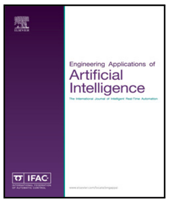
Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai



Uncertainty quantification in Neural Networks by Approximate Bayesian Computation: Application to fatigue in composite materials

Juan Fernández [a,*], Manuel Chiachío [a], Juan Chiachío [a], Rafael Muñoz [c], Francisco Herrera [b,d]

[a] Department of Structural Mechanics and Hydraulic Engineering, Andalusian Research Institute in Data Science and Computational Intelligence (DaSCI), University of Granada (UGR), Granada 18001, Spain
[b] Department of Computer Science and Artificial Intelligence, Andalusian Research Institute in Data Science and Computational Intelligence (DaSCI), University of Granada (UGR), Granada 18071, Spain
[c] Department of Civil Engineering, University of Granada (UGR), Granada 18001, Spain
[d] Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 21589, Saudi Arabia

ARTICLE INFO

Keywords:
Bayesian Neural Network
Approximate Bayesian Computation
Subset Simulation
Uncertainty quantification
Gradient-free training
Non-parametric formulation

ABSTRACT

Modern machine learning algorithms excel in a great variety of tasks, but at the same time, it is also known that those complex models need to deal with uncertainty from different sources. Consequently, understanding if the model is indeed making accurate predictions or simply guessing at random is not trivial, and measuring the confidence bounds becomes very important. Bayesian machine learning seems to provide the solution, however, many of the state-of-the-art Bayesian algorithms use rigid parametric representations of the uncertainty where the learning process depends on the gradient of a predefined cost function. In this article, a new gradient-free training algorithm based on Approximate Bayesian Computation by Subset Simulation is proposed, where the likelihood function and the weights are defined by non-parametric formulations, resulting in a flexible and fairer representation of the uncertainty. The experiments, specially the engineering case study on composite materials subject to fatigue damage, show the ability of the proposed algorithm to consistently reach accurate predictions while avoiding gradient related instabilities, and most importantly, it provides a realistic and coherent quantification of the uncertainty represented by confidence bounds. All this may lead to a reduction of safety factors in engineering problems, and in general, allows us to make well-informed decisions in situations with a high degree of uncertainty and risk. A comparison with the state-of-the-art Bayesian Neural Networks is also carried out.

1. Introduction

Artificial Intelligence (AI) has experienced a fast pace development during the last decade and promises large benefits in many fields of different nature. Particularly, Deep Neural Networks (DNN) have revolutionized the world of machine learning, with more complex models that have made computer vision (Voulodimos et al., 2018) and speech recognition (Arora and Singh, 2012) a reality, in some cases even reaching human accuracy (Sturman et al., 2020). In engineering, hybrid models have dramatically improved wind speed forecasting (Altan et al., 2021) or digital currency forecasting (Altan et al., 2019). As a result, we often encounter this technology in our daily life, in the form of email classification (Awad and Elseuofi, 2011), fraud prevention (Sadgali et al., 2019) or border controls (Carlos-Roca et al., 2018), to name but a few examples. However, the predictions made by these models are subject to uncertainty which is critical in applications where small variations might cause disproportionate consequences, such as in safety evaluation of power plants (Varshney and Alemzadeh, 2017) or trajectory and safety assessment in civil aviation (Zhang and Mahadevan, 2020). The motivation of this paper lies in the importance of quantifying the *degree of belief* on the model predictions, which is of great value for the subsequent decision-making process.

Delving into the rational quantification of the uncertainty, this can be classified into two categories, *epistemic* and *aleatory* (Hüllermeier and Waegeman, 2021). In machine learning, epistemic uncertainty mainly refers to the lack of training data in some areas of the input domain. In addition, modellers have to tackle the epistemic uncertainty when choosing the optimum Artificial Neural Network (ANN) architecture, which delivers the right balance between model complexity and low generalization error (Beck, 2010). On the other hand, the aleatory uncertainty refers to the randomness inherent in nature and implicit in the data, such as noise in the measurements. While aleatory uncertainty is mostly irreducible, epistemic related to lack of knowledge can be mitigated in some ways, for instance gathering more data (Depeweg

* Corresponding author.
E-mail address: juanfdez@ugr.es (J. Fernández).

https://doi.org/10.1016/j.engappai.2021.104511
Received 2 June 2021; Received in revised form 28 September 2021; Accepted 12 October 2021
Available online 5 November 2021

et al., 2018). In all cases, quantifying the total uncertainty in the predictions provides valuable information (Ghahramani, 2015).

A branch of methods have appeared in the literature for quantifying uncertainty in ANN. Among those, Bayesian Neural Networks (BNN) are experiencing an increase in popularity within the machine learning community. BNN emerged in the early 90s to robustly quantify uncertainty in the neural network modelling using the *Bayesian Inverse Problem* for updating the network parameters (Buntine and Weigend, 1991; MacKay, 1992; Neal, 1992, 1996; Lampinen and Vehtari, 2001). The best known training methods used in BNNs are the Variational Inference method (VI) (Graves, 2011; Hoffman et al., 2013; Wang et al., 2020) (more specifically Bayes by Backprop (Blundell et al., 2015; Jia et al., 2020)), Probabilistic Backpropagation (PBP) (Hernandez-Lobato and Adams, 2015) and Hamiltonian Monte Carlo (HMC) (Benker et al., 2020; Levy et al., 2018). Generally, these methods require the evaluation of the gradient of a cost function using the back-propagation algorithm (Rumelhart et al., 1986), which is prone to suffer from drawbacks like *exploding gradient* (Pascanu et al., 2013) when large derivatives propagate down the model, *vanishing gradient* (Pascanu et al., 2013) if derivatives are small, or *dying ReLU* (Lu, 2020), all of which affect the learning process. It is also common among these methods the adoption of a particular probability model for the likelihood function and the posterior probability density function (PDF) of the parameters, often assumed to be Gaussian, which leads to a constrained quantification of the uncertainty.

This paper proposes a novel technique to train BNNs using the Approximate Bayesian Computation (ABC) method (Marjoram et al., 2003; Del Moral et al., 2012) combined with Subset Simulation (SS) (Au and Beck, 2001), the so called ABC-SS method (Chiachio et al., 2014). The proposed training approach, defined here as *BNN by ABC-SS*, produces samples of the model parameters which are accepted or rejected based on a performance metric. This process is repeated through sequential subsets until a predefined tolerance value is reached. The posterior PDF of the parameters is finally obtained from the samples in the last subset. The advantages of using *BNN by ABC-SS* as a training algorithm are twofold. First, it circumvents the parametric definition of both the likelihood function and the posterior PDF of weights and bias. Such non-parametric formulation provides flexibility to the model and enables a more realistic quantification of the uncertainty (Ghahramani, 2015). Second, the gradient-free nature of the algorithm, which avoids the known gradient-based issues mentioned above. Both benefits originate from the use of ABC, which is improved with SS for computation efficiency (Prangle et al., 2018).

In the literature, ANN and ABC methods can be found in a combined manner, like in Jiang et al. (2017), Radev et al. (2020), Blum and François (2010), Blum et al. (2013) where ANNs, and more recently BNNs (Grazian and Fan, 2020), are used within ABC to select a suitable summary statistics to be applied in the distance function. ABC has also been implemented to obtain the uncertainty of the output in deterministic ANN (Das et al., 2019), or as the optimization method in inverse problems using a trained ANN as the forward model (List and Lewis, 2020). Other Bayesian methods have been used to infer and optimize the hyperparameters of a DNN and machine learning algorithms (Vong et al., 2006; Shin et al., 2020). Nevertheless, and to the best knowledge of the authors, the ABC-SS method has not been used before as the training algorithm for a BNN.

The proposed method has been illustrated with two academic examples and applied to an engineering case study about remaining useful life prediction of carbon fibre reinforced polymer (CFRP) laminates subject to fatigue damage. Also, a comparison on performance with VI (Bayes by Backprop), PBP and HMC is provided. The results of the experiments have validated the main contribution of the proposed method, namely the accurate and flexible quantification of the uncertainty in the observed data, which adds valuable information to the predictions made by the BNN.

The rest of this paper is organized as follows. Section 2 provides a theoretical background, from basic principles of ANN through BNN to the ABC-SS method. Section 3 describes how ABC-SS is adapted to train a BNN and presents two illustrative regression problems. A real case study on remaining useful life prediction of carbon fibre reinforced polymer (CFRP) laminates subject to fatigue cycles (with data taken from the NASA Ames Prognostics Data Repository - CFRP Composites Dataset (Saxena et al., 2008)) is provided in Section 4, including a description of the experimental framework, the results and discussion on the experiment, and a comparison with the state-of-the-art BNN. An overall discussion on the methodology and the experiments is provided in Section 5. And finally, the conclusions are given in Section 6.

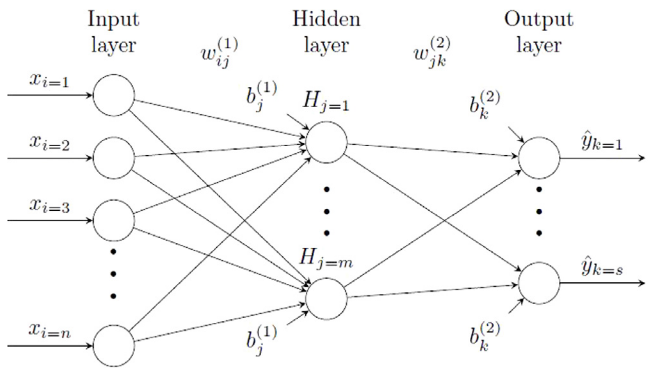


Fig. 1. Generic example of a basic FNN.

2. Background

This section aims to provide the theoretical foundations of this article. Beginning with a short introduction to ANN in Section 2.1, Section 2.2 will dive into the principles and motivations of BNN, explaining how the uncertainty is quantified by using the Bayes theorem. Finally, the ABC-SS method along with its mechanisms to find the posterior distribution of the parameters is described in Section 2.3.

2.1. Artificial neural networks

ANN have been an active line of research in the recent years, however, their invention dates from 1943 and is attributed to Warren McCulloch, a Neurophysiologist, and Walter Pitts, a mathematician (Mcculloch and Pitts, 1943). The principles of ANNs are inspired in the behaviour of biological neurons, although their architecture and mechanisms to process information differ notably.

Many different types of ANN have been developed to specifically solve a diverse set of tasks, such as regression, classification, visual recognition or natural language processing. Feedforward Neural Networks (FNN) are considered the simplest type and the one many others are built upon (Goodfellow et al., 2016). Feedforward models can be understood as a function f, defined by a set of parameters including weights w and bias b, which maps some input information $x \in \mathcal{X} \subset \mathbb{R}^n$ to a predicted output $\hat{y} \in \mathcal{O}$, where $\mathcal{O} \subset \mathbb{R}^l$ for a regression task, thus $\hat{y} = f(x; w, b)$. These models can comprise several layers, where each of them executes a linear transformation of the input information using the parameters w and b, followed by a non-linear transformation using an activation function. The number of layers indicates the depth of our model. Fig. 1 shows a simple FNN, with one input layer, one hidden layer and one output layer. The mapping from inputs to outputs for the generic case in Fig. 1, also known as forward propagation, is formulated as follows:

$$\hat{y}_k = f(x; w, b) = g\left(\sum_{j=1}^{m} w_{jk}^{(2)} h\left(\sum_{i=1}^{n} w_{ij}^{(1)} x_i + b_j^{(1)}\right) + b_k^{(2)}\right) \tag{1}$$

where x_i is the ith input unit; $w_{ij}^{(1)}$, $w_{jk}^{(2)}$, $b_j^{(1)}$ and $b_k^{(2)}$ represent the weights and biases; $\hat{y}_k$ is the kth output neuron; H_j is the jth neuron in the hidden layer; h and g are the activation functions in the hidden and output layers respectively; n is the number of input neurons; m is the number of neurons in the hidden layer; and s the number of output neurons.

The selection of the activation functions in the hidden units is an active area of research, and it is difficult to know which one will perform best, thus a trial and error process is often followed. Rectified Linear Units (ReLU) have proved to work well in a wide variety of models, while others such as Maxout Units (Goodfellow et al., 2013) have the potential to reduce the number of parameters required. On the contrary, the activation function in the output units is task-specific and its choice is critical for a good performance of the FNN.

With regard to the output, a FNN model defines a probability distribution $p(y|x; w, b)$, where $y \in \mathcal{Y}$ represents the observed outputs in a training data set $\mathcal{D} = (x, y) \in \mathcal{X} \times \mathcal{Y}$, and in most cases, the principle of maximum likelihood estimation is used to learn the parameters w and b. This is equivalent to minimizing the negative log-likelihood, namely the cost function $C(w, b) = -\log p(y|x)$. The performance of a FNN is measured by such cost function, whose form depends on the output units, and therefore, on the task. A regularization method is often used to avoid overfitting and reduce the generalization error.

In many cases, the parameters of a FNN are learnt via a training algorithm based on descending the cost function using the gradient, such as stochastic gradient descent. Information from the cost function flows backward, computing the gradient using the back-propagation algorithm (Rumelhart et al., 1986).

2.2. Bayesian neural networks

From a frequentist point of view, the parameters w and b of an ANN are assumed to be known deterministically with a single value which we want to find. Given a training data set $\mathcal{D} = (x, y)$, a learning algorithm could be used, as specified in Section 2.1, to approximate the optimal value of the parameters. However, there is an implicit uncertainty about the value of those parameters that is not covered by the frequentist approach.

If a Bayesian interpretation is followed, such uncertainty is considered and the objective is no longer to find the *true* value of the parameters but instead, a distribution of plausible values of the parameters that are consistent with the training data set. Under this perspective, neural network predictions are obtained with quantified uncertainty, by considering the most plausible values of the parameters rather than a single one. In this context, the parameters of a Bayesian Neural Network (BNN) are inferred using a probability logic approach based on the Bayes' theorem (Bayes, 1763; Laplace, 1812; Jeffreys, 1961; Cox, 1946).

From a mathematical point of view, a BNN provides a probabilistic output $\hat{y}$ based on the uncertainty about a set of model parameters $\theta = \{w, b\} \in \Theta \subseteq \mathbb{R}^d$ given a model class $\mathcal{M}$. In this framework, the model class $\mathcal{M}$ refers to the network architecture, namely, the number of layers and neurons per layer; the activation functions in each of the hidden and output layers; along with the *prior information* about the model parameters θ, referred to as $p(\theta|\mathcal{M})$. Using Bayes' theorem, this prior information can be updated according to the training data set $\mathcal{D}(x, y)$, as follows:

$$p(\theta|\mathcal{D}, \mathcal{M}) = \frac{p(\mathcal{D}|\theta, \mathcal{M})\, p(\theta|\mathcal{M})}{p(\mathcal{D}|\mathcal{M})} \tag{2}$$

where $p(\theta|\mathcal{D}, \mathcal{M})$ is the posterior PDF of the model parameters given the data, and $p(\mathcal{D}|\theta, \mathcal{M})$ is known as the likelihood function. This function measures how likely the model $\mathcal{M}$ specified by the parameters θ reproduces the observed data $\mathcal{D}$. As described above, the term $p(\theta|\mathcal{M})$ is the prior PDF which quantifies our initial belief about the plausibility of the values of parameters θ given a model class $\mathcal{M}$, and automatically enforces a regularization effect thus preventing the over-fitting of the network model. Finally, the term $p(\mathcal{D}|\mathcal{M})$ is known as the *evidence* and represents how likely the data $\mathcal{D}$ is reproduced if model class $\mathcal{M}$ is adopted. The computation of the *evidence* comprises the evaluation of a multidimensional integral which is analytically intractable in most of the cases. However, stochastic simulation methods such as Markov chain Monte Carlo (MCMC) (Neal, 1993; Gilks et al., 1996) can be used to draw samples from the posterior while circumventing the evaluation of the evidence.

Besides, in many cases the evaluation of the likelihood function is computationally prohibitive or even analytically intractable. However, methods such as ABC may be used to approximate the posterior distribution of the parameters.

2.3. Approximate Bayesian computation by subset simulation

ABC methods were born with the purpose of evaluating the posterior distribution of the parameters in those cases where the likelihood function is analytically intractable (Marin et al., 2012). Also known as *likelihood-free computation algorithms*, ABC use a stochastic simulation approach to avoid evaluating the likelihood function explicitly.

Let $\hat{y} = f(\theta, x) \in \mathcal{O} \subset \mathbb{R}^l$ be the predicted outcome from $p(\hat{y}|\theta, \mathcal{M})$, which is the forward model class $\mathcal{M}$ with parameters $\theta \in \Theta$, and $\mathcal{D}(x, y)$ a data set where $x \in \mathcal{X}$ are the inputs and $y \in \mathcal{Y}$ the observed outputs. Then, Eq. (2) can be adapted when applied to the pair $(\theta, \hat{y}) \in \Theta \times \mathcal{O} \subset \mathbb{R}^{d+l}$ as follows:

$$p(\theta, \hat{y}|\mathcal{D}) \propto p(\mathcal{D}|\hat{y}, \theta)\, p(\hat{y}|\theta)\, p(\theta) \tag{3}$$

where the conditioning to the model class $\mathcal{M}$ has been omitted for clarity since the method is valid for any $\mathcal{M}$.

From the last equation, it is clear that when the likelihood function $p(\mathcal{D}|\hat{y}, \theta)$ is intractable or directly unknown, the posterior $p(\theta, \hat{y}|\mathcal{D})$ cannot be obtained. The ABC methods provide us with an efficient alternative, bypassing the evaluation of the likelihood function using an approximated simulation based approach (Santoso et al., 2011). Indeed, through the use of a tolerance parameter ϵ and a user-defined metric function ρ, the method selects as posterior samples the pairs $(\theta, \hat{y}) \in \mathcal{S} \subseteq \Theta \times \mathcal{O}$ which satisfy that $\hat{y} \sim p(\hat{y}|\theta)$ lay within a specified region around the data y given by $\mathcal{B}_\epsilon(y) = \{\hat{y} \in \mathcal{O} : \rho(\eta(\hat{y}), \eta(y)) \leqslant \epsilon\}$, where the metric function $\rho(\cdot)$ evaluates the closeness between $\hat{y}$ and y using a vector of summary statistics $\eta(\cdot)$ (Fearnhead and Prangle, 2012) which, if required, allows the comparison between both vectors in a weak manner. Thus, under the ABC perspective, Eq. (3) can be rewritten as (Chiachio et al., 2014):

$$p_\epsilon(\theta, \hat{y}|\mathcal{D}) \propto P\left(\hat{y} \in \mathcal{B}_\epsilon(y)|\theta\right) p(\hat{y}|\theta) p(\theta) \tag{4}$$

where $P\left(\hat{y} \in \mathcal{B}_\epsilon(y)|\theta\right)$ is the approximated likelihood function which takes the unity when $\rho(\eta(\hat{y}), \eta(y)) \leqslant \epsilon$, and 0 otherwise. In the last equation, $P(\cdot)$ denotes probability and $p(\cdot)$ a PDF. By this means, the ABC marginal posterior of the parameters can be straightforwardly obtained as:

$$p_\epsilon(\theta|\mathcal{D}) \propto P\left(\hat{y} \in \mathcal{B}_\epsilon(y)|\theta\right) p(\theta) \tag{5}$$

Observe that this basic form of the ABC method is conceived as a rejection algorithm which generates $(\theta, \hat{y}) \sim p(\hat{y}|\theta)\, p(\theta)$ and accepts them conditional on $\hat{y}$ being close to y under a tolerance value ϵ. It should be noted that ϵ is desired to be very small so predictions $\hat{y}$ are accurate, but this is at the expense of highly inefficient computations. Also, if $\epsilon \to 0$ then necessarily $\eta(\hat{y}) \simeq \eta(y)$, which is unlikely under a probabilistic forward model $p(\hat{y}|\theta)$. On the contrary, choosing a high value for ϵ would make the approximate posterior $p_\epsilon(\theta|\mathcal{D})$ very similar to the prior $p(\theta)$, given that the majority of samples drawn from the prior would be accepted.

As can be seen, choosing the tolerance parameter ϵ entails a trade-off between accuracy of the posterior approximation and computational cost. In the literature, a branch techniques have been proposed to address this trade-off by combining the ABC principles with sampling methods like Markov Chain Monte Carlo (Marjoram et al., 2003), Parallel Tempering (Baragatti et al., 2013), or Population Monte Carlo (Beaumont et al., 2009). While some of them have demonstrated efficiency, using $\epsilon \to 0$ still translates into heavy computation. Thus, other techniques that use a decreasing sequence of tolerance levels ϵ have emerged, which achieve improved computational performance for low ϵ values (Del Moral et al., 2012). Among those, the so-called Approximate Bayesian Computation by Subset Simulation algorithm, namely ABC-SubSim (ABC-SS) algorithm (Chiachio et al., 2014), has proved to be one of the most efficient ABC algorithms in the literature, having been included in several well-known ABC user-platforms like ABCpy (Dutta et al., 2017) and Pi4U (Hadjidoukas, 2021).

ABC-SS exploits the ABC principles with the Subset Simulation method (Au and Beck, 2001) which transforms a rare event simulation problem into a sequence of simulations with larger probabilities, resulting in a reduction of the computational cost (Ching et al., 2005; Au et al., 2007). Indeed, in ABC-SS the region $\mathcal{S}$ containing the possible solutions under a tolerance ϵ is defined as the intersection of a sequence of nested regions $\mathcal{S}_j, j = 1, \ldots, \ell$ such that $\mathcal{S}_1 \supset \mathcal{S}_2 \ldots \supset \mathcal{S}_\ell = \mathcal{S}$, where:

$$\mathcal{S}_j = \{(\theta, \hat{y}) : \rho(\eta(\hat{y}), \eta(y)) \leqslant \epsilon_j\}, \text{ and } \epsilon_{j+1} < \epsilon_j \quad \forall j = 1, \ldots, j \tag{6}$$

Following this approach, the probability of a predicted outcome $\hat{y}$ from a Bayesian neural network to belong to a specified region $\mathcal{S}$, which is referred to as $P((\theta, \hat{y}) \in \mathcal{S})$ and denoted for simplicity as $P(\mathcal{S})$, can be defined as:

$$P(\mathcal{S}) = P(\mathcal{S}_1) \prod_{j=2}^{\ell} P\left(\mathcal{S}_j | \mathcal{S}_{j-1}\right) \tag{7}$$

where $P(\mathcal{S}_1)$ can be efficiently obtained using the Monte Carlo method, whilst the remaining factors $P\left(\mathcal{S}_j|\mathcal{S}_{j-1}\right)$, $j \geqslant 2$, can be estimated through samples by satisfying that $P\left(\mathcal{S}_j|\mathcal{S}_{j-1}\right) = P_0$, where P_0 is a conditional probability acting as a hyper-parameter defined by the modeller.

3. Training Bayesian neural networks by ABC-SubSim

In this section the ABC-SS algorithm is adapted to train BNN, thus the need for evaluating the gradient of the cost function is avoided. The forward model class $\mathcal{M}$ in Section 2.3 now represents the architecture of the BNN, and $\theta = \{w, b\}$. As specified in Section 1, gradient evaluation is a common task in most training algorithms of Bayesian neural networks where the parameters θ are updated using the gradient of the cost function $\nabla_\theta C(\theta)$. Instead, ABC-SS pursues obtaining the posterior distribution function of the parameters θ through statistical simulation, so that the most plausible values of θ, which better explain the observed data y, are obtained under a specified tolerance value ϵ chosen by the user. The capabilities of this training algorithm are illustrated in this section with two low complexity problems. These examples, with scaled architectures, allow us to graphically appreciate the learning process and the mechanisms of *BNN by ABC-SS* to capture both the aleatory and epistemic uncertainty.

3.1. Proposed methodology

The method starts by generating N random samples of parameters $\theta = \{w, b\}$ from the prior PDF $p(\theta)$ defined by the user, which are subsequently used to run a forward pass and obtain N outputs $\hat{y}(\theta)$. The resulting N simulated pairs $\{\theta, \hat{y}(\theta)\}$ form the preliminary subset $\mathcal{S}_0$ and they are distributed as $p\left((\theta, \hat{y})_0 | \mathcal{S}_0\right)$. At this stage, the metric $\rho(\eta(\hat{y}), \eta(y))$ is evaluated for each sample $\{\theta, \hat{y}(\theta)\} \in \mathcal{S}_0$ and an amount of NP_0 of those with the lowest metric value ρ are selected as *seeds* of the next subset $\mathcal{S}_1$. These seeds are distributed as $p\left((\theta, \hat{y})_1 | \mathcal{S}_1\right)$ and are used to: (1) automatically fix the tolerance value ϵ_1, as the highest metric value $\rho(\cdot)$ among the seeds; (2) obtain $(^1/_{P_0} - 1)$ new samples from each seed within the region $\mathcal{S}_1$, until the total population in $\mathcal{S}_1$

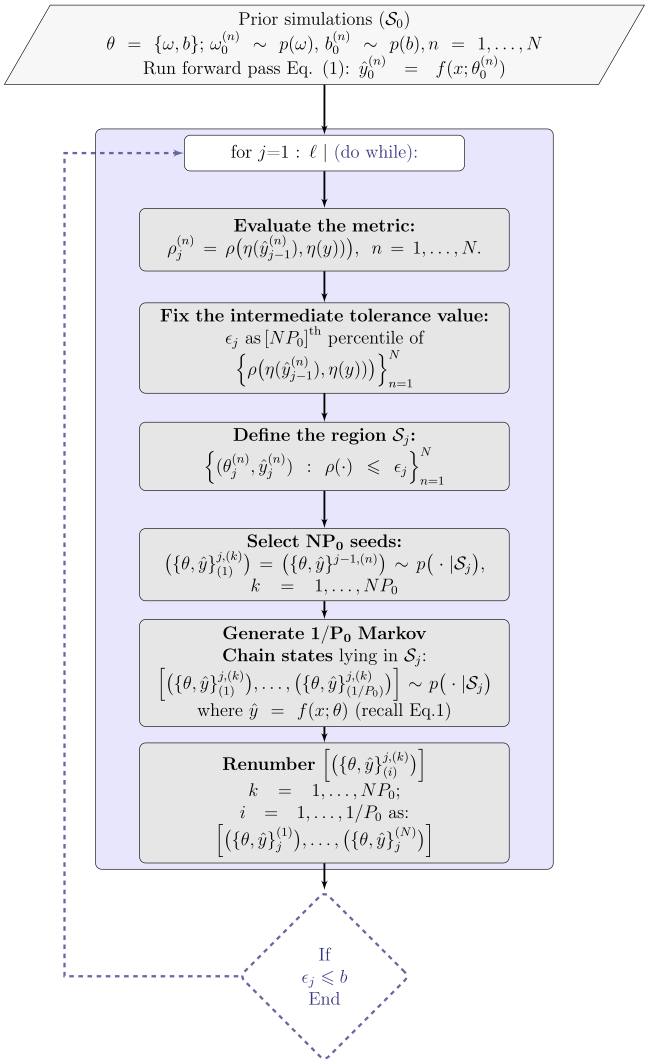

Prior simulations ($\mathcal{S}_0$)
$\theta = \{\omega, b\}; \omega_0^{(n)} \sim p(\omega), b_0^{(n)} \sim p(b), n = 1, \ldots, N$
Run forward pass Eq. (1): $\hat{y}_0^{(n)} = f(x; \theta_0^{(n)})$
for j=1 : ℓ | (do while):
Evaluate the metric:
$\rho_j^{(n)} = \rho(\eta(\hat{y}_{j-1}^{(n)}), \eta(y))$, $n = 1, \ldots, N$.
Fix the intermediate tolerance value:
ϵ_j as $[NP_0]^{\text{th}}$ percentile of
$\left\{\rho(\eta(\hat{y}_{j-1}^{(n)}), \eta(y))\right\}_{n=1}^{N}$
Define the region $\mathcal{S}_j$:
$\left\{(\theta_j^{(n)}, \hat{y}_j^{(n)}) : \rho(\cdot) \leqslant \epsilon_j\right\}_{n=1}^{N}$
Select NP$_0$ seeds:
$(\{\theta, \hat{y}\}_{(1)}^{j,(k)}) = (\{\theta, \hat{y}\}^{j-1,(n)}) \sim p(\cdot|\mathcal{S}_j)$,
$k = 1, \ldots, NP_0$
Generate 1/P$_0$ Markov Chain states lying in $\mathcal{S}_j$:
$\left[(\{\theta, \hat{y}\}_{(1)}^{j,(k)}), \ldots, (\{\theta, \hat{y}\}_{(1/P_0)}^{j,(k)})\right] \sim p(\cdot|\mathcal{S}_j)$
where $\hat{y} = f(x; \theta)$ (recall Eq.1)
Renumber $\left[(\{\theta, \hat{y}\}_{(i)}^{j,(k)})\right]$
$k = 1, \ldots, NP_0$;
$i = 1, \ldots, 1/P_0$ as:
$\left[(\{\theta, \hat{y}\}_j^{(1)}), \ldots, (\{\theta, \hat{y}\}_j^{(N)})\right]$
If
$\epsilon_j \leqslant b$
End

Fig. 2. Conceptual scheme of the main steps of the *BNN by ABC-SS* method. The steps corresponding to the while-loop option are depicted using dashed-blue line.

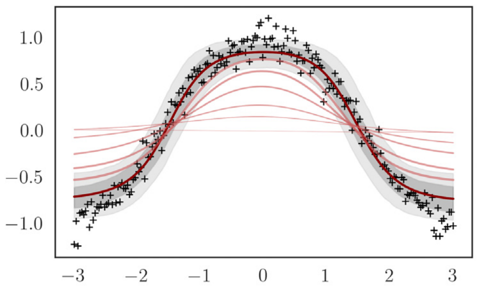
reaches N samples. The generation of samples is done by the Modified Metropolis Algorithm (MMA) (Au and Beck, 2001; Zuev et al., 2012), ensuring that the new samples generated from the seeds lie within $\mathcal{S}_1$, which is done by verifying that $\rho(\cdot) \leqslant \epsilon_1$. The method is repeated until the final subset $\mathcal{S}$, associated to the desired tolerance ϵ, is achieved, whereby the final approximate posterior $p((\theta, \hat{y})|\mathcal{S})$ is defined. Note that the final subset $\mathcal{S}$ constitute a set of N parameter configurations $\theta_{\mathcal{S}}^{(1)}, \theta_{\mathcal{S}}^{(2)}, \ldots, \theta_{\mathcal{S}}^{(n)}, \ldots, \theta_{\mathcal{S}}^{(N)}$, whose predicted outputs $\hat{y}(\theta_{\mathcal{S}}^{(n)})$ lie within a tolerance ϵ, under the metric $\rho(\cdot)$, given the data $\mathcal{D}$. The distribution of parameters in the final subset constitute the marginalized posterior $p_\epsilon(\theta|y)$ whose information is used to produce robust predictions and quantify their uncertainty. A conceptual scheme is provided in Fig. 2 to help understanding the *BNN by ABC-SS* method. Besides, a pseudo-code implementation of the *BNN by ABC-SS* method is provided in Algorithm 1.

Note that a slightly different version from the one provided in Algorithm 1 can be obtained by changing the *for* loop (step 16) by a *while* loop, so instead of specifying the number of simulations levels to be carried out, the algorithm performs as many simulations levels as needed to reach the desired tolerance value ϵ, which should be specified within the inputs to the algorithm. Also, to ease the reproducibility of the pseudo-code, it should be noted that a matrix M can be used to store the N sets of parameters and their corresponding metrics values $\rho(\cdot)$ for each j_{th} simulation level. This matrix is further updated throughout Algorithm 1 in steps 11–14, 22, 23, 32, 35 and 40. Thus, such matrix can be rearranged in ascending order of the metric ρ, step 17, and the *seeds* may be easily selected, step 22. Parameters w and b are extracted from the matrix M and appropriately rearranged to undertake a forward pass, step 29.

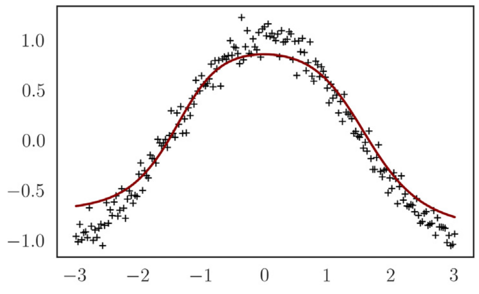
Finally, the authors remark the care needed for the selection of the algorithm hyper-parameters N, P_0 along with the standard deviation σ_j in the proposal PDF of the MMA at every region S_j. Recommendations for the selection of those values can be found in Chiachio et al. (2014), while some advances in the scaling of the Subset Simulation algorithm is covered in Zuev et al. (2012).

Algorithm 1 *BNN by ABC-SS*

```
1:  Inputs:
2:  FNN architecture and activation functions
3:  $P_0 \in [0,1]$
4:  $\ell$ {number of simulation levels}
5:  $N$ {samples per simulation level}
6:  $\rho()$ {metric/cost function, such as MSE}
7:  $\eta()$ {summary statistic, such as median}
8:  $\sigma_0 \leftarrow (\ell+1)0.1$
9:  Begin:
10: for $n : 1, ..., N$ do
11:     Sample initial parameters $\theta_0^{(n)}$, where $\theta_0$ includes all weights $w$ and bias $b$ of the BNN, from priors $p(w)$ and $p(b)$, such as $\mathcal{N}(0, I)$
12:     $\hat{y}_0^{(n)} \leftarrow$ Use Equation (1) to run a forward pass with parameters $\theta_0^{(n)}$ and input $x$ from D
13:     $\rho_0^{(n)} \leftarrow \rho(\eta(\hat{y}_0^{(n)}), \eta(y))$
14:     Set $M = \{\theta_0^{(n)}, \hat{y}_0^{(n)}, \rho_0^{(n)}\}_{n=1}^{N}$
15: end for
16: for $j : 1, ..., \ell$ do
17:     Renumber $[\theta_{j-1}^{(n)}, n : 1, \ldots, N]$ so that $\rho_{j-1}^{(1)} \leqslant \ldots \leqslant \rho_{j-1}^{(n)} \leqslant \ldots \leqslant \rho_{j-1}^{(N)}$
18:     $\epsilon_j \leftarrow \rho_{j-1}^{NP_0}$
19:     $\sigma_j \leftarrow \sigma_0 - 0.1\ell$ {proposed standard deviation decreases in each simulation level}
20:     $C \leftarrow 1$ {set counter to 1}
21:     for $i : 1, \ldots, NP_0$ do
22:         $\theta_j^{(i)} \leftarrow \theta_{j-1}^{(i)}$ {select seeds}
23:         $\rho_j^{(i)} \leftarrow \rho_{j-1}^{(i)}$
24:     end for
25:     for $k : 1, \ldots, NP_0$ do
26:         $\mu \leftarrow \theta_j^{(k)}$
27:         for $i : 1, \ldots, (^1/_{P_0}) - 1$ do
28:             $\theta^* \sim \mathcal{N}(\mu, \sigma_j)$
29:             $\hat{y}^* \leftarrow$ Use Equation (1) to run a forward pass with parameters $\theta^*$
30:             $\rho^* \leftarrow \rho(\eta(\hat{y}^*), \eta(y))$
31:             if $\rho^* \leqslant \epsilon_j$ then
32:                 $\theta_j^{(NP_0+C)} \leftarrow \theta^*$, and $\rho_j^{(NP_0+C)} \leftarrow \rho^*$
33:                 $\mu \leftarrow \theta^*$
34:             else
35:                 $\theta_j^{(NP_0+C)} \leftarrow \theta_j^k$, and $\rho_j^{(NP_0+C)} \leftarrow \rho_j^k$
36:             end if
37:             $C \leftarrow C+1$
38:         end for
39:     end for
40:     Update $M$ as $M = \{\theta_j^{(n)}, \hat{y}_j^{(n)}, \rho_j^{(n)}\}_{n=1}^{N}$
41: end for
```



(a) *BNN by ABC-SS*



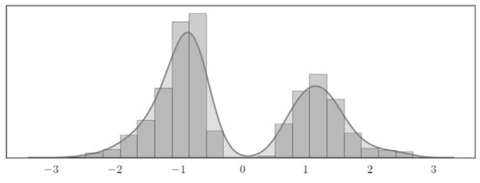
(b) FNN trained with Batch Gradient Descent

Fig. 3. *BNN by ABC-SS* (panel a) and Batch Gradient Descent (panel b), Illustrative Problem 1. Black crosses are training samples, dark red lines are median predictions, light red lines are intermediate levels median predictions, dark grey region is the interquantile range (IQR) of predictions, and the light grey region is the range between percentile 5 and 95 of predictions, also known as the uncertainty band.

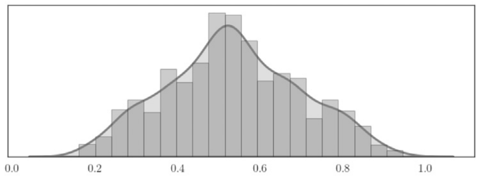
3.2. Illustrative problem 1

Training data for the first illustrative problem is generated from the cosenoidal function $y = \cos(x) + \zeta$, where $\zeta \sim \mathcal{N}(0, 0.1)$ simulates some noise in the observed data y. The domain of the training inputs x is uniformly distributed over the interval [−3,3]. The training data set comprises a single batch of 200 samples with no preprocessing. The architecture of the BNN consists of one input layer with one neuron, one hidden layer with two neurons and an output layer with one neuron. The activation function used for both the hidden and output layers is the hyperbolic tangent, as this function fits particularly well the training data. The hyper-parameters chosen are: $P_0 = 0.2$, $N = 5000$ and $\ell = 6$. Mean Squared Error (MSE) has been used as the cost function, or metric ρ in the ABC-SS language. The same training data has been used to fit a conventional FNN with the same architecture, and trained with a batch gradient descent algorithm (learning rate (lr) = 0.001 and $epochs$ = 10000), for reference purposes.

As can be seen in Fig. 3, both algorithms have obtained similar predictions, however, *BNN by ABC-SS* consistently reached similar outcomes while those from the FNN, trained with ordinary gradient descent, experience more variability between different runs of the algorithm. This suggests that *BBN by ABC-SS* is robust regardless the initialization of the parameters. In addition, *BNN by ABC-SS* provides an accurate quantification of the uncertainty in its predictions, representing the degree of belief in light of data. This output uncertainty is obtained by simulating the model considering the posterior distribution of the weight parameters learnt by the BNN, as shown in Fig. 4. Of special interest is Fig. 4(a), as it provides us with the posterior PDF of parameter $w_{1,1}^{(1)}$, without being constrained by any hypothesis about the family of functions it may belong to. Light red lines in Fig. 3(a) shows the learning process throughout the intermediate simulation levels.



(a) Posterior PDF of weight $w_{11}^{(1)}$.



(b) PDF of prediction $\hat{y}$ for x=-1.0.

Fig. 4. *BNN by ABC-SS*, illustration of the uncertainty about the trained parameters (panel a) and predictions (panel b), Illustrative Problem 1.

3.3. Illustrative problem 2

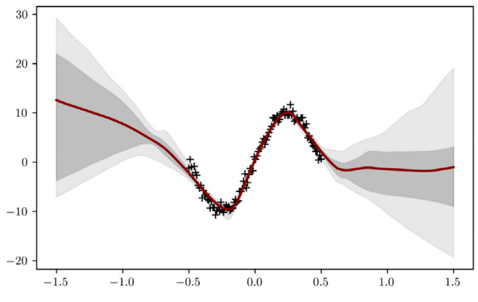
A second illustrative example is provided to show a more complex architecture, which is able to better capture the uncertainty in its parameters and predictions. In this case, the training data $\mathcal{D} = (x, y)$ is generated from the sinusoidal function $y = 10\ \sin(2\pi x) + \zeta$ with $\zeta \sim \mathcal{N}(0, 0.1)$. The training data set comprises 100 samples with $x \in [-0.5, 0.5]$. The proposed architecture consists of one input layer with one neuron, two hidden layers with 15 neurons each, and one output layer with one neuron, making a total of 286 parameters to be learned. A ReLU activation function is assigned to the neurons of the hidden layers, while a linear function, $f(x) = x$, is applied to the neuron in the output layer. Similarly to the first illustrative problem, the hyper-parameters chosen are: $P_0 = 0.1$, $N = 20000$ and $\ell = 8$. Again, the MSE function has been used as the cost function.

As shown in Figs. 5(a) and 6(b), the quantified uncertainty within the input domain of the training data is relatively small and proportional to the noise introduced by ζ, which can be classified as aleatory. However, as we exit the domain of the training data the epistemic uncertainty comes into play and grows as we move further away from the training data. This can be interpreted as a mechanism to express its lack of confidence when making predictions about regions of data it has not seen before. In contrast, a conventional neural network, Fig. 5(b), is equally confident both inside and outside the domain of the training data.

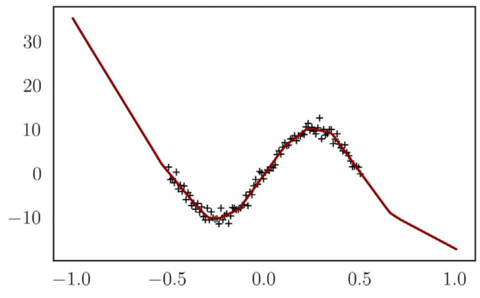
Regarding the posterior PDF of the parameters, Fig. 6(a) shows a more complex function, which varies between different runs of the *BNN by ABC-SS* algorithm. This confirms that, given a tolerance value ϵ, the number and diversity of valid sets of parameters $\theta = \{w, b\} \in \Theta \subseteq \mathbb{R}^d$, under tolerance ϵ, increases with the dimension d of the parameter space. Fig. 6(a) also shows a statistical correlation in the weights which is taken into account by the proposed algorithm when making predictions.

4. Engineering case study - fatigue damage in composite materials

The proposed *BNN by ABC-SS* has been applied to a real case study using experimental damage data from composite materials fracture experiments. The complex nature of this problem and the uncertainty about the potential damage modes constitute a significant challenge for physics-only based models, but at the same time create opportunities for data-driven models to demonstrate they can be a real alternative. The experimental framework is presented in this section, including a description of the composite materials case study in Section 4.1, the baseline algorithms used for comparison purposes in Section 4.2 and the metrics chosen to evaluate the performance of the algorithms in



(a) *BNN by ABC-SS*



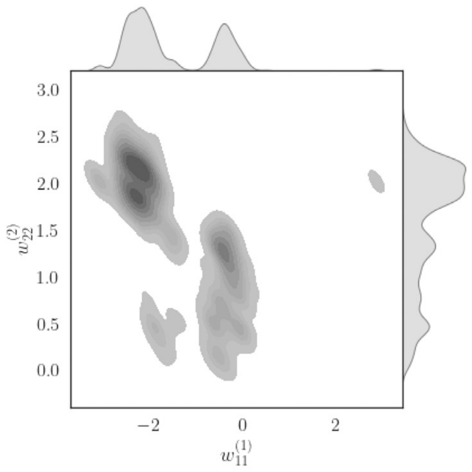
(b) FNN trained with Batch Gradient Descent

Fig. 5. BNNs by ABC-SS (panel a) and Batch Gradient Descent (panel b), Illustrative Problem 2. Black crosses are training samples, dark red lines are median predictions, dark grey region is the interquantile range (IQR) of predictions, and light grey region is the range between percentile 5 and 95 of predictions, also known as the uncertainty band.

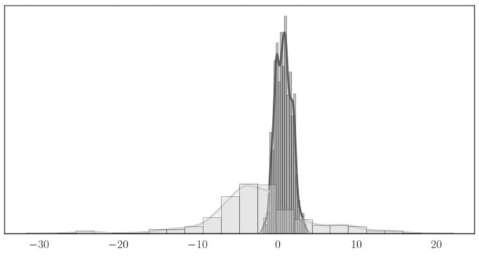
Section 4.3. The application of *BNN by ABC-SS* to the case study along with a discussion on the results and the potential implications to the engineering field and structural health monitoring systems is given in Section 4.4. The same data has been used to train another three BNN, namely Variational Inference (VI) or more specifically Bayes by Backprop (BBP), Probabilistic Backpropagation (PBP) and Hamiltonian Monte Carlo (HMC), and a comparison with *BNN by ABC-SS* has been carried in Section 4.5.

4.1. Description of the engineering case study - fatigue damage in composite materials

The performance of the *BNN by ABC-SS* is investigated using experimental data about fatigue damage in carbon fibre composite materials. These are high performance heterogeneous materials with very high strength-to-weight ratios extensively used in the aerospace and wind energy industries, among others. Damage in composites typically comprises several families of internal fractures (both *intralaminar* and *interlaminar* cracks (Talreja, 2008)) which result in changes in the macro-scale mechanical properties of the material. The temporal evolution and propagation of these damage modes is a complex and partially unknown process subject to much uncertainty (Chiachío et al., 2015). In this particular case study, the data consist of sequences of both intralaminar micro-cracks density and stiffness reduction measurements for three different laminates with the same cross-ply ($[0_2/90_4]_s$) layup. The data used are taken from the NASA Ames Prognostics Data Repository (CFRP Composites Dataset) (Saxena et al., 2008) and correspond to the laminates TD19, TD21 and TD22. This monitoring data were collected from a network of 12 piezoelectric (PZT) sensors using Lamb wave signals and three triaxial strain-gages (Larrosa Wilson and Chang, 2012). For this study the dataset is designated as $\mathcal{D}(x, y_1, y_2)$, which comprises loading cycles as inputs x and micro-cracks density and stiffness reduction as observed outputs y_1 and y_2, respectively. Thus, the *BNN by ABC-SS* method is used to predict two different outputs $\hat{y}_1$



(a) Posterior PDF of weights $w_{11}^{(1)}$ and $w_{22}^{(2)}$.



(b) PDF of prediction $\hat{y}$ for x=0 (within the domain of the training data) in dark grey, and for x=0.6 (outside the domain of the training data) in light grey.

Fig. 6. Illustration of the uncertainty about the trained parameters (panel a), and predictions made by the BNN trained with ABC-SS (panel b) both within and outside the domain of the training data set, Illustrative Problem 2. Figure (b) clearly shows greater uncertainty in those predictions made outside the domain of the training data.

and $\hat{y}_2$ from one single input x. It should be noted that some stiffness measurements are missing for the last loading cycles of the test so they have been synthetically generated to complete the data. Also, the training data set has been normalized to take values in the range $[0, 1]$. For the comparison exercise, the different BNN are asked to predict the micro-crack density ($\hat{y}_1$) given the loading cycles x as inputs.

4.2. Baseline algorithms

The baseline architecture used in the comparison exercise by the different algorithms comprises one input layer with one neuron (loading cycles), two hidden layers with 5 neurons each, and one output layer with one neuron (micro-cracks density). The activation functions are ReLU for the hidden layers and linear for the output layer. The rest of the hyperparameters have been chosen individually for each algorithm as follows:

- *BNN by ABC-SS*: A BNN trained with Algorithm 1, adapted with a *while* loop and $\sigma_j = \sigma_0 p$ as per Section 3. Two different architectures are used, the baseline architecture for the comparison in Section 4.5, and a modified version with two output neurons to test the performance of *BNN by ABC-SS* when providing heterogeneous outputs, micro-cracks density and stiffness reduction, in Section 4.4. The hyper-parameters chosen are $P_0 = 0.1$, $N =$ 100,000, $\sigma_0 = 0.75$, $p = 0.58$ and tolerance value $\epsilon = 0.012$ ($\epsilon =$ 0.007 is used in the comparison exercise).
- Variational Inference, Bayes by Backprop (BBP) (Blundell et al., 2015): A BNN with the baseline architecture, trained with an open source algorithm[1] implemented in Keras (Chollet et al., 2015). The hyperparameters have been chosen based on those found in the original code and slightly adjusted to suit the training data, including a scale mixture prior $P(\theta)$ of two Gaussian densities with $\sigma_1 = 1.5$, $\sigma_2 = 0.1$, $\pi = 0.5$, *Adam* optimizer (Kingma and Ba, 2015), $lr = 0.001$ and $epochs = 100,000$. The same BNN has also been trained with LeakyReLU (Maas et al., 2013) as the activation function in the hidden layers.
- Probabilistic Backpropagation (PBP) (Hernandez-Lobato and Adams, 2015): A BNN with the baseline architecture, trained with the open source algorithm[2] provided in Hernandez-Lobato and Adams (2015). The only hyperparameter to be adjusted is the number of epochs, which has been chosen based on the regression task found in the original code, $epochs =$ 30.
- Hamiltonian Monte Carlo (HMC) (Betancourt, 2017): A BNN with the baseline architecture, trained with *hamiltorch.*[3] The hyperparameters have been chosen based on those found in the regression task of the original code and Benker et al. (2020) as follows; step size $\epsilon = 0.001$, leapfrog steps $L = 10$, prior $p(\theta)$ a Gaussian with prior precision for the parameters $\tau = 1$, likelihood output precision $\tau_{out} = 100$ and 500 samples where 250 are burned (not included during inference or to evaluate the metric). The same BNN has also been trained with LeakyReLU as the activation function in the hidden layers.

4.3. Performance metric

The performance of *BNN by ABC-SS* is evaluated using the full loading cycle from the first sensor in TD19 as test data, and in two different ways. First, by its ability to simultaneously predict two heterogeneous output values (micro-crack density and stiffness reduction) from one single input (loading cycles), while quantifying the uncertainty in the predictions for each of the outputs individually. The mean squared error (MSE) is used as the metric, and the capacity to quantify the uncertainty is graphically assessed by its Inter Quantile Range (IQR). Second, a comparison with BBP, PBP and HMC is undertaken by running the different algorithms 50 times independently and calculating their MSE in each of the runs. The performance of the algorithms throughout the 50 runs, shown in Fig. 9, Fig. 10, Fig. 7(a) and Table 1, is expressed in the following terms: precision, measured by the median and the maximum and minimum MSE; variability, measured by the quartiles (Q1 and Q3), IQR and the lower/upper whiskers; stability, measured by the number of outliers; computation time (Intel® Core™ i7-10510U CPU @ 1.80 GHz (8 Threads) ∼2.3 GHz, 8 GB RAM); and the capacity of the algorithms to quantify the uncertainty, evaluated graphically by the ability of the uncertainty band to capture the variability observed in the data.

4.4. Application of *BNN by ABC-SS* to fatigue in composite materials

As shown in Fig. 7, the proposed BNN methodology has shown accuracy and efficiency in simultaneously representing the evolution of different damage features of CFRP composites while accounting for the uncertainty associated to the different outputs, namely micro-cracks density and stiffness reduction. This is a relevant practical result given the complexity to accurately reproduce the damage evolution in composites using physics-based models (Talreja, 2008). The presented modelling exercise is based on experimental damage data taken under laboratory-controlled conditions yet showing high variability. This translates into high modelling uncertainty that will significantly

[1] https://github.com/krasserm/bayesian-machine-learning - Variational Inference in Bayesian Neural Networks.
[2] https://github.com/HIPS/Probabilistic-Backpropagation.
[3] https://github.com/AdamCobb/hamiltorch

Table 1
Comparison between *BNN by ABC-SS*, Variational Inference (VI) with Bayes by Backprop, Hamiltonian Monte Carlo (HMC) and Probabilistic Backpropagation (PBP). Each of the algorithms have been run 50 times independently and the results, expressed in terms of MSE, are summarized in this table.

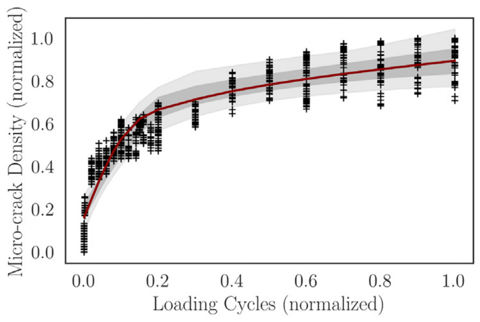
Statistics of MSE obtained in 50 independent runs of the training algorithm									
	Q1 (P_{25})	Median (P_{50})	Q3 (P_{75})	IQR (Q3-Q1)	Lower Whisker	Upper Whisker	Absolute Min-Max	Outliers	Comput. Time
BNN by ABC-SS	0.0057	0.0060	0.0062	**0.0005**	0.0052	0.0068	0.0052-**0.0068**	**0**	144s
VI ReLU	0.0307	0.0317	0.0394	0.0087	0.0302	0.0449	0.0114–0.1159	21	315s
VI LeakyReLU	0.0089	0.0162	0.0168	0.0079	0.0081	0.0179	0.0081–0.0309	2	324s
HMC ReLU	0.0054	0.0069	0.0158	0.0104	0.0046	0.0291	0.0046–0.1148	6	48s
HMC LeakyReLU	0.0052	0.0060	0.0076	0.0024	0.0048	0.0087	0.0048–0.0564	4	52s
PBP	**0.0049**	**0.0052**	**0.0055**	0.0006	**0.0041**	**0.0064**	**0.0041**-0.0187	3	**44s**

increase under real-life conditions, making the adoption of deterministic physics-based models unfeasible (Sriramula and Chryssanthopoulos, 2009). Failing to account this uncertainty results in high safety factors, which undermines the high utilization potential of composites in material-extensive industries such as civil engineering, among others. In fact, a BNN model such as the proposed here can be useful for on-board structural health monitoring systems where damage data is collected in a sequential manner and predictions with quantified uncertainty can be made during operation using incomplete data. If a large enough amount of data has been collected up to a particular time then predictions consistent with the future damage evolution can be made, as shown in Fig. 8. It should be noted that this predictive capability will entirely depend on whether the BNN has learnt enough from the collected data, which in turn will depend on whether or not the damage process has reached an almost stationary stage (as shown in the experimental dataset during the last 70% of the process).

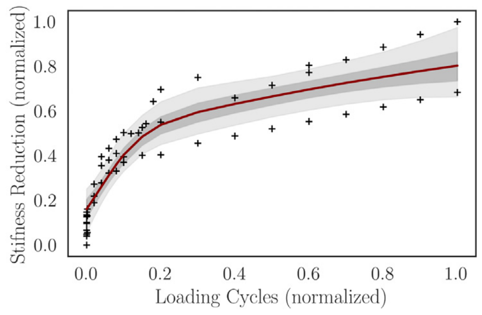
As a remark, nowadays physics-based models have proved efficiency and predictability only in low-scale structures and highly controlled environments. Furthermore, they are deterministic and do not consider the uncertainty inherent in fatigue damage of composite materials, even in laboratory conditions. The proposed data-driven method allows for the scalability to complex structures and environments, while the uncertainty in the observed data is quantified. Therefore, *BNN by ABC-SS* has the potential to create new opportunities for the application of prognosis and health management systems to real life scenarios.

4.5. Comparison with the state-of-the-art BNN

To further explore the potential and limitations of the proposed method for BNN inference, a comparative assessment is carried out here using the composites dataset presented in Section 4.1. In particular, the results were compared with those obtained using the Variational Inference (VI) method, more specifically Bayes by Backprop (BBP) with Keras (Chollet et al., 2015), Hamiltonian Monte carlo (HMC) and Probabilistic Backpropagation (PBP). The architecture selected for the chosen BNN has been presented in Section 4.2. Fig. 9 provides a box plot of the MSE obtained after training each BNN 50 times independently, and the numerical values are shown in Table 1. In terms of precision, PBP provides the most accurate predictions although closely followed by the proposed *BNN by ABC-SS* and HMC. However, *BNN by ABC-SS* has achieved the lowest IQR value, which translates into reliability thanks to the low variability of its predictions in different independent runs of the algorithm. *BNN by ABC-SS* has demonstrated high stability, which is measured by the number of outliers. It is presumed that such outliers, present in the other BNN, may be caused by the saturation of some neurons, meaning that their gradient falls to 0, or the so called *Dying ReLU* effect (Lu, 2020). When this phenomenon happens and the training algorithm is based on backpropagation, as is the case with VI(BBP), HMC and PBP, the learning process is affected and the weights stop updating. This may be overcome using different activation functions such as Leaky-ReLU, which, as observed in Fig. 9 and Table 1, outperforms ReLU in terms of MSE and the number of outliers. The capability of the different BNN to quantify the uncertainty, or the degree of belief on the predictions, is graphically assessed and



(a) Predictions of micro-cracks density.

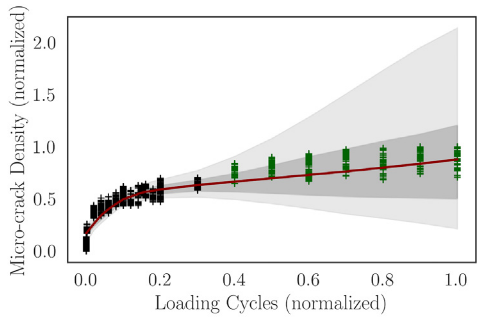


(b) Predictions of stiffness reduction.

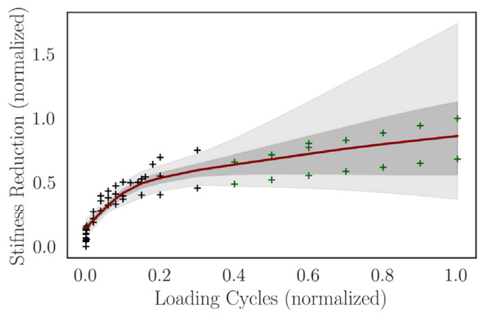
Fig. 7. Real Case Study, *BNN by ABC-SS* trained with data set from NASA. Black crosses are training samples, dark red lines are median predictions, dark grey region is the interquantile range (IQR) of predictions, and light grey region is the range between percentile 5 and 95 of predictions, also known as the uncertainty band.

illustrated in Figs. 10 and 7(a). There is a significant number of micro-crack density measurements (black crosses) with high variability for each loading cycle, which translates in high uncertainty in the training data. As it can be seen, most of the those training points fall within the uncertainty band of *BNN by ABC-SS*, resulting in a more flexible and realistic representation of the actual uncertainty inherent in the data. In terms of computation time, PBP and HMC have proved to be the fastest algorithms, in line with Hernandez-Lobato and Adams (2015). While *BNN by ABC-SS* seems to demand a longer computation time, it also needs to be noted that the proposed algorithm has been implemented in *Python* (Van Rossum and Drake Jr., 1995) without using optimized libraries, unlike HMC and PBP which are implemented in *Pytorch* (Paszke et al., 2019) and *Theano* (Theano Development Team, 2007) respectively. It is presumed that the computation time of *BNN by ABC-SS* may be improved by using libraries based on graphs like *Tensorflow* (Abadi et al., 2015), and possibly with parallel computation, remaining both options as a potential continuation of this research.

By looking at the results obtained, it could be concluded that *BNN by ABC-SS* provides accurate predictions, comparable to those from PBP and HMC, along with low variability and high stability in different runs of the algorithm, presumably due to its gradient-free



(a) Predictions of micro-cracks density.



(b) Predictions of stiffness reduction.

Fig. 8. Real Case Study with data set from NASA. *BNN by ABC-SS*. Black crosses are training samples, dark green crosses represent unseen data, dark red lines are median predictions, dark grey region is the interquantile range (IQR) of predictions, and light grey region is the range between percentile 5 and 95 of predictions, also known as the uncertainty band. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

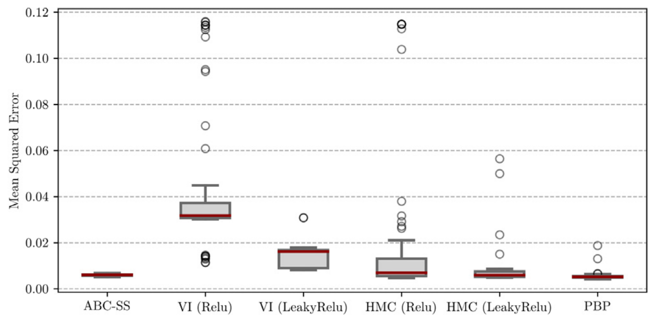
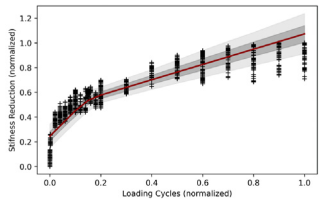


Fig. 9. Analysis of the MSE achieved in 50 independent simulations of *BNN by ABC-SS* (ReLU) and Variational Inference with Bayes by Backprop (ReLU and LeakyReLU). The MSE achieved with each neural network throughout the 50 simulations is represented by their minimum, first quartile, median, third quartile, maximum and outliers.

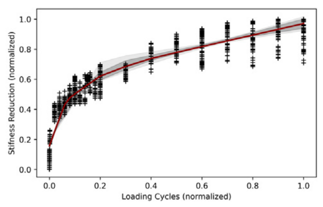
nature, which denotes reliability. And more importantly, *BNN by ABC-SS* complements its predictions with a fairer representation of the uncertainty, which provides valuable information for the subsequent decision making process. It is therefore this robustness and capability to accurately quantify the uncertainty that could make the proposed algorithm more suitable for the task in hand than other state-of-the-art BNN, given the high variability and uncertainty inherent in fatigue data from composite materials.

5. Discussion

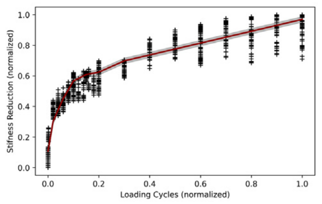
One of the key challenges of machine learning algorithms is to make predictions about unobserved data, based on a model learnt from a set of limited training data. This learning process has an inherent uncertainty, which needs to be quantified in order to evaluate the



(a) Variational Inference (Bayes by Backprop)



(b) Hamiltonian Monte Carlo



(c) Probabilistic Backpropagation

Fig. 10. Illustrative comparison between different BNN on uncertainty quantification. Black crosses are training samples, dark red lines are median predictions, dark grey region is the interquantile range (IQR) of predictions, and light grey region is the range between percentile 5 and 95 of predictions, also known as the uncertainty band. For Probabilistic Backpropagation the uncertainty is expressed as ±3 standard deviations from the mean, as per the original paper (Hernandez-Lobato and Adams, 2015). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

degree of belief on the model predictions. The proposed *BNN by ABC-SS* has demonstrated efficiency and flexibility in extracting uncertain (plausible) knowledge from data by inferring non-parametric posterior PDFs of the parameters $p(\theta|\mathcal{D})$ and likelihood function $P\left(\hat{y} \in \mathcal{B}_{\epsilon}(y)|\theta\right)$. This uncertainty quantified by the proposed algorithm, can be interpreted as aleatory when it comes from within the domain of the training data, and originates in the variability of the observed data. Outside that domain, the uncertainty is regarded as epistemic, given that in machine learning lack of data translates into lack of knowledge.

As opposed to other learning methods for BNN, these non-parametric posterior PDFs enable a much richer and flexible uncertainty quantification and therefore a better idealization of the reality given by the data, as shown in Figs. 7 and 10. *BNN by ABC-SS* allows data to "speak for themselves" with no restrictions, which translates into such increased flexibility without making the model more complex (Ghahramani, 2015). In view of Fig. 6(a), the adoption of a predefined probability model for the weights and bias (i.e. a Gaussian model) could well be defined as a rigid uncertainty quantification, leading to a constrained representation of the reality. Moreover, *BNN by ABC-SS* has consistently reached a similar outcome in each run of the algorithm regardless of the initialization of the parameters, which suggests robustness. It is presumed that such robustness is also due to the gradient-free nature of the algorithm, as explained in Section 4.5. The scalability analysis of *BNN by ABC-SS* to deal with high-dimensional parameters, in the order of thousands or millions, is out of the scope of this work, and constitutes the natural continuation of the present research.

6. Conclusions

Modern ANN provide us with very accurate predictions, however, when these are used in a decision-making context, the quantification of the prediction uncertainty gains importance. It forms the basis to define the degree of belief on those predictions and helps us to decide how we make use of them. Many state-of-the-art Bayesian training algorithms use rigid parametric PDFs for the likelihood function and/or the weights and bias, such as a Gaussian PDF defined by their mean and standard deviation, which limits their capacity to represent the uncertainty in the observed data. Moreover, they are often subject to the drawbacks of gradient descent and backpropagation.

A novel training method for Bayesian Neural Networks has been developed in this paper using the Approximate Bayesian Computation combined with Subset Simulation as inference engine. The resulting methodology, named here as *BNN by ABC-SS*, has been illustrated using two academic examples and applied to an engineering case study based on damage data in composite structures. The results have revealed that the non-parametric formulation of the likelihood function and the PDF of the weights provides a realistic uncertainty quantification according to the training data. Besides, through comparison with the VI method, HMC and PBP, *BNN by ABC-SS* showed more stability when making predictions, presumably due to absence of gradient. Particularly for the composite fatigue damage case study, the proposed data-driven methodology can be seen as an alternative to purely physics-based models which fail at quantifying the real amount of uncertainty of this process.

This new training algorithm could become specially useful when applied to problems where a decision is significantly dependent on the amount of uncertainty (Ghahramani, 2015). The scalability of the proposed method to train deep neural networks, with high-dimensional parameter spaces and large training data sets, could further extend the range of potential applications of this methodology, and establishes a natural continuation to this line of research.

CRediT authorship contribution statement

Juan Fernández: Methodology, Writing – original draft, Formal analysis, Software, Validation, Investigation, Visualization. **Manuel Chiachío:** Conceptualization, Methodology, Investigation, Resources, Writing – review & editing, Project administration, Funding acquisition. **Juan Chiachío:** Investigation, Resources, Writing - review & editing, Supervision, Funding acquisition. **Rafael Muñoz:** Writing – review & editing, Investigation, Resources. **Francisco Herrera:** Writing – review & editing, Supervision, Validation.

Declaration of competing interest

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

Acknowledgements

This paper is part of the ENHAnCE project, which has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 859957. Funding for open access charge: Universidad de Granada / CBUA. The authors would like to thank Dr. Charles Blundell at DeepMind for his selfless assistance and useful discussions; the Structures and Composites lab at Stanford University and NASA ARMD/AvSafe project SSAT for experimental composites data; Mr. Ramón Suárez Pérez for his IT support; and Mr. Luis Fernández Rios for his encouragement and support.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X., 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Altan, A., Karasu, S., Bekiros, S., 2019. Digital currency forecasting with chaotic meta-heuristic bio-inspired signal processing techniques. Chaos Solitons Fractals 126, 325–336.

Altan, A., Karasu, S., Zio, E., 2021. A new hybrid model for wind speed forecasting combining long short-term memory neural network, decomposition methods and grey wolf optimizer. Appl. Soft Comput. 100, 106996.

Arora, S., Singh, R., 2012. Automatic speech recognition: a review. Int. J. Comput. Appl. 60, 34–44.

Au, S.K., Beck, J.L., 2001. Estimation of small failure probabilities in high dimensions by subset simulation. Probab. Eng. Mech. 16 (4), 263–277.

Au, S., Ching, J., Beck, J., 2007. Application of subset simulation methods to reliability benchmark problems. Struct. Saf. 29 (3), 183–193.

Awad, W., Elseuofi, S., 2011. Machine learning methods for spam e-mail classification. Int. J. Comput. Sci. Inf. Technol. 3 (1), 173–184.

Baragatti, M., Grimaud, A., Pommeret, D., 2013. Likelihood-free parallel tempering. Stat. Comput. 23 (4), 535–549.

Bayes, T., 1763. An essay towards solving a problem in the doctrine of chances. Phil. Trans. R. Soc. London 53, 370–418.

Beaumont, M.A., Cornuet, J.-M., Marin, J.-M., Robert, C.P., 2009. Adaptive approximate Bayesian computation. Biometrika 96 (4), 983–990.

Beck, J.L., 2010. BayesIan system identification based on probability logic. Struct. Control Health Monit. 17 (7), 825–847.

Benker, M., Furtner, L., Semm, T., Zaeh, M.F., 2020. Utilizing uncertainty information in remaining useful life estimation via Bayesian neural networks and Hamiltonian Monte Carlo. J. Manuf. Syst. In Press.

Betancourt, M., 2017. A conceptual introduction to Hamiltonian Monte Carlo. arXiv preprint arXiv:1701.02434.

Blum, M.G., François, O., 2010. Non-linear regression models for approximate Bayesian computation. Stat. Comput. 20, 63–73.

Blum, M.G., Nunes, M., Prangle, D., Sisson, S., 2013. A comparative review of dimension reduction methods in approximate Bayesian computation. Statist. Sci. 28 (2), 189–208.

Blundell, C., Cornebise, J., Kavukcuoglu, K., Wierstra, D., 2015. Weight uncertainty in neural network. In: Bach, F., Blei, D. (Eds.), Proceedings of the 32nd International Conference on Machine Learning. In: Proceedings of Machine Learning Research, vol. 37, PMLR, Lille, France, pp. 1613–1622.

Buntine, W.L., Weigend, A.S., 1991. BayesIan back-propagation. Complex Syst. 5, 603–643.

Carlos-Roca, L.R., Torres, I.H., Tena, C.F., 2018. Facial recognition application for border control. In: 2018 International Joint Conference on Neural Networks. IJCNN. pp. 1–7.

Chiachio, M., Beck, J.L., Chiachio, J., Rus, G., 2014. Approximate Bayesian computation by subset simulation. SIAM J. Sci. Comput. (3), A1339—A1358.

Chiachío, J., Chiachío, M., Saxena, A., Sankararaman, S., Rus, G., Goebel, K., 2015. BayesIan model selection and parameter estimation for fatigue damage progression models in composites. Int. J. Fatigue 70, 361–373.

Ching, J., Au, S., Beck, J., 2005. Reliability estimation for dynamical systems subject to stochastic excitation using subset simulation with splitting. Comput. Methods Appl. Mech. Engrg. 194 (12), 1557–1579.

Chollet, F., et al., 2015. Keras. https://keras.io. (Accessed 6 Mar 2021).

Cox, R.T., 1946. Probability, frequency, and reasonable expectation. Amer. J. Phys. 14 (2), 1–13.

Das, V., Pollack, A., Wollner, U., Mukerji, T., 2019. Convolutional neural network for seismic impedance inversion. Geophysics 84 (6), R869–R880.

Del Moral, P., Doucet, A., Jasra, A., 2012. An adaptive sequential Monte Carlo method for approximate Bayesian computation. Stat. Comput. 22 (5), 1009–1020.

Depeweg, S., Hernandez-Lobato, J.-M., Doshi-Velez, F., Udluft, S., 2018. Decomposition of uncertainty in Bayesian deep learning for efficient and risk-sensitive learning. In: International Conference on Machine Learning. PMLR, pp. 1184–1193.

Dutta, R., Schoengens, M., Pacchiardi, L., Ummadisingu, A., Widmer, N., Onnela, J.-P., Mira, A., 2017. ABCpy: A high-performance computing perspective to approximate Bayesian computation. arXiv preprint arXiv:1711.04694.

Fearnhead, P., Prangle, D., 2012. Constructing summary statistics for approximate Bayesian computation: Semi-automatic approximate Bayesian computation. J. R. Stat. Soc. Ser. B Stat. Methodol. 74 (3), 419–474.

Ghahramani, Z., 2015. Probabilistic machine learning and artificial intelligence. Nature 521 (7553), 452–459.

Gilks, W., Richardson, S., Spiegelhalter, D., 1996. Markov Chain Monte Carlo in Practice. Chapman and Hall/CRC, Boca Raton.

Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. MIT Press.

Goodfellow, I., Warde-Farley, D., Mirza, M., Courville, A., Bengio, Y., 2013. Maxout networks. In: Dasgupta, S., McAllester, D. (Eds.), Proceedings of the 30th International Conference on Machine Learning, Vol. 28. (3), PMLR, Atlanta, Georgia, USA, pp. 1319–1327.

Graves, A., 2011. Practical variational inference for neural networks. In: Proceedings of the 24th International Conference on Neural Information Processing Systems. NIPS'11, Curran Associates Inc., Red Hook, NY, USA, pp. 2348–2356.

Grazian, C., Fan, Y., 2020. A review of approximate Bayesian computation methods via density estimation: Inference for simulator-models. WIREs Comput. Stat. 12 (4), e1486.

Hadjidoukas, P., 2021. URL https://github.com/cselab/pi4u/tree/master/inference. (Accessed Feb 25, 2021).

Hernandez-Lobato, J.M., Adams, R., 2015. Probabilistic backpropagation for scalable learning of Bayesian neural networks. In: Bach, F., Blei, D. (Eds.), Proceedings of the 32nd International Conference on Machine Learning. In: Proceedings of Machine Learning Research, vol. 37, PMLR, Lille, France, pp. 1861–1869.

Hoffman, M.D., Blei, D.M., Wang, C., Paisley, J., 2013. Stochastic variational inference. J. Mach. Learn. Res. 14 (5).

Hüllermeier, E., Waegeman, W., 2021. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. Mach. Learn. 110 (3), 457–506.

Jeffreys, H., 1961. Theory of Probability, third ed. Oxford, Oxford, England.

Jia, S., Yue, Y., Yang, Z., Pei, X., Wang, Y., 2020. Travelling modes recognition via Bayes neural network with Bayes by backprop algorithm. In: CICTP 2020. pp. 3994–4004.

Jiang, B., Wu, T.-Y., Zheng, C., Wong, W.H., 2017. Learning summary statistic for approximate Bayesian computation via deep neural network. Statist. Sinica 27 (4), 1595–1618.

Kingma, D.P., Ba, J., 2015. Adam: A method for stochastic optimization. In: Bengio, Y., LeCun, Y. (Eds.), 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.

Lampinen, J., Vehtari, A., 2001. BayesIan approach for neural networks—review and case studies. Neural Netw. 14 (3), 257–274.

Laplace, P., 1812. Théorie Analytique Des Probabilités. Courcier, Paris.

Larrosa Wilson, C., Chang, F.-K., 2012. Real time in-situ damage classification, quantification and diagnosis for composite structures. In: 19th International Congress on Sound and Vibration 2012, Vol. 4. ICSV 2012. pp. 2696–2704.

Levy, D., Sohl-dickstein, J., Hoffman, M., 2018. Generalizing Hamiltonian Monte Carlo with neural networks. In: ICLR 2018 Conference.

List, F., Lewis, G.F., 2020. A unified framework for 21 cm tomography sample generation and parameter inference with progressively growing GANs. Mon. Not. R. Astron. Soc. 493 (4), 5913–5927.

Lu, L., 2020. Dying ReLU and initialization: Theory and numerical examples. Commun. Comput. Phys. 28 (5), 1671–1706.

Maas, A.L., Hannun, A.Y., Ng, A.Y., 2013. Rectifier nonlinearities improve neural network acoustic models. In: ICML Workshop on Deep Learning for Audio, Speech and Language Processing.

MacKay, D.J., 1992. A practical Bayesian framework for backpropagation networks. Neural Comput. 4 (3), 448–472.

Marin, J.M., Pudlo, P., Robert, C.P., Ryder, R., 2012. Approximate Bayesian computational methods. Stat. Comput. 1167—1180.

Marjoram, P., Molitor, J., Plagnol, V., Tavaré, S., 2003. Markov chain Monte Carlo without likelihoods. Proc. Natl. Acad. Sci. 100 (26), 15324–15328.

Mcculloch, W., Pitts, W., 1943. A logical calculus of ideas immanent in nervous activity. Bull. Math. Biophys. 5, 127–147.

Neal, R., 1992. Bayesian Training of Backpropagation Networks by the Hybrid Monte Carlo Method. Tech. Rep., (CRG-TR-92-1), Department of Computer Science, U. of Toronto.

Neal, R.M., 1993. Probabilistic Inference Using Markov Chain Monte Carlo Methods. Tech. Rep., (CRG-TR-93-1), Department of computer science. University of Toronto.

Neal, R.M., 1996. Bayesian Learning for Neural Networks. Springer-Verlag, Berlin, Heidelberg.

Pascanu, R., Mikolov, T., Bengio, Y., 2013. On the difficulty of training recurrent neural networks. In: International Conference on Machine Learning. PMLR, pp. 1310–1318.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S., 2019. Pytorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems 32. Curran Associates, Inc., pp. 8024–8035.

Prangle, D., Everitt, R.G., Kypraios, T., 2018. A rare event approach to high-dimensional approximate Bayesian computation. Stat. Comput. 28 (4), 819–834.

Radev, S.T., Mertens, U.K., Voss, A., Köthe, U., 2020. Towards end-to-end likelihood-free inference with convolutional neural networks. Br. J. Math. Stat. Psychol. 73 (1), 23–43.

Rumelhart, D.E., Hinton, G.E., Williams, R.J., 1986. Learning representations by back-propagating errors. Nature 323, 533–536.

Sadgali, I., Sael, N., Benabbou, F., 2019. Performance of machine learning techniques in the detection of financial frauds. Procedia Comput. Sci. 148, 45–54.

Santoso, A., Phoon, K., Quek, S., 2011. Modified Metropolis–Hastings algorithm with reduced chain correlation for efficient subset simulation. Probab. Eng. Mech. 26 (2), 331–341.

Saxena, A., Goebel, K., Larrosa, C., Chank, F.-K., 2008. CFRP Composites Data Set, NASA Ames Prognostics Data Repository. NASA Ames Research Center, Moffett Field, CA. URL https://ti.arc.nasa.gov/project/prognostic-data-repository.

Shin, S., Lee, Y., Kim, M., Park, J., Lee, S., Min, K., 2020. Deep neural network model with Bayesian hyperparameter optimization for prediction of NOx at transient conditions in a diesel engine. Eng. Appl. Artif. Intell. 94, 103761.

Sriramula, S., Chryssanthopoulos, M.K., 2009. Quantification of uncertainty modelling in stochastic analysis of FRP composites. Composites A 40 (11), 1673–1684.

Sturman, O., von Ziegler, L., Schläppi, C., Akyol, F., Privitera, M., Slominski, D., Grimm, C., Thieren, L., Zerbi, V., Grewe, B., et al., 2020. Deep learning-based behavioral analysis reaches human accuracy and is capable of outperforming commercial solutions. Neuropsychopharmacology 45 (11), 1942–1952.

Talreja, R., 2008. Damage and fatigue in composites–a personal account. Compos. Sci. Technol. 68 (13), 2585–2591.

Theano Development Team, 2007. Theano: A python framework for fast computation of mathematical expressions.

Van Rossum, G., Drake Jr., F.L., 1995. Python Reference Manual. Centrum voor Wiskunde en Informatica Amsterdam.

Varshney, K.R., Alemzadeh, H., 2017. On the safety of machine learning: Cyber-physical systems, decision sciences, and data products. Big Data 5 (3), 246–255.

Vong, C.-M., Wong, P.-K., Li, Y.-P., 2006. Prediction of automotive engine power and torque using least squares support vector machines and Bayesian inference. Eng. Appl. Artif. Intell. 19 (3), 277–287.

Voulodimos, A., Doulamis, N., Doulamis, A., Protopapadakis, E., Andina, D., 2018. Deep learning for computer vision: a brief review. Comput. Intell. Neurosci. 2018, 7068349.

Wang, H., Bai, X., Tan, J., 2020. Uncertainty quantification of bearing remaining useful life based on convolutional neural network. In: 2020 IEEE Symposium Series on Computational Intelligence. SSCI. pp. 2893–2900.

Zhang, X., Mahadevan, S., 2020. BayesIan neural networks for flight trajectory prediction and safety assessment. Decis. Support Syst. 131, 113246.

Zuev, K.M., Beck, J.L., Au, S.-K., Katafygiotis, L.S., 2012. BayesIan post-processor and other enhancements of Subset Simulation for estimating failure probabilities in high dimensions. Comput. Struct. 92–93, 283–296.