Reliability Engineering and System Safety 243 (2024) 109822

Contents lists available at ScienceDirect

Reliability Engineering and System Safety

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

Physics-guided recurrent neural network trained with approximate Bayesian computation: A case study on structural response prognostics

Juan Fernández [a,*], Juan Chiachío [a], José Barros [b], Manuel Chiachío [a], Chetan S. Kulkarni [c]

[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] Faculty of Engineering, Catholic University of Santiago de Guayaquil, Guayaquil, Ecuador
[c] KBR, Inc., NASA Ames Research Center, Moffett Field, 94035, CA, United States

ARTICLE INFO

Keywords:
Physics-guided recurrent neural networks
Approximate Bayesian computation
Uncertainty quantification
Structural health monitoring
Forecasting

ABSTRACT

A new physics-guided Bayesian recurrent neural network is proposed in this manuscript. This hybrid algorithm benefits from the knowledge in physics-based models, the capability of recurrent neural networks to handle sequential data, and the flexibility of Bayesian methods to quantify the uncertainty. The introduction of physics in the forward pass of the neural network significantly improves the results in multistep-ahead forecasting, and the gradient-free nature of the Bayesian learning engine provides great flexibility to adapt to the observed data. The proposed algorithm has been applied to a data-driven problem about fatigue in composites, and a case study about accelerations in concrete buildings, where a comparison against the state-of-the-art algorithms is also provided. The results revealed: (1) the accuracy of the proposals, comparable to the state-of-the-art recurrent neural networks; (2) its stability during multiple runs of the algorithm, proving that it is a more reliable option; (3) its precise quantification of the uncertainty, which provides useful information for the subsequent decision-making process. As potential future applications to real world scenarios, the proposed Bayesian recurrent neural network could be used in on-board PHM systems in the aerospace industry, or as an on-site prediction tool in buildings for seismic events and/or aftershocks.

1. Introduction

When the records of a data set are stored consecutively, and there exists a dependence between them, this data set is considered to be sequential. Time-series data are a good example, such as gene sequences, weather data, heart rate or stock prices. But they are also found in engineering problems, like failure rate prediction of water distribution networks [1] or remaining useful life (RUL) predictions [2], and they play a fundamental part in reliability engineering, system health monitoring (SHM) and prognostics. Although these kind of data can be processed by different well-known methods, like Autoregressive Integrated Moving Average [3] or Exponential Smoothing [4] models, artificial neural networks (ANN) [5] have increased in popularity for the last two decades. More specifically, recurrent neural networks (RNN) have provided an outstanding performance when applied to time-series data. There are many different variants of RNN, from Long Short Term Memory (LSTM) [6] to Gated Recurrent Unit (GRU) [7,8], and they are responsible for many tools and software we use in our daily lives. To name a few, speech recognition [9], machine translation [10], image captioning [11], sentiment analysis [12], music generation [13] or video activity recognition [14], were made possible thanks to RNN. The engineering industry has also become increasingly interested in the use of RNN [15–17], even in construction related tasks [18]. Their success in so many different fields of application mainly lies in their capacity to store useful information from the past, and then use it to make predictions about the future.

Nevertheless, RNN also present some issues, mostly related to the use of the backpropagation algorithm to train the weights of the neural network, namely *vanishing* and *exploding gradients* [19]. While these problems are not exclusive of RNN, the fact that the gradient of the loss function is backpropagated through many time-steps makes this type of neural network more prone to suffer from them [19]. Different solutions have been proposed but the gated algorithms, such as LSTM and GRU, have provided the best results [20]. Another usual drawback of neural networks is their poor performance when data is scarce and/or imbalanced [21], which is a common situation in reliability engineering problems, and their inability to extrapolate [22]. Both issues have been addressed by hybrid methods, which introduce physics-based models into data-driven algorithms. Furthermore, we

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

https://doi.org/10.1016/j.ress.2023.109822
Received 18 February 2023; Received in revised form 17 October 2023; Accepted 14 November 2023
Available online 17 November 2023

can find in the literature different RNN that have been combined with physics to create hybrid RNN. Depending on the nature of the laws of physics, these are inserted in the RNN in a different manner. For example, Zhonghai Ma et al. [23] proposed using the physics-of-failure information from an Electro-Hydrostatic Actuator (EHA) system to construct a LSTM neural network, thus enhancing the prediction capability of the model. Zhanhang Li et al. [24] presented a new hybrid method for degradation prediction, by combining bidirectional LSTM neural networks with degradation tendency information from the physics-based model. Abhinav Subramanian et al. [25] developed a physics-informed machine learning approach for response prediction in dynamic systems, where the probabilistic machine learning model is recursively trained to predict the discrepancy between the output of the physics-based model and the observed data. Primarily based on the works of Raissi et al. [26], some authors have successfully applied physics-informed RNN to problems where the physics are governed by partial differential equations [27–31]. In these cases the neural network is forced to minimize a loss function that includes the physics-based differential equations and boundary conditions. RG Nascimiento et al. [32,33] developed a recurrent cell that combined physics-based and data-driven models for fleet prognosis and cumulative damage modeling. Physics-guided recurrent neural networks are another family of hybrid algorithms that are increasing in popularity, where the physics are often introduced in the loss function. Thus, the neural network needs to balance the deviation of its outputs from both, the observed data and the physics-based model. In this line of research, Xiaowei Jia et al. [34,35] successfully combined RNN and physics-based models for simulating temperature profiles in lakes, and predicting flow and temperature in river networks. Bayesian methods, such as Monte Carlo Dropout (MC Dropout), have also been combined with physics-guided RNN for quantification of the uncertainty, as can be seen in the works of Arka Daw [36]. Despite the promising results provided by those hybrid RNN, they are still subject to the drawbacks of backpropagation and the evaluation of the gradient of a predefined loss function [37].

In this manuscript, a new gradient-free training method for RNN based on Approximate Bayesian Computation with Subset Simulation (ABC-SS) is presented, hereafter called BRNN by ABC-SS. Also, its physics-guided version (PG-BRNN by ABC-SS) is proposed and applied to an engineering case study about accelerations in concrete buildings during an experimental seismic event. The performance of these Bayesian algorithms is evaluated in two experiments and compared against the state-of-the-art RNN, including MC Dropout. The results showed that the proposed Bayesian RNN achieve comparable accuracy to the state-of-the-art RNN but with a realistic quantification of the uncertainty, which may be critical in reliability engineering problems. Furthermore, the absence of gradient evaluation and loss function, coupled with the non-parametric formulation of the weights, explain the stability and reliability of the proposed algorithms. Finally, the introduction of physics-based models in PG-BRNN by ABC-SS significantly improves the extrapolation capabilities of the neural network, which may be key when making operational decisions, in predictive maintenance tools, and in prognostics and health management (PHM) systems.

The remainder of this manuscript is structured as follows. Section 2 provides a brief theoretical background about the main principles that form the foundations of the proposed algorithms. Section 3 describes how RNN can be trained with ABC-SS, and how physics-based models are introduced in the forward pass of the neural network. The experimental framework, comprising a data-driven problem and an engineering case study, is presented in Section 4. A discussion about the results obtained in both experiments is given in Section 5. Finally, the conclusions are provided in Section 6.

2. Background

This section provides an overview of the methodologies and principles that form the foundations of the proposals presented in this manuscript. The fundamentals of RNN and their applications are briefly described, followed by how ABC-SS can be used to train the weights of ANN, and finally, the benefits of combining physics-based models with ANN are outlined.

2.1. Recurrent neural networks

The concept of RNN was born in the 1980s [38–40], motivated by the need of ANN to handle sequential data. Indeed, RNN have tackled two major challenges inherent in such type of data: input information of varying lengths, and the fact that previous inputs may influence future inputs and outputs. All that is possible thanks to 'parameter sharing', where the same weights are recursively applied through each time step of the neural network. This way, information about the immediate past is stored and added to the present, to then make predictions about the future. This principle is illustrated in Fig. 1, and the formulation of the forward pass for the most basic form of RNN is shown in Eq. (1) (note that $tanh$ and $sigmoid$ are interchangeable by any other activation function, and b and c are bias parameters).

$$\begin{aligned} a_t &= b + W h_{t-1} + U x_t \\ h_t &= tanh(a_t) \\ o_t &= c + V h_t \\ \hat{y}_t &= sigmoid(o_t) \end{aligned} \tag{1}$$

Many different types of RNN can be found in the literature. Depending on the input–output relationship present in the data, there exist several design patterns: *one to one*, *one to many*, *many to one* or *many to many* [41]. Sentiment analysis or text classification are good examples of *many to one* patterns, just as machine translation is of *many to many*. The number of variables will also determine if the model is uni-variate or multi-variate. Furthermore, the original concept of RNN has evolved into more sophisticated and complex algorithms, such as Bidirectional RNN [42], GRU or LSTM [6,7]. The interested reader is referred to [41] (Chapter 10) for further details about the implementation of the different versions of RNN and their characteristics.

Despite the great success achieved by RNN in various applications, they are not without drawbacks. They mostly rely on the backpropagation algorithm [39], which becomes more complex in RNN and is the source of issues such as *vanishing and exploding gradients*. Also, RNN do not perform well when making predictions outside the domain of the training data, just like any other ANN. In fact, this problem is aggravated in univariate multistep-ahead forecasting, where the predicted value of the current time step is used to determine the value of the next time step, recursively. Interestingly, this situation is common in engineering, and more specifically, in SHM and prognostics and health management (PHM) [43], where the updated information about the health state of the structure is recursively used to prognosticate the future health states of the system. While those issues can be mitigated in varying degrees, like using LSTM for gradient-related problems, Sections 2.2 and 2.3 will present a different approach to avoid such drawbacks.

2.2. ABC-SS as a learning engine

Bayesian methods are a family of uncertainty quantification and probabilistic approaches, which are used to update our degree of belief about the value of uncertain parameters in a model, in light of new data and in a systematic manner. All these methods are based on the Bayes' theorem [44–46], shown in Eq. (2) below:

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

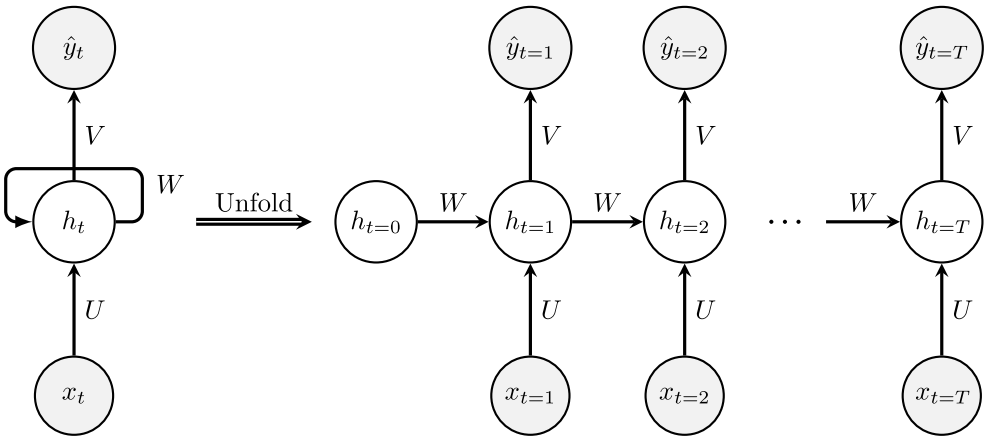


Fig. 1. Schematic representation of the folded and unfolded RNN.

where $\mathcal{M}$ refers to the model class (e.g. the neural network architecture), $\theta = \{W, U, V, b, c\} \in \Theta \subseteq \mathbb{R}^d$ are the uncertain parameters of the model, and $\mathcal{D}(x, y)$ is the observed data. The prior degree of belief about the model parameters is given by $p(\theta|\mathcal{M})$, and the updated/posterior information about those parameters is given by the probability density function (PDF) $p(\theta|\mathcal{D}, \mathcal{M})$. The likelihood function, which provides the probability of the observed data to be reproduced by the model $\mathcal{M}$ (specified by θ) is represented by $p(\mathcal{D}|\theta, \mathcal{M})$. The evidence, or probability of the overall model class to represent the observed data, is given by $p(\mathcal{D}|\mathcal{M})$.

Bayesian methods have become a popular inference engine in science and engineering, which also quantifies the uncertainty about the inferred parameters based on our prior knowledge and the information available. However, in most real cases, the likelihood function become very complex and the evidence function practically intractable, hence finding the posterior distribution of the parameters may be unfeasible. Here is where Approximate Bayesian Computation (ABC) [47] plays a major role, given that it avoids the formulation of the evidence function [48] and replaces the likelihood function by a binary function, which will take the unity when the output $\hat{y}$ of the model is close enough to the real data y. Such distance is measured by a user-defined metric function $\rho(\cdot)$ based on summary statistics $\eta(\cdot)$ [49], and the limit on how close it needs to be is defined by a tolerance value ϵ. Then, the Subset Simulation method [50] helps to reduce the computational cost [51,52] by transforming the simulation of a rare event (i.e., $\hat{y}$ is close enough to y under a predefined ϵ) in a sequence of simulations with larger probabilities. The interested reader is referred to [53] for further information about ABC-SS and its implementation.

This Bayesian method has been applied to different fields [54], including feed-forward ANN [55], providing accurate results along with a realistic representation of the uncertainty. The gradient-free nature of the algorithm, the absence of likelihood function and the non-parametric formulation of the weights are responsible for its good performance. This suggests that RNN could significantly benefit from ABC-SS, given its sensitivity to gradient-related problems found in the backpropagation algorithm.

The ABC-SS training process starts by using the prior information PDF $p(\theta)$ to generate N samples of the model parameters, in our case the weights and bias $\theta = \{W, U, V, b, c\}$. Subsequently, all samples are used to run the model, or the forward pass in the case of ANN/RNN, and the error made on training data is measured using the metric function ρ. Based on such metric, NP_0 samples with the lowest value ρ are selected as *seeds* for the next subset, and the tolerance value ϵ_j is fixed. Using the Modified Metropolis Algorithm, $(1/P_0 - 1)$ samples are drawn from each seed, until the sub-region is repopulated with $N - NP_0$ new samples. This process is repeated until the desired tolerance value is reached. Fig. 2 provides an schematic representation of the ABC-SS training process.

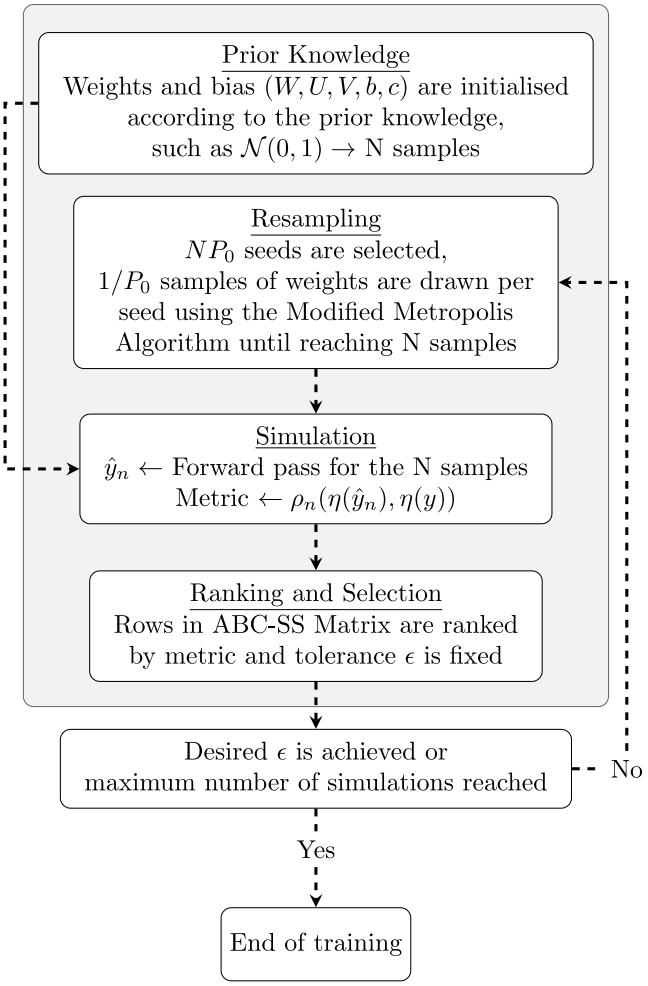


Fig. 2. Schematic representation of ABC-SS training for ANN.

2.3. Physics-guided neural networks

Physics-based models aim to explain natural phenomena and processes through mathematical expressions, and have provided predictive tools for the last few centuries by using relatively small amounts of

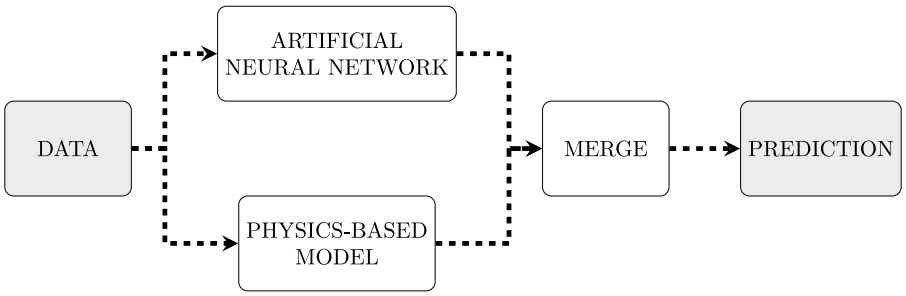


Fig. 3. Generic diagram of physics-guided artificial neural networks.

data. However, despite their good performance in many fields, it is fair to say that they are just a representation of a hypothesized physical reality, which will always differ from the *true* reality. This is because physics-based models cannot take into account every single subtlety of the real world. Furthermore, the closer to reality, the more complex these models need to be [56], reaching computational inefficiency and becoming unsuitable for large-scale real-time prediction activities. In addition, the amount and frequency of data collection nowadays may aggravate the need for computation power in such complex physics-based models. On the contrary, ANN have the flexibility of capturing patterns and processes by learning directly from real-world data. They have excelled in many fields and different tasks, such as regression or classification problems. However, they also suffer from some drawbacks, and their need for huge amounts of data is an important one. In simplistic terms, the performance of ANN depends on the amount and quality of the data used to train them. Unfortunately, data is still scarce and noisy in many industries.

Therefore, it seems just sensible to make use of both approaches and create hybrid models [57,58] that benefit from the knowledge gained through physics-based models and the flexibility of ANN to learn from data. These models can be found in the literature under different names, such as *Physics-Informed Neural Networks (PINN)* [26], *Neural Networks Augmented Physics Models (NNAP)* [59] or *Physics-Guided Neural Networks (PGNN)* [60]. The main difference between them lies in their architecture, and how the physics are introduced in the overall model. While PGNN is a more generic term, it normally refers to hybrid models that merge the output from the physics-based model and from the ANN to create a single output. PINN usually refers to hybrid models where the physics are formulated through differential equations, including certain boundary conditions. Conversely, NNAPs blend physics-inspired layers and neural layers, so that the later compensate for the unknown parameters or conditions. Also, the way physics and ANN are combined depends on the nature of the physics-based models and their formulation, influencing different parts of the hybrid model, such as the loss function or the forward pass. In this manuscript we will focus on PGNN (Residual Model), given their ability to fill the gaps between reality and the physics and to extrapolate outside the training data domain. Fig. 3 illustrates the concept of physics-guided ANN. This methodology may help the problem presented in Section 2.1, about extrapolation and multistep-ahead forecasting.

3. PG-BRNN by ABC-SS

This manuscript proposes a novel RNN, which combines the three methods described in Sections 2.1–2.3, to avoid some of the drawbacks of the state-of-the-art approaches. Starting from a vanilla RNN, the most basic form of the forward pass is maintained in the proposed RNN, as per Eq. (1). However, the hidden states no longer need to be stored, given that the weights are now trained with ABC-SS instead of backpropagation, and the gradient of the loss function does not need to be evaluated. For this same reason, long-term dependencies may be learnt without the need for more complex architectures with gated

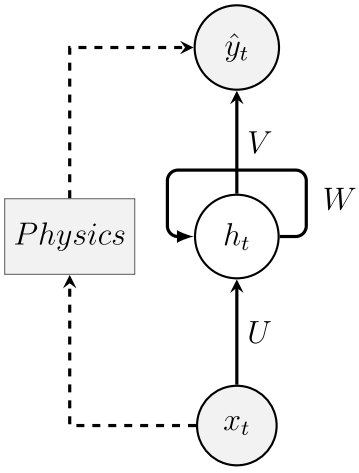


Fig. 4. Schematic representation of the folded Physics-guided BRNN by ABC-SS.

units, as ABC-SS do not suffer from vanishing gradients. A detailed pseudocode is provided in Algorithm 1, which describes the adaptation of the ABC-SS method to train RNN. So far in this section, the principles and algorithms presented in Sections 2.1 and 2.2 have been combined into a data-driven Bayesian RNN, hereafter called BRNN by ABC-SS.

The next step consists of introducing the physics-based model into the proposed BRNN by ABC-SS. As stated in Section 2.3, this manuscript follows the physics-guided approach, illustrated in Fig. 3, where the physics are introduced in the forward pass through the output neurons, like an extra bias parameter. Fig. 4 illustrates the proposed physics-guided forward pass, and Eq. (3) provides its mathematical formulation (note that f_1 and f_2 represent the activation functions chosen by the user).

$$\begin{aligned} a_t &= b + W h_{t-1} + U x_t \\ h_t &= f_1(a_t) \\ o_t &= c + V h_t + Physics(x_t) \\ \hat{y}_t &= f_2(o_t) \end{aligned} \tag{3}$$

The result is a physics-guided RNN trained with ABC-SS, hereafter called PG-BRNN by ABC-SS, which provides probabilistic outputs based on physics-based knowledge and observed data. The Bayesian training, and more specifically, the absence of gradient evaluation and non-parametric formulation of the weights, may provide a series of benefits, such as: avoid problems like *exploding/vanishing gradients* or getting stuck at an undesired local minima of the loss function; probabilistic predictions coupled with a flexible quantification of the uncertainty; and a Bayesian regularization effect in the inference of the weights thanks to the prior PDF. Finally, the physics-based model is part of the forward pass, thus it is expected to improve the extrapolation capabilities of the neural network. These benefits are evaluated and discussed in the experimental section.

Algorithm 1 *Training algorithm for RNN with ABC-SS*

Begin:
Create ABC-SS matrix M_{RNN}, with N rows and as many columns as the total number of parameters (θ) in U, W, V, b and c, plus one additional column to store the metric ρ
for $n : 1, ..., N$ **do**
  Sample initial parameters $\theta_0^{(n)}$, a vector containing all the weights and bias in the RNN, from prior PDFs, such as $\mathcal{N}(0, I)$, and store them in M_{RNN}
  Rearrange $\theta_0^{(n)}$ in matrices U, W, V, b and c
  for $t : 1, ..., T$ **do**
    $h_0 \leftarrow$ Initiate the hidden state at time-step $t = 0$ with a zero matrix
    $\hat{y}_0^{(n)} \leftarrow$ Recurrent forward pass using Equation (1) (or Equation (3) for physics-guided RNN), $U_{\theta_0^{(n)}}$, $W_{\theta_0^{(n)}}$, $V_{\theta_0^{(n)}}$, $b_{\theta_0^{(n)}}$ and $c_{\theta_0^{(n)}}$, and input x from $\mathcal{D}$
  end for
  $\rho_0^{(n)} \leftarrow \rho(\eta(\hat{y}_0^{(n)}), \eta(y))$
  Store $\rho_0^{(n)}$ in $M_{RNN} = \{\theta_0^{(n)}, \rho_0^{(n)}\}_{n=1}^{N}$
end for
Set $j \leftarrow 1$, $\ell \leftarrow$ Maximum number of simulations, and $L_\epsilon \leftarrow$ Tolerance threshold
while $j < \ell$ and $\epsilon_j < L_\epsilon$ **do**
  Renumber rows in M_{RNN} $[\theta_{j-1}^{(n)}, n : 1, \dots, N]$ so that $\rho_{j-1}^{(1)} \leqslant \dots \leqslant \rho_{j-1}^{(n)} \leqslant \dots \leqslant \rho_{j-1}^{(N)}$
  $\epsilon_j \leftarrow \rho_{j-1}^{NP_0}$
  $\sigma_j \leftarrow \sigma_{j-1} * p$ {where p represents the decrease rate of standard deviation per simulation level}
  $M_{SubSet} \leftarrow$ Initiate the SubSet matrix with NP_0 rows and same number of columns than M_{RNN}
  $C \leftarrow 1$ {Initiate counter}
  for $i : 1, \dots, NP_0$ **do**
    $\theta_j^{(i)} \leftarrow \theta_{j-1}^{(i)}$ and $\rho_j^{(i)} \leftarrow \rho_{j-1}^{(i)}$
  end for
  $M_{Seeds} = \{\theta_j^{(n)}, \rho_j^{(n)}\}_{n=1}^{NP_0}$ {Create the Seeds matrix}
  for $k : 1, \dots, NP_0$ **do**
    $\mu \leftarrow \theta_j^{(k)}$
    for $i : 1, \dots, ({}^{1}/_{P_0}) - 1$ **do**
      $\theta^* \sim \mathcal{N}(\mu, \sigma_j)$
      Rearrange θ^* in matrices U, W, V, b and c
      for $t : 1, ..., T$ **do**
        $h_0 \leftarrow$ Initiate the hidden state at time-step $t = 0$ with a zero matrix
        $\hat{y}^* \leftarrow$ Use Equation (1), or (3) for physics-guided RNN, to run a recurrent forward pass with U_{θ^*}, W_{θ^*}, V_{θ^*}, b_{θ^*} and c_{θ^*}
      end for
      $\rho^* \leftarrow \rho(\eta(\hat{y}^*), \eta(y))$
      if $\rho^* \leqslant \epsilon_j$ **then**
        Store θ^* and ρ^* in M_{SubSet} as $\{\theta_j^{(NP_0+C)}, \rho_j^{(NP_0+C)}\}$
        $\mu \leftarrow \theta^*$
      else
        Store θ_j^k and ρ_j^k in M_{SubSet} as $\{\theta_j^{(NP_0+C)}, \rho_j^{(NP_0+C)}\}$
      end if
      $C \leftarrow C + 1$
    end for
  end for
  Update $M_{RNN} = \{\theta_j^{(n)}, \rho_j^{(n)}\}_{n=1}^{N}$ as the concatenation of M_{Seeds} and M_{SubSet}
  $j \leftarrow j + 1$
end while

4. Experimental framework

The proposed BRNN by ABC-SS has firstly been applied to a data-driven experiment about fatigue damage evolution in composite materials, so as to illustrate the benefits of Bayesian training over conventional backpropagation in RNN. Next, PG-BRNN by ABC-SS is demonstrated in an engineering case study about accelerations in concrete buildings during an experimental earthquake, which represent the main application of this manuscript. The experimental framework for both, the data-driven example and the engineering case study, is set out in this section, including a description of the experiments and information about the data source and the data preparation process. Finally, the proposed algorithms have been compared with the state-of-the-art RNN, and the details of their implementation along with the metrics used in the experiments are presented at the end of the section.

4.1. Data-driven example: Fatigue in composite materials

Composite materials, and especially carbon fiber polymer composites (CFRP), have become very popular in several industries such as wind energy and aerospace, thanks to their outstanding performance and characteristics. Indeed, these materials can reach strength-to-weight and stiffness-to-weight ratios up to 5 times larger than grade steel, and they are corrosion resistant [61]. However, they are heterogeneous materials and their damage response is very complex to understand [62], let alone predicting with accuracy the damage behavior of several interacting parts subjected to fatigue loads. In simplistic terms, fatigue damage in CFRP comprises different families of intralaminar and interlaminar cracks, which in turn may or may not influence the propagation of each other. That uncertainty is one of the reasons why CFRP are still not massively used in safety-critical applications [63].

In this manuscript, RNN are used to determine the evolution of micro-crack density in a CFRP coupon subjected to tension–tension fatigue loading. The data used are taken from the NASA Ames Prognostics Data 283 Repository (CFRP Composites Dataset) [64], corresponding to the cross-ply laminates TD19 and TD21 ($[0_2/90_4]_s$). The damage data were collected using Lamb wave signals and piezoelectric sensors located on either side of the coupon. Each coupon underwent 500,000 loading cycles, and measurements from each pair of sensors (36 possible combinations) were taken at 18 specific cycles during the experiment, resulting on 648 data points per specimen. For this illustrative example, only information about micro-crack density is used, which is lagged to form the inputs and outputs as shown below, resulting in univariate forecasting. The data have been structured in 72 sequential samples, where each sample includes the micro-crack density for the full loading history of each pair of sensors. The full loading cycle of the first pair of sensors in TD19 has been held out as test data. Both training and test data have been normalized.

$$\text{Lagged micro-crack density data} \begin{cases} \text{Input array} \rightarrow (x_1, x_2, \dots, x_{17}) \\ \text{Output array} \rightarrow (x_2, x_3, \dots, x_{18}) \end{cases}$$

While these data have been used in several publications [65] about fatigue diagnostics in composites, [55] used several feed-forward Bayesian neural networks to predict the micro-crack density given the number of cycles, while quantifying the uncertainty. Therefore, this data-driven example will also provide a fair evaluation of the benefits of using RNN to process sequential data, over standard feed-forward architectures.

4.2. Engineering case study: Accelerations in seismic events

In this subsection, the potential of PG-BRNN by ABC-SS to become part of a wider engineering system is explored. Specifically, a case study about an SHM framework to evaluate the response of buildings during seismic events is presented. The application of machine

learning algorithms to SHM has already been studied in the existing literature [66,67], however, those approaches usually consist on using a computational model of the structure to produce synthetic data, which are then used to train a neural network, by generating scenarios of damage and evaluating its impact on the response of specific parts of the structure. The trained neural network is then utilized to predict the health of the structure, using accelerations measurements of the real structure. This approach relies on the capabilities of the model and mainly, on the decisions made during the development of the data set to train the neural network. Therefore, it does not benefit from all the advantages of both, the model and the data. The application of PG-BRNN by ABC-SS allows to compensate and improve the structural model using the observed data, resulting in a hybrid model that can be used in the context of an SHM system to predict the response of buildings, in terms of accelerations, and inform the post-earthquake decision-making process.

The data used in this case study comes from a experimental seismic test of a 17-story concrete structure on a shake table, performed by [68] (data available in https://datacenterhub.org/deedsdv/publications/view/564). The specimen was subjected to several impulsive seismic records and the response was measured in terms of displacement and acceleration in some floors of the building. In particular, the acceleration data used in this experiment corresponds to *TS1-Run2*, 9th floor. The input signal at the base of the building is also available in that data set.

Unlike the previous data-driven example, where the data comprised 72 full loading cycles with 18 time-steps each, the data in this experiment consist of a single sequence of accelerations measured at the 9th floor of the experimental structure during the simulated seismic event. In order to make those data suitable for training RNN, they were lagged and divided in 10-time-steps long samples, as shown below. Therefore, we face univariate time-series forecasting, where predictions about the target variable are made based on historic values of the same variable. The data set is split into training and test data with a 60/40 ratio, meaning that the RNN are trained only with the first 60% of the accelerations. Furthermore, the test data is not fed into the RNN, but starting from the last sample of training data the RNN are asked to recursively predict the next values, where the output from the single-step ahead prediction becomes the last time-step in the input for the next prediction, also known as multi-step ahead forecasting. The test data is used to calculate the metric, and evaluate the performance of the algorithms.

$$\text{Lagged acceleration data} \begin{cases} \text{Input array (1)} \rightarrow (x_1, x_2, \dots, x_{10}) \\ \text{Output array (1)} \rightarrow (x_2, x_3, \dots, x_{11}) \\ \text{Input array (2)} \rightarrow (x_2, x_3, \dots, x_{11}) \\ \text{Output array (2)} \rightarrow (x_3, x_4, \dots, x_{12}) \\ \vdots \end{cases}$$

The physics-based model used in the hybrid neural networks is based on the computational model proposed in [69]. The model comprises 17 masses of 250 kg that are jointed in series by elastic perfectly plastic springs and linear dashpots. The stiffness and damping of the building specimen were not defined in the available information of the test, therefore, Barros et al. [69] applied the method called $\mathcal{A}^2$BC-SubSim to obtain the values of the model parameter that better explain accelerations in the monitored floors. Table 1 shows the maximum a posteriori (MAP) values of the parameters, together with the mean, median and standard deviation of each parameter distribution. The MAP value of such parameters has been used to create the physics-based model.

4.3. Baseline algorithms

The proposed algorithms have been applied to two different experiments, as per Sections 4.1 and 4.2. Their performance have been evaluated and compared against the state-of-the-art RNN. The details of each algorithm and their implementation are given below. An architecture with one input neuron, one output neuron and one single hidden layer with 5 hidden units has been chosen as the baseline for all algorithms, unless specified otherwise. The results and discussion can be found in Section 5.

- *BRNN by ABC-SS*: A vanilla RNN trained with ABC-SS, as detailed in Section 3. The values of the hyperparameters are: P_0=0.2, N=10,000, σ_0=0.9 and $p = 0.50$. In the data-driven example (Section 4.1) the activation function for the hidden layer is *tanh*, and *ReLU* in the engineering case study (Section 4.2). The tolerance value also varies depending on the experiment, with ϵ=0.003 and 0.005 respectively.
- LSTM RNN: This advanced type of RNN was first published in 1997 [6], and is still very popular in the scientific community. The LSTM cell comprises a series of gates that allows the algorithm to forget irrelevant information, add or update new information, and finally pass such updated information forward. This process allows the network to remember long-term dependencies, while mitigates the *vanishing gradient* problem. In our case, the algorithm has been implemented using *Tensorflow* [70] and the hyperparameters used are: minimum squared error (MSE) as the loss function, the optimizer *Adam* [71] and batchsize=1. The number of epochs varies depending on the experiment, with 200 for the data-driven example and 100 for the engineering case study.
- Bidirectional LSTM RNN: First proposed in 2005 [72], this neural network is used primarily on natural language processing. In brief, this type of RNN adds an extra LSTM layer that conveys the information on the other direction, so input information flows back and forth. Therefore, the output information at a certain time-step is influenced by past, present and future inputs. This algorithm has been implemented using *Tensorflow* and the hyperparameters used are: minimum squared error (MSE) as the loss function, the optimizer *RMSprop* and batchsize=1. The number of epochs varies depending on the experiment, with 25 for the data-driven example and 50 for the engineering case.
- GRU RNN: This type of recurrent neural network first appeared in 2014 [7], and like LSTM, it is a gated algorithm but with fewer parameters. While it has a forget gate, it lacks the output gate, making it slightly less complex. GRU also mitigates the *vanishing gradient* problem and its performance is comparable to that of LSTM. This algorithm has been implemented using *Tensorflow* and the hyperparameters used are: minimum squared error (MSE) as the loss function, the optimizer *Adam* and batchsize=1. The number of epochs varies depending on the experiment, with 200 for the data-driven example and 100 for the engineering case.
- Monte Carlo Dropout RNN (MC Dropout): The *Dropout* method, published in 2014 [73], meant to mitigate the overfitting problem. This was the origin of a new Bayesian training algorithm for neural networks, MC Dropout [74,75]. This algorithm provides probabilistic outputs with quantification of the uncertainty, and has demonstrated better performance than other state-of-the-art Bayesian methods such as *Variational Inference* [76]. Furthermore, such uncertainty is a combination of epistemic uncertainty (related to model architecture, when different sets of neurons are switched off) and aleatoric uncertainty (related to the variability/noise in the data). Both types are captured by this method and blended into the final output of the model, a probability density function. In our case, MC Dropout LSTM has been implemented following [77] and the code in *GitHub*[1], with *Pytorch* [78]. The dropout method is based on some neurons being dropped on each

[1] https://github.com/PawaritL/BayesianLSTM

Table 1
Values of the model parameters proposed in [69].

	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8
MAP	9.60E+07	7.90E+07	2.50E+08	2.30E+07	9.10E+07	1.80E+08	2.80E+08	2.50E+08
μ	9.30E+07	8.00E+07	2.40E+08	2.40E+07	9.10E+07	1.80E+08	2.70E+08	2.50E+08
σ	9.50E+06	1.60E+07	4.00E+06	2.60E+06	3.50E+06	9.30E+06	2.50E+07	1.80E+07
	θ_9	θ_{10}	θ_{11}	θ_{12}	θ_{13}	θ_{14}	θ_{15}	θ_{16}
MAP	2.46E+08	1.03E+08	2.39E+08	1.06E+08	1.77E+08	1.85E+08	1.22E+07	1.76E+08
μ	2.47E+08	1.02E+08	2.36E+08	1.07E+08	1.79E+08	1.85E+08	1.30E+07	1.75E+08
σ	5.14E+06	5.84E+06	3.00E+07	8.68E+06	3.41E+06	1.83E+07	3.41E+06	1.32E+07
	θ_{17}	θ_{18}	θ_{19}	θ_{20}	θ_{21}	θ_{22}	θ_{23}	θ_{24}
MAP	1.70E+08	3.92E+07	1.75E+07	3.50E+07	3.21E+07	3.88E+07	4.50E+07	4.53E+07
μ	1.70E+08	3.89E+07	1.72E+07	3.54E+07	3.22E+07	3.85E+07	4.45E+07	4.55E+07
σ	3.54E+06	2.38E+06	1.70E+06	1.97E+06	390 594	2.66E+06	2.31E+06	2.66E+06
	θ_{25}	θ_{26}	θ_{27}	θ_{28}	θ_{29}	θ_{30}	θ_{31}	θ_{32}
MAP	3.77E+07	1.13E+07	3.05E+07	1.23E+07	4.25E+07	4.32E+07	1.24E+07	2.93E+07
μ	3.77E+07	1.23E+07	3.03E+07	1.29E+07	4.25E+07	4.33E+07	1.20E+07	2.96E+07
σ	895 961	5.07E+06	3.18E+06	4.28E+06	4.84E+06	697 790	1.12E+06	1.83E+06
	θ_{33}	θ_{34}	θ_{35}					
MAP	4.00E+07	2.04E+07	7424.2					
μ	4.04E+07	2.01E+07	7529.9					
σ	2.40E+06	1.05E+06	278.306					

Note: θ_1 to θ_{17} are the corresponding elastic stiffness of each floor. θ_{18} to θ_{34} are the corresponding yield strength of each floor. θ_{35} is the damping coefficient of every floor.

forward/backward pass with a given probability. That means that a higher number of hidden units are required for the RNN to work. In our experiments, the architecture of MC Dropout RNN has been chosen so it reaches a similar performance than the other state-of-the-art approaches. That is, two LSTM layers with 128 and 32 hidden units respectively. Regarding the hyperparameters, the dropout probability is 0.5, the loss function is MSE and batch size=8. The number of epochs and the learning rate (lr) varies between experiments: in the data-driven example the number of epochs is 150 and lr=0.01, while in the engineering case we need 50 epochs and lr=0.0001.

- PG-BRNN by ABC-SS: A physics-guided RNN trained with ABC-SS as per Section 3. The physics-based model, specified in Section 4.2, is introduced into the RNN through the output neuron. The hyperparameters are: P_0=0.2, N=10,000, σ_0=0.9 and p = 0.50. The activation function in the hidden layer is *ReLU* and the tolerance value ϵ=0.0047.
- PG-LSTM RNN: A physics-guided LSTM RNN, with the physics-based model introduced in the output neuron, as per Section 2.3. The hyperparameters are: MSE as the loss function, the optimizer *Adam*, batchsize=1 and 10 epochs.
- PG-Bidirectional LSTM RNN: A physics-guided Bidirectional LSTM RNN, with the physics-based model introduced in the output neuron, as per Section 2.3. The hyperparameters are: MSE as the loss function, the optimizer *RMSprop*, batchsize=1 and 30 epochs.
- PG-GRU RNN: A physics-guided GRU RNN, with the physics-based model introduced in the output neuron, as per Section 2.3. The hyperparameters are: MSE as the loss function, the optimizer *Adam*, batchsize=1 and 10 epochs.
- PG-MC Dropout LSTM RNN: A physics-guided MC Dropout LSTM RNN, with the physics-based model introduced in the output neuron, as per Section 2.3. The hyperparameters are: dropout probability is 0.5, the loss function is MSE, batch size=8, lr=0.0001 and 40 epochs.
- PbM: Physics-based model to be used as a reference in the engineering case study. Details about the model can be found in Section 4.2.

The performance of the different algorithms has been evaluated after 30 independent runs for each experiment. The accuracy of each algorithm is determined by the average and median MSE obtained, while the stability and reliability of the algorithm is measured by the interquartile range (IQR). The ability to quantify the uncertainty, in those algorithms trained with Bayesian methods, is evaluated both using the width of the confidence bounds and graphically. Such uncertainty, understood as the degree of belief in the predictions, comprises epistemic uncertainty arising from the model architecture and/or lack of data, and aleatoric uncertainty related to the inherent randomness or natural variability in the data. Both types of uncertainty are combined and expressed through the probabilistic outputs of the Bayesian algorithms.

5. Results and discussion

In this section, the results from the data-driven example and the engineering case study are presented and discussed. The performance of the proposed algorithms are evaluated in detail, and compared against the state-of-the-art RNN.

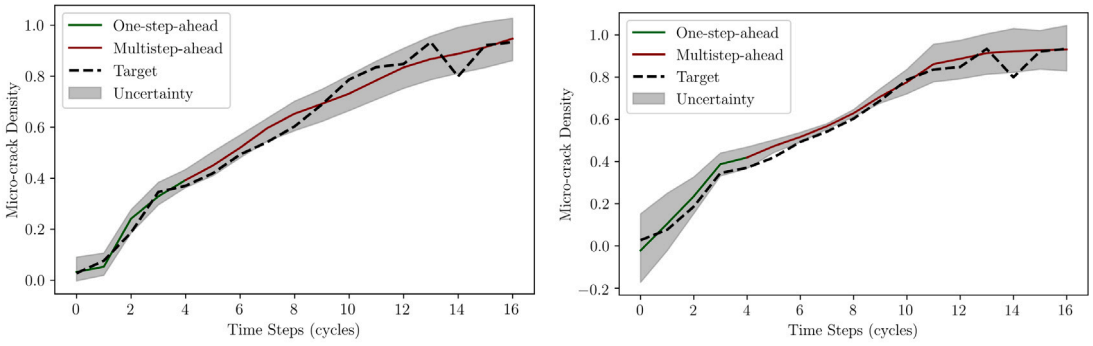
5.1. Data-driven example

The uncertainty about the long term behavior of CFRP materials under fatigue conditions and our inability to accurately predict their remaining useful life, are among the main obstacles for these materials to be massively used as main structural materials in industries such as aerospace engineering. There is a clear need for efficient physics-based models that could unify all the different damage modes into a single formulation. Moreover, these models should also take into account small imperfections that appear during the manufacturing process. While this could be extraordinarily complex, data-driven methods are providing promising results and are becoming a solid alternative for the evaluation of fatigue and its propagation in composites.

In this experiment, the proposed BRNN by ABC-SS along with the state-of-the-art RNN have been trained using full sequences about fatigue damage evolution, and then tested on unseen data, as per Section 4.1. During testing, the RNN were provided with the first 4 time-steps of the sequence, corresponding to 22% of such loading sequence, and were then asked to make predictions about future time-steps, taking as inputs their own output from the previous time-step, also called recursive multi-step forecasting. That way, the ability of the different RNN to make long-term predictions about the progression of micro-crack density can be assessed. The results are shown in Table 2. In terms of accuracy, it can be seen that BRNN by ABC-SS have comparable performance to LSTM and MC Dropout, although

Table 2
Performance of Recurrent Neural Networks, evaluated using MSE after 30 independent runs of the algorithms.

Statistics of MSE obtained in 30 independent runs of the training algorithms					
	Average	Q1(P_{25})	Median(P_{50})	Q3(P_{75})	IQR(Q3-Q1)
RBNN by ABC-SS	**0.00149**	0.00141	0.00150	**0.00155**	**0.00014**
MC Dropout LSTM RNN	0.00161	0.00144	0.00152	0.00191	0.00047
LSTM RNN	0.00153	**0.00126**	**0.00148**	0.00172	0.00047
Bidirectional LSTM RNN	0.05857	0.04420	0.06019	0.06835	0.02414
GRU RNN	0.00220	0.00167	0.00207	0.00256	0.00089



(a) RBNN by ABC-SS (b) MC Dropout LSTM RNN

Fig. 5. Predictions made by RBNN by ABC-SS and MC Dropout LSTM RNN on test data. The green line represents one-step-ahead predictions at time 't', where the input data is the real value of the target variable at time 't-1'. The red line are multi-steps-ahead predictions, where the input data are the previous predictions made by the RNN. The dashed black line represents the target values. The grey hatches are the uncertainty bounds (P_5 − P_{95}).

this last one required a more complex architecture with significantly more neurons. GRU obtained slightly less accurate results, while the performance of Bidirectional LSTM was significantly worse than the rest of the algorithms for this experiment. The lowest values of MSE, on an individual basis, were achieved by LSTM, as can be seen from its P_{25}. In terms of reliability, BRNN by ABC-SS achieved a low IQR value, circa 4 times lower than the state-of-the-art algorithms. This means that throughout the 30 runs of the algorithm, the proposed Bayesian RNN consistently obtained similar results, with little deviation. The ability to quantify the uncertainty can be observed in Fig. 5, where the proposed BRNN by ABC-SS is compared against the state-of-the-art Bayesian RNN, namely MC Dropout. The uncertainty bounds of BRNN by ABC-SS naturally increases as we move rightwards along the time-steps axis. Furthermore, the width of this uncertainty bounds at time-steps/cycles 0, 4 and 16 are 0.09, 0.12 and 0.17 (normalized data) respectively, which makes sense given that, in a real world scenario, predictions about the structural behavior of an asset in the far future tend to be more uncertain than those ones about the near future. Contrarily, the width of the uncertainty bounds predicted by MC Dropout for those time-steps are 0.31, 0.04 and 0.21, which does not follow the expected monotonically increasing behavior. In addition, the uncertainty bounds of RBNN by ABC-SS encapsulates most target values. As a note, the results obtained improve those in [55], which confirm that RNN generally perform better on sequential data than feed-forward neural networks.

Overall, RBNN by ABC-SS has demonstrated great accuracy, comparable to that of LSTM and MC Dropout, while showing more stability, becoming a more reliable option. The absence of gradient evaluation is behind such stability, given that ABC-SS does not rely on finding a local minima of a loss function which, in the case of the state-of-the-art RNN, varies on each run of the algorithm. The quantification of the uncertainty is significantly more accurate in BRNN by ABC-SS, mostly thanks to the non-parametric formulation of the weights, which provide great flexibility to adapt to the observed data. Providing that enough data was available, BRNN by ABC-SS might become a useful option for an onboard structural health monitoring system.

5.2. Engineering case study

As explained in Section 4.2, the engineering case study can be considered a multi-step-ahead time-series forecasting task. This is one of the greatest challenges that RNN face nowadays, that is, making predictions many time-steps ahead based on its own previous predictions. Multi-step-ahead forecasting might be seen as a form of extrapolation, where the neural network needs to make predictions about data it has not seen before. And that is exactly where physics-based models may help, as explained in Section 2.3, given their capacity to generalize well through the whole domain of the data space. Therefore, PG-BRNN by ABC-SS along with the physics-guided version of the state-of-the-art RNN are applied to this case study.

The results from the experiment, after 30 independent runs of each algorithm, are shown in Table 3. Overall, it can be seen that the physics-guided version of the different RNN have clearly achieved better results than their data-driven counterparts. The neural networks seemed to have learnt a pattern in the discrepancy between the physics-based model and the observed reality during training, and then applied such pattern to the test data, also improving the results obtained from the stand-alone physics-based model. This superiority is even more obvious when looking at the violin plot in Fig. 6, where the left side of each plot represents the distribution of the MSE obtained by the physics-guided versions, and the right side refers to the data-driven versions of each RNN. Clearly, the error made by the physics-guided algorithms is negligible in comparison with their data-driven counterpart. The main reason behind the poor performance of the data-driven RNN lies in their inability to extrapolate, and the fact that the error of each prediction is sequentially added up, thus new predictions are built upon increasingly wrong predictions. That leads to unreliable outputs just a few time-steps away from the starting point. In this case, making a comparison between the different data-driven approaches seems irrelevant, given their volatility and instability. Regarding the physics-guided versions of the RNN, they provide accurate results with comparable precision across all the algorithms. PG-GRU RNN and PG-LSTM RNN

Table 3
Performance of data-driven Recurrent Neural Networks, Physics-Guided Recurrent Neural Networks and the Physics-based Model, evaluated using MSE after 30 independent runs of the algorithms.

Statistics of MSE obtained in 30 independent runs of the training algorithms					
	Average	Q1(P_{25})	Median(P_{50})	Q3(P_{75})	IQR(Q3-Q1)
PG-RBNN by ABC-SS	**0.00046**	0.00045	0.00046	**0.00048**	**0.00003**
PG-MC Dropout LSTM RNN	0.00076	0.00071	0.00076	0.00081	0.00010
PG-LSTM RNN	0.00053	0.00042	**0.00044**	0.00057	0.00015
PG-Bidirectional LSTM RNN	0.00052	0.00046	0.00049	0.00055	0.00008
PG-GRU RNN	0.00049	**0.00042**	0.00045	0.00057	0.00015
RBNN by ABC-SS	0.06020	0.03319	0.05689	0.08901	0.05281
MC Dropout LSTM RNN	0.09804	0.08399	0.10925	0.11719	0.0332
LSTM RNN	0.28129	0.10344	0.17070	0.33212	0.22868
Bidirectional LSTM RNN	0.07643	0.04177	0.06278	0.08725	0.04548
GRU RNN	0.21473	0.09021	0.14304	0.21798	0.12776
Physics-based Model			0.00825		

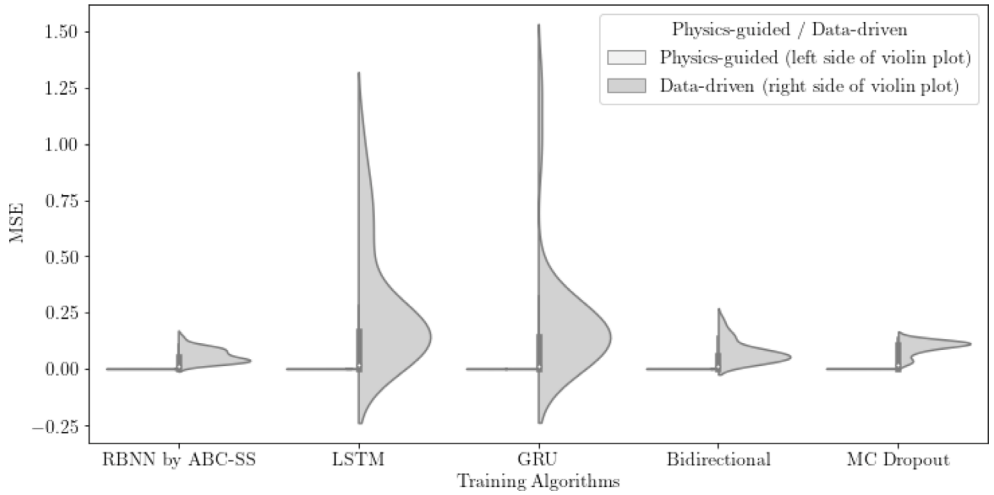


Fig. 6. Violin plot showing the distribution of the MSE obtained by the physics-guided and data-driven version of each algorithm after 30 independent runs. The left side of each plot represents the errors made by the physics-guided version, and the right side represents the data-driven version of each algorithm.

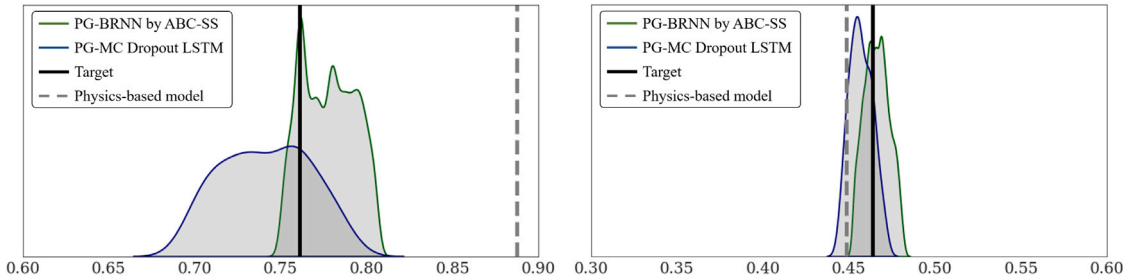
have achieved the lowest P_{25} and median MSE values respectively, which demonstrates their capacity to find the optimal local minimum of the loss function, while PG-BRNN by ABC-SS provided the lowest average MSE and P_{75}. That, added to the low IQR value, reinforces the stability and reliability of PG-BRNN by ABC-SS over the state-of-the-art PG-RNN. Also, the quantification of the uncertainty of PG-BRNN by ABC-SS is more realistic than that of its Bayesian competitor PG-MC Dropout, as shown in Fig. 7. PG-BRNN by ABC-SS seems slightly more confident as well as precise. Another reading from Fig. 7 is that both Bayesian RNN are quite confident about the point when accelerations change in direction, but not so much about when those reach the highest levels. This suggest that the inflexion point between accelerating and decelerating is the most difficult part to predict. Finally, the computational cost of the top-performing algorithms is also comparable. The training time of PG-BRNN by ABC-SS in this case study is 36 s, PG-LSTM RNN 32 s, PG-GRU RNN 46 s and PG-MC Dropout 71 s (presumably due to its more complex architecture). Nonetheless, the performance of PG-BRNN by ABC-SS in terms of computational cost could be improved if parallel computing was used during the sampling phase, or by using optimized machine learning libraries.

The predictions provided by PG-BRNN by ABC-SS have also been evaluated in the frequency domain after applying the fast Fourier transform (fft) [79], as shown in Fig. 8. These values are used in structural engineering to identify the modal characteristics of a dynamical system, such as their fundamental frequency. It can be seen that the physics-based model captures the dynamic characteristics of the specimen, as it correctly predicts the fundamental frequency of the system, however, it clearly overestimates the amplitude of the movement. On the other hand, PG-RNN by ABC-SS clearly corrects the deficiency of the physics-based model, providing an accurate estimation of the fft. This information could become valuable for future assessments about the structural integrity of the building.

6. Conclusions

The success of RNN in all their different versions is unquestionable, however, their performance heavily rely on big training data sets, and those are a rare sight in the civil and structural engineering industry. Furthermore, the training process of the state-of-the-art RNN is based on the evaluation of a loss function and the use of the backpropagation algorithm, which implies some well known drawbacks such as *vanishing* and *exploding gradients*, or reaching different local minima in each run of the algorithm, providing varying results.

In this manuscript, a novel physics-guided Bayesian RNN trained with ABC-SS was proposed. The physics-based models are introduced in the forward pass of the RNN, which mitigates the problems related to lack of data and allows for extrapolation. This is especially important when multistep-ahead forecasting is required. At the same time, the use of ABC-SS as the learning engine translates into non-parametric probabilistic weights, Bayesian regularization, and probabilistic outputs with accurate quantification of the uncertainty. Also, the absence of gradient evaluation in ABC-SS allows for long-term dependencies to be



(a) Time-step 270 (acceleration reaches maximum levels)

(b) Time-step 427 (change in direction of the acceleration)

Fig. 7. PDF (P_5 - P_{95}) of the predictions made by PG-RBNN by ABC-SS (green) PG-MC Dropout LSTM RNN (blue). The black line represents the target value and the dashed grey line is the prediction made by the physics-based model.

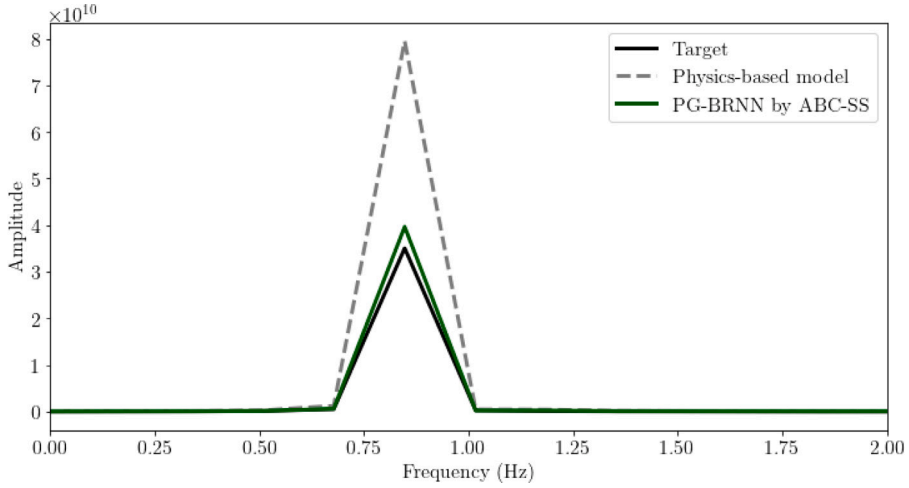


Fig. 8. Comparison of the Fourier amplitudes provided by the physics-based model [69] (dashed-grey), PG-BRNN by ABC-SS (green), and experimental measurements (black).

learnt without the need for more complex architectures. The proposed Bayesian RNN has been applied to two different structural engineering experiments about fatigue damage progression in composites and seismic accelerations in reinforced concrete buildings. The results have shown that while PG-BRNN by ABC-SS provide comparable accuracy to the state-of-the-art PG-RNN, its predictions in different runs of the algorithm present very little deviation, resulting in a more reliable option. Also, when compared with its Bayesian competitor MC Dropout, the proposed algorithm provided a more precise and realistic quantification of the uncertainty. In relation to real-world applications, BRNN by ABC-SS could be explored as a quasi-real time predictor for onboard SHM systems, provided that enough real data is available. Likewise, PG-BRNN by ABC-SS has demonstrated potential to become an on-site prediction tool for seismic events and/or aftershocks in buildings, thus helping to evaluate its structural integrity and the safety of the utility systems.

Finally, ABC-SS training is currently limited by the dimension of the parameter space, and may not be suitable to train very large RNNs with millions of neurons, such as those used for video activity recognition. Also, such complex model architectures would require a large number of samples N, which would result in a significant increase in computational cost. Considering those limitations, future lines of research should focus on solving the dimensional problem, potentially through new and more efficient sampling techniques. The computational cost could also be reduced if parallel computing was used during sampling. Lastly, the benefits of adding the forget, input and output gates to the forward pass of BRNN by ABC-SS, like in LSTM and GRU, may be assessed.

CRediT authorship contribution statement

Juan Fernández: Conceptualization, Methodology, Writing – Original Draft, Formal analysis, Software, Validation, Investigation, Visualization. **Juan Chiachío:** Supervision, Investigation, Resources, Writing – Review & Editing. **José Barros:** Writing – Review & Editing, Investigation, Resources. **Manuel Chiachío:** Investigation, Resources, Writing – Review & Editing, Project administration, Funding acquisition. **Chetan S. Kulkarni:** Writing – review & editing, Supervision, Investigation.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Manuel Chiachio reports financial support was provided by European Commission Marie Sklodowska-Curie Actions.

Data availability

Data will be made available on request.

Acknowledgments

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 Matteo Corbetta at KBR, Inc. NASA Ames Research Center for his comments and useful discussions.

References

[1] Fan X, Zhang X, Yu XB. Uncertainty quantification of a deep learning model for failure rate prediction of water distribution networks. Reliab Eng Syst Saf 2023;109088.

[2] Bae J, Xi Z. Learning of physical health timestep using the LSTM network for remaining useful life estimation. Reliab Eng Syst Saf 2022;226:108717.

[3] Box GE, Pierce DA. Distribution of residual autocorrelations in autoregressive-integrated moving average time series models. J Am Stat Assoc 1970;65(332):1509–26.

[4] Hyndman R, Koehler AB, Ord JK, Snyder RD. Forecasting with exponential smoothing: the state space approach. Springer Science & Business Media; 2008.

[5] McCulloch WS, Pitts W. A logical calculus of the ideas immanent in nervous activity. Bull Math Biophys 1943;5(4):115–33.

[6] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput 1997;9(8):1735–80.

[7] Chung J, Gulcehre C, Cho K, Bengio Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. 2014, arXiv preprint arXiv:1412.3555.

[8] Cao L, Zhang H, Meng Z, Wang X. A parallel GRU with dual-stage attention mechanism model integrating uncertainty quantification for probabilistic RUL prediction of wind turbine bearings. Reliab Eng Syst Saf 2023;235:109197.

[9] Graves A, Mohamed A-r, Hinton G. Speech recognition with deep recurrent neural networks. In: 2013 IEEE international conference on acoustics, speech and signal processing. Ieee; 2013, p. 6645–9.

[10] Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate. 2014, arXiv preprint arXiv:1409.0473.

[11] Wang C, Yang H, Bartz C, Meinel C. Image captioning with deep bidirectional LSTMs. In: Proceedings of the 24th ACM international conference on multimedia. 2016, p. 988–97.

[12] Wang Y, Huang M, Zhu X, Zhao L. Attention-based LSTM for aspect-level sentiment classification. In: Proceedings of the 2016 conference on empirical methods in natural language processing. 2016, p. 606–15.

[13] Mangal S, Modak R, Joshi P. LSTM based music generation system. 2019, arXiv preprint arXiv:1908.01080.

[14] Donahue J, Anne Hendricks L, Guadarrama S, Rohrbach M, Venugopalan S, Saenko K, et al. Long-term recurrent convolutional networks for visual recognition and description. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2015, p. 2625–34.

[15] Pan Y, Sun Y, Li Z, Gardoni P. Machine learning approaches to estimate suspension parameters for performance degradation assessment using accurate dynamic simulations. Reliab Eng Syst Saf 2023;230:108950.

[16] Wang C, Dou M, Li Z, Outbib R, Zhao D, Zuo J, et al. Data-driven prognostics based on time-frequency analysis and symbolic recurrent neural network for fuel cells under dynamic load. Reliab Eng Syst Saf 2023;109123.

[17] Jalali SMJ, Ahmadian S, Khodayar M, Khosravi A, Ghasemi V, Shafie-khah M, et al. Towards novel deep neuroevolution models: Chaotic levy grasshopper optimization for short-term wind speed forecasting. Eng Comput 2021;1–25.

[18] Wang D, Fan J, Fu H, Zhang B. Research on optimization of big data construction engineering quality management based on RNN-LSTM. Complexity 2018;2018.

[19] Pascanu R, Mikolov T, Bengio Y. On the difficulty of training recurrent neural networks. In: International conference on machine learning. PMLR; 2013, p. 1310–8.

[20] Graves A. Long short-term memory. Supervised sequence labelling with recurrent neural networks. Springer; 2012, p. 37–45.

[21] He H, Garcia EA. Learning from imbalanced data. IEEE Trans Knowl Data Eng 2009;21(9):1263–84.

[22] Haley PJ, Soloway D. Extrapolation limitations of multilayer feedforward neural networks. In: [Proceedings 1992] IJCNN international joint conference on neural networks, vol. 4. IEEE; 1992, p. 25–30.

[23] Ma Z, Liao H, Gao J, Nie S, Geng Y. Physics-informed machine learning for degradation modeling of an electro-hydrostatic actuator system. Reliab Eng Syst Saf 2023;229:108898.

[24] Li Z, Zhou J, Nassif H, Coit D, Bae J. Fusing physics-inferred information from stochastic model with machine learning approaches for degradation prediction. Reliab Eng Syst Saf 2023;232:109078.

[25] Subramanian A, Mahadevan S. Probabilistic physics-informed machine learning for dynamic systems. Reliab Eng Syst Saf 2023;230:108899.

[26] Raissi M, Perdikaris P, Karniadakis GE. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J Comput Phys 2019;378:686–707.

[27] Xu Y, Kohtz S, Boakye J, Gardoni P, Wang P. Physics-informed machine learning for reliability and systems safety applications: State of the art and challenges. Reliab Eng Syst Saf 2022;108900.

[28] Wu B, Hennigh O, Kautz J, Choudhry S, Byeon W. Physics informed RNN-DCT networks for time-dependent partial differential equations. 2022, arXiv preprint arXiv:2202.12358.

[29] Shokouhi P, Kumar V, Prathipati S, Hosseini SA, Giles CL, Kifer D. Physics-informed deep learning for prediction of CO2 storage site response. J Contam Hydrol 2021;241:103835.

[30] Ren P, Rao C, Liu Y, Wang J-X, Sun H. PhyCRNet: Physics-informed convolutional-recurrent network for solving spatiotemporal PDEs. Comput Methods Appl Mech Engrg 2022;389:114399.

[31] Nascimento RG, Fricke K, Viana FA. A tutorial on solving ordinary differential equations using python and hybrid physics-informed neural network. Eng Appl Artif Intell 2020;96:103996.

[32] Nascimento RG, Viana FA. Fleet prognosis with physics-informed recurrent neural networks. 2019, arXiv preprint arXiv:1901.05512.

[33] Nascimento RG, Viana FA. Cumulative damage modeling with recurrent neural networks. AIAA J 2020;58(12):5459–71.

[34] Jia X, Willard J, Karpatne A, Read J, Zwart J, Steinbach M, et al. Physics guided RNNs for modeling dynamical systems: A case study in simulating lake temperature profiles. In: Proceedings of the 2019 SIAM international conference on data mining. SIAM; 2019, p. 558–66.

[35] Jia X, Zwart J, Sadler J, Appling A, Oliver S, Markstrom S, et al. Physics-guided recurrent graph networks for predicting flow and temperature in river networks. 2020, arXiv preprint arXiv:2009.12575.

[36] Daw A, Thomas RQ, Carey CC, Read JS, Appling AP, Karpatne A. Physics-guided architecture (PGA) of neural networks for quantifying uncertainty in lake temperature modeling. In: Proceedings of the 2020 Siam international conference on data mining. SIAM; 2020, p. 532–40.

[37] Gori M, Tesi A. On the problem of local minima in backpropagation. IEEE Trans Pattern Anal Mach Intell 1992;14(1):76–86.

[38] Hopfield JJ. Neural networks and physical systems with emergent collective computational abilities. Proc Natl Acad Sci 1982;79(8):2554–8.

[39] Rumelhart DE, Hinton GE, Williams RJ. Learning representations by back-propagating errors. Nature 1986;323(6088):533–6.

[40] Elman JL. Finding structure in time. Cogn Sci 1990;14(2):179–211.

[41] Goodfellow I, Bengio Y, Courville A. Deep learning. MIT Press; 2016.

[42] Schuster M, Paliwal KK. Bidirectional recurrent neural networks. IEEE Trans Signal Process 1997;45(11):2673–81.

[43] Razavi-Far R, Chakrabarti S, Saif M. Multi-step-ahead prediction techniques for lithium-ion batteries condition prognosis. In: 2016 IEEE international conference on systems, man, and cybernetics. IEEE; 2016, p. 004675–80.

[44] Bayes T. LII. An essay towards solving a problem in the doctrine of chances. By the late Rev. Mr. Bayes, FRS communicated by Mr. Price, in a letter to John Canton, AMFR S. Philos Trans R Soc Lond 1763;(53):370–418.

[45] de Laplace PS. Théorie analytique des probabilités, vol. 7. Courcier; 1820.

[46] Jeffreys H. Theory of probability. third ed.. Oxford, England: Oxford; 1961.

[47] Marin J-M, Pudlo P, Robert CP, Ryder RJ. Approximate Bayesian computational methods. Stat Comput 2012;22(6):1167–80.

[48] Santoso A, Phoon K, Quek S. Modified Metropolis–Hastings algorithm with reduced chain correlation for efficient subset simulation. Probab Eng Mech 2011;26(2):331–41.

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

[50] Au S-K, Beck JL. Estimation of small failure probabilities in high dimensions by subset simulation. Probab Eng Mech 2001;16(4):263–77.

[51] Au SK, Ching J, Beck J. Application of subset simulation methods to reliability benchmark problems. Struct Saf 2007;29(3):183–93.

[52] Ching J, Au S-K, Beck JL. Reliability estimation for dynamical systems subject to stochastic excitation using subset simulation with splitting. Comput Methods Appl Mech Engrg 2005;194(12–16):1557–79.

[53] Chiachio M, Beck JL, Chiachio J, Rus G. Approximate Bayesian computation by subset simulation. SIAM J Sci Comput 2014;36(3):A1339–58.

[54] Hazra I, Pandey MD, Manzana N. Approximate Bayesian computation (ABC) method for estimating parameters of the gamma process using noisy data. Reliab Eng Syst Saf 2020;198:106780.

[55] Fernández J, Chiachío M, Chiachío J, Muñoz R, Herrera F. Uncertainty quantification in neural networks by approximate Bayesian computation: Application to fatigue in composite materials. Eng Appl Artif Intell 2022;107:104511.

[56] White A, Tolman M, Thames HD, Withers HR, Mason KA, Transtrum MK. The limitations of model-based experimental design and parameter estimation in sloppy systems. PLoS Comput Biol 2016;12(12):e1005227.

[57] Willard J, Jia X, Xu S, Steinbach M, Kumar V. Integrating physics-based modeling with machine learning: A survey. 2020, p. 1–34, arXiv preprint arXiv:2003.04919, 1, 1.

[58] Rai R, Sahu CK. Driven by data or derived through physics? a review of hybrid physics guided machine learning techniques with Cyber-Physical System (CPS) focus. IEEE Access 2020;8:71050–73.

[59] De Groote W, Kikken E, Hostens E, Van Hoecke S, Crevecoeur G. Neural network augmented physics models for systems with partially unknown dynamics: Application to slider-crank mechanism. IEEE/ASME Trans Mechatronics 2021.

[60] Daw A, Karpatne A, Watkins W, Read J, Kumar V. Physics-guided neural networks (PGNN): An application in lake temperature modeling. 2017, arXiv preprint arXiv:1710.11431.

[61] Markovičová L, Zatkalíková V, Hanusová P. Carbon fiber polymer composites. In: Conference quality production improvement–CQPI, vol. 1, no. 1. 2019, p. 276–80.

[62] Talreja R. Damage and fatigue in composites–a personal account. Compos Sci Technol 2008;68(13):2585–91.
[63] Chiachío J, Chiachío M, Saxena A, Sankararaman S, Rus G, Goebel K. Bayesian model selection and parameter estimation for fatigue damage progression models in composites. Int J Fatigue 2015;70:361–73.
[64] Saxena A, Goebel K, Larrosa C, Chank F-K. CFRP composites data set, NASA Ames prognostics data repository. NASA Ames Research Center, Moffett Field, CA, URL http://ti.arc.nasa.gov/project/prognostic-data-repository.
[65] Dabetwar S, Ekwaro-Osire S, Dias JP. Damage classification of composites using machine learning. In: ASME international mechanical engineering congress and exposition, vol. 83501. American Society of Mechanical Engineers; 2019, V013T13A017.
[66] Chang C-M, Lin T-K, Chang C-W. Applications of neural network models for structural health monitoring based on derived modal properties. Measurement 2018;129:457–70. http://dx.doi.org/10.1016/j.measurement.2018.07.051, URL https://www.sciencedirect.com/science/article/pii/S0263224118306559.
[67] Yu Y, Wang C, Gu X, Li J. A novel deep learning-based method for damage identification of smart building structures. Struct Health Monit 2019;18(1):143–63. http://dx.doi.org/10.1177/1475921718804132.
[68] Pratap P, Pujol S. Dynamic tests of an idealized long-period structure. DEEDS 2021.
[69] Barros J, Chiachío M, Chiachío J, Cabanilla F. Adaptive approximate Bayesian computation by subset simulation for structural model calibration. Comput-Aided Civ Infrastruct Eng 2022;37(6):726–45. http://dx.doi.org/10.1111/mice.12762, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/mice.12762, URL https://onlinelibrary.wiley.com/doi/abs/10.1111/mice.12762.
[70] Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, et al. TensorFlow: Large-scale machine learning on heterogeneous systems. 2015, Software available from tensorflow.org.
[71] Kingma DP, Ba J. Adam: A method for stochastic optimization. 2014, arXiv preprint arXiv:1412.6980.
[72] Graves A, Schmidhuber J. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural Netw 2005;18(5–6):602–10.
[73] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: A simple way to prevent neural networks from overfitting. J Mach Learn Res 2014;15(1):1929–58.
[74] Gal Y, Ghahramani Z. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In: International conference on machine learning. PMLR; 2016, p. 1050–9.
[75] Gal Y, Ghahramani Z. A theoretically grounded application of dropout in recurrent neural networks. In: Advances in neural information processing systems, vol. 29. 2016.
[76] Chen J, Pi D, Wu Z, Zhao X, Pan Y, Zhang Q. Imbalanced satellite telemetry data anomaly detection model based on Bayesian LSTM. Acta Astronaut 2021;180:232–42.
[77] Zhu L, Laptev N. Deep and confident prediction for time series at uber. In: 2017 IEEE international conference on data mining workshops. IEEE; 2017, p. 103–10.
[78] Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. PyTorch: An imperative style, high-performance deep learning library. In: Advances in neural information processing systems, vol. 32. Curran Associates, Inc.; 2019, p. 8024–35.
[79] Brigham EO, Morrow RE. The fast Fourier transform. IEEE Spectr 1967;4(12):63–70. http://dx.doi.org/10.1109/MSPEC.1967.5217220.