hyperparameter that transforms the LF model $f^l(\mathbf{x})$ to the HF responses.

Other investigations use more expressive transfer-learning models without defining a different model for the residual [90, 91]:

$$f^h(\mathbf{x}) = g\left(f^l(\mathbf{x}), \mathbf{x}\right) \tag{4.3}$$

Here, we argue in favor of generalizing these modeling assumptions by establishing a simple transfer-learning model $g(\mathbf{x})$ that is trained on HF data, choosing an appropriate deterministic LF model $f^l(\mathbf{x})$ and a Bayesian HF model for the residual $r(\mathbf{x})$.

4.2.1. RELATED WORK FOR DATA-SCARCE SCENARIOS

In the data-scarce regime, GPR or Kriging [44] stand out due to their elegant derivation and exact integration for Gaussian likelihoods and priors –see Section 2.3.2 for a short introduction. Early in their development, GPR was extended to handle MF problems according to Equation (4.2), in a method originally called Co-Kriging [27, 92]. This method is based on expanding the covariance function of GPRs in the form of

$$\mathbf{C} = \begin{bmatrix} \mathbf{K}(\mathbf{X}^h, \mathbf{X}^h) & \mathbf{K}(\mathbf{X}^h, \mathbf{X}^l) \\ \mathbf{K}(\mathbf{X}^l, \mathbf{X}^h) & \mathbf{K}(\mathbf{X}^l, \mathbf{X}^l) \end{bmatrix} \tag{4.4}$$

Different variants of this formulation can be found, including multi-task GPR [26, 93] that consider more than one output, or latent mapping GPR for cases with categorical variables [94, 95]. Unfortunately, Co-Kriging has a complexity of order $\mathcal{O}\left((N^l + N^h)^3\right)$ due to the inversion of the covariance matrix $\mathbf{C}$ (see Equation (2.36)), imposing practical limits on the dimensionality and number of training data points that can be considered [25, 89, 96]. In part this issue is mitigated in Hierarchical Kriging [97] and its variants [93, 98, 99] by defining a diagonal sub-matrix of $\mathbf{C}$. Recursive Co-Kriging [100] employs a

fast hyperparameter optimization strategy and cross-validation to identify the parameter ρ, leading to comparable performance to Co-Kriging in some cases. In addition, Scale Kriging [101, 102] adopts a simplification by considering a fixed parameter with $\rho = 1$ in Equation (4.2) for the transfer-learning model. However, in practice, striking a balance between training accurate MF-GPR models and improving their complexity such that they are trained on large datasets remains a significant challenge [103, 104]. Furthermore, the literature is scarce on their application to noisy MF data [105].

In addition, data-scarce MF problems can be addressed by other methods. For example, MF-PCE [105] has been applied to noisy HF data and noiseless LF data. This method also relies on Equation (4.2), but suffers from similar scalability issues as MF-GPR. In contrast, deterministic MF models have shown better scalability but do not predict uncertainty. Examples include MF linear regression [106], MF-RBF regression [107], and MF support vector regression [108]. Inspired by the strengths of deterministic and Bayesian methods, this article explores the combination of deterministic and Bayesian models in MF regression.

4.2.2. RELATED WORK FOR DATA-RICH SCENARIOS

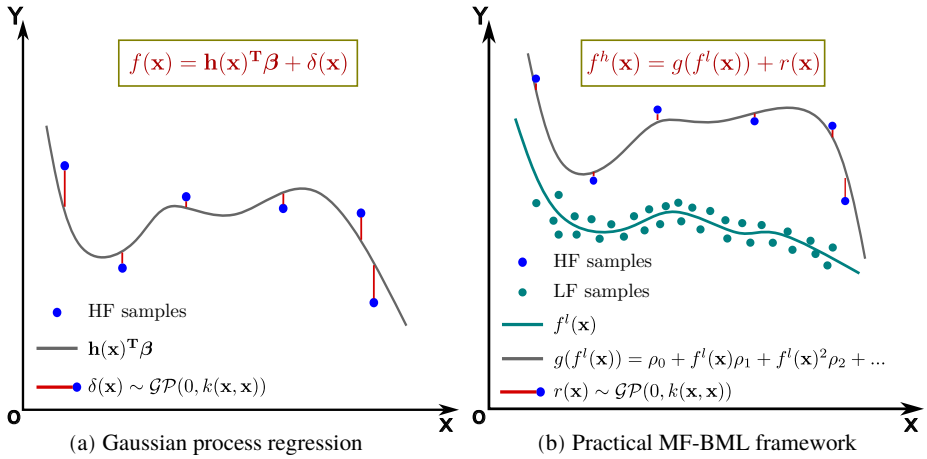
Machine learning models trained on data-rich MF datasets can be seen in different contexts, including design [109, 110], optimization [111], and uncertainty quantification [112]. When solving regression problems with deterministic models such as DNNs, predicting the full posterior is avoided by finding a point estimate. For example, the MF-DNN approach proposed by Aydin et al. [29] passes data sequentially from the LF to the HF into the same DNN and includes an error metric for fidelity switching. Other strategies involve transfer-learning [113, 114] by pre-training the weights of a DNN on the LF dataset and then finalizing training on the HF dataset. Other relevant examples include an MF Graph Neural Network developed by Black et al. [115], and an MF-DNN architecture introduced by Motamed et al. [112] that establishes the correlation between the HF and LF models via another neural network. Some strategies concatenate LF- and HF-DNN models [91, 116, 117], as well as Recurrent Neural Networks [118], and Convolutional Neural Networks [119].

Although deterministic MF-DNNs are common in the literature, they are ineffective in predicting uncertainty and are prone to over-fitting [90, 116, 117]. BNNs [21] address these limitations but introduce other issues. In particular, the integration of the posterior distribution and the posterior predictive distribution (PPD) is intractable analytically, so Bayesian inference is performed by Markov Chain Monte Carlo [21, 50], Variational Inference [72], Monte Carlo Dropout [23], among others [22]. Unfortunately, Bayesian inference is computationally expensive and makes it challenging to adopt these models for large or high-dimensional datasets. This is visible in the promising work of Meng et al. [90], who extended a deterministic MF-DNN architecture [116] by replacing the HF-DNN with a BNN, concatenating it with a LF-DNN, and using Hamiltonian Monte Carlo to perform inference [49, 120]. Yet, the method is only demonstrated on small datasets. Similar limitations are reported using Bayesian models for both LF and HF data, as in the work of Baptiste et al. [121] where GPR is used for LF and a BNN for HF. The same authors note that using BNNs for both LF and HF datasets is possible, but the inference time becomes worse than combining GPR with BNN.

The literature spans different architectural choices involving deterministic or Bayesian models. Most strategies rely on the linear model shown in Equation (4.2), and apply methods to data-scarce scenarios without considering noisy data, especially at the LF level. To our knowledge, addressing the trade-off between inference time and accuracy remains a knowledge gap in the literature.

4.3. PRACTICAL MULTI-FIDELITY BAYESIAN MACHINE LEARNING

In practice, creating MF models involves so many choices [25, 26] that is difficult to perform principled model selection and hyperparameter optimization, especially for large datasets. We propose a simple strategy summarized in Equation (4.1) to develop practical MF models based on two ideas, as shown in Figure 4.1. First, consider a deterministic LF model $f^l(\mathbf{x})$ together with a probabilistic model of the residual $r(\mathbf{x})$. Second, consider a transfer-learning model $g(\mathbf{x})$ as a simple linear regression model whose features become the outputs of the LF model, i.e. $g\left(f^l(\mathbf{x})\right)$. As we demonstrate in the remainder of the paper, the proposed strategy leverages the advantages of deterministic and probabilistic models while keeping the transfer-learning model simple.



(a) Gaussian process regression (b) Practical MF-BML framework

Figure 4.1: Schematic (a) shows a linear regression model with basis function $\mathbf{h}(\mathbf{x}) = [1, \mathbf{x}, \mathbf{x}^2, \ldots]^T$ and coefficients $\boldsymbol{\beta}$ that is augmented by the residual $\delta(\mathbf{x})$ modeled by zero mean GPR. Schematic (b) shows the proposed MF-BML strategy where the transfer-learning model $g\left(f^l(\mathbf{x})\right)$ is a linear regression model whose features are the outputs of the LF surrogate model $f^l(\mathbf{x})$ obtained by training on LF data. The linear transfer-learning function that acts on $f^l(\mathbf{x})$ adjusts the LF model to the HF data by determining the coefficients $\boldsymbol{\rho}$, and facilitates the determination of the residual $r(\mathbf{x})$ by a Bayesian model with a simple prior.

Writing the transfer-learning model $g(\mathbf{x})$ in Equation (4.2) as a linear model (linear in the weights) and considering its features as the LF model leads to:

$$
\begin{aligned}
f^h(\mathbf{x}) &= g\left(f^l(\mathbf{x})\right) + r(\mathbf{x}) \\
&= \boldsymbol{m}(\mathbf{x})^T \boldsymbol{\rho} + r(\mathbf{x}) \\
&= \rho_0 + f^l(\mathbf{x})\rho_1 + f^l(\mathbf{x})^2\rho_2 + \ldots + f^l(\mathbf{x})^{M-1}\rho_{M-1} + r(\mathbf{x})
\end{aligned} \tag{4.5}
$$

where $\mathbf{m}(\mathbf{x}) = [1, f^l(\mathbf{x}), f^l(\mathbf{x})^2, \ldots, f^l(\mathbf{x})^{M-1}]^T$ is the vector with M polynomial basis features of the LF model, and $\boldsymbol{\rho} = [\rho_0, \rho_1, \ldots, \rho_{M-1}]^T$ is the corresponding coefficient vector to be determined from training on the HF data, i.e. these coefficients are not imposed as hyperparameters. This contrasts with the models reviewed in Section 4.2.1 that reduce Equation (4.5) to Equation (4.2) by considering $\mathbf{m}(\mathbf{x}) = f^l(\mathbf{x})$ and then considering ρ_1 as a hyperparameter.

In addition to defining the transfer-learning model, there are many possible choices for LF and HF models. Table 4.1 summarizes common models, as reviewed in the introduction. We suggest two different MF model choices for spanning a large number of data-scarce and data-rich MF regression problems: 1) kernel ridge regression together with Gaussian process regression (KRR-LR-GPR)[1] for data-scarce or low-dimensionality scenarios; and 2) DNN together with a Bayesian neural network (DNN-LR-BNN) for data-rich or multi-output scenarios.

Table 4.1: Candidates for LF and HF surrogates within the MF-BML framework

LF surrogates	HF surrogates
Linear regression (LR)	Gaussian process regression (GPR)
Kernel ridge regression (KRR)	Polynomial chaos expansion (PCE)
Deep neural network (DNN)	Bayesian neural network (BNN)
Support Vector Regression (SVM)	...
Gaussian process regression (GPR) *	
Polynomial chaos expansion (PCE) *	
...	

* Often used in conjunction with the same model at the HF, and considering $\mathbf{m}(\mathbf{x}) = f^l(\mathbf{x})$. Examples: GPR-LR-GPR in the form of Co-Kriging [92] or Hierarchical Kriging [97]; PCE-LR-PCE [105]; among others [122].

Remark 4.3.1. *LF datasets are usually larger than HF datasets because, by definition, LF data is acquired faster than HF data. If the LF dataset is large, deterministic LF models are advantageous because probabilistic models are not sufficiently scalable. In addition, deterministic LF models have lower complexity and fewer hyperparameters, while still being able to handle noisy data (aleatoric uncertainty).*

[1] Naming convention for the MF model (<LF model>-<transfer-learning model>-<residual model>): the abbreviation on the left is the surrogate for the LF model, the middle is the transfer-learning model, and the last abbreviation is the model for the residual. In the absence of one of the models, for example, not considering the residual, then the abbreviation only has two models (e.g. DNN-BNN refers to a DNN model for LF with a BNN doing the transfer-learning and no residual).

Remark 4.3.2. *The key disadvantage of choosing deterministic LF models is their inability to characterize epistemic (or model) uncertainty (at the LF). However, this will be shown to have limited relevance for the LF model when the residual or the transfer-learning model is probabilistic. In such cases, the HF predictions include uncertainty quantification and harness the typical advantages of probabilistic models.*

4.3.1. MODEL FOR DATA-SCARCE OR LOW-DIMENSIONAL SCENARIOS: KRR-LR-GPR

GPRs are one of the most successful models for data-scarce scenarios [44], having only a few hyperparameters and performing Bayesian inference for Gaussian observation distributions and priors without needing numerical integration. Consequently, they are easy to train for small datasets, making them an important HF model.

We argue that a logical LF model to pair with GPRs in a data-scarce HF scenario is KRR, i.e. the deterministic formulation of GPR [27]. KRR is a kernel machine learning method that is a point estimate of a GPR with the same kernel, therefore also equivalent to a DNN with an infinitely wide hidden layer [123]. KRR is robust to noisy data, and easy to train due to having few hyperparameters. More importantly, existing KRR approaches can lower complexity to a range between $\mathcal{O}(N^2)$ and $\mathcal{O}(N)$, thus handling larger datasets than PCE or GPR [124, 125] – invaluable for LF datasets because they are typically larger than HF ones. Other LF models, such as LR or DNNs (e.g., see Table 4.1) have important drawbacks. LR models are sensitive to the choice of the basis functions (often considered to be polynomials whose order is a hyperparameter); this is why we argue for their use as a simple transfer-learning model $g(\mathbf{x})$ but not as a LF or residual model. Conversely, DNNs have a large number of hyperparameters and are therefore less practical to train. Good practice involves starting with simpler models such as KRR that are easier to train and only considering DNNs when the simpler methods fail. We note that GPRs can also be a good choice for a LF model if the LF dataset is small, but training and inference time increase when compared to KRR, as shown later.

A common kernel of choice for both KRR and GPR when there is not enough prior information is the RBF kernel [27], where detailed introduced and derivation can be referred to Section 2.3.1. It can be formulated as:

$$k(\mathbf{x}^i, \mathbf{x}^j) = \exp\left(- \sum_{c=1}^{d} \theta_l \left(x_c^i - x_c^j \right)^2 \right) \tag{4.6}$$

where $\boldsymbol{\theta}$ is a d-dimensional vector that controls the length scale of each dimension.

The proposed KRR-LR-GPR model established on Equation (4.5) is easy to implement and the main steps are listed in Algorithm 6, while additional details are provided in Appendix C.1. Recall that the key idea is to train a GPR model where, instead of considering a zero mean function, we assume it to be a linear model whose features are the LF model. This is advantageous as there is an explicit solution for finding the parameters $\boldsymbol{\rho}$ from the HF data, no longer treating them as hyperparameters: Step 2.1 in Algorithm 6 or Equation (C.4) in Appendix C.1. The mean model of the GPR then captures most of the HF response by linear transfer-learning of the LF model, leaving the remaining nonparametric approximation for the residual via a Gaussian process whose kernel hyperparameters are optimized as usual (Step 2.2 in Algorithm 6). The resulting KRR-LR-GPR model predicts

the response and corresponding uncertainty, usually by following a Normal distribution $\mathcal{N}(\hat{f}^h(\mathbf{x}), \hat{\sigma}_h^2(\mathbf{x}))$ to ensure fast training and inference (exact integration of posterior and posterior predictive distributions). We find that a first-order linear transfer-learning model including a bias term, i.e. $\mathbf{m}(\mathbf{x}) = [1, f^l(\mathbf{x})]^T$, provides robust results according to the ablation study for different problems (see Appendix C.4).

Algorithm 6: KRR-LR-GPR model training

Data: LF dataset $\mathcal{D}(\mathbf{X}^l, \mathbf{y}^l)$, HF dataset $\mathcal{D}(\mathbf{X}^h, \mathbf{y}^h)$
Result: $\mathcal{N}(\hat{f}^h(\mathbf{x}), \hat{\sigma}_h^2(\mathbf{x}))$
Step 1: Train $f^l(\mathbf{x})$ by optimizing $\boldsymbol{\theta}^l$ based on LF dataset $\mathcal{D}(\mathbf{X}^l, \mathbf{y}^l)$
Step 2: Train $g(\mathbf{x})$ and $r(\mathbf{x})$ based on $\mathcal{D}(\mathbf{X}^h, \mathbf{y}^h)$ and $f^l(\mathbf{x})$
  Step 2.1: Calculate $\hat{\boldsymbol{\rho}} = \left(\mathbf{m}(\mathbf{X}^h)\mathbf{K}(\mathbf{X}^h, \mathbf{X}^h)^{-1}\mathbf{m}(\mathbf{X}^h)^T\right)^{-1}\mathbf{m}(\mathbf{X}^h)\mathbf{K}(\mathbf{X}^h, \mathbf{X}^h)^{-1}\mathbf{y}^h$
  Step 2.2: Optimize concentrated ln-likelihood function for $\boldsymbol{\theta}^h$

4.3.2. MODEL FOR DATA-RICH OR HIGH-DIMENSIONAL SCENARIOS: DNN-LR-BNN

Pairing a DNN as the LF model with a BNN in a MF model leverages the scalability of DNNs with the predictive abilities of BNNs. We propose a novel configuration summarized in Figure 4.2 labeled DNN-LR-BNN, where transfer-learning is done via LR as in Equation (4.5), and the residual is modeled by a BNN.

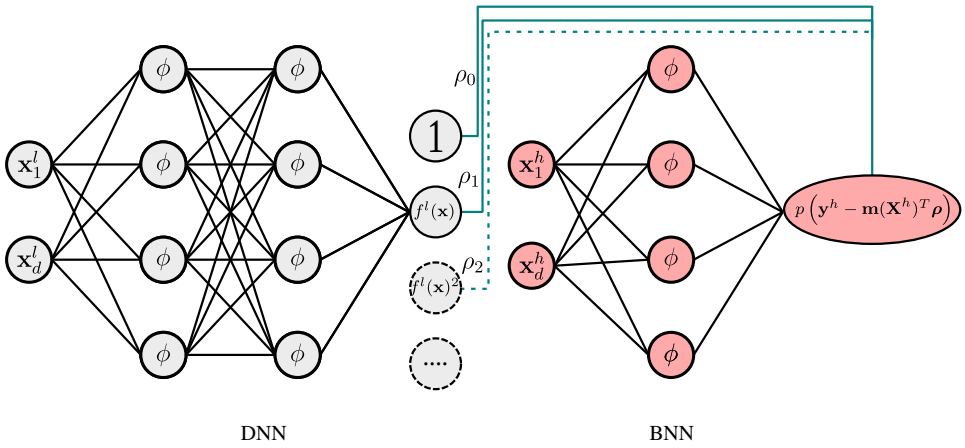


Figure 4.2: Schematic of the DNN-LR-BNN architecture. The DNN is trained on LF data, then it is used as a basis function of a linear transfer-learning model g that better explains the HF data and that is represented by the green connections, leading to $\mathbf{m}(\mathbf{x}^h)^T\boldsymbol{\rho}$. The BNN is then trained on the residual $\mathbf{r}(\mathbf{x}^h) = \mathbf{y}^h - \mathbf{m}(\mathbf{x}^h)^T\boldsymbol{\rho}$.

In the DNN-LR-BNN model, the BNN learns the residual $r(\mathbf{x}^h) = \mathbf{y}^h - \mathbf{m}(\mathbf{x}^h)^T\boldsymbol{\rho}$, where the order of $\mathbf{m}(\mathbf{x}^h)$ is a hyperparameter. Usually, we assume Normal distributions for the prior and observation distribution of BNNs, leading to a Normal PPD; Therefore, the DNN-LR-BNN model also has a Normal PPD. As mentioned previously, there are many viable strategies for inference in BNNs: from Markov Chain Monte Carlo approaches like Hamiltonian Monte Carlo [49, 120] to SGLD [50], passing through Variational Inference

approaches such as Bayes by Back-propagation [72]. In practice, for large datasets, we recommend the use of a variant of SGLD called pSGLD [52] due to its superior scalability [126] – additional details are provided in Section 2.5.2

Contrary to the KRR-LR-GPR model, the determination of the transfer-learning parameters $\boldsymbol{\rho}$ is not done by Equation (C.4) because it becomes intractable to calculate the covariance matrix of a BNN. Simultaneously, using cross-validation or other hyperparameter tuning strategies can be computationally intensive due to the time needed for BNN inference. Therefore, we propose to estimate $\boldsymbol{\rho}$ by solving the following optimization problem before training the BNN:

$$\hat{\boldsymbol{\rho}} = \arg\min_{\boldsymbol{\rho}} \sum_{i=1}^{N^h} \left\| \mathbf{y}^h - \mathbf{m}(\mathbf{X}^h)^T \boldsymbol{\rho} \right\|^2 \tag{4.7}$$

This equation is easy to optimize, akin to what is done in linear regression. Note that if we assumed $\rho_0 = 0$ and $\rho_1 = 1$, a vanilla MF model would be obtained (fusing a DNN with a BNN without additional parameters). The main steps of DNN-LR-BNN are listed in Algorithm 7, leading to a prediction following a Normal distribution $\mathcal{N}(\hat{f}^h(\mathbf{x}), \hat{\sigma}_h^2(\mathbf{x}))$ at any unknown point.

Algorithm 7: DNN-LR-BNN model training

Data: LF dataset $\mathscr{D}(\mathbf{X}^l, \mathbf{y}^l)$, HF dataset $\mathscr{D}(\mathbf{X}^h, \mathbf{y}^h)$
Result: $\mathcal{N}(\hat{f}^h(\mathbf{x}), \hat{\sigma}_h^2(\mathbf{x}))$
Step 1: Train DNN based on LF dataset $\mathscr{D}(\mathbf{X}^l, \mathbf{y}^l)$ and corresponding DNN settings
Step 2: Train $g(\mathbf{x})$ and $r(\mathbf{x})$ based on $\mathscr{D}(\mathbf{X}^h, \mathbf{y}^h)$ and $f^l(\mathbf{x})$
  Step 2.1: Obtain $\boldsymbol{\rho}$ by minimizing Equation (4.7)
  Step 2.2: BNN inference with pSGLD

4.4. EXPERIMENTS

We separately analyze the performance of KRR-LR-GPR and DNN-LR-BNN models and compare them with state-of-the-art strategies. For comparison with the KRR-LR-GPR method, we select Co-Kriging [27], Hierarchical Kriging [97], and Scaled Kriging [101]. We first consider 10 different functions to be learned using datasets with two fidelity levels, as described in Appendix C.2. These are low-dimensional functions with an input dimension spanning from $d = 1$ to $d = 8$, so they only need a small number of HF training samples. Importantly, the datasets utilized in this section are assumed to be noisy[2]. Additional experiments are conducted in Appendix C.3 where noiseless datasets are also considered. Then, we train KRR-LR-GPR to predict the aerodynamic coefficients for a NACA 0012 airfoil in Section 4.4.1. Concerning the performance assessment of DNN-LR-BNN, comparison with other models is more challenging due to the multitude of hyperparameters and model choices that are possible when considering neural networks in different fidelity levels. Therefore, we compare DNN-LR-BNN with single-fidelity BNN and DNN-BNN models [90] considering the same hyperparameters, as summarized in Appendix C.5. Several different numerical examples are considered in Section 4.4.2 and

[2]We implemented the existing methods such that noisy datasets could also be considered.

Section 4.4.2; and we finally validate its performance with a material structure-property linkages problem in Section 4.4.2.

For model comparison, considering that no single metric can objectively evaluate model performance, we use NRMSE and R2 Score to assess the predicted mean performance [127]. Concerning the uncertainty quantification metric, we adopt the TLL when we only have access to the noisy test dataset, as occurs in practice [128]. In addition, we implemented all the algorithms by ourselves and in the same coding language to record the CPU execution time for LF and HF. Each experiment is conducted on a node of an HPC cluster platform with an Intel® Xeon(R) E5-2643v3 CPU with 6 cores of 3.40GHz and 128 GB of RAM. Rigorous comparison of computational cost is challenging, as each method can be optimized differently. Our implementations and results are made publicly available, in an attempt to demonstrate fairness in our comparisons and motivate future research on this topic.

4.4.1. DATA-SCARCE HF AND LOW-DIMENSIONAL PROBLEMS: KRR-LR-GPR

ILLUSTRATIVE EXAMPLE

Given the traditional importance of the Forrester function [27, 97], we start by illustrating the effectiveness of the KRR-LR-GPR method for this example – see Equation (C.8). First, we compare the proposed method with methods in the literature that involve GPR models for both fidelity levels. We begin in Figure 4.3 and Figure 4.4 by considering LF and HF datasets with 200 and 7 uniformly sampled points, respectively. Note that our datasets include Gaussian noise $\mathcal{N}(0, 0.3^2)$ on both fidelity levels (LF and HF). Figure 4.6 considers different LF dataset sizes and different noise levels. Unsurprisingly, due to the simplicity of the Forrester function and its low-dimensionality, this first example was found to be trivial for most methods.

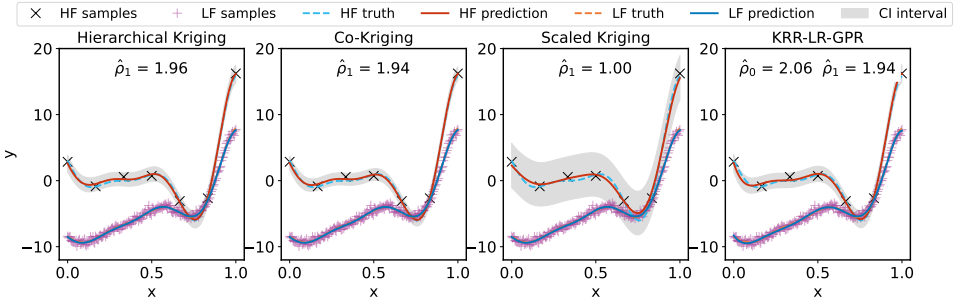


Figure 4.3: Fitting performance of KRR-LR-GPR on the illustrative example.

Figure 4.3 shows the results for all three MF-GPR models compared to the proposed KRR-LR-GPR model (for one of the 5 realizations chosen at random). Table 4.2 summarizes the performance metrics obtained from 5 independent runs with random seeds for each method. As mentioned, the Forrester function is easily predicted by all methods, except Scaled Kriging. The HF predicted means overlap almost perfectly with the HF ground-truth, as quantified in Table 4.2 by an NRMSE approaching zero and an R2 Score close to

Table 4.2: Performance comparison between KRR-LR-GPR and other MF-GPR methods on the illustrative example over 5 independent runs.

Methods	NRMSE	R2 Score	TLL	$\hat{\sigma}_h$	Training time(s)	
					LF	HF
Hierarchical Kriging	**0.1408** (±0.044)	**0.9927** (±0.005)	**-0.7895** (±0.189)	0.4360 (±0.167)	4.2705 (±1.048)	0.2429 (±0.056)
Co-Kriging	0.1429 (±0.041)	0.9925 (±0.004)	-0.9651 (±0.468)	**0.3446** (±0.178)	4.2305 (±0.676)	0.4458 (±0.078)
Scaled Kriging	0.2435 (±0.032)	0.9796 (±0.005)	-1.3306 (±0.359)	0.5284 (±0.485)	4.0597 (±0.792)	0.2202 (±0.038)
KRR-LR-GPR	0.1455 (±0.047)	0.9922 (±0.005)	-0.8573 (±0.308)	0.3803 (±0.164)	**2.4128** (±1.348)	0.2422 (±0.037)

1. Co-Kriging, Hierarchical Kriging, and KRR-LR-GPR all converge to similar $\hat{\rho}_1$. Note that Scaled Kriging considers $\hat{\rho}_1 = 1$, leading to inferior predictions. Unsurprisingly, KRR-LR-GPR requires less training time than all MF-GPR methods due to the use of KRR for the LF.

Figure 4.4 and Table 4.3 summarize the performance of KRR-LR-GPR when considering datasets with different correlations between LF and HF data. Three different LF functions are used to generate three different LF datasets, as explained in Appendix C.2, leading to the labels *LF data 1*, *LF data 2* and *LF data 3* in Figure 4.4.

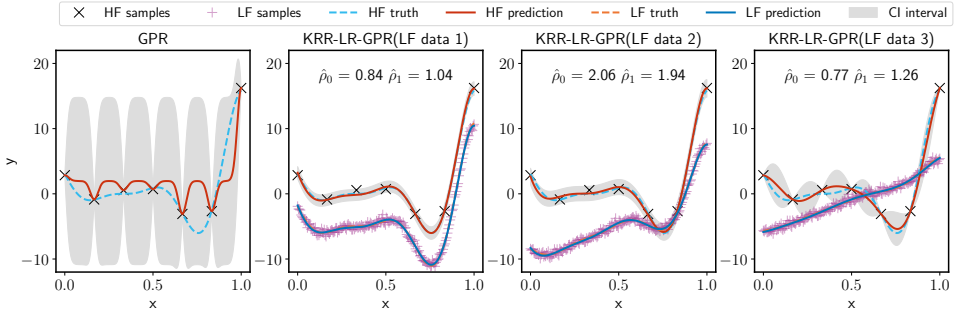


Figure 4.4: Predictions of KRR-LR-GPR for the Forrester function with different LF datasets of decreasing correlation with the HF dataset (from left to right), as indicated in Table 4.3. Different correlations between LF and HF data are obtained according to Equation (C.9), where the parameters A, B, and C of the LF function are changed as follows: *LF data 1* considers $A = 1, B = 0, C = 5$; *LF data 2* considers $A = 0.5, B = 10, C = 5$; *LF data 3* considers $A = 0.1, B = 10, C = 0.1$. The figure on the left corresponds to the single-fidelity result of GPR.

Figure 4.4 and Table 4.3 clarify that the single-fidelity GPR model cannot learn the underlying function well because it fails to identify the noise level. However, the transfer-learned LF model of KRR-LR-GPR provides appropriate conditioning of the GPR residual and leads to good predictions (see appropriate performance scores, NRMSE, and R2 Score). Performance deteriorates as the Pearson correlation coefficient r between the two fidelity levels decreases from *LF data 1* to *3*. Similarly, model uncertainty at the HF is lower when the LF datasets are more correlated with the HF dataset (higher Pearson correlation

Table 4.3: Summary of performance of KRR-LR-GPR model considering different LF datasets over 5 independent runs.

Methods		GPR	KRR-LR-GPR		
			LF data 1	LF data 2	LF data 3
Pearson cor. coef. r		–	1.0	0.737	0.407
NRMSE		1.3176 (±0.006)	0.0823 (±0.024)	0.1216 (±0.025)	0.6524 (±0.057)
R2 Score		0.4147 (±0.005)	0.9975 (±0.001)	0.9948 (±0.002)	0.8556 (±0.024)
TLL		-3.7502 (±1.490)	-0.5303 (±0.300)	-0.8650 (±0.391)	-2.2474 (±0.094)
Training time (s)	HF	0.1560 (±0.031)	0.1731(±0.023)	0.1707 (±0.016)	0.1398 (±0.039)
	LF	–	1.5543 (±0.562)	1.3730 (±0.326)	1.2043 (±0.523)
$\hat{\sigma}_h$		0.000060 (±0.000001)	0.2851 (±0.050)	0.2447 (±0.082)	2.6200 (±0.307)

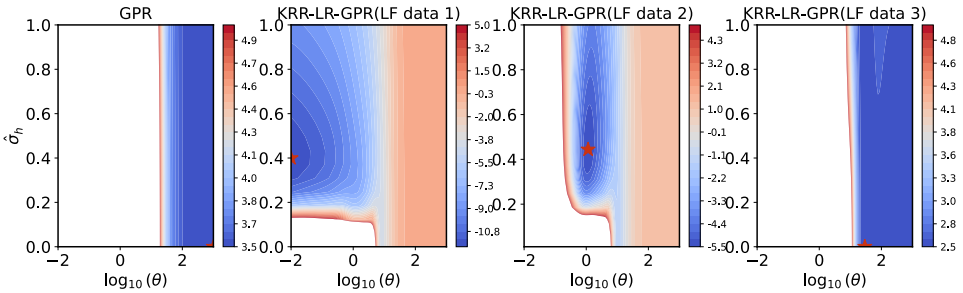


Figure 4.5: Negative log marginal likelihood values of GPR and KRR-LR-GPR within parameter space. The contour plots only show regions where the negative log marginal likelihood value is smaller than 5 for better illustration, the maximum values for each method are more than 10^7. The X-axis represents kernel parameter $\log_{10}(\theta)$ and the Y-axis is the estimated noise level $\hat{\sigma}_h$, the red point is the best parameter found for every method by "L-BFGS-B" with 10 restarts.

coefficient). Figure 4.5 also explains this by showing the negative log marginal likelihood landscape in the parameter space for a particular run of each of the three cases, where it is highlighted that in the third case (*LF data 3*) there is no clear optimum value (rightmost figure), exhibiting a similar marginal likelihood to the single-fidelity GPR model (leftmost figure) – this explains why the noise level at the HF is not properly estimated. The results are intuitive: if the LF data has limited or nonexistent correlation with the HF, then there is a degradation of the prediction quality of the MF model.

Furthermore, we investigate the performance of the KRR-LR-GPR model with different noise levels and different LF dataset sizes (see Figure 4.6 and Table 4.4). We consider different Gaussian noise for the LF with standard deviations of $\sigma_l = 0.0$, $\sigma_l = 0.3$, $\sigma_l = 0.5$, and $\sigma_l = 1.0$, such that the robustness of KRR-LR-GPR to LF noise is assessed while maintaining the HF noise as $\sigma_h = 0.3$. The figure and table clarify that the model predicts both the function and noise level accurately for the cases with enough LF data. Conversely, it struggles to identify proper noise levels when the LF dataset only has 11 points and for higher LF noise levels, as expected.

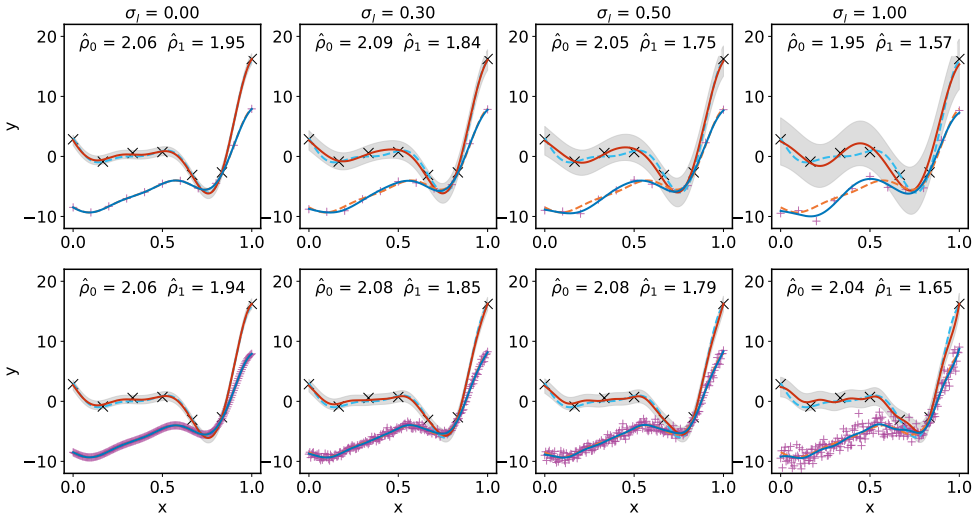


Figure 4.6: Performance of KRR-LR-GPR model on Forrester function with different number of LF samples and noise levels: the top and bottom rows show results obtained for a LF dataset with 11 and 200 samples, respectively. In this figure, the HF dataset is not changed and considers 7 samples. The noise level of LF is indicated by σ_l. The number of LF samples is clear from the cross markers in each plot.

Table 4.4: Results of KRR-LR-GPR on Forrester function with different LF samples and LF noise levels over 5 independent runs

LF samples	σ_l	NRMSE	R2 Score	TLL	$\hat{\sigma}_h$
11	0.0	0.1267 (±0.012)	0.9945 (±0.001)	-0.6279 (±0.007)	0.3300 (±0.052)
	0.3	0.2572 (±0.088)	0.9751 (±0.018)	-11304.37 (±14787.37)	0.3229 (±0.378)
	0.5	0.3184 (±0.073)	0.9640 (±0.016)	-11202.4 (±14779.75)	0.4992 (±0.523)
	1.0	0.4125 (±0.139)	0.9361 (±0.045)	-11128.88 (±14842.75)	0.6761 (±0.682)
200	0.0	0.1184 (±0.015)	0.9952 (±0.001)	-0.6106 (±0.092)	0.3374 (±0.057)
	0.3	0.1119 (±0.016)	0.9957 (±0.001)	-0.5398 (±0.080)	0.3159 (±0.017)
	0.5	0.1184 (±0.020)	0.9951 (±0.001)	-0.5693 (±0.094)	0.3136 (±0.061)
	1.0	0.1563 (±0.026)	0.9915 (±0.003)	-0.7243 (±0.121)	0.3731 (±0.055)

COMPREHENSIVE NUMERICAL EXPERIMENTS OF KRR-LR-GPR

The methods mentioned above are trained on 10 other analytical benchmark functions, reporting results by repeating every experiment with 5 random initializations of the training data generation. All 10 functions lead to similar conclusions; therefore, we arbitrarily select the *Booth* function to show in the main text (results for other functions are reported in Appendix C.3). Figure 4.7 shows the results assuming a fixed number of LF samples ($200 \times d$ where d is the dimension of the particular function being trained), and then considering a different number of HF samples (from $5 \times d$ to $30 \times d$). Figure 4.8 pertains to a different experiment where we fix the number of HF samples to $20 \times d$, and consider different numbers of LF samples (from $50 \times d$ to $300 \times d$). Moreover, Gaussian noise with standard deviations of $\sigma_l = 0.3$ and $\sigma_h = 0.3$ is considered for the datasets in both cases.

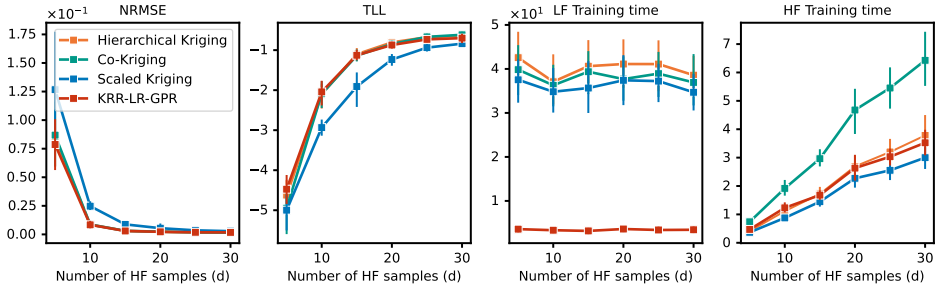


Figure 4.7: Comparison of KRR-LR-GPR with state-of-the-art MF methods for different sizes of HF datasets when learning the Booth function, and considering $200 \times d$ LF samples (in the case of the Booth function $d = 2$, and the Pearson correlation coefficient is $r = 0.925$). Different colors represent different methods. To interpret the results, note that smaller NRMSE or higher TLL indicate better performance. Note the significantly faster LF training time for KRR-LR-GPR compared to others.

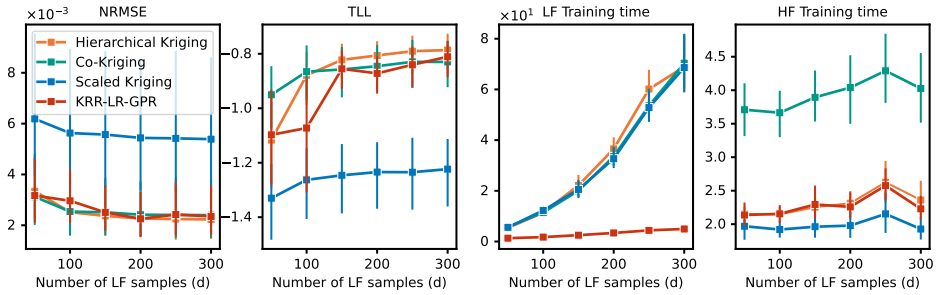


Figure 4.8: Comparison of KRR-LR-GPR with state-of-the-art methods when learning the *Booth* function but now considering different sizes of the LF dataset while keeping the number of HF samples fixed to $20 \times d$.

The figures (Figure 4.7, Figure 4.8, Figure C.1, Figure C.2) reinforce the argument that considering KRR for the LF and GPR for the residual creates a MF model with comparable performance to considering a GPR for all fidelity levels (hierarchically or not), while the computational cost decreases dramatically. Essentially, the computational complexity is reduced to $\mathcal{O}((N^h)^3 + (N^l)^2)$ instead of $\mathcal{O}((N^h + N^l)^3)$. For cases where the HF training data is scarce and low dimensional, we recommend using a Bayesian method like GPR for that fidelity and a scalable method like KRR for the LF. It is noted that we also conducted experiments with different HF noise levels in Appendix C.4 as well as an ablation study on changing both LF and HF samples in Appendix C.3, those results also support the above arguments.

ENGINEERING APPLICATION: NACA 0012 AIRFOIL

The KRR-LR-GPR method is also used to determine the aerodynamic coefficients of a NACA 0012 airfoil, a common engineering problem for assessing MF models [129, 130]. The aim is to predict the lift and drag coefficients (C_l and C_d) considering two independent variables (freestream Mach number MA and angle of attack AoA). The dataset was obtained from [129], in which 5 HF points (simulated by Reynolds-averaged

Navier Stokes equation) and 40 LF points (simulated by Euler equation) are provided for training, and 20 HF points are used for testing. Table 4.5 presents the results, and Figure 4.9 offers a comparison between all selected data-scarce MF methods.

Table 4.5: Results of different data-scarce methods for the NACA0012 airfoil MF problem for the two QoIs.

QoI	Methods	NRMSE	R2 Score	TLL	Training time (s)	
					HF	LF
C_l	Hierarchical Kriging	0.0058	**0.9999**	0.1636	0.0686	1.6828
	Co-Kriging	0.0065	**0.9999**	4.2375	0.2138	1.4089
	Scaled Kriging	0.0085	**0.9999**	3.3848	0.1270	2.1036
	KRR-LR-GPR	**0.0057**	**0.9999**	**4.3414**	**0.0258**	**1.0354**
C_d	Hierarchical Kriging	0.1038	0.9107	**5.9189**	0.1259	1.2885
	Co-Kriging	0.1089	0.9016	4.7347	0.1214	1.3689
	Scaled Kriging	0.0987	0.9192	5.2907	0.0799	1.4836
	KRR-LR-GPR	**0.0923**	**0.9293**	5.5690	**0.0674**	**0.0745**

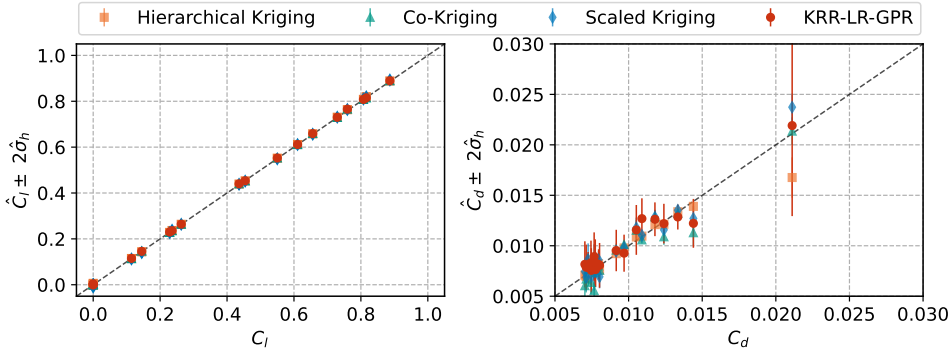


Figure 4.9: Comparison results of the NACA0012 airfoil. The error bars of each point in the plot indicate predicted uncertainty

Table 4.5 shows that the performance of the proposed KRR-LR-GPR is better than other methods, especially when predicting C_d but also for C_l (observe the NRMSE values). The TLL value when predicting C_d is the only metric that is marginally worse than Hierarchical Kriging. We note that KRR-LR-GPR is the fastest method to train, although this is not beneficial in this example because the HF and LF data acquisition time is much larger than the time needed to train any of the MF algorithms. In fact, this example is data-scarce even in the LF (only 40 points), and so the better scalability of KRR for LF regression is not needed in this case (this only becomes relevant when the LF dataset has over a few thousand points).

4.4.2. DATA-RICH HF AND HIGH-DIMENSIONAL PROBLEMS: DNN-LR-BNN

ILLUSTRATIVE EXAMPLE

We start with a one-dimensional illustrative example (data-scarce and low-dimensional) with 11 HF samples and 201 LF samples distributed uniformly. Aiming at explaining the limitations of the different MF models with DNNs and BNNs, we select three different LF datasets, labeled as "LF data 1", "LF data 2" and "LF data 3", obtained by establishing a different correlation with the HF data according to Equation (C.30b), (C.30c) and (C.30d), respectively. Model performance and accuracy metrics are shown in Figure 4.10 and Table 4.6.

Table 4.6: Results comparison of DNN-LR-BNN on the 1D noisy illustrative example with 5 independent runs

LF Dataset	Methods	NRMSE	R2 Score	TLL
—	BNN	1.0470 (±0.028)	-0.6888 (±0.093)	-5.7825 (±0.578)
LF Data 1 (r = -0.1019)	DNN-BNN	0.4865 (±0.163)	0.6029 (±0.247)	**0.2808** (±0.308)
	DNN-LR^1-BNN	1.0501 (±0.013)	-0.6983 (±0.043)	-6.4183 (±0.662)
	DNN-LR^2-BNN	**0.3161** (±0.005)	**0.8461** (±0.005)	0.2329(±0.090)
LF Data 2 (r = 0.9999)	DNN-BNN	0.3743 (±0.092)	0.7738 (±0.102)	0.5946 (±0.136)
	DNN-LR^1-BNN	**0.1051** (±0.005)	**0.9830** (±0.001)	**1.2296** (±0.018)
LF Data 3 (r = -0.0018)	DNN-BNN	1.1012 (±0.115)	-0.8834 (±0.403)	**-5.5084** (±1.899)
	DNN-LR^1-BNN	**1.0643** (±0.012)	**-0.7445** (±0.038)	-6.1546 (±0.711)

* The subscripts 1 and 2 indicate that the LR model used for transfer-learning is order 1 (linear polynomial basis functions) and order 2 (quadratic polynomial basis functions), respectively.

The datasets *LF data 1* and *LF data 3* were deliberately created with a low Pearson correlation coefficient concerning the HF dataset. Figure 4.10 and Table 4.6 show that for such cases the LR transfer-learning model used in DNN-LR-BNN automatically finds low values for the ρ_1 coefficient after training the LF model on the HF data (independently of the BNN training on the residual). We believe that this is a strength of the proposed strategy because it automatically adjusts the influence of the LF model and informs the analyst about the correlation between LF and HF data. We note that the order of the LR transfer-learning model is a hyperparameter, but the coefficients $\boldsymbol{\rho}$ are found by training via minimizing Equation (4.7). Simple hyperparameter search strategies can quickly find the optimal order of the LR transfer-learning model for each problem (it is only one hyperparameter). For example, a quadratic LR transfer-learning model is better than a linear one for "LF data 1" (this is labeled as DNN-LR^2-BNN to distinguish from the LR model with linear basis functions labeled as DNN-LR^1-BNN). Evidently, if LF and HF data tend to be linearly correlated (as in the *LF data 2*), the DNN-LR-BNN model performs better.

Interestingly, even when the correlation between LF data and HF data is highly non-linear, we have not found the performance of the DNN-BNN to be better, despite this model using a BNN to perform transfer-learning instead of a simple LR model. The next sections demonstrate this for different datasets, including a material modeling example that illustrates a more realistic scenario encountered in engineering practice. This might explain why early MF literature considered first-order linear relationships between LF and HF models that were governed by a single hyperparameter (ρ_1). In essence, we find that

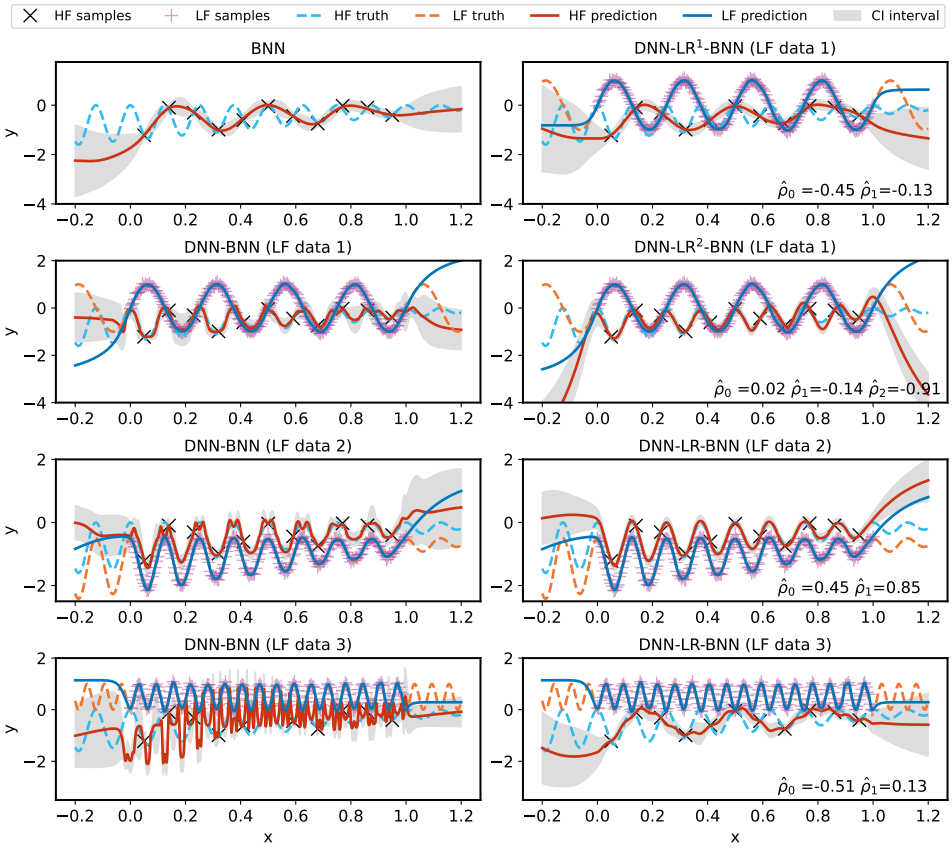


Figure 4.10: One-dimensional illustrative example comparing the proposed DNN-LR-BNN method (right column) with a BNN trained exclusively with HF data, and with a simple DNN-BNN (left column). Three LF datasets are employed whose Pearson correlation coefficients are −0.1019, 0.9999, and −0.0018 correspondingly.

performing model selection and establishing a simple (but not simpler) transfer-learning model between the DNN and BNN is an important consideration.

We also want to bring to the reader's attention that neural network training is highly dependent on architectural and hyperparametric choices. Appendix C.5 details the dependence on hyperparameters and reinforces that the DNN-LR-BNN model is simpler to train and is less sensitive to hyperparameter choices than the more flexible DNN-BNN model. We observed in all cases that the prediction of uncertainty depends on the width of the BNN, i.e. on the number of neurons in the hidden layers. A wide network leads to similar uncertainty predictions as the ones obtained by GPR, while the prediction of the expected value (the mean) converges quickly even for narrow BNNs.

HIGHER DIMENSIONAL NUMERICAL EXAMPLES

We also demonstrate the performance of DNN-LR-BNN and DNN-BNN in problems with higher dimensions and larger datasets. We consider two functions, a 4-dimensional

function extracted from [90] and another function from [116] where we set the dimension to be [20, 50, 100] in this subsection. The results are summarized in Table 4.7.

Table 4.7: Results comparison of DNN-LR-BNN on the high-dimensional examples

Dimension	Methods	NRMSE	R2 Score	TLL	$\hat{\rho}_0$	$\hat{\rho}_1$
4	BNN	0.7624 (±0.031)	-0.6296 (0.131)	-5.3555 (±0.458)	-	-
	DNN-BNN	0.1163 (±0.013)	0.9617 (±0.009)	1.2680 (±0.040)	-	-
	DNN-LR^1-BNN	**0.0777** (±0.013)	**0.9826** (±0.006)	**1.4090** (±0.045)	-0.41 (±0.002)	0.86 (±0.006)
20	BNN	0.1598 (±0.004)	0.7418 (±0.013)	-23.5060 (±1.673)	-	-
	DNN-BNN	0.0723 (±0.010)	0.9463 (±0.015)	**-8.1125** (±1.137)	-	-
	DNN-LR^1-BNN	**0.0662** (±0.008)	**0.9550** (±0.012)	-8.7991 (±0.905)	55.75 (±17.091)	1.26 (±0.005)
50	BNN	0.1725 (±0.003)	0.2341 (±0.024)	-59.2736 (±3.671)	-	-
	DNN-BNN	0.0879 (±0.001)	0.8008 (±0.007)	-17.7602 (±0.9644)	-	-
	DNN-LR^1-BNN	**0.0851** (±0.001)	**0.8137** (±0.003)	**-17.3543** (±0.0.8955)	62.0892 (±5.219)	1.2504 (±0.001)
100	BNN	0.1343 (±0.001)	0.0564 (±0.020)	-113.4960 (±8.007)	-	-
	DNN-BNN	0.0867 (±0.002)	0.6062 (±0.018)	-58.8356 (±6.7228)	-	-
	DNN-LR^1-BNN	**0.0845** (±0.001)	**0.6265** (±0.008)	**-58.7305** (±4.458)	-100.0 (±0.0)	1.2868 (±0.013)

Table 4.7 shows that the single-fidelity BNN leads to poor predictions for all examples according to all accuracy metrics. Also for all cases, the DNN-LR-BNN outperforms the DNN-BNN model. The coefficients $\hat{\boldsymbol{\rho}}$ learned for the DNN-LR-BNN model are reasonable when compared to the ground-truth functions shown in Appendix C.2.

ENGINEERING APPLICATION: MATERIAL STRUCTURE-PROPERTY PREDICTION DATASET

This section focuses on finding structure-property relationships in materials, as described in [10]. The dataset considered here was introduced by Olivier et al. [32], where effective material properties of two-phase material microstructures are determined by computer simulations of material samples, called SVEs. Each SVE results from randomizing the material microstructure, then deforming the SVE according to periodic boundary conditions, and susbequently calculate the average (or homogenized) response of the SVE. The authors of that study only considered outputs from a single-fidelity, obtained by direct numerical analysis (DNS) via finite element analysis (FEA), and showed that a BNN can predict both the mean and standard deviation for chosen quantities of interest (QoIs). We extend this problem to the MF scenario by generating LF data using an open-source implementation [131] of a significantly faster simulation method called self-consistent clustering analysis (SCA) [87, 132]. Figure 4.11 summarizes the material structure-property regression problem, and additional information can be found in the

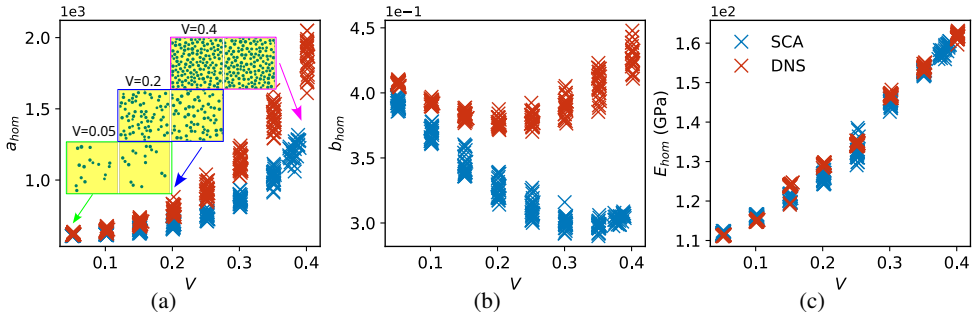


Figure 4.11: Material structure-property prediction problem illustration. This problem involves four input descriptors: volume fraction V of the particle material, Young's modulus of the linear elastic fibers $E_{\texttt{fiber}}$, matrix hardening law coefficients $\sigma = 400 + a_{\text{matrix}}\varepsilon_p^{b_{\text{matrix}}}$. The simulations of the material response lead to three outputs: homogenized Young's modulus E_{hom}, and homogenized hardening law with coefficients a_{hom} and b_{hom}. We utilize two fidelity levels for data generation: HF data is generated through a FEA method via commercial software (ABAQUS) based on [133] and the LF data is obtained via the SCA method from the CRATE open source package [131]. Figures (a), (b), and (c) show three examples of how one output property changes concerning a particular input variable, making it clear that there is aleatoric uncertainty (noise). Appendix C.5 contains additional information about the data generation process for this problem.

original publication [32]. We highlight that a reader lacking domain knowledge of the Mechanics of Materials and multi-scale simulations may choose to ignore the physical meaning of the input and output variables of this dataset. In that case, the reader only needs to know that there are four input variables with the following bounds:

$$\begin{aligned} V &\in [0.05, 0.40], \\ E_{\text{fiber}} &\in [200000, 600000], \\ a_{\text{matrix}} &\in [300, 500], \\ b_{\text{matrix}} &\in [0.2, 0.55], \end{aligned}$$

that lead to noisy measurements captured by three output variables: E_{hom}, a_{hom} and b_{hom}, where the subscript "hom" indicates that these properties correspond to the homogenized (average) response of the two-phase material. The LF output measurements are obtained by a fast but less accurate method, while the HF measurements of the same three output variables are obtained by a slower but more accurate method. Note that both LF and HF measurements are noisy, as the same input point leads to different output values each time it is evaluated (due to material microstructure randomness). Figure 4.12 illustrates the variation of one of the outputs with respect to one of the inputs, while keeping the remaining inputs fixed. In that figure, red crosses correspond to HF data (DNS) and blue crosses to LF data (SCA).

We generated 500 HF samples with the FEA method and split them into 400 for training and 100 for testing. In addition to this, we also generate 3200 LF samples using the SCA method (using 3 material clusters – the hyperparameter of SCA that controls discretization). We then test the proposed DNN-LR-BNN strategy on this problem where we use 300 HF samples and 3200 LF samples to train the MF models, i.e. DNN-BNN and DNN-LR-BNN. We also train a single-fidelity BNN model using 400 HF samples, as this

corresponds to the number of samples that can be generated with the same computational resources and time needed to create the MF dataset (in other words, 100 HF simulations generated with FEA require approximately the same time as 3200 LF simulations with the SCA method). Table 4.8 compares the performance of the different MF models, including the single-fidelity baseline of BNN. Figure 4.12 presents the output predictions compared to the measured output values in the test set. However, recall that the outputs are noisy and that their ground-truth mean values are not known, nor their corresponding aleatoric distribution (noise distribution). Therefore, the figure only shows the mean prediction (y-axis value), and corresponding epistemic uncertainty estimation (vertical bar) for each test sample (x-axis value).

Table 4.8: Results comparison of DNN-LR-BNN on the material structure-property linkages

QoIs	Methods	NRMSE	R2 Score	TLL	$\hat{\rho}_0$	$\hat{\rho}_1$	$\hat{\rho}_2$
a_{eff} ($r = 0.9294$)	BNN	0.0902	0.8165	-5.3577	-	-	
	DNN-BNN	0.1254	0.6450	-5.2192	-	-	
	DNN-LR^1-BNN	0.0700	0.8894	-5.3748	10.00	0.92	-
	DNN-LR^2-BNN	**0.0442**	**0.9558**	**-4.9248**	10.00	1.24	-3.55 $\times 10^{-4}$
b_{eff} ($r = 0.7681$)	BNN	0.0268	0.9489	3.2622	-	-	
	DNN-BNN	0.0345	0.9155	3.1881	-	-	
	DNN-LR^1-BNN	0.0178	0.9774	3.3436	0.10	0.88	-
	DNN-LR^2-BNN	**0.0134**	**0.9872**	**3.5999**	0.05	1.11	0.32
E_{eff} ($r = 0.9845$)	BNN	0.0025	0.9556	-10.8660	-	-	
	DNN-BNN	0.0229	0.9649	-10.2588	-	-	
	DNN-LR^1-BNN	0.0146	0.9857	-9.2766	-7.97	1.01	-
	DNN-LR^2-BNN	**0.0121**	**0.9901**	**-9.2447**	-1.27	0.99	-7.75 $\times 10^{-8}$

Table 4.8 compares different MF models and also includes the single-fidelity BNN prediction. This table clarifies that the proposed DNN-LR-BNN model with a quadratic linear regression transfer learning model (DNN-LR^2-BNN) has the best performance for this material structure-property problem. These results demonstrate the effectiveness of the DNN-LR-BNN method in real-world scenarios where the dataset contains multiple outputs. Figure 4.12 also highlights that the single-fidelity BNN has the largest predicted uncertainty for all test points. Conversely, the DNN-LR-BNN has a more reasonable uncertainty prediction, achieving the best TLL values among all methods. We can see that in cases where the outputs of the two fidelity levels have a high Pearson correlation coefficient ($r \to 1$), then the performance is similar for DNN-LR^1-BNN and DNN-LR^2-BNN because the training procedure identifies that the quadratic coefficient of the transfer-learning model in DNN-LR^2-BNN tends to zero ($\hat{\rho}_2 \to 0$). Instead, for the b_{hom} output where the two fidelity levels are not as linearly correlated, the quadratic transfer learning model improves the prediction, as it facilitates training the BNN on the HF data (note that in this case we have $\hat{\rho}_2 \neq 0$). We experienced difficulties in avoiding overfitting when training the DNN-BNN method for some of the outputs in this problem.

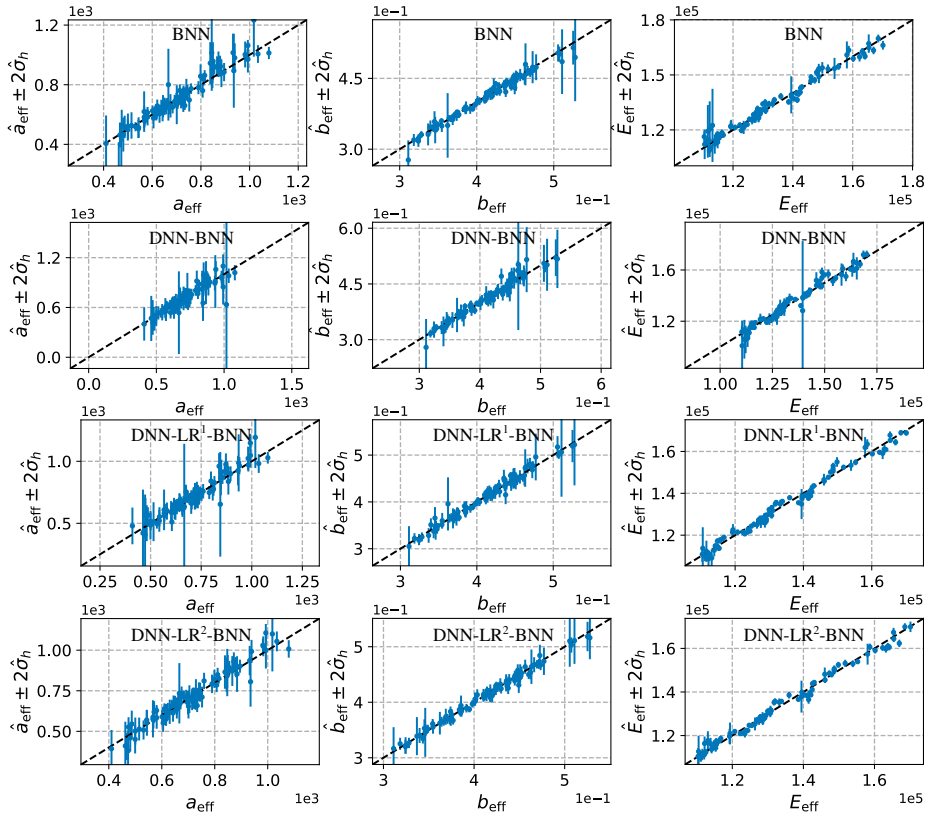


Figure 4.12: Predicted outputs with uncertainties for material structure-property linkages problem. The size of the points represents the predicted uncertainty

4.5. DISCUSSION AND LIMITATIONS

Expressive machine learning algorithms typically have limited extrapolation ability. Their large number of parameters is used to explain the training data without necessarily discovering the underlying functional form that explains the data. DNNs often make unreliable predictions for points away from the support domain; conversely, Bayesian models such as GPRs and BNNs eventually reverse to the prior [134]. A similar issue also occurs in the DNN-BNN architecture where the HF prediction is distorted in the unsampled region as reported in Figure 4.13. However, KRR-LR-GPR and DNN-LR-BNN do not exhibit this behavior because they remain smooth and close to the HF truth. The proposed transfer-learning model in Equation (4.5) narrows the gap between $\mathbf{m}(\mathbf{X}^h)^T\boldsymbol{\rho}$ and $\mathbf{y}^h$, while the BNN captures the residual utilizing a prior with zero mean that reverses to another constant after training (see purple line in Figure 4.13). Therefore, the HF prediction is influenced by the parsimonious (simple) transfer-learning model, improving its extrapolation ability.

However, when we compare the DNN-LR-BNN model prediction with the KRR-LR-GPR one, we see that although the mean is similar, the uncertainty prediction is not. The

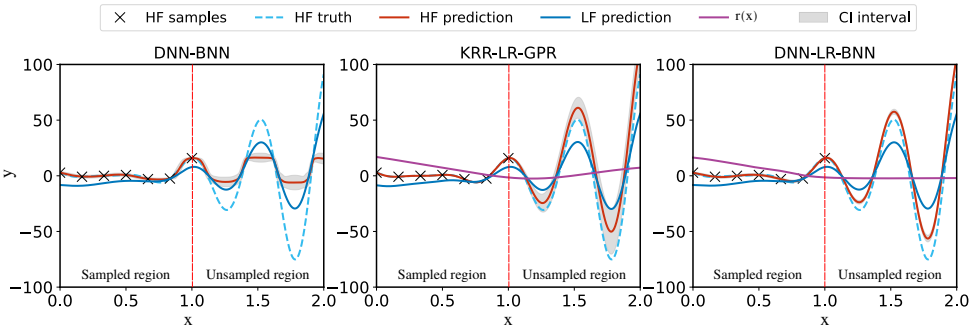


Figure 4.13: Illustration of what occurs to the DNN-BNN, KRR-LR-GPR and DNN-LR-BNN model predictions when there is a domain region without HF samples ($x > 1.0$), but considering LF samples throughout the entire domain ($0 < x < 2$).

predicted uncertainty of DNN-LR-BNN is smaller in the extrapolation region – the model is overconfident. We identify two reasons for this observation. First, BNN inference is approximate and does not yield the exact posterior predictive distribution. This bottleneck can be resolved for a specific problem by fine-tuning the BNN hyperparameters using test data or cross-validation, as well as by increasing the number of neurons in the hidden layers. Second, there is an inherent bottleneck in evaluating uncertainty via Equation (2.31) when a BNN is employed for $f^h(\mathbf{x})$ inference: the covariance matrix $\mathbf{K}(\mathbf{X},\mathbf{X})$ is difficult to compute and so the uncertainty present in the LF model is difficult to transport to the HF prediction.

4.6. CONCLUSIONS

In this study, we created a general MF framework that encompasses different MF strategies in the literature. We propose using non-probabilistic LF models together with Bayesian HF models and recommend two practical strategies to deal with data-scarce and data-rich scenarios. For the first case, we propose the KRR-LR-GPR MF model that considers KRR for training on LF data, linear regression for LF to HF data transfer-learning, and Gaussian process regression for modeling the residual that remains. For data-rich, high-dimensional cases and/or multiple outputs, we recommend the DNN-LR-BNN model that involves training a deep neural network on LF data, transfer-learning via linear regression, and Bayesian neural networks for learning the residual. If a linear regression model does not provide an effective transfer-learning method, a more flexible DNN-BNN model is recommended using pSGLD for Bayesian inference.

The KRR-LR-GPR model was assessed for cases with and without noise for both fidelity levels, and its performance was found to be similar to state-of-the-art MF-GPR methods. However, KRR-LR-GPR exhibits two key advantages: 1) it can be trained on larger LF datasets, and 2) it is significantly faster to train (in practice, the method is only bounded by the number of HF samples, due to the cubic complexity of GPR models with the number of training samples). The method was also successfully applied to a dataset from an engineering application – prediction of aerodynamic coefficients of NACA 0012 airfoil. We believe that the simplicity of the training strategy and robustness of the method

makes it suitable for most engineering applications because the most common scenario involves data-scarce HF samples (usually originating from expensive or time-consuming experiments or simulations).

We also proposed the DNN-LR-BNN model and compared it with both a single-fidelity BNN and a flexible MF model composed of a DNN and a BNN (DNN-BNN model). We observed that the DNN-LR-BNN model has a lower tendency to overfit when compared to DNN-BNN because its linear regression (LR) transfer-learning model isolates the HF from the LF when they are not linearly correlated. Furthermore, we highlight that the proposed strategy determines the coefficients of the LR model, $\boldsymbol{\rho}$, directly from training (instead of considering them as hyperparameters, as done in the literature for other models). The learned $\hat{\boldsymbol{\rho}}$ also indicates the correlation between HF and LF data, as shown in different examples such as the material structure-property relationship problem. Nevertheless, the DNN-LR-BNN model has more hyperparameters than the KRR-LR-GPR model, so it should be selected only when training the latter is not viable.

5

SINGLE- TO MULTI-FIDELITY HISTORY-DEPENDENT LEARNING WITH UNCERTAINTY DISENTANGLEMENT: APPLICATION TO DATA-DRIVEN CONSTITUTIVE MODELING

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 are 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 opens opportunities for future real-world applications in diverse scientific and engineering domains; especially, the most challenging cases involving design and analysis under uncertainty.

This chapter is based on the manuscript: **Yi, J.**, Ferreira, B. P., & Bessa, M. A. (2025). *Single-to multi-fidelity history-dependent learning with uncertainty quantification and disentanglement: application to data-driven constitutive modeling. Computer Methods in Applied Mechanics and Engineering.* https://doi.org/10.1016/j.cma.2025.118479

5.1. 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 [135, 136], in general, they require large quantities of data [137]. 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.

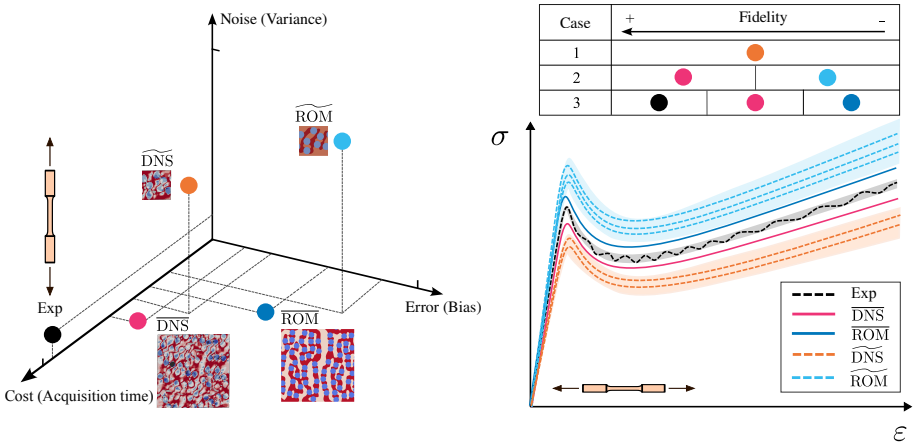


Figure 5.1: Schematic summarizing different data generation methods for discovering the constitutive behavior of materials. Each method and corresponding dataset from which it originates is organized according to cost, noise, and accuracy. A method or a dataset is considered to have **high-fidelity (HF)** 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 ($\overline{\text{DNS}}$)** 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 ($\widetilde{\text{DNS}}$)** 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 ($\overline{\text{ROM}}$)** 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 ($\widetilde{\text{ROM}}$)** is obtained by SCA of an SVE (noisy data with the lowest fidelity and fastest acquisition time).

Without loss of generality, the accuracy-efficiency paradox addressed by MF modeling will be illustrated in this chapter 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 [138], 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**

[1] This chapter 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.

elements (RVEs)[2] of materials, were shown to create sufficiently large datasets for training accurate models [10]. This can be achieved by finite element analyses of thousands of RVEs when they are composed of elastic or hyperelastic materials [139], but it becomes more challenging when they undergo history-dependent phenomena such as plasticity because the simulations are slower [10]. A viable pathway is to consider faster simulation methods, e.g., the **self-consistent clustering analysis (SCA)** [87], 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 [34]. Since then, several researchers followed data-driven strategies to learn constitutive models for a wide range of materials, e.g., [24, 140–143], including their use to accelerate topology optimization [144].

Figure 5.1 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 Figure 5.1). 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 Figure 5.1, 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 [145].

In addition to conducting real experiments, performing a deterministic DNS (denoted as $\overline{\text{DNS}}$) 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. Figure 5.1 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 $\widetilde{\text{DNS}}$) 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 volume elements (SVEs)** because randomization of the microstructure leads to different responses (indicated in orange in Figure 5.1). 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 chapter, we choose the ROM to be the previously mentioned SCA

[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 [10].

[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.

method [87], 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 [132] recently became open-source [131], so these (and all other) results obtained in this chapter 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 chapter generalizes data-driven learning to all types of datasets exemplified in Figure 5.1. 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 [145], 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 [146] 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 [147, 148] with the aim of merging them into a general learning paradigm, as presented herein.

Remark 5.1.1. *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,*

[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.

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 [15, 68, 149], as it facilitates decision-making and design optimization [150–152]. Initially, most solutions relied on Gaussian process regression [153], and assumed homoscedastic aleatoric uncertainty (constant noise). Even when considering heteroscedastic noise [33], 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 [49, 154]. Various approximate Bayesian inference methods have been proposed, such as Markov chain Monte Carlo (MCMC) sampling [49], Variational Inference (VI) [53], MC-Dropout [23], and Deep Ensembles [22]. This has enabled the recent application of BNNs to cases without history-dependency, such as estimating material properties [32, 155] and learning hyperelastic constitutive models [156]. 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 [82, 157, 158] for early work on the topic. Two recent methodologies have improved their scalability [148, 159]. In this work, we use the method we developed – **Variance estimation Bayesian recurrent neural networks (VeBRNNs)** [148] – 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)** [52], and they effectively disentangle uncertainties due to the cooperative training of a variance estimation network [62] 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 [10], but gathering it is often time-consuming and impractical. As previously referred, this is particularly relevant when involving complex experiments or expensive simulations [160]. Effective MF approaches can mitigate this challenge by leveraging the useful information embedded in LF data, thereby reducing the dependence on HF data [25, 26]. MF machine learning has been successfully

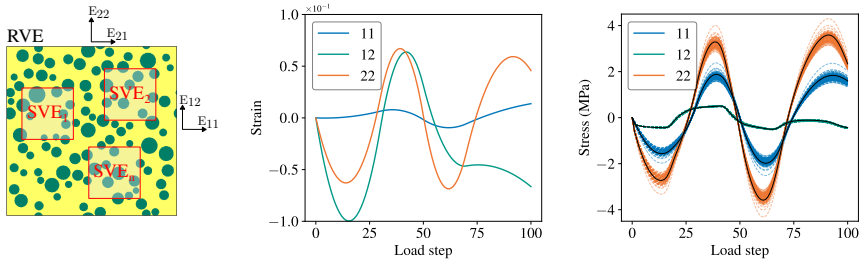
[5] Note, however, that estimating aleatoric uncertainty can still be beneficial because it regularizes the model for noisy data.

applied in various domains, such as solving expensive partial differential equations [116], optimization [111, 161], and material design [151]. Initially, MF machine learning was limited to data scarce scenarios and low-dimensional problems by considering MF Gaussian process regression [27, 90, 153]. However, the literature has adopted deterministic methods when dealing with large datasets, e.g., using MF recurrent neural networks [118], or continual learning of subnetworks [35] 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.

5.2. PROPOSED DATASETS: FOUR DIFFERENT SINGLE- AND MULTI-FIDELITY LEARNING SCENARIOS

We created four different scenarios based on Figure 5.1 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 (Figure 5.2a) are subjected to synthetic random polynomial strain paths (Figure 5.2b), from which the stress response is obtained (Figure 5.2c). The biphasic material consists of:

- a matrix phase (yellow area of Figure 5.2a) characterized by a von Mises elastoplastic model with isotropic strain hardening with elastic modulus $E_{\text{matrix}} = 100\,\text{MPa}$, Poisson's ratio $\nu_{\text{matrix}} = 0.3$, 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 Figure 5.2a) characterized by a linear elastic isotropic model with $E_{\text{fiber}} = 15\,\text{MPa}$ and $\nu_{\text{fiber}} = 0.19$.



(a) RVE vs SVE (b) Strain inputs (c) Stress outputs

Figure 5.2: 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.

As seen in Figure 5.2a, 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 microstucture (Figure 5.2a) lead to different average stress responses (dashed lines in Figure 5.2c), even when subjecting the SVEs to the same input deformation path (Figure 5.2b).[6] In contrast, randomization of the microstructure of an RVE still leads to the same homogenized stress response (see the solid lines in Figure 5.2c, embedded in the dashed lines corresponding to different realizations of the SVE).

Remark 5.2.1. *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 Figure 5.2c. However, experimental data that contains aleatoric uncertainty that can arise from different sources (e.g., data acquisition, intrinsic material heterogeneity, manufacturing variations).*

Without loss of generality, all learning problems in this chapter 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:

$$\boldsymbol{\sigma}_{:} = f(\boldsymbol{\varepsilon}_{:}, \boldsymbol{\theta}), \tag{5.1}$$

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 [34]). Additional details on the generation of input strain paths are found in Appendix D.1, and on the origins of aleatoric uncertainty in this context are in Figure 5.2 and Appendix D.1.

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 Figure 5.3. 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 datasets with different fidelities, as presented in Table 5.1. Importantly, we introduce the concept of cost ratio to properly evaluate the computational cost of generating a single

[6] Readers unfamiliar with homogenization theory are referred to [10, 34], 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.

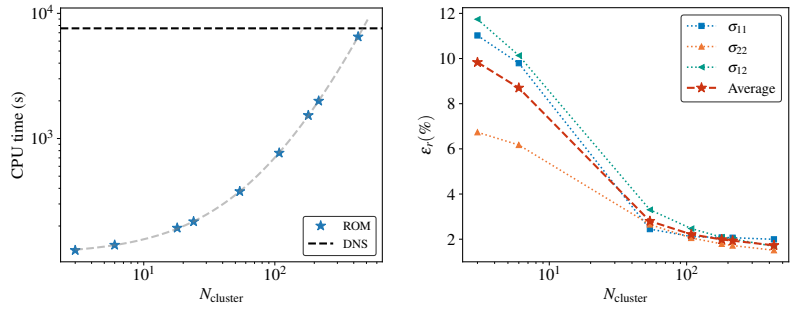


Figure 5.3: Performance comparison between FEA-based DNS and SCA in the computational homogenization of biphasic microstructure RVE. The accuracy and computational cost of the SCA method depend on the number of clusters.

5

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 $\left(\varepsilon_{ij,\min}, \varepsilon_{ij,\max}\right) = (-0.1, 0.1),\ \{i,j\} = 1,2$. The OOD testing paths were obtained considering a wider strain range of $\left(\varepsilon_{ij,\min}, \varepsilon_{ij,\max}\right) = (-0.125, 0.125),\ \{i,j\} = 1,2$. 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 5.1) – 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.

Table 5.1: 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 (alea. 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)

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 5.1:

- **Dataset 1 (stochastic single fidelity): $\widetilde{\text{DNS}}$** – The only single fidelity dataset considered in this chapter 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).

- **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.

5.3. 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 (Figure 5.4) that highlights the modular nature of the framework, hoping to facilitate navigation among possible choices. In the remainder of the chapter, 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).

5.3.1. NEURAL NETWORK ARCHITECTURE CHOICE

Figure 5.5 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 [148], 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,

$$f_j(\mathbf{x}_t) \sim \mathcal{N}\left(y_j(\mathbf{x}_t), s_j^2(\mathbf{x}_t)\right) \quad \text{where} \quad t = 1, ..., T, \tag{5.2}$$

where $y_j(\mathbf{x})$ is the ground truth mean of the output component j (note that $\mathbf{y} \equiv \boldsymbol{\sigma}$ and

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

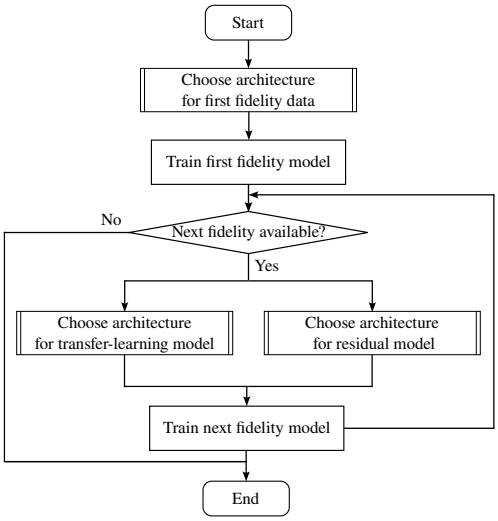


Figure 5.4: Flowchart of the proposed practical multi-fidelity framework.

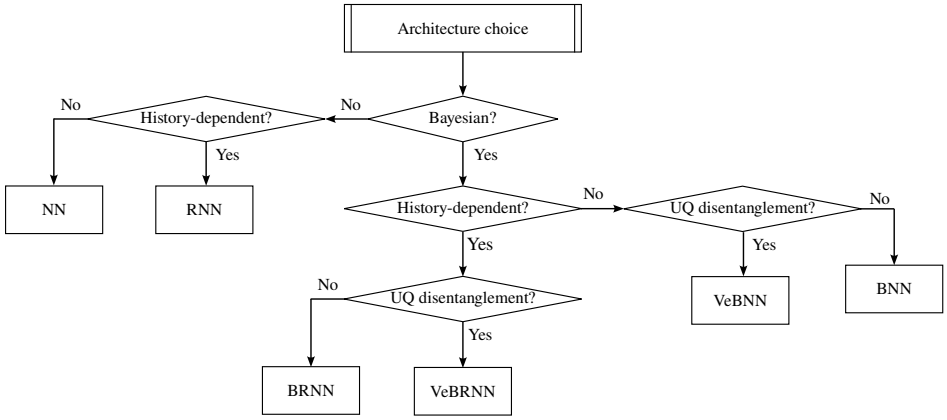


Figure 5.5: Flowchart for choosing a neural network architecture that learns from data of given fidelity. UQ is an abbreviation for uncertainty quantification.

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 $\boldsymbol{\varepsilon}$), it is referred as heteroscedastic aleatoric uncertainty; otherwise, it is homoscedastic. Figure 5.2 and Figure D.3 clearly show that SVEs have heteroscedastic aleatoric uncertainty (noise).

[8] From Section 5.2, $\boldsymbol{\sigma}$ has three components for two-dimensional problems under plane strain conditions: σ_{11}, σ_{12}, and σ_2.

Algorithm 8 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.2 describes the 3 steps, but a complete description is also provided in the original publication [148]. Section 5.4.1 presents results obtained from training a VeBRNN on Dataset 1 (obtained by DNS of an SVE).

Algorithm 8: Cooperative training of VeBRNN

Data: Mean RNN network $f(\mathbf{x};\boldsymbol{\theta})$, variance RNN network $s^2(\mathbf{x};\boldsymbol{\phi})$, training data $\mathscr{D} = \{\mathbf{X},\mathbf{y}\}$ 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 Equation (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 Equation (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 Equation (D.10) to update mean and epistemic variance estimates for fixed aleatoric variance from Step 2 using *pSGLD* sampler illustrated in Section 2.5.2.

Compute the log marginal likelihood of $\texttt{LMglk}[i]$

end

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

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 [148]. 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 [148].

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 (Section 2.5). This step allows to update the mean response prediction (from

[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.

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 [148]. 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.

5.3.2. MULTI-FIDELITY FRAMEWORK

In a recent chapter, we proposed a unification of the MF machine learning literature into a practical MF framework [147]. We postulated that MF machine learning can be expressed through the general formulation,

$$f^h(\mathbf{x}) = g(f^l(\mathbf{x}), \mathbf{x}) + r(\mathbf{x}), \tag{5.3}$$

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 $\mathscr{D}$ consist of HF inputs $\boldsymbol{X}_t^h \in \mathbb{R}^{N^h \times d^{in}}$ and outputs $\boldsymbol{Y}_t^h \in \mathbb{R}^{N^h \times d^{out}}$, as well as LF inputs $\boldsymbol{X}_t^l \in \mathbb{R}^{N^l \times d^{in}}$ and outputs $\boldsymbol{Y}_t^l \in \mathbb{R}^{N^l \times d^{out}}$. 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 Equation (5.3) to explicitly account for time-series data as

$$f^h(\mathbf{x}_t^h) = g(f^l(\mathbf{x}_t^h), \mathbf{x}_t^h) + r(\mathbf{x}_t^h); \quad t = 1, \cdots, T\,, \tag{5.4}$$

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]

Equation (5.4) 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 D.4 four solutions for comparison. For brevity, the main text only includes the best solution – illustrated in Figure 5.6.

[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 Figure 5.4 by recurrently adding new fidelities.

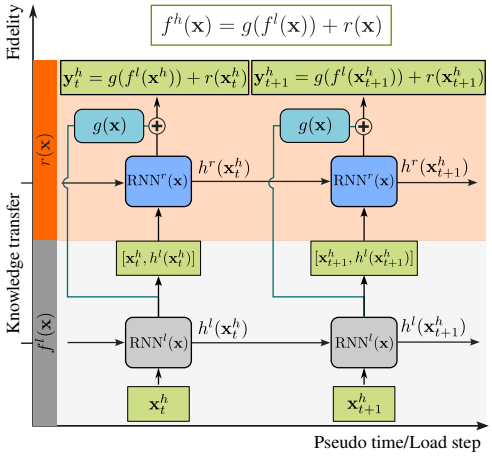


Figure 5.6: Schematic of the proposed multi-fidelity recurrent neural network architectures for predicting uncertainty-aware history-dependent plasticity.

The MF model in Figure 5.6 can be regarded as a particular realization of Figure 5.4. 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 Figure 5.6 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 Figure 5.6). This is reflected in Figure 5.6 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 D.4. Moreover, we also found that using a linear transfer learning model (i.e., $g(\mathbf{x})$ is simply an identity map in Figure 5.6) 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 D.4. We did not find any benefit in using expressive models for both transfer learning and residual learning. Nevertheless, we do not claim that Figure 5.6 represents the best MF model for all possible datasets. Appendix D.4 simply shows that it is the most effective solution for the four datasets that we consider in this chapter, and that we discuss in the next sections.

Remark 5.3.1. *For simplicity, this subsection and Figure 5.6 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 Figure 5.4 and Algorithm 8). 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.*

Table 5.2: Summary of architectural choices for single- and multi-fidelity datasets. See Figure 5.4 for understanding single- and multi-fidelity training with neural network architectures chosen according to Figure 5.5. Notation: "LF model"+"HF model"

Model	Comment	LF predictions			HF predictions		
		Mean	Alea.	Epis.	Mean	Alea.	Epis.
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	✓	✓	✓	✓	✓	✓

5.3.3. 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 5.2) illustrate different scenarios encountered in Engineering practice. Table 5.2 summarizes the different models, from simpler (top) to more complex (bottom) – confront with Figure 5.4 and Figure 5.5.

5.4. RESULTS

We analyze the proposed framework (Figure 5.6) on four datasets (Section 5.2), and compare the performance of different model architectures (Table 5.2) while considering the total computational cost of data generation as,

$$T_c = N^h_{\text{train}} c^h + N^l_{\text{train}} c^l \tag{5.5}$$

where N^h_{train} and N^l_{train} 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 5.1.

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 Table 5.3. A more detailed description of these metrics is provided in Appendix D.3. Note that the ground truth mean in the proposed dataset is obtained from $\overline{\text{DNS}}$ (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 D.6. Additional results are presented in Appendix D.5, and we will refer to this appendix when appropriate.

Table 5.3: 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	**Abbr.**	**Description**	**Trend**
Relative error ([35])	ϵ_r	Relative error between mean prediction and ground truth.	$\downarrow$
Test Log-Likelihood ([128])	TLL	Test log-likelihood of ground truth mean under the predicted epistemic distribution.	$\uparrow$
Wasserstein Distance ([162])	WA	Wasserstein distance between predicted aleatoric and ground truth distributions.	$\downarrow$
Prediction Interval Coverage Probability ([163])	PICP	Ratio of ground truth mean within the predicted epistemic interval under given confidence level.	$\rightarrow (1-\alpha)$
Mean Prediction Interval Width ([163])	MPIW	Mean width of predicted epistemic uncertainty interval.	$\downarrow$

5.4.1. RESULTS FOR DATASET 1: $\widetilde{\text{DNS}}$ (STOCHASTIC SINGLE FIDELITY PROBLEM FROM FEA OF AN SVE)

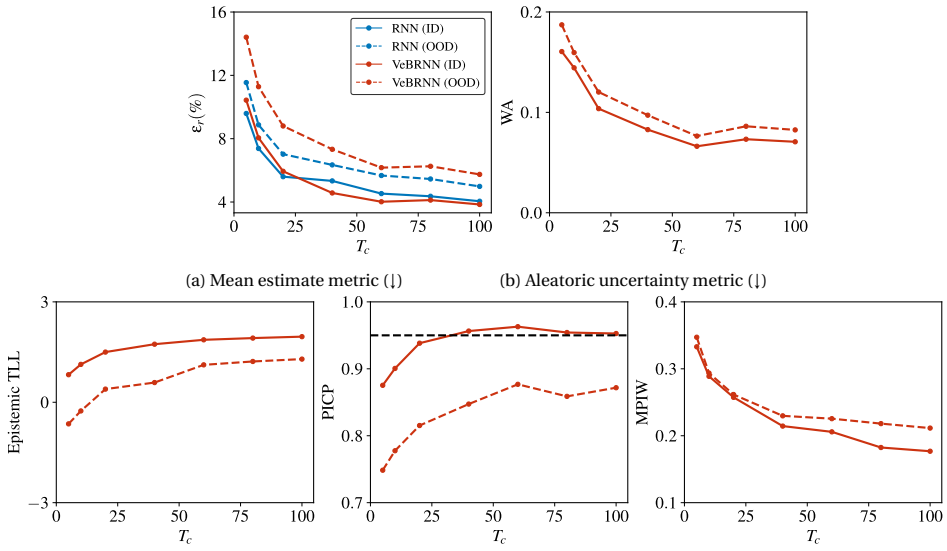
This first dataset includes aleatoric uncertainty and only considers one fidelity ($\widetilde{\text{DNS}}$). It is introduced to demonstrate the ability of the VeBRNN to predict mean, aleatoric uncertainty variance, and epistemic uncertainty variance. This contrasts with the widely used RNNs that only predict the mean response, as they are deterministic models. A summary of the setup of experiments is given by Table 5.4.

Table 5.4: Experimental setup for dataset 1

Data fidelities	Model	Evaluation metrics	Results
$\widetilde{\text{DNS}}$	RNN	ϵ_r	Figure 5.7
$\widetilde{\text{DNS}}$	VeBRNN	ϵ_r, TLL, WA, PICP, MPIW	Figure 5.7

As indicated in Table 5.1, 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 $T_c = N^h_{\text{train}} \times \frac{1}{20} + 0$. The corresponding results are shown in Figure 5.7. Note that we also have access to the noiseless data ($\overline{\text{DNS}}$) that we can use to precisely assess the error of this dataset for the relative error ϵ_r and Epistemic TLL.

Remark 5.4.1. *The single-fidelity case trivializes the flowchart in Figure 5.4, reducing to training a single RNN (or VeBRNN).*



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

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

Figure 5.7: 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).

As shown in Figure 5.7, 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 (Figure 5.7c) improves with more training paths, indicating that the VeBRNN reduces model uncertainty with more training data (as expected). The Wasserstein distance (Figure 5.7b) 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 Figure 5.7e (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,

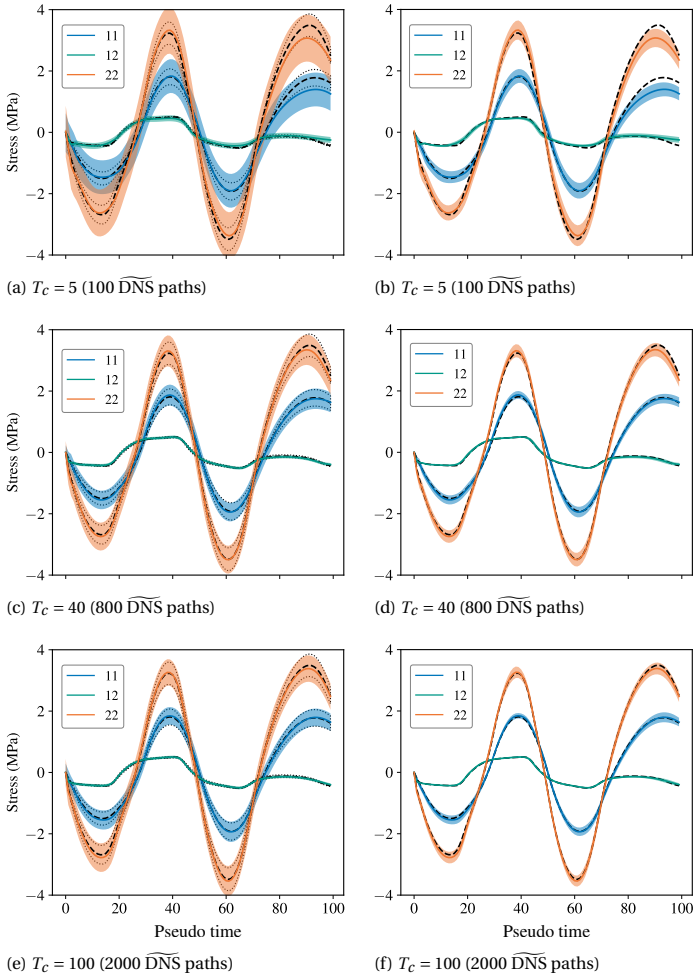


Figure 5.8: Predictions of VeBRNN on an ID test path after training with 2000 paths ($T_c = 100$) from Dataset 1 ($\widetilde{\text{DNS}}$). The left column shows the comparison between 0.95 confidence intervals of the predicted aleatoric uncertainty (colored shaded areas) and the ground truth distributions (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.

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 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.

Figure 5.8 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}. Figure 5.8f 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 Figure D.11, 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, Figure 5.8 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 Figure 5.7. 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 $T_c = 40$ (Figure 5.8c to $T_c = 100$ (Figure 5.8e) while the predicted epistemic uncertainties (Figure 5.8d and Figure 5.8f) decrease, as expected.

5.4.2. RESULTS FOR DATASET 2: $\overline{\text{ROM}}$+$\overline{\text{DNS}}$ (MF PROBLEM WITH DETERMINISTIC LF & HF DATA)

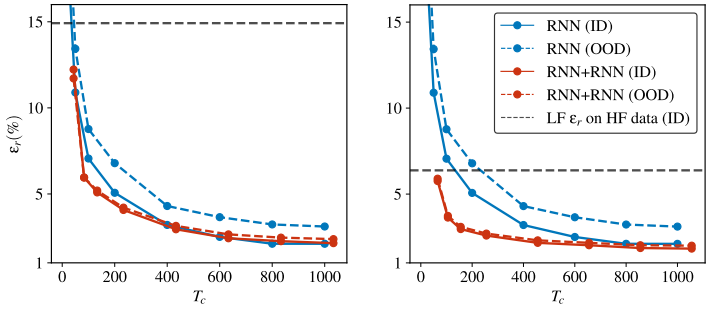
This MF dataset does not have aleatoric uncertainty (noise); therefore, we can consider RNNs for both fidelities. However, we can also consider VeBRNNs (which reduce to BRNNs because they find zero aleatoric uncertainty), in case we want to predict epistemic uncertainty. We include both cases in this section. Unless otherwise stated, all the available LF training paths indicated in Table 5.1 are used, and models are assessed for an increasing number of HF training paths. Detailed experiment setup is summarized in Table 5.5.

Table 5.5: Experimental setup for dataset 2

Data fidelities	Model	Evaluation metrics	Results
$\overline{\text{DNS}}$	RNN	ϵ_r	Figure 5.9
$\overline{\text{ROM}}$+$\overline{\text{DNS}}$	RNN+RNN	ϵ_r	Figure 5.9
$\overline{\text{DNS}}$	VeBRNN	ϵ_r, TLL, WA, PICP, MPIW	Figure 5.10
$\overline{\text{ROM}}$+$\overline{\text{DNS}}$	RNN+VeBRNN	ϵ_r, TLL, WA, PICP, MPIW	Figure 5.10

MF model with RNN+RNN architectures A MF model with fully deterministic architectures is trained using *Adam* [46]. Figure 5.9 presents the results, including a comparison

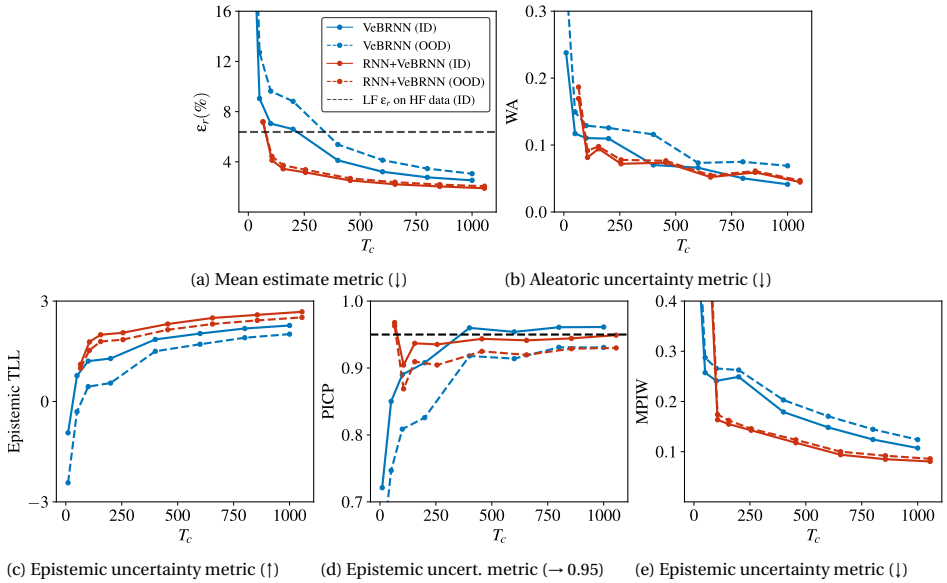
with a single-fidelity RNN model trained only on HF data ($\overline{\text{DNS}}$). Up to a certain computational cost, the MF model (RNN+RNN) is more accurate than the single-fidelity model, especially for OOD test paths. The MF model advantage is more pronounced when the LF model is better (using a higher number of clusters for the SCA method – Figure 5.9b). 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.



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

Figure 5.9: 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.

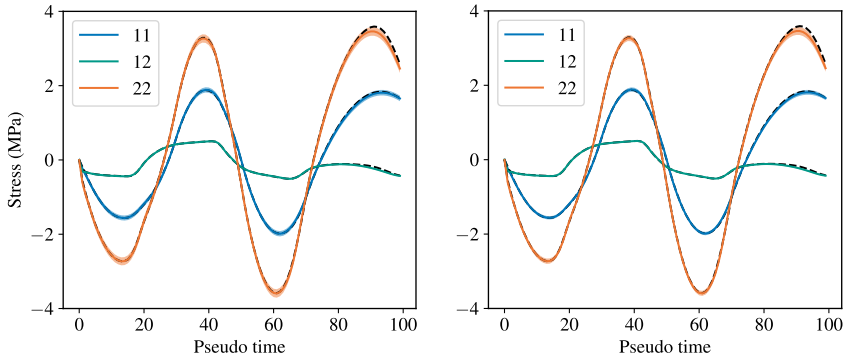
MF model with RNN+VeBRNN architectures Considering the VeBRNN (which will end up reducing to a BRNN) for the HF data allows to quantify epistemic uncertainty (and to estimate zero aleatoric uncertainty). Results are shown in Figure 5.10. Note that now we compare with a single-fidelity model with the 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 5.4.1, 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



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

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

Figure 5.10: 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) RNN+VeBRNN Aleatoric uncertainty (b) RNN+VeBRNN Epistemic uncertainty

Figure 5.11: 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.

finds that the dataset has negligible noise.[11]

Figure 5.11 presents the RNN+VeBRNN prediction for an ID test path for a model

[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.

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 Figure 5.11a. 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.

5.4.3. RESULTS FOR DATASET 3: $\widetilde{\text{ROM}}$+$\widetilde{\text{DNS}}$ (MF PROBLEM WITH STOCHASTIC LF & HF DATA)

Therefore, we have selected to use five different MF modeling strategies from Table 5.2, and the details are listed in Table 5.6.

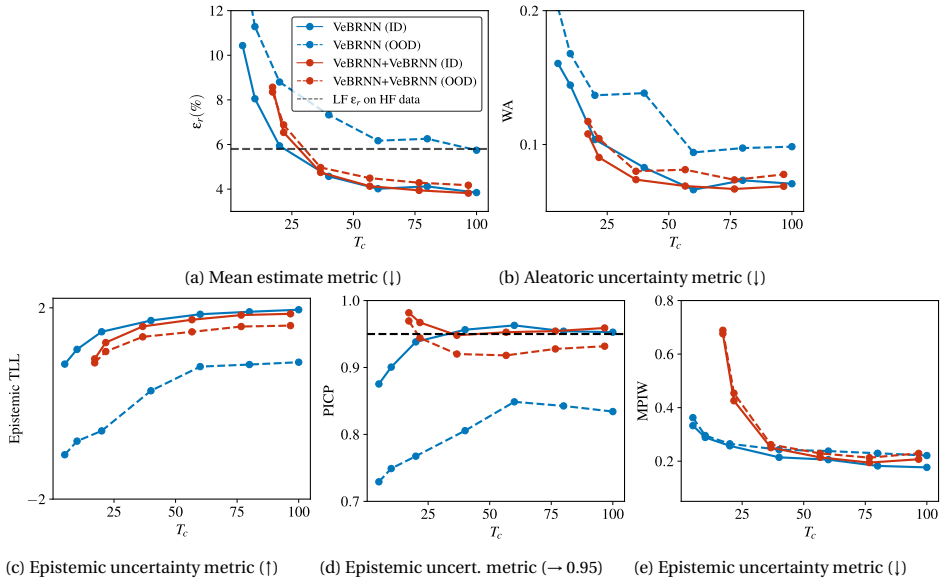
Table 5.6: Experimental setup for dataset 3

Data fidelities	Model	Evaluation metrics	Results
$\widetilde{\text{DNS}}$	VeBRNN	ϵ_r, TLL, WA, PICP, MPIW	Figure 5.12
$\widetilde{\text{ROM}}$+$\widetilde{\text{DNS}}$	VeBRNN+VeBRNN	ϵ_r, TLL, WA, PICP, MPIW	Figure 5.12
$\widetilde{\text{DNS}}$	RNN	ϵ_r	Figure D.12
$\widetilde{\text{ROM}}$+$\widetilde{\text{DNS}}$	RNN+RNN	ϵ_r	Figure D.12
$\widetilde{\text{ROM}}$+$\widetilde{\text{DNS}}$	RNN+VeBRNN	ϵ_r, TLL, WA, PICP, MPIW	Figure D.13

This is the most challenging dataset, as both fidelities are stochastic (noisy). Figure 5.12 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 D.5, 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 D.5 for the RNN+VeBRNN model results).

Figure 5.12 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 Figure 5.12.

Remark 5.4.2. *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.*



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

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

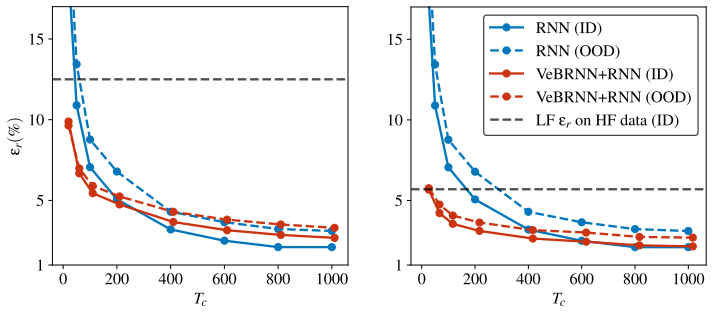
Figure 5.12: Results for the VeBRNN+VeBRNN model, when compared with a single-fidelity VeBRNN trained on HF data. The LF data is obtained by SCA of an SVE with $N_{\text{cluster}} = 18$. The number of HF training paths ($\widetilde{\text{DNS}}$) used in MF training is $[10, 100, 400, 800, 1200, 1600]$, 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.

5.4.4. RESULTS FOR DATASET 4: $\widetilde{\text{ROM}}$+$\overline{\text{DNS}}$ (MF PROBLEM WITH STOCHASTIC LF & DETERMINISTIC HF)

Figure 5.13 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, with details setup listed in Table 5.7. The results are similar to Figure 5.9, 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 5.2). With all the 2000 training paths used to train the LF model, the VeBRNN achieves $\epsilon_r = 3.87\%$, Epistemic TLL = 1.78, WA = 0.074, PICP = 0.949, and MPIW = 0.214. The predictions for a test path are shown in Figure 5.14, 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.

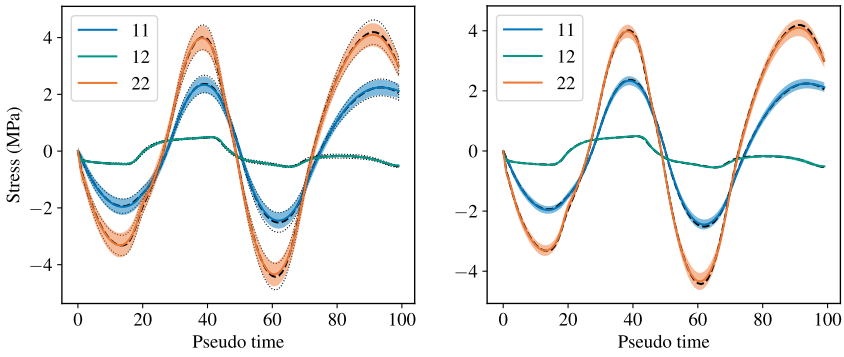
Table 5.7: Experimental setup for dataset 4

Data fidelities	Model	Evaluation metrics	Results
$\overline{\text{DNS}}$	RNN	ϵ_r	Figure 5.13
$\widetilde{\text{ROM}}$+$\overline{\text{DNS}}$	VeRNN+RNN	ϵ_r	Figure 5.13



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

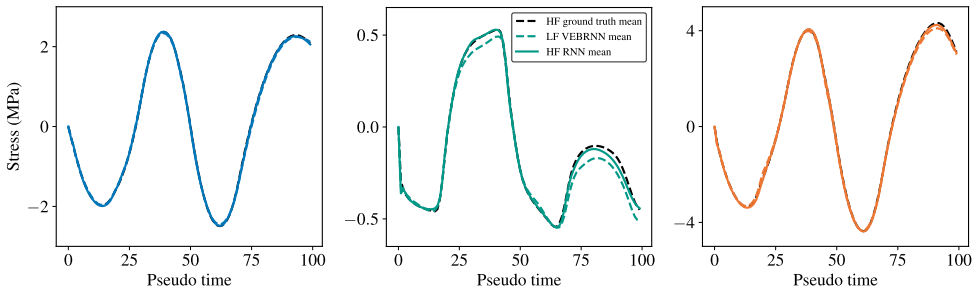
Figure 5.13: Results for VeBRNN+RNN model trained on Dataset 4 ($\widetilde{\text{ROM}}$+$\overline{\text{DNS}}$) compared to single-fidelity RNN trained on HF data. (a) LF model trained on data obtained by SCA of an SVE with $N_{\text{cluster}} = 3$; (b). LF model trained on data obtained by SCA of an SVE with $N_{\text{cluster}} = 18$.



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

Figure 5.14: 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

Figure 5.15 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 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.



(a) σ_{11} (b) σ_{12} (c) σ_{22}

Figure 5.15: 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.

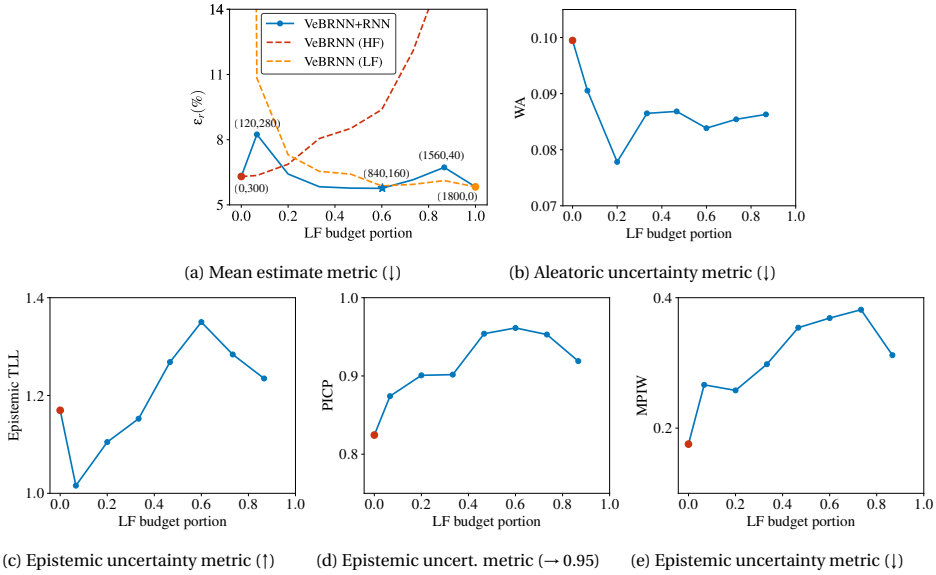
5

5.5. 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 ($\widetilde{\text{ROM}}$+$\widetilde{\text{DNS}}$), 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 5.4.3). However, Figure 5.16 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 acquisition of the HF and LF data only differs by a factor of 6 (see Table 5.1). 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.

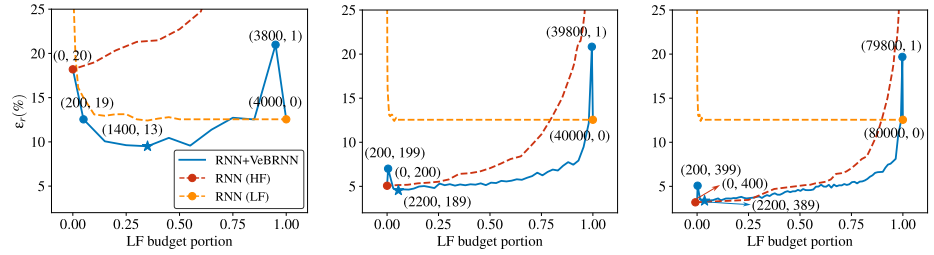
A similar observation arises from considering Dataset 4 ($\widetilde{\text{ROM}}$+$\overline{\text{DNS}}$). Section 5.4.4 showed that if the LF data is obtained by a reasonably accurate ROM (SCA method discretized by $N_{\text{cluster}} = 18$ clusters), then the MF model is better for a large range of total cost values. This is no longer the case if the ROM is less accurate (SCA discretized by $N_{\text{cluster}} = 3$ clusters). Figure 5.17 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 5.4.4). Unsurprisingly, if the total budget is high ($T_c = 200$ or $T_c = 400$), then there is already



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

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

Figure 5.16: Results for Dataset 3 ($\widetilde{\text{ROM}}$+$\widetilde{\text{DNS}}$ 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) $T_c = 20$ (b) $T_c = 200$ (c) $T_c = 400$

Figure 5.17: Results for Dataset 4 ($\widetilde{\text{ROM}}$+$\overline{\text{DNS}}$ with $N_{\text{cluster}} = 18$) 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 Figure 5.13b, obtained using 2000 LF training paths and a different number of HF training paths leading to the corresponding total cost in that figure.

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 (Figure 5.16 and Figure 5.17) 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.

5.6. CONCLUSIONS

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 (Figure 5.4 explains how to reduce a MF model with an arbitrary number of fidelities to a single-fidelity model; Figure 5.5 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 8. If aleatoric uncertainty does not need to be estimated, we can skip Step 2 of Algorithm 8. If epistemic uncertainty is also not needed, we can skip Step 3 of Algorithm 8. 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 chapter 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).

6

SCALABLE BAYESIAN MACHINE LEARNING FOR SUSTAINABLE COMPOSITE DESIGN: UNCERTAINTY-AWARE PREDICTION AND OPTIMIZATION OF RECYCLED POLYPROPYLENE/POLYETHYLENE BLENDS

This chapter presents a scalable Bayesian machine learning and uncertainty-aware inverse optimization framework to address the unpredictability of recycled polypropylene (PP)/polyethylene (PE) blends. First, a micro-scale finite element analysis model is developed to generate large-scale synthetic datasets. Second, a Variance estimation Bayesian Neural Network (VeBNN) is trained on this noisy data, achieving a relative error of less than 3.5% *while disentangling aleatoric and epistemic uncertainties, with noise estimates that closely follow empirical distributions and epistemic uncertainty informatively indicating the confidence of mean fitting. Finally, an uncertainty-aware inverse design optimization is performed, leading to PP/PE blends with reduced uncertainty and improved safe lower bounds compared to conventional data-driven approaches.*

This chapter is based on the manuscript: **Yi, J.**, Cózar, I. R., Caglar, B., & Bessa, M. A. (2025). *Bayesian learning for sustainable composite design: uncertainty-aware prediction and optimization of recycled polypropylene/polyethylene blends.* In preparation for submission.

6.1. INTRODUCTION

Plastic consumption has increased dramatically over the past few decades due to the low cost of polymers, their lightweight nature, and durability [164]. The prevalent *take-make-consume-throw away* model has led to serious environmental problems, with millions of tons of plastic waste accumulating in landfills every year [165, 166]. This has created a pressing need to recycle these materials by chemical [167–169] or mechanical [170–173] processes. Unfortunately, recycled plastics do not result in perfectly separated and homogeneous polymeric materials that would behave similarly to an equivalent virgin plastic. Instead, recycled plastics result in a blend of polymers with different grades that come from unknown origins, resulting in materials with very high aleatoric uncertainty. This poses enormous challenges in using them for engineering applications because their uncertainty makes them unreliable. For example, mechanical testing of samples from the same batch of a recycled material can lead to failure strains spanning orders of magnitude (from 12% to 600%) [171]. This chapter develops a computational data-driven methodology to analyze and design materials that exhibit large uncertainty, as occurs in sustainable plastics.

As a model system, we consider blends of polypropylene (PP) and polyethylene (PE). These two polymers account for almost two-thirds of the world's plastic consumption and form a large portion of the waste stream [174]. Unfortunately, they are difficult to economically separate from each other during recycling because they have similar density [175]. As such, traces of contamination from one plastic in another lead to degradation of mechanical performance due to immiscibility [176, 177], i.e., the two polymers have very low interfacial adhesion. The interface strength between these polymers can be improved by resorting to advanced compatibilizers that have been shown to create viable composite materials when combining virgin PP and PE [171, 172]. Few successful attempts have been made to design such a composite, and they have all been based on trial-and-error experimentation [171, 178]. Designing post-consumer recycled PP/PE blends would be even more challenging, as the chemical structure of the recycled materials that would need to be compatibilized is unknown, which would contribute to even larger uncertainty.

Finite Element Analysis (FEA) is widely employed to compute the material properties of composites, offering a cost-effective alternative to labor-intensive experiments [179, 180]. In particular, Representative Volume Element (RVE) and Statistical Volume Element (SVE) simulations [181] are employed to capture the heterogeneous response of composites, as multiple phases simultaneously govern the overall material behavior. Despite the advantages of FEA, most existing studies were concentrated on fiber-reinforced composite systems [182], where we summarized several state-of-the-art modeling strategies in Table 6.1. In contrast, polymer–polymer systems such as PP/PE blends present additional challenges, as their constituents exhibit viscoelastic–viscoplastic characteristics even at the single-phase level. Moreover, most available constitutive models for polymers primarily address plasticity [179, 183–185], whereas reliable prediction of failure remains a bottleneck for RVE/SVE simulations of polymer–polymer systems. In the processing of polymers, the microstructural morphology is strongly governed by phase weight fractions: dominant–minor phase combinations yield regular inclusions, whereas balanced fractions result in irregular, amorphous structures [176, 186, 187]. While this trend persists with compatibilizer addition, the interface becomes more refined due to enhanced

interfacial adhesion [171, 177, 188–190] since the compatibilizer reduces the interfacial tension. This motivates the development of an FEA-based modeling strategy for recycled PP/PE blends that incorporates microstructural morphology and enables simulation up to failure.

Data-driven design and analysis of materials leveraging machine learning (ML) techniques has become increasingly ubiquitous [10, 191, 192]. Within this paradigm, large datasets can be generated from RVE/SVE simulations under arbitrary boundary conditions and loading magnitudes, and ML models trained on these datasets can subsequently provide rapid and accurate predictions of effective properties [193], such as homogenized permeability [194], elastic moduli [195], plastic behavior [173], thermal response [196], and failure characteristics [197]. Beyond direct property prediction, ML has also been applied in multi-scale analysis [10], where micro-scale FEA simulations are coupled with ML-based surrogate models; in inverse design [39, 40] to tailor material performance; and in data augmentation [195, 198] to alleviate the scarcity of experimental data. However, the existing data-driven approaches for composite material property prediction rely on deterministic ML models to estimate the quantities of interest (QoIs) [191, 199]. In other words, they fail to account for uncertainty from both the data (uncertainty due to the recycling process and material impurities, usually referred to as *aleatoric* uncertainty) and the model (uncertainty arising from the ML model itself, also referred to as *epistemic* uncertainty) [18]. Lacking the capacity to predict both uncertainties may lead to serious issues when predicting and designing recycled composites. For instance, when optimizing a target property of recycled PP/PE blends, such as failure strain or toughness, a conventional inverse design approach that relies solely on the predicted mean may yield a material with high nominal failure strain but accompanied by large data noise. As a result, the designed material may exhibit unreliable performance, with significant variability across realizations, which ultimately undermines its practical applicability.

In the literature, Gaussian Process Regression (GPR) [153] has been widely employed for composite modeling due to its capability to quantify epistemic uncertainty [10, 199]. However, its scalability is restricted by the *curse of dimensionality*, which limits its applicability to high-dimensional and large-scale problems. In contrast, Bayesian Neural Networks (BNNs) work at scale and provide principled epistemic uncertainty quantification [45, 49]. They have been successfully applied to predict homogenized micromechanical properties of fiber-reinforced polymers [200] as well as full-field material distributions [201]. Although these approaches have improved performance, they are generally limited to problems with small data noise and often assume homoscedasticity. To the best of our knowledge, only limited efforts have been devoted to disentangling aleatoric and epistemic uncertainties. For example, several emerging studies attempt to quantify large data uncertainty in a computationally expensive manner, such as relying on multiple experimental or simulation realizations to probe predictive variability [39, 41, 42]. In this regard, we developed a cooperative training strategy that leads to the Variance estimation Bayesian Neural Network (VeBNN) method, which successfully disentangles heteroscedastic data noise from the ML model uncertainty. It has been successfully applied to model history-dependent plasticity, explicitly accounting for data uncertainty arising from microstructural variability [202, 203].

Summary of Contribution: In this paper, we propose a scalable Bayesian design

framework for recycled composite polymers, with a particular focus on designing recycled PP/PE blends under large heteroscedastic data uncertainty. The approach consists of three key steps: (1) data-acquisition, (2) training of a scalable Bayesian machine learning model that quantifies both aleatoric and epistemic uncertainties, and (3) uncertainty-aware inverse design to discover a PP/PE polymer blend with improved mechanical properties and reduced data uncertainty.

Data acquisition in this work involves the simulation of periodic material volume elements that predict efficiently and accurately the mechanical behavior of different PP/PE polymer blends. This first step of the data-driven process is crucial, as the quality and quantity of data largely determine the success of subsequent steps. Therefore, a novel FEA model was developed to simulate PP/PE blends up to failure. Unfortunately, these simulations involve large deformations and onset of failure prediction, making them computationally expensive. In order to generate sufficient data for the data-driven process, SVE simulations of recycled PP/PE blends needed to be considered instead of RVE simulations. SVEs are stochastic in nature, i.e., randomly generating a microstructure with the same geometric descriptors leads to a different homogeneous response (unlike what occurs with RVEs, that lead to the same response). The SVEs developed herein consider different weight fractions for each phase (PP and PE), and they also consider different compatibilizers, i.e., the material that improves adhesion between PP and PE. The SVEs have distinct morphologies that range from particle-like microstructures to amorphous ones, similarly to what is reported in the literature when considering different weight fractions for each phase. Leveraging this capability, we automatically generate a large dataset with thousands of training points [133, 204]. Then, our recently proposed scalable VeBNN method [202] is trained on the dataset such that it learns and predicts the homogeneous failure strain of the PP/PE composite, as well its toughness while quantifying both aleatoric and epistemic uncertainties. Results show the VeBNN exhibit low errors for the mean prediction (less than 3.5% relative error) and estimate epistemic uncertainty better than a GPR method, while also being able to estimate aleatoric uncertainty. The quality of the predictions is assessed by simulating new SVEs that are not included in the training data, and finding the empirical distribution of test points generated with Monte Carlo Simulation. Finally, the trained VeBNN is used to enable efficient and uncertainty-aware optimization of composite designs, leading to optimized PP/PE blends with better material properties while reducing the data noise (aleatoric uncertainty). The proposed Bayesian design framework opens a new avenue for sustainable material design and offers a robust and scalable pathway for other model systems.

The remainder of this chapter is organized as follows. The proposed micro-scale finite element analysis model for recycled PP/PE blends is presented in Section 6.2. Details of VeBNN as a forward setup and corresponding uncertainty-aware inverse design methodologies are summarized in Section 6.3. Section 6.4 and Section 6.5 show the key results and discussions for recycled PP/PE blends, their material properties prediction, and uncertainty-aware (or robust) optimization. Finally, key findings and conclusions are summarized in Section 6.6.

6.2. SVE DEVELOPMENT FOR RECYCLED PP/PE BLENDS

Table 6.1 highlights several models proposed in the literature to predict the mechanical behavior of polymers, including viscoelastic and viscoplastic models, as well as cohesive zone models (CZM) for modeling interface damage in composites. However, the authors were not able to find a validated model for recycled PP/PE blends with or without compatibilizers. Admittedly, this system is particularly challenging to simulate due to its highly heterogeneous and processing-dependent morphology, large deformations that it can sustain, and complex interfacial mechanisms controlling debonding, cavitation, and fracture. Furthermore, the work presented herein not only relies on the ability to model these materials reasonably accurately, but also on the ability to do it sufficiently fast to generate a large database from where its homogeneous mechanical properties can be learned in spite of their large uncertainty. This justifies the important simplifications that were introduced when developing and analyzing fast SVE simulations to capture the homogenized response and onset of failure, as this results from large deformations, plasticity and localized damage. Despite the simplicity of the FEA developed herein, the predictions are in reasonable agreement with the experimental data provided by LyondellBasell Industries Holdings, who conducted the tensile tests.

Table 6.1: Features of the constitutive models previously developed to predict the mechanical behavior of polymers, CZM refers to the use a cohesive zone model at the interface in composites.

Polymer model	Material system	Reference
Elasto-viscoplastic damage	PP	[205–207]
Viscoelasto-viscoplastic damage	PP or PE	[208, 209]
Viscoelasto damage	Other polymers	[210]
Elastoplastic damage	Other polymers	[211]
Elastoplastic with CZM	Fiber-reinforced PP	[212]
Elastoplastic with CZM	Fiber-reinforced polymer	[213–215]
Elasto-viscoplastic with CZM	Fiber-reinforced polymer	[216]
Elastoplastic damage with CZM	Fiber-reinforced polymer	[217–220]

6.2.1. MICROSTRUCTURE

The immiscibility of PP and PE is well documented in the literature: the minor component forms an amorphous dispersed phase, while the major component acts as a continuous matrix. One of the main factors affecting the microstructure of PP/PE blends is the blend composition. As the fraction of the minor phase increases, the structure becomes increasingly irregular [176, 188]. Jose et. al [221] analyzed the morphology of PP/PE blends and identified three distinct regions as a function of the weight fraction of PE (w_{PE}). For $w_{PE} \leq 30\%$, PE appears as dispersed particles in a PP matrix. For $w_{PE} \geq 70\%$, PP becomes the dispersed phase. In the intermediate range ($w_{PE} \in [30,70]\%$), the morphology is co-continuous. Based on these observations, we defined the microstructure of the FE models using the random generator method proposed in [222] to create co-continuous morphologies for $w_{PE} = (30,70)\%$. For the other ranges of w_{PE}, particle-based morphologies are generated, where the particles are modeled as circular shapes using the proposed method in [223].

The particle diameter in PP/PE blends without compatibilizer was fitted to the morphological data from [221], yielding

$$\text{Diameter without compatibilizer} = \begin{cases} 0.094\, w_{\text{PE}} - 0.331 & \text{if } w_{\text{PE}} \leq 30\% \\ -0.183\, w_{\text{PE}} + 17.544 & \text{if } w_{\text{PE}} \geq 70\%, \end{cases} \tag{6.1}$$

where the units are μm. The particle diameter of compatibilized blends was obtained by scaling the above relations with experimental data from [171, 172], leading to

$$\text{Diameter with compatibilizer} = \begin{cases} 0.044\, w_{\text{PE}} - 0.156 & \text{if } w_{\text{PE}} \leq 30\% \\ -0.087\, w_{\text{PE}} + 8.296 & \text{if } w_{\text{PE}} \geq 70\%, \end{cases} \tag{6.2}$$

with units of μm.

Another key factor affecting the microstructure is the use of compatibilizers, which localize at the PP/PE interface and form a diffuse phase that encapsulates the minor phase [172, 177, 224–226]. Increasing the compatibilizer content also leads to more irregular structures [177, 188, 189]. We modeled the compatibilizer component as a third phase encapsulating the minor, see Figure 6.1b.

6

6.2.2. GENERAL MODELING STRATEGY

A two-dimensional FEA micro-scale model based on finite deformation theory is employed to capture the large strains observed at the constituent level. Both recycled PP and PE are described using an elasto-plastic framework, with a progressive damage model applied only to PP to reproduce its softening behavior. In contrast, no damage model is assigned to PE, since the supplier reports no failure below 400% axial strain. The compatibilizer is modeled using an isotropic elasticity formulation.

Given the limited experimental information available for the recycled PE and PP polymers selected, and the computational efficiency requirements, an isotropic elasticity law combined with a von Mises plasticity model is employed for both polymer phases (PP and PE). PP failure is represented through an isotropic progressive damage model implemented in Abaqus [227] and previously applied to PP [208]. Damage initiation occurs when $\omega_D = 1$, where ω_D is defined as

$$\omega_D = \int \frac{d\bar{\varepsilon}^{pl}}{\bar{\varepsilon}^{pl}_D(\eta)} dt \tag{6.3}$$

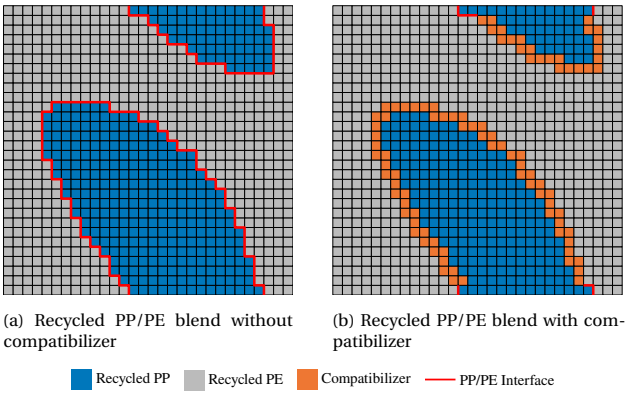
where $\bar{\varepsilon}^{pl}$ is the equivalent plastic strain and ε^{pl}_D is a material parameter defining the onset of damage as a function of stress triaxiality (η) (under uniaxial tension, $\eta = 1/3$). Damage evolution in PP is described using the Crack Band model [228], which ensures objectivity by regularizing the fracture energy with respect to the characteristic element size. Furthermore, a linear softening law is adopted [208].

Although hydrostatic pressure strongly influences polymer behavior [229, 230], the chosen yield criterion does not incorporate this effect. Nevertheless, the damage formulation partially accounts for hydrostatic pressure dependence. The constitutive models

adopted herein for PP and PE can be calibrated with a single uniaxial tensile test for each constituent (separately). Those were the only experiments that were provided for this work for the separate phases. If additional experimental data was available, models accounting for hydrostatic dependency would be considered.

The PP/PE blends develop a distinct PE crystal morphology; the PE lamellae align parallel to the PE/PP interface without crystal interpenetration [231], which leads to a weak PP/PE interface [171, 176, 231–233]. As a result, failure in pure PP/PE blends is dominated by interfacial debonding due to poor adhesion. This mechanism is represented in the model using a cohesive zone model at the PP/PE interfaces, see Figure 6.1a.

In PP/PE blends containing compatibilizers, the failure mechanism is mainly controlled by the molecular weight of the compatibilizer [171, 172]. The molecular weight of polymers is typically correlated with their melt flow rate. For the materials analyzed, the melt flow rate of the compatibilizers ranges from 0 g/10 min to 0.87 g/10 min [234], as measured following ISO 1133 [235]. Such low melt flow rates correspond to high molecular weights [236–239], which promote strong interfacial adhesion between the PP and PE phases. Under these conditions, failure occurs within the PP or PE phases rather than occurring at the compatibilized interfaces [171, 172]. Consequently, the compatibilizer interfaces with PP and PE are assumed to be perfectly bonded. Moreover, the addition of compatibilizers can also lead to imperfect bonding at certain PP/PE interfaces [240]. For this reason, the PP/PE interfaces are still modeled using the cohesive zone formulation described above when the compatibilizer is not present (e.g., for low weight fractions of the compatibilizer), see Figure 6.1b.



(a) Recycled PP/PE blend without compatibilizer

(b) Recycled PP/PE blend with compatibilizer

Figure 6.1: Schematic representation of the finite element modeling strategy for PP/PE blends.

Abaqus Explicit/FE solver is used to overcome convergence challenges, following the approach proposed in [212, 215, 220, 241, 242]. To accelerate the simulations, a balance between mass scaling and the ratio of internal to kinetic energy is maintained, ensuring that the kinetic energy remains below 10% of the internal energy to avoid spurious dynamic effects [215]. Reduced integration elements are adopted to improve computational efficiency while preserving solution accuracy.

Convergence analyses were conducted with respect to microstructure size, mesh resolution, and microstructural configuration. The detailed results are provided in Ap-

pendix E.2. Based on these studies, an SVE with a size of 25μm was adopted and discretized using a 100 × 100 structured mesh. This choice balances the need for accurately resolving microstructural features, particularly the narrow and spatially variable compatibilizer regions, variation in the plastic regime, and maintaining reasonable computational cost. Furthermore, the mesh element size is set sufficiently small to prevent the snap-back effect in the damage propagation region [243, 244]. Finally, periodic boundary conditions are applied using the relative formulation [242, 245].

6.2.3. MATERIAL CHARACTERIZATION

The elastic material properties of each phase are summarized in Table 6.2. The Young's modulus for PP and PE were obtained from experimental data provided by the material supplier, following the ISO 527 standard [246]. The Poisson's ratio for both PP and PE was defined based on measured values from the literature, as well as comparable values used in numerical studies [247–249]. The Poisson's ratio of each compatibilizer was determined according to its affinity for PP or PE (e.g., Compatibilizers 1 and 2 shown in Figure E.2 share the same Poisson's ratio as PP).

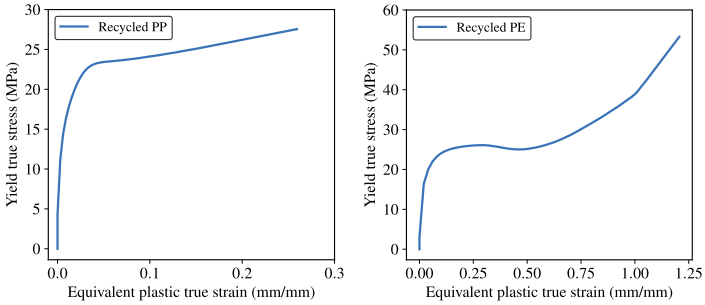
Table 6.2: Material properties used for recycled PP/PE blend simulation. The values adopted from literature have corresponding references; other material parameters are mainly from experimental data.

Category	Property	Recycled PP	Recycled PE	Interface	Compatibilizer
Elastic	Young's modulus (MPa)	797.97	664.03	–	5 ~ 200 [234]
	Poisson's ratio (-)	0.42 [250, 251]	0.45 [252]	–	0.42
Plastic	Hardening law	Figure 6.2a	Figure 6.2b	–	–
Damage	ε_p^D (mm/mm)	0.33	–	–	–
	Fracture toughness (N/mm)	0.22	–	0.0286 [231]	–
CZM	Penalty stiffness (N/mm^3)	–	–	10^6 [253]	–
	Strength (MPa)	–	–	18.00	–

The hardening master curves for PP and PE were estimated using experimental data from tensile tests. A constant elastic modulus was assumed, and the elastic component of the total strain was subtracted. These hardening master curves are presented in Figure 6.2. Similarly, the damage-related strain parameter for PP, denoted as (ε_D^{pl}), was defined under the assumption of a constant elastic modulus until the onset of damage. Upon damage initiation, the failure propagation in PP exhibited a sharp vertical decrease, indicating catastrophic failure.

The cohesive zone model parameters include penalty stiffness, strength, and fracture toughness. The penalty stiffness is a numerical parameter that ensures a stiff connection between PP and PE phases before the onset of interfacial failure. It was defined to be sufficiently large to maintain reasonable stiffness and numerical stability, thus preventing spurious traction oscillations [253]. The fracture toughness was defined based on the experimental value measured in [231], in which similar PP and PE materials were used, having comparable melt flow rate and molecular weight—key parameters that govern the interfacial failure mechanism in PP/PE blends.

Commonly in practical settings, there are properties that are difficult to characterize. In this case, the Young's modulus of each compatibilizer and the interfacial strength



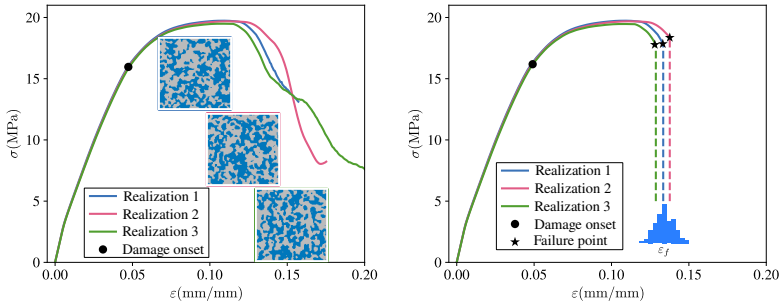
(a) Recycled PP (b) Recycled PE

Figure 6.2: Hardening master curves of PP and PE.

of the PP/PE blend without compatibilizer were not available. The interface strength between PP and PE (without compatibilizer) was estimated by a least squares fit of the FE simulations to experimental data obtained from tensile tests of a PP/PE blend (without compatibilizer). Furthermore, noting that existing compatibilizers are significantly more compliant than the respective constituents to be compatibilized (i.e., the PP and PE phases), the Young's modulus for each compatibilizer was then adjusted to match the experimental data of blends containing the corresponding compatibilizer.

6.2.4. HOMOGENIZATION AND SOFTENING TOLERANCE CRITERION

We present three simulations without compatibilizers and with $\nu_{PP} = 0.50$ and with different microstructure realizations in Figure 6.3a. The homogenized stress-strain curves are obtained by computing the volume-averaged micro-scale stress and strain over all elements at each load increment. The microstructure of each realization is also provided within the corresponding color box. They exhibit similar behavior before damage starts (indicated by the circle markers in Figure 6.3a), but variations appear and magnify with damage progression. Similar softening behavior is observed experimentally.



(a) Homogenized stress and strain curves (b) Postprocess results and aleatoric uncertainty of failure strain

Figure 6.3: Simulation results of SVEs without compatibilizer and corresponding post-processed failure strains.

However, note that we simplify failure prediction by avoiding the complete damage propagation observed in Figure 6.3a and instead considering a simplified criterion where the failure strain is postprocessed from the SVE simulation according to a stress softening tolerance threshold. Specifically, we terminate the simulation when the homogenized stress is less than 80% of the yield (maximum) stress. After applying the post-processing procedure, the simulation results in Figure 6.3a are updated as shown in Figure 6.3b, where the stars denote the extracted failure strain of the SVE. A clear variation is observed in the failure strain due to different microstructure realizations, which successfully mimics the experimental data variation.

6.3. METHODOLOGY

6.3.1. FORWARD PROBLEM: STRUCTURE-PROPERTY PREDICTION WITH UNCERTAINTY QUANTIFICATION AND DISENTANGLEMENT

The objective of the forward problem is to learn the input–output relationship in a data-driven manner. We assume that the data noise follows a Gaussian distribution. Accordingly, the learning problem can be formulated as:

$$f_i(\mathbf{x}) \sim \mathcal{N}\left(y_i(\mathbf{x}), s_i^2(\mathbf{x})\right) \tag{6.4}$$

where $\mathbf{x}$ denotes the design variables, $y_i(\mathbf{x})$ represents the ground truth mean of the i-th output, and $s_i^2(\mathbf{x})$ is the corresponding input dependent data noise variance.

Remark 6.3.1. *In our formulation Equation (6.4), the outputs are treated as random variables while the inputs are kept deterministic. This represents a realistic case where variability comes from repeating experiments under fixed conditions (inputs), so aleatoric uncertainty is attributed to the outputs. In contrast, another common approach in the literature assumes that aleatoric uncertainty originates from the design variables, where their variability propagates through a deterministic forward model, resulting in uncertainty in the outputs [18].*

Conventional machine learning approaches can only construct surrogate models to approximate the mean response $y(\mathbf{x})$, while failing to capture the inherent data noise. To characterize the material behavior of recycled PP/PE blends under uncertainty, we employ our previously developed Variance estimation Bayesian neural network (VeBNN), which is capable of capturing both aleatoric and epistemic uncertainties. A brief overview of the VeBNN framework is provided in Section 6.3.1, and further methodological details are available in [202].

VARIANCE ESTIMATION BAYESIAN NEURAL NETWORK

The VeBNN algorithm is easy to implement and can be adapted to different neural architectures, which employ a cooperative training strategy with three steps: (1) warm-start training of a mean network; (2) variance estimation training with a fixed mean of a variance network; (3) Bayesian training of the mean network with the variance fixed. This algorithm procedure is summarized in Algorithm 9, and its architecture and predictions for material properties are visualized in Figure 6.4.

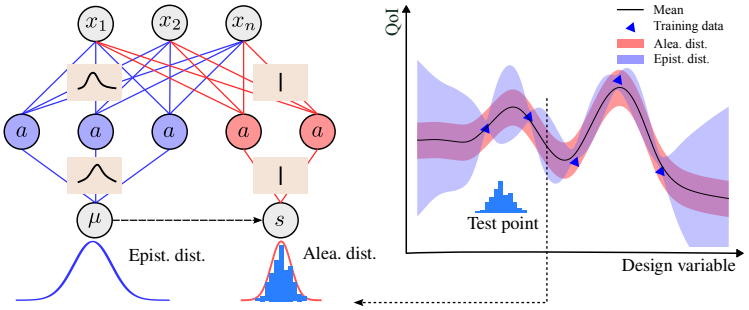


Figure 6.4: Schematic of Variance estimation Bayesian neural network (VeBNN) for material properties prediction.

Algorithm 9: Cooperative training of VeBNN

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

Result: Predictive distribution:

$$\mathcal{N}\left(\mathbf{y}\,\middle|\,\underbrace{\mathbb{E}_{p(\boldsymbol{\theta}|\mathscr{D})}\left[\mu(\mathbf{x};\boldsymbol{\theta}^*)\right]}_{\text{Predictive Mean}},\underbrace{s_a^2(\mathbf{x};\boldsymbol{\phi}^*)}_{\text{Aleatoric}}+\underbrace{\mathbb{V}_{p(\boldsymbol{\theta}|\mathscr{D})}\left[\mu(\mathbf{x};\boldsymbol{\theta}^*)\right]}_{\text{Epistemic}}\right) \tag{6.5}$$

Step 1: Mean network training:
Train the mean network by minimizing the MSE loss.

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

Step 2: Variance network training (Aleatoric uncertainty):
Train the variance network by minimizing the Gamma likelihood NLL:

$$\mathscr{L}_2 = \sum_{n=1}^{N}\left[\log\Gamma(\alpha_n) - \alpha_n\log\lambda_n - (\alpha_n - 1)\log r_n + \lambda_n r_n\right] \tag{6.6}$$

where $r_n = (y_n - \mu(\mathbf{x}_n;\boldsymbol{\theta}))^2$, $\alpha_n = \alpha(\mathbf{x}_n;\boldsymbol{\phi})$, $\lambda_n = \lambda(\mathbf{x}_n;\boldsymbol{\phi})$.

Step 3: Bayesian network training (Epistemic uncertainty):
Fix $s_a^2(\mathbf{x}_n) = \frac{\alpha_n}{\lambda_n}$ and sample from the posterior using e.g., pSGLD:

$$\log p(\boldsymbol{\theta}\mid\mathscr{D}) = \sum_{n=1}^{N}\left[-\frac{1}{2}\log(2\pi s_a^2(\mathbf{x}_n)) - \frac{(y_n - \mu(\mathbf{x}_n;\boldsymbol{\theta}))^2}{2s_a^2(\mathbf{x}_n)}\right] - \frac{\kappa}{2}\|\boldsymbol{\theta}\|^2 + \text{const} \tag{6.7}$$

And use posterior samples to compute log marginal likelihood `LMglk[`i`]`, as well as update the mean and provide epistemic uncertainty estimation.

end

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

We demonstrate the procedure using a single quantity of interest[1], as illustrated in Figure 6.4. The VeBNN framework employs two neural networks: a *mean network* (light red), parameterized by $\boldsymbol{\theta}$, and a *variance network* (orange), parameterized by $\boldsymbol{\phi}$. The training begins with a deterministic warm-up step (**Step 1** in Algorithm 9), where the mean network is trained by minimizing the Mean Squared Error (MSE). In **Step 2**, the variance network is trained to estimate the data variance, using either the fixed mean from **Step 1** or an updated mean from **Step 3**. Notably, we have shown that the squared residual between the mean prediction and observation follows a Gamma distribution, assuming Gaussian observation noise [202]. Therefore, the Gamma likelihood loss function (Equation (6.6)) is used to train the variance network. In **Step 3**, Bayesian inference is applied to the mean network with fixed variance, enabling posterior sampling and estimation of epistemic uncertainty. We adopt preconditioned Stochastic Gradient Langevin Dynamics (pSGLD) [52] for posterior inference, which directly samples from the posterior in Equation (6.7). If desired, **Step 2** and **Step 3** can be iterated to improve convergence. The final predictive expression of VeBNN is provided in Equation (6.5), where $\mu(\mathbf{x};\boldsymbol{\theta})$ denotes the predicted mean, serving as an approximation to the ground truth mean $y(\mathbf{x})$. The associated epistemic uncertainty, represented by $\mathbb{V}_{p(\boldsymbol{\theta}|\mathscr{D})}\left[\mu(\mathbf{x};\boldsymbol{\theta})\right]$, reflects the model's confidence in the mean prediction. In parallel, the predicted aleatoric variance $s_a^2(\mathbf{x};\boldsymbol{\phi})$ aims to approximate the inherent data noise $s^2(\mathbf{x})$ as defined in Equation (6.4).

The right panel of Figure 6.4 schematically depicts the model's behavior using five training data points. Once trained, VeBNN provides the *mean prediction* (black line), *aleatoric uncertainty* (orange shaded region), and *epistemic uncertainty* (light red shaded area). The epistemic uncertainty quantifies the model's confidence: it narrows near the training points and expands in regions lacking data. In contrast, aleatoric uncertainty captures intrinsic data noise. At a test input (blue vertical line), the true response distribution—represented by a blue histogram—can be empirically obtained by repeating the simulation or experiment (see Figure 6.5). The goal of aleatoric uncertainty estimation is to learn a parametric distribution (in this work, a Gaussian) that explains this data variability.

Remark 6.3.2. *Each input* $\mathbf{x}$ *appears only once with a random realization of microstructure in the training set. Aleatoric uncertainty is thus learned via the variance estimation network, rather than from repeated observations. This should lead to more efficient learning with less training data.*

6.3.2. INVERSE PROBLEM: DESIGN BETTER SUSTAINABLE PP/PE BLENDS CONSIDERING UNCERTAINTY

When the VeBNN is trained, we can consider the inverse problem: optimizing the mechanical performance of the recycled PP/PE blend while explicitly accounting for predictive uncertainty. The optimization problem can be generally formulated as follows:

$$\mathbf{x}^{\text{best}} = \arg\max_{\mathbf{x}\in\mathscr{X}} \left\{ \mathbb{E}_{\boldsymbol{\theta}\sim p(\boldsymbol{\theta}|\mathscr{D})}\left[\mu(\mathbf{x};\boldsymbol{\theta})\right] - 1.96 \cdot \left(s_a(\mathbf{x};\boldsymbol{\phi}) + \sqrt{\mathbb{V}_{\boldsymbol{\theta}\sim p(\boldsymbol{\theta}|\mathscr{D})}\left[\mu(\mathbf{x};\boldsymbol{\theta})\right]} \right) \right\} \tag{6.8}$$

[1] Only one learning objective is shown for clarity; the method readily extends to multi-output prediction by increasing the number of output neurons.

where $\mathscr{X}$ is the design space listed in Table 6.3, $\mathbb{E}_{\boldsymbol{\theta}\sim p(\boldsymbol{\theta}|\mathscr{D})}\left[\mu(\mathbf{x};\boldsymbol{\theta})\right]$ denotes the predicted mean of the objective, $s_a(\mathbf{x};\boldsymbol{\phi})$ and $\sqrt{\mathbb{V}_{\boldsymbol{\theta}\sim p(\boldsymbol{\theta}|\mathscr{D})}\left[\mu(\mathbf{x};\boldsymbol{\theta})\right]}$ represent the standard deviations of the predicted aleatoric and epistemic uncertainties, respectively (–see Algorithm 9). The coefficient 1.96 corresponds to the 95% confidence interval, ensuring that the optimized design strikes a balance between high performance and controlled uncertainty.

Remark 6.3.3. *Equation (6.8) translates the optimization of the surrogate model, that can also be referred to as offline optimization: it yields an optimal design scheme characterized by a large mean value and a small total uncertainty, resulting in the most conservative solution within the surrogate model (the regression model mapping inputs to output properties of interest). By contrast, in an online setting such as Bayesian optimization, each iteration would propose a new design that would be simulated by FEA, subsequently updating the regression model and leading to the next iteration, successively. In Bayesian optimization it is natural to target designs with large epistemic uncertainty and small aleatoric uncertainty, thereby achieving a better trade-off between exploitation and exploration. This is an interesting future research direction.*

Note that if the objective of the inverse problem was to find the design with the best mean response, then Equation (6.8) would be reduced to a simpler and familiar form:

$$\mathbf{x}^{\text{best}} = \arg\max_{\mathbf{x}\in\mathscr{X}} \mathbb{E}_{\boldsymbol{\theta}}\left[\mu(\mathbf{x};\boldsymbol{\theta})\right] \tag{6.9}$$

where $\mathbb{E}_{\boldsymbol{\theta}}\left[\mu(\mathbf{x};\boldsymbol{\theta})\right]$ is predicted by the VeBNN model. Alternatively, the optimization problem could be formulated only considering epistemic uncertainty:

$$\mathbf{x}^{\text{best}} = \arg\max_{\mathbf{x}\in\mathscr{X}} \mathbb{E}_{\boldsymbol{\theta}}\left[\mu(\mathbf{x};\boldsymbol{\theta})\right] - 1.96\sqrt{\mathbb{V}_{\boldsymbol{\theta}}\left[\mu(\mathbf{x};\boldsymbol{\theta})\right]}, \tag{6.10}$$

which would lead to the upper confidence bound (UCB) acquisition function commonly used in Bayesian optimization [254]. When only the aleatoric uncertainty is considered together with the mean prediction, the optimization objective is written as:

$$\mathbf{x}^{\text{best}} = \arg\max_{\mathbf{x}\in\mathscr{X}} \mathbb{E}_{\boldsymbol{\theta}}\left[\mu(\mathbf{x};\boldsymbol{\theta})\right] - 1.96\, s_a(\mathbf{x};\boldsymbol{\phi}), \tag{6.11}$$

The advantage of using a Bayesian surrogate model in robust optimization stems from the fact that the response can be denoised, and the optimization process does not require new design evaluations (iterations occur in the surrogate response surface). Equation (6.8) to Equation (6.11) correspond to different inverse design problems with a single objective. Multi-objective optimization formulations would also be possible, but are not considered herein.

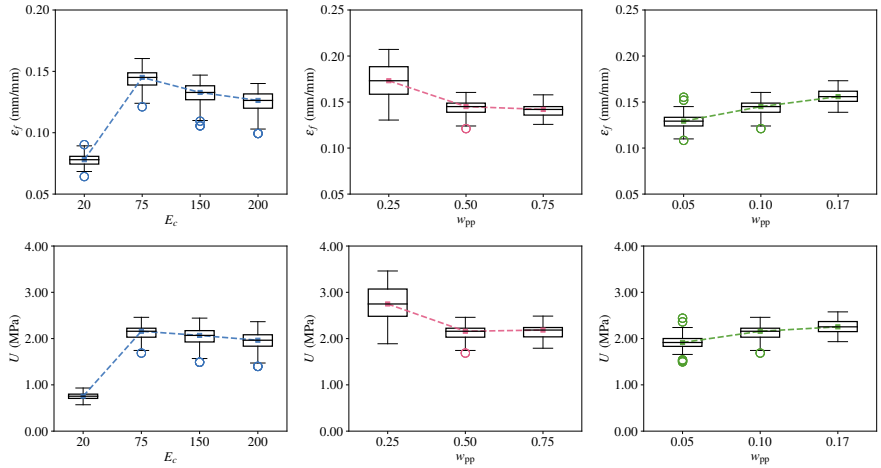
6.4. RESULTS

6.4.1. DATA-ACQUISITION

According to the literature and our investigation of the proposed FEA model, the mechanical performance of recycled PP/PE blends is primarily governed by two key factors: the type of compatibilizer and the weight fractions of each phase. To capture the effect

Table 6.3: Design variables and design space for recycled PP/PE blends. Here, $w_{PP} + w_{pe} = 1$, and the weight fractions of all phases are recalculated by accounting for the compatibilizer when generating the microstructure.

Design variable	Design space
Weight fraction of recycled PP : w_{PP}	$[0.2, 0.9]$
Weight fraction of compatibilizer : w_c	$[0.02, 0.2]$
Young's modulus of the compatibilizer phase : E_c	$[5, 200]$



(a). $w_{PP} = 0.5$ and $w_c = 0.1$ (b). $E_c = 75$ and $w_c = 0.10$ (c.) $E_c = 75$ and $w_{PP} = 0.50$

Figure 6.5: Visualization of three slices of test points, where each test point is obtained by 300 random microstructure realizations.

of different compatibilizers, we vary the Young's modulus of the compatibilizer phase, denoted as E_c. The design space of each design variable is summarized in Table 6.3.

Our analysis focuses on two material properties as quantities of interest: the failure strain ε_f and the toughness U. We employ Latin Hypercube sampling [255] considering 2,000 training data points where each point has a random realization of the microstructure. However, only simulations that lead to complete stress-strain curves are considered, which led to 1,473 training data points. Figure 6.3b provides a broad view on how each quantity of interest changes with respect to the three inputs (design variables in Table 6.3), including variations as seen in the box plot. Figure E.10 shows a scatter plot visualizing the relationship between inputs and outputs, where a clear non-linear relationship can be observed. For testing purposes, we generated 36 test points with grid search, with $E_c \in [20, 75, 150, 200]$, $w_{PP} \in [0.25, 0.50, 0.75]$, and $w_c \in [0.05, 0.10, 0.17]$, respectively. Each test point is simulated with 300 different realizations to obtain a proper estimate of the ground truth mean and aleatoric uncertainty. This is a brute-force Monte Carlo sampling to estimate the empirical distribution corresponding to a given input configuration. Specifically, we fix the input variable $[E_c, w_{PP}, w_c]$, and then randomly sample different realizations of microstructure, using a different seed, as shown in Figure 6.3b. We visualize three slices of test points in Figure 6.5. Increasing E_c leads to an increase of both ε_f and U first, and then it reaches a peak followed by a slight drop. Meanwhile, a larger w_c

also increases both QoIs. On the contrary, increasing w_{PP} will decrease both QoIs. More importantly, significant input-dependent data variability is observed across all cases.

6.4.2. RESULTS OF FORWARD PREDICTION

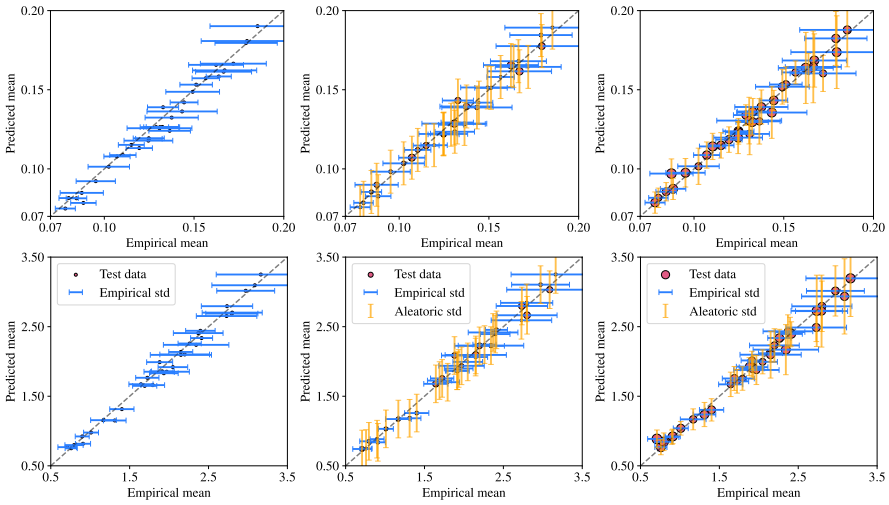
We employed a two-layer feedforward neural network with 128 hidden neurons and *Tanh* activation for the mean estimation network, and a single-layer network with 8 hidden neurons for the variance estimation network. Deterministic training (Step 1) was performed using the *Adam* optimizer with a learning rate of 10^{-3} for 2000 epochs, while the variance estimation network was trained for 4000 epochs. For posterior sampling, we used pSGLD with a learning rate of 10^{-2}, a burn-in of 100 epochs, and collected 200 posterior samples every 10 epochs over 2100 epochs. The batch size was 512. The training dataset was split 80/20 for deterministic training (step 1), whereas cooperative training used no validation split and was run with $K = 2$ iterations.

Since VeBNN outputs the predictive mean, aleatoric, and epistemic uncertainties, its overall performance needs to be evaluated with more than one metric. Therefore, five complementary metrics are adopted (see Appendix D.3): the relative error (ϵ_r) evaluates the accuracy of the mean prediction; the Wasserstein Distance (WA) assesses the quality of aleatoric uncertainty estimates; and the Epistemic Test Log-Likelihood (Epistemic TLL), the Prediction Interval Coverage Probability (PICP), and Mean Prediction Interval Width (MPIW) to evaluate epistemic uncertainty. A comparative analysis of these metrics for both failure strain and toughness is provided in Table 6.4, comparing VeBNN to a DNN trained using MSE loss and a GPR under the homoscedastic noise assumption.

Table 6.4: Summarizing the performance metrics for the test dataset across different quantities of interest.

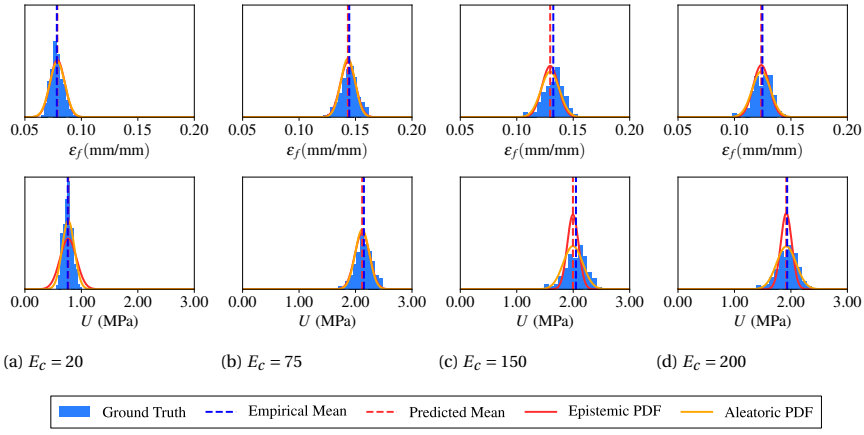
QoI	Method	Mean	Aleatoric uncertainty	Epistemic uncertainty		
		ϵ_r (↓)	WA (↓)	Epistemic TLL (↑)	PICP (→ 0.95)	MPIW (↓)
ε_f (mm/mm)	DNN	0.0317	-	-	-	-
	GPR	0.0241	0.0065	3.8718	0.7941	**0.0123**
	VeBNN	**0.0240**	**0.0042**	**3.8953**	**0.9706**	0.0249
U (MPa)	MSE	0.0366	-	-	-	
	GPR	0.0349	0.1472	0.7482	0.1003	**0.2208**
	VeBNN	**0.0341**	**0.0778**	**0.9888**	**1.0000**	0.4867

As shown in Table 6.4, all DNN, GPR, and VeBNN models achieve accurate mean predictions ($\epsilon_r < 3.5\%$), although the VeBNN slightly outperforms both baselines. However, DNN only predicts the mean, which makes it incapable of uncertainty quantification. On the other hand, both GPR and VeBNN provide epistemic uncertainty quantification. GPR can provide an estimate of aleatoric uncertainty by considering homoscedastic noise as a hyperparameter, unlike the VeBNN that actually learns the heteroscedastic noise and clearly outperforms GPR in this estimation. In particular, VeBNN achieves WA values of 0.0042 and 0.0778 for failure strain and toughness, respectively, which are almost half those of GPR (0.0065 and 0.1472). In terms of epistemic uncertainty, VeBNN also demonstrates superiority over GPR, achieving higher Epistemic TLL. It is noteworthy, however, that while GPR produces a smaller MPIW than VeBNN, this comes at the cost of a substantially lower PICP. Both metrics need to be considered together when evaluating the quality of the epistemic uncertainty estimation.



(a). Results of DNN (b). Results of GPR (c). Results of VeBNN

Figure 6.6: Comparison between the predictive and empirical distributions from the test dataset. The top and bottom rows correspond to failure strain and toughness, respectively. The dashed diagonal line indicates the ideal case, where the predicted mean perfectly matches the empirical mean. Marker size reflects the magnitude of epistemic uncertainty, with larger circles denoting higher predicted epistemic standard deviation.



(a) $E_c = 20$ (b) $E_c = 75$ (c) $E_c = 150$ (d) $E_c = 200$

Figure 6.7: Predicted uncertainty distributions for varying values of E_{comp}, with fixed $w_{\text{PP}} = 0.5$ and $w_c = 0.1$, across different QoIs.

To better evaluate the model performance, we visualize the predicted means together with the empirical distributions of the test data for all compared methods in Figure 6.6. The first column presents the results of the DNN, which also provides accurate mean predictions that closely follow the empirical means (vertical dashed blue lines). However, its lack of uncertainty estimation prevents the model from reflecting the variability and confidence levels inherent in the data. The second column shows the GPR predictions,

where the assumption of homoscedastic aleatoric uncertainty results in identical error bars across all samples, thereby failing to capture the heteroscedastic noise investigated in this study. In contrast, the proposed VeBNN (third column) achieves consistent agreement with the empirical distributions and more accurately represents the input-dependent aleatoric uncertainty. Notably, regarding the predicted epistemic uncertainty of GPR and VeBNN, GPR has tighter bounds that reflect higher confidence in its predictions (size of the circular markers is smaller). However, this overconfidence reported in Table 6.4 is undesirable because the PICP metric indicates that only a small percentage of sampling points is included in the estimated 95% condifence interval, i.e., most samples fall outside the interval. Encouragingly, the VeBNN estimation of the confidence interval is close to the empirical 95% confidence interval.

In addition, Figure 6.7 shows the VeBNN predictions for several test points (unseen designs) against the empirical distribution obtained from a histogram plot obtained from collecting 300 samples of that (unseen) design. The predicted mean aligns well with the empirical mean, and the predicted epistemic uncertainty appropriately encompasses the empirical mean in all cases. Notably, the predicted aleatoric uncertainty captures the shape and spread of the ground-truth distributions accurately. This capability is crucial for downstream tasks such as inverse design under uncertainty.

Remark 6.4.1. *While the predicted aleatoric uncertainty demonstrates excellent agreement with the ground truth, as shown in Figure 6.7, it is constrained to a Gaussian distribution in this study. However, the true data distribution may not be Gaussian. In such cases, although the predicted Gaussian distribution approximates the empirical data reasonably well, a more appropriate choice of distribution could further improve the modeling accuracy, which is a potential direction for future research.*

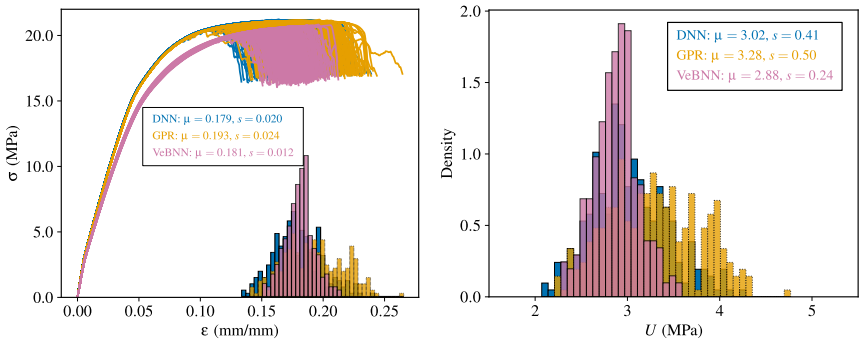
6.4.3. RESULTS OF UNCERTAINTY-AWARE INVERSE DESIGN

In this section, we conduct inverse design using the trained model from Section 6.4.2, focusing on the optimization of toughness U, which serves as a more comprehensive target by integrating the stress–strain response up to failure. In this study, we use a Differential Evolution algorithm [256] with a population size of 100 over 100 generations. For comparison, the inverse design results obtained with DNN and GPR are also included and summarized in Table 6.5.

Table 6.5: Optimal design configuration and predictions of optimizing U using different models. The results illustrate the optimal design found when considering surrogate models obtained from the respective machine learning method (DNN, GPR and VeBNN).

Method	Optimal scheme			Predicted U (MPa)		
	E_c (MPa)	w_{PP}	w_c	Mean	Aleatoric std	Epistemic std
DNN (Equation (6.9))	145.86	0.2000	0.0962	3.22	–	–
GPR (Equation (6.10))	137.20	0.2189	0.1049	3.31	0.2702	0.0528
VeBNN (Equation (6.8))	95.40	0.3072	0.1566	2.95	0.2680	0.1183

As summarized in Table 6.5, the three models that were trained on the same data lead to distinct optimal designs. The DNN model can only be used considering the mean response (conventional inverse design), as in Equation (6.9). The GPR only has reliable



(a) Homogenized stress-strain curves

(b) Histogram plot of U

Figure 6.8: Comparison between the empirical distributions from 300 SVE realizations for all three optimal design configurations of optimization U. The empirical mean and standard deviation of each optimal design are presented in the black box.

epistemic uncertainty, as its aleatoric uncertainty remains constant (0.2702) across the design space. Therefore, GPR finds an optimal design according to Equation (6.10). The VeBNN is capable of estimating both epistemic and aleatoric uncertainty, allowing to search for an optimal design that reflects both quantities, as in Equation (6.8).

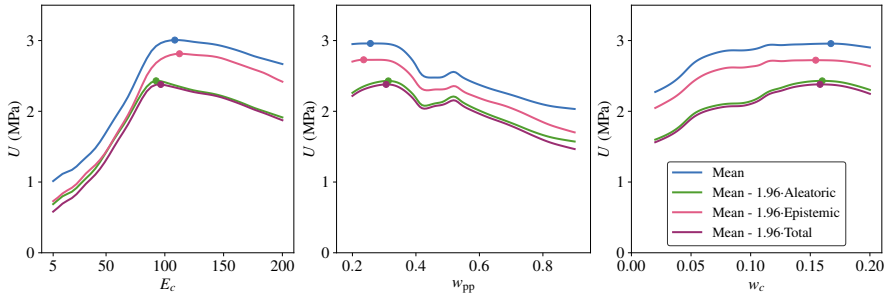
In order to assess the quality of the designs that are obtained from the different surrogate models, Monte Carlo simulations were conducted to determine the distribution, as shown in Figure 6.8. The homogenized stress-strain curves in Figure 6.8a of all three optimal design schemes indicate that VeBNN produces about half the spread in failure strain at the cost of a slightly lower failure stress. A similar pattern is observed in the empirical distribution of toughness (U) in Figure 6.8b. Compared to the predictions in Table 6.5, all models capture the mean values reasonably well. However, only VeBNN succeeds in predicting the aleatoric uncertainty, with predicted standard deviation of 0.2680 that closely matches the empirical value of 0.24. Although its mean toughness is slightly lower (2.88), VeBNN provides a safer lower bound of 2.41 at the 95% confidence interval, compared with 2.2164 and 2.2800 for DNN and GPR, respectively.

6.5. DISCUSSION

6.5.1. WHAT CAUSES DESIGN DIFFERENCES WHEN INCORPORATING UNCERTAINTY?

Although incorporating uncertainty into inverse design can yield more robust optimal schemes (see Table 6.5), the underlying reason for the observed shift in the optimal design is not immediately evident. To provide further interpretation, we conduct objective landscape analyses based on the VeBNN, as shown in Figure 6.9.

The objective landscape analysis is performed around the optimal design of VeBNN by varying one design variable while keeping the others fixed. As shown in Figure 6.9, the influence of the epistemic uncertainty is limited, as it does not significantly change the optimal design that is found, since the epistemic uncertainty predicted by VeBNN



(a) $w_{PP} = 0.3072$ and $w_c = 0.1566$ (b) $E_c = 95.40$ and $w_c = 0.1566$ (c) $E_c = 95.40$ and $w_{PP} = 0.3072$

Figure 6.9: Objective landscapes of VeBNN around the optimal design, obtained by varying one design parameter while keeping the others fixed. The circle on each curve represents the maximum value of the corresponding curve.

remains largely consistent. This is because sufficient data samples have been used in training, ensuring accurate mean predictions. Notably, epistemic uncertainty increases only in the extrapolation region ($w_{PP} > 0.9$), as illustrated in Figure 6.9b, but this region is not considered in the inverse design process. In contrast, accounting for aleatoric uncertainty induces a substantial shift in the optimal design, particularly in E_c and w_{PP}. For example, in Figure 6.9b, the predicted mean toughness exhibits a descending trend with increasing w_{PP}, which causes the optimal design that was found when considering only the mean response to lie at the boundary. However, the presence of large aleatoric uncertainty near $w_{PP} = 0.2$ shifts the optimal solution to $w_{PP} = 0.3072$, thereby favoring a design with improved robustness despite a marginally lower mean toughness.

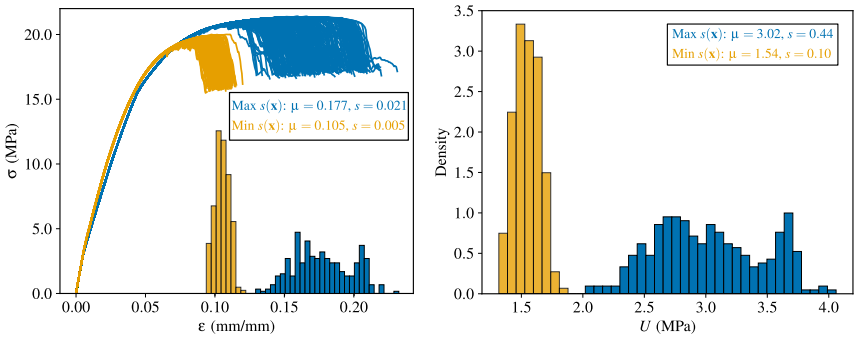
6.5.2. CONTRASTING CASES OF LARGEST AND SMALLEST ALEATORIC UNCERTAINTIES

We also analyzed the FEA predictions for a design with the largest aleatoric uncertainty estimated by the VeBNN when compared to another with the smallest. These two extreme cases are obtained by minimizing and maximizing $s_a(\mathbf{x})$ in the surrogate model obtained by the VeBNN. The corresponding designs and their Monte Carlo finite element evaluation are summarized in Table 6.6 and illustrated in Figure 6.10, respectively.

Table 6.6: Optimal design configuration and predictions of optimizing U using different models. The results illustrate the predictions obtained under each model for its own optimal design; the values are not directly comparable across models.

Method	design scheme			Predicted U (MPa)		
	E_c (MPa)	w_{PP}	w_c	Mean	Aleatoric std	Epistemic std
Max $s_a(\mathbf{x})$	200.00	0.2000	0.1122	3.05	0.6443	0.1879
Min $s_a(\mathbf{x})$	47.39	0.8277	0.02	1.49	0.0958	0.1308

These contrasting cases highlight the fundamental trade-off in uncertainty-aware design: changing the design variables ($E_c\uparrow$, $w_{PP}\downarrow$) can improve toughness but it also increases aleatoric uncertainty, whereas shifting towards higher w_{PP} and lower w_c reduces



(a) Homogenized stress-strain curves

(b) Histogram plot of U

Figure 6.10: Empirical distributions from 300 SVE realizations for both design configurations of maximizing/minimizing $s_a(\mathbf{x})$ for U.

uncertainty at the expense of toughness.

6.6. CONCLUSIONS

This study presents a scalable Bayesian machine learning and inverse design framework for recycled PP/PE blends, establishing a new paradigm for data-driven design of sustainable composite materials. A finite element analysis model is first developed to simulate the micro-scale structure–property relationships of PP/PE blends, enabling the generation of a large-scale dataset with pronounced data variability associated with failure mechanisms. To address the inherent data noise and model uncertainty, we employ a Variance Estimation Bayesian Neural Network (VeBNN), which simultaneously predicts the mean response, estimates aleatoric uncertainty, and disentangles epistemic uncertainty. The trained VeBNN achieves less than 3.5% relative error for both failure strain and toughness, while its aleatoric uncertainty estimation aligns well with empirical distributions, and the epistemic uncertainty prediction reflects appropriate model error or confidence. The trained VeBNN is further used in an uncertainty-aware inverse design problem aimed at maximizing composite toughness while minimizing data noise. Results demonstrate that accounting for uncertainty yields a significantly more robust and reliable optimal design compared to conventional inverse methods that neglect uncertainty or consider homoscedastic noise.

The proposed framework is broadly applicable to other sustainable composite systems with heterogeneous and uncertain behavior. Future work includes extending the model to handle non-Gaussian data noise, validating the approach with experimental data on real recycled PP/PE blends, and the consideration of Bayesian optimization with VeBNNs.

7

CONCLUSIONS AND OUTLOOK

I proposed several principle probabilistic machine learning approaches and advance the application to material modeling and design from a Bayesian perspective. This chapter concludes the dissertation by summarizing the key findings and discussing the outlook with emphasis on further methodology development, real-world application, and long-term impacts.

7.1. CONCLUSIONS

This dissertation is motivated by the goal of advancing data-driven methods from a Bayesian perspective, and integrating multi-fidelity data sources that differ in cost, error, and noise levels. Since the topic of probabilistic machine learning is less familiar to certain domains such as computational mechanics, I begin with an educational chapter (Chapter 2) that introduces regression models from a Bayesian perspective. This chapter covers different models ranging from the simple linear model to the complex neural networks, while highlighting the connections and distinctions with their deterministic counterparts. Subsequently, I introduce inference methods, which are central to enabling scalable Bayesian models—particularly Bayesian neural networks—since obtaining analytical posteriors is generally impractical.

In Chapter 3, I propose a novel cooperative training strategy to address a key bottleneck in Bayesian deep learning—namely, the disentanglement of *aleatoric* (data) uncertainty and *epistemic* (model) uncertainty, as posed in **RQ1**. The strategy begins with deterministic training for the mean, then employs a separate neural network to learn heteroscedastic data noise using a proposed Gamma loss, and finally applies a Bayesian treatment to update the mean and quantify epistemic uncertainty. This approach is straightforward to implement and compatible with various neural architectures. It demonstrates excellent performance in both low-data and large-scale settings, from a one-dimensional toy problem with only 5 training points using a DNN, to an image regression task with tens of thousands of samples using a ResNet-34 architecture (approximately 28.1 million parameters).

In Chapter 4, I present a practical multi-fidelity machine learning framework that addresses **RQ2** by unifying the topic through a general formulation (Equation (4.1)). The objective of Chapter 4 is to accurately predict the high-fidelity mean and *epistemic* uncertainty. Thus, the proposed strategy employs a deterministic model for the low-fidelity dataset, followed by a linear transfer model to map the learned low-fidelity representation to the high-fidelity space. Finally, a Bayesian model is employed to learn the residual while providing *epistemic* uncertainty prediction. Within the practical framework, it versatile to different model choices. In particular, we recommend modeling choices for two scenarios, and argue in favor of using a linear transfer-learning model that fuses 1) kernel ridge regression for low-fidelity with Gaussian process for high-fidelity; or 2) deep neural network for low-fidelity with a Bayesian neural network for high-fidelity. We demonstrate the effectiveness and efficiency of the proposed strategies and contrast them with the state-of-the-art based on various numerical examples and two engineering problems.

By combining the cooperative training strategy with the multi-fidelity machine learning framework, Chapter 5 extends these approaches to history-dependent datasets based on recurrent neural networks (addressing **RQ3**), with a particular focus on constitutive law modeling. The generalized data-driven modeling framework follows a staged scheme, ranging from the simplest deterministic model for single-fidelity problems to a variance-estimation Bayesian recurrent neural network (from Chapter 3) applied across all fidelity levels, where aleatoric and epistemic uncertainties are disentangled for each fidelity. The proposed method is evaluated using novel datasets for history-dependent constitutive modeling with multiple data fidelity levels and including both noiseless and noisy cases.

Chapter 6 deals with sustainable composites design problem in a probabilistic dia-

gram for addressing **RQ4**. We first propose a novel finite element modeling strategy for polymer-polymer composites, to simulate the homogenized mechanical properties up to failure. The variance estimation Bayesian neural network is then trained for the forward problem of predicting the quantities of interests while disentangling both uncertainties. Building on this, we formulate an uncertainty-aware inverse design strategy that optimizes not only the mean performance but also minimizes uncertainties. The proposed design scheme achieves comparable mean performance to the conventional deterministic inverse design optimum, while reducing data variation to roughly half. These findings open new avenues for the design and optimization of sustainable composite polymers with robust and predictable mechanical behavior, and for other data-driven engineering design problems involving uncertainty.

7.2. OUTLOOK

Multi-fidelity Bayesian machine learning with uncertainty quantification and disentanglement, as investigated in this dissertation, holds significant value for both the artificial intelligence community and real-world applications in material and structural design. In this section, I further outline several promising research directions and highlight the associated challenges that need to be addressed to advance the field.

Application to large language models: Large language models (LLMs), exemplified by ChatGPT [1], are now widely used in everyday life across diverse personal and professional contexts. Despite their impressive performance, LLMs face the bottleneck of getting into local optimum easily, i.e., often repeating or converging to similar responses for a given set of prompts [154], due to the fact that they are trained deterministically. Moreover, LLMs lack the capability of providing their confidence (i.e., *epistemic* uncertainty) about corresponding predictions. The cooperative training strategy proposed in Chapter 3 has great potential to further ahead LLMs into a Bayesian paradigm, enabling both better exploration of the solution space and explicit quantification of predictive uncertainty. However, the practical challenge lies the enormous computational and memory budget of Bayesian training for such LLMs because they often contains hundreds of billions of neural parameters. Existing Bayesian inference methods are mostly time-consuming and memory-intensive, making it difficult to directly apply the cooperative training strategy to LLMs. To this end, developing scalable and fast approximate Bayesian inference method is the urgent challenge for bringing Bayesian training for LLMs.

Extend to arbitrary data noise distributions: In this dissertation, data noise is assumed to be known and to follow a Gaussian distribution. The results presented across different problems are promising and largely align with the corresponding ground truth. However, in real-world problems, prior information on the noise distribution is often unknown or only partially known, which is a limitation of the current cooperative training strategy. For example, Figure 6.7 shows a case where the ground truth more closely follows a log-normal distribution than a Gaussian distribution. In this case, although the predicted Gaussian aleatoric distribution achieves an excellent match in uncertainty width, the accuracy could be further improved if the correct noise distribution were assumed. This will be

an interesting topic for the further research, where extend cooperative training to learn or adaptive infer arbitrary noise distribution from data. Recently, prior work has begun to emerge under the term imprecise probabilistic machine learning [83]. Furthermore, generative models [45] can also be considered for learning data variance, given their capacity to model and generate samples from arbitrary distributions.

Scaling multi-fidelity modeling to experiments: One of the main motivations for using multi-fidelity data-driven modeling is to reduce cost or compensate for the inability to generate a sufficient number of high-fidelity training points. Although its advantages have been demonstrated computationally in both Chapter 4 and Chapter 5, the approach has not yet been applied to problems involving experimental data, which would represent the ideal scenario for multi-fidelity data-driven modeling. It will be interesting to advance the developed generalized data-driven modeling from Chapter 5 to handle problems with experiments, in which the potential limitation will be the liability of the low-fidelity model to account for the manufacturing parameters.

Active learning and Bayesian optimization: In the context of Bayesian learning, we have demonstrated that the estimated *aleatoric* uncertainty provides insight into intrinsic data variation, whereas the *epistemic* uncertainty offers informative evidence of the model's performance. This makes active learning and Bayesian optimization [257] two promising directions for future exploration. Both approaches share a similar paradigm: an initial model is trained with a small number of samples, and the model is then sequentially updated with new samples selected according to a specified acquisition function. In particular, active learning aims to produce a final model with accurate global predictive performance, while Bayesian optimization focuses on efficiently optimizing the quantities of interest. In the literature, single-fidelity active learning and Bayesian optimization have been extensively studied for data-scarce scenarios. With the proposed generalized data-driven modeling framework, there is an opportunity to explore scalable variants of these methods. Furthermore, it would be valuable to extend active learning and Bayesian optimization to multi-fidelity datasets, where acquisition functions must be defined to jointly account for cost, high-to-low fidelity correlations, and expected improvements in accuracy or optimization objectives.

A

RVESIMULATOR: AN AUTOMATED REPRESENTATIVE VOLUME ELEMENT SIMULATOR FOR DATA-DRIVEN MATERIAL DISCOVERY

In this appendix, I will introduce an open-source platform rvesimulator, which aims to provide a user-friendly, automated Python-based platform conducting Representative Volume Element (RVE) simulation. By utilizing this repository, large amount of reliable dataset generation is possible with RVEs encompassing materials from elastic to plastic composites. By sharing the developing platform, we are aiming to reduce the labor-intensive process of generating massive simulations data for new materials and structure discovery. Therefore, it facilitates the application and development of machine learning method for new material discovery. More details about the rvesimulator can be referred to the GitHub repository: https://github.com/bessagroup/rvesimulator.git

This appendix is partly based on the manuscript: **Yi, J.**, & Bessa, M. A. (2023). rvesimulator: An automated representative volume element simulator for data-driven material discovery. In AI for Accelerated Materials Design-NeurIPS 2023 Workshop.

A

A.1. INTRODUCTION

The pursuit of discovering new materials remains a central goal in materials science. [258] reviewed the development of Finite Element Method (FEM) over the past eighty years, and claimed that Neural Networks (NNs) integrated with FEM have opened new avenues for discovering new material with unprecedented properties. Recently, advanced machine learning technologies such as Feedforward Neural Networks (FNNs) [259], Convolutional Neural Networks (CNNs) [30], Graph Neural Network (GNN)[260], Recurrent Neural Network (RNN)[34], etc. are utilized for learning and finding better material properties. Large high-fidelity data-set is expected for training well performed aforementioned machine learning models, whereas the process of generating such data is notably absent from the existing literature. There are several open-source FEM packages, such as Fenics[261], JAX-FEM[262], however their capability for high nonlinear and complex material like composite is limited. Abaqus [263], renowned for its powerful nonlinear solver capable of handling intricate and large-scale simulations, is prevalent among material science researchers. However, GUI operations are still commonly conducted to obtain datasets for researchers who are using Abaqus even though it is a repeatable process. In addition, Reduced Order Modeling (ROM) has been proposed as an efficient alternative for generating large datasets [10], where Self-Consistent Clustering Analysis (SCA) [87, 131] is employed as a faster solver. With the advent of open-source Python packages such as CRATE [131], it is now possible to run SCA simulations using provided Python script templates; thus, additional manual effort is still required for batch simulations.

To this end, the *rvesimulator* is a flexible and user-friendly repository based on Python, designed specifically for RVE simulations with support for both Abaqus and CRATE solvers. With the developed platform, the automates repeatable tasks such as geometry modeling, periodic boundary condition, job submission, and post-processing are already coded. For the potential design variables which are microstructures, material models, and loading; the user can use design of experiments (DoE) methods to draw their samples (data points) [255]. Then, the DoE can be regard as the arguments to the *rvesimulator* to run simulations automatically. By introduce this ease to use repository, we hope to narrow of gap of machine learning and new material design.

A.2. GENERAL WORKFLOW

The schematic of the developing *rvesimulator* is shown in Figure A.1, which comprises two important modules. The first module is encapsulated as functions and could be seamlessly embedded into the data-driven modeling process. The other is Python scripts for executing RVE simulations within given solver, i.e., Abaqus or CRATE. Specifically, in the **Master control** module, the essential configurations like microstructure information, material model parameters, and loading are set and then pass to different simulators selected by the user. Finally, the jobs will be submitted and results will be collected back to the **Master control** automatically, which can be used to train machine learning models. In the following subsections, we will briefly introduce the cross-platform function for executing Abaqus and CRATE scripts.

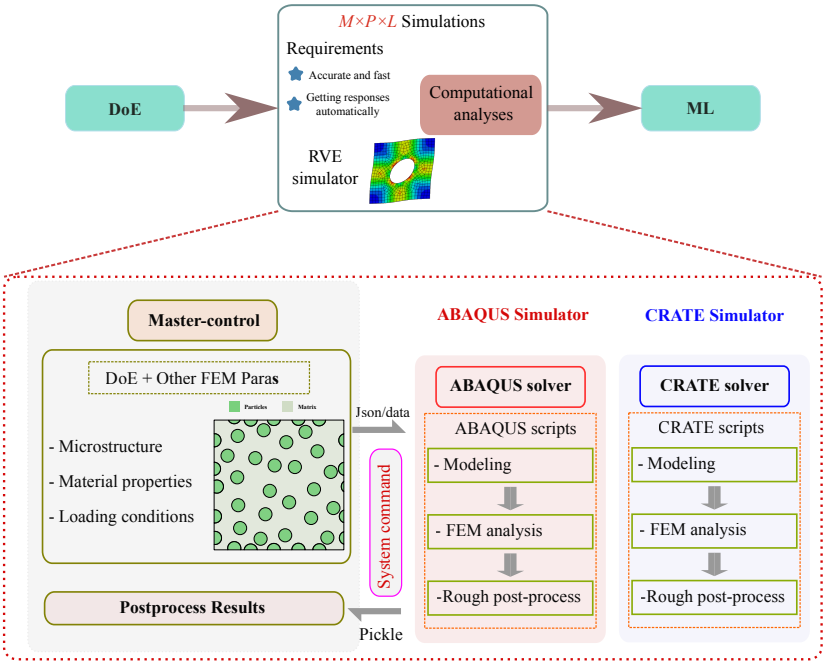


Figure A.1: Workflow of the developing *rvesimulator*. DoE goes into the Master control module. The *rvesimulator* encodes DoE and other FEM parameters into a JSON file. Subsequently, *rvesimulator* has two class objects, i.e., *AbaqusSimulator* and *CrateSimulator* to execute Abaqus and CRATE scripts, respectively. The resulting Pickle file from simulation will be read back to Python and can be utilized for various machine learning methods.

CROSS-PLATFORM FUNCTION FOR NOGUI ABAQUS JOB EXECUTION

While advanced Abaqus users can run simulations through terminal operations without GUI, this approach offers only marginal speed improvements and does not integrate RVE simulations into data-driven modeling workflows. To address these limitations, we implemented a class object, *AbaqusSimulator*, which enables seamless communication between Python and Abaqus while retrieving simulation data silently. The usage of this cross-platform class is outlined as follows:

```
from rvesimulator.abaqus2py.abaqus_simulator import AbaqusSimulator
# folder info dict contains locations of ABAQUS benchmark scripts
folder_info = {}
# sim_info dict contains one raw of design of experiment numpy array
sim_info = {}
# intialize the simulator
simulator = AbaqusSimulator(sim_info=sim_info, folder_info=folder_info)
# run simulation
simulation.run()
# get results
results = simulator.read_back_results()
```

With this *AbaqusSimulator* object in place, it can handle any Abaqus simulation theoretically as long as the user slightly modifies their scripts according to our online guidance.

A

CROSS-PLATFORM FUNCTION FOR CRATE JOB EXECUTION

In the case of CRATE, although it is also developed in Python, the core engine still requires script-based input to define simulation details—ranging from microstructure information and material parameters to simulation settings. Therefore, we develop a similar interface class object, *CrateSimulator*, which can be used similarly as shown in Appendix A.2 by replacing *AbaqusSimulator* to *CrateSimulator*.

A.3. EXAMPLES

In this section, we present a concrete example based on datasets generated using the Cooperative Data-Driven Modelling (CDDM) method. The dataset includes four distinct RVEs, each with unique microstructural features and material parameters; further details can be found in [35]. This dataset is designed to capture the history-dependent plasticity behavior of fiber-reinforced composites, where stress–strain paths are used to train RNNs. To generate a dataset corresponding to a single task consisting of 1000 stress–strain paths, the following script is provided:

```
# import objects from rvesimulator
from rvesimulator.benchmarks.cddm_rve import CDDM_RVE
from rvesimulator.additions.ampitudesampler import AmplitudeGenerator
from rvesimulator.additions.hardening_law import LinearHardeningLaw
# number of path
num_paths = 1000
# initialize path sampler
path_sampler = AmplitudeGenerator(num_dim=3)
# get strain paths
paths = path_sampler.get_amplitude(
num_amplitude=num_paths,
)
# initialization for simulation task
task = CDDM_RVE()
# update simulation info
task.update_sim_info(mesh_partition=100,
vol_req=0.45,
radius_mu=0.01,
radius_std=0.003,
youngs_modulus_fiber=10,
youngs_modulus_matrix=100,
hardening_law=LinearHardeningLaw(a=0.5, b=0.5, yield_stress=0.5))
# empty dict to save results
results = {}
# calculate responses of simulation in sequential
for ii in range(len(paths)):
  results[ii] = task.run_simulation(sample=paths[ii], folder_index=ii
  )

```

As summarized in the Python script, it can easily generate 1000 stress-strain paths that can used to train RNN model that learns the plasticity law for the given fiber reinforced composite. The microstucture generated by the Python script is shown in Figure A.2.

According to Figure A.2 and the provided script, the RVE size is 0.048 mm, and the particles are not uniformly distributed. Instead, their radii follow a normal distribution $\mathcal{N}(0.01, 0.003)$. For the material properties, both phases are modeled with elastic behavior

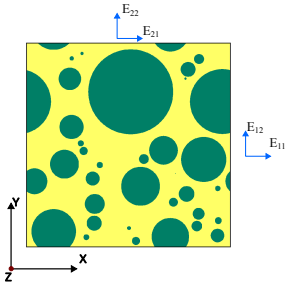
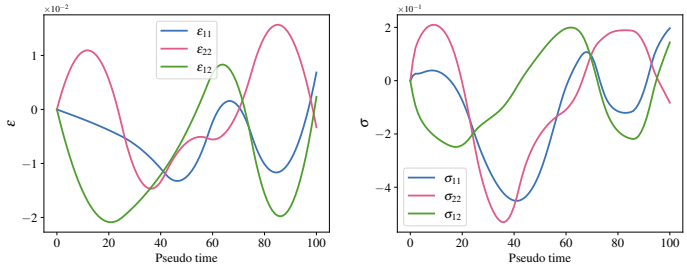


Figure A.2: Microstructure of the illustrated RVE simulation task generated from the given script.

using a Young's modulus of 10 MPa and a Poisson's ratio of 0.19. The matrix phase, however, follows a von Mises plasticity law [264], with a Young's modulus of 100 MPa, Poisson's ratio of 0.3, and a linear hardening law given by $\sigma_y = 0.5 + 0.5\,\bar{\varepsilon}$, where $\bar{\varepsilon}$ denotes the accumulated plastic strain. Figure A.3 presents an arbitrary realization of strain and stress paths, where the strain components shown in Figure A.3a are applied to the RVE simultaneously, and the corresponding stress responses are depicted in Figure A.3b.



(a) Strain inputs (b) Stress outputs

Figure A.3: An arbitrary realization of Strain inputs and corresponding history-dependent stress outputs

A.4. CONCLUSION

The shared *rvesimulator* is a flexible and user-friendly platform for generating high fidelity FEM simulation data for RVEs. It includes a variety of implemented benchmarks that cover a wide range of commonly investigated RVEs. Users also have the capability to generate their specific datasets by designing unique experiments tailored for different machine learning methods. For users seeking more advanced RVE simulations, the platform allows for the adaptation of their own Python-Abaqus scripts, enabling the execution of numerous simulations with ease. Additionally, the developing repository can easily adapt to parallelization tools, providing the potential for further acceleration of simulations.

B

APPENDICES FOR CHAPTER 3

B.1. MEAN VARIANCE ESTIMATION NETWORKS

COMPARISON BETWEEN TRAINING SINGLE NETWORK AND SEPARATE TRAINING OF TWO NETWORKS

As discussed in Section 3.3.4, we can train a mean variance estimation (MVE) network that predicts both mean and variance together, or we can train two separate networks: a mean mean estimation network and a variance estimation network. Figure B.1 shows the difference when they are trained for the one-dimensional dataset.

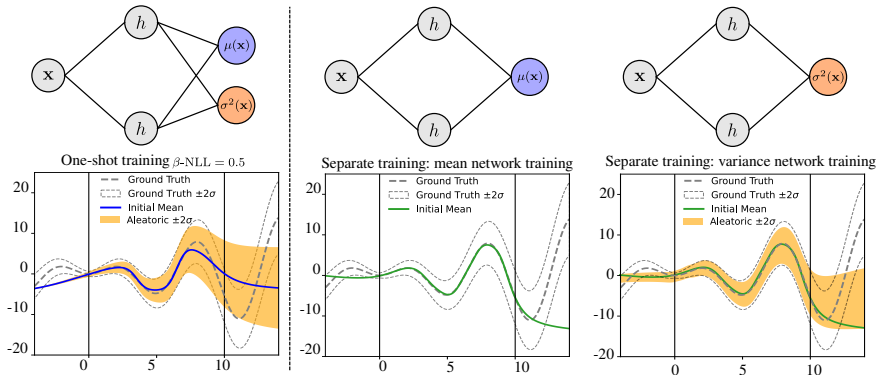


Figure B.1: Comparison between MVE network training (left figure) and separate training of ME network and VE network (right figure). The MVE network outputs $\mu(\boldsymbol{x})$ and $\sigma_a^2(\boldsymbol{x})$ and is trained by minimizing Equation (3.2). The ME network is trained first using Equation (3.2) assuming constant aleatoric uncertainty, and then the VE network is trained using Equation (3.6) and assuming a fixed mean obtained from the ME network.

MVE gives worse predictions in the region ($x > 7$) with higher aleatoric variance prediction. The reason why this happened is that the high-order variance term appears in the denominator of the gradient of Equation (3.2), which will be introduced in the next section. Conversely, this should not be surprising in light of the vast empirical evidence

throughout the literature on the successes of training deterministic models that only predict the mean as shown in the middle figure. Based on it, the variance prediction of separate training also aligns with the ground truth.

B

β-NLL

β-NLL loss [57] was proposed by introducing an additional variance-weighting term, which is given as follows:

$$\mathscr{L}_{\beta\text{-NLL}} = \frac{1}{N}\lfloor\sigma_a^{2\beta}(\boldsymbol{x}_n)\rfloor \sum_{i=n}^{N}\left[\frac{(y_n-\mu(\boldsymbol{x}_n))^2}{2\sigma_a^2(\boldsymbol{x}_n)} + \frac{\log(\sigma_a^2(\boldsymbol{x}_n))}{2}\right] \tag{B.1}$$

where $\lfloor\cdot\rfloor$ represents the stop gradient operation. Under this setup, the gradient of the $\beta-$NLL loss becomes:

$$\nabla_{\mu}\mathscr{L}_{\beta\text{-NLL}} = \frac{1}{N}\sum_{n=1}^{N}\left(\frac{\mu(\boldsymbol{x}_n)-y_n}{\sigma_a^{2-2\beta}(\boldsymbol{x}_n)}\right) \tag{B.2}$$

$$\nabla_{\sigma_a^2}\mathscr{L}_{\beta\text{-NLL}} = \frac{1}{2N}\sum_{n=1}^{N}\left(\frac{\sigma_a^2(\boldsymbol{x}_n)-(y_n-\mu(\boldsymbol{x}_n))^2}{\sigma_a^{4-2\beta}(\boldsymbol{x}_n)}\right) \tag{B.3}$$

According to Equation (B.1), the variance-weighting term β can interpolate between the original NLL loss and equivalent MSE, recovering the original NLL loss when $\beta = 0$. The gradient of Equation (B.2) is equivalent to MSE for $\beta = 1$.

B.2. PERFORMANCE METRICS

We use the Root Mean Square Error (RMSE) and Test Log-Likelihood (TLL) for synthetic, UCI regression, and plasticity law datasets. In addition, as we know the ground truth aleatoric uncertainty for the last dataset (material plasticity), we also use the Wasserstein distance [162] to evaluate the correctness of the learned aleatoric uncertainty for that problem. Details of those metrics are listed:

- **Root Mean Square Error (RMSE)**
  - For MLP:

$$\text{RMSE} = \sqrt{\frac{1}{N_{test}}\sum_{i=1}^{N_{test}}\left(\bar{\boldsymbol{\mu}}_i-\mathbf{y}_i\right)^T\left(\bar{\boldsymbol{\mu}}_i-\mathbf{y}_i\right)} \tag{B.4}$$

  where N_{test} is the number of test data points, $\bar{\boldsymbol{\mu}}$ is the predictive mean, and $\mathbf{y}$ is the observation values of the test points.
  - For RNN:

$$\text{RMSE} = \frac{1}{N_{test}T}\sum_{i=1}^{N_{test}}\sum_{j=1}^{T}\left(\frac{\left\|\bar{\boldsymbol{\mu}}_{i,j}-\bar{\mathbf{y}}_{i,j}\right\|_F}{\left\|\bar{\mathbf{y}}_{i,j}\right\|_F}\right)\cdot 100\%, \tag{B.5}$$

  where N_{test} is the number of testing points, $\|\cdot\|_F$ is a Frobenius norm, $\bar{\mathbf{y}}_{i,j}$ and $\bar{\boldsymbol{\mu}}_{i,j}$ are ground truth and the predictive mean of i^{th} test sample and j^{th} time step, correspondingly.

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

$$\text{TLL} = \frac{1}{N_{test}T}\sum_{i=1}^{N_{test}}\sum_{j=1}^{T}\log p(\bar{\mathbf{y}}_{i,j}|\boldsymbol{\theta}) \tag{B.6}$$

We also clarify that we use the ground truth $\bar{\mathbf{y}}$ for the plasticity law dataset. Observation values $\mathbf{y}$ are used for synthetic and UCI regression datasets and $T = 1$.

- **Wasserstein distance (WA)**

$$\text{WA} = \frac{1}{N_{\text{test}}T}\sum_{i=1}^{N_{\text{test}}}\sum_{j=1}^{T}\text{W}_2\left(p\left(\bar{\boldsymbol{\mu}}_{i,j},\boldsymbol{\sigma}^2_{a_{i,j}}\mathbf{I}\,|\,\boldsymbol{\theta},\boldsymbol{\phi}\right),q\left(\bar{\mathbf{y}}_{i,j}\right)\right) \tag{B.7}$$

where $p\left(\bar{\boldsymbol{\mu}}_{i,j},\boldsymbol{\sigma}^2_{a_{i,j}}\mathbf{I}\,|\,\boldsymbol{\theta},\boldsymbol{\phi}\right)$ and $q\left(\bar{\mathbf{y}}_{i,j}\right)$ are the predictive and ground truth distributions, respectively. Since Gaussian assumption is applied, we can rewrite eq. (B.7) into:

$$\text{WA} = \frac{1}{N_{\text{test}}T}\sum_{i=1}^{N_{\text{test}}}\sum_{j=1}^{T}\sqrt{\left(\bar{\mathbf{y}}_{i,j}-\bar{\boldsymbol{\mu}}_{i,j}\right)^T\left(\bar{\mathbf{y}}_{i,j}-\bar{\boldsymbol{\mu}}_{i,j}\right)+\left(\boldsymbol{\sigma}^2_{i,j}-\boldsymbol{\sigma}^2_{a\,i,j}\right)^T\left(\boldsymbol{\sigma}^2_{i,j}-\boldsymbol{\sigma}^2_{a\,i,j}\right)} \tag{B.8}$$

Regarding the Image regression datasets, as introduced by respective authors, the aim is to evaluate the reliability of regression uncertainty estimation under distribution shift. In this case, the prediction is first calibrated with a validation dataset (in distribution). Then the test coverage for the test dataset, which has a distribution shift, is evaluated [79].

- **Mean absolute error (MAE)**

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}\left|\bar{\boldsymbol{\mu}}_i-\mathbf{y}_i\right| \tag{B.9}$$

where N is the number of points for validation or testing.

- **val Interval length**

$$\hat{C}_\alpha(\mathbf{x},\mathscr{D}_{val}) = [L_\alpha(\mathbf{x},\mathscr{D}_{val}) - Q_{1-\alpha}(E,\mathscr{D}_{val}),\ U_\alpha(\mathbf{x},\mathscr{D}_{val}) + Q_{1-\alpha}(E,\mathscr{D}_{val})], \tag{B.10}$$

where L_α and U_α are the lower and upper bounds of confidence interval under $\alpha = 0.1$. $E = \{\max(L_\alpha(x_i) - y_i,\ y_i - U_\alpha(x_i)) : i \in \mathscr{D}_{val}\}$ are conformity scores of the validation dataset, and $Q_{1-\alpha}(E,\mathscr{D}_{val})$ is the $(1-\alpha)$-th quantile.

- **Test coverage**
Following the same procedure, we can get the predictive confidence interval for the test dataset calibrated by the validation dataset.

$$\hat{C}_\alpha(\mathbf{x},\mathscr{D}_{test}) = [L_\alpha(\mathbf{x},\mathscr{D}_{test}) - Q_{1-\alpha}(E,\mathscr{D}_{val}),\ U_\alpha(\mathbf{x},\mathscr{D}_{test}) + Q_{1-\alpha}(E,\mathscr{D}_{val})], \tag{B.11}$$

With the predictive confidence interval, we can obtain the test coverage with the following formula:

$$\text{Test coverage} = \frac{1}{N_{test}}\sum_{i=1}^{N_{test}}\mathbb{I}\{y'_i \in \hat{C}_\alpha(\mathbf{x},\mathscr{D}_{test})\}. \tag{B.12}$$

B

B.3. ADDITIONAL RESULTS FOR PLASTICITY LAWS DATASETS

Additional plots of predictions for correct identification We present the predictions of MVE (β-NLL), MVE (MC-Dropout), and VeBNN (MC-Dropout) on the plasticity law discovery dataset in Figure B.2. We first observe that MVE (β-NLL) provides reliable aleatoric uncertainty estimates when sufficient training data is available. However, its predictions—both in terms of the mean and aleatoric uncertainty—deteriorate when the training size is reduced to $N = 50$. For MVE (MC-Dropout), the predictions remain sub-optimal, even with $N = 800$ training sequences. In contrast, the proposed VeBNN (MC-Dropout) improves prediction quality, particularly in estimating aleatoric uncertainty.

It is important to note that predictions for plasticity laws are expected to be smooth. The inherent bumpiness of MC-Dropout-based approaches poses challenges for deployment in Finite Element Analysis in real-world applications.

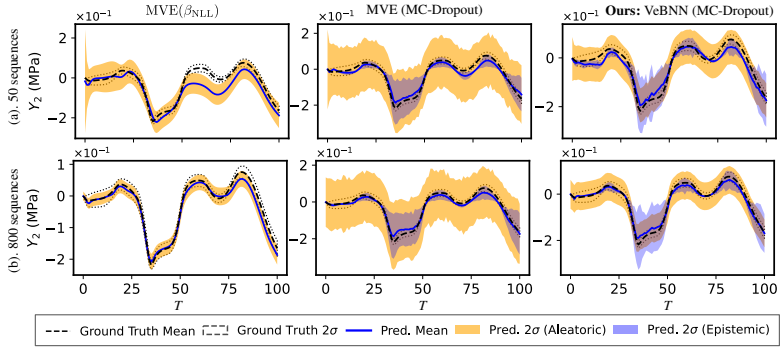


Figure B.2: Predictions of MVE (β-NLL), MVE (MC-Dropout), and VeBNN (MC-Dropout) methods on plasticity law discovery dataset. We upper and bottom rows are 50 and 800 training sequences, respectively.

Experiments on another problem with larger aleatoric uncertainty The same experiment is conducted as Section 3.4.3 for the second problem of Table B.1, where the results are shown in Figure B.3.

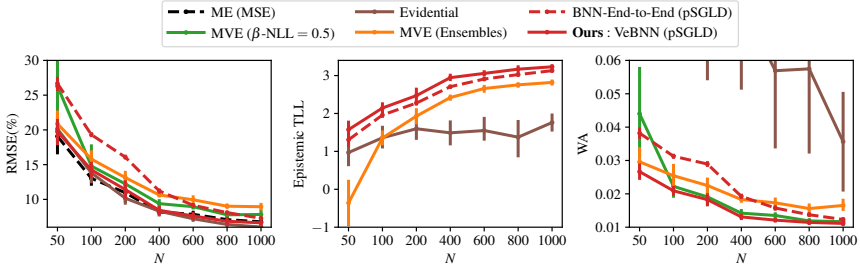


Figure B.3: Accuracy metrics (RMSE ↓, TLL ↑, and WA ↓) obtained for the second problem of the plasticity law discovery dataset. The Wasserstein distance (WA) represents the closeness of the estimated aleatoric uncertainty distribution to the ground truth distribution. All metrics result from repeating the training of each method 5 times by resampling points randomly from the training datasets.

The same pattern appears as Figure 3.9, which shows that the proposed cooperative learning strategy also works for problems with larger data uncertainty. There is no surprise that all approaches are less accurate in this problem under the same amount of training data, comparing results in Figure 3.9 since this problem has larger data uncertainty. Then, we can also see that the proposed cooperative learning strategy has consistently outstanding performance across all accuracy metrics. It is worth mentioning that the proposed cooperative learning strategy has a more significant advantage in the mean prediction compared with all other methods, especially ME (MSE). This demonstrates that other methods are gradually facing troubles with data uncertainty increasing, while the proposed cooperative learning strategy still has robust and trustworthy predictions.

B.4. HYPERPARAMETER SETTINGS

SYNTHETIC DATASETS

Heteroscedastic noise In Section 3.4.1, we consider the following one-dimensional example [56]:

$$y = x \sin x + 0.3 \cdot x \cdot \varepsilon_1 + 0.3 \cdot \varepsilon_2 \tag{B.13}$$

where $\varepsilon_1, \varepsilon_2 \sim \mathcal{N}(0,1)$. In the experiments of Section 3.4.1, we sample points uniformly from $[0,10]$ for training. We generate 1000 points in $[0,10]$ and 1000 points $[-4,0] \cup [10,14]$ to test the performance of different methods.

We depict the results of the illustrative example in Figure 3.1, Figure 3.2, and Figure B.1, for which the hyperparameters are summarized as follows.

- **Architectures:** We use two-layer multi-layer perception (MLP) with 256 neurons for the methods that only require one neural network, except BNN (BBB), where one-layer with 50 neurons is adopted [1]. *Tanh* function is used as the activation function. We note that only the mean is outputted for the ME network but there are two outputs, mean and aleatoric variance, for the MVE network. For the proposed cooperative learning strategy we employ the same architecture for the BNN as the ME network. An additional one-layer MLP with 5 neurons is employed as the variance neural network to learn data uncertainty.

- **Optimizer/Inference:** We use *Adam* optimizer with a learning rate of 0.001 for 20000 epochs to optimize Equation (3.2) for all deterministic methods and MC-Dropout (Dropout rate is 0.1 for each layer). *Adam* is also used to optimize the ELBO of BBB for 10000 epochs with a learning rate of 0.01. As for pSGLD, we set the burn-in epoch to be 10000 and collect 100 posterior samples every 100 epochs. We also optimize the variance network for 5000 epochs with a learning rate of 0.001 and early stopping of 100 epochs for the proposed cooperative learning strategy.

- **Hyperparameter selection:** We select 70% data points for training, and the remaining data is used for finding the best hyperparameters, namely number of training epochs and β in the case based on the NLL loss value. After identifying the best number of epochs, we use all data points to re-train the model under the best

[1] We also try the same architecture with other approaches; however, BBB has difficulty with such a large MLP architecture

hyperparameters. For the proposed cooperative learning strategy, we feed all the data to Algorithm 5 and set the iteration $K = 2$.

B

Homoscedastic noise The homoscedastic noise case has the same ground truth, but instead of input-dependent noise, we consider constant noise, expressed as:

$$y = x \sin x + 0.5 \cdot \varepsilon_2 \tag{B.14}$$

where $\varepsilon_2 \sim \mathcal{N}(0,1)$.

UCI REGRESSION DATASETS

We carefully reviewed the experiments and hyperparameters that were found in other studies using UCI regression datasets [56, 57, 64]. We used similar configurations to these studies. Our dataset splits and randomizations can be found in our code for reproducibility because the UCI regression dataset results can be sensitive to data splits [57]. We also considered multiple experiments per dataset to minimize the impact of randomization. The dataset size and input/output dimensions of each problem are listed in Table 3.1 and Table 3.2 under each column within parentheses. We report the metrics at the original scale and averaged over all outputs.

- **Architectures:** We use a one-layer MLP with 50 neurons followed by *ReLU* activation function for all ME and MVE methods. For the proposed cooperative learning strategy, we employ the same architecture for mean net and BNN, and we additionally use an MLP with 5 neurons followed by *ReLU* activation function as the variance network.
- **Optimizer/Inference:** The methods, including ME, MVE, BBB, as well as the warm-up of the proposed cooperative learning strategy, are trained via *Adam* for 20000 epochs with a learning rate of 0.001 (0.01 for training ELBO), in which the MC-Dropout has a Dropout rate of 0.1 for each layer. As for the pSGLD, we set the burn-in epoch to be 5000 and collected 150 posterior samples every 100 epochs (In total 20000 epochs, which is the same as that of *Adam*). For the additional variance network of the proposed cooperative learning strategy, we use *Adam* to train for 10000 epochs with an early stopping of 100 epochs. The batch size is set to be 256 for all approaches.
- **Hyperparameter selection:** We split each dataset into train-test by 80%-20% randomly 20 times. The train set is divided into 80%-20% for training and validation. We search for the best learning rate within $\{10^{-4}, 3\cdot10^{-4}, 10^{-3}, 3\cdot10^{-3}, 7\cdot10^{-4}\}$ and as well as record the best epoch utilizing the validation NLL loss. After finding the best learning rate and epoch, the final model is trained using all training data points, and metrics are calculated for the test set. It is noted that the presented results of MVE (β-NLL) in Table 3.1 and Table 3.2 are the overall best results among $\{\beta = 0.0, \beta = 0.25, \beta = 0.5, \beta = 0.75, \beta = 1.0\}$. We also conducted a grid search for BNN-Homo methods for suitable constant noise, where the space is set between zero and one, with 10 different values given the consideration of similar computational resources. For the proposed cooperative learning strategy, we feed all the data to Algorithm 5 and set the iteration $K = 2$.

IMAGE REGRESSION DATASETS

We take the image regression dataset introduced in [79], which consists of relatively large-scale problems, with each containing between 6,592 and 20,614 training images depending on the specific task. Each image has a resolution of 64×64 and the target is a one-dimensional output y. Full details can be found in [79]; a summary of the relevant information is provided below:

- **Cells** The dataset contains 10000, 2000, and 10000 images for training, validation, and testing, respectively. The labels have a range of $[0, 200]$, and there is no distribution shift among all the datasets.
- **Cells-Tail** The training and validation datasets contain images with labels in the range of $[50, 150]$; the test dataset has labels with a range of $[0, 200]$
- **Cells-Gap** The training and validation datasets contain images with labels in the range of $[0, 50] \cup [150, 200]$; the test dataset has labels with a range of $[0, 200]$
- **ChairAngle** The dataset 17640, 4410, and 11225 images for training, validation, and testing, respectively. The labels have a range of $[0.1°, 89.9°]$, there is no distribution shift among all the datasets.
- **ChairAngle-Tail** The training/validation datasets contain images whose labels have a range of $[15°, 75°]$; and the test labels have a range of $[0.1°, 89.9°]$.
- **ChairAngle-Gap** The labels have a range of $[0.1°, 30°] \cup [60°, 89.9°]$, and the test labels are in $[0.1°, 89.9°]$.
- **SkinLesion** The dataset contains 6592, 1164, and 2259 images for training, validation, and testing, respectively. The dataset contains four different sub-datasets, in which the first three are split into train/val with 85%/15%; and the fourth sub-dataset is used as the test dataset.
- **AreaBuilding** The dataset contains 180 large aerial images with corresponding building segmentation masks. Specifically, the train/val is obtained from two densely populated American cities cities while the test dataset is from rural European cities. Overall, it contains 11184, 2797, and 3890 images for training, validation, and testing, respectively.

We leverage the hyperparameters in [79] and define ours as follows:

- **Architectures:** ResNet34 backbone [80] is employed for this problem. We use a two-layer MLP to decode the prediction into a Gaussian distribution for MVE (β-NLL), MVE (Ensembles), and MVE (MC-Droout). It is noted that a dropout layer with a dropout rate of 0.1 is followed for each MLP layer. As for the variance net and deep evidential regression, the decoding layer is set to be the corresponding outputs after the ResNet34 backbone.
- **Optimizer/Inference** We use *Adam* with a learning rate of 0.001 for MVE, as well as the warm-up step of the proposed cooperative learning strategy and employ

Adam to run for 75 epochs for the above methods with batch size of 32. As for the pSGLD, we set the burn-in epoch to be 20 and we sample 10 posterior samples every 2 epochs (in total 40 epochs). As for the additional variance network, which is trained with *Adam* for 20 epochs.

B

PLASTICITY LAW DATASETS

Hyperparameter setting for Section 4.4 In the literature of data-driven constitutive laws, several studies address similar problems without considering noise [34, 35]. We leverage their setups and define the hyperparameter setting as follows:

- **Architectures:** For the mean network and BNN, we adopt a two-layer GRU architecture with 128 hidden neurons. It is noted that we only apply the Dropout operation to the hidden-to-decoding layer with a Dropout rate of 0.02. The reason is that this inference method does not show compatible performance when making all weights and biases the Dropout layer. For the proposed cooperative learning strategy, a smaller GRU network with two layers and 8 hidden neurons is employed for the variance network.

- **Optimizer/Inference** We use *Adam* with a learning rate of 0.001 for ME, MVE, as well as the warm-up step of the proposed cooperative learning strategy and employ *Adam* to run for 2000 epochs for the above methods. As for the pSGLD, we set the burn-in epoch to be 100 and we sample 100 posterior samples every 10 epochs (in total 1100 epochs). As for the additional variance network, which is trained with *Adam* for 4000 epochs with an early stopping patience of 50 epochs.

- **Hyperparameter selection:** For every experiment of different training points we reserve 100 validation data points from the training dataset to determine the best epoch for ME and MVE. Subsequently, we combine the validation points with the training set and retrain the final model using the best epoch configuration. For the same reason, the results depicted of MVE (β-NLL) is the overall best among $\{\beta = 0.0, \beta = 0.25, \beta = 0.5, \beta = 0.75, \beta = 1.0\}$. As for the BNN-Homo, we follow the same strategy to execute a grid search for the best homoscedastic noise, where the noise variance is set to be $\{0.04, 0.06, 0.08, 0.10, 0.12\}$ empirically. For the proposed cooperative learning strategy, we set the iteration $K = 2$

Hyperparameter setting for variance network size To investigate the influence of the variance network architecture, we consider eight configurations where the number of layers varies from 1 to 2, and the number of hidden neurons is set to $\{8, 16, 64, 128\}$. The largest configuration, $\{128, 2\}$, is identical to the mean network. All other hyperparameters are consistent with those used in the previous section.

B.5. DESCRIPTION OF PLASTICITY LAW DATASET

The fundamental mechanical law of materials is called a constitutive law. It relates average material deformations to average material stresses at any point in a structure. Constitutive laws can model different Physics behaviors, such as elasticity, hyperelasticity, plasticity,

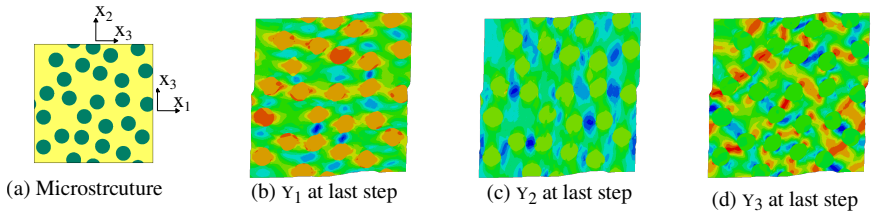
and damage. In this paper, we focus on generating datasets for plastically deforming composite materials, coming from prior work [35, 133]. Without loss of generality, the constitutive law of such path-dependent materials can be formulated as follows:

$$\boldsymbol{y} = f(\boldsymbol{x}, \dot{\boldsymbol{x}}, \tau, \dot{\tau}, \boldsymbol{h}) \tag{B.15}$$

where $\boldsymbol{y}$, $\boldsymbol{x}$, τ are stress, strain, and temperature respectively, $\boldsymbol{h}$ is a set of internal variables. The constitutive law can be predicted by micro-scale simulations of material domains that are called stochastic volume elements (SVEs) – see Figure B.4. These SVEs are simulated by rigorous Physics simulators based on the Finite Element Method (FEM). Each material SVE is utilized as the basic simulation unit. Many factors bring data uncertainty into the data generation process; we focus on data uncertainties from two aspects: (1) SVE size; and (2) particle distribution. As we randomize particle distribution, the stress obtained for an input deformation exhibits stochasticity (aleatoric uncertainty). Therefore, two datasets are generated from simulations according to Table B.1.

Table B.1: Parameter configuration material plasticity law simulations (Units: SI(mm)).

Name	Microstructure Parameters			Hardening Law	E_{fiber}	Size	E_{matrix}	ν_{matrix}	ν_{fiber}
	v_f	r	r_{std}						
Material 1	0.30	0.003	0.0	$\sigma_y = 0.5 + 0.5(\bar{\varepsilon})^{0.4}$	1	0.048	100	0.30	0.19
Material 2	0.30	0.003	0.0	$\sigma_y = 0.5 + 0.5(\bar{\varepsilon})^{0.4}$	1	0.030	100	0.30	0.19



(a) Microstrcuture (b) Y_1 at last step (c) Y_2 at last step (d) Y_3 at last step

Figure B.4: Material plasticity law simulation illustration. Figure. (a) shows an arbitrary realization of material microstructure and the following figures show the contour plot of this material simulation at the final step.

According to Table B.1, we have two materials that have different SVE sizes that control the noise sources of the data. Materials with a smaller SVE size has larger data noise. Figure B.4 and Figure B.5 illustrate details of the simulations and how the uncertainty in the data originates. Specifically, according to the microstructure configuration in Table B.1, we generate SVEs using the Monte Carlo Sampling strategy [223] and simulate the stress responses through the commercial software ABAQUS [265] with the input of the strain sequence shown in the first row of Figure B.5. After simulation, we get a series of contours of stress components in Figure B.4 and average the field (color contours) for each input sequence point, leading to the output sequence. An example of 3 realizations of SVEs and corresponding 3 output response sequences, shown as a dashed line in

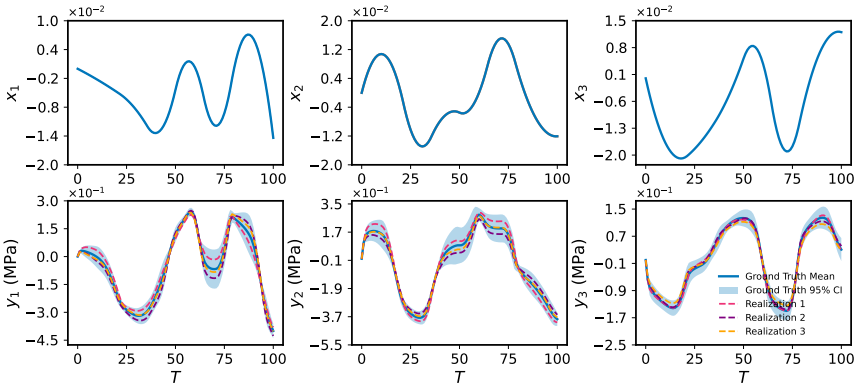


Figure B.5: Plasticity law data illustration. The first column is the strain inputs for material law simulation and the second row is the stress outputs. Each dashed line represents one particle material microstructure realization in Figure B.4; the mean and the confidence interval are obtained via multiple realizations.

the second row of Figure B.5. Each realization corresponds to a randomization of the microstructure of the material (particle distribution). By running multiple realizations, we can obtain the statistics of those strain inputs. By definition, the variation in the stress output is the uncertainty of the data.

Each input deformation sequence and output stress sequence used in training contains 100 points. In total, we use 100 different SVEs to ensure that we have enough realizations to calculate the ground truth aleatoric uncertainty. Overall, we simulate 1000 sequences for training and 100 sequences for testing respectively, for each problem shown in Table B.1.

The new dataset is made available as open-source in the hope of creating a more interesting problem for assessing future methods because we had difficulties in finding more challenging heteroscedastic problems with ground truth aleatoric uncertainty to assess our method. This dataset is three-dimensional and history-dependent, i.e., $\mathscr{D} = \{\mathbf{x}_{n,t}, \mathbf{y}_{n,t}\}$ with features $\mathbf{x}_{n,t} \in \mathbb{R}^3$ and targets $\mathbf{y}_{n,t} \in \mathbb{R}^3$, where $n = 1, ..., N$ are the training sequences (deformation paths) and $t = 1, ..., T$ are the points in each sequence. We highlight two aspects about this dataset. First, the targets $\mathbf{y}$ are history-dependent, so estimating a new state $\mathbf{y}'$ requires to know the sequence of states needed to reach that state, i.e., regression requires recurrent neural network architectures [34, 35], specifically adopting a Gated Recurrent Unit (GRU) architecture [81, 82] in this work. Furthermore, the dataset was created synthetically by physically-accurate computer simulations of materials under mechanical deformation, so it was possible to generate enough data to determine the ground-truth aleatoric uncertainty (arising from variations within the material). In other words, we have a good estimate of the heteroscedastic noise in the data.

C

APPENDICES FOR CHAPTER 4

C.1. KRR-LR-GPR PARAMETER ESTIMATION AND PREDICTION

The parameter estimation of KRR-LR-GPR involves two stages, the first stage is to determine parameters for LF kernel regression, and the second is to tune parameters for the transfer-learning model and residual GPR.

LF-KRR PARAMETER ESTIMATION

KRR based on the LF dataset can be defined by

$$\mathbf{y}^l = \mathbf{K}(\mathbf{X}^l, \mathbf{X}^l)\mathbf{w} \tag{C.1}$$

where $\mathbf{w}$ are the weights determined by LF data points, $\mathbf{K}(\mathbf{X}^l, \mathbf{X}^l)$ is a $N^l \times N^l$ Gram or covariance matrix with $\mathbf{K}_{ij} = k(\mathbf{x}_i^l, \mathbf{x}_j^l)$. The value of $\mathbf{w}$ can be determined analytically:

$$\mathbf{w} = \mathbf{K}(\mathbf{X}^l, \mathbf{X}^l)^{-1}\mathbf{y}^l \tag{C.2}$$

If the data has noise, $\mathbf{w} = \left(\mathbf{K}(\mathbf{X}^l, \mathbf{X}^l) + \lambda \mathbf{I}\right)^{-1}\mathbf{y}^l$ where λ is the LF noise precision.

TRANSFER-LEARNING MODEL AND RESIDUAL GPR PARAMETER ESTIMATION

Regarding the parameter estimation of the transfer-learning model and residual GPR, it follows the same procedure as Section 2.3.2, where the log marginal likelihood function can be formulated as:

$$\ln L\left(\mathbf{y}^h | \boldsymbol{\rho}, \boldsymbol{\theta}^h\right) = -\frac{N^h}{2}\ln(2\pi) - \frac{1}{2}\ln|\mathbf{K}(\mathbf{X}^h, \mathbf{X}^h)| - \frac{(\mathbf{y}^h - \mathbf{m}(\mathbf{X}^h)^T\boldsymbol{\rho})^T\mathbf{K}(\mathbf{X}^h, \mathbf{X}^h)^{-1}(\mathbf{y}^h - \mathbf{m}(\mathbf{X}^h)^T\boldsymbol{\rho})}{2} \tag{C.3}$$

By taking the derivative of Equation (C.3) and setting it to zero, $\hat{\boldsymbol{\rho}}$ can be written by:

$$\hat{\boldsymbol{\rho}} = \left(\mathbf{m}(\mathbf{X}^h)\mathbf{K}(\mathbf{X}^h, \mathbf{X}^h)^{-1}\mathbf{m}(\mathbf{X}^h)^T\right)^{-1}\mathbf{m}(\mathbf{X}^h)\mathbf{K}(\mathbf{X}^h, \mathbf{X}^h)^{-1}\mathbf{y}^h \tag{C.4}$$

Then, optimizing the ln-concentrated function for optimal $\boldsymbol{\theta}^h$

$$\ln L\left(\mathbf{y}^h|\boldsymbol{\theta}^h\right) = -\frac{1}{2}\ln|\mathbf{K}(\mathbf{X}^h,\mathbf{X}^h)| - \frac{(\mathbf{y}^h - \mathbf{m}(\mathbf{X}^h)^T\hat{\boldsymbol{\rho}})^T\mathbf{K}(\mathbf{X}^h,\mathbf{X}^h)^{-1}(\mathbf{y}^h - \mathbf{m}(\mathbf{X})^T\hat{\boldsymbol{\rho}})}{2} \tag{C.5}$$

PREDICTIONS

We can obtain the predictions of KRR-LR-GPR by modifying Equation (2.31) into:

$$\hat{f}^h(\mathbf{x}') = \mathbf{m}(\mathbf{x}')^T\hat{\boldsymbol{\rho}} + \mathbf{k}(\mathbf{x}',\mathbf{X}^h)\mathbf{K}(\mathbf{X}^h,\mathbf{X}^h)^{-1}(\mathbf{y}^h - \mathbf{m}(\mathbf{X}^h)^T\hat{\boldsymbol{\rho}}) \tag{C.6}$$

$$\hat{\sigma}_h^2(\mathbf{x}') = k(\mathbf{x}',\mathbf{x}') - \mathbf{k}\left(\mathbf{x}',\mathbf{X}^h\right)\mathbf{K}\left(\mathbf{X}^h,\mathbf{X}^h\right)^{-1}\mathbf{k}\left(\mathbf{X}^h,\mathbf{x}'\right) + \mathbf{r}^T(\mathbf{m}(\mathbf{X}^h)\mathbf{K}(\mathbf{X}^h,\mathbf{X}^h)^{-1}\mathbf{m}(\mathbf{X}^h)^T)^{-1}\mathbf{r} \tag{C.7}$$

where $\mathbf{r} = \mathbf{m}(\mathbf{x}') - \mathbf{m}(\mathbf{X}^h)\mathbf{K}(\mathbf{X}^h,\mathbf{X}^h)^{-1}\mathbf{k}(\mathbf{X}^h,\mathbf{x}')$.

C.2. NUMERICAL FUNCTIONS

NUMERICAL FUNCTIONS FOR TESTING KRR-LR-GPR

This section includes details on the low-dimensional numerical functions considered in this chapter. Originally, those functions were single-fidelity functions used to verify optimization algorithms [266]. They were extended by Jiang et al [267], Sander et al. [268] and Mainini et al. [269] for verifying the performance of MF optimization algorithms.

- Forrester function

$$f^h(x) = (6x-2)^2\sin(12x-4) \tag{C.8}$$

$$f^l(x) = A(6x-2)^2\sin(12x-4) + B(x-0.5) - C \tag{C.9}$$

where $x \in [0,1]$. By selecting different values of A, B, and C, LF functions with different correlations to the HF function.

- Hartman3 function

$$f^h(\mathbf{x}) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right) \tag{C.10}$$

$$f^l(\mathbf{x}) = 0.585 - 0.324x_1 - 0.379x_2 - 0.431x_3 \tag{C.11}$$

$$-0.208x_1x_2 + 0.326x_1x_3 + 0.193x_2x_3 + 0.225x_1^2 + 0.263x_2^2 + 0.274x_3^2 \tag{C.12}$$

where

$$\mathbf{c} = \begin{bmatrix} 1 \\ 1.2 \\ 3 \\ 3.2 \end{bmatrix}, \quad \mathbf{p} = \begin{bmatrix} 0.3689 & 0.117 & 0.2673 \\ 0.4699 & 0.4387 & 0.747 \\ 0.1091 & 0.8732 & 0.5547 \\ 0.03815 & 0.5743 & 0.8828 \end{bmatrix}, \quad \mathbf{a} = \begin{bmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 30 \\ 0.1 & 10 & 35 \end{bmatrix}$$

where $\mathbf{x} \in [0,1]$

- Hartman6 function

$$f^h(\mathbf{x}) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right) \tag{C.13}$$

$$f^l(\mathbf{x}) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij}(l_j x_j - p_{ij})^2\right) \tag{C.14}$$

where

$$\mathbf{p} = \begin{bmatrix} 0.1312 & 0.1696 & 0.5569 & 0.0124 & 0.8283 & 0.5886 \\ 0.2329 & 0.4135 & 0.8307 & 0.3736 & 0.1004 & 0.9991 \\ 0.2348 & 0.1451 & 0.3522 & 0.2883 & 0.3047 & 0.6650 \\ 0.4047 & 0.8828 & 0.8732 & 0.5743 & 0.1091 & 0.0381 \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} 1.0 \\ 1.2 \\ 3.0 \\ 3.2 \end{bmatrix},$$

$$\mathbf{a} = \begin{bmatrix} 10.00 & 3.0 & 17.00 & 3.5 & 1.7 & 8 \\ 0.05 & 10.0 & 17.00 & 0.1 & 8.0 & 14 \\ 3.00 & 3.5 & 1.70 & 10.0 & 17.0 & 8 \\ 17.00 & 8.0 & 0.05 & 10.0 & 0.1 & 14 \end{bmatrix}, \quad \mathbf{l} = \begin{bmatrix} 0.75 \\ 1.0 \\ 0.8 \\ 1.3 \\ 0.7 \\ 1.1 \end{bmatrix}$$

where $\mathbf{x} \in [0,1]$

- Six-hump function

$$f^h(\mathbf{x}) = (4 - 2.1x_1^2 + \frac{x_1^4}{3})x_1^2 + x_1x_2 - 4x_2^2 \tag{C.15}$$

$$f^l(\mathbf{x}) = f^h(0.7\mathbf{x}) - x_1x_2 - 15 \tag{C.16}$$

where $\mathbf{x} \in [-2,2]$

- Bohachevsky function

$$f^h(\mathbf{x}) = (x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7)^2 \tag{C.17}$$

$$f^l(\mathbf{x}) = f^h(0.7x_1, x_2) + x_1x_2 - 12 \tag{C.18}$$

where $\mathbf{x} \in [-5,5]$

- Booth function

$$f^h(\mathbf{x}) = (x_1^2 + 2x_2^2 - 7)^2 + (2x_1^2 + x_2 - 5)^2 \tag{C.19}$$

$$f^l(\mathbf{x}) = f_h(0.4x_1, x_2) + 1.7 \cdot x_1x_2 - x_1 + 2x_2 \tag{C.20}$$

where $\mathbf{x} \in [-10,10]$

C

- Borehole function

$$f^b(\mathbf{x}, A, B) = \frac{AT_u(H_u - H_l)}{\log\left(\frac{r}{r_w}\right)\left(1 + \frac{2LT_u}{\log\left(\frac{r}{r_w}\right)r_w^2 K_w} + \frac{T_u}{T_l}\right)} \tag{C.21}$$

$$f^h(\mathbf{x}) = f^b(\mathbf{x}, 2\pi, 1) \tag{C.22}$$

$$f^l(\mathbf{x}) = f^b(\mathbf{x}, 5, 1.5) \tag{C.23}$$

where

$$\begin{aligned} rw &\in [0.05, 0.15] \\ r &\in [100, 50000] \\ Tu &\in [63070, 115600] \\ Hu &\in [990, 1110] \\ Tl &\in [63.1, 116] \\ Hl &\in [700, 820] \\ L &\in [1120, 1680] \\ Kw &\in [9855, 12045] \end{aligned}$$

- CurrinExp function

$$f^h(x_1, x_2) = (1 - \exp(-1/(2x_2))) \times \frac{2300x_1^3 + 1900x_1^2 + 2092x_1 + 60}{100x_1^3 + 500x_1^2 + 4x_1 + 20} \tag{C.24}$$

$$\begin{aligned} f^l(x_1, x_2) = &\frac{1}{4}\left[f^h(x_1 + 0.05, x_2 + 0.05) + f^h(x_1 + 0.05, x_2 - 0.05)\right] \\ &+ \frac{1}{4}\left[f^h(x_1 - 0.05, x_2 + 0.05) + f^h(x_1 - 0.05, x_2 - 0.05)\right] \end{aligned} \tag{C.25}$$

where $\mathbf{x} \in [0, 1]$

- Park91A function

$$f^h(\mathbf{x}) = \frac{x_1}{2}\left(\sqrt{1 + (x_2 + x_3^2)\frac{x_4}{x_1^2}} - 1\right) + (x_1 + 3x_4)e^{1+\sin(x_3)} \tag{C.26}$$

$$f^l(\mathbf{x}) = (1 + \sin(x_1)/10)f^h(\mathbf{x}) - 2x_1 + x_2^2 + x_3^2 + 0.5 \tag{C.27}$$

where $\mathbf{x} \in [0, 1]$

- Park91B function

$$f^h(\mathbf{x}) = \frac{2}{3}e^{x_1 + x_2} - x_4\sin(x_3) + x_3 \tag{C.28}$$

$$f^l(\mathbf{x}) = 1.2f^h(\mathbf{x}) - 1 \tag{C.29}$$

where $\mathbf{x} \in [0, 1]$

NUMERICAL FUNCTIONS FOR TESTING DNN-LR-BNN

- 1D illustrative function [90]

$$f^h(\mathbf{x}) = (x - \sqrt{2})\sin(8\pi x)^2 \tag{C.30a}$$
$$f^{l1}(\mathbf{x}) = \sin(8\pi x) \tag{C.30b}$$
$$f^{l2}(\mathbf{x}) = 1.2 f^h(\mathbf{x}) - 0.5 \tag{C.30c}$$
$$f^{l3}(\mathbf{x}) = \sin(16\pi x)^2 \tag{C.30d}$$

  where $x \in [0,1]$. In the illustrative case, both LF and HF data are assumed to be corrupted by white noise with $\mathcal{N}(0, 0.05^2)$.

- 4D function [90]

$$f^h(\mathbf{x}) = \frac{1}{2}(0.1\exp(x_1 + x_2) - x_4\sin(12\pi x_3) + x_3) \tag{C.31}$$
$$f^l(\mathbf{x}) = 1.2 f^h(\mathbf{x}) - 0.5 \tag{C.32}$$

  where $x \in [0,1]$. The LF has noise with $\mathcal{N}(0, 0.05^2)$ and HF has noise with $\mathcal{N}(0, 0.01^2)$

- high dimensional function

  The high dimensional function used in [116] is adopted, which can be expressed by

$$f^h(\mathbf{x}) = \sum_{i=2}^{20}\left(2x_i^2 - x_{i-1}\right)^2 + (x_1 - 1)^2 \tag{C.33}$$
$$f^l(\mathbf{x}) = 0.8 f^h(\mathbf{x}) + \sum_{i=2}^{20} 0.4 x_{i-1}x_i - 50 \tag{C.34}$$

  where $x \in [-3,3]$. We set the dimension in [20, 50, 100] to challenge DNN-LR-BNN in high-dimensional problems. Moreover, this function is assumed to have zero noise in LF and HF has noise with $\mathcal{N}(0, 50^2)$.

C.3. ADDITIONAL EXPERIMENTS FOR KRR-LR-GPR

ADDITIONAL RESULTS OF KRR-LR-GPR COMPREHENSIVE EXPERIMENTS

In Section 4.4.1, we only show the results of the *Booth* function, and the remaining comparison results are shown in Figure C.1 and Figure C.2.

It can be observed that the proposed KRR-LR-GPR method achieves performance comparable to other state-of-the-art approaches in both scenarios presented in Figure C.1 and Figure C.2. Moreover, its training time for LF data is significantly lower than that of other methods. These results highlight the potential of KRR-LR-GPR in advancing data-scarce modeling by leveraging larger LF datasets, particularly in machine learning modeling scenarios where computational cost is dominated by LF data. On the other hand, when both LF and HF data are limited, KRR-LR-GPR remains a reliable and competitive choice as demonstrated in Section 4.4.1.

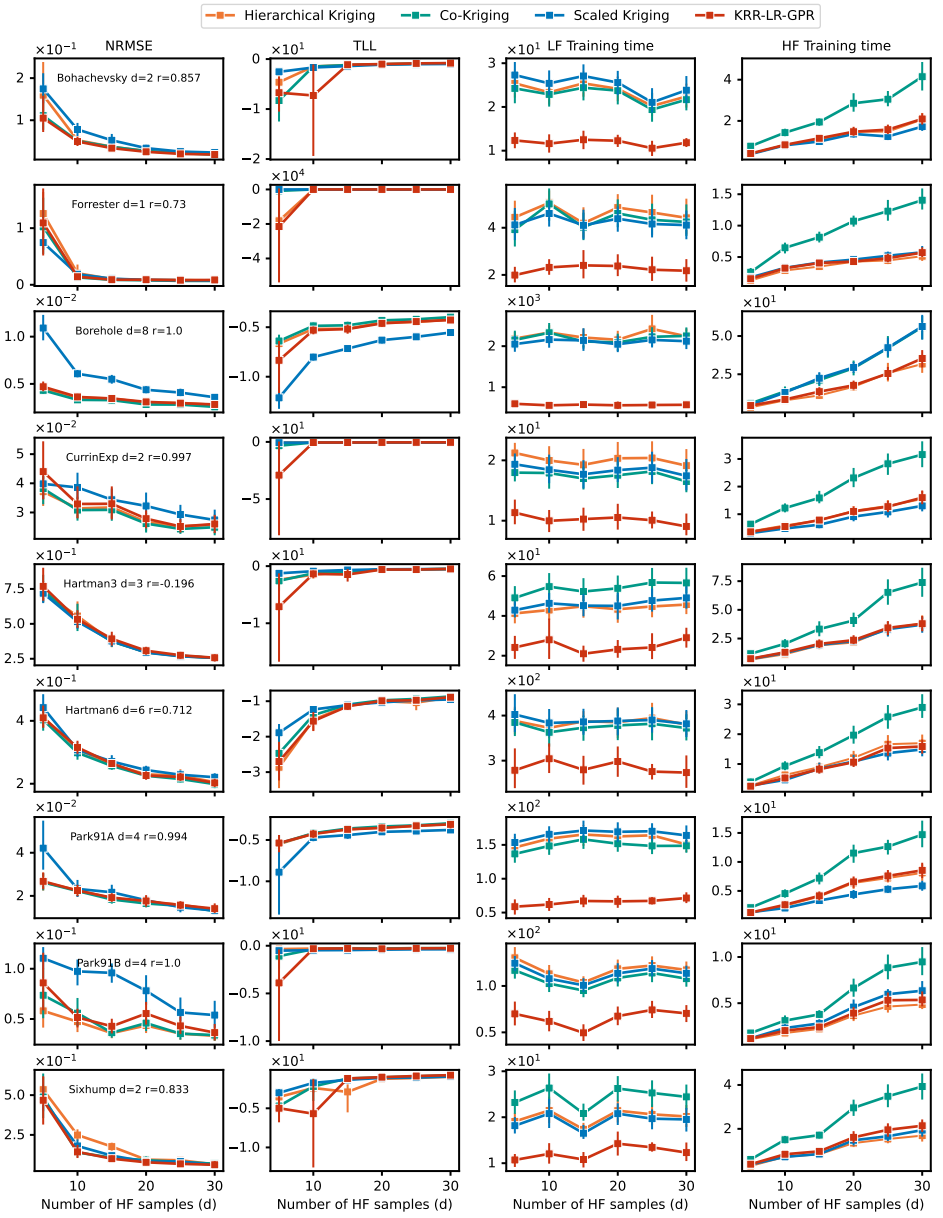


Figure C.1: Additional comprehensive experiment results of KRR-LR-GPR with different amounts of HF data. Different colors represent different methods, where NRMSE is the smaller the better and TLL is the larger the better.

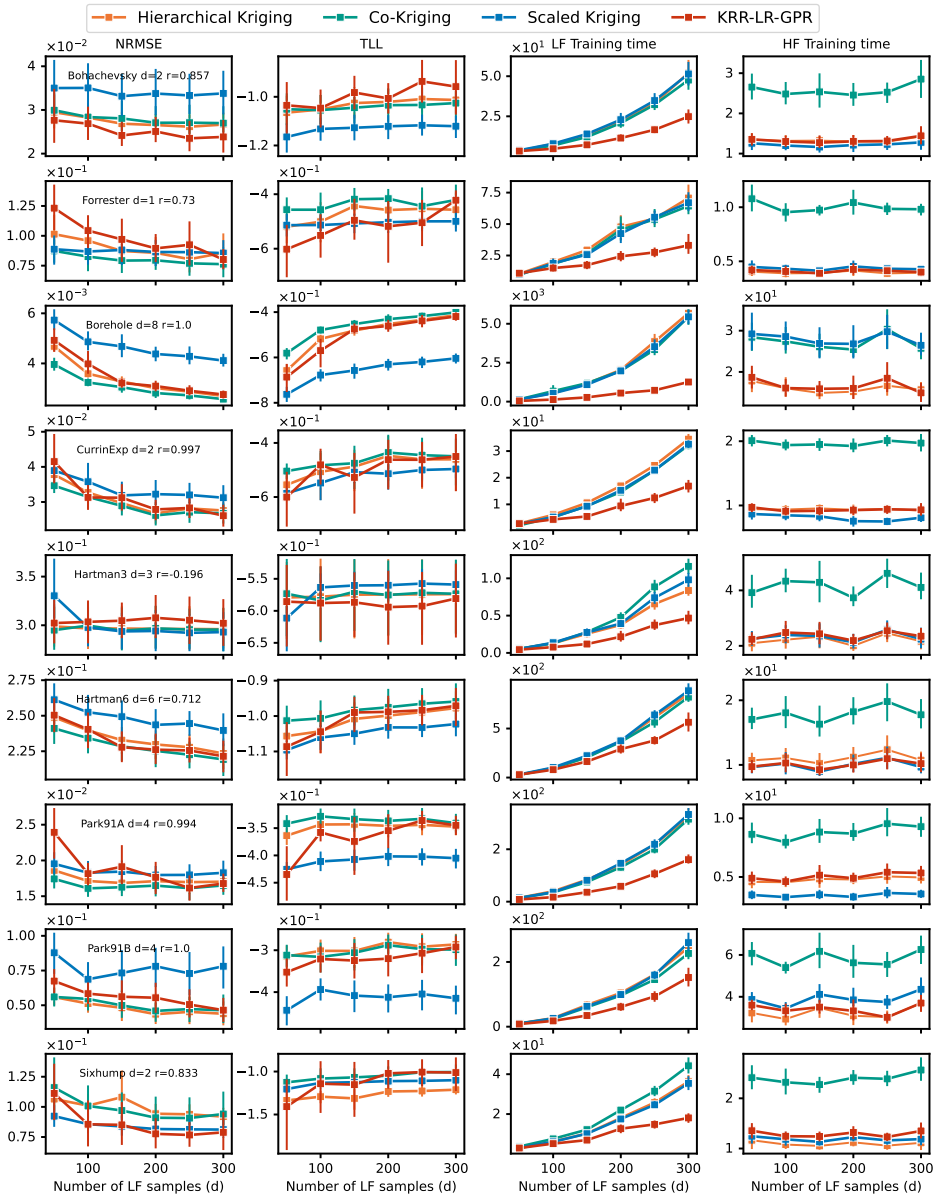


Figure C.2: Additional comprehensive experiment results of KRR-LR-GPR with different amounts of LF data. Different colors represent different methods, where NRMSE is the smaller the better and TLL is the larger the better.

EXPERIMENTS WITH NOISELESS DATA-SCARCE PROBLEMS: KRR-LR-GPR

As reported in the literature [105] and to the best of our knowledge, the existing MF-GPR methods have been considered for noiseless problems, which is a special case in this paper where $\sigma_h = 0$ and $\sigma_l = 0$. To demonstrate the effectiveness of the KRR-LR-GPR in those cases, we present the illustrative example based on the typical MF case presented by Forrester [97], the predicted performance of KRR-LR-GPR compared to other MF-GPR approaches is presented in Figure C.3 and different performance metrics for this problem are included in Table C.1.

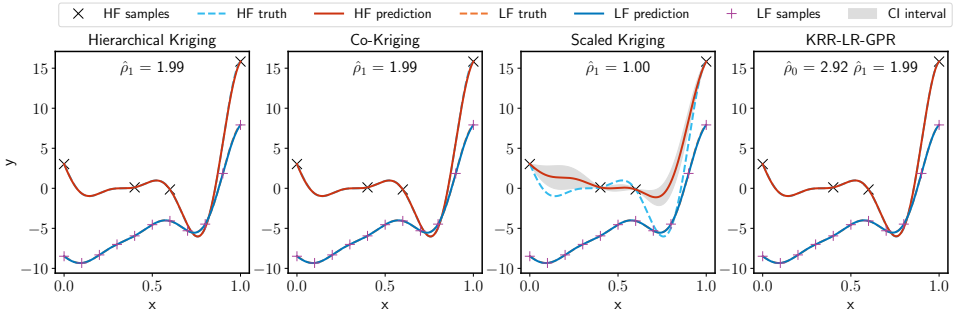


Figure C.3: Fitting performance of MF models on noiseless illustrative example under 5 independent runs of random seeds

Table C.1: Results comparison of the noiseless illustrative example with 5 independent runs

Methods	NRMSE	R2 Score	Training time (s)	
			LF	HF
Hierarchical Kriging	0.0228 ($\pm 1.49 \times 10^{-5}$)	0.9998 ($\pm 2.07 \times 10^{-7}$)	0.2673 ($\pm 6.05 \times 10^{-2}$)	**0.0426** ($\pm 6.67 \times 10^{-3}$)
Co-Kriging	0.0228 ($\pm 1.49 \times 10^{-5}$)	0.9998 ($\pm 2.06 \times 10^{-7}$)	0.2963 ($\pm 3.80 \times 10^{-2}$)	0.4214 ($\pm 7.25 \times 10^{-2}$)
Scaled Kriging	0.9754 ($\pm 1.78 \times 10^{-4}$)	0.6792 ($\pm 6.99 \times 10^{-5}$)	0.2934 ($\pm 3.54 \times 10^{-2}$)	0.0886 ($\pm 4.29 \times 10^{-4}$)
KRR-LR-GPR	**0.0072** ($\pm 2.44 \times 10^{-6}$)	**1.0000** ($\pm 1.00 \times 10^{-8}$)	**0.0003** ($\pm 5.51 \times 10^{-5}$)	0.0569 ($\pm 9.00 \times 10^{-3}$)

We can observe that the findings are the same as the results reported in Figure 4.3 and Table 4.2. KRR-LR-GPR has a slightly better performance compared with other MF approaches. Meanwhile, we can see that KRR-LR-GPR is faster to train compared with other methods.

C.4. ABLATION STUDIES OF KRR-LR-GPR

INFLUENCE OF LF POLYNOMIAL ORDER

We provide an opportunity to select different polynomial orders of the LF surrogate in Equation (4.5) and Figure 4.1. We conducted an ablation study on how the selection of $\mathbf{m}(\mathbf{x})$ would influence the performance of the KRR-LR-GPR[1]. Figure C.4 shows the fitting performance on an illustrative example, and Table C.2 summarizes the corresponding accuracy metric values.

[1]To eliminate the influence of noise, the experiment is conducted with datasets without noise.

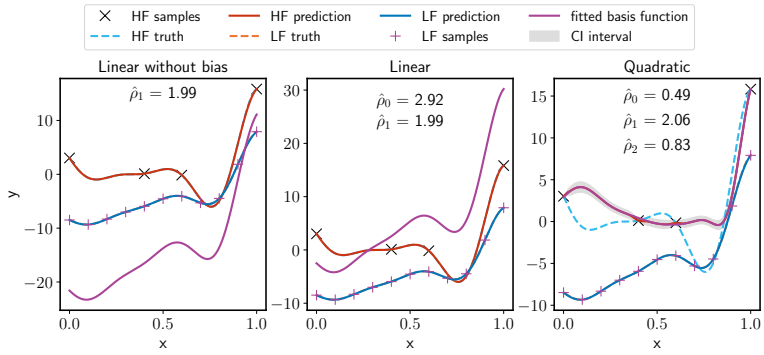


Figure C.4: Performance of KRR-LR-GPR based on different LF polynomial orders

Table C.2: Results comparison of KRR-LR-GPR based on different LF polynomial orders

Methods	NRMSE	R2 Score	Training time (s)	
			LF	HF
Linear without bias	0.0073 ($\pm 2.55 \times 10^{-6}$)	0.9999 ($\pm 9.74 \times 10^{-9}$)	0.0002 ($\pm 2.19 \times 10^{-4}$)	0.0417 ($\pm 5.13 \times 10^{-3}$)
Linear	**0.0072** ($\pm 2.55 \times 10^{-6}$)	**1.0000** ($\pm 9.76 \times 10^{-9}$)	0.0002 ($\pm 7.45 \times 10^{-5}$)	0.0342 ($\pm 8.93 \times 10^{-3}$)
Quadratic	1.1358 ($\pm 1.79 \times 10^{-4}$)	0.5651 ($\pm 8.73 \times 10^{-5}$)	0.0002 ($\pm 3.11 \times 10^{-5}$)	0.0450 ($\pm 1.11 \times 10^{-2}$)

With quadratic LF features, the KRR-LR-GPR overfits the Forrester function, though $\mathbf{m}(\mathbf{x})\boldsymbol{\rho}$ have a small residual to $\mathbf{y}^h$. On the other hand, KRR-LR-GPR can fit the Forrester function well with the linear LF feature, where we observe that the bias does influence the value of ρ_1, but shifts the entire LF surrogate closer to the HF data. As listed in Table C.2 the bias term ρ_0 also improves accuracy. To gain a deeper understanding of the effect of the LF polynomial order, Figure C.5 gives the NRMSE changes as the HF sample increases on the set of test examples.

As shown in Figure C.5, KRR-LR-GPR with quadratic LF polynomial order has large errors, leading to overfitting in most cases when the number of LF samples is smaller. The cases with linear polynomial order alleviate this issue to some extent. Regarding the effect of the bias term ρ_0, it does not influence most of the cases, whereas it brings improvements in some cases like *Borehole* and *Park91A*. Moreover, the bias term has slight positive effects on KRR-LR-GPR when LF samples are less in most cases. To this end, we adopt $\mathbf{m}(\mathbf{x}) = [1, f^l(\mathbf{x})]^T$ for KRR-LR-GPR in this paper.

INFLUENCE OF HF-TO-LF CORRELATION AND HF NOISE LEVEL

KRR-LR-GPR converges to good accuracy with more HF data as shown in Section 4.4.1, while differences exist among different examples due to the HF-to-LF correlation, see Appendix C.3. We fix the noise level to be $\sigma_h = 0.3$ in Section 4.4.1. Therefore, we design the ablation study here to investigate the influence of the HF-to-LF correlation and the HF noise level on KRR-LR-GPR compared to GPR. Figure C.6 depicts the corresponding results.

We can see from Figure C.6 that the HF noise level has a significant impact on KRR-

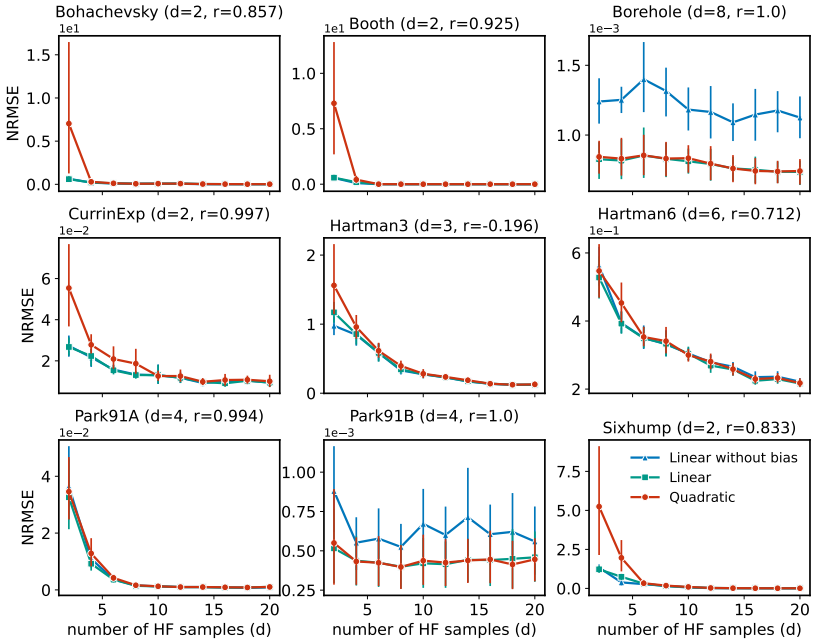


Figure C.5: NRMSE of MF-RBF-Kriging based on different LF polynomial orders (LF data $100 \times d$)

LR-GPR, where larger HF noise levels would deteriorate the performance even with an adequate amount of HF data. Specifically, we provide three noise levels for each test function that are $\sigma_h \in [0.1, 0.3, 0.5]$. It is clear to see that the accuracy metrics are better in the case where the noise is smaller for each test function. In addition, higher HF-to-LF correlations are preferred when utilizing KRR-LR-GPR, where clear advantages are observed on both NRMSE and TLL such as in the cases of *Bohachevsky*, *Booth*, *Borehole*, *CurrinExp*, *Hartman6*, *Park91A*, *Park91B*, and the *Sixhump* functions. Conversely, poor HF-to-LF correlation decreases the performance of KRR-LR-GPR, for instance, there is margin improvement over GPR in the case of *Hartman3* function where the Pearson correlation coefficient between HF and LF is -0.196.

INFLUENCE OF HF AND LF SAMPLES

Section 4.4.1 included two experiments using a fixed number of HF or LF samples and explored the effect of another factor. To have a better overview of how HF and LF samples influence the KRR-LR-GPR method, Figure C.7 [2] shows the whole design of experiments instead of slices shown in Figure 4.7 and 4.8.

In Figure C.7, the R2 Score is primarily influenced by the number of HF samples, and the LF samples will improve the robustness of KRR-LR-GPR. The training time plotted at the right heatmap is the total training time for both HF and LF models, more CPU time is needed with a greater total amount of data. However, the influence of HF data on execution time is more critical than that of LF data.

[2] R2 score is selected because it does not depend on the actual magnitude of the response.

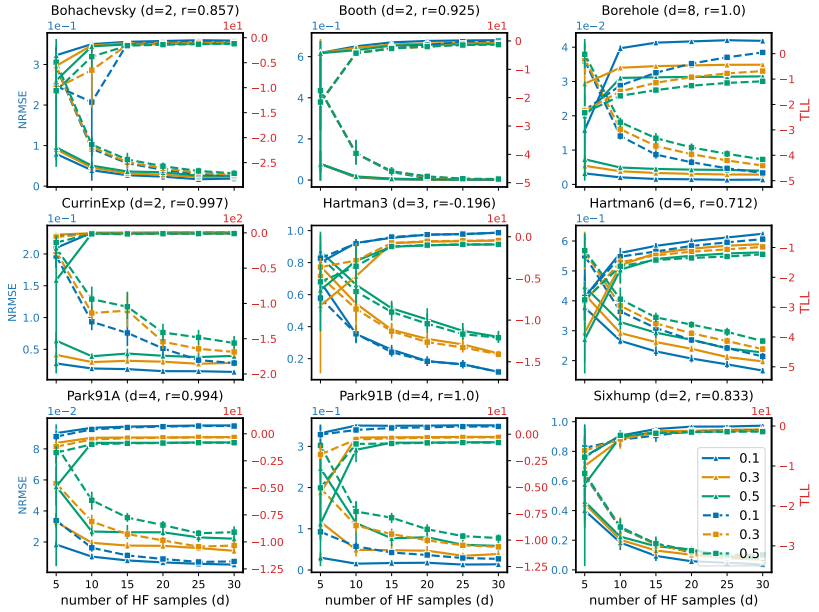


Figure C.6: NRMSE and TLL of different examples with increasing HF data (LF data $200 \times d$): The solid and dash lines represent KRR-LR-GPR and GPR respectively, and line color shows different HF noise levels

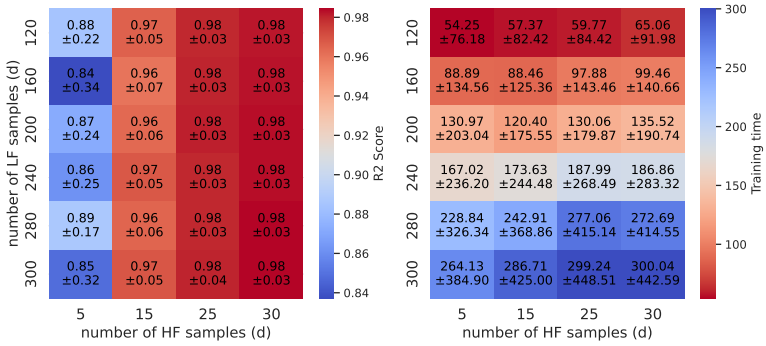


Figure C.7: Influence of HF and LF samples on KRR-LR-GPR. The heatmap aggregates all test functions listed in Section C.2 and every experiment is repeated 5 times by resampling from the design space independently

C.5. HYPERPARAMETERS OF EXPERIMENTS AND ADDITIONAL EXPERIMENTS ON DNN-LR-BNN

HYPERPARAMETERS OF THE HIGHER DIMENSIONAL EXPERIMENTS

These experiments were extracted from [90, 116, 117] for examples of 1D, 4D, and 20D, with the same hyperparameters except for the case of 20D and consider $1000 \times d$ and $100 \times d$ samples for LF and HF, respectively.

Table C.3: Hyperparameters for the DNN-LR-BNN experiments

Example	Hyperparameter Name		Hyperparameters setting			Samples
			BNN	DNN-BNN	DNN-LR-BNN	
1D	DNN	Architecture	-	2 hidden, 50 neurons, *Tanh*	2 hidden, 50 neurons, *Tanh*	201 LF 11 HF
		Learning Rate	-	10^{-3}	10^{-3}	
		Optimizer	-	Adam	Adam	
		Epoch	-	10000	10000	
	BNN	Architecture	2 hidden, 512 neurons, *Tanh*	2 hidden, 50 neurons, *Tanh*	2 hidden, 512 neurons, *Tanh*	
		Learning rate	0.001	0.001	0.001	
		Posterior collection	300 samples, Burn-in: 20000, Frequency: 100	300 samples, Burn-in: 20000, Frequency: 100	300 samples, Burn-in: 20000, Frequency: 100	
4D	DNN	Architecture	-	2 hidden, 256 neurons, *Tanh*	2 hidden, 256 neurons, *Tanh*	25000 LF 150 HF
		Learning Rate	-	10^{-3}	10^{-3}	
		Optimizer	-	Adam	Adam	
		Epochs	-	50000	50000	
	BNN	Architecture	2 hidden, 50 neurons , *Tanh*	2 hidden, 50 neurons , *Tanh*	2 hidden, 50 neurons , *Tanh*	
		Learning Rate	10^{-3}	10^{-3}	10^{-3}	
		Posterior Collection	300 samples, Burn-in: 20000, Frequency: 100	300 samples, Burn-in: 20000, Frequency: 100	300 samples, Burn-in: 20000, Frequency: 100	
20D	DNN	Architecture	-	2 hidden, 200 neurons, *Tanh*	2 hidden, 200 neurons, *Tanh*	30000 LF 5000 HF
		Learning Rate	-	10^{-3}	10^{-3}	
		Optimizer	-	Adam	Adam	
		Epochs	-	80000	80000	
	BNN	Architecture	2 hidden, 512 neurons , *ReLU*	2 hidden, 512 neurons , *ReLU*	2 hidden, 512 neurons , *ReLU*	
		Learning Rate	10^{-3}	10^{-3}	10^{-3}	
		Posterior Collection	300 samples, Burn-in: 20000, Frequency: 100	300 samples, Burn-in: 20000, Frequency: 100	300 samples, Burn-in: 20000, Frequency: 100	
50D	DNN	Architecture	-	2 hidden, 256 neurons, *Tanh*	2 hidden, 256 neurons, *Tanh*	50000 LF 5000 HF
		Learning Rate	-	10^{-3}	10^{-3}	
		Optimizer	-	Adam	Adam	
		Epochs	-	50000	50000	
	BNN	Architecture	2 hidden, 512 neurons , *ReLU*	2 hidden, 512 neurons , *ReLU*	2 hidden, 512 neurons , *ReLU*	
		Learning Rate	10^{-3}	10^{-3}	10^{-3}	
		Posterior Collection	400 samples, Burn-in: 10000, Frequency: 100	400 samples, Burn-in: 10000, Frequency: 100	400 samples, Burn-in: 10000, Frequency: 100	
100D	DNN	Architecture	-	2 hidden, 256 neurons, *Tanh*	2 hidden, 256 neurons, *Tanh*	100000 LF 10000 HF
		Learning Rate	-	10^{-3}	10^{-3}	
		Optimizer	-	Adam	Adam	
		Epochs	-	50000	50000	
	BNN	Architecture	2 hidden, 512 neurons , *ReLU*	2 hidden, 512 neurons , *ReLU*	2 hidden, 512 neurons , *ReLU*	
		Learning Rate	10^{-3}	10^{-3}	10^{-3}	
		Posterior Collection	400 samples, Burn-in: 10000, Frequency: 100	300 samples, Burn-in: 10000, Frequency: 100	400 samples, Burn-in: 10000, Frequency: 100	

HYPERPARAMETERS OF THE MATERIAL STRUCTURE-PROPERTY PROBLEM

For the DNN settings, both DNN-BNN and DNN-LR-BNN use an architecture with two hidden layers of 256 neurons and ReLU activation. The learning rate is set to 10^{-3}, and the optimizer used is Adam. Both models are trained for 5000 epochs. For the BNN settings, BNN employs an architecture with two hidden layers of 256 neurons and ReLU activation, The learning rate for all models is 0.001. Posterior collection involves 100 data points with a burn-in of 10,000 iterations and a frequency of 100. The results presented in Appendix C.5 follow the same settings, except that BNN uses a two-layer architecture with 100 neurons per layer.

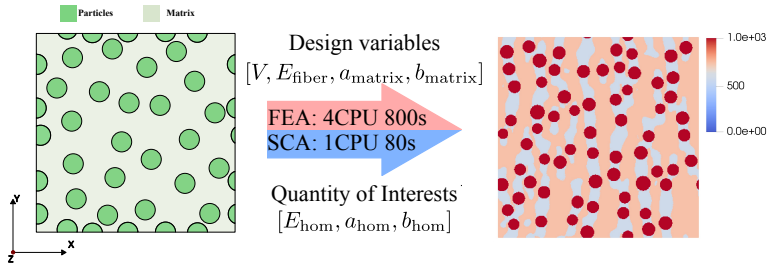


Figure C.8: Illustration of the simulation process for the 2-phase material SVE. The left figure presents a schematic representation of a 2-phase microstructure consisting of matrix and particle materials. For any given set of input variables, the simulation can be performed using either FEA or SCA, yielding the corresponding strain and stress fields. In the right figure, we depict the stress field for an arbitrary input: $V = 0.2$, $E_{\text{fiber}} = 400000$, $a_{\text{matrix}} = 400$, and $b_{\text{matrix}} = 0.3$. Based on the simulation results, we post-process the strain and stress fields to extract the QoIs, with further details provided in Figure C.9.

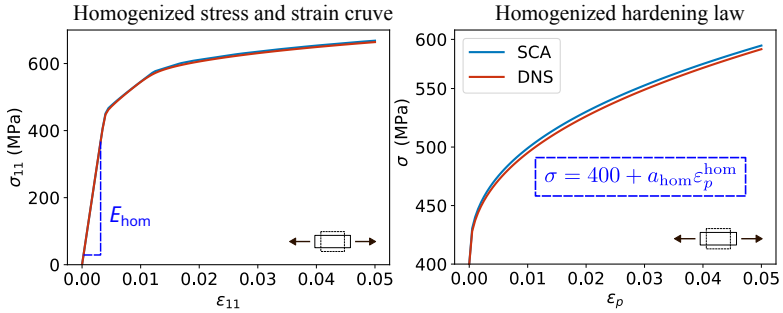


Figure C.9: Comparison between the homogenized properties of FEA and SCA. The left figure shows the homogenized strain and stress curve along the loading direction(11 direction from Mechanics of Material) and from this curve, we can obtain E_{hom} which is depicted slope. The right figure gives the homogenized hardening law which describes the relation between accumulated plastic strain ε_p and Von Mises stress σ. From which, we can get a_{hom} and b_{hom} with a simple curve fit.

The dataset consists of 3200 LF data points and 500 HF data points. For single-fidelity BNN training, 400 HF data points are utilized; and the MF methods employ 3200 LF and 100 HF data points. We claim that both single-fidelity and MF methods have the same computational cost for data generation in total.

Regarding the simulation setups, the design variables given in Section 4.4.2 which are the $[V, E_{\text{fiber}}, a_{\text{matrix}}, b_{\text{matrix}}]$. Other simulation parameters include a Poisson's ratio of 0.25 for the elastic fiber and 0.30 for the plastic matrix. Young's modulus of the matrix is set to 100000 MPa, and the yield stress in the matrix hardening law is 400 MPa. The SVE is discretized into 300 elements per edge, resulting in 90,000 structural elements. We give an illustration of the simulation in Figure C.8 where the design variables have been passed into the simulation and corresponding strain and stress fields could be obtained through FEA or SCA. Based on the stain/stress fields of the simulation. we then post-process and obtain the homogenized QoIs as shown in Figure C.9. SCA serves as an effective simplification, with both curves in Figure C.9 closely matching those obtained from FEA. Moreover, SCA significantly accelerates the simulation, reducing CPU execution time to approximately $\frac{1}{32}$ of that required by FEA.

INFLUENCE OF NEURAL ARCHITECTURE

Illustrative example We know that hyperparameters are crucial to DNNs. Therefore, we conduct a brief study on the influence of neural architecture on the performance of the developed DNN-LR-BNN method based on the illustrative example in Section 4.4.2. First of all, we show that network architecture has a limited impact on the predicted mean but influences the predicted uncertainty significantly as highlighted in Figure C.10. We show the shallow network would be overconfident, meaning a smaller uncertainty bound, while the uncertainty would increase and gradually converge to the case of GPR shown in Figure C.10. This is reasonable because GPR is equivalent to an infinitely wide neural network with a single hidden layer [21].

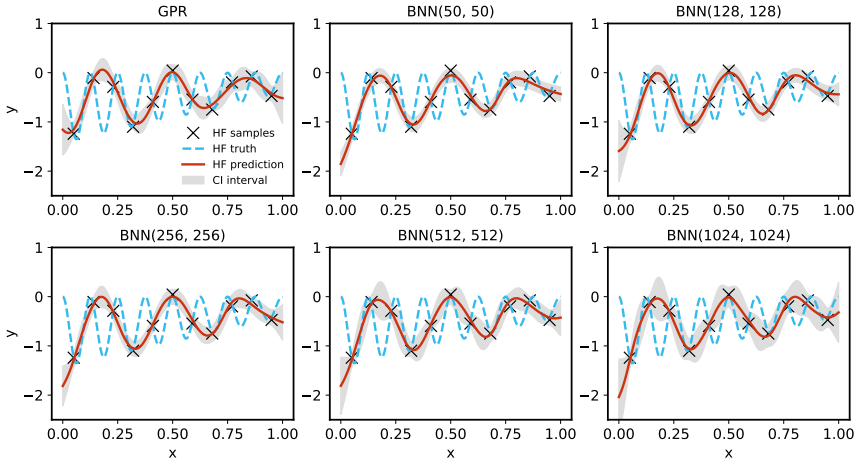


Figure C.10: Influence of network architecture on the performance of single-fidelity BNN. All the hyperparameters are fixed as Table C.3 except expanding the network from (50, 50) to (1024, 1024) neurons in each of the two hidden layers.

Secondly, we compare the corresponding influence of neural architecture on DNN-BNN and DNN-LR-BNN , and their results are depicted in C.11 amd Figure C.12. We observe the same pattern in the results of DNN-LR-BNN when enlarging the neural

network architecture. Interestingly, the learned $\boldsymbol{\rho}$ is also robust to the neural network architecture choice. However, the DNN-BNN is significantly sensitive to the neural network architecture, leading to reasonable results for small networks but significant overfitting for larger networks.

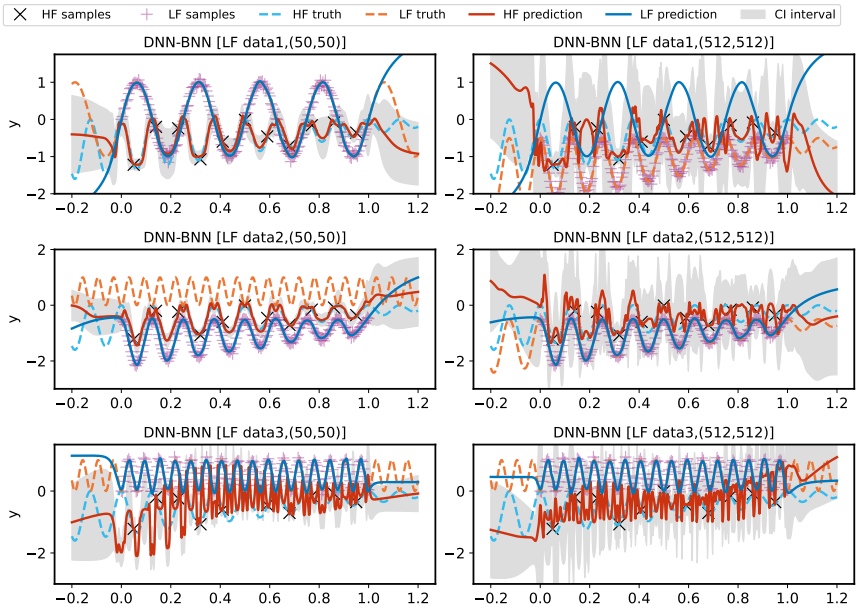


Figure C.11: Comparison of DNN-BNN considering two different architectures (left) using 50 neurons, and (right) using 512 neurons in each of the two hidden layers.

Material structure-property linkages problem As shown in Table 4.8 the DNN-BNN performs poorly when using a BNN architecture with 2 layers and 256 neurons. Therefore, we conduct another experiment where we employ a smaller architecture for BNN with 2 layers and 100 neurons, and the results are listed in Table C.4.

Table C.4: Results comparison of DNN-LR-BNN on the material structure-property linkages with 2-layer 100-neuron BNNs

QoIs	Methods	NRMSE	R2 Score	TLL	$\hat{\rho}_0$	$\hat{\rho}_1$	$\hat{\rho}_2$
a_{eff} ($r = 0.9294$)	DNN-BNN	0.0436	0.9569	-5.7493	-	-	
	DNN-LR^1-BNN	0.0603	0.9182	-5.7139	10.00	0.92	-
	DNN-LR^2-BNN	**0.0403**	**0.9633**	**-5.2613**	10.00	1.24	-3.56×10^{-4}
b_{eff} ($r = 0.7681$)	DNN-BNN	0.0115	0.9905	2.9890	-	-	
	DNN-LR^1-BNN	**0.0111**	0.9913	3.6212	0.10	0.88	-
	DNN-LR^2-BNN	0.0108	**0.9917**	3.5549	0.05	1.17	0.39
E_{eff} ($r = 0.9845$)	DNN-BNN	0.0153	0.9843	**-12.1512**	-	-	
	DNN-LR^1-BNN	**0.0107**	**0.9923**	-12.8826	-7.97	1.01	-
	DNN-LR^2-BNN	0.0130	0.9887	-12.3157	-5.24	0.99	1.09×10^{-7}

We observe that the performance of DNN-BNN improves significantly compared with those in Table 4.8. However, the proposed DNN-LR-BNN is still slightly better and the learned $\hat{\boldsymbol{\rho}}$ values are almost the same. The results demonstrate the robustness of the DNN-LR-BNN with different neuron architectures, which is an important feature for practical application.

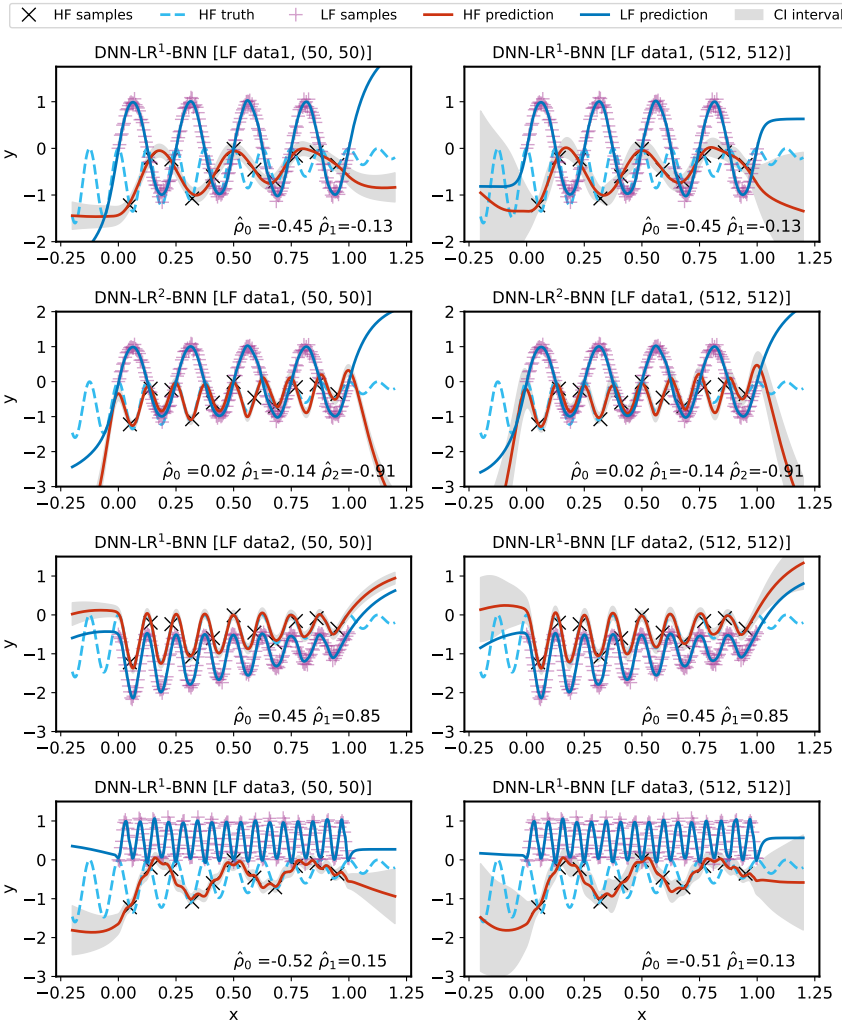


Figure C.12: Comparison of DNN-LR-BNN considering two different architectures (left) using 50 neurons, and (right) using 512 neurons in each of the two hidden layers.

D

APPENDICES FOR CHAPTER 5

D.1. ADDITIONAL INFORMATION ABOUT THE DATASETS

RANDOM POLYNOMIAL STRAIN-STRESS PATHS

We adopt the random polynomial strain path developed in our previous work [34, 35], whose generation process is briefly given by Figure D.1.

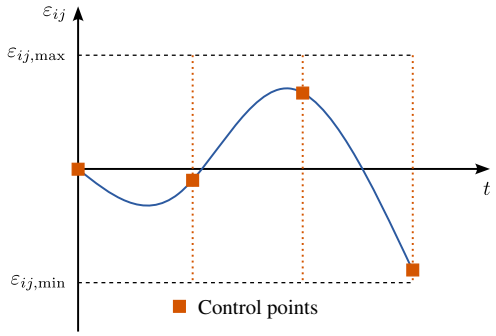


Figure D.1: Generation of the random polynomial strain paths.

As illustrated in Figure D.1, 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 Figure 5.2.

CONVERGENCE ANALYSIS TO DETERMINE THE SIZE OF AN RVE

By definition, a Representative Volume Element (RVE) should has negligible aleatoric uncertainty when randomizing the microstructure. In Figure D.2, when reducing the size

of the material volume element (Figure 5.2a), the response exhibits aleatoric uncertainty (noise), i.e., we have a Stochastic Volume Element (SVE) instead of an RVE.

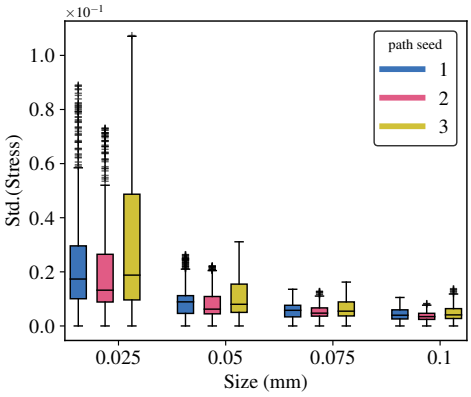


Figure D.2: Convergence analysis of the size of the microstructure volume element and the origin of aleatoric uncertainty (noise). Different colors represent different seeds for strain path generation. The size of the volume element varies from 0.025 mm to 0.10 mm, and we repeat the simulation 100 times to get the statistics.

We conduct a convergence analysis based on finite element analyses of different volume element sizes. The results are summarized in Figure D.2, 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 Figure D.2 that the stress variation decreases as the size of the microstructure unit cell increases, highlighting the fact that aleatoric 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 5.2.

ALEATORIC UNCERTAINTY

NOTE ON THE ALEATORIC UNCERTAINTY OF $\widetilde{\text{DNS}}$ (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 [270]. 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.

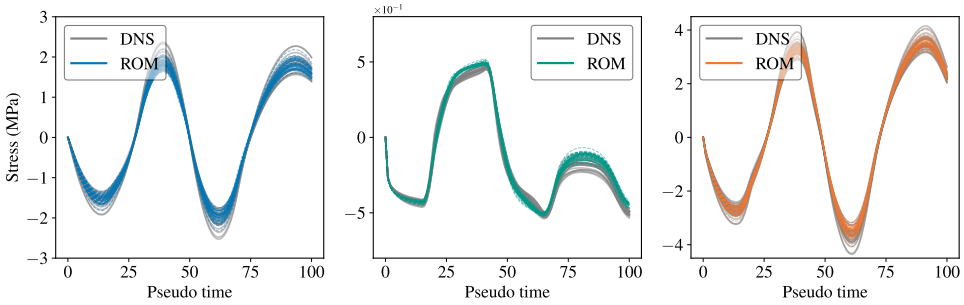
Specifically, the particles – occupying a 30% volume fraction – are modeled as circles with radii sampled from a normal distribution $\mathcal{N}(0.003, 0.001^2)$. Based on a convergence analysis presented in Appendix D.1, we set the RVE size to 0.1 mm with a realization shown in Figure 5.2a. Finite Element Analysis (FEA) is adopted as direct numerical simulation (DNS) [133], where the RVE is discretized into 400×400 elements. The homogenized stress response undergoing a prescribed random polynomial strain path (Figure 5.2b) is depicted as the solid lines in Figure 5.2c.

On the other hand, the SVE is defined with a domain size of 0.025 mm as illustrated by

the red window in Figure 5.2a. 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 Figure 5.2c, 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.

NOTE ON THE ALEATORIC UNCERTAINTY OF $\widetilde{\text{ROM}}$ (DATA OBTAINED FROM SCA OF AN SVE)

Figure D.3 compares the noisy results obtained from $\widetilde{\text{DNS}}$ (i.e., FEA of an SVE) with the noisy results obtained from $\widetilde{\text{ROM}}$ (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 Figure D.3 are obtained for the same strain path shown in Figure 5.2a.



(a) σ_{11} (b) σ_{12} (c) σ_{22}

Figure D.3: 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).

Figure D.3 shows that the $\widetilde{\text{ROM}}$ has higher aleatoric uncertainty than the $\widetilde{\text{DNS}}$. 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) [87, 271]. However, in this study, we focus on the total aleatoric uncertainty and do not further decompose its individual contributions.

BIAS ERROR INTRODUCED BY $\overline{\text{ROM}}$ COMPARED TO $\overline{\text{DNS}}$ (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 Figure 5.1. This acceleration is achieved by the self-consistent clustering analysis (SCA) method [87, 131, 271], a type of reduced order model (ROM) that we developed previously. We have shown

the accuracy and computational time comparison in Figure 5.3, and the quadratic scaling of SCA with the discretization level (number of clusters N_{cluster}).

In this appendix, we further analyze the differences between $\overline{\text{DNS}}$ and $\overline{\text{ROM}}$ when varying the N_{cluster} for the SCA method – see Figure D.4. 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 (Figure 5.3). 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.

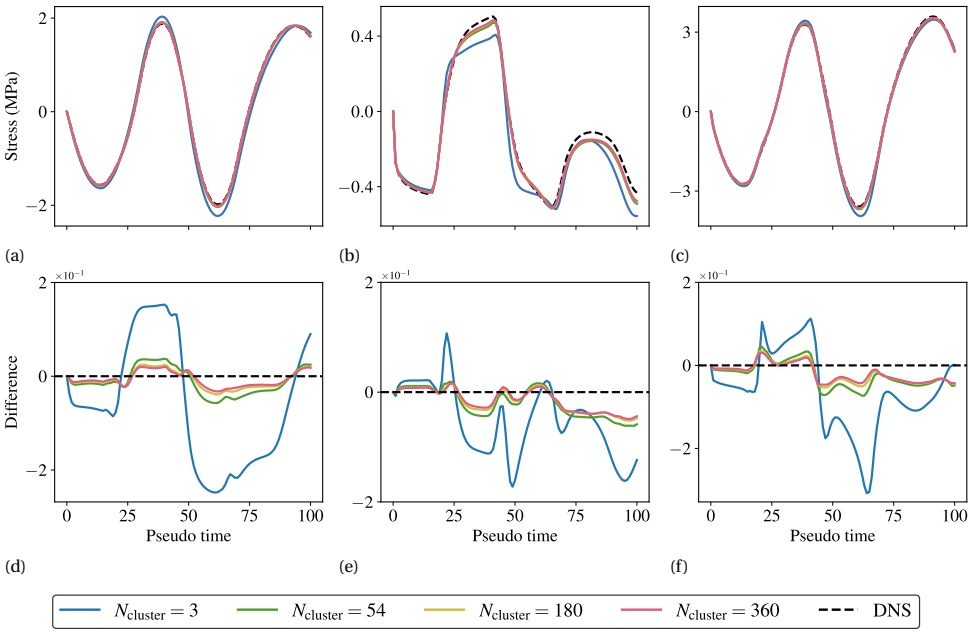


Figure D.4: Bias error introduced by **SCA-RVE** when compared to **DNS-RVE** and considering different numbers of clusters for the SCA method. The columns from left to right refer to the different stress components: σ_{11}, σ_{12}, and σ_{22}, respectively.

D.2. 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 [68], or embed it directly into the likelihood for end-to-end Bayesian inference [23]. We recently proposed a more efficient strategy to quantify aleatoric uncertainty [148]. 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 8.

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 MSE loss between the RNN outputs and the learning objective. As noted in Theorem 5.3.1, the specific inputs and outputs of the RNN may vary depending on the architecture, as summarized in Table D.1. For clarity, we denote the input as $\mathbf{x}$ and the output as $\mathbf{y}$. Without any loss of generality, we demonstrate the proposed framework by adopting a GRU model [34, 35] due to its simplicity and effectiveness, which is written as:

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

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

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

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

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* [46] to optimize the MSE loss between the decoded $f(\mathbf{x}_t)$ and $\mathbf{y}_t$, finding the point estimate of the learnable parameters $\boldsymbol{\theta} = [\boldsymbol{W}, \boldsymbol{b}]$. Thus, the loss function can be given explicitly as:

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

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

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 8. 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 Equation (5.2). 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 $\boldsymbol{r}_t = \big(f(\mathbf{x}_t;\boldsymbol{\theta}) - \mathbf{y}_t\big)^2$ follows a Gamma distribution since $\mathbf{y}_t$ is Gaussian. The corresponding Negative log-likelihood loss to train the variance network is defined by:

$$\mathscr{L}_{\text{Gamma}}(\boldsymbol{\phi}) = \sum_{n=1}^{N}\sum_{t=1}^{T}\Big[\log\Gamma\big(\alpha(\mathbf{x}_{n,t};\boldsymbol{\phi})\big) - \alpha(\mathbf{x}_{n,t};\boldsymbol{\phi})\log\lambda(\mathbf{x}_{n,t};\boldsymbol{\phi}) - \big(\alpha(\mathbf{x}_{n,t};\boldsymbol{\phi})-1\big)\log r_{n,t} + \lambda(\mathbf{x}_{n,t};\boldsymbol{\phi})\, r_{n,t}\Big]. \quad \text{(D.6)}$$

where $\alpha(\mathbf{x}_t;\boldsymbol{\phi}) > 0$ and $\lambda(\mathbf{x}_t;\boldsymbol{\phi}) > 0$ 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:

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

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 [1]. The posterior distribution of the parameters can be obtained using Bayes' rule, expressed by [272],

$$p(\boldsymbol{\theta}|\mathscr{D}) = \frac{p(\mathscr{D}|\boldsymbol{\theta})\,p(\boldsymbol{\theta})}{p(\mathscr{D})}; \quad p(\mathscr{D}) = \int p(\mathscr{D}|\boldsymbol{\theta})\,p(\boldsymbol{\theta})\,d\boldsymbol{\theta} \quad \text{(D.8)}$$

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

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

where $s^2(\mathbf{x};\boldsymbol{\phi})$ is the aleatoric (data) uncertainty (from **step 2**). Obtaining the posterior distribution in Equation (D.8) is difficult because of the intractable high-dimensional probability integral. Thus, approximate methods such as MCMC [45] or VI [45, 53] are required; we provide details in the Section 2.5. After obtaining the posterior distribution of the parameters in Equation (D.8), the predicted posterior distribution for any unknown point $\mathbf{x'}_{1:T}$ can be computed as

[1] 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

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

The predicted mean can be approximately obtained as

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

and predicted variance as

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

D

D.3. PERFORMANCE METRICS

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

- **Relative error (ϵ_r)**

$$\epsilon_r = \frac{1}{N_{\text{test}} T} \sum_{i=1}^{N_{\text{test}}} \sum_{t=1}^{T} \left(\frac{\left\| \bar{\mathbf{y}}_{i,t} - \hat{\mathbf{y}}_{i,t} \right\|_F}{\left\| \bar{\mathbf{y}}_{i,t} \right\|_F} \right) \cdot 100\%, \tag{D.13}$$

  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 i^{th} test sample and t^{th} time step, correspondingly.

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

$$\mathrm{TLL} = \frac{1}{N_{\text{test}} T} \sum_{i=1}^{N_{\text{test}}} \sum_{t=1}^{T} \log p(\bar{\mathbf{y}}_{i,t} | \boldsymbol{\theta}) \tag{D.14}$$

- **Wasserstein distance (WA)**

$$\mathrm{WA} = \frac{1}{N_{\text{test}} T} \sum_{i=1}^{N_{\text{test}}} \sum_{t=1}^{T} \mathrm{W}_2\left(p(\bar{\mathbf{y}}_{i,t} | \boldsymbol{\theta}), q(\bar{\mathbf{y}}_{i,t} | \boldsymbol{\theta})\right) \tag{D.15}$$

  where $p(\bar{\mathbf{y}}_{i,t} | \boldsymbol{\theta})$ and $q(\bar{\mathbf{y}}_{i,t} | \boldsymbol{\theta})$ are the predicted and ground truth distributions, respectively. If Gaussian assumptions are applied, we can rewrite Equation (D.15) into:

$$\mathrm{WA} = \frac{1}{N_{\text{test}} T} \sum_{i=1}^{N_{\text{test}}} \sum_{t=1}^{T} \sqrt{(\bar{\mathbf{y}}_{i,t} - \hat{\mathbf{y}}_{i,t})^2 + (\bar{\mathbf{s}}_{i,t} - \hat{\mathbf{s}}_{i,t})^2} \tag{D.16}$$

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

$$\mathrm{PICP} = \frac{1}{N_{\text{test}} T} \sum_{i=1}^{N_{\text{test}}} \sum_{t=1}^{T} \mathbb{I}\left[\mathbf{y}_{i,t} \in \left[\hat{\mathbf{y}}_{i,t}^{\text{lower}}, \hat{\mathbf{y}}_{i,t}^{\text{upper}}\right]\right] \tag{D.17}$$

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

$$\text{MPIW} = \frac{1}{N_{\text{test}}\,T} \sum_{i=1}^{N_{\text{test}}} \sum_{t=1}^{T} \left(\hat{\mathbf{y}}_{i,t}^{\text{upper}} - \hat{\mathbf{y}}_{i,t}^{\text{lower}} \right) \tag{D.18}$$

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.

D.4. ALTERNATIVE MULTI-FIDELITY MODELS AND COMPREHENSIVE INVESTIGATIONS

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 5.4 demonstrated its effectiveness across various datasets. However, we also clarified that multi-fidelity architecture is one particular architecture drawn from Figure 5.4. Herein, we provide three other variants that can be obtained following Equation (5.3), showing in Figure D.5, Figure D.6, and Figure D.7.

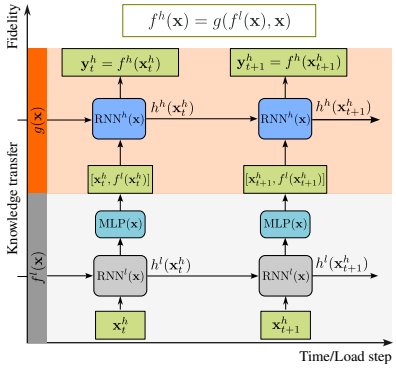


Figure D.5: Schematic of the proposed multi-fidelity recurrent neural network architecture: *MF-Nest-Output*. This configuration follows the formulation $f^h(\mathbf{x}) = g(f^l(\mathbf{x}), \mathbf{x})$, where $g(\cdot)$ is a non-linear transfer learning model—specifically, another RNN. This architecture 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.

As shown in Figure D.5, Figure D.6, and Figure D.7, 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 Figure 5.6) adopt a linear transfer learning model[2] and additionally employ a residual model, another RNN, to learn the discrepancy between the high-fidelity and low-fidelity predictions.

[2]We adopt the identity function as the transfer learning model in this work.

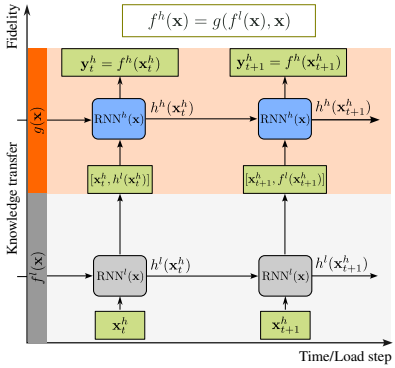


Figure D.6: Schematic of the proposed multi-fidelity recurrent neural network architectures: *MF-Nest-Hidden*. This configuration follows the formulation $f^h(\mathbf{x}) = g(f^l(\mathbf{x}), \mathbf{x})$, where $g(\cdot)$ is also another RNN and without a residual model. Instead, the transfer model takes as input the hidden state from the low-fidelity RNN, concatenated with the high-fidelity strain inputs.

D

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 D.1.

Table D.1: Summary of inputs and outputs for different multi-fidelity model variants from Figure 5.4

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	$[\mathbf{x}^h, h^l(\mathbf{x}^h)]$	$\mathbf{y}^h - g(f^l(\mathbf{x}^h))$	$f^h(\mathbf{x}) = g(f^l(\mathbf{x})) + r(\mathbf{x})$

COMPREHENSIVE INVESTIGATION ON MULTI-FIDELITY ARCHITECTURES

We proposed four distinct multi-fidelity models in this paper including Figure 5.6, namely *MF-Nest-Hidden*, *MF-Nest-Output*, *MF-Residual-Hidden*, and *MF-Residual-Original*, with detailed input and learning objective summarized in Table D.1. 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 D.4, and followed by an RNN+VeBRNN architecture trained in Appendix D.4.

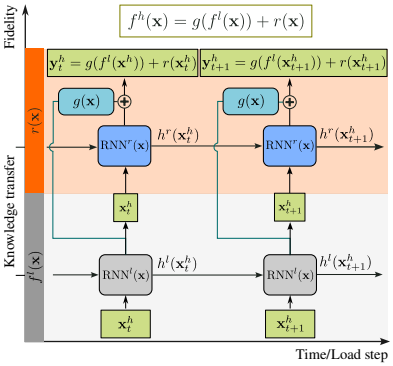


Figure D.7: Schematic of the proposed multi-fidelity recurrent neural network architecture: *MF-Residual-Original*. This configuration adopts the formulation $f^h(\mathbf{x}) = g(f^l(\mathbf{x})) + r(\mathbf{x})$, where $g(\cdot)$ is a linear transfer learning model and $r(\mathbf{x})$ is a residual model implemented as another RNN. Unlike the configuration shown in Figure 5.6, this variant trains the residual model using only the original high-fidelity strain input.

RNN+RNN ARCHITECTURE TRAINING

We follow the same setup as Figure 5.9 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 Figure D.8.

Different colors in Figure D.8 represent different MF model configurations proposed according to Equation (5.4) and Figure 5.4, 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 Figure D.8. 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 Figure D.5. Furthermore, hidden states in Figure D.6 and Figure 5.6 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*

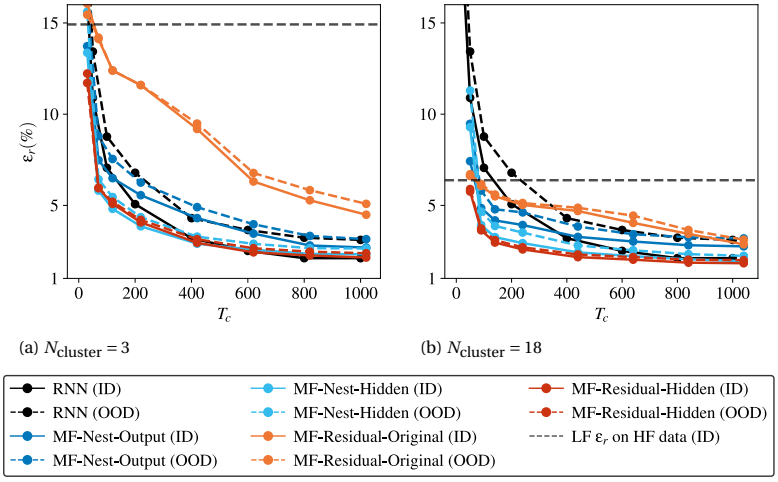


Figure D.8: Comparison of different knowledge transfer configurations for **Dataset 2** with RNN+RNN training. The single-fidelity RNN training based on DNS results are shown with black lines for comparison. Meanwhile, the results of RNN+RNN training have offsets of 33.3 and 55.5 for $N_{\text{cluster}} = 3$ and $N_{\text{cluster}} = 18$ in the x-axis, considering the computational cost of low-fidelity data.

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 [147].

RNN+VeBRNN ARCHITECTURE TRAINING

In this section, we continue to test the four multi-fidelity model configurations developed in Appendix D.4, 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 ROM, i.e., Ncluster = 3 and Ncluster = 18, are shown in Figure D.9 and Figure D.10, respectively.

According to Figure D.9 and Figure D.10, 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 Figure D.9e and Figure D.10e, 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 shows in Figure 5.11 that the *MF-Nest-Hidden* already converged to a reasonable level for this deterministic dataset.

In summary, the results in Appendix D.4 and Appendix D.4 demonstrate that *MF-*

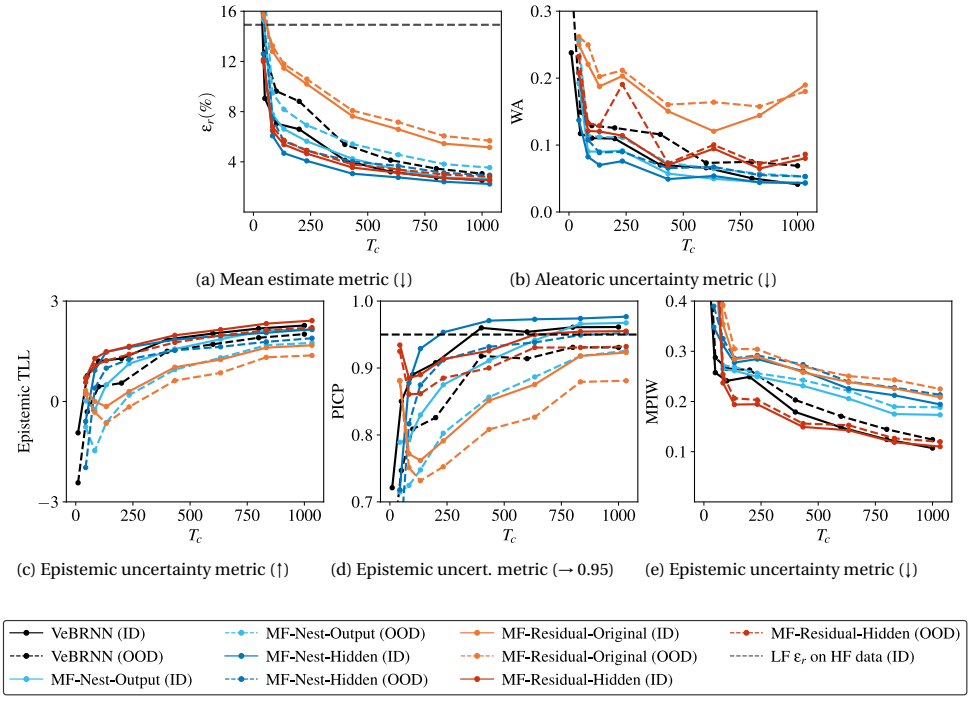


Figure D.9: Comparison of different multi-fidelity model configurations proposed in this paper for **Dataset 2** with RNN+VeBRNN architecture under low-fidelity of $N_{\text{cluster}} = 3$

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.

D.5. ADDITIONAL EXPERIMENTAL RESULTS

PREDICTIONS OF RNN ON THE SINGLE-FIDELITY PROBLEM

The predictions of RNN with 2000 training paths obtained by $\widetilde{\text{DNS}}$ are given in Figure D.11. 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.

ADDITIONAL TRAINING RESULTS FOR DATASET 3

Section 5.4.3 presents the VeBRNN+VeBRNN model for Dataset 3, where both fidelity levels are noisy (see also Figure D.3 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 Figure D.12; and (2) a RNN+VeBRNN model where the HF prediction is Bayesian but the

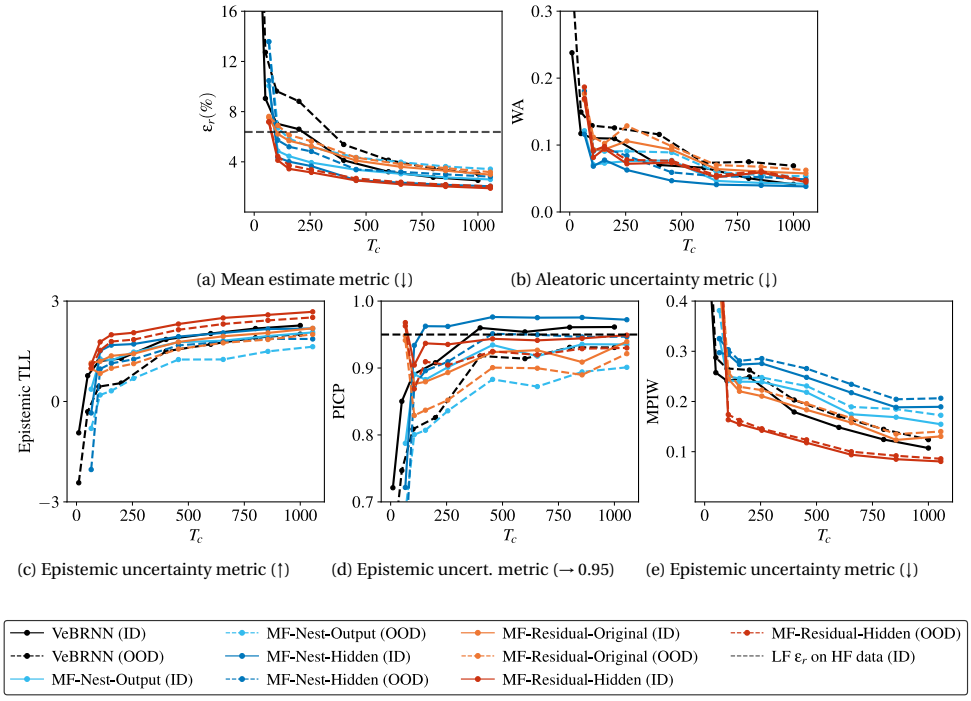


Figure D.10: Comparison of different multi-fidelity model configurations proposed in this paper for **Dataset 2** with RNN+VeBRNN architecture under low-fidelity of $N_{\text{cluster}} = 18$

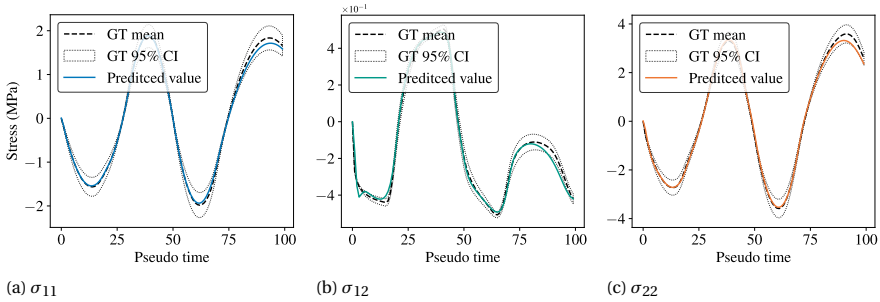
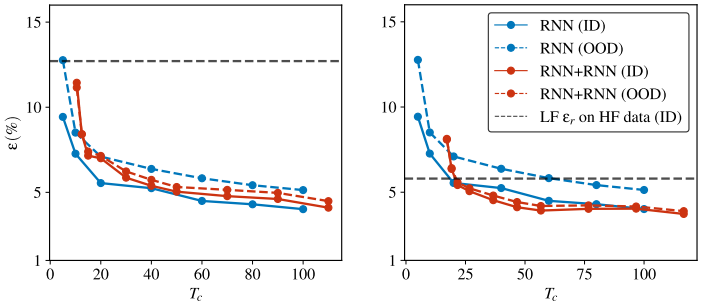


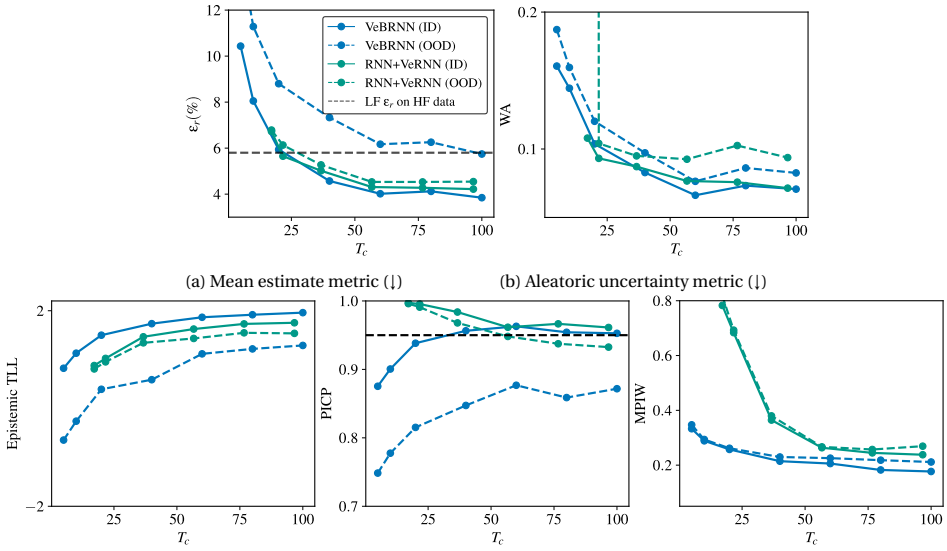
Figure D.11: Predictions of the RNN with 2000 training paths ($T_c = 2000$) on the ID test path

LF is deterministic (Figure D.13). 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) $N_{\text{cluster}} = 3$ (b) $N_{\text{cluster}} = 18$

Figure D.12: 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 (↓)

Figure D.13: 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 $N_{\text{cluster}} = 18$.

D.6. HYPERPARAMETER SETTINGS

For the hyperparameter settings, we draw on insights from prior studies that address similar problems in a deterministic setting [34, 35], 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, 10000 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.

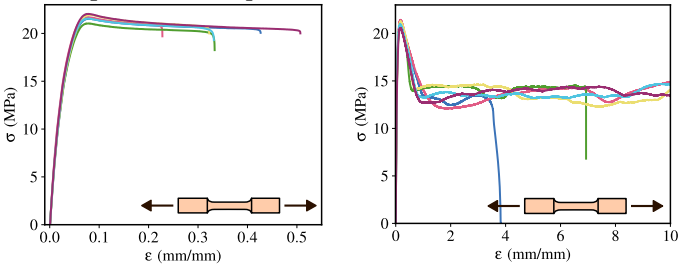
- **Training / Validation:**
  - For deterministic training, the dataset is split into 80%/20% for training and validation, respectively.
  - For cooperative training, no validation split is needed. We fix the iteration count to $K = 2$ for all experiments.

E

APPENDICES FOR CHAPTER 6

E.1. EXPERIMENTAL TENSILE TESTS

Experimental data are provided by LyondellBasell Industries Holdings, including tensile test results. The tensile test results of single recycled polypropylene (PP) and polyethylene (PE) under room temperature and quasi-static tension are presented in Figure E.1.



(a) Uniaxial tensile test of recycled PP

(b) Uniaxial tensile test of recycled PE

Figure E.1: Stress vs. strain curves from a tensile test using recycled PP and PE. Experimental data obtained following the ISO 527 standard [246] and repeated six times.

As shown in Figure E.1, the tensile tests were repeated six times under identical conditions. The two recycled polymers exhibit distinct mechanical responses: recycled PP tends to fail earlier with minimal strain softening, whereas recycled PE demonstrates higher ductility accompanied by pronounced post-yield softening. It is worth noting that both materials exhibit significant variability in their stress-strain behavior, likely due to impurities introduced during the recycling process.

We then present the tensile test results of recycled PP/PE blends with and without compatibilizers in Figure E.2. They are consistent with prior findings that PP and PE are inherently immiscible [171], as evidenced by the lowest observed failure strain in the uncompatibilized blend. Upon introducing compatibilizers, the failure strain increases significantly, albeit at the expense of reduced yield stress. As shown in the TEM images on

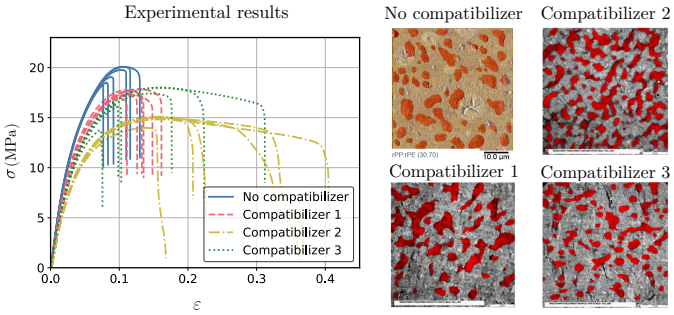


Figure E.2: Stress and strain curves of PP/PE blend without/with compatibilizer and TEM micrograph using staining by Ruthenium tetroxide (RuO_4).

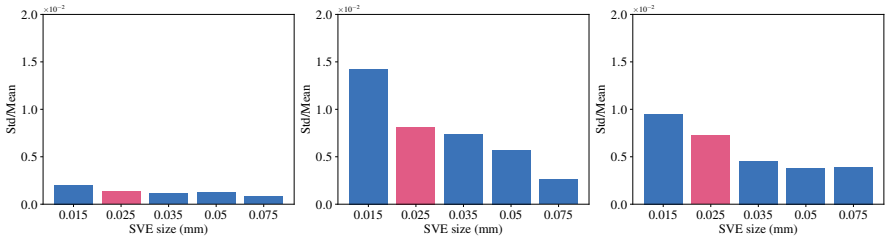
the right panel of Figure E.2, the microstructural morphology also changes with different compatibilizer types, which agrees with observations in [176, 188].

E

E.2. CONVERGENCE ANALYSES

We perform convergence analyses on two critical FEA parameters: the size of the SVE and the mesh resolution. Since the developed FEA model aims to generate reliable datasets across a range of volume fractions for both recycled PP and compatibilizers, we first assess convergence using uncompatibilized recycled PP/PE blends at several selected volume fractions of recycled PP. These initial analyses are used to determine a suitable SVE size and mesh resolution. The chosen parameters are then validated on compatibilized blends with varying volume fractions of recycled PP and compatibilizer.

CONVERGENCE ANALYSES FOR UNCOMPATIBILIZED RECYCLED PP/PE BLEND

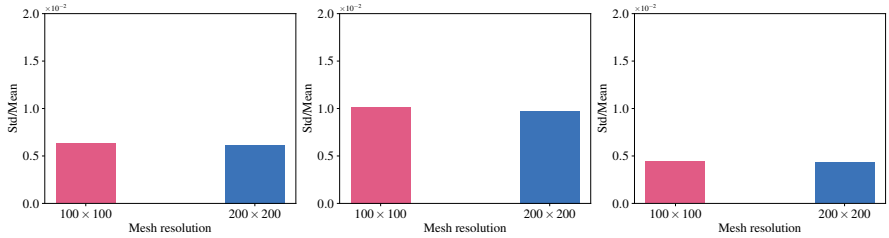


(a) $v_{pp} = 0.30$ (b) $v_{pp} = 0.50$ (c) $v_{pp} = 0.70$

Figure E.3: Average relative standard deviation (std/mean) with six arbitrary simulations of different SVE sizes.

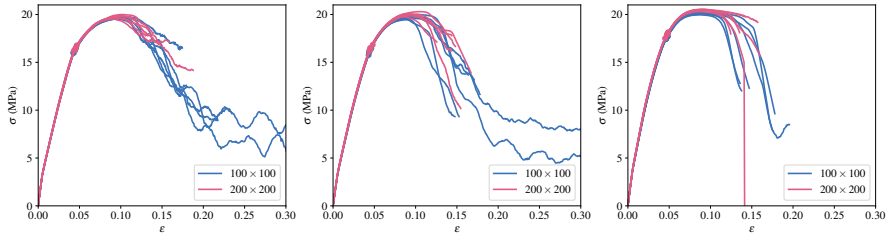
Size of the SVE In the field of computational homogenization, the size of the SVE or RVE significantly influences the resulting predictions [273]. Larger SVEs generally capture more microstructural features, yielding more stable and representative responses, albeit at the cost of increased computational expense. To evaluate convergence, we consider five SVE sizes: $[0.015, 0.025, 0.035, 0.05, 0.075]$ mm, and three volume fractions of recycled PP ($v_{pp} \in [0.3, 0.5, 0.7]$). Each configuration is simulated six times, and the ratio of the

standard deviation to the mean is computed for each load step up to the point of damage energy dispersion. The results are presented in Figure E.3. With the increase of SVE size, the variation of the homogenized strain-stress curve up to damage occurs. Interestingly, the SVE size exhibits limited influence on the homogenized stress–strain response, with less than 2% variation observed across all cases. Considering both computational cost and the representativeness of microstructural features, we select a size of 0.025 mm as the final SVE size, as highlighted by the pink bars, where the variation remains below 1%.



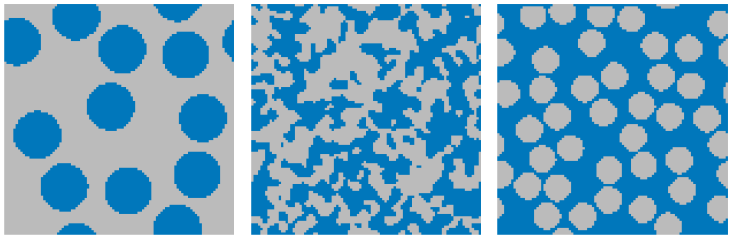
(a) $v_{pp} = 0.30$ (b) $v_{pp} = 0.50$ (c) $v_{pp} = 0.70$

Figure E.4: Average relative standard deviation (std/mean) across different mesh resolutions.



(a) $v_{pp} = 0.30$ (b) $v_{pp} = 0.50$ (c) $v_{pp} = 0.70$

Figure E.5: Average relative standard deviation (std/mean) across different mesh resolutions. The dot point for each curve represents the damage energy dissipation that occurs for the corresponding simulation.



(a) $v_{pp} = 0.30$ (b) $v_{pp} = 0.50$ (c) $v_{pp} = 0.70$

Figure E.6: An arbitrary realization of the recycled PP/PE blends microstructure under different v_{pp} without compatibilizer.

Mesh size Based on the selected SVE size, we subsequently conduct mesh size analysis. In the previous study on the SVE size, the analyses were performed based on a mesh resolution of 100 × 100 structural elements. In this section, we simulate with a fine mesh of

200×200 structural elements, the results comparison is given by Figure E.4. The developed FEA model performs stably with different mesh resolutions. Although different v_{pp} have a slight influence on the homogenized strain-stress results, the *Std/Mean* values are below 1 % and are quite robust to the mesh resolution. The simulation with a mesh resolution of 100×100 costs around 100 s using 1 CPU on a platform of an AMD Ryzen™ 9 7945HX processor, 16 GB RAM, while that of 100×100 is around 9500 s.

Then, we further visualize the homogenized stress-strain curves for all cases of different v_{pp} and mesh resolution in Figure E.5. We can observe that the homogenized stress-strain curves almost overlap before the dot points, which demonstrates the robustness of the SVE simulations before damage energy dissipates. Furthermore, we plot an arbitrary microstructure realization for each v_{pp} in Figure E.6.

CONVERGENCE ANALYSES FOR COMPATIBILIZED RECYCLED PP/PE BLEND

Since the size of the SVE and the mesh scheme is determined based on the case without compatibilizer, a verification study is performed in this section for the compatibilized recycled PP/PE blend. In order to cover an adequate range of volume fractions for PP and compatibilizer phases, $v_{pp} \in [0.2, 0.5, 0.7]$ and $v_c \in [0.01, 0.05, 0.10]$ are chosen, leading to nine combinations. The relative variation results regarding the ratio between the standard deviation and mean of the homogenized stress-strain up to damage energy dissipation are given by Figure E.7.

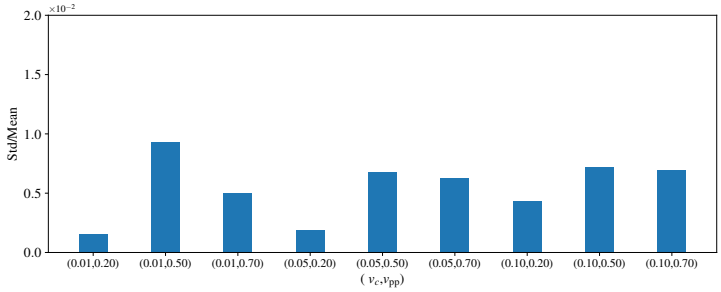
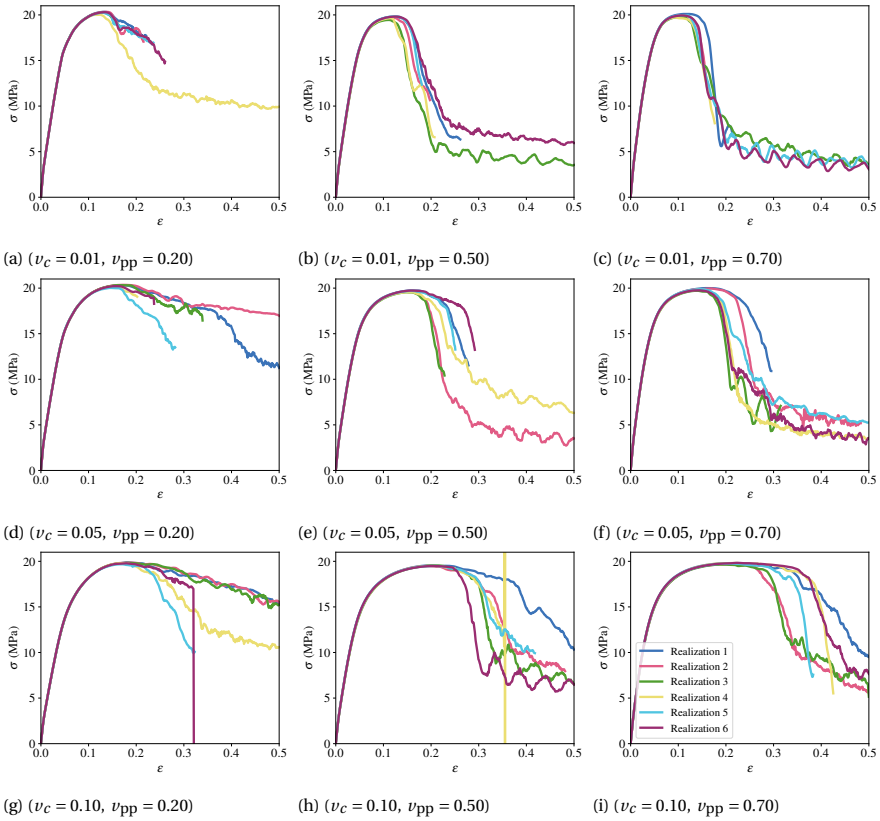


Figure E.7: Average relative standard deviation (std/mean) across different combinations of v_c and v_{pp}.

According to Figure E.7, the developed FEA model—calibrated using the mesh size and resolution from the non-compatibilized case—remains stable when applied to compatibilized blends. Specifically, while the mechanical response depends on the volume fractions of the constituent phases, all cases exhibit a coefficient of variation (i.e., *Std/Mean*) below 1%. In addition, Figure E.8 presents the homogenized stress–strain curves, along with an example realization of the microstructure corresponding to each combination of v_c and v_{pp}. As expected, the incorporation of the compatibilizer leads to significant variation in the mechanical performance of the recycled PP/PE blend.



(a) ($v_c = 0.01$, $v_{pp} = 0.20$) (b) ($v_c = 0.01$, $v_{pp} = 0.50$) (c) ($v_c = 0.01$, $v_{pp} = 0.70$)

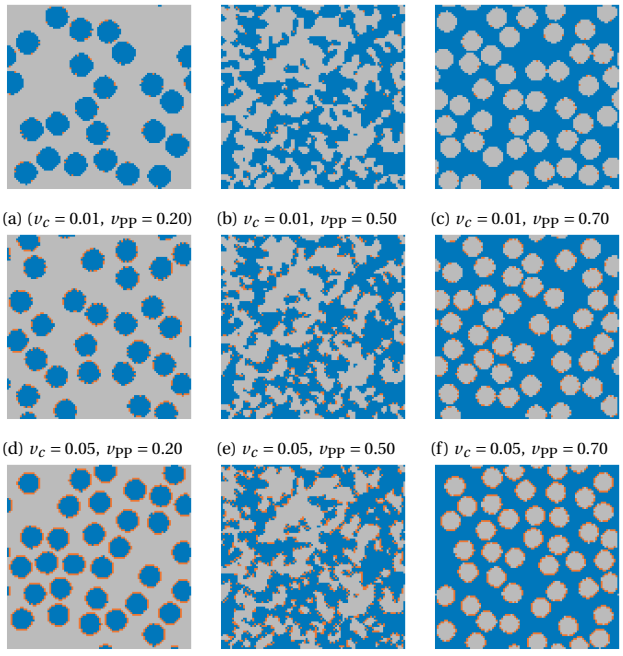
(d) ($v_c = 0.05$, $v_{pp} = 0.20$) (e) ($v_c = 0.05$, $v_{pp} = 0.50$) (f) ($v_c = 0.05$, $v_{pp} = 0.70$)

(g) ($v_c = 0.10$, $v_{pp} = 0.20$) (h) ($v_c = 0.10$, $v_{pp} = 0.50$) (i) ($v_c = 0.10$, $v_{pp} = 0.70$)

Figure E.8: Homogenized stress-strain curves for each combination of v_c and v_{pp}

E.3. DATASETS

In this appendix, the training dataset is visualized in Figure E.10 using a 3×2 scatter plot, where each column corresponds to an input variable and each row to an output variable. It is observed that the compatibilizer modulus E_c has a nonlinear correlation with the failure strain ε_f and the toughness U. The weight fraction of recycled PP and compatibilizer has linear impacts: increasing w_{PP} leads to smaller failure strain and toughness, while w_c shows an opposite pattern. Overall, the presence of substantial noise and nonlinear interactions renders this a challenging learning task for conventional machine learning approaches.



(a) ($v_c = 0.01$, $v_{PP} = 0.20$) (b) $v_c = 0.01$, $v_{PP} = 0.50$ (c) $v_c = 0.01$, $v_{PP} = 0.70$

(d) $v_c = 0.05$, $v_{PP} = 0.20$ (e) $v_c = 0.05$, $v_{PP} = 0.50$ (f) $v_c = 0.05$, $v_{PP} = 0.70$

(g) $v_c = 0.10$, $v_{PP} = 0.20$ (h) $v_c = 0.10$, $v_{PP} = 0.50$ (i) $v_c = 0.10$, $v_{PP} = 0.70$

Figure E.9: An example realization of the recycled PP/PE blend microstructure with varying v_c and v_{PP}. The microstructure consists of three phases: recycled PP (blue), recycled PE (gray), and compatibilizer (orange).

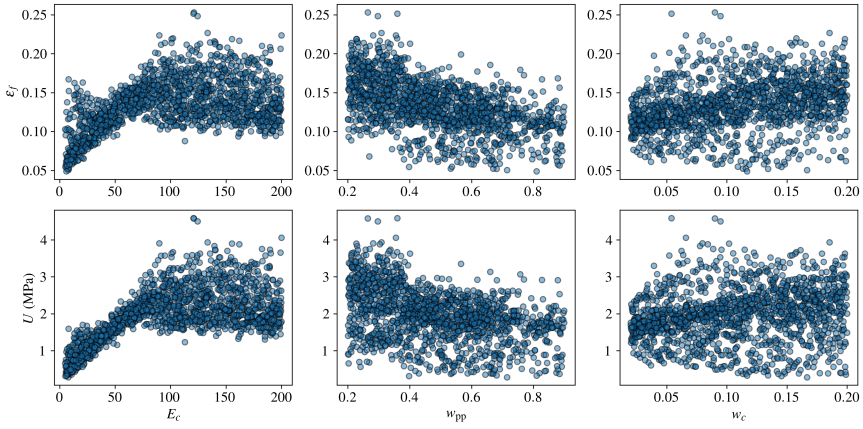


Figure E.10: Visualization of the training dataset

BIBLIOGRAPHY

[1] OpenAI et al. *GPT-4 Technical Report*. 2024. arXiv: 2303.08774 [cs.CL].

[2] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2021.

[3] Ian Goodfellow et al. *Deep learning*. Vol. 1. 2. MIT press Cambridge, 2016.

[4] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*. Vol. 1. 1. MIT press Cambridge, 1998.

[5] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets". In: *Neural computation* 18.7 (2006), pp. 1527–1554.

[6] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (2002), pp. 2278–2324.

[7] Jeffrey L Elman. "Finding structure in time". In: *Cognitive science* 14.2 (1990), pp. 179–211.

[8] Jascha Sohl-Dickstein et al. *Deep Unsupervised Learning using Nonequilibrium Thermodynamics*. 2015. arXiv: 1503.03585 [cs.LG].

[9] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[10] Miguel A Bessa et al. "A framework for data-driven analysis of materials under uncertainty: Countering the curse of dimensionality". In: *Computer Methods in Applied Mechanics and Engineering* 320 (2017), pp. 633–667.

[11] Xin Liu et al. "A review of artificial neural networks in the constitutive modeling of composite materials". In: *Composites Part B: Engineering* 224 (2021), p. 109152. ISSN: 1359-8368. DOI: https://doi.org/10.1016/j.compositesb.2021.109152.

[12] Jan Niklas Fuhg et al. *A review on data-driven constitutive laws for solids*. 2024. arXiv: 2405.03658 [cs.CE].

[13] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2.5 (1989), pp. 359–366. ISSN: 0893-6080. DOI: https://doi.org/10.1016/0893-6080(89)90020-8.

[14] Chiyuan Zhang et al. "Understanding Deep Learning (Still) Requires Rethinking Generalization". In: *Commun. ACM* 64.3 (Feb. 2021), pp. 107–115. ISSN: 0001-0782. DOI: 10.1145/3446776.

[15] Moloud Abdar et al. "A review of uncertainty quantification in deep learning: Techniques, applications and challenges". In: *Information Fusion* 76 (2021), pp. 243–297. ISSN: 1566-2535. DOI: https://doi.org/10.1016/j.inffus.2021.05.008.

[16] Eyke Hüllermeier and Willem Waegeman. “Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods”. In: *Machine learning* 110.3 (2021), pp. 457–506.

[17] Alex Kendall and Yarin Gal. “What uncertainties do we need in bayesian deep learning for computer vision?” In: *Advances in neural information processing systems* 30 (2017).

[18] Armen Der Kiureghian and Ove Ditlevsen. “Aleatory or epistemic? Does it matter?” In: *Structural Safety* 31.2 (2009). Risk Acceptance and Risk Communication, pp. 105–112. ISSN: 0167-4730. DOI: https://doi.org/10.1016/j.strusafe.2008.06.020.

[19] Freddie Bickford Smith et al. *Rethinking Aleatoric and Epistemic Uncertainty*. 2024. arXiv: 2412.20892 [cs.LG].

[20] Christopher M. Bishop. “Mixture density networks”. Technical Report. Birmingham, 1994.

[21] Radford M Neal. *Bayesian learning for neural networks*. Vol. 118. Springer Science & Business Media, 1995.

[22] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. “Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017.

[23] Yarin Gal and Zoubin Ghahramani. “Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning”. In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by Maria Florina Balcan and Kilian Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 1050–1059.

[24] Xiaolong He and Jiun-Shyan Chen. “Thermodynamically consistent machine-learned internal state variable approach for data-driven modeling of path dependent materials”. In: *Computer Methods in Applied Mechanics and Engineering* 402 (2022). A Special Issue in Honor of the Lifetime Achievements of J. Tinsley Oden, p. 115348. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2022.115348.

[25] M. Giselle Fernández-Godino. “Review of multi-fidelity models”. In: *Advances in Computational Science and Engineering* 1.4 (2023), pp. 351–400. DOI: 10.3934/acse.2023015.

[26] Haitao Liu, Jianfei Cai, and Yew-Soon Ong. “Remarks on multi-output Gaussian process regression”. In: *Knowledge-Based Systems* 144 (2018), pp. 102–121. ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2017.12.034.

[27] Alexander I.J Forrester, András Sóbester, and Andy J Keane. “Multi-fidelity optimization via surrogate modelling”. In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 463.2088 (2007), pp. 3251–3269. eprint: https://royalsocietypublishing.org/doi/pdf/10.1098/rspa.2007.1900.

[28] David J. J. Toal. “Some considerations regarding the use of multi-fidelity Kriging in the construction of surrogate models”. In: *Structural and Multidisciplinary Optimization* 51.6 (2015), pp. 1223–1245. ISSN: 1615-1488. DOI: 10.1007/s00158-014-1209-5.

[29] Roland Can Aydin, Fabian Albert Braeu, and Christian Johannes Cyron. “General Multi-Fidelity Framework for Training Artificial Neural Networks With Computational Models”. In: *Frontiers in Materials* 6 (2019). ISSN: 2296-8016. DOI: 10.3389/fmats.2019.00061.

[30] Zi Yang et al. “Deep learning approaches for mining structure-property linkages in high contrast composites from simulation datasets”. In: *Computational Materials Science* 151 (2018), pp. 278–287. ISSN: 0927-0256. DOI: https://doi.org/10.1016/j.commatsci.2018.05.014.

[31] Harikrishnan Vijayakumaran et al. “Consistent machine learning for topology optimization with microstructure-dependent neural network material models”. In: *Journal of the Mechanics and Physics of Solids* 196 (2025), p. 106015.

[32] Audrey Olivier, Michael D. Shields, and Lori Graham-Brady. “Bayesian neural networks for uncertainty quantification in data-driven materials modeling”. In: *Computer Methods in Applied Mechanics and Engineering* 386 (2021), p. 114079. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2021.114079.

[33] Ozge Ozbayram, Audrey Olivier, and Lori Graham-Brady. “Heteroscedastic Gaussian Process Regression for material structure-property relationship modeling”. In: *Computer Methods in Applied Mechanics and Engineering* 431 (2024), p. 117326. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2024.117326.

[34] M. Mozaffar et al. “Deep learning predicts path-dependent plasticity”. In: *Proceedings of the National Academy of Sciences* 116.52 (2019), pp. 26414–26420. ISSN: 0027-8424. DOI: 10.1073/pnas.1911815116. eprint: https://www.pnas.org/content/116/52/26414.full.pdf.

[35] Aleksandr Dekhovich et al. “Cooperative data-driven modeling”. In: *Computer Methods in Applied Mechanics and Engineering* 417 (2023), p. 116432. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2023.116432.

[36] M.A. Maia et al. “Physically recurrent neural networks for path-dependent heterogeneous materials: Embedding constitutive models in a data-driven surrogate”. In: *Computer Methods in Applied Mechanics and Engineering* 407 (2023), p. 115934. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2023.115934.

[37] Baekjun Kim, Sangwon Lee, and Jihan Kim. “Inverse design of porous materials using artificial neural networks”. In: *Science advances* 6.1 (2020), eaax9324.

[38] Yiwen Zheng et al. “AI-Guided Inverse Design and Discovery of Recyclable Vitrimeric Polymers”. In: *Advanced Science* 12.6 (2025), p. 2411385.

[39] Chan Soo Ha et al. “Rapid inverse design of metamaterials based on prescribed mechanical behavior through machine learning”. In: *Nature Communications* 14.1 (2023), p. 5765.

[40] Yuxiang Gao et al. “An inverse design framework for optimizing tensile strength of composite materials based on a CNN surrogate for the phase field fracture model”. In: *Composites Part A: Applied Science and Manufacturing* (2025), p. 108758.

[41] Tianyu Huang et al. “Microstructure-guided deep material network for rapid nonlinear material modeling and uncertainty quantification”. In: *Computer Methods in Applied Mechanics and Engineering* 398 (2022), p. 115197.

[42] Akshay J. Thomas et al. “Bayesian inference of fiber orientation and polymer properties in short fiber-reinforced polymer composites”. In: *Composites Science and Technology* 228 (Sept. 2022), p. 109630. ISSN: 0266-3538. DOI: 10.1016/j.compscitech.2022.109630.

[43] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*. Vol. 4. 4. Springer, 2006.

[44] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, Nov. 2006. ISBN: 9780262256834. DOI: 10.7551/mitpress/3206.001.0001.

[45] Kevin P. Murphy. *Probabilistic Machine Learning: Advanced Topics*. MIT Press, 2023.

[46] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. DOI: 10.48550/ARXIV.1412.6980.

[47] Sebastian Ruder. *An overview of gradient descent optimization algorithms*. 2017. arXiv: 1609.04747 [cs.LG].

[48] Jun S Liu and Jun S Liu. *Monte Carlo strategies in scientific computing*. Vol. 10. Springer, 2001.

[49] Radford M Neal. “MCMC using Hamiltonian dynamics”. In: *Handbook of markov chain monte carlo* 2.11 (2011), p. 2.

[50] Max Welling and Yee W Teh. “Bayesian learning via stochastic gradient Langevin dynamics”. In: *Proceedings of the 28th international conference on machine learning (ICML-11)*. 2011, pp. 681–688.

[51] Gareth O Roberts and Richard L Tweedie. “Exponential convergence of Langevin distributions and their discrete approximations”. In: (1996).

[52] Chunyuan Li et al. “Preconditioned stochastic gradient Langevin dynamics for deep neural networks”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 30. 1. 2016.

[53] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. “Variational Inference: A Review for Statisticians”. In: *Journal of the American Statistical Association* 112.518 (2017), pp. 859–877. DOI: 10.1080/01621459.2017.1285773. eprint: https://doi.org/10.1080/01621459.2017.1285773.

[54] Chuan Guo et al. “On calibration of modern neural networks”. In: *International conference on machine learning*. PMLR. 2017, pp. 1321–1330.

[55] Chuan Guo et al. “Simple black-box adversarial attacks”. In: *International conference on machine learning*. PMLR. 2019, pp. 2484–2493.

[56] Nicki Skafte, Martin Jørgensen, and Søren Hauberg. "Reliable training and estimation of variance networks". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019.

[57] Maximilian Seitzer et al. *On the Pitfalls of Heteroscedastic Uncertainty Estimation with Probabilistic Neural Networks*. 2022. arXiv: 2203.09168 [cs.LG].

[58] Stefan Depeweg et al. "Decomposition of uncertainty in Bayesian deep learning for efficient and risk-sensitive learning". In: *International conference on machine learning*. PMLR. 2018, pp. 1184–1193.

[59] Shakir Mohamed and Balaji Lakshminarayanan. "Learning in implicit generative models". In: *arXiv preprint arXiv:1610.03483* (2016).

[60] Murat Sensoy et al. "Uncertainty-aware deep classifiers using generative models". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. 04. 2020, pp. 5620–5627.

[61] Ali Harakeh et al. "Estimating Regression Predictive Distributions with Sample Networks". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 37.6 (June 2023), pp. 7830–7838. DOI: 10.1609/aaai.v37i6.25948.

[62] David A Nix and Andreas S Weigend. "Estimating the mean and variance of the target probability distribution". In: *Proceedings of 1994 ieee international conference on neural networks (ICNN'94)*. Vol. 1. IEEE. 1994, pp. 55–60.

[63] Gregory P Meyer and Niranjan Thakurdesai. "Learning an uncertainty-aware object detector for autonomous driving". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 10521–10527.

[64] Alexander Immer et al. "Effective Bayesian Heteroscedastic Regression with Deep Neural Networks". In: *Thirty-seventh Conference on Neural Information Processing Systems*. 2023.

[65] Laurens Sluijterman, Eric Cator, and Tom Heskes. "Optimal training of mean variance estimation neural networks". In: *Neurocomputing* (2024), p. 127929.

[66] Andrew Stirn and David A. Knowles. *Variational Variance: Simple, Reliable, Calibrated Heteroscedastic Noise Variance Parameterization*. 2020. arXiv: 2006.04910 [cs.LG].

[67] Peng Cui, Wenbo Hu, and Jun Zhu. "Calibrated reliable regression using maximum mean discrepancy". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 17164–17175.

[68] Andrew G Wilson and Pavel Izmailov. "Bayesian Deep Learning and a Probabilistic Perspective of Generalization". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 4697–4708.

[69] Florian Wenzel et al. *How Good is the Bayes Posterior in Deep Neural Networks Really?* 2020. DOI: 10.48550/ARXIV.2002.02405.

[70] Pavel Izmailov et al. *What Are Bayesian Neural Network Posteriors Really Like?* 2021. arXiv: 2104.14421 [cs.LG].

[71] Alex Graves. “Practical Variational Inference for Neural Networks”. In: *Advances in Neural Information Processing Systems*. Ed. by J. Shawe-Taylor et al. Vol. 24. Curran Associates, Inc., 2011.

[72] Charles Blundell et al. *Weight Uncertainty in Neural Networks*. 2015. DOI: 10.48550/ARXIV.1505.05424.

[73] Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. *Deep Ensembles: A Loss Landscape Perspective*. 2020. arXiv: 1912.02757 [stat.ML].

[74] Kaizheng Wang et al. “Credal deep ensembles for uncertainty quantification”. In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 79540–79572.

[75] Matias Valdenegro-Toro and Daniel Saromo Mori. “A Deeper Look into Aleatoric and Epistemic Uncertainty Disentanglement”. In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2022, pp. 1508–1516. DOI: 10.1109/CVPRW56347.2022.00157.

[76] Bálint Mucsányi, Michael Kirchhof, and Seong Joon Oh. “Benchmarking Uncertainty Disentanglement: Specialized Uncertainties for Specialized Tasks”. In: *ICML 2024 Workshop on Structured Probabilistic Inference & Generative Modeling*. 2024.

[77] Alexander Amini et al. “Deep Evidential Regression”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 14927–14937.

[78] Jose Miguel Hernandez-Lobato and Ryan Adams. “Probabilistic Backpropagation for Scalable Learning of Bayesian Neural Networks”. In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, July 2015, pp. 1861–1869.

[79] Fredrik K. Gustafsson, Martin Danelljan, and Thomas B. Schön. *How Reliable is Your Regression Model’s Uncertainty Under Real-World Distribution Shifts?* 2023. arXiv: 2302.03679 [cs.LG].

[80] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[81] Kyunghyun Cho. “Learning phrase representations using RNN encoder-decoder for statistical machine translation”. In: *arXiv preprint arXiv:1406.1078* (2014).

[82] Zhe Gan et al. *Scalable Bayesian Learning of Recurrent Neural Networks for Language Modeling*. 2017. arXiv: 1611.08034 [cs.CL].

[83] Michele Caprio et al. “Credal Bayesian Deep Learning”. In: *Transactions on Machine Learning Research* (2024). ISSN: 2835-8856.

[84] Nicholas G. Polson and Vadim Sokolov. “Deep Learning: A Bayesian Perspective”. In: *Bayesian Analysis* 12.4 (Dec. 2017). DOI: 10.1214/17-ba1082.

[85] Jie Chen et al. “Multi-fidelity neural optimization machine for Digital Twins”. In: *Structural and Multidisciplinary Optimization* 65.12 (2022), p. 340. ISSN: 1615-1488. DOI: 10.1007/s00158-022-03443-2.

[86] Qi Zhou et al. “A two-stage adaptive multi-fidelity surrogate model-assisted multi-objective genetic algorithm for computationally expensive problems”. In: *Engineering with Computers* 37.1 (2021), pp. 623–639. ISSN: 1435-5663. DOI: 10.1007/s00366-019-00844-8.

[87] Zeliang Liu, M.A. Bessa, and Wing Kam Liu. “Self-consistent clustering analysis: An efficient multi-scale scheme for inelastic heterogeneous materials”. In: *Computer Methods in Applied Mechanics and Engineering* 306 (2016), pp. 319–341. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2016.04.004.

[88] Xinshuai Zhang et al. “Multi-fidelity deep neural network surrogate model for aerodynamic shape optimization”. In: *Computer Methods in Applied Mechanics and Engineering* 373 (2021), p. 113485. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2020.113485.

[89] M. Giselle Ferná ndez-Godino et al. “Issues in Deciding Whether to Use Multifidelity Surrogates”. In: *AIAA Journal* 57.5 (May 2019), pp. 2039–2054. DOI: 10.2514/1.j057750.

[90] Xuhui Meng, Hessam Babaee, and George Em Karniadakis. “Multi-fidelity Bayesian neural networks: Algorithms and applications”. In: *Journal of Computational Physics* 438 (2021), p. 110361. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2021.110361.

[91] Kurt Cutajar et al. *Deep Gaussian Processes for Multi-fidelity Modeling*. 2019. DOI: 10.48550/ARXIV.1903.07320.

[92] Marc C Kennedy and Anthony O’Hagan. “Predicting the output from a complex computer code when fast approximations are available”. In: *Biometrika* 87.1 (2000), pp. 1–13.

[93] Quan Lin et al. “A multi-output multi-fidelity Gaussian process model for non-hierarchical low-fidelity data fusion”. In: *Knowledge-Based Systems* 254 (2022), p. 109645. ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2022.109645.

[94] Jonathan Tammer Eweis-Labolle, Nicholas Oune, and Ramin Bostanabad. “Data Fusion With Latent Map Gaussian Processes”. In: *Journal of Mechanical Design* 144.9 (June 2022). 091703. ISSN: 1050-0472. DOI: 10.1115/1.4054520. eprint: https://asmedigitalcollection.asme.org/mechanicaldesign/article-pdf/144/9/091703/6887630/md_144_9_091703.pdf.

[95] Nicholas Oune and Ramin Bostanabad. “Latent map Gaussian processes for mixed variable metamodeling”. In: *Computer Methods in Applied Mechanics and Engineering* 387 (2021), p. 114128. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2021.114128.

[96] Robert B. Gramacy. *Surrogates: Gaussian Process Modeling, Design and Optimization for the Applied Sciences*. http://bobby.gramacy.com/surrogates/. Boca Raton, Florida: Chapman Hall/CRC, 2020.

[97] Zhong-Hua Han and Stefan Görtz. "Hierarchical Kriging Model for Variable-Fidelity Surrogate Modeling". In: *AIAA Journal* 50.9 (2012), pp. 1885–1896. DOI: 10.2514/1.J051354. eprint: https://doi.org/10.2514/1.J051354.

[98] Zhong-Hua Han, Ralf Zimmermann, and Stefan Goretz. "A new cokriging method for variable-fidelity surrogate modeling of aerodynamic data". In: *48th AIAA Aerospace sciences meeting including the new horizons forum and Aerospace exposition*. 2010, p. 1225.

[99] Zhong-Hua Han, Stefan Görtz, and Ralf Zimmermann. "Improving variable-fidelity surrogate modeling via gradient-enhanced kriging and a generalized hybrid bridge function". In: *Aerospace Science and Technology* 25.1 (2013), pp. 177–189. ISSN: 1270-9638. DOI: https://doi.org/10.1016/j.ast.2012.01.006.

[100] Loic Le Gratiet and Josselin Garnier. "Recursive co-kriging model for design of computer experiments with multiple levels of fidelity". In: *International Journal for Uncertainty Quantification* 4.5 (2014), pp. 365–386. ISSN: 2152-5080.

[101] Chanyoung Park, Raphael T. Haftka, and Nam H. Kim. "Low-fidelity scale factor improves Bayesian multi-fidelity prediction by reducing bumpiness of discrepancy function". In: *Structural and Multidisciplinary Optimization* 58.2 (2018), pp. 399–414. ISSN: 1615-1488. DOI: 10.1007/s00158-018-2031-2.

[102] Jiaxiang Yi, Yuansheng Cheng, and Jun Liu. "A novel fidelity selection strategy-guided multifidelity kriging algorithm for structural reliability analysis". In: *Reliability Engineering & System Safety* 219 (2022), p. 108247. ISSN: 0951-8320. DOI: https://doi.org/10.1016/j.ress.2021.108247.

[103] Haitao Liu et al. "When Gaussian Process Meets Big Data: A Review of Scalable GPs". In: *IEEE Transactions on Neural Networks and Learning Systems* 31.11 (2020), pp. 4405–4423. DOI: 10.1109/TNNLS.2019.2957109.

[104] Dongxia Wu et al. "Multi-fidelity Hierarchical Neural Processes". In: *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM, Aug. 2022. DOI: 10.1145/3534678.3539364.

[105] Katerina Giannoukou, Stefano Marelli, and Bruno Sudret. *A comprehensive framework for multi-fidelity surrogate modeling with noisy data: a gray-box perspective*. 2024. arXiv: 2401.06447 [stat.ME].

[106] Yiming Zhang et al. "Multifidelity Surrogate Based on Single Linear Regression". In: *AIAA Journal* 56.12 (2018), pp. 4944–4952. DOI: 10.2514/1.J057299. eprint: https://doi.org/10.2514/1.J057299.

[107] Xueguan Song et al. "A radial basis function-based multi-fidelity surrogate model: exploring correlation between high-fidelity and low-fidelity models". In: *Structural and Multidisciplinary Optimization* 60.3 (2019), pp. 965–981. ISSN: 1615-1488. DOI: 10.1007/s00158-019-02248-0.

[108] Maolin Shi et al. "A multi-fidelity surrogate model based on support vector regression". In: *Structural and Multidisciplinary Optimization* 61.6 (2020), pp. 2363–2375. ISSN: 1615-1488. DOI: 10.1007/s00158-020-02522-6.

[109] Lucas Zimmer, Marius Lindauer, and Frank Hutter. "Auto-Pytorch: Multi-Fidelity MetaLearning for Efficient and Robust AutoDL". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.9 (2021), pp. 3079–3090. DOI: 10.1109/TPAMI.2021.3067763.

[110] Dehao Liu and Yan Wang. "Multi-Fidelity Physics-Constrained Neural Network and Its Application in Materials Modeling". In: *Journal of Mechanical Design* 141.12 (Sept. 2019). 121403. ISSN: 1050-0472. DOI: 10.1115/1.4044400. eprint: https://asmedigitalcollection.asme.org/mechanicaldesign/article-pdf/141/12/121403/5873989/md_141_12_121403.pdf.

[111] Shibo Li et al. "Multi-Fidelity Bayesian Optimization via Deep Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 8521–8531.

[112] Mohammad Motamed. "A multi-fidelity neural network surrogate sampling method for uncertainty quantification". In: *International Journal for Uncertainty Quantification* 10.4 (2020), pp. 315–332. ISSN: 2152-5080.

[113] Souvik Chakraborty. "Transfer learning based multi-fidelity physics informed deep neural network". In: *Journal of Computational Physics* 426 (2021), p. 109942. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2020.109942.

[114] Milan Papež and Anthony Quinn. "Transferring model structure in Bayesian transfer learning for Gaussian process regression". In: *Knowledge-Based Systems* 251 (2022), p. 108875. ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2022.108875.

[115] Nolan Black and Ahmad R. Najafi. "Learning finite element convergence with the Multi-fidelity Graph Neural Network". In: *Computer Methods in Applied Mechanics and Engineering* 397 (2022), p. 115120. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2022.115120.

[116] Xuhui Meng and George Em Karniadakis. "A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse PDE problems". In: *Journal of Computational Physics* 401 (2020), p. 109020. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2019.109020.

[117] Mengwu Guo et al. "Multi-fidelity regression using artificial neural networks: Efficient approximation of parameter-dependent output quantities". In: *Computer Methods in Applied Mechanics and Engineering* 389 (2022). ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2021.114378.

[118] Paolo Conti et al. "Multi-fidelity surrogate modeling using long short-term memory networks". In: *Computer Methods in Applied Mechanics and Engineering* 404 (2023), p. 115811. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2022.115811.

[119] Jinhong Wu et al. "A deep learning-based multi-fidelity optimization method for the design of acoustic metasurface". In: *Engineering with Computers* 39.5 (2023), pp. 3421–3439. ISSN: 1435-5663. DOI: 10.1007/s00366-022-01765-9.

[120] Michael Betancourt. *A Conceptual Introduction to Hamiltonian Monte Carlo*. 2018. arXiv: 1701.02434 [stat.ME].

[121] Baptiste Kerleguer, Claire Cannamela, and Josselin Garnier. "A BAYESIAN NEURAL NETWORK APPROACH TO MULTI-FIDELITY SURROGATE MODELING". In: *International Journal for Uncertainty Quantification* 14.1 (2024), pp. 43–60. ISSN: 2152-5080. DOI: 10.1615/int.j.uncertaintyquantification.2023044584.

[122] Chanyoung Park, Nam Ho Kim, and Raphael T Haftka. "Including ρ in multi-fidelity surrogate prediction can make discrepancy extrapolation accurate by reducing bumpiness". In: *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. 2018, p. 0915.

[123] Alexander Forrester, Andras Sobester, and Andy Keane. *Engineering design via surrogate modelling: a practical guide*. John Wiley & Sons, 2008.

[124] Gil Shabat et al. "Fast and Accurate Gaussian Kernel Ridge Regression Using Matrix Decompositions for Preconditioning". In: *SIAM Journal on Matrix Analysis and Applications* 42.3 (2021), pp. 1073–1095. DOI: 10.1137/20M1343993. eprint: https://doi.org/10.1137/20M1343993.

[125] Giacomo Meanti et al. *Kernel methods through the roof: handling billions of points efficiently*. 2020. arXiv: 2006.10350 [cs.LG].

[126] Wei Deng et al. *Interacting Contour Stochastic Gradient Langevin Dynamics*. 2022. arXiv: 2202.09867 [stat.ML].

[127] Juyoung Lee et al. "A comprehensive multi-fidelity surrogate framework based on Gaussian process for datasets with heterogeneous responses". In: *Knowledge-Based Systems* 295 (2024), p. 111827. ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2024.111827.

[128] Sameer K. Deshpande et al. *Are you using test log-likelihood correctly?* 2024. arXiv: 2212.00219 [stat.ML].

[129] Quan Lin et al. "Multi-output Gaussian process prediction for computationally expensive problems with multiple levels of fidelity". In: *Knowledge-Based Systems* 227 (2021), p. 107151. ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2021.107151.

[130] Peng Liao et al. "Multi-fidelity convolutional neural network surrogate model for aerodynamic optimization based on transfer learning". In: *Physics of Fluids* 33.12 (2021).

[131] Bernardo P Ferreira, FM Andrade Pires, and Miguel A Bessa. "CRATE: A Python package to perform fast material simulations". In: *Journal of Open Source Software* 8.87 (2023), p. 5594.

[132] Bernardo P Ferreira, FM Andrade Pires, and Miguel A Bessa. "Adaptivity for clustering-based reduced-order modeling of localized history-dependent phenomena". In: *Computer Methods in Applied Mechanics and Engineering* 393 (2022), p. 114726.

[133] Jiaxiang Yi and Miguel Anibal Bessa. “rvesimulator: An automated representative volume element simulator for data-driven material discovery”. In: *AI for Accelerated Materials Design-NeurIPS 2023 Workshop*. 2023.

[134] Steven Adriaensen et al. “Efficient Bayesian Learning Curve Extrapolation using Prior-Data Fitted Networks”. In: *Thirty-seventh Conference on Neural Information Processing Systems*. 2023.

[135] Isaac E Lagaris, Aristidis Likas, and Dimitrios I Fotiadis. “Artificial neural networks for solving ordinary and partial differential equations”. In: *IEEE transactions on neural networks* 9.5 (1998), pp. 987–1000.

[136] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations”. In: *Journal of Computational physics* 378 (2019), pp. 686–707.

[137] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *nature* 521.7553 (2015), pp. 436–444.

[138] J. Ghaboussi, J. H. Garrett, and X. Wu. “Knowledge-Based Modeling of Material Behavior with Neural Networks”. In: *Journal of Engineering Mechanics* 117.1 (1991), pp. 132–153. DOI: 10.1061/(ASCE)0733-9399(1991)117:1(132). eprint: https://ascelibrary.org/doi/pdf/10.1061/%28ASCE%290733-9399%281991%29117%3A1%28132%29.

[139] BA Le, Julien Yvonnet, and Q-C He. “Computational homogenization of nonlinear elastic materials using neural networks”. In: *International Journal for Numerical Methods in Engineering* 104.12 (2015), pp. 1061–1084.

[140] Ling Wu et al. “A recurrent neural network-accelerated multi-scale model for elasto-plastic heterogeneous materials subjected to random cyclic and non-proportional loading paths”. In: *Computer Methods in Applied Mechanics and Engineering* 369 (2020), p. 113234. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2020.113234.

[141] Mahdad Eghbalian, Mehdi Pouragha, and Richard Wan. “A physics-informed deep neural network for surrogate modeling in classical elasto-plasticity”. In: *Computers and Geotechnics* 159 (2023), p. 105472. ISSN: 0266-352X. DOI: https://doi.org/10.1016/j.compgeo.2023.105472.

[142] Karl A. Kalina et al. “Neural networks meet anisotropic hyperelasticity: A framework based on generalized structure tensors and isotropic tensor functions”. In: *Computer Methods in Applied Mechanics and Engineering* 437 (2025), p. 117725. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2024.117725.

[143] Asghar Arshad Jadoon, Knut Andreas Meyer, and Jan Niklas Fuhg. “Automated model discovery of finite strain elastoplasticity from uniaxial experiments”. In: *Computer Methods in Applied Mechanics and Engineering* 435 (2025), p. 117653. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2024.117653.

[144] Harikrishnan Vijayakumaran et al. “Consistent machine learning for topology optimization with microstructure-dependent neural network material models”. In: *Journal of the Mechanics and Physics of Solids* 196 (2025), p. 106015.

[145] Bernardo P Ferreira and Miguel A Bessa. “Automatically Differentiable Model Updating (ADiMU): conventional, hybrid, and neural network material model discovery including history-dependency”. In: *arXiv preprint arXiv:2505.07801* (2025).

[146] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. “Deep Bayesian Active Learning with Image Data”. In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 1183–1192.

[147] Jiaxiang Yi, Ji Cheng, and Miguel A. Bessa. *Practical multi-fidelity machine learning: fusion of deterministic and Bayesian models*. 2024. arXiv: 2407.15110 [cs.LG].

[148] Jiaxiang Yi and Miguel A. Bessa. *Cooperative Bayesian and variance networks disentangle aleatoric and epistemic uncertainties*. 2025. arXiv: 2505.02743 [cs.LG].

[149] Apostolos F. Psaros et al. “Uncertainty quantification in scientific machine learning: Methods, metrics, and comparisons”. In: *Journal of Computational Physics* 477 (2023), p. 111902. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2022.111902.

[150] Miguel A. Bessa, Piotr Glowacki, and Michael Houlder. “Bayesian Machine Learning in Metamaterial Design: Fragile Becomes Supercompressible”. In: *Advanced Materials* 31.48 (2019), p. 1904845. DOI: https://doi.org/10.1002/adma.201904845.

[151] Dongil Shin et al. “Spiderweb Nanomechanical Resonators via Bayesian Optimization: Inspired by Nature and Guided by Machine Learning”. In: *Advanced Materials* n/a.n/a (), p. 2106248. DOI: https://doi.org/10.1002/adma.202106248.

[152] Yan Wang and David L. McDowell. “Uncertainty quantification in materials modeling”. In: *Uncertainty Quantification in Multiscale Materials Modeling*. Ed. by Yan Wang and David L. McDowell. Elsevier Series in Mechanics of Advanced Materials. Woodhead Publishing, 2020, pp. 1–40. ISBN: 978-0-08-102941-1. DOI: https://doi.org/10.1016/B978-0-08-102941-1.00001-8.

[153] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*. Vol. 2. 3. MIT press Cambridge, MA, 2006.

[154] Theodore Papamarkou et al. “Position: Bayesian deep learning is needed in the age of large-scale AI”. In: *arXiv preprint arXiv:2402.00809* (2024).

[155] George D. Pasparakis, Lori Graham-Brady, and Michael D. Shields. *Bayesian neural networks for predicting uncertainty in full-field material response*. 2024. arXiv: 2406.14838 [stat.ML].

[156] Kevin Linka, Gerhard A Holzapfel, and Ellen Kuhl. “Discovering uncertainty: Bayesian constitutive artificial neural networks”. In: *Computer Methods in Applied Mechanics and Engineering* 433 (2025), p. 117517.

[157] Sotirios P Chatzis. “Sparse Bayesian recurrent neural networks”. In: *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2015, Porto, Portugal, September 7-11, 2015, Proceedings, Part II 15*. Springer. 2015, pp. 359–372.

[158] Meire Fortunato, Charles Blundell, and Oriol Vinyals. *Bayesian Recurrent Neural Networks*. 2017. DOI: 10.48550/ARXIV.1704.02798.

[159] Dario Coscia et al. “BARNN: A Bayesian Autoregressive and Recurrent Neural Network”. In: *arXiv preprint arXiv:2501.18665* (2025).

[160] Clyde Fare et al. “A multi-fidelity machine learning approach to high throughput materials screening”. In: *npj Computational Materials* 8.1 (2022), p. 257.

[161] Ji Cheng, Qiao Lin, and Jiaxiang Yi. “An enhanced variable-fidelity optimization approach for constrained optimization problems and its parallelization”. In: *Structural and Multidisciplinary Optimization* 65.7 (2022), p. 188.

[162] L. V. Kantorovich. “Mathematical Methods of Organizing and Planning Production”. In: *Management Science* 6.4 (1960), pp. 366–422. DOI: 10.1287/mnsc.6.4.366. eprint: https://doi.org/10.1287/mnsc.6.4.366.

[163] Laurens Sluijterman, Eric Cator, and Tom Heskes. “How to evaluate uncertainty estimates in machine learning for regression?” In: *Neural Networks* 173 (May 2024), p. 106203. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2024.106203.

[164] Monika Dokl et al. “Global projections of plastic use, end-of-life fate and potential changes in consumption, reduction, recycling and replacement with bioplastics to 2050”. In: *Sustainable Production and Consumption* 51 (2024), pp. 498–518.

[165] Jefferson Hopewell, Robert Dvorak, and Edward Kosior. “Plastics recycling: challenges and opportunities”. In: *Philosophical Transactions of the Royal Society B: Biological Sciences* 364.1526 (2009), pp. 2115–2126.

[166] Hannah Ritchie, Veronika Samborska, and Max Roser. “Plastic pollution”. In: *Our world in data* (2023).

[167] Cristina Moliner, Giovanni Pasquale, and Elisabetta Arato. “Municipal Plastic Waste Recycling through Pyrogasification”. In: *Energies* 17.5 (2024), p. 1206.

[168] MM Harussani et al. “Pyrolysis of polypropylene plastic waste into carbonaceous char: Priority of plastic waste management amidst COVID-19 pandemic”. In: *Science of The Total Environment* 803 (2022), p. 149911.

[169] Mădălina Elena Grigore. “Methods of recycling, properties and applications of recycled thermoplastic polymers”. In: *Recycling* 2.4 (2017), p. 24.

[170] Md Nuruzzaman et al. “Composite materials from waste plastics: A sustainable approach for waste management and resource utilization”. In: *Polymers and Polymer Composites* 33 (2025), p. 09673911251318542.

[171] James M Eagan et al. *Combining polyethylene and polypropylene: Enhanced performance with PE/iPP multiblock polymers*. 6327. 2017, pp. 814–816.

[172] Jun Xu et al. “Compatibilization of Isotactic Polypropylene (iPP) and High-Density Polyethylene (HDPE) with iPP–PE Multiblock Copolymers”. In: *Macromolecules* 51.21 (2018), pp. 8585–8596. DOI: 10.1021/acs.macromol.8b01907.

[173] Zoé OG Schyns and Michael P Shaver. “Mechanical recycling of packaging plastics: a review”. In: *Macromolecular rapid communications* 42.3 (2021), p. 2000415.

[174] Andreza Salles Barone et al. “Rethinking single-use plastics: Innovations, polices, consumer awareness and market shaping biodegradable solutions in the food packaging industry”. In: *Trends in Food Science & Technology* (2025), p. 104906.

[175] Moritz Kranzlein et al. “One-Step Radical-Induced Synthesis of Graft Copolymers for Effective Compatibilization of Polyethylene and Polypropylene”. In: *Journal of the American Chemical Society* 147.22 (2025), pp. 19052–19060.

[176] Gui-Fang Shan et al. “Mechanical properties and morphology of LDPE/PP blends”. In: *Journal of Macromolecular Science, Part B: Physics* 46.5 (2007), pp. 963–974.

[177] Antimo Graziano, Shaffiq Jaffer, and Mohini Sain. “Review on modification strategies of polyethylene/polypropylene immiscible thermoplastic polymer blends for enhancing their mechanical behavior”. In: *Journal of elastomers & plastics* 51.4 (2019), pp. 291–336.

[178] Amar K. Mohanty et al. “Composites from renewable and sustainable resources: Challenges and innovations”. In: *Science* 362.6414 (2018), pp. 536–542. DOI: 10.1126/science.aat9072.

[179] S.M. Mirkhalaf, F.M. Andrade Pires, and R. Simoes. “An elasto-viscoplastic constitutive model for polymers at finite strains: Formulation and computational aspects”. In: *Computers and Structures* 166 (2016), pp. 60–74. ISSN: 0045-7949. DOI: https://doi.org/10.1016/j.compstruc.2016.01.002.

[180] Joakim Johnsen et al. “A thermo-elasto-viscoplastic constitutive model for polymers”. In: *Journal of the Mechanics and Physics of Solids* 124 (2019), pp. 681–701. ISSN: 0022-5096. DOI: https://doi.org/10.1016/j.jmps.2018.11.018.

[181] Xiaolei Yin et al. “Statistical volume element method for predicting microstructure–constitutive property relations”. In: *Computer methods in applied mechanics and engineering* 197.43-44 (2008), pp. 3516–3529.

[182] Sarah David Müzel et al. “Application of the Finite Element Method in the Analysis of Composite Materials: A Review”. In: *Polymers* 12.4 (2020). ISSN: 2073-4360. DOI: 10.3390/polym12040818.

[183] Lallit Anand et al. “A thermo-mechanically coupled theory for large deformations of amorphous polymers. Part I: Formulation”. In: *International Journal of Plasticity* 25.8 (2009), pp. 1474–1494. ISSN: 0749-6419. DOI: https://doi.org/10.1016/j.ijplas.2008.11.004.

[184] Joakim Johnsen et al. “Experimental set-up for determination of the large-strain tensile behaviour of polymers at low temperatures”. In: *Polymer Testing* 53 (2016), pp. 305–313. ISSN: 0142-9418. DOI: https://doi.org/10.1016/j.polymertesting.2016.06.011.

[185] Bernardo P. Ferreira, A. Francisca Carvalho Alves, and F.M. Andrade Pires. "An efficient finite strain constitutive model for amorphous thermoplastics: Fully implicit computational implementation and optimization-based parameter calibration". In: *Computers & Structures* 281 (2023), p. 107007. ISSN: 0045-7949. DOI: https://doi.org/10.1016/j.compstruc.2023.107007.

[186] Amir Bashirgonbadi et al. "Accurate determination of polyethylene (PE) and polypropylene (PP) content in polyolefin blends using machine learning-assisted differential scanning calorimetry (DSC) analysis". In: *Polymer Testing* 131 (2024), p. 108353.

[187] Hiwa A HamaSalih, Rzgar M Abdalrahman, and Sarkawt Rostam. "Optimizing the blending ratio and processing parameters for ternary blends of recycled polypropylene with recycled high and virgin linear low-densities polyethylene". In: *Results in Engineering* 18 (2023), p. 101171.

[188] S Vervoort et al. "Compatibilization of polypropylene–polyethylene blends". In: *Polymer Engineering & Science* 58.4 (2018), pp. 460–465.

[189] Angela Marotta et al. "Tuning the morphology of HDPE/PP/PET ternary blends by nanoparticles: a simple way to improve the performance of mixed recycled plastics". In: *Polymers* 14.24 (2022), p. 5390.

[190] Nenglong Yang et al. "Effect of micro-scale fibre uncertainties on the mechanical behaviour of natural/synthetic hybrid fibre composites". In: *Composites Part A: Applied Science and Manufacturing* 188 (2025), p. 108570.

[191] Chun-Teh Chen and Grace X. Gu. "Machine learning for composite materials". In: *MRS Communications* 9.2 (2019), pp. 556–566. DOI: 10.1557/mrc.2019.32.

[192] Nagababu Andraju et al. "Machine-learning-based predictions of polymer and postconsumer recycled polymer properties: A comprehensive review". In: *ACS Applied Materials & Interfaces* 14.38 (2022), pp. 42771–42790.

[193] Xin Liu et al. "A review of artificial neural networks in the constitutive modeling of composite materials". In: *Composites Part B: Engineering* 224 (2021), p. 109152.

[194] Baris Caglar et al. "Deep learning accelerated prediction of the permeability of fibrous microstructures". In: *Composites Part A: Applied Science and Manufacturing* 158 (2022), p. 106973.

[195] N Mentges, B Dashtbozorg, and SM Mirkhalaf. "A micromechanics-based artificial neural networks model for elastic properties of short fiber composites". In: *Composites Part B: Engineering* 213 (2021), p. 108736.

[196] Donghyun Shin et al. "A Deep Material Network Approach for Predicting the Thermomechanical Response of Composites". In: *Composites Part B: Engineering* 272 (2024), p. 111177.

[197] Xiaomeng Wang et al. "Machine learning empowered failure criterion of fiber-reinforced polymer composite". In: *Engineering Structures* 334 (2025), p. 120217.

[198] Hon Lam Cheung, Petter Uvdal, and Mohsen Mirkhalaf. "Augmentation of scarce data—A new approach for deep-learning modeling of composites". In: *Composites Science and Technology* 249 (2024), p. 110491.

[199] Aanchna Sharma et al. “Advances in computational intelligence of polymer composite materials: machine learning assisted modeling, analysis and design”. In: *Archives of Computational Methods in Engineering* 29.5 (2022), pp. 3341–3385.

[200] Audrey Olivier, Michael D Shields, and Lori Graham-Brady. “Bayesian neural networks for uncertainty quantification in data-driven materials modeling”. In: *Computer methods in applied mechanics and engineering* 386 (2021), p. 114079.

[201] George D Pasparakis, Lori Graham-Brady, and Michael D Shields. “Bayesian neural networks for predicting uncertainty in full-field material response”. In: *Computer Methods in Applied Mechanics and Engineering* 433 (2025), p. 117486.

[202] Jiaxiang Yi and Miguel A Bessa. “Cooperative Bayesian and variance networks disentangle aleatoric and epistemic uncertainties”. In: *arXiv preprint arXiv:2505.02743* (2025).

[203] Jiaxiang Yi, Bernardo P. Ferreira, and Miguel A. Bessa. *Single- to multi-fidelity history-dependent learning with uncertainty quantification and disentanglement: application to data-driven constitutive modeling*. 2025. arXiv: 2507.13416 [cs.LG].

[204] MP van der Schelling, BP Ferreira, and MA Bessa. “f3dasm: Framework for data-driven design and analysis of structures and materials”. In: *Journal of Open Source Software* 9.100 (2024), p. 6912.

[205] Zhong-xiang Gui, Xiao Hu, and Zi-jian Wang. “An elasto-visco-plastic constitutive model of polypropylene incorporating craze damage behavior and its validation”. In: *Journal of Central South University* 24.6 (2017), pp. 1263–1268.

[206] Nadia Temimi-Maaref, Alain Burr, and Noëlle Billon. “Damaging processes in polypropylene compound: Experiment and modeling”. In: *Polymer Science Series A* 50 (2008), pp. 558–567.

[207] Tim B Van Erp et al. “Prediction of yield and long-term failure of oriented polypropylene: Kinetics and anisotropy”. In: *Journal of Polymer Science Part B: Polymer Physics* 47.20 (2009), pp. 2026–2035.

[208] Ditho Pulungan et al. “Characterizing and modeling the pressure-and rate-dependent elastic-plastic-damage behavior of polypropylene-based polymers”. In: *Polymer Testing* 68 (2018), pp. 433–445.

[209] Anouar Krairi and Issam Doghri. “A thermodynamically-based constitutive model for thermoplastic polymers coupling viscoelasticity, viscoplasticity and ductile damage”. In: *International Journal of Plasticity* 60 (2014), pp. 163–181.

[210] Dan-Andrei Şerban, Liviu Marşavina, and Niels Modler. “Finite element modelling of the progressive damage and failure of thermoplastic polymers in puncture impact”. In: *Procedia Engineering* 109 (2015), pp. 97–104.

[211] Bo Yin and Michael Kaliske. “Fracture simulation of viscoelastic polymers by the phase-field method”. In: *Computational Mechanics* 65.2 (2020), pp. 293–309.

[212] Sascha Fliegener, Tobias Kennerknecht, and Matthias Kabel. “Investigations into the damage mechanisms of glass fiber reinforced polypropylene based on micro specimens and precise models of their microstructure”. In: *Composites part B: engineering* 112 (2017), pp. 327–343.

[213] Carlos González and Javier LLorca. “Mechanical behavior of unidirectional fiber-reinforced polymers under transverse compression: Microscopic mechanisms and modeling”. In: *Composites Science and Technology* 67.13 (2007), pp. 2795–2806.

[214] Essam Totry, Carlos González, and Javier LLorca. “Prediction of the failure locus of C/PEEK composites under transverse compression and longitudinal shear through computational micromechanics”. In: *Composites Science and Technology* 68.15-16 (2008), pp. 3128–3136.

[215] A Sharma et al. “On the prediction of the bi-axial failure envelope of a UD CFRP composite lamina using computational micromechanics: Effect of microscale parameters on macroscale stress–strain behavior”. In: *Composite Structures* 251 (2020), p. 112605.

[216] Luis P Canal, Javier Segurado, and Javier LLorca. “Failure surface of epoxy-modified fiber-reinforced composites under transverse tension and out-of-plane shear”. In: *International journal of solids and structures* 46.11-12 (2009), pp. 2265–2274.

[217] AR Melro et al. “Micromechanical analysis of polymer composites reinforced by unidirectional fibres: Part I–Constitutive modelling”. In: *International Journal of Solids and Structures* 50.11-12 (2013), pp. 1897–1905.

[218] Fernando Naya et al. “Computational micromechanics of the transverse and shear behavior of unidirectional fiber reinforced polymers including environmental effects”. In: *Composites Part A: Applied Science and Manufacturing* 92 (2017), pp. 146–157.

[219] M Rezasefat et al. “A hybrid micro-macro mechanical damage model to consider the influence of resin-rich zones on the transverse tensile behaviour of unidirectional composites”. In: *Composite Structures* 308 (2023), p. 116714.

[220] Lei Yang et al. “Microscopic failure mechanisms of fiber-reinforced polymer composites under transverse tension and compression”. In: *Composites Science and Technology* 72.15 (2012), pp. 1818–1825.

[221] Seno Jose et al. “Phase morphology, crystallisation behaviour and mechanical properties of isotactic polypropylene/high density polyethylene blends”. In: *European polymer journal* 40.9 (2004), pp. 2105–2115.

[222] Max Kukkola. “Offline stage acceleration of the self-consistent clustering analysis method: towards real-time material predictions”. In: (2024).

[223] A.R. Melro, P.P. Camanho, and S.T. Pinho. “Generation of random distribution of fibres in long-fibre reinforced composites”. In: *Composites Science and Technology* 68.9 (2008), pp. 2092–2102. ISSN: 0266-3538. DOI: https://doi.org/10.1016/j.compscitech.2008.03.013.

[224] L D'orazio et al. "Effect of the addition of ethylene-propylene random copolymers on the properties of high-density polyethylene/isotactic polypropylene blends: Part 1—morphology and impact behavior of molded samples". In: *Polymer Engineering & Science* 22.9 (1982), pp. 536–544.

[225] Yijian Lin et al. "Adhesion of olefin block copolymers to polypropylene and high density polyethylene and their effectiveness as compatibilizers in blends". In: *Polymer* 52.7 (2011), pp. 1635–1644.

[226] Dipl-Ing Erdal Karaagac. "Blends of post-consumer recycled polypropylene (PP)/polyethylene (PE): rheology, morphology, and mechanics". PhD thesis. Institute of Materials Science, 2021.

[227] SIMULIA. *ABAQUS User's Manual, Version 2022*. Dassault Systèmes Simulia Corp, 2024.

[228] Zdeněk P. Bažant and B H Oh. "Crack band theory for fracture of concrete". In: *Matériaux et Construction* 16 (3 1983), pp. 155–177. DOI: 10.1007/BF02486267.

[229] DR Mears, KD Pae, and JA Sauer. "Effects of hydrostatic pressure on the mechanical behavior of polyethylene and polypropylene". In: *Journal of Applied Physics* 40.11 (1969), pp. 4229–4237.

[230] KD Pae and SK Bhateja. "The effects of hydrostatic pressure on the mechanical behavior of polymers". In: *Journal of Macromolecular Science—Reviews in Macromolecular Chemistry* 13.1 (1975), pp. 1–75.

[231] Alex M Jordan et al. "Role of crystallization on polyolefin interfaces: an improved outlook for polyolefin blends". In: *Macromolecules* 51.7 (2018), pp. 2506–2516.

[232] Ch Tselios et al. "In situ compatibilization of polypropylene–polyethylene blends: a thermomechanical and spectroscopic study". In: *Polymer* 39.26 (1998), pp. 6807–6817.

[233] Nina Vranjes Penava, Vesna Rek, and Ivona Fiamengo Houra. "Effect of EPDM as a compatibilizer on mechanical properties and morphology of PP/LDPE blends". In: *Journal of Elastomers & Plastics* 45.4 (2013), pp. 391–403.

[234] *MS Windows NT LyondellBasell Industries Holdings*. https://www.lyondellbasell.com/. Accessed: 2025-02-13.

[235] ISO. *Determination of the mode II fracture resistance for unidirectionally reinforced materials using the calibrated end-loaded split (C-ELS) est and an effective crack length approach. ISO 15114:2014*. 2011.

[236] S Muke et al. "The melt extensibility of polypropylene". In: *Polymer international* 50.5 (2001), pp. 515–523.

[237] EE Ferg and LL Bolo. "A correlation between the variable melt flow index and the molecular mass distribution of virgin and recycled polypropylene used in the manufacturing of battery cases". In: *Polymer Testing* 32.8 (2013), pp. 1452–1459.

[238] J Teh, P Lam, and C Dobbin. "Prediction of melt rheological properties from GPC molecular weights". In: *Polymer Testing* 47 (2015), pp. 101–112.

[239] Kejie Xu, Yuan Wen, and Xiangbin Xu. “Melt flow ratio: A way to identify the type of polyethylene”. In: *Advanced Industrial and Engineering Polymer Research* 6.1 (2023), pp. 79–82.

[240] Changing Fang et al. “Characterization of polypropylene–polyethylene blends made of waste materials with compatibilizer and nano-filler”. In: *Composites Part B: Engineering* 55 (2013), pp. 498–505.

[241] Geng Han et al. “Microscopic progressive damage simulation and scale-span analysis of cross-ply laminate based on the elastic–plastic theory”. In: *Applied Composite Materials* 22 (2015), pp. 1–12.

[242] D Garoz et al. “Consistent application of periodic boundary conditions in implicit and explicit finite element simulations of damage in composites”. In: *Composites Part B: Engineering* 168 (2019), pp. 254–266.

[243] E. V. González et al. “Simulating drop-weight impact and compression after impact tests on composite laminates using conventional shell finite elements”. In: *International Journal of Solids and Structures* 144-145 (July 2018), pp. 230–247. ISSN: 00207683. DOI: 10.1016/j.ijsolstr.2018.05.005.

[244] Ivan Ruiz Cozar. “Development of constitutive models for the accurate simulation of advanced polymer-based composites under complex loading states”. PhD thesis. Universitat de Girona, 2024.

[245] Antonio R Melro and Riccardo Manno. “Microscale representative volume element: generation and statistical characterization”. In: *Multi-Scale Continuum Mechanics Modelling of Fibre-Reinforced Polymer Composites*. Elsevier, 2021, pp. 31–54.

[246] ISO. *Plastics — Determination of tensile properties*. 2019.

[247] Chuntao Zhang and Ian D Moore. “Nonlinear mechanical response of high density polyethylene. Part I: Experimental investigation and model evaluation”. In: *Polymer Engineering & Science* 37.2 (1997), pp. 404–413.

[248] M Khalajmasoumi et al. “Hyperelastic analysis of high density polyethylene under monotonic compressive load”. In: *Applied Mechanics and Materials* 229 (2012), pp. 309–313.

[249] Rocio Seltzer et al. “Determination of the Drucker–Prager parameters of polymers exhibiting pressure-sensitive plastic behaviour by depth-sensing indentation”. In: *International Journal of Mechanical Sciences* 53.6 (2011), pp. 471–478.

[250] Manoj Kumar, Kumresh Kumar Gaur, and Chandra Shakher. “Measurement of material constants (Young’s modulus and Poisson’s ratio) of polypropylene using digital speckle pattern interferometry (DSPI)”. In: *Journal of the Japanese Society for Experimental Mechanics* 15.Special_Issue (2015), s87–s91.

[251] Thomas Parenteau et al. “Structure, mechanical properties and modelling of polypropylene for different degrees of crystallinity”. In: *Polymer* 53.25 (2012), pp. 5873–5884.

[252] Junbiao Lai. “Non-linear time-dependent deformation behaviour of high-density polyethylene.” In: (1996).

[253] A. Turon et al. “A damage model for the simulation of delamination in advanced composites under variable-mode loading”. In: *Mechanics of Materials* 38.11 (2006), pp. 1072–1089. ISSN: 0167-6636. DOI: https://doi.org/10.1016/j.mechmat.2005.10.003.

[254] Donald R. Jones, Matthias Schonlau, and William J. Welch. “Efficient Global Optimization of Expensive Black-Box Functions”. In: *Journal of Global Optimization* 13.4 (1998), pp. 455–492. ISSN: 1573-2916. DOI: 10.1023/A:1008306431147.

[255] Sushant S. Garud, Iftekhar A. Karimi, and Markus Kraft. “Design of computer experiments: A review”. In: *Computers & Chemical Engineering* 106 (2017). ESCAPE-26, pp. 71–95. ISSN: 0098-1354. DOI: https://doi.org/10.1016/j.compchemeng.2017.05.010.

[256] Rainer Storn and Kenneth Price. “Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces”. In: *Journal of global optimization* 11.4 (1997), pp. 341–359.

[257] Roman Garnett. *Bayesian Optimization*. Cambridge University Press, 2023.

[258] Wing Kam Liu, Shaofan Li, and Harold S. Park. “Eighty Years of the Finite Element Method: Birth, Evolution, and Future”. In: *Archives of Computational Methods in Engineering* 29.6 (2022), pp. 4431–4453. ISSN: 1886-1784. DOI: 10.1007/s11831-022-09740-9.

[259] Chun-Teh Chen and Grace X. Gu. “Machine learning for composite materials”. In: *MRS Communications* 9.2 (2019), pp. 556–566. DOI: 10.1557/mrc.2019.32.

[260] Nolan Black and Ahmad R. Najafi. “Learning finite element convergence with the Multi-fidelity Graph Neural Network”. In: *Computer Methods in Applied Mechanics and Engineering* 397 (2022), p. 115120. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2022.115120.

[261] Matthew W. Scroggs et al. “Construction of Arbitrary Order Finite Element Degree-of-Freedom Maps on Polygonal and Polyhedral Cell Meshes”. In: *ACM Transactions on Mathematical Software* 48.2 (May 2022), pp. 1–23. DOI: 10.1145/3524456.

[262] Tianju Xue et al. “JAX-FEM: A differentiable GPU-accelerated 3D finite element solver for automatic inverse design and mechanistic data science”. In: *Computer Physics Communications* 291 (2023), p. 108802. ISSN: 0010-4655. DOI: https://doi.org/10.1016/j.cpc.2023.108802.

[263] *ABAQUS: FINITE ELEMENT ANALYSIS FOR MECHANICAL ENGINEERING AND CIVIL ENGINEERING*. https://www.3ds.com/products-services/simulia/products/abaqus/. W3C WebRTC Working Group.

[264] R. von Mises. “Mechanik der festen Körper im plastisch-deformablen Zustand”. In: *Göttin. Nachr. Math. Phys.* 1 (1913), pp. 582–592.

[265] Dassault Systèmes. *Abaqus 2024*. Computer software. 2024.

[266] S. Surjanovic and D. Bingham. *Virtual Library of Simulation Experiments: Test Functions and Datasets*. Retrieved December 8, 2023, from http://www.sfu.ca/~ssurjano.

[267] Ping Jiang et al. “Variable-Fidelity Lower Confidence Bounding Approach for Engineering Optimization Problems with Expensive Simulations”. In: *AIAA Journal* 57.12 (2019), pp. 5416–5430. DOI: 10.2514/1.J058283. eprint: https://doi.org/10.2514/1.J058283.

[268] Sander van Rijn and Sebastian Schmitt. “MF2: A Collection of Multi-Fidelity Benchmark Functions in Python”. In: *Journal of Open Source Software* 5.52 (2020), p. 2049. DOI: 10.21105/joss.02049.

[269] L. Mainini et al. *Analytical Benchmark Problems for Multifidelity Optimization Methods*. 2022. DOI: 10.48550/ARXIV.2204.07867.

[270] Rodney Hill. “Elastic properties of reinforced solids: some theoretical principles”. In: *Journal of the Mechanics and Physics of Solids* 11.5 (1963), pp. 357–372.

[271] Bernardo P. Ferreira, F. M. Andrade Pires, and Miguel A. Bessa. “Adaptive Clustering-based Reduced-Order Modeling Framework: Fast and accurate modeling of localized history-dependent phenomena”. In: (2021). arXiv: 2109.11897 [math.NA].

[272] Laurent Valentin Jospin et al. “Hands-On Bayesian Neural Networks—A Tutorial for Deep Learning Users”. In: *IEEE Computational Intelligence Magazine* 17.2 (2022), pp. 29–48. DOI: 10.1109/MCI.2022.3155327.

[273] SM Mirkhalaf, FM Andrade Pires, and Ricardo Simoes. “Determination of the size of the Representative Volume Element (RVE) for the simulation of heterogeneous polymers at finite strains”. In: *Finite Elements in Analysis and Design* 119 (2016), pp. 30–44.

CURRICULUM VITÆ

Jiaxiang YI

28-06-1996 Born in Jiangxi, China.

EDUCATION

2014.09–2018.06 BSc in Naval Architecture and Ocean Engineering
Huazhong University of Science and Technology, China

2018.09–2021.06 MSc in Design and Construction of Naval Architecture and Ocean Structure
Huazhong University of Science and Technology, China

2021.10–2026.01 PhD. Mechanical Engineering
Delft University of Technology, the Netherlands

ACKNOWLEDGEMENTS

With the completion of this PhD dissertation, my doctoral journey is approaching its end. I would like to take this opportunity to express my sincere gratitude to all those who have supported and accompanied me throughout this incredible yet challenging experience.

First of all, I would like to thank my promotor, Dr. Miguel Bessa, for giving me the opportunity to work with him. I am deeply grateful for your patience in helping me improve my communication skills. Your kind support, both in research and in my well-being, has been invaluable throughout this journey. Your trust, rigorous academic attitude, and high expectations have been a constant source of motivation for my research. Although we chose a highly challenging topic, your guidance and encouragement enabled me to carry it through to completion. I am also sincerely grateful to my promotor, Dr. ir. Marcel Sluiter, for always offering new perspectives that encouraged me to step outside my own comfort zone of thinking. I also greatly appreciate your support in handling administrative matters. I would also like to thank my copromotor, Dr. Baris Caglar for the constructive feedback, for always listening to my thoughts, and for providing me with platforms to practice and improve my presentation skills.

I want to thank all my colleagues in the Bessa Research Group. Four years of companionship, academic discussions, and joint activities have created many wonderful memories that I will always cherish. In particular, I would like to thank Bruno. I still remember the early days we spent together practicing communication skills and the trips we shared, which remain among my fondest memories. I am also grateful to Martin for introducing me to Dutch culture and helping me better understand the country. I also thank Aleksandr for his listening ear and helpful suggestions whenever I was stuck. I thank Hari and Surya for the nice Indian food and our morning badminton sessions. I would also like to thank my co-authors, Aleksandr, Bernardo, and Ivan, for the fruitful collaborations and their valuable contributions to our joint publications.

I am thankful to my friends Xiaohui, Zhaoying, Jianing, Dingshan, Keer, Ziyu, Kai, Shushu, Yaqi, Jiaqi, Yan, Zixiong, Ziqing, Xinhai, Stella, Sifeng, Yang, Mingkai, Na, and Zhen for the time we spent together enjoying food. I am grateful to my basketball mates Guofeng, Wenjie, Jingwei, Jiandong, Lifei, Lexin, and Jinlai for their companionship and the joyful moments on the court. I am especially grateful to Ji, Rong, and Sipei, whose discussions and support have greatly enriched my PhD journey.

To my girl friend, Yiquan, you are the light of my life and thank you for always being here for me. Finally, I owe my deepest gratitude to my parents, whose unconditional love and unwavering support have been the foundation of my perseverance throughout this journey.

Jiaxiang Yi
Delft, January 2026

List of Publications

Journal Papers and Preprints

5. **J. Yi**, I. R. Cózar, B. Caglar, and M. A. Bessa, *Scalable Bayesian machine learning for sustainable composite design: uncertainty-aware prediction and optimization of recycled polypropylene/polyethylene blends*, In preparation.

4. **J. Yi**, B. P. Ferreira, and M. A. Bessa, *Single-to-multi-fidelity history-dependent learning with uncertainty quantification and disentanglement: application to data-driven constitutive modeling*, *Computer Methods in Applied Mechanics and Engineering*, 118479, (2025).

3. **J. Yi**, and M. A. Bessa, *Cooperative Bayesian and variance networks disentangle aleatoric and epistemic uncertainties*, arXiv preprint arXiv:2505.02743, (2025).

2. **J. Yi**, J. Cheng, and M. A. Bessa, *Practical multi-fidelity machine learning: fusion of deterministic and Bayesian models*, arXiv preprint arXiv:2407.15110, (2024).

1. A. Dekhovich, O. T. Turan, **J. Yi**, and M. A. Bessa, *Cooperative data-driven modeling*, *Computer Methods in Applied Mechanics and Engineering*, 116432, (2023).

Conference Proceedings

2. **J. Yi**, and M. A. Bessa, *rvesimulator: An automated representative volume element simulator for data-driven material discovery*, AI for Accelerated Materials Design – NeurIPS 2023 Workshop (2023).

1. A. Dekhovich, O. T. Turan, **J. Yi**, and M. A. Bessa, *Cooperative data-driven modeling: continual learning of different material behavior*, Workshop on "Machine Learning for Materials", ICLR 2023.

Included in this thesis.