Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 386 (2021) 114079
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

Bayesian neural networks for uncertainty quantification in data-driven materials modeling

Audrey Olivier[a,*,1], Michael D. Shields[b], Lori Graham-Brady[b]

[a] Sonny Astani Department of Civil and Environmental Engineering, University of Southern California, United States of America
[b] Department of Civil and Systems Engineering, Johns Hopkins University, United States of America

Received 4 March 2021; received in revised form 16 July 2021; accepted 23 July 2021
Available online 18 August 2021

Abstract

Modern machine learning (ML) techniques, in conjunction with simulation-based methods, present remarkable potential for various scientific and engineering applications. Within the materials science field, these data-based methods can be used to build efficient structure–property linkages that can be further integrated within multi-scale simulations, or guide experiments in a materials discovery setting. However, a critical shortcoming of state-of-the-art ML techniques is their lack of reliable uncertainty/error estimates, which severely limits their use for materials or other engineering applications where data is often scarce and uncertainties are substantial. This paper presents methods for Bayesian learning of neural networks (NN) that allow consideration of both aleatoric uncertainties that account for the inherent stochasticity of the data-generating process, and epistemic uncertainties, which arise from consideration of limited amounts of data. In particular, algorithms based on approximate variational inference and (pseudo-)Bayesian model averaging achieve an appropriate trade-off between accuracy of the uncertainty estimates and accessible computational cost. The performance of these algorithms is first presented on simple 1D examples to illustrate their behavior in both extrapolation and interpolation settings. The approach is then applied for the prediction of homogenized and localized properties of a composite material. In this setting, data is generated from a finite element model, which permits a study of the behavior of the probabilistic learning algorithms under various amounts of aleatoric and epistemic uncertainties.

Keywords: Data-driven materials modeling; Bayesian neural network; Variational inference; Probabilistic model averaging; Epistemic and aleatoric uncertainties

1. Introduction

1.1. Machine learning in materials modeling and design

Data-based models are becoming extremely popular to approximate process–structure–property (PSP) relationships from data collected from experiments or computational simulations [1–5]. Applications of ML to property prediction include, among many, prediction of formation energies for various materials compositions [6], prediction

* Corresponding author.
E-mail address: audreyol@usc.edu (A. Olivier).

1 This work was initiated at Johns Hopkins University and finalized at Columbia University.

https://doi.org/10.1016/j.cma.2021.114079
0045-7825/© 2021 Elsevier B.V. All rights reserved.

of melting temperature for various compounds [7], and prediction of mechanical properties of metal alloys [8]. Data-centric models have been developed that serve as surrogates, replacing costly simulations, for both homogenization (communication of information from a lower length scale to a higher length scale), e.g. [9], and localization linkages (communication from a higher length scale to a lower length scale), e.g. [1,10]. These data-based models are cheap to evaluate on new data, and can thus be leveraged in inverse design for materials discovery as they allow the search over many possible material compositions [6], or efficient methods of optimization to find materials that maximize a given property of interest [7,11]. These efficient surrogate models also enable expensive UQ tasks such as global sensitivity analysis [11,12].

Many of the previously cited works make use of 'traditional' machine learning algorithms to build a linkage between a set of features or 'predictors' and the macro property of interest; algorithms such as decision trees and random forests [6,13], support vector regressors [14], and shallow neural networks (NNs) [8,15]. Application of 'traditional' ML algorithms require an important first step of feature extraction from the raw data [10,16]. Recently, attention has also risen for the use of deep learning algorithms [17] within the materials modeling community, as they are capable of automatically discovering the needed representation from raw data, e.g., images of microstructures; somewhat alleviating the need for hand-designed feature extraction/selection [18]. Deep learning algorithms are showing great promise in a variety of fields in engineering and science; physics-informed deep learning can be used to solve and discover partial differential equations [19,20], deep neural networks can serve as surrogate models enabling efficient optimization and design [21] or expensive UQ tasks [22]. In materials applications in particular, deep learning has been applied to problems of microstructure reconstruction [23,24], classification [25], property prediction (homogenization, [18,24]) and field prediction (localization, [26]). Another advantage of neural networks, deep or not, over other machine learning techniques is their applicability to various types of input data: structured data such as conventional predictors, image data – e.g. micrographs – through the use of convolutional neural networks (CNNs) and temporal data through the use of recurrent neural networks (RNNs). Despite the remarkable potential of ML techniques, in particular deep learning algorithms, for materials applications, the direct use of their outputs is limited due to the lack of reliable uncertainty estimates [8].

1.2. Uncertainty quantification within machine learning pipelines

Within science and engineering, the process of quantifying uncertainties is crucial to provide decision makers with predictions along with quantitative information about the level at which these predictions can be trusted. Scientific applications often deal with significant amounts of both aleatoric and epistemic uncertainties [27] (sometimes referred to as reducible vs. irreducible uncertainties, or uncertainty vs. risk in the reinforcement learning literature). Aleatoric uncertainties are those inherent to the underlying phenomena being studied: physical properties of a system may be random, its input excitations may be stochastic, and the associated experimental data is typically very noisy. On the other hand, epistemic uncertainties arise during modeling from lack of knowledge about underlying physical phenomena or, of particular interest for ML applications, lack of data and imperfect models. Some of these epistemic uncertainties can potentially be reduced by gathering more data. Contrary to mainstream ML applications that rely on large amounts of data, scientific applications often deal with small amounts of complex data, thus incurring large epistemic uncertainties. It is then crucial for the modeler to account for both aleatoric and epistemic uncertainties during training of the model and in making predictions.

Many machine learning algorithms including classical neural networks, deep or not, only provide point estimates of the predictions, i.e., they do not quantify uncertainties. A well-known class of ML models that inherently include uncertainty predictions are Gaussian processes (GPs), algorithms that are inherently rooted in a Bayesian learning framework. Integration of neural networks within the Bayesian framework has also been discussed early on in the machine learning community [28], with so-called Bayesian NNs. However, with the growth of the amount of data and network size, BNNs quickly became impractical and have thus not been widely adopted in practice. Lately however, the integration of UQ within neural networks has regained interest both within the machine learning community, with development of reinforcement learning algorithms, as well as within the physical sciences, [24,29–31]. Algorithms that perform Bayesian inference within (deep) neural networks can be roughly partitioned into three groups:

- Bayesian Neural Networks (BNNs) that use Markov Chain Monte Carlo (MCMC) algorithms for inference of $p(\omega|\mathcal{D})$, [32],

- Algorithms that make use of approximate variational inference [33–35],
- Algorithms that estimate uncertainty via ensembling over several networks, using bootstrapping on the data for instance, [36,37].

Approximation of uncertainties via ensembling is popular and applicable for various machine learning models. For instance, random forests are composed of an ensemble of trees, each built using a bootstrap sample of the data. In the context of neural networks, such bootstrapping methods do not allow for incorporation of prior information on the weights, however a method was recently proposed in [37] that combines bootstrapping and randomized prior functions to alleviate this issue. BNNs that make use of MCMC algorithms such as the Hamiltonian Monte Carlo algorithm (HMC, [38]) theoretically yield the true posterior pdf. Recently, [39] applied HMC to physics-informed NNs and reported better performance than algorithms based on dropout or simple VI. However, MCMC algorithms in general do not scale well with the number of parameters or data points, and convergence to the true posterior can be difficult to assess [40].

Variational inference (VI) is a popular alternative to MCMC methods when posterior pdfs cannot be computed analytically and sampling is intractable. VI algorithms make an assumption on the form of the posterior distribution and are thus less accurate than MCMC in their estimation of the posterior, but they are typically faster, they rely on optimization and can thus leverage existing frameworks for NN training, and convergence is easier to assess. Two popular algorithms that use this concept for training neural networks are the BayesByBackprop [33,34] and MC Dropout [35], which has been shown to perform VI under certain assumptions and approximations. In particular, MC Dropout has been used for estimation of uncertainties for problems in the physical sciences [41–43], due in large part to its ease of implementation and low computational cost. This algorithm aims at achieving Bayesian-like behavior by dropping out neurons in the network at random. Despite its apparent good behavior, several authors have noted that this algorithm sometimes does not perform well on simple problems [44,45] and that posterior samples from dropout do not look like any sensible posterior sample [36]. Finally, it is interesting to point out that another methodology that is receiving attention to alleviate some of the issues arising from the over-parameterization of neural networks is to adopt a function-space point of view [46]. In [47] this train of thought is used to train Bayesian NNs via particle optimization in the space of regression functions, while in [48] it is used to define more meaningful priors in function space.

In this paper, we consider algorithms based on variational Bayesian inference for probabilistic training of neural networks from materials data. We highlight some of the weaknesses of VI and study potential enhancements such as the use of different distance metrics between the true posterior and variational approximation and the use of pseudo-Bayesian methods for model averaging. Firstly, the performance of these algorithms is presented on simple 1D examples to illustrate their behavior in both extrapolation and interpolation settings. A materials data example is then presented where a Bayesian neural network is trained to serve as a surrogate of a costly finite element model of a composite material with randomly placed circular fibers. This random placement of fibers induces some aleatoric uncertainty in the data, while the amount of epistemic uncertainty can be controlled by varying the amount of data used for training.

2. Quantifying epistemic uncertainties in neural networks

2.1. BayesIan neural networks for regression with limited data

In the materials modeling context, the process of building a data-based structure–property linkage consists of a supervised regression (or classification) task. Given some input–output ($x \rightarrow y$) data, one wishes to learn a regressor $f(\cdot)$ for prediction of $y^\star = f(x^\star)$ given a new input $x^\star$. The input parameter vector x could be a set of predictors (structured input data), or an image of a microstructure for instance (image input data). The output y represents the targeted property(ies) of interest. Such outputs can also be of various data types: in homogenization tasks the output usually consists of a few continuous variables (structured data), while for localization problems one wants to predict a full stress/strain field over the microstructure, in which case the output itself could be an image.

Neural networks are a type of regressor that consists of multiple layers of simple nonlinear functions, parameterized by a set of weights $\omega \in \mathbb{R}^d$ (also called kernels or biases), where d is the number of weights and depends on the number of nodes in the network. Training a neural network refers to calibrating the value of these weights ω using the available training data $\mathcal{D} = \{\mathbf{x}, \mathbf{y}\} = \{x_i, y_i\}_{i=1:n}$. In traditional learning (Fig. 1, left plot),

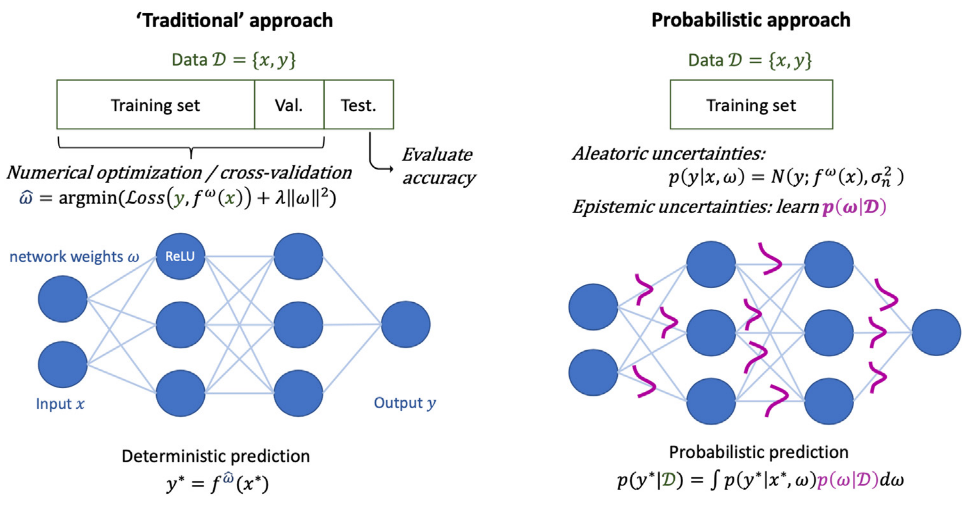

'Traditional' approach
Data $\mathcal{D} = \{x, y\}$
Training set
Val.
Test.
Evaluate accuracy
Numerical optimization / cross-validation
$\hat{\omega} = \text{argmin}(\mathcal{L}oss(y, f^{\omega}(x)) + \lambda\|\omega\|^2)$
network weights ω
ReLU
Input x
Output y
Deterministic prediction
$y^* = f^{\hat{\omega}}(x^*)$
Probabilistic approach
Data $\mathcal{D} = \{x, y\}$
Training set
Aleatoric uncertainties:
$p(y|x,\omega) = N(y; f^{\omega}(x), \sigma_n^2)$
Epistemic uncertainties: learn $p(\omega|\mathcal{D})$
Probabilistic prediction
$p(y^*|\mathcal{D}) = \int p(y^*|x^*,\omega)p(\omega|\mathcal{D})d\omega$

Fig. 1. Traditional deterministic neural networks (left) do not account for uncertainties in model predictions. Bayesian neural networks (right) are designed to account for uncertainties within the data and model.

when no consideration is given to quantification of uncertainties, training consists of minimizing an error function, such as the mean-squared-error between the data y_i and network outputs $f^{\omega}(x_i)$. A regularization term is often added to the loss to penalize overly complex models that tend to overfit the data.

In order to account for aleatoric and epistemic uncertainties during network training, a probabilistic approach must be adopted (Fig. 1, right plot). Firstly, in order to represent aleatoric uncertainties inherently present in real-life systems, one adopts a probability model over the data, for instance the normal distribution[2]

$$p\left(y|x,\omega\right) = \mathcal{N}\left(y;\, f^{\omega}(x), \sigma_n^2\right) \tag{1}$$

having mean $f^{\omega}(x)$ and variance σ_n^2. In the present context, the focus is on quantification of epistemic uncertainties as detailed below, thus it is assumed that the value of the measurement noise variance σ_n^2 is known a priori. However this assumption could easily be relaxed by learning this value as part of the training process [49]. It can also be noted that more complex, e.g. heteroscedastic or non-Gaussian models could be used within this context, by replacing Eq. (1) with an appropriate probabilistic model $p\left(y|x,\omega\right)$. Moreover, as demonstrated later in this paper, it can be advantageous to compare/aggregate several probabilistic models using a model averaging scheme. With the model of Eq. (1), and assuming that the training data points are independent and identically distributed (i.i.d.), the likelihood over the whole data set $\mathcal{D}$ is computed as[3]:

$$p\left(\mathcal{D}|\omega\right) = \prod_{i=1}^{n} \mathcal{N}\left(y_i;\, f^{\omega}(x_i), \sigma_n^2\right) \tag{2}$$

Epistemic uncertainties on the other hand are quantified by learning a probability model over the parameters of the model ω. In a Bayesian setting a prior pdf, $p_0\left(\omega\right)$, is assigned to the parameters, which is equivalent to choosing a space of possible functions $f^{\omega}(\cdot)$, prior to observing any data. This pdf is then updated using the data and Bayes' theorem to yield the posterior pdf $p\left(\omega|\mathcal{D}\right)$ as:

$$p\left(\omega|\mathcal{D}\right) = \frac{p\left(\mathcal{D}|\omega\right) p_0(\omega)}{p(\mathcal{D})} \tag{3}$$

At prediction time, given a new input one can compute the predictive probability of the output as:

$$p\left(y^{\star}|x^{\star},\mathcal{D}\right) = \int p\left(y^{\star}|x^{\star},\omega\right) p\left(\omega|\mathcal{D}\right) d\omega \tag{4}$$

[2] Noise in the dependent variables x is not considered herein, similar to the framework presented in [28].

[3] Strictly speaking, this should be written as $p\left(\mathbf{y}|\mathbf{x},\omega\right)$ since these models do not predict the distribution over the dependent variables x; we use this liberty of notation for simplicity, as in [28].

In particular, using the law of total expectation and total variance one obtains the following mean and uncertainty (variance) predictions:

$$\mathrm{E}\left[y^\star|x^\star,\mathcal{D}\right]=\mathrm{E}_{p(\omega|\mathcal{D})}\left[\mathrm{E}\left[y^\star|x^\star,\omega\right]\right] \tag{5a}$$

$$=\mathrm{E}_{p(\omega|\mathcal{D})}\left[\underbrace{f^\omega(x^\star)}_{\substack{\text{output of NN}\\ \text{with weights }\omega}}\right] \tag{5b}$$

$$\mathrm{Var}\left(y^\star|x^\star,\mathcal{D}\right)=\mathrm{E}_{p(\omega|\mathcal{D})}\left[\mathrm{Var}\left(y^\star|x^\star,\omega\right)\right]+\mathrm{Var}_{p(\omega|\mathcal{D})}\left(\mathrm{E}\left[y^\star|x^\star,\omega\right]\right) \tag{5c}$$

$$=\underbrace{\sigma_n\left(x^\star\right)^2}_{\text{aleatoric uncertainty}}+\underbrace{\mathrm{Var}_{p(\omega|\mathcal{D})}\left(f^\omega(x^\star)\right)}_{\text{epistemic uncertainty}} \tag{5d}$$

Performing a maximum *a posteriori* estimation instead of a full Bayesian estimation, i.e., learning a point estimate for ω as:

$$\hat{\omega}=\underset{w}{\operatorname{argmax}}\quad \ln\left[p\left(\mathcal{D}|\omega\right)p_0\left(\omega\right)\right] \tag{6}$$

is equivalent to standard neural network training procedures. For instance, computing (6) with a homoscedastic probability model and an independent Gaussian prior pdf for the weights is mathematically equivalent to minimizing the mean squared error with weight decay as regularization, a standard NN training procedure for regression problems. Such an optimization procedure largely neglects epistemic uncertainties, which are of primary importance when small amounts of data are considered. In order to accurately quantify epistemic uncertainties within the NN pipelines, the full posterior pdf must be inferred.

Bayesian NNs also better utilize the available data, again a key consideration when dealing with small amounts of data. In 'traditional' learning procedures, hyper-parameters (such as parameters that define the network architecture or the amount of regularization) are often learned via cross-validation, i.e., by dividing the data into a training set and validation set. The latter is then used to evaluate the performance of the various models with different hyper-parameters and choose the best model. Often, another portion of the data, unused at training time, is allocated to evaluating the accuracy of the finalized network. All three partitions of the data should be representative of the overall data set, which becomes more difficult to achieve when dealing with small amounts of data. On the other hand, Bayesian neural networks provide uncertainty estimates around predictions without the need for separate training, validation, and test data sets. Also, instead of choosing a single model (i.e., one set of hyper-parameters) that provides appropriate performance, (pseudo-)Bayesian model averaging can be leveraged to make use of several trained models with different hyper-parameters, weighing them according to their predictive power, as will be shown later in the paper.

Performing Bayesian inference to train neural networks is a cumbersome task as the parameter space is high-dimensional (number of weights $\omega_{i=1:d}$ in the network), rendering exact integration or sampling methods such as MCMC difficult to utilize. In this manuscript we focus on variational inference and potential enhancements.

2.2. *Variational inference for quantifying epistemic uncertainties in neural networks*

2.2.1. *Variational inference: Theory and application to training neural networks*

The idea behind variational inference is to approximate the true posterior $p\left(\omega|\mathcal{D}\right)$ with a variational distribution $q_\theta\left(\omega\right)$, parameterized by θ, whose structure is easy to evaluate. As an example, $q_\theta\left(\cdot\right)$ is often chosen as a factorized Gaussian distribution

$$q_\theta\left(\omega\right)=\prod_{i=1}^{d}\mathcal{N}\left(\omega_i;\mu_i,\sigma_i\right) \tag{7}$$

where θ consists of the means and standard deviations of each independent Gaussian, $\theta=\{\mu_i,\sigma_i\}_{i=1:d}$. Probabilistic predictions are then performed by integrating over the variational distribution in place of the posterior:

$$p\left(y^\star|x^\star,\mathcal{D}\right)\approx\int p\left(y^\star|x^\star,\omega\right)q_\theta\left(\omega\right)d\omega \tag{8}$$

Similarly, mean and variance predictions are obtained by replacing the posterior $p(\omega|\mathcal{D})$ with the variational distribution $q_\theta(\omega)$ in Eqs. (5). Numerically, this integration is performed via Monte Carlo sampling, using N samples from the variational distribution $q_\theta(\omega)$.

In this context, training the Bayesian NN entails learning the variational parameters θ of the variational distribution. These parameters are optimized so that the variational density $q_\theta(\omega)$ is as close as possible to the target posterior pdf $p(\omega|\mathcal{D})$, according to a given distance metric. Most commonly used in this context is the Kullback–Leibler (KL) divergence, defined as:

$$\mathrm{KL}(q||p) = E_q[\ln q - \ln p] = \int q(\omega) \ln \frac{q(\omega)}{p(\omega)} d\omega \tag{9}$$

Thus, the variational parameters θ of the approximate variational distribution are computed by minimizing the following loss function[4]:

$$\hat{\theta} = \underset{\theta}{\mathrm{argmin}} \quad \mathrm{KL}\left(q_\theta(\omega)\,||\,p(\omega|\mathcal{D})\right) \tag{10a}$$

$$= \underset{\theta}{\mathrm{argmin}} \quad \mathrm{KL}\left(q_\theta(\omega)\,||\,p_0(\omega)\right) - \mathrm{E}_{q_\theta(\omega)}\left[\ln p(\mathcal{D}|\omega)\right] \tag{10b}$$

Applying this concept to training Bayesian neural networks yields the so-called BayesByBackprop algorithm [33,34]. When the variational density and prior[5] consist of factorized Gaussian densities, the first term in Eq. (10b) can be computed analytically, while Monte Carlo sampling is used to evaluate the expectation of the second term, i.e., $\mathrm{E}_{q_\theta(\omega)}\left[\ln p(\mathcal{D}|\omega)\right] = \frac{1}{N}\sum_{i=1}^{N} \ln p(\mathcal{D}|\omega^{(i)})$, with $\omega^{(i)} \sim q_\theta(\omega)$ and the likelihood being defined by Eq. (2).

However, minimizing the KL divergence between the variational distribution and the true posterior pdf as presented above tends to produce variational distributions that underestimate the true uncertainty. This is illustrated in Fig. 2 for two simple examples where the target distribution p is either multi-modal (Fig. 2a) or exhibits correlation between its components (Fig. 2b). In both cases, the distribution q, a factorized Gaussian fitted via minimization of the KL divergence, underestimates the variance of the target p.

In order to obtain better estimates of the uncertainty, one may consider more complex variational densities, such as mixture models or Gaussian densities that account for some correlation between parameters. Our numerical experiments in this respect showed only small improvement of the uncertainty estimates, as briefly discussed in Appendix A. A more attractive approach is to consider different distance metrics between the variational distribution and the target posterior in place of the KL divergence. The so-called α-BlackBox algorithm presented in [50] makes use of the α-divergence metric, defined as:

$$D_\alpha = \frac{1}{\alpha(1-\alpha)}\left(1 - \int p(\omega)^\alpha q(\omega)^{1-\alpha} d\omega\right) \tag{11}$$

Fig. 2 illustrates how minimizing the α-divergence yields Gaussian variational densities with larger standard deviation. Results depend on the chosen value of α. As noted in [50], the alpha-divergence is equivalent to the KL divergence $\mathrm{KL}(q||p)$ as $\alpha \to 0$, and equivalent to $\mathrm{KL}(p||q)$ as $\alpha \to 1$. In the Bayesian NN examples presented hereafter, it was noticed that increasing the value of α from 0 to 1 yielded approximate posteriors with larger uncertainty (running the $\alpha-$BlackBox with $\alpha \to 0$ yields equivalent results to running the BayesByBackprop algorithm).

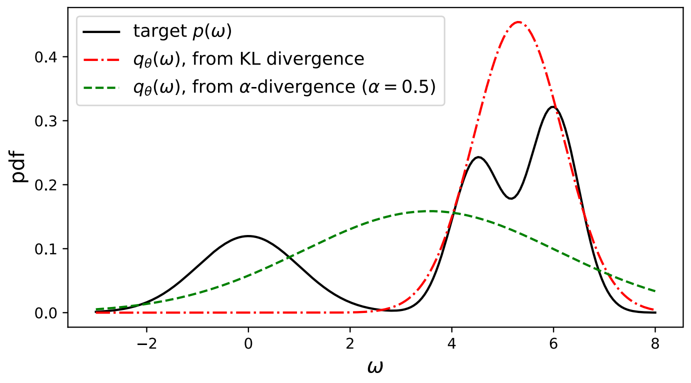
Similar to the BayesByBackprop algorithm, the $\alpha-$BlackBox algorithm presented in [50] minimizes a loss function to estimate the parameters of a Gaussian approximation $q_\theta(\omega)$ to the intractable posterior $p(\omega|\mathcal{D}) \propto \prod_{i=1}^{n} p(y_i|\omega)p_0(\omega)$. This algorithm derives from the so-called power-EP algorithm [51], which assumes a variational approximation of the form:

$$q_\theta(\omega) \propto \prod_{i=1}^{n} g_i(\omega) p_0(\omega) \tag{12}$$

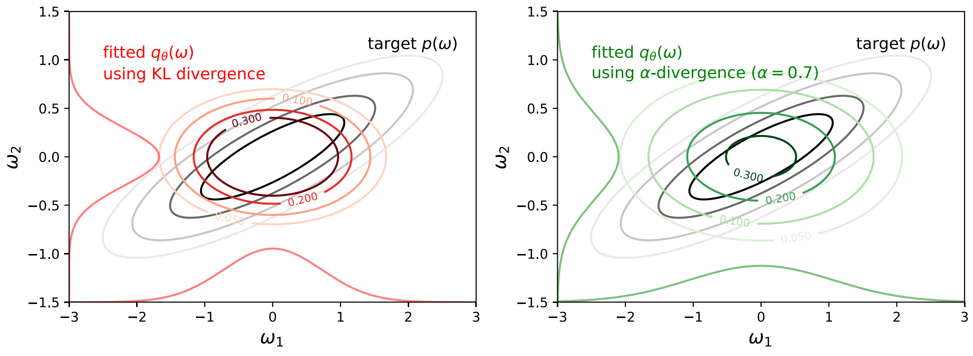
where $g_i(\omega)$ is a 'site approximation' used to approximate the effect of the ith likelihood term $p(y_i|\omega)$. The $\alpha-$BlackBox algorithm [50] makes the further approximation that all site approximations are equal, i.e., $g_i(\omega) =$

[4] This optimization problem can be seen instead as maximizing $E_{q_\theta(\omega)}[\ln p(\omega, \mathcal{D}) - \ln q_\theta(\omega)]$ which, by Jensen's equality, is a lower bound on the model evidence $p(\mathcal{D})$. Thus this formulation is often referred to as ELBO (Evidence Lower BOund) maximization.

[5] Note that the parameters of the prior could also be optimized. This strategy is known as empirical Bayes and some authors have found that it can lead to poor performance for the BayesByBackprop algorithm [34] discussed herein.



(a)



(b)

Fig. 2. Fitting a variational Gaussian distribution $q_\theta(\omega)$ to a target distribution $p(\omega)$ by minimizing a distance metric (KL divergence vs. α-divergence): (a) $p(\omega)$ is multi-modal, (b) $p(\omega)$ is bi-variate with correlated marginals (the contour plots all show probabilities 0.3, 0.2, 0.1, 0.05, from darkest to lightest shade).

$g(\omega)$, $\forall i$, where $g(\omega)$ represents the average effect of each likelihood term on the posterior. It is further assumed that the prior and average site approximation belongs to the same exponential family, i.e., are all (factorized) Gaussian, with parameters $\theta_0 = \{\mu_0, \sigma_0\}$ and $\theta_{av} = \{\mu_{av}, \sigma_{av}\}$ respectively. $g(\omega)$ then refers to the unnormalized Gaussian density, written in its canonical form:

$$g(\omega) = \exp\left(\frac{\mu_{av}}{\sigma_{av}^2}\omega - \frac{1}{2\sigma_{av}^2}\omega^2\right) \tag{13}$$

Under these assumptions, the resulting product $q_\theta(\omega)$ is also Gaussian by construction, with parameters $\theta = \{\mu, \sigma\}$

$$\mu = \sigma^2\left(n\frac{\mu_{av}}{\sigma_{av}^2} + \frac{\mu_0}{\sigma_0^2}\right) \tag{14a}$$

$$\frac{1}{\sigma^2} = \frac{n}{\sigma_{av}^2} + \frac{1}{\sigma_0^2} \tag{14b}$$

where n again is the number of data points. In the $\alpha-$BlackBox algorithm, the energy function to be minimized is the following [50]:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \left(-\ln Z(\theta) - \frac{1}{\alpha}\sum_{i=1}^{n} \ln \mathrm{E}_q\left[\left(\frac{p(y_i|\omega)}{g(\omega)}\right)^\alpha\right]\right) \tag{15}$$

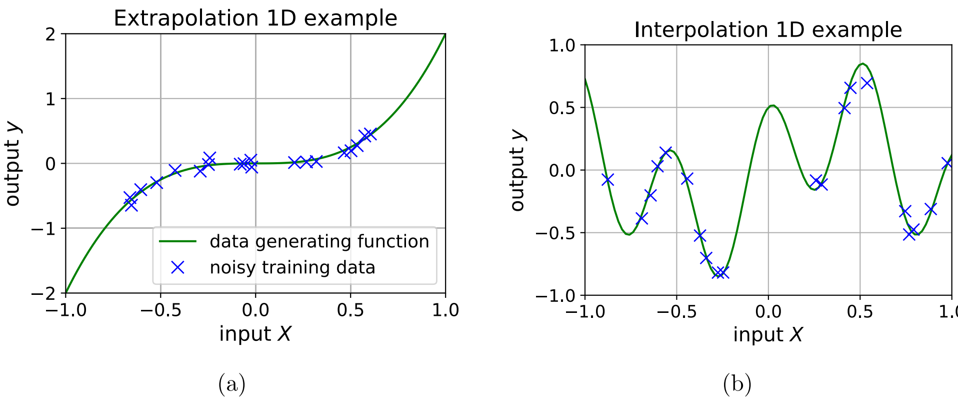


Fig. 3. Simple 1D examples to test behavior of algorithms in extrapolation and interpolation settings.

where $\ln Z(\theta)$ is the log-partition function of the variational Gaussian distribution, i.e., $\ln Z(\theta) = \frac{\mu^2}{2\sigma^2} + \ln \sigma$. As in the BayesByBackprop, the expectation over the variational $\mathrm{E}_q[\cdot]$ is computed via MC sampling.

2.2.2. Benchmark examples

In the following, we demonstrate the capabilities of the VI-based neural networks on two simple 1D examples meant to illustrate their behavior in both extrapolation and interpolation settings (Fig. 3). In the first benchmark example, data is generated from a noisy cubic function, i.e., $y = 2x^3 + \varepsilon$, $\varepsilon \sim \mathcal{N}\left(0, \sigma_n^2\right)$, with $\sigma_n = 0.05$ (aleatoric uncertainty, known). The data set consists in 20 (x, y) noisy data pairs. A Bayesian neural network with 1 hidden layer (100 units) is fitted to this data. The BayesByBackprop and α-BlackBox algorithms are compared, along with an MCMC based algorithm (Fig. 4). In particular, the HMC algorithm [38], was run on this problem to obtain a good estimate of the true uncertainty and provide a basis for comparison between various algorithms (details of the implementation of HMC and convergence analysis are provided in Appendix B). Fig. 4 shows that the BayesByBackprop algorithm predicts larger uncertainty away from the data, as it should. But it underestimates the magnitude of the uncertainty compared to the ground truth provided by the MCMC. The $\alpha-$divergence algorithm, on the other hand, yields similar mean predictions with larger uncertainty estimates, even more so as the value of α is increased.

The second 1D example is a sinusoidal function of the form $y = 0.4 \sin 4x + 0.5 \cos 12x$, with added zero-mean Gaussian noise with standard deviation $\sigma = 0.05$. The network consists of 3 hidden layers with 20 units each. This example seeks to understand whether the VI algorithms studied herein can provide acceptable uncertainty predictions in between sparse data points (interpolation). Fig. 5a-b illustrates how various algorithms perform differently for this problem. Fig. 5c shows that independent runs of the VI algorithms – either the BayesByBackprop or $\alpha-$BlackBox – that use different starting points for the optimization[6] can yield different predictions, both in mean and uncertainty.

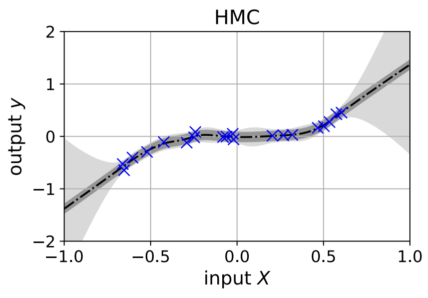
From the above simple examples, it can be seen that VI algorithms can provide approximations of the posterior uncertainty, but that results can widely vary depending on the chosen algorithm, the hyper-parameters (α value), and the starting point for the optimization. In the following section we illustrate how to obtain improved estimates by averaging over various probabilistic models.

3. Improving accuracy of variational inference via probabilistic model averaging

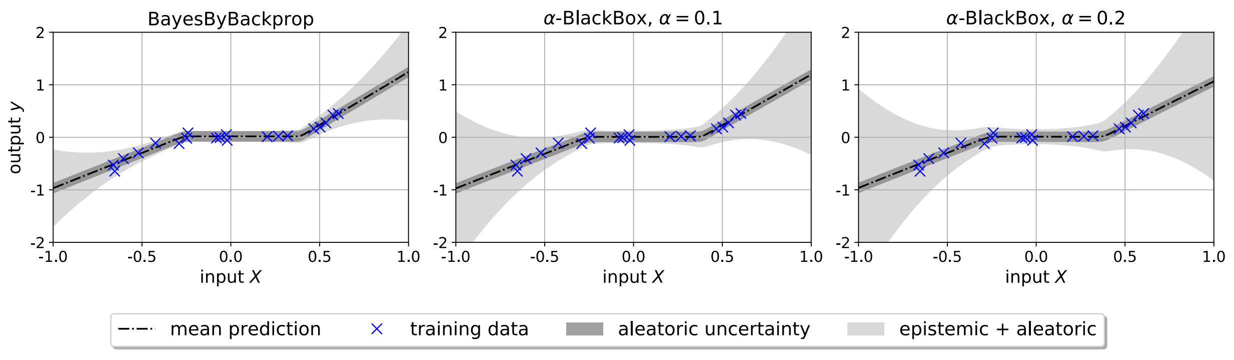
3.1. Methods for (pseudo-)Bayesian model averaging

As illustrated in the examples above, results of the probabilistic training depend on the chosen algorithm, as well as certain parameters of the stochastic optimization such as the starting points for the network weights. In this section, we propose an approach based on pseudo-Bayesian model averaging to mitigate this dependence and

[6] Some details about algorithmic implementation of the algorithms, in particular the choice of starting values for parameters θ, are provided in Appendix B.



(a)



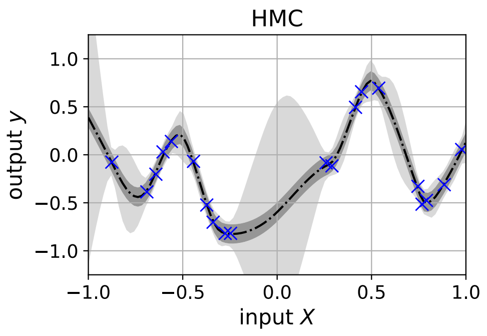
(b)

Fig. 4. A benchmark example: extrapolation away from training data, mean and uncertainty predictions from the Hamiltonian Monte Carlo (a) vs. various VI algorithms (b). The plots show the training data (blue crosses), the mean prediction from the NN (black dashed line) and total uncertainty prediction (light gray shaded area). The uncertainty is plotted as two standard deviations around the mean.

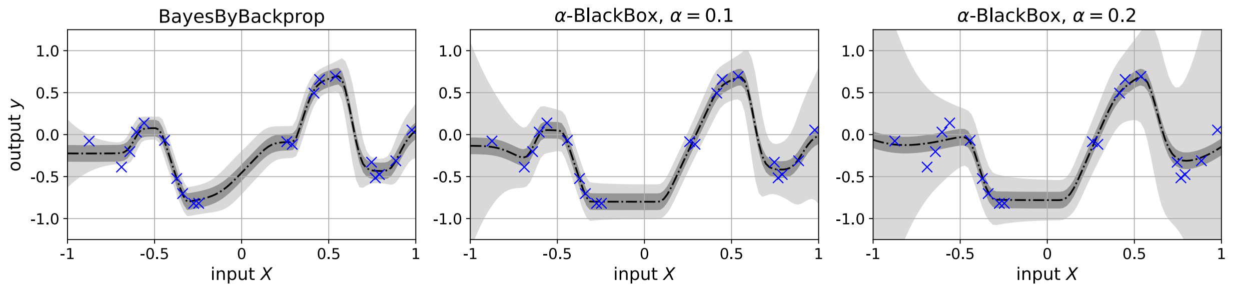
produce more robust Bayesian neural networks. A similar idea is used in [52] where several NNs trained via dropout are aggregated, assigning each NN the same weight. Bayesian-like model averaging procedures instead assign a weight for each model based on model evidence or predictive accuracy. Examples of Bayesian-like model averaging procedures applied to Bayesian neural networks include [53] where model averaging is performed using the Bayesian information criterion (BIC), or in [54] to select a neural network architecture using a logarithmic score utility function. In these works however, the predictive density for a given neural network model is obtained via some sort of linearization. In the following we propose to use a pseudo-Bayesian model averaging procedure to combine outputs of various VI training algorithms, a concept illustrated for instance in [55] for combining VI with different starting points for a simple 2D bi-modal posterior. Note that, along with different VI-training algorithms, this procedure could also be applied to assess and combine networks with different architectures, nonlinear activation functions or priors. This does not pose significant practical challenges and is not discussed herein for brevity.

In the following we refer to different algorithms or optimization parameters as different models $\mathcal{M}_{m=1:M}$. A popular way to choose hyper-parameters, i.e. models, in machine learning is through cross-validation. Cross-validation fits various models to a certain training set, evaluates their performance on a leave-out set and chooses the model that provides the best prediction. In [54], a utility criterion based on posterior predictive log-densities, similar to the method used herein, is used to choose a network architecture, and it is reported to be more capable of detecting the true model compared to a standard cross-validation scheme when small amounts of data are available, albeit with a slightly larger computational cost. However, choosing one model that gives optimal performance on unseen data can be unstable and somewhat wasteful of information since several models are fitted to data but only one is used for prediction [55]. More robust predictions can be obtained instead by combining different models. In this context, each model is assigned a model weight β_m, and probabilistic predictions are performed via averaging as:

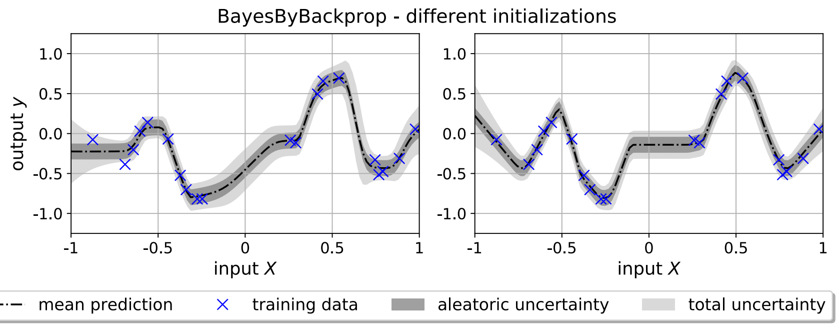
$$p\left(y^{\star}|x^{\star},\mathcal{D}\right)=\sum_{m=1}^{M}\beta_m p\left(y^{\star}|x^{\star},\mathcal{D},\mathcal{M}_m\right) \tag{16}$$



(a)



(b)



(c)

Fig. 5. A benchmark example: interpolation between data points. Predictions using various algorithms (Hamiltonian Monte Carlo (a) vs. VI algorithms (b)) and various hyper-parameters (b) show vastly different results. Predictions using the same VI algorithm but different starting points for the optimization (c) can also produce inconsistent models for both mean and uncertainty.

where $p\left(y^{\star}|x^{\star}, \mathcal{D}, \mathcal{M}_m\right)$ is computed via Eq. (8).

In a Bayesian setting, a favored approach to assess and select or average models is through Bayesian model averaging (BMA), where each model is weighted according to its posterior probability based on data $\beta_m = P(\mathcal{M}_m|\mathcal{D}) \propto p(\mathcal{D}|\mathcal{M}_m)P(\mathcal{M}_m)$. This approach has various drawbacks though. For instance, it is flawed in a setting where the data generating process is not one of the candidate models [55], which is the case considered here as the fitted networks are all surrogates of the true data generating process. Furthermore, computing the model evidence is not straightforward in general, but various approximations such as the Bayesian information criterion (BIC) or the Kashyap information criterion (KIC) can be used [56]. In particular, the KIC evaluated at the MAP $\hat{\omega}$ is computed as:

$$\mathrm{KIC} = -2\ln p(\mathcal{D}|\hat{\omega}) - 2\ln p_0(\hat{\omega}) - d\ln 2\pi - \ln|C_{post}| \tag{17}$$

where d is the number of parameters in the network, $\hat{\omega}$ the MAP estimate and C_{post} the posterior covariance, both of which are readily available from the variational density. In our experiments (see Table 1 and discussion in Section

Table 1
Comparison of various weighting schemes for the 1D extrapolation problem.

	$\beta(\mathcal{M}_1)$	$\beta(\mathcal{M}_2)$	$\beta(\mathcal{M}_3)$	$\beta(\mathcal{M}_4)$	$\beta(\mathcal{M}_5)$
KIC at MAP	0.	0.	0.	0.03	0.97
KIC at MAP with variance window	0.	0.	0.02	0.17	0.81
DIC	1.	0.	0.	0.	0.
WAIC	1.	0.	0.	0.	0.
LOO	0.47	0.26	0.14	0.10	0.03
LOO approx.	0.70	0.16	0.09	0.03	0.007

Section 3.4), we saw that using the KIC with variance window [57] is an acceptable option for Bayesian model averaging in this context.

Pseudo-Bayesian model averaging methods instead compare models based on some measure of predictive accuracy computed using the training data, such as the expected log pointwise predictive density (*elpd*, [55]). As explained in [58], well-known criteria such as the Akaike, deviance or Watanabe–Akaike information criterion (AIC, DIC, WAIC) for model selection are approximations of the *elpd*. Herein, we focus on an approximation of the *elpd* based on the leave-one-out (LOO) predictive densities [55], that is:

$$\text{elpd}_{LOO,m} = \sum_{i=1}^{n} \ln p(y_i | x_i, \mathcal{D}_{-i}, \mathcal{M}_m) \tag{18}$$

where $\mathcal{D}_{-i} = \{\mathbf{x}_{-i}, \mathbf{y}_{-i}\}$ refers to the data set without point i and $z_{i,m} = \ln p(y_i|x_i, \mathcal{D}_{-i}, \mathcal{M}_m)$ is the LOO predictive density of data point i. Then the pseudo-BMA weight for model $\mathcal{M}_m$ is:

$$\beta_m \propto \exp(\text{elpd}_{LOO,m}) \tag{19}$$

The weights are normalized so that they sum up to 1 over all considered models.[7]

The LOO predictive density can be computed by iteratively fitting the model to all LOO data sets $\mathcal{D}_{-i}, \quad i = 1:n$. More specifically, running the variational inference procedure for model $\mathcal{M}_m$ and data set $\mathcal{D}_{-i}$ yields a LOO variational distribution $q_{\theta[-i][m]}(\omega)$ that is used to compute the LOO predictive density as:

$$p(y_i|x_i, \mathcal{D}_{-i}, \mathcal{M}_m) \approx \int p\left(y_i|x_i, \omega\right) q_{\theta[-i][m]}(\omega)\, d\omega \approx \frac{1}{N} \sum_{j=1}^{N} p\left(y_i|x_i, \omega^{(j)}\right) \tag{20}$$

where $\omega^{(j)}$ is sampled from the variational distribution $q_{\theta[-i][m]}(\omega)$. This procedure is computationally costly as it requires running the variational inference n times for each model. This method is tractable in our case though, since we consider small data sets. One could also consider using 'leave-several-out' predictive densities, thereby reducing the number of times the system would need to be refitted. In the following section, we show a way to estimate the LOO predictive density without refitting the model several times using a peculiarity of the $\alpha-$BlackBox algorithm.

3.2. An inexpensive estimate of the LOO predictive density from the α-BlackBox algorithm

In this section we show how to estimate the LOO variational density $q_{\theta[-i]}(\omega)$ for each model considered (here we drop the reference to the model for ease of notation) by only fitting the network to the complete data set $\mathcal{D}$, i.e., removing the need to re-fit to all LOO data sets $\mathcal{D}_{-i}, \quad i = 1:n$. Recall that the power-EP/$\alpha-$BlackBox algorithm assumes a variational approximation $q_\theta(\omega)$ to the intractable posterior $p(\omega|\mathcal{D}) \propto \prod_{i=1}^{n} p(y_i|\omega)p_0(\omega)$ of the form:

$$q_\theta(\omega) \propto \prod_{i=1}^{n} g_i(\omega) p_0(\omega) \tag{21}$$

[7] In our numerical examples, we use a further small enhancement of the weights based on the Bayesian bootstrap, as presented in [55]. This yields model weights that do not tend toward the limit values 0, 1, thus helping preserving diversity in the set.

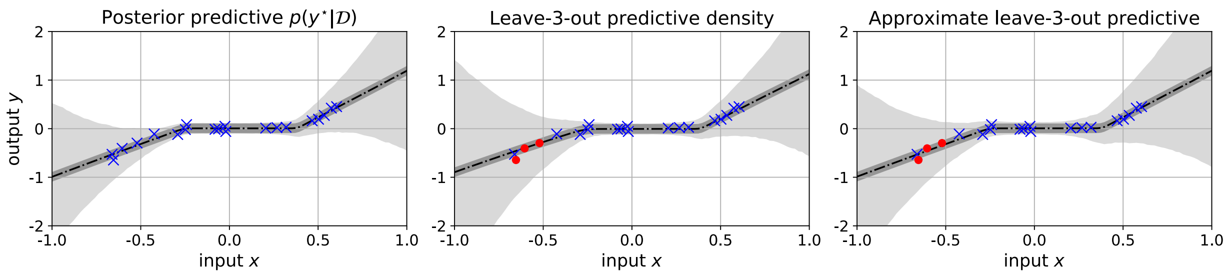


Fig. 6. Leave-three-out predictive density on benchmark problem vs. its approximation via Eq. (22c) All three plots are generated by running an α-BlackBox algorithm with $\alpha = 0.1$. The left plot uses the full data set for training (blue crosses). The middle plot shows the true leave-out density, i.e., fitted to a data set where three data points are removed (red dots), yielding higher predicted uncertainty in the negative x-region where data was removed. The right plot shows an approximation of the leave-out density, which directly uses the results from the model fitted to the full dataset through Eq. (22c).

where $g_i(\omega) = g(\omega), \forall i$ for the $\alpha-$BlackBox. The interesting idea here is that this approximation provides readily-available estimates of the LOO posterior densities as:

$$p(\omega|\mathcal{D}_{-i}) \propto \prod_{j\neq i} p(y_j|\omega)p_0(\omega) \quad \text{[true LOO posterior]} \tag{22a}$$

$$q_{\theta[-i]}(\omega) \propto \prod_{j\neq i} g_j(\omega)p_0(\omega) \quad \text{[approx. via power-EP algorithm]} \tag{22b}$$

$$q_{\theta[-i]}(\omega) \propto \prod_{j\neq i} g(\omega)p_0(\omega) = g(\omega)^{(n-1)}p_0(\omega) \quad \text{[approx. via } \alpha-\text{BlackBox algorithm]} \tag{22c}$$

Fig. 6 illustrates the difference between the true leave-out density (computed by re-fitting a model to a leave-out data set) and the approximation above, which directly uses the results from the model fitted to the full dataset through Eq. (22c). Both methods for computing the leave-out density predict higher levels of uncertainty than the predictive density based on the full data set, however the true leave-out predicts more appropriately higher levels of uncertainty in the negative$-x$ region of the input space, where the leave-out data lies. On the other hand, as Eq. (22c) uses an average site approximation for all data, all data points have a similar effect on the leave-out density, which is a general increase of the output uncertainty. Thus the approximation above will provide a very rough estimate of the LOO predictive density, which could be enhanced via Importance Sampling or running a few steps of a MCMC to target the true LOO. It is noteworthy to mention that a better trade-off between the expensive leave-out computation via refitting and less accurate approximation above could be obtained by using several approximating factors in the $\alpha-$BlackBox algorithm (instead of a single average factor) that each represents the effect of a group of similar data points. This idea is not studied further at the moment, the interested reader is referred to [59] for a deeper discussion on controlling granularity of the approximation by introducing several approximating factors.

3.3. Proposed methodology

We showed in Section 2 that variational algorithms such as the BayesByBackprop or $\alpha-$BlackBox algorithms provide approximations of the posterior uncertainty that can very much vary depending on the chosen algorithm, the hyper-parameters (α value), and the starting point for the optimization. Averaging over various models, i.e., various training algorithms, will provide more robust estimates of the posterior uncertainty. The proposed approach is thus as follows:

- Choose a set of M models $\mathcal{M}_{m=1:M}$, where each model corresponds to a VI algorithm with a given set of hyper-parameters and a starting point for the weights initialization. As will be explained with an example later on, this can be seen as defining and sampling from a prior over the hyper-parameters (in our case the α value). The starting values of the weights are chosen at random, following guidelines in Appendix B.
- Train each model on the available data.

- For each model, use the approximation of the LOO predictive density presented above to approximate the *elpd* for each model, i.e., combining Eqs. (18) and (20):

$$\text{elpd}_{LOO,m} = \sum_{i=1}^{n} \ln p(y_i|x_i, \mathcal{D}_{-i}, \mathcal{M}_m) \approx \sum_{i=1}^{n} \ln \frac{1}{N} \sum_{j=1}^{N} p\left(y_i|x_i, \omega^{(j)}\right) \tag{23}$$

  where $\omega^{(j=1:N)}$ are sampled from the approximate LOO density $q_{\theta[-i][m]}(\omega)$ as in Eq. (22c). As a reminder, using this approximation of the LOO density avoids having to re-fit the model n times, where n is the number of data points.
- Compute the weight for each model as in Eq. (19), or its bootstrapped version.
- The posterior predictive is the weighted sum over all trained models, Eq. (16), or more specifically for the mean and variance predictions:

$$\mathrm{E}\left[y^\star|x^\star, \mathcal{D}\right] = \sum_{m=1}^{M} \beta_m \mathrm{E}\left[y^\star|x^\star, \mathcal{D}, \mathcal{M}_m\right] \tag{24a}$$

$$Var\left(y^\star|x^\star, \mathcal{D}\right) = \sum_{m=1}^{M} \beta_m \left(Var\left(y^\star|x^\star, \mathcal{D}, \mathcal{M}_m\right) + \left(\mathrm{E}\left[y^\star|x^\star, \mathcal{D}, \mathcal{M}_m\right] - \mathrm{E}\left[y^\star|x^\star, \mathcal{D}\right]\right)^2\right) \tag{24b}$$

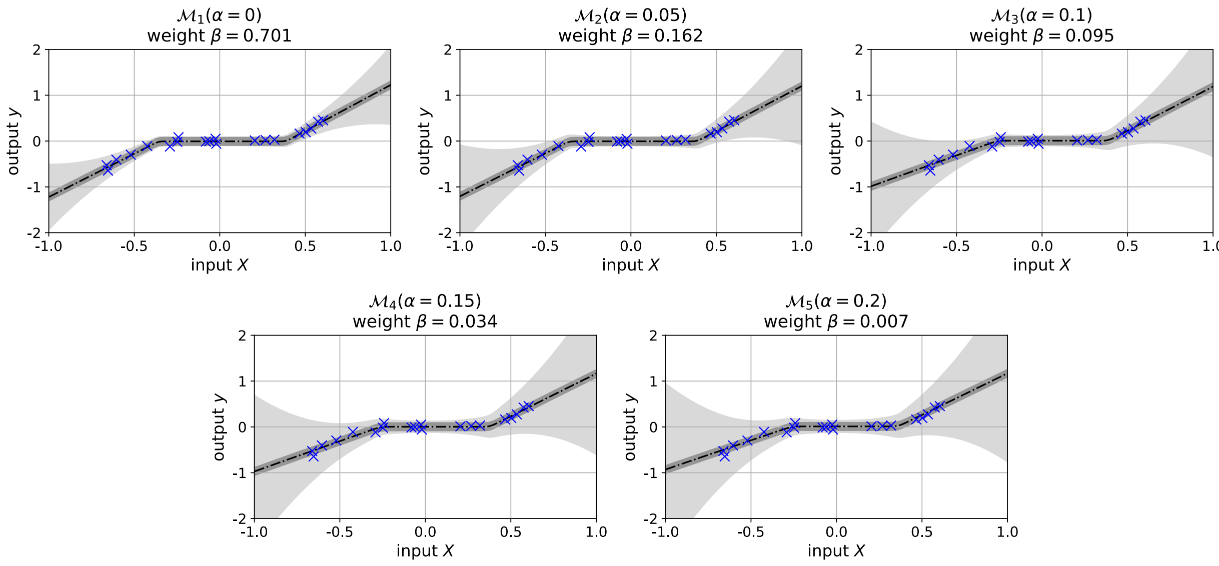
  where $\mathrm{E}\left[y^\star|x^\star, \mathcal{D}, \mathcal{M}_m\right]$ and $Var\left(y^\star|x^\star, \mathcal{D}, \mathcal{M}_m\right)$ are the mean and variance of the posterior predictive density obtained using a single VI algorithm, as described in Section 2. As a reminder, for e.g. the expectation, we use Eq. (5) with the appropriate VI approximation $q_{\theta[m]}(\omega)$ in place of the true posterior, i.e. $\mathrm{E}\left[y^\star|x^\star, \mathcal{D}, \mathcal{M}_m\right] = \mathrm{E}_{q_{\theta[m]}(\omega)}\left[f^{\omega}(x^\star)\right]$, computed via MC sampling. From Eq. (24b), we see that the posterior uncertainty arises from both the uncertainty predicted by each single model $Var\left(y^\star|x^\star, \mathcal{D}, \mathcal{M}_m\right)$, but also from their potential variations in mean predictions. This will be illustrated later in the simple interpolation 1D example.

3.4. Demonstration on simple 1D problems

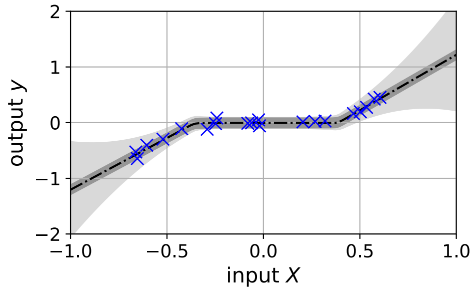
Figs. 7 and 8 illustrate how the proposed methodology can be used to obtain improved mean and uncertainty predictions for the two simple 1D problems. In the first problem, the various predictive models $\mathcal{M}_{k=1:5}$ are built using different α hyper-parameters in the $\alpha-$BlackBox variational inference algorithm that yield similar mean predictions but different levels of uncertainty. In particular, we use $\alpha = 0., 0.05, 0.1, 0.15, 0.2$ in this case. Choosing this set of five models is equivalent to setting a discrete uniform prior over this α hyper-parameter. In our experiments we have observed overall that the model weights decrease quickly with increasing α, thus adding models with larger values of α will likely yield models that will be discarded (assigned close to 0 weight). The model averaging framework tends to prioritize the model with $\alpha = 0$, – the model that minimizes the KL divergence between the true posterior and the variational distribution, and is known to underestimate uncertainties – but it does assign non-negligible weight to other models too, thus somewhat correcting this under-estimation of uncertainties.

For completeness, we also performed averaging using a variety of criteria. The DIC, WAIC and LOO are all criteria that aim at approximating the expected predictive accuracy of a model, the reader is referred to [58] for a very good discussion on the subject. For this example, these criteria assign larger weights to models fitted using small values of α, however the DIC and WAIC assign all the weight to a single model and thus do not perform any model averaging. The proposed LOO approximation provides decent results in comparison with the full LOO obtained from refitting all models, but with a smaller compute time. On the other hand, strategies based on evaluating the model evidence such as the KIC evaluated at the MAP [56], or its smoothed version [57], tend to prefer model $\mathcal{M}_5$ with a larger α. Indeed, these criteria involve two terms: a data fit term that rewards models that fit the data well, and a penalty term that penalizes against too much information gain from the prior to the posterior (Occam razor). In the case considered herein, the penalty term increases for smaller values of α as the posterior narrows, i.e., diverges away from the uninformative prior. This illustrates how weighting schemes based on model evidence strongly depend on the choice of prior. For machine learning models such as neural networks where parameters are non-physical and the choice of prior is quite arbitrary, we prefer a weighting method that solely relies on predictive model accuracy.

In the second problem (Fig. 8), the various (fifteen) models are built using different hyper-parameters (α value, sampled between 0 and 0.1) and different initializations of the optimization procedure, which yield predictive models



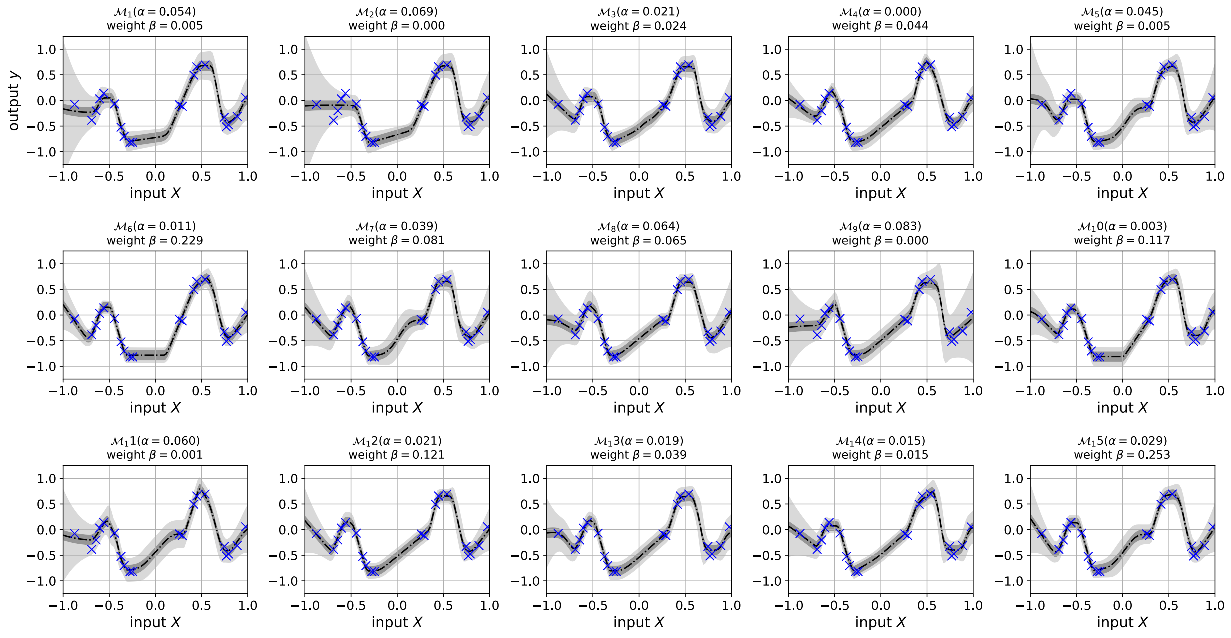
(a)



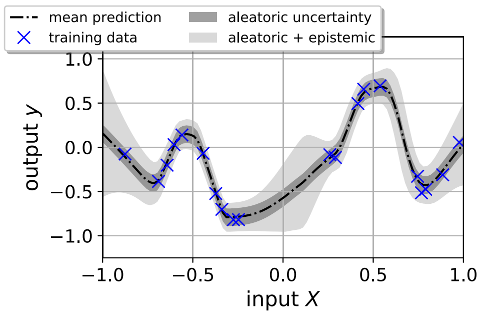
(b)

Fig. 7. Pseudo-Bayesian model averaging for the first problem. (a) Predictions of various models (different α hyper-parameters in the $\alpha-$BlackBox algorithm) and their respective weights. (b) Prediction averaged over the various models.

with different mean and uncertainty predictions. In this setting, the pseudo-Bayesian model averaging framework allows us to average over different mean predictions, and discard models that find mean predictions that are too unlikely. For instance, we can observe that models 6, 7 and 8 all carry significant weight and make different mean predictions where data is scarce (around $x = -1$ and between $x = -0.2$ and $x = 0.2$). This difference in mean prediction, along with the uncertainty predicted by each model, both participate to the overall uncertainty in the averaged model. Also, it can be noted that a certain numbers of models are assigned zero weight. When choosing the number of models, we thus recommend training enough models that at least a few of them carry significant weight, thus ensuring diversity in the set. In comparing with other weighting schemes for this problem we observed that the KIC with variance window also performed satisfactorily in averaging over several models. However, because the results of the VI-algorithm for this problem depend on the stochastic initialization and optimization path, weights computed via an exact LOO estimation, i.e, by re-fitting to all LOO datasets instead of using the proposed approximation, led to very unstable results as each re-fit to a LOO dataset, and thus the model weights, is affected by the stochastic initialization of the weights.



(a)



(b)

Fig. 8. Pseudo-Bayesian model averaging for the second problem. (a) Predictions of various models (different α hyper-parameters in the $\alpha-$BlackBox algorithm and optimization initialization) and their respective weights. (b) Prediction averaged over the various models.

4. Building probabilistic material structure–property linkages

4.1. Predicting homogenized and localized properties of a 2-phase microstructure

The illustrative example considered herein consists of building a linkage between a 2-phase microstructure and a set of homogenized and localized properties that characterize the mechanical behavior of a statistical volume element (SVE). The microstructure represents a composite material with linear elastic fibers randomly distributed within a matrix that exhibits a plastic behavior governed by a power law of the form:

$$\sigma = a_m + b_m \epsilon^{c_m} \tag{25}$$

We are specifically interested in understanding how certain material and/or geometrical parameters affect the mechanical behavior of the composite SVE. In particular, the input parameters considered herein are the volume fraction of fibers in the matrix vf, the Young's modulus of the fibers E_f and the parameters b_m and c_m of the matrix plasticity law. Parameter a_m is fixed to 400 MPa. The mechanical behavior of the composite SVE can be represented

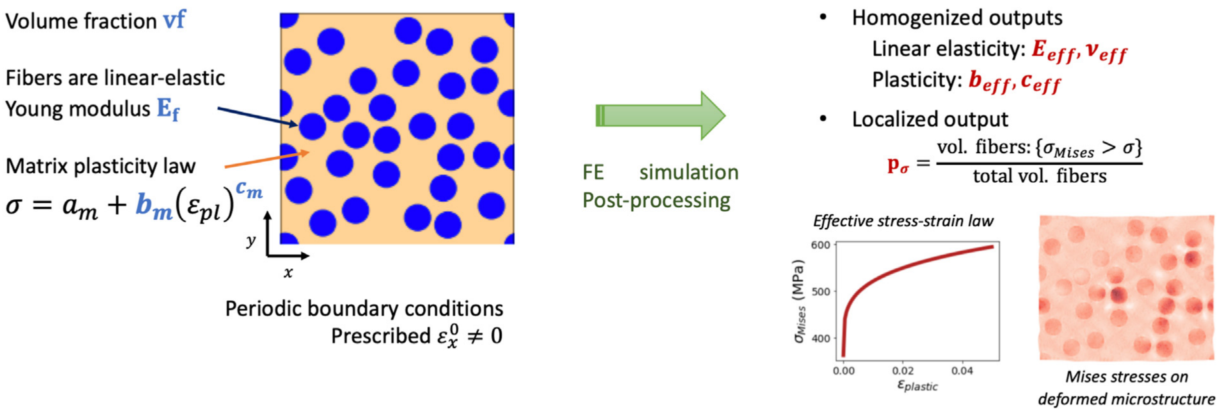

Volume fraction vf
Fibers are linear-elastic
Young modulus E_f
Matrix plasticity law
$\sigma = a_m + b_m(\varepsilon_{pl})^{c_m}$
y
x
Periodic boundary conditions
Prescribed $\varepsilon_x^0 \neq 0$
FE simulation
Post-processing
Homogenized outputs
Linear elasticity: E_{eff}, ν_{eff}
Plasticity: b_{eff}, c_{eff}
Localized output
$p_\sigma = \frac{\text{vol. fibers: } \{\sigma_{Mises} > \sigma\}}{\text{total vol. fibers}}$
Effective stress-strain law
σ_{Mises} (MPa)
$\varepsilon_{plastic}$
Mises stresses on deformed microstructure

Fig. 9. Structured inputs $x = \{\text{vf}, E_f, b_m, c_m\}$ and outputs $y = \{E_{\text{eff}}, \nu_{\text{eff}}, b_{\text{eff}}, c_{\text{eff}}, p_{th}\}$ considered in the materials example.

by various effective properties such as the parameters governing the effective stress–strain law or localized properties such as the value of stresses or strains at various locations in the microstructure, which may be of particular interest if one is interested in detecting localized failure mechanisms. In this example, five output properties of the SVE are considered as follows:

- E_{eff}, ν_{eff} the effective Young's modulus and Poisson ratio of the composite RVE,
- b_{eff}, c_{eff} the parameters governing the effective plasticity law,
- p_{th} the proportion of fibers that undergo a Von-Mises stress above a certain threshold $th = 0.9$ GPa (localized property).

Fig. 9 summarizes the input parameters and output properties considered in this example.

In a traditional context, the linkage from input geometric/material parameters to output properties would be provided from a computational analysis or experiment. In the example considered herein a 2D finite element model is utilized to generate synthetic data (input/output pairs) and study the capability of Bayesian neural networks to serve as a surrogate model in place of this expensive FE simulation. These finite element simulations were run using computational resources at the Maryland Advanced Research Computing Center (MARCC). In the FE analysis, periodic boundary conditions are applied to the 2D microstructure (plane strain case) that is subjected to a uni-axial tensile strain in the x-direction, i.e. $u_{\text{x}}(\text{x} = L, \text{y}) = u_{\text{x}}(\text{x} = 0, \text{y}) + \epsilon_{\text{x}}^0, \forall \text{y}$ where u_{x} is the x-direction displacement and the imposed strain ϵ_{x}^0 is increased from 0 to 0.05 in 100 increments. For a given input $x = \{\text{vf}, E_f, b_m, c_m\}$, a 2D microstructure with 30 randomly distributed circular fibers (see a few realizations of the SVE on Fig. 10) is generated and used as input to the FE simulation previously described. The FE simulation yields stress and strain fields over the RVE, and a post-processing step is conducted to extract the outputs of interest, in particular the homogenized properties $\{E, \nu, b, c\}_{\text{eff}}$. The homogenized stress and strain values in all three directions are obtained as e.g. $\sigma_{\text{x}} = \frac{1}{V} \sum_e \sigma_{\text{x}}^{(e)} V^{(e)}$ where $\sigma_{\text{x}}^{(e)}$ and $V^{(e)}$ are the stress and volume of a given finite element and V is the total volume. The elastic effective properties $\{E, \nu\}_{\text{eff}}$ are computed by inverting Hooke's law for plane strain, using the effective stress and strain values prior to observing any plasticity in the RVE. The effective plasticity parameters $\{b, c\}_{eff}$ are computed by fitting a power law $\sigma_{\text{Mises}} = a + b_{\text{eff}} \epsilon_{\text{pl}}^{c_{\text{eff}}}$ where σ_{Mises} and ϵ_{pl} are the Mises stress and equivalent plastic strain, computed from the effective stresses previously described, and a is fixed to 400 MPa (same value as the fixed a_m parameter of the matrix plasticity law).

Having selected a set of input geometric/material parameters and output properties to be predicted, a set of simulations are run in order to get some input/output data and learn a surrogate linkage based on Bayesian neural networks. It is useful to note at this point the user may choose a variety of structured inputs/outputs, as relevant to the task at hand. Also, unstructured inputs/outputs such as image data may be highly relevant, which could be handled via advanced machine learning algorithms such as convolutional neural networks (CNNs). This task is left for future consideration. Herein only structured inputs/outputs are considered which allows utilization of feedforward neural networks for surrogate modeling.

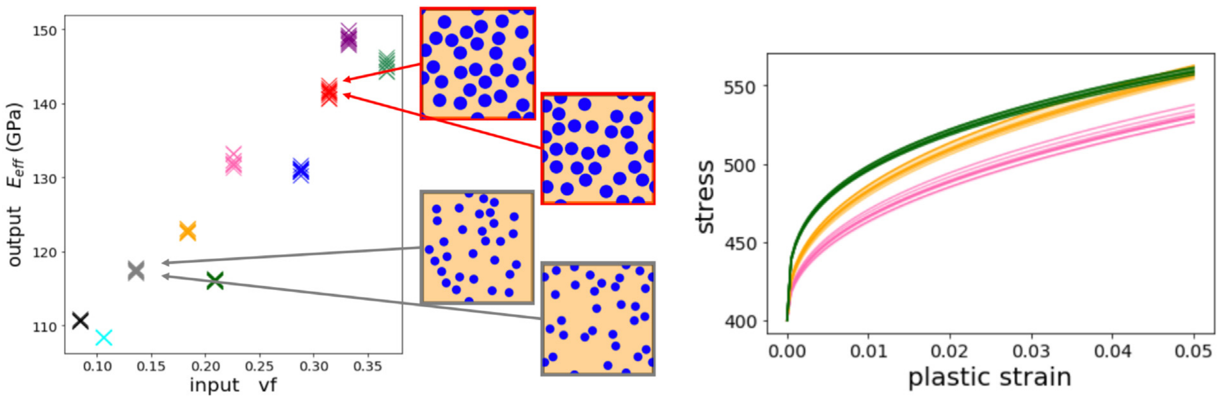


Fig. 10. Aleatoric uncertainties emerge from the random placement of fibers in the microstructure: data plotted in a given color are produced by running several FE simulations using the same input parameters $x = \{\text{vf}, E_f, b_m, c_m\}$ but re-sampling the fibers locations. Left plot: input v_f- output E_{eff} noisy relationship. Note that this is a 2D projection of a 4 inputs/5 outputs relationship, which explains the non-monotonic increase of E_{eff} with v_f. For example, the input Young's Modulus of the fibers may vary from one cluster to another. Right plot: effect of aleatoric uncertainties on effective plasticity law. The colors match the left plot, where again curves of a given color were generated using a value of the input $x = \{\text{vf}, E_f, b_m, c_m\}$ but different fiber placements. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4.2. Representation of aleatoric uncertainties

As previously mentioned, when building surrogate models from small amounts of potentially noisy data, careful consideration must be given to the various uncertainties within the system. Epistemic uncertainties arise from lack of knowledge of the model form and its parameters; more precisely in the present context, this uncertainty stems in part from lack of training data for the surrogate model, and can thus be reduced by collecting more data. Aleatoric uncertainties are intrinsic to the physical problem or data collection procedure considered: sensors are often imperfect and yield noisy measurements, certain inputs to the experimental setup or numerical simulation may not be easy to control, etc. In the present context, aleatoric uncertainty emerges from randomness in the fibers placement in the microstructure, which yields a noisy relationship between the inputs X and outputs y, as illustrated on Fig. 10. This uncertainty is inherent to the system and cannot be reduced by gathering more data.

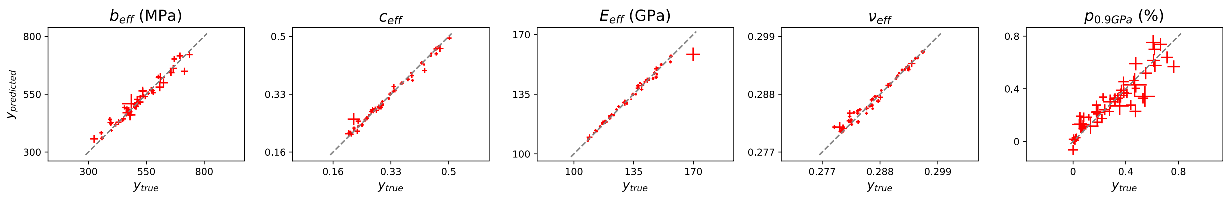
When building a surrogate neural network, the aleatoric uncertainty is accounted for by considering a probability model over the data $p(y|x, \omega) = \mathcal{N}(y; f^{\omega}(x), \Sigma)$. In the study presented herein, the focus is set on evaluating the epistemic uncertainty, thus the aleatoric uncertainty (i.e., the value of Σ) is assumed to be known (this assumption could be relaxed by learning Σ as an additional output to the network, see e.g. [49]). In order to evaluate the value of Σ herein (assumed to be diagonal), n_1 simulations were performed using the same input vector but different random fibers placements, yielding several output values $y^{(i,1:n_1)}$ for a given input parameter $x^{(i)}$. This procedure was repeated for n_0 values of the input vector $x^{(i)}$, providing an approximation of the aleatoric covariance as:

$$\Sigma = \mathrm{E}_x\left[\mathrm{Cov}(y|x)\right] = \frac{1}{n_0}\sum_{i=1}^{n_0}\frac{1}{n_1-1}\sum_{j=1}^{n_1}\left(y^{(i,j)} - \bar{y}^{(i)}\right)\left(y^{(i,j)} - \bar{y}^{(i)}\right)^T \tag{26}$$

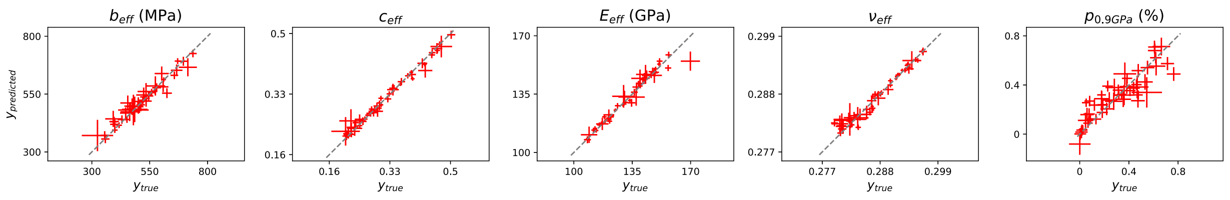
In performing the procedure above, it was observed that the aleatoric uncertainties are quite small for the effective linear elastic properties $\{E, \nu\}_{eff}$, while the data on the properties that characterize the nonlinear behavior of the composite material ($\{b, c\}_{eff}$) is noisier, even more so for the localized stress feature p_{th}. This suggests that these properties are more strongly affected by the random placement of the fibers in the RVE. In the next section, the covariance Σ is assumed known, while epistemic uncertainties are quantified by learning a distribution over the NN regressor's weights, as previously detailed.

4.2.1. Efficient data-based structure–property linkage with uncertainty estimation

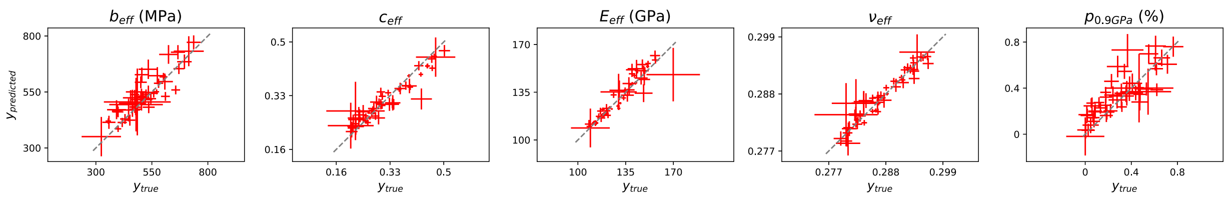
In the following, a Bayesian neural network is fitted to some training data synthesized using the procedure previously described and used in place of the finite element model to make predictions on unseen data. These



(a)



(b)



(c)

Fig. 11. Predicted outputs and their associated epistemic uncertainties the crosses are plotted as mean $+/-$ one standard deviation for a fixed test set of 50 data points and a varying amount of training data — size of the training set from top to bottom: (a) 100, (b) 50 and (c) 25 training data points.

predictions from the test set, along with their uncertainties, are compared to the 'true' values computed using the FE model. In the first set of numerical experiments, both the training and test data originate from the same distribution, more specifically the inputs are sampled from uniform distributions in the ranges $\mathrm{vf} \in [0.05, 0.4]$, $\mathrm{E_f} \in [200, 600]$, $\mathrm{b_m} \in [300, 500]$, $\mathrm{c_m} \in [0.2, 0.55]$.[8]

In the probabilistic setting considered herein, a prior is chosen as a combination of (1) a network architecture and (2) a prior over the weights. The chosen architecture consists of 3 hidden layers with 20 units each (bridging 4 inputs to 5 outputs), with ReLU activation functions. The prior pdf over the weights was chosen as a standard normal. This prior and network architecture was not optimized in any specific way, i.e. using additional validation data. On the contrary, the goal of the study is to demonstrate the capabilities of the probabilistic inference framework without the need to optimize the network architecture a priori. The pseudo-Bayesian model averaging procedure previously described is utilized with $\mathrm{M} = 10$ models, which consist of fitting an $\alpha-$BlackBox algorithm with different α values (between 0 and 0.1) and different starting point for the optimization procedure.

Fig. 11a shows the capabilities of the network ensemble in predicting all 5 outputs (mean and variance predictions) when a relatively large amount of data (100 training data points) is used. Figs. 11 further shows how the predicted epistemic uncertainty correctly decreases as the amount of training data increases (from 25 to 50 to 100 data points). Fig. 12 shows 2D representations of the approximate posterior predictive distributions $p(y^{\star}|x^{\star}, \mathcal{D})$ of the 5 output quantities of interest (for two data points $x^{\star}$ from the test set), obtained using this ensemble of Bayesian NNs. Finally, Fig. 13 further shows the prediction of the homogenized behavior law for four data points from the test set, for different sizes of the training set. Again, the probabilistic approach presented herein allows the user to predict not only mean values for the quantities of interest (such as this homogenized behavior law of the composite SVE), but also to quantify the error/uncertainty due to the fact that the surrogate model was trained on a limited

[8] In order to avoid having vector components with various orders of magnitude, all the input x and output y vectors are scaled by a fixed amount so that all their components approximately lie within the $[-1, 1]$ range prior to feeding them to the network.

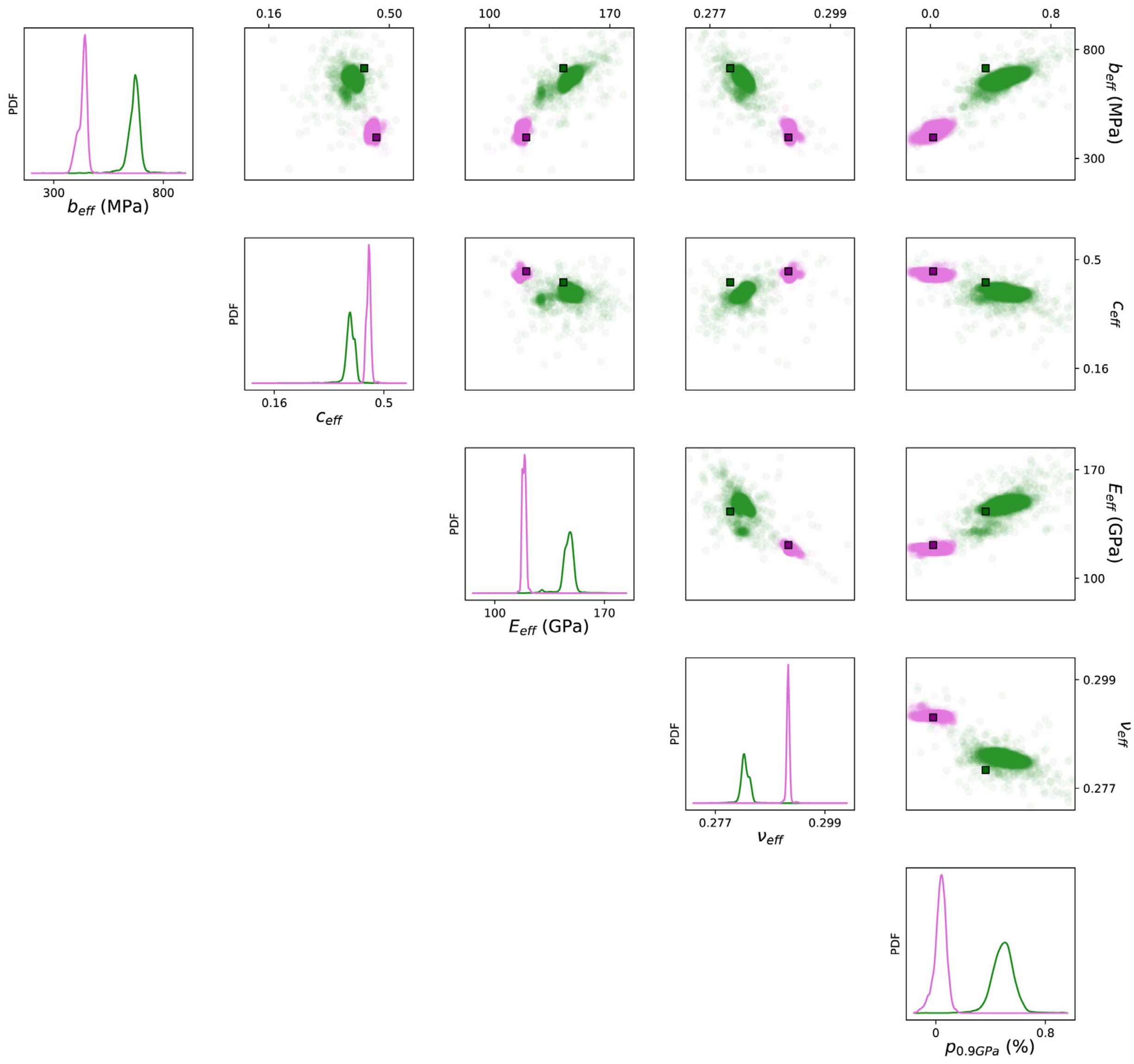
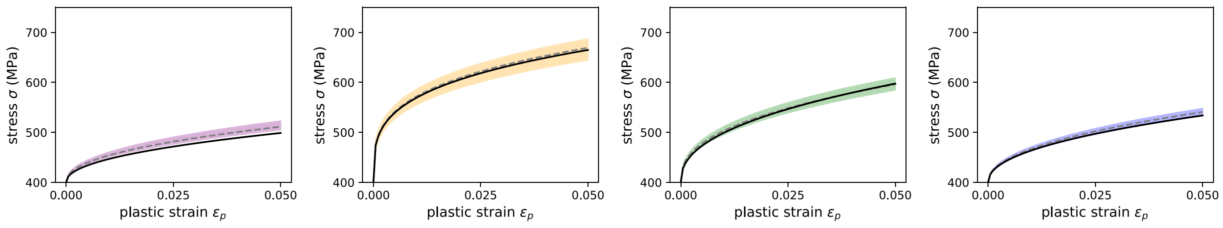


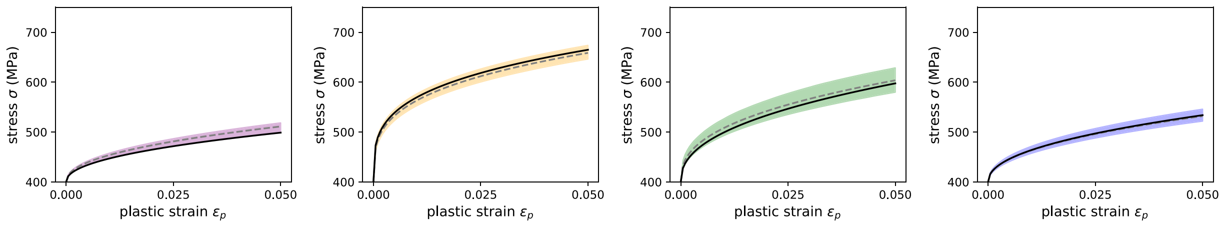
Fig. 12. Pair-plot of distributions of predicted properties (i.e., $y^{(i)} = f^{\omega^{(i)}}(x)$ where $\omega^{(i)} \sim q_\theta(\omega) \approx p(\omega|\mathcal{D})$) for a couple of test data x. Green and purple indicate the two different sets of test data. The squares represent the true values. The diagonal plots show the marginal densities of the posterior predictive densities while the off-diagonal show scatter plots between pairs of output quantities. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

amount of data. One can again observe from Fig. 13 that this predicted epistemic uncertainty around the quantity of interest decreases as the amount of training data increases.

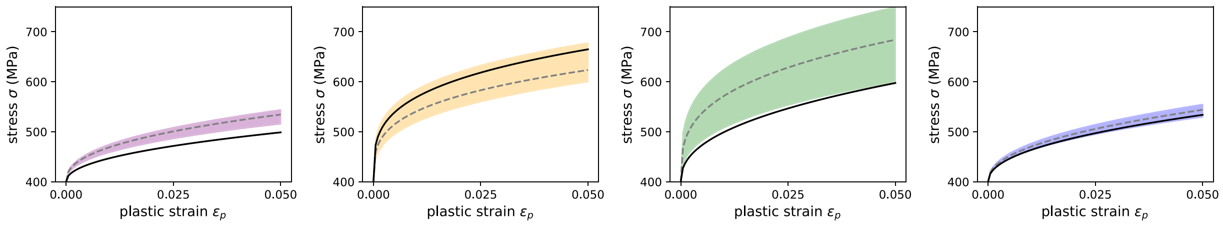
In the second set of experiments, the behavior of the probabilistic regressor is studied in the case where the test data is sampled from a different distribution than the training set, in order to further study the performance of the algorithm in predicting properties and associated uncertainties away from training data. Figs. 14 and 15 illustrate the behavior of the Bayesian NN when the training data is sampled from a different distribution (with smaller support) than the test data. It can be seen that the Bayesian NN predicts larger epistemic uncertainties in this case, informing the user that its predictions may contain some errors due to the mismatch between the training and test data sets. These uncertainty estimates could be further used to guide data collection procedures in design of experiments/active learning settings.



(a)



(b)



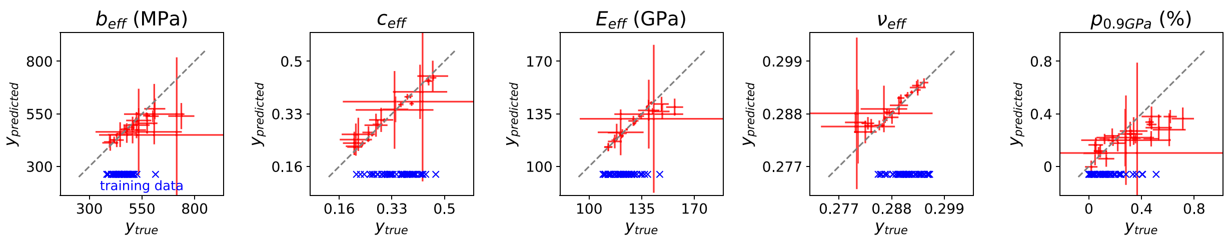
(c)

Fig. 13. Predicted effective plasticity laws and their associated epistemic uncertainties (95% confidence interval) for 4 test data points (from left to right) and a varying amount of training data — size of the training set from top to bottom: (a) 100, (b) 50 and (c) 25 data points. The solid black line represents the true plasticity law to be predicted, the gray dashed line represents the mean prediction.

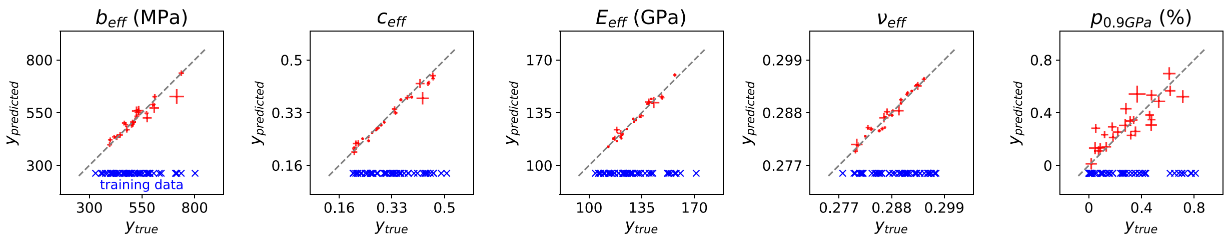
5. Concluding remarks

As machine learning techniques are increasingly integrated within scientific and engineering decision workflows, careful attention must be placed on evaluating the performance of those algorithms to ensure that they provide accurate predictions along with adequate estimates of the uncertainty surrounding those predictions. ML algorithms must be integrated within a robust probabilistic framework that allows the user to account for both aleatoric and epistemic uncertainties, where the former arises from the inherent randomness of the data generation process itself, and the latter arises from data sparsity (reducible uncertainty). This paper examined the performance of variational inference-based algorithms for probabilistic training of neural networks. Very importantly, it was shown that combining various models (e.g., algorithms with various hyper-parameters) via probabilistic model averaging can yield improved mean estimates and more robust uncertainty predictions. These VI-based algorithms are appealing as they achieve an appropriate trade-off between accuracy of the uncertainty estimates and accessible computational cost, compared to MCMC methods for instance. An example related to the prediction of homogenized and localized properties of a 2-phase composite material with randomly placed fibers was considered.

In materials engineering, access to unstructured data is quite frequent, for instance images of microstructures or even 3D representations of materials at various length-scales. In such a setting, (deep) neural networks are the natural choice of algorithms to use for data-based modeling as they are known to perform well on unstructured data such as images. In the present work only structured data was considered, the next logical step will consist in integrating these algorithms within convolutional neural network frameworks (the BayesByBackprop algorithm can be used in this context [60]). Furthermore, it was noticed in the materials example that the assumption of a Gaussian

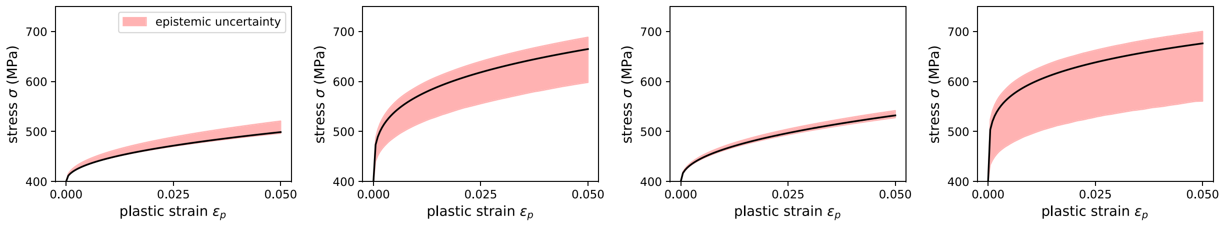


(a)

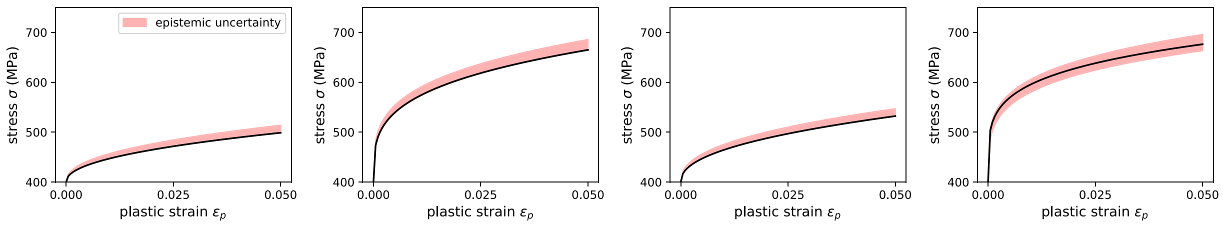


(b)

Fig. 14. Predicted outputs and their associated epistemic uncertainties when the training and test data have (a) different support vs. (b) the same support.



(a)



(b)

Fig. 15. Predicted effective plasticity laws and their associated epistemic uncertainties when the training and test data have (a) different support vs. (b) the same support.

homoscedastic probabilistic model to represent aleatoric uncertainties may not be adequate for certain problems, this also should be further studied.

Declaration of competing interest

None.

Acknowledgments

Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-12-2-0022. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

Appendix A. Improving variational inference by using more complex variational distributions

In variational inference, an approximation $q_\theta(\omega)$ to the intractable posterior pdf $p(\omega|\mathcal{D})$ is found by assuming an analytical form for $q_\theta(\omega)$, for instance a Gaussian pdf, and computing the variational parameters, then minimizing a certain distance metric between the variational density and the true posterior density. When the chosen metric is the KL divergence, this is equivalent to maximizing a lower bound on the evidence of the data (ELBO maximization). A usual approximation for the variational density is a factorized Gaussian pdf:

$$q_\theta(\omega) = \mathcal{N}(\omega; \mu, D) = \prod_{j=1}^{d} \mathcal{N}\left(\omega_j; \mu_j, \sigma_j\right) \tag{27}$$

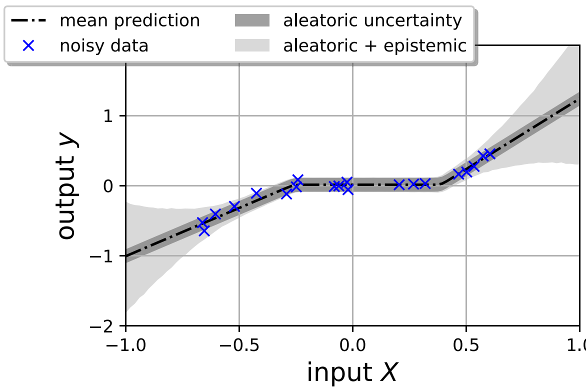
Figs. 16a, 16c illustrate the behavior of the BayesByBackprop algorithm that uses such Gaussian variational approximation. Fig. 16a shows the resulting prediction for the simple 1D problem described in Section 2.2.2, while Fig. 16c plots the approximate posterior densities $\mathcal{N}\left(\omega_j; \mu_j, \sigma_j\right)$, $j = 1 : d$. In this plot W^0, b^0 refer to the kernels and biases of the 100-unit hidden layer, W^1, b^1 refer to the kernels and biases of the output layer. As mentioned in [28], it is important to study separately these weights/biases as they govern various parts of the network and can scale independently of each other. During training, the network finds a few weights that are being learned with small variance. These few weights govern the mean function being learned by the network. Meanwhile, a large portion of the weights share a similar posterior approximation, with larger variance; those weights dictate the uncertainty prediction around the mean. These considerations can be used to prune the network as in [33], if one is solely interested in the mean prediction and not the uncertainty around it. Also, storage costs and potentially training costs could be greatly reduced by using the fact that many weights share the same mean and variance, thus greatly reducing the number of parameters to be stored/learned.

As observed in Section 2.2.2, the BayesByBackprop algorithm with factorized Gaussian variational approximation tends to underestimate posterior uncertainties. A possible remedy is to consider more complex variational approximations. As part of this work, a couple of improvements were considered:

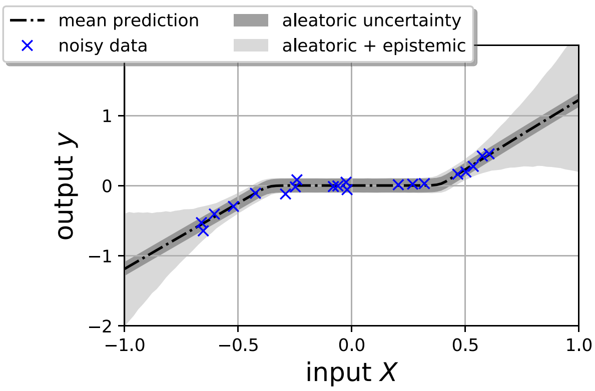
- introducing some correlation within the Gaussian variational density, i.e., $q_\theta(\omega) = \mathcal{N}(\omega; \mu, \Sigma)$ where Σ is non-diagonal,
- using a mixture model as variational density, i.e. for a Gaussian mixture: $q_\theta(\omega) = \sum_{k=1}^{K} \pi^k \mathcal{N}\left(\omega; \mu^k, D^k\right)$.

Introducing correlation within the Gaussian variational density increases substantially the number of parameters to be learned since the number of independent parameters in a full-covariance matrix Σ scales quadratically with the dimension. The number of parameters can be reduced by assuming a low-rank covariance matrix, i.e., $\Sigma = D + AA^T$ where $D = \text{diag}\left(\left[\sigma_1^2, \ldots, \sigma_d^2\right]\right)$ and $A \in \mathbb{R}^{(d,r)}$ with $r \ll d$. Figs. 16b, 16d illustrate the effect of introducing such low-rank correlation ($r = 5$) in the variational approximation for a 1D benchmark problem first presented in Section 2.2.2. Most of the correlation is introduced within the low-dimensional subspace of weights that govern the mean behavior of the network, yielding marginal densities with larger variance for those few weights. However adding such low-dimensional correlation does not substantially increase the overall uncertainty estimation.

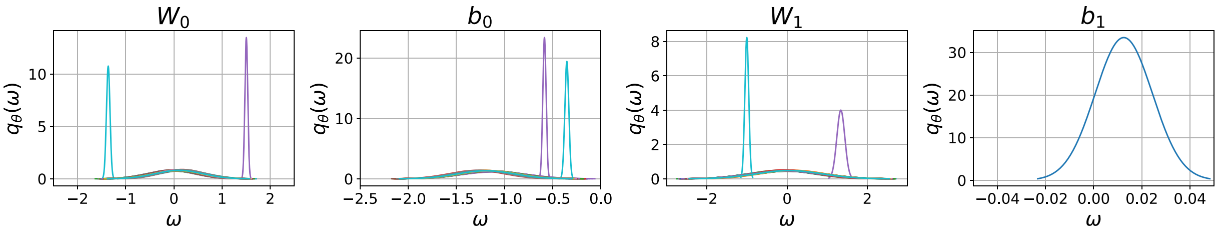
Using mixture models instead of Gaussian variational approximations allows us to capture several modes of the target posterior. However, the entropy term $\mathrm{E}_q[\ln q]$ in the Kullback–Leibler divergence (Eq. (9)) becomes intractable for a mixture model, rendering the ELBO optimization impractical to perform in this context. Various strategies have been introduced in the variational inference and machine learning literature to deal with this



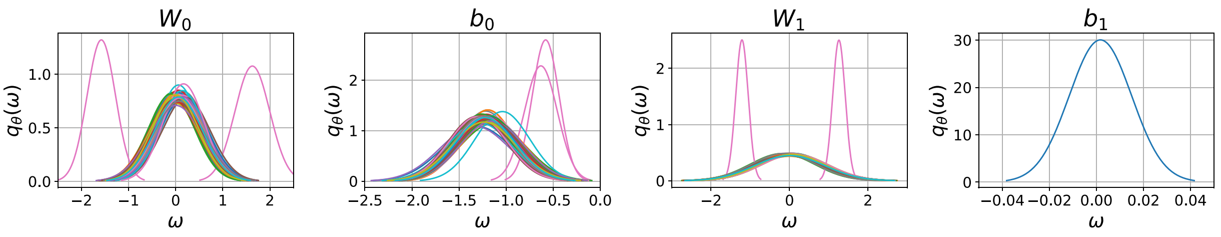
(a)



(b)



(c)



(d)

Fig. 16. Results of the variational inference training for a simple 1D function when using a fully factorized Gaussian variational approximation vs. a correlated Gaussian. (a) Prediction with uncertainty using a factorized Gaussian approximation, (b) Prediction with uncertainty using a low-rank correlated Gaussian approximation, (c) Weight distributions for the factorized Gaussian approximation, (d) Weight distributions for the low-rank Gaussian approximation.

entropy term (e.g., [61]). For simplicity, in our experiments we used a restricted mixture of Gaussians (all components have the same weight), and we replace the entropy term with a lower bound [61,62] $E_q[\ln q] \geq -\sum_k \pi^k \ln \sum_j \pi^j \int q^k(\omega) q^j(\omega) d\omega$ where $q^k(\omega)$ are the components of the mixture, i.e., Gaussians. In this setting, the parameters to be learned in the variational approximation are the parameters (mean and diagonal covariance) of all Gaussian components. Fig. 17 shows the uncertainty predictions achieved with such a variational approximation for the two simple 1D functions described above. Such approximations can account for some multi-modality in the posterior and thus provide a better uncertainty estimate in some cases, such as the interpolation problem considered herein (right plot). In certain cases such as the 1D extrapolation problem, the various components converge to posterior distributions that are equivalent, i.e., yield the same output predictions. This happens because NNs are highly over-parameterized and there thus exist many equivalent configurations of the network that will yield the same predictions. In such case, which is basically equivalent to a collapse of the posterior to a single Gaussian, using a mixture will yield the same results as using one VI-trained network (left plot of Fig. 17). In this paper, we propose instead to average over various VI-based algorithms that use different hyper-parameters, in particular different α parameters in the $\alpha-$BlackBox algorithm, since these algorithms yield output predictions with different levels of uncertainty.

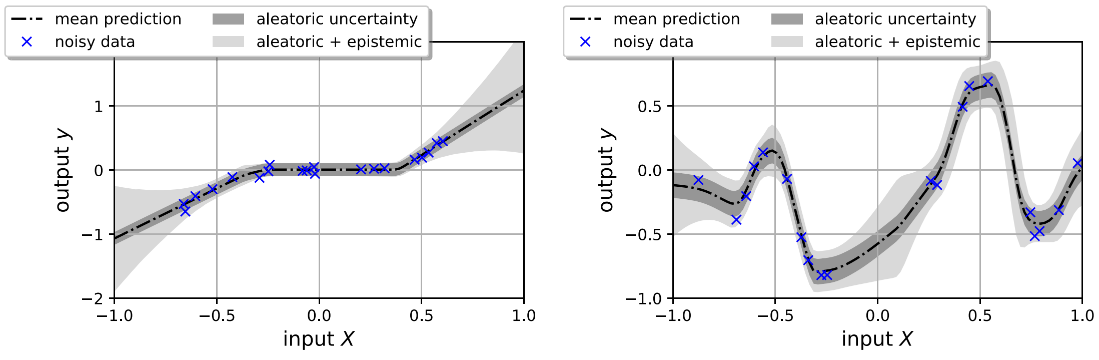


Fig. 17. Results of the variational inference training when assuming a Gaussian vs. a mixture of Gaussians for the variational approximation, for two 1D problems.

Appendix B. Comments on implementation of algorithms

In this section, we share some details on implementation of the various algorithms discussed in the paper. Algorithms were coded in Python/TensorFlow and are available on GitHub (https://github.com/AudOlivier/UQ_in_ML).

B.1. VI based algorithms

- The parameters θ being optimized for each Gaussian are its mean μ, and ρ such that $\sigma = \ln(1 + \exp(\rho))$ so that σ is always non-negative [34],
- The initial values for the mean parameters μ of the variational density are sampled in the same way one would initialize the kernels and biases of a non-probabilistic neural network, i.e. Xavier normal initializer, while the initial values for the σ's (or equivalently ρ's) are set to very small values. This makes the stochastic optimizer initially resembles a point estimator method, i.e. a non-probabilistic neural network, that quickly finds a solution for the mean parameters μ with good predictive properties on the training data, then progressively increases the variance parameters σ to capture the uncertainty around the mean of the approximate posterior [50],
- Optimization is performed via gradient descent using the Adam optimizer and a learning rate of 0.005. Convergence is assessed by looking at the loss function and convergence of the variational parameters — for the 1D extrapolation and interpolation problems, the models are trained for 5000 and 8000 epochs respectively.
- We use N = 20 MC samples from the variational distribution to compute the cost, in the BayesByBackprop paper [34] trials were made with up to 10 MC samples.

B.2. MCMC based algorithms

For the 1D benchmark problems, the Hamiltonian Monte Carlo [38] is run for comparison with VI-based methods. We run 5 independent chains and the hyper-parameters of the algorithm were chosen so as to achieve an acceptance rate between 0.6 and 0.9, as advised in the Tensorflow documentation. One drawback of MCMC algorithms is that convergence is more difficult to assess than for VI-based algorithms. A combination of diagnostics must be used, for instance the effective sample size (ESS) and the $\hat{\mathrm{R}}$ diagnostics. In our experiments, we use the thresholds used in [40], i.e., we target an effective sample size of 400 or more and a value of $\hat{\mathrm{R}}$ of 1.01 or less. In our experiments, we observed, in accordance with the discussion in [40], that convergence of the parameters to their true posterior pdf is very hard to achieve; in our case running the chains for more than 100 000 iterations does not yield convergence of the parameters according to the above thresholds on ESS and $\hat{\mathrm{R}}$. However, even though the chains have apparently not converged to the true posterior $p(\omega|\mathcal{D})$, very good approximations of the predictive

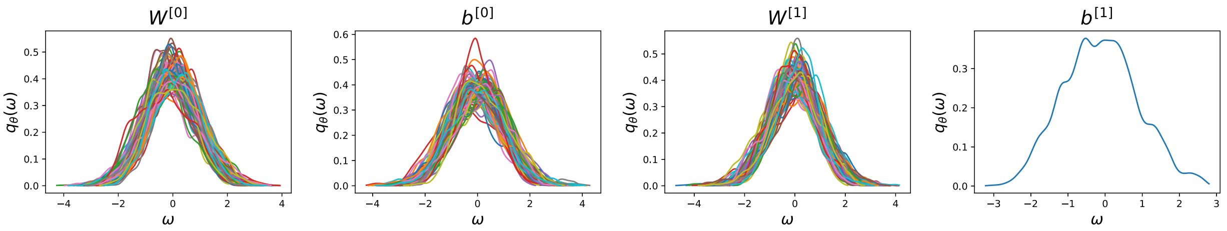


Fig. 18. Marginal posterior densities from HMC for the 1D extrapolation problem.

density $p(y|x, \mathcal{D})$ can be obtained. Thus in our experiments we chose a few points $x^\star$ – 5 points, away from data – where to evaluate the output prediction, and ran the chains until convergence was achieved for the functionals $\{f^{\omega^{(i)}}(x^\star)\}_{i=1:N_{samples}}$, instead of looking at the parameter chains $\{\omega^{(i)}\}_{i=1:N_{samples}}$. For the 1D extrapolation problem (with 301 parameters), such convergence was achieved in about 140 000 iterations; and about 200 000 iterations for the interpolation problem (with 901 parameters). In terms of computational time, training a single VI-based NN is typically faster than running HMC. The procedure proposed herein, i.e., running several VI algorithms, appears to have ballpark similar compute times as running HMC.

For completeness, Fig. 18 shows the marginal densities of the NN weights learned via HMC. Contrary to VI-based algorithms, where a few weights dominate and explain the mean function (see Appendix A), here the marginal densities are still close to the standard Gaussian prior, but there is non-zero correlation between all the weights. This indicates that methods of pruning available to VI-based algorithms (see Appendix A) will not be applicable in this case, and that all the posterior samples will need to be saved.

References

[1] Y. Liu, T. Zhao, W. Ju, S. Shi, Materials discovery and design using machine learning, J. Materiomics 3 (3) (2017) 159–177, http://dx.doi.org/10.1016/j.jmat.2017.08.002.

[2] T. Mueller, A.G. Kusne, R. Ramprasad, Machine learning in materials science: recent progress and emerging applications, in: A.L. Parrill, K.B. Lipkowitz (Eds.), Reviews in Computational Chemistry, Vol. 29, John Wiley & Sons, Inc, 2016, http://dx.doi.org/10.1002/9781119148739.ch4.

[3] R. Ramprasad, R. Batra, G. Pilania, A. Mannodi-Kanakkithodi, C. Kim, Machine learning in materials informatics: recent applications and prospects, Comput. Mater. 3 (54) (2017) http://dx.doi.org/10.1038/s41524-017-0056-5.

[4] A. Agrawal, A. Choudhary, Perspective: Materials informatics and big data: Realization of the “fourth paradigm” of science in materials science, APL Mater. 4 (5) (2016) 053208, http://dx.doi.org/10.1063/1.4946894.

[5] S.R. Kalidindi, A.J. Medford, D.L. McDowell, Vision for data and informatics in the future materials innovation ecosystem, JOM 68 (8) (2016) 2126–2137, http://dx.doi.org/10.1007/s11837-016-2036-5.

[6] B. Meredig, A. Agrawal, S. Kirklin, J.E. Saal, J.W. Doak, A. Thompson, K. Zhang, A. Choudhary, C. Wolverton, Combinatorial screening for new materials in unconstrained composition space with machine learning, Phys. Rev. B 89 (2014) 094104, http://dx.doi.org/10.1103/PhysRevB.89.094104.

[7] A. Seko, T. Maekawa, K. Tsuda, I. Tanaka, Machine learning with systematic density-functional theory calculations: Application to melting temperatures of single- and binary-component solids, Phys. Rev. B 89 (2014) 054303, http://dx.doi.org/10.1103/PhysRevB.89.054303.

[8] H.K.D.H. Bhadeshia, R.C. Dimitriu, S. Forsik, J.H. Pak, J.H. Ryu, Performance of neural networks in materials science, Mater. Sci. Technol. 25 (2009) 504–510, http://dx.doi.org/10.1179/174328408X311053.

[9] A. Gupta, A. Cecen, S. Goyal, A.K. Singh, S.R. Kalidindi, Structure-property linkages using a data science approach: Application to a non-metallic inclusion/steel composite system, Acta Mater. 91 (2015) 239–254, http://dx.doi.org/10.1016/j.actamat.2015.02.045.

[10] R. Liu, Y.C. Yabansu, A. Agrawal, S.R. Kalidindi, A.N. Choudhary, Machine learning approaches for elastic localization linkages in high-contrast composite materials, Integr. Mater. Manuf. Innov. 4 (2015) 192–208, http://dx.doi.org/10.1186/s40192-015-0042-z.

[11] Y. He, M. Razi, C. Forestiere, L. Dal Negro, R.M. Kirby, Uncertainty quantification guided robust design for nanoparticles’ morphology, Comput. Methods Appl. Mech. Engrg. 336 (2018) 578–593, http://dx.doi.org/10.1016/j.cma.2018.03.027.

[12] M.A. Bessa, R. Bostanabad, Z. Liu, A. Hu, Daniel W. Apley, L.C. Brinson, W. Chen, W.K. Liu, A framework for data-driven analysis of materials under uncertainty: countering the curse of dimensionality, Comput. Methods Appl. Mech. Engrg. 320 (2017) 633–667, http://dx.doi.org/10.1016/j.cma.2017.03.037.

[13] L. Ward, A. Agrawal, A. Choudhary, C. Wolverton, A general-purpose machine learning framework for predicting properties of inorganic materials, Npj Comput. Mater. 2 (2016) 16028, http://dx.doi.org/10.1038/npjcompumats.2016.28.

[14] W-C. Lu, X-B. Ji, M-J. Li, L. Liu, B-H. Yue, L-M. Zhang, Using support vector machine for materials design, Adv. Manuf. 1 (2) (2013) 151–159, http://dx.doi.org/10.1007/s40436-013-0025-2.

[15] X. Lu, D.G. Giovanis, J. Yvonnet, V. Papadopoulos, F. Detrez, J. Bai, A data-driven computational homogenization method based on neural networks for the nonlinear anisotropic electrical response of graphene/polymer nanocomposites, Comput. Mech. (2019) http://dx.doi.org/10.1007/s00466-018-1643-0.
[16] L.M. Ghiringhelli, J. Vybiral, S.V. Levchenko, C. Draxl, M. Scheffler, Big data of materials science: Critical role of the descriptor, Phys. Rev. Lett. 114 (2015) 105503, http://dx.doi.org/10.1103/PhysRevLett.114.105503.
[17] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (2015) 436–444, http://dx.doi.org/10.1038/nature14539.
[18] R. Kondo, S. Yamakawa, Y. Masuoka, S. Tajima, R. Asahi, Microstructure recognition using convolutional neural networks for prediction of ionic conductivity in ceramics, Acta Mater. 141 (2017) 29–38, http://dx.doi.org/10.1016/j.actamat.2017.09.004.
[19] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707, http://dx.doi.org/10.1016/j.jcp.2018.10.045.
[20] E. Haghighat, R. Juanes, Sciann: A keras/tensorflow wrapper for scientific computations and physics-informed deep learning using artificial neural networks, Comput. Methods Appl. Mech. Engrg. 373 (2021) 113552, http://dx.doi.org/10.1016/j.cma.2020.113552.
[21] X. Zhang, F. Xie, T. Ji, Z. Zhu, Y. Zheng, Multi-fidelity deep neural network surrogate model for aerodynamic shape optimization, Comput. Methods Appl. Mech. Engrg. 373 (2021) 113485, http://dx.doi.org/10.1016/j.cma.2020.113485.
[22] R.K. Tripathy, I. Bilionis, Deep UQ: Learning deep neural network surrogate models for high dimensional uncertainty quantification, J. Comput. Phys. 375 (2018) 565–588, http://dx.doi.org/10.1016/j.jcp.2018.08.036.
[23] N. Lubbers, T. Lookman, K. Barros, Inferring low-dimensional microstructure representations using convolutional neural networks, Phys. Rev. E 96 (2017) 052111, http://dx.doi.org/10.1103/PhysRevE.96.052111.
[24] X. Li, Y. Zhang, R. Zhao, C. Burkhart, C.L. Brinson, W. Chen, A transfer learning approach for microstructure reconstruction and structure-property predictions, Sci. Rep. 8 (2018) 13461, http://dx.doi.org/10.1038/s41598-018-31571-7.
[25] B.L. DeCost, T. Francis, E.A. Holm, Exploring the microstructure manifold: Image texture representations applied to ultrahigh carbon steel microstructures, Acta Mater. 133 (2017) 30–40, http://dx.doi.org/10.1016/j.actamat.2017.05.014.
[26] Z. Nie, H. Jiang, L.B. Kara, Deep learning for stress field prediction using convolutional neural networks, 2018, arXiv:1808.08914.
[27] A. Der Kiureghian, O. Ditlevsen, Aleatory or epistemic? Does it matter?, in: Special Workshop on Risk Acceptance and Risk Communication, 2007.
[28] David J.C. MacKay, Bayesian Methods for Adaptive Models (Ph.D. thesis), California Institute of Technology, 1992.
[29] H.M.D. Kabir, A. Khosravi, M.A. Hosen, S. Nahavandi, Neural network-based uncertainty quantification: A survey of methodologies and applications, IEEE Access 6 (2018) 36218–36234, http://dx.doi.org/10.1109/ACCESS.2018.2836917.
[30] E. Mazloumi, G. Rose, G. Currie, S. Moridpour, Prediction intervals to account for uncertainties in neural network predictions: Methodology and application in bus travel time prediction, Eng. Appl. Artif. Intell. 24 (2011) 534–542, http://dx.doi.org/10.1016/j.engappai.2010.11.004.
[31] K.S. Kasiviswanathan, K.P. Sudheer, J. He, Quantification of prediction uncertainty in artificial neural network models, in: Artificial Neural Network Modelling. Studies in Computational Intelligence, Vol. 628, Springer, Cham, 2016, pp. 145–159.
[32] R.M. Neal, Bayesian Learning for Neural Networks (Ph.D. thesis), University of Toronto, 1995.
[33] A. Graves, Practical variational inference for neural networks, in: Advances in Neural Information Processing Systems 24 (NIPS 2011), 2011, pp. 2348–2356.
[34] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, Daan Wierstra, Weight uncertainty in neural networks, in: Proceedings of the 32nd International Conference on Machine Learning, Vol. 37, JMLR: W&CP, Lille, France, 2015.
[35] Yarin Gal, Uncertainty in Deep Learning (Ph.D. thesis), University of Cambridge, 2016.
[36] I. Osband, C. Blundell, A. Pritzel, B. Van Roy, Deep exploration via bootstrapped dqn, in: 30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, 2016.
[37] I. Osband, J. Aslanides, A. Cassirer, Randomized prior functions for deep reinforcement learning, in: In 32nd Conference on Neural Information Processing Systems (NIPS 2018), Montreal, Canada, 2018.
[38] R. Neal, MCMC Using hamiltonian dynamics, in: Steve Brooks, Andrew Gelman, Galin L. Jones, Xiao-Li Meng (Eds.), Handbook of Markov Chain Monte Carlo, Routledge Handbooks Online, 2011.
[39] L. Yang, X. Meng, G.E. Karniadakis, B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data, J. Comput. Phys. 425 (2021) 109913, http://dx.doi.org/10.1016/j.jcp.2020.109913.
[40] T. Papamarkou, J.M. Hinkle, T. Young, D. Womble, Challenges in Markov chain Monte Carlo for Bayesian neural networks, 2019, arXiv:1910.06539.
[41] C. Leibig, V. Allken, M.S. Ayhan, P. Berens, S. Wahl, Leveraging uncertainty information from deep neural networks for disease detection, Sci. Rep. 7 (2017) 17816, http://dx.doi.org/10.1038/s41598-017-17876-z.
[42] S. Ryu, Y. Kwon, W.Y. Kim, Uncertainty quantification of molecular property prediction with Bayesian neural networks, 2019, arXiv:1903.08375.
[43] D. Zhang, L. Lu, L. Guo, G.E. Karniadakis, Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems, J. Comput. Phys. 397 (2019) 108850, http://dx.doi.org/10.1016/j.jcp.2019.07.048.
[44] I. Osband, Risk versus uncertainty in deep learning: bayes, bootstrap and the dangers of dropout, in: Workshop on Bayesian Deep Learning, NIPS, Barcelona, Spain, 2016.
[45] T. Pearce, N. Anastassacos, M. Zaki, A. Neely, Bayesian Inference with anchored ensembles of neural networks, and application to exploration in reinforcement learning, in: Exploration in Reinforcement Learning Workshop At the 35th International Conference on Machine Learning, 2018.
[46] A.M. Stuart, Inverse problems: A Bayesian perspective, Acta Numer. 19 (2010) 451559.

[47] Z. Wang, T. Ren, J. Zhu, B. Zhang, Function space particle optimization for Bayesian neural networks, in: ICLR, 2019.
[48] D. Flam-Shepherd, J. Requeima, D. Duvenaud, Characterizing and warping the function space of Bayesian neural networks, in: Bayesian Deep Learning (NeurIPS 2018), 2018.
[49] B. Lakshminarayanan, A. Pritzel, C. Blundell, Simple and scalable predictive uncertainty estimation using deep ensembles, in: Workshop on Bayesian Deep Learning, NIPS 2016, Barcelona, Spain, 2016.
[50] J. Hernández-Lobato, Y. Li, M. Rowland, T. Bui, D. Hernández-Lobato, R. Turner, Black-box alpha-divergence minimization, in: Proceedings of the 33rd International Conference on Machine Learning, Vol. 48, PMLR, 2016, pp. 1511–1520.
[51] T. Minka, Power EP, Technical Report MSR-TR-2004-149, Microsoft Research Ltd., Cambridge, UK, 2004.
[52] A.D. Cobb, M.D. Himes, F. Soboczenski, S. Zorzan, M.D. O'Beirne, A.G. Baydin, Y. Gal, S. Domagal-Goldman, G. Arney, D. Angerhausen, An ensemble of Bayesian neural networks for exoplanetary atmospheric retrieval, Astron. J. 158 (1) (2019) http://dx.doi.org/10.3847/1538-3881/ab2390.
[53] N. Chitsazan, A.A. Nadiri, F.T.-C. Tsai, Prediction and structural uncertainty analyses of artificial neural networks using hierarchical Bayesian model averaging, J. Hydrol. 528 (2015) 52–62, http://dx.doi.org/10.1016/j.jhydrol.2015.06.007.
[54] J-P. Vila, V. Wagner, P. Neveu, Bayesian Nonlinear model selection and neural networks: A conjugate prior approach, IEEE Trans. Neural Netw. 11 (2) (2000).
[55] Y. Yao, A. Vehtari, D. Simpson, A. Gelman, Using stacking to average Bayesian predictive distributions (with discussion), Bayesian Anal. 13 (3) (2018) 917–1003, http://dx.doi.org/10.1214/17-BA1091.
[56] A. Schöniger, T. Wöhling, L. Samaniego, W. Nowak, Model selection on solid ground: Rigorous comparison of nine ways to evaluate Bayesian model evidence, Water Resour. Res. 50 (12) (2014) 9484–9513, http://dx.doi.org/10.1002/2014WR016062.
[57] F.T.-C. Tsai, X. Li, Inverse groundwater modeling for hydraulic conductivity estimation using Bayesian model averaging and variance window, Water Resour. Res. 44 (9) (2008) http://dx.doi.org/10.1029/2007WR006576.
[58] A. Gelman, J. Hwang, A. Vehtari, Understanding predictive information criteria for Bayesian models, Stat. Comput. 24 (2014) 997–1016, http://dx.doi.org/10.1007/s11222-013-9416-2.
[59] Y. Li, J.M. Hernández-Lobato, R. Turner, Stochastic expectation propagation, in: Advances in Neural Information Processing Systems (NIPS), 2015, pp. 2323–2331, http://dx.doi.org/10.17863/CAM.21346.
[60] K. Shridhar, F. Laumann, M. Liwicki, A comprehensive guide to bayesian convolutional neural network with variational inference, 2019, arXiv:1901.02731.
[61] E. Nalisnick, P. Smyth, Variational inference with stein mixtures, in: Advances in Approximate Bayesian Inference, NIPS 2017 Workshop, 2017.
[62] S.J. Gershman, M.D. Hoffman, D.M. Blei, Nonparametric variational inference, in: Proceedings of the 29th International Conference on Machine Learning, 2012, pp. 235–242.