Universidad de Granada

A PROBABILISTIC FRAMEWORK FOR PROGNOSTICS WITH UNCERTAINTY QUANTIFICATION BASED ON PHYSICS-GUIDED BAYESIAN NEURAL NETWORKS

BY:

Juan Fernández Salas

ADVISOR:

Juan Chiachío Ruano

A THESIS SUBMITTED TO:

University of Granada

Doctoral Programme in Civil Engineering (B23/56/1)
Department of Structural Mechanics and Hydraulic Engineering
University of Granada
Granada (Spain)

March 2023

Editor: Universidad de Granada. Tesis Doctorales
Autor: Juan Fernández Salas
ISBN: 978-84-1117-991-1
URI: https://hdl.handle.net/10481/84438

El doctorando Juan Fernández Salas y el director de la tesis Dr. Juan Chiachío Ruano, garantizamos, al firmar esta tesis doctoral, que el trabajo ha sido realizado por el doctorando bajo la dirección del director de la tesis y hasta donde nuestro conocimiento alcanza, en la realización del trabajo, se han respetado los derechos de otros autores a ser citados, cuando se han utilizado sus resultados o publicaciones.

Director de la Tesis	El Doctorando
Fdo.: JUAN CHIACHIO RUANO	Fdo.: JUAN FERNANDEZ SALAS

Aknowledgements

I would like to thank my supervisor, Juan Chiachío, and the coordinator of the H2020 Marie Curie ENHANCE program, Manuel Chiachío, from the department of Structural Mechanics and Hydraulic Engineering of the University of Granada. I owe them the opportunity to work on such an exciting project, which has doubtlessly widen my horizons in both my career and personal life. Their approach to science and the academic life is truly inspirational, and has had a big impact in my work throughout these years. And my tutor, Rafael Gallego, for welcoming me onto the department and making me feel part of the team. I must show my gratitude to KBR, the System-Wide Safety project, and my colleagues at the Diagnostics and Prognostics Group during my research stay in NASA Ames Research Center, specially Matteo Corbetta and Chetan S. Kulkarni. This collaboration has been an invaluable experience, which I will always remember with pride. Thanks to my colleagues from iPMLab and ENHANCE, for their friendly support during this journey. Also, thanks to Enrique Hernández and Rafael Muñoz for our meaningful conversations and your valuable advice.

This thesis would not have been possible without the funding provided by the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No 859957, or without the support from the University of Granada.

Most importantly, I must thank my wife and my family. The last few years have not been particularly easy, and I would not have been able to reach this point without their unconditional support and understanding.

To my late father, my best mentor.

Summary

Artificial neural networks are a family of algorithms whose principles are inspired by the behaviour of biological neurons in the human brain. They have been very successful in performing a wide variety of tasks, and are making a considerable impact on our daily lives. Moreover, many industries have been reaping the rewards from the use of these technologies for decades. But this is not the case in many areas of engineering, and more specifically, civil/structural engineering, where their application is mainly confined within the research domain. Even though civil engineering is a sector with tight margins where safety is the number one priority, and therefore, the benefits of a successful implementation could be significant. The reasons behind such little interest are diverse, from scarcity of quality data, to general distrust about their potential and applicability in the sector.

Indeed, artificial neural networks are often considered as a black box system, given that they can approximate any function but without providing insights about its structure or form. Besides, they suffer from a series of drawbacks and their predictions are not always correct. Hence, quantifying the uncertainty in the outputs of the neural networks becomes of great importance. The state-of-the-art Bayesian neural networks, such as *Variational Inference*, *Hamiltonian Monte Carlo*, or *Probabilistic Backpropagation*, have largely contributed to cast light on this matter, but their method to quantify the uncertainty may be considered as rigid. This is mainly due to the use of parametric probability models for the likelihood function and/or the weights and bias, but also because of the limitations specific to the backpropagation algorithm.

In this thesis, a novel training algorithm for Bayesian neural networks based on approximate Bayesian computation is proposed, hereafter called BNN by ABC-SS. The weights and bias parameters are trained probabilistically without backpropagation or gradient evaluation, thus issues such as local minima are avoided and the stability of the algorithm is improved. Also, predefined parametric probability models are not used for the weights and bias, but they can adopt any form depending on the training data. As a result, BNN by ABC-SS presents great flexibility to learn from the observed data, and more importantly, to quantify the uncertainty inherent in such data. The output of the Bayesian neural network trained with ABC-SS is a non-parametric probability density function that can be understood as the degree of belief of such output in light of the data available.

As previously mentioned, lack of data is also an important limitation for artificial neural networks, as their training is entirely dependent on them. Furthermore, extrapolation is

outside their capabilities, which means that the predictions made outside the domain of the training data are often random, and should not be trusted in most cases. This problem can be overcome, or at least mitigated, by introducing the knowledge extracted from physics-based models into the neural network architecture. While these hybrid algorithms are becoming more and more popular within the scientific community, they normally insert the physics in the loss function through some known boundary conditions, in the form of partial differential equations. The error is then backpropagated to adjust the weights, forcing them to comply with the given laws of physics. A different approach is followed in this thesis, where the physics are introduced independently in three parts of the neural network, namely the input neurons, the output neurons and the metric function (ρ in ABC-SS), resulting in three different variants which are then trained with ABC-SS. The need for data is reduced and the extrapolation capabilities of the overall model improved notably, especially when the physics are added to the output neurons like an extra bias parameter. Additionally, the use of ABC-SS as learning engine provides stability and a more realistic quantification of the uncertainty, yielding a more reliable algorithm. This is especially interesting in engineering, as it allows us to exploit both the valuable knowledge within the physics-based models and the flexibility of artificial neural networks to capture the nonlinear behaviour often found in real data.

The aforementioned principles lead us to the last stage of this doctoral dissertation, when they are applied to prognostics, an engineering discipline that focuses on predicting how the damage and performance of a system will evolve through time. To that end, the capacity of handling sequential data is of great importance, and that is particularly where recurrent neural networks excel at. In the literature, these data-driven algorithms have also been combined with physics-based models and provided promising results, however, they are specially sensitive to gradient related problems, such as vanishing gradients. More complex architectures like *Long-Short-Term-Memory* have proven to mitigate this issue, but at the expense of increasing the number of parameters and activation functions. In this thesis, a physics-guided recurrent neural network trained with ABC-SS is proposed to make predictions about the future performance of an engineering system based on historical sequential data and physics-based knowledge. The probabilistic nature of the ABC-SS algorithm, along with its flexible quantification of the uncertainty, translates into a reliable algorithm that avoids the issues associated to the evaluation of the gradient and its propagation through time, thus long-term dependencies can be learnt without the need for more complex architectures. Moreover, the combination of physics-based knowledge and Bayesian regularization contributes to an improved extrapolation capacity of the proposed recurrent neural network, which is paramount in multi-step ahead forecasting.

Several case studies are presented to evaluate the performance of the proposed algorithms in different engineering problems, from fatigue in composite materials to displacement and accelerations in concrete structures subjected to seismic loads. The key findings

from those case studies are the realistic quantification of the uncertainty provided by ABC-SS, high accuracy comparable to that of the state-of-the-art neural networks, stability thanks to the absence of gradient evaluation, and the ability to make precise predictions beyond the domain of the training data when combined with physics-based models. Regarding real-world applications, the proposed Bayesian neural networks can be envisaged becoming part of a wider PHM tool, helping to make informed decisions about future maintenance operations based on prognostics about the structural integrity of the system.

Resumen

Las redes neuronales artificiales son una familia de modelos computacionales inspirados en el comportamiento de las neuronas biológicas del cerebro humano. Estos algoritmos han tenido un considerable éxito en diversas aplicaciones y están teniendo un gran impacto en nuestra vida diaria. De hecho, muchas industrias llevan décadas cosechando los frutos del uso de estas tecnologías. Pero ese no es el caso en muchas áreas de la ingeniería y, más específicamente, en la ingeniería civil/estructural, donde su aplicación se limita principalmente al terreno de la investigación. Incluso cuando la ingeniería civil es un sector con márgenes ajustados donde la seguridad es la prioridad número uno y por lo tanto, los beneficios de su implementación podrían ser significativos. Las razones detrás de este limitado interés son diversas, desde la escasez de datos de calidad, hasta la desconfianza generalizada sobre su potencial y aplicabilidad en el sector.

En efecto, las redes neuronales artificiales a menudo se consideran como un sistema de caja negra, dado que pueden aproximar cualquier función pero sin proporcionar información sobre su estructura o forma. Además, adolecen de una serie de inconvenientes y sus predicciones no siempre son correctas. Por lo tanto, la cuantificación de la incertidumbre sobre los resultados proporcionados por las redes neuronales se vuelve de gran importancia. Particularmente, las redes neuronales Bayesianas actuales, como "*Variational Inference*", "*Hamiltonian Montecarlo*" o "*Probabilistic Backpropagation*", han contribuido en gran medida a arrojar luz sobre este asunto, pero su método para cuantificar la incertidumbre puede considerarse como rígido. Esto se debe principalmente al uso de modelos de probabilidad paramétricos para definir la función de densidad de los pesos y sesgos, pero también a otras limitaciones específicas del algoritmo de retropropagación ("*backpropagation*").

En esta tesis se propone un nuevo algoritmo de entrenamiento para redes neuronales Bayesianas basado en computación Bayesiana aproximada, en adelante denominado BNN by ABC-SS. Los pesos y sesgos de la red se entrenan de forma probabilística sin retropropagación ni evaluación del gradiente o derivadas parciales, por lo que se evitan problemas como el estancamiento en mínimos locales y se mejora la estabilidad del algoritmo. Además, no se predefinen modelos de probabilidad paramétricos para la función de densidad de los pesos y sesgos, sino que estas pueden adoptar cualquier forma acorde a los datos de entrenamiento. Como resultado, BNN by ABC-SS presenta una gran flexibilidad para aprender de los datos observados y, lo que es más importante, para cuantificar la incertidumbre presente en dichos datos. Las predicciones de esta red neuronal Bayesiana entrenada con

ABC-SS son funciones de densidad no paramétricas, que pueden entenderse como el grado de creencia en dichas predicciones en base a los datos disponibles.

Como se mencionó anteriormente, la falta de datos también es una limitación importante para las redes neuronales artificiales, ya que su entrenamiento depende completamente de ellos. Además, la extrapolación está fuera de sus capacidades, lo que significa que las predicciones realizadas fuera del dominio de los datos de entrenamiento suelen ser aleatorias y, en la mayoría de los casos, no se debe confiar en ellas. Este problema se puede superar, o al menos mitigar, introduciendo modelos basados en física dentro de la arquitectura de la red neuronal. Si bien estos algoritmos híbridos son cada vez más populares dentro de la comunidad científica, normalmente la física es insertada en la función de coste a través de algunas condiciones de contorno conocidas, en forma de ecuaciones diferenciales parciales. Luego, el error se retropopaga para ajustar los pesos y sesgos, obligándolos a cumplir con las leyes de la física dadas. En esta tesis se sigue un enfoque diferente, donde la física se introduce de forma independiente en tres partes de la red neuronal, a saber, las neuronas de entrada, las neuronas de salida y la función métrica (ρ en ABC-SS), lo que da como resultado tres variantes que son entrenadas con ABC-SS. En consecuencia, la necesidad de datos se reduce y las capacidades de extrapolación del modelo híbrido mejoran notablemente, especialmente cuando la física se agrega a las neuronas de salida como un parámetro de sesgo adicional. Además, el uso de ABC-SS como motor de aprendizaje proporciona estabilidad y una cuantificación más realista de la incertidumbre, lo que genera un algoritmo más fiable. Esto es especialmente interesante en ingeniería, ya que nos permite aprovechar y explotar el valioso conocimiento que existe dentro de los modelos basados en física, así como la flexibilidad de las redes neuronales artificiales para capturar el comportamiento no lineal que a menudo se encuentra en los datos reales.

Los principios antes mencionados nos llevan a la última etapa de esta tesis doctoral, cuando estos son aplicados a la ingeniería de pronóstico, una disciplina que se enfoca en predecir cómo evolucionará el daño y el rendimiento de un sistema a lo largo del tiempo. Para ello, la capacidad de manejar datos secuenciales es de gran importancia, y es ahí donde destacan las redes neuronales recurrentes. En la literatura existente se puede observar como estos algoritmos basados en datos también se combinaron con modelos basados en física. Mientras estos modelos recurrentes híbridos han proporcionado resultados prometedores, también han mostrado ser especialmente sensibles a problemas relacionados con la retropropagación del gradiente, conocidos como "*vanishing gradients*". Arquitecturas más complejas como "*Long-Short-Term-Memory*" han demostrado mitigar este problema, pero a expensas de aumentar la cantidad de parámetros y funciones de activación. En esta tesis, se propone una red neuronal recurrente guiada por física y entrenada con ABC-SS, para hacer predicciones sobre el rendimiento futuro de un sistema de ingeniería basándose en datos secuenciales históricos y modelos físicos. La naturaleza probabilística del entrenamiento Bayesiano ABC-SS, junto con su cuantificación flexible de la incertidumbre, proporciona un algoritmo fiable que evita los problemas asociados con la evaluación del gradiente y

su retropropagación en el tiempo, por lo que las dependencias a largo plazo pueden ser aprendidas sin la necesidad de arquitecturas más complejas. Además, la combinación del conocimiento basado en física y la regularización Bayesiana contribuye a mejorar la capacidad de extrapolación de la red neuronal recurrente propuesta, lo que es fundamental para realizar predicciones sobre un horizonte lejano.

Para evaluar el rendimiento de los algoritmos propuestos en esta tesis se presentan varios casos de estudio con diferentes problemas de ingeniería, desde fatiga en materiales compuestos hasta desplazamientos y aceleraciones en estructuras de hormigón armado sometidas a cargas sísmicas. En todos ellos se observa una cuantificación realista de la incertidumbre proporcionada por ABC-SS, alta precisión comparable a la de las redes neuronales actuales, estabilidad gracias a la ausencia de evaluación del gradiente, y la capacidad de hacer predicciones precisas más allá del dominio de los datos de entrenamiento cuando se combinan con modelos basados en física. Con respecto a las aplicaciones a casos reales, las redes neuronales Bayesianas propuestas se podrían considerar como parte de una herramienta de PHM más amplia, ayudando a tomar decisiones mejor informadas sobre futuras operaciones de mantenimiento, basadas en pronósticos sobre la integridad estructural del sistema.

Acronyms

ABC	Approximate Bayesian Computation
ABC-SS	Approximate Bayesian Computation by Subset Simulation
AI	Artificial intelligence
ANN	Artificial neural network
aPC	Arbitrary polynomial chaos
BBP	Bayes by Backprop
BNN	Bayesian neural network
BNN by ABC-SS	Bayesian neural network trained with ABC-SS
BRNN by ABC-SS	Bayesian recurrent neural network trained with ABC-SS
CFRP	Carbon fiber reinforced polymer
ConvRNN	Convolutional recurrent neural network
CPDF	Cross physics-data fusion
CPU	Central Processing Unit
FFT	Fast Fourier transform
FNN	Feedforward neural network
GPU	Graphics processing unit
GRU	Gated recurrent unit
HMC	Hamiltonian Monte Carlo
IQR	Interquantile range
LSTM	Long short term memory
LSTM by ABC-SS	Long short term memory trained with ABC-SS
MA	Moving average
MC	Monte Carlo method
MCMC	Markov chain Monte Carlo
M-H	Metropolis-Hasting algorithm
MMA	Modified Metropolis algorithm
MPA	Maximum a posteriori
MSE	Mean squared error
NASA	National Aeronautics and Space Administration
NNAP	Neural network augmented physics
PbM	Physics-based model
PBP	Probabilistic backpropagation

PCA	Principal components analysis
PDF	Probability density function
PGNN	Physics-guided neural network
PG -	Physics-guided (followed by the type of ANN)
PHM	Prognostics and health management
PhyCNN	Physics-guided convolutional neural network
PINN	Physics-informed neural network
PZT	Piezoelectric
Q1	First quartile
Q2	Second quartile
Q3	Third quartile
ReLU	Rectified linear unit
RMSE	Root mean squared error
RNN	Recurrent neural network
SHM	Structural health monitoring
SOTA	State-of-the-art
VI	Variational inference

Symbols

Symbol	Description
x	Input variable
$\mathcal{X}$	Subregion of x-inputs
$\hat{y}$	Predicted output
$\mathcal{O}$	Subregion of $\hat{y}$-outputs
y	Observed output
$\mathcal{Y}$	Subregion of y-outputs
$\mathcal{D}$	Data set
w_{ij}	Weights in artificial neural networks
b_j	Bias in artificial neural networks
$g(\cdot), h(\cdot)$	Activation functions in ANN
θ	Model parameter vector
Θ	Set of plausible values of model parameters
$\mathcal{M}$	Model class (in the context of ANN architectures)
$C(\theta)$	Cost/Loss function of ANN defined by parameters θ
$\nabla_\theta C(\theta)$	Gradient of cost function with respect to parameters θ
$P(\cdot)$	Probability
$p(\cdot)$	Probability density function
ϵ_j	Tolerance parameter (metric distance) j-th simulation level
$\rho(\cdot)$	Metric function
$\eta(\cdot)$	Summary statistics
$\mathcal{B}_\epsilon(y)$	Metric ball of radius ϵ centered at y
$\mathcal{S}_j$	Nested regions of possible solutions
P_0	Conditional probability for Subset Simulation
N	Number of samples
ℓ	Simulation levels
σ_j	Standard deviation of proposal function (j-th simulation level)
p	Decrease rate of standard deviation σ in proposal function
M	Weights and bias matrix in BNN by ABC-SS
M_{RNN}	Weights and bias matrix in BRNN by ABC-SS

Symbol	Description
M_{Seeds}	Seed weights and bias matrix in ABC-SS matrix
M_{SubSet}	Weights and bias SubSet matrix in ABC-SS matrix
M_{LSTM}	Weights and bias matrix in LSTM by ABC-SS
ζ	Random noise
lr	Learning rate in standard ANN
$\rho_d(\cdot)$	Data-driven metric
$\rho_p(\cdot)$	Physics-based metric
$\rho_f(\cdot)$	Overall or final metric
y_p	Prediction from physics-based model
α	Weight given to the data-driven approach in PG-BNN by ABC-SS
β	Weight given to the physics-based approach in PG-BNN by ABC-SS
v_0	Initial velocity of projectile
λ_0	Initial angle with respect to the horizontal
$\mathcal{R}$	Projectile range given by physics-based model
d_t	Distance reached by projectile
g	Gravitational acceleration (9.82 m/s^2)
$Loss_d$	Data-driven loss function
$Loss_p$	Physics-based loss function
λ_p	Weight given to physics-based loss function
P_{25}	Percentile 25
P_{50}	Percentile 50
P_{75}	Percentile 75
U	Input weight matrix in RNN
W	Hidden state weight matrix in RNN
V	Output weight matrix in RNN
h_t	Hidden state at time step t in RNN
$\hat{y}_t$	Output at time step t in RNN
i_t	Output of input gate at time step t in LSTM
f_t	Output of forget gate at time step t in LSTM
o_t	Output of the output gate at time step t in LSTM
C_t	Cell state at time step t at time step t in LSTM
$\tilde{C}_t$	Candidate input information at time step t in LSTM

Contents

List of Figures

List of Tables

List of Algorithms

I

INTRODUCTION

"Learning and innovation go hand in hand. The arrogance of success is to think that what you did yesterday will be sufficient for tomorrow"

— WILLIAM POLLARD

1

Status quo and motivation

Artificial Intelligence (AI) has experienced a fast pace development during the last decade and promises large benefits in many fields of different nature. Particularly, artificial neural networks (ANN) have revolutionised the world of machine learning, with more complex models that have made computer vision [4] and speech recognition [5] a reality, in some cases even reaching human accuracy [6]. As a result, we often encounter this technology in our daily life, in the form of email classification [7], fraud prevention [8] or border controls [9], to name but a few examples.

This technology is not limited to the aforementioned applications but it has also been explored in the civil engineering field. The first journal article about the application of ANN to structural/civil engineering dates from 1989, by Adeli and Yeh 1989 [10], and since then, a significant number of articles have been published in a variety of prestigious journals, covering a wide spectrum of disciplines. ANN have been applied to building materials to calculate some of their properties, such as the elastic modulus or ultimate strength [11]; in hydrology for identifying the value of certain aquifer parameters [12]; in geomechanical engineering for the estimation of soil properties such as the consolidation pressure [13]; and even in construction management for cost estimation [14]. However, the application of ANN to a real-world scenario in the civil engineering industry is not common, and often limited to research. The main reason is that ANN are not without drawbacks, which prevent them from being used in critical systems.

Predictions made by the standard ANN are deterministic by nature, providing numerical values or labels as outputs. But those predictions are not always correct, given that

the model only has partial information about the observed reality. Thus, it becomes difficult to determine if the output of the model should be trusted. Therefore, it is clear that ANN are subject to uncertainty, which may be critical in applications where small variations in the predictions might cause disproportionate consequences, such as in safety evaluation of power plants [15] or trajectory and safety assessment in civil aviation [16]. This uncertainty can be classified into two categories, *epistemic* and *aleatory* [17]. In machine learning, epistemic uncertainty mainly refers to the lack of training data in some areas of the input domain, and the choice of hyperparameters that delivers the right balance between model complexity and low generalization error [18]. 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 [19]. In all cases, quantifying the total uncertainty in the predictions provides valuable information [20].

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 [21, 22, 23, 24, 25]. The best known training methods used in BNNs are the Variational Inference method (VI) [26, 27, 28] (more specifically Bayes by Backprop [29, 30]), Probabilistic Backpropagation (PBP) [1] and Hamiltonian Monte Carlo (HMC) [31, 32]. Generally, these methods require the evaluation of the gradient of a cost function using the back-propagation algorithm [33], which is prone to suffer from drawbacks such as *exploding gradient* [34] when large derivatives propagate down the model, *vanishing gradient* [34] if derivatives are small, or *dying ReLU* [35], all of which affect the learning process and the accuracy of the predictions. 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. However, identifying and quantifying such uncertainty is not the only challenge that ANN need to deal with in civil engineering, as we are about to see.

Modern algorithms are capable of learning patterns in complex natural processes without the need to identify and/or understand them, provided that enough data are available [36, 37, 38]. And for that same reason, in those situations where data is scarce or imbalanced [39], their performance may be poor and unreliable. Moreover, machine learning algorithms do not perform well when making predictions about events or processes which are outside the training data space (extrapolating) [40]. Contrariwise, physics-based models approximate reality relatively well and are transparent to human understanding. But despite their successful applications, it is complicated for them to include comprehensive details of real natural phenomena without becoming overly complex themselves and difficult to use, with

practically unidentifiable parameters [41]. There is a wide range of engineering applications where there only exist relatively simple models that partially explain the phenomenon of interest and the availability of data is very limited. Therefore, it seems sensible to seek hybrid models that can benefit from both, physics-based and data-driven approaches.

During the last few years, ANN that include some physics-based knowledge about the process that generated the experimental data in their loss function, such as boundary conditions, have increased in popularity. The way this physics-based knowledge is embedded within the machine learning algorithm is very diverse and depends on the application in hand. One of the most prominent algorithms in this area of research is the so-called *physics-informed neural networks* (PINN) [42], which encourages the ANN to follow certain laws of physics, described by partial differential equations, by increasing the cost of solutions that do not satisfy them. This methodology has also set the foundations for a wide range of ANN algorithms [43] and applications [44, 45, 46, 47, 48, 49]. Another interesting approach to introduce prior domain knowledge in neural networks is by specifying certain constraints, such as logical or algebraic expressions, that should hold over the output space. This method has shown efficiency in computer vision, when mapping from an image to the location of an object it contains [50]. Following the same principles, in the field of mechatronic systems (e.g., presses, pumps, hydraulic valves or compressors) the *neural network augmented physics* (NNAP) [51] algorithm proposes neural layers that are inserted in the physics-based model, with the novelty of simultaneously optimizing both the neural network and physical parameters. Physics-guided neural networks (PGNN) are another family of hybrid methods that are providing promising results. Among the different variants of PGNN that can be found in the literature, Jinjiang Wang et al. [52] proposed a cross physics-data fusion (CPDF) scheme to combine the information obtained by a physics-based model and a data driven model for machining tool wear prediction. Ruiyang Zhang et al. [53] presented the Physics-guided Convolutional Neural Network (PhyCNN) for prediction of building's response subjected to earthquakes. Uduak Inyang-Udoh and Sandipan Mishra [54] developed a physics-guided convolutional recurrent neural network (ConvRNN) for droplet-based additive manufacturing and proved that the data required to train this model are much less compared to a pure black-box model. Anuj Karpatne et al. [55, 56] was probably the first attempt to introduce some physics-based principles within the neural network architecture, achieving low errors in a lake temperature modeling problem. While most of these algorithms are deterministic in nature, there are some hybrid models that are able to quantify the uncertainty in their predictions, using Bayesian methods [57], arbitrary polynomial chaos (aPC) and *Dropout* [58, 59], or Monte Carlo Dropout [60], among others. However, this quantification of the uncertainty could be defined as *rigid* [20], given that the weights and/or the likelihood function of those neural networks are parametric and defined by a pre-shaped likelihood function, typically Gaussian again. In addition, their learning process is based on the evaluation of the gradient of a physics-based loss function via the back-propagation algorithm [33], which

may suffer from problems like Dying ReLU or vanishing/exploding gradient, as explained before.

Data in civil engineering are not only scarce, but in most cases, they appear in a sequential manner. This type of data is also known as time-series and can be found in many engineering problems, such as electricity production forecasting [61] or fatigue prediction [62]. Recurrent neural networks (RNN) have provided an outstanding performance when applied to sequential data. There are many different variants of RNN, from Long Short Term Memory (LSTM) [63] to Gated Recurrent Unit (GRU) [64], and they are responsible for many tools and software, such as speech recognition [65], or sentiment analysis [66]. Moreover, they have been also applied to construction related tasks [67].

Nevertheless, RNN also present some issues, mostly related to the use of the backpropagation algorithm to train the weights of the neural network. 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 [34]. RNN also suffer from data scarcity and poor extrapolation capacity, just like standard ANN.

Different types of hybrid RNN can be found in the literature, which deal with those problems by combining physics-based and data-driven approaches. Depending on the nature of the laws of physics, these are inserted in the RNN in a different manner. For example, Y Chen et al. [68] proposed LSTM neural networks for fault detection in gearboxes. Manu Lahariya et al. [69] integrated physics-based constraints into the training process of the LSTM to identify the flexibility in the evaporative cooling process. Primarily based on the works of Raissi et al. [42], some authors have successfully applied physics-informed RNN to problems where the physics are governed by partial differential equations [70, 71, 72, 73]. In these cases the neural network is forced to minimise a loss function that includes those physics-based differential equations and boundary conditions. In the area of Prognostics and Health Management (PHM), RG Nascimiento et al [74, 75, 76] 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, where the physics are often introduced in the loss function. In this line of research, Xiaowei Jia et al. [77] successfully combined RNN and physics-based models for simulating temperature profiles in lakes. Bayesian methods, such as Monte Carlo Dropout (MC Dropout), have also been combined with physics-guided RNN for quantification of the prediction uncertainty, as can be seen in the works of Arka Daw [60]. 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 [78].

In summary, ANN have demonstrated great potential in a wide variety of applications, but not so much in civil engineering. The main challenges that ANN face in this industry is the quantification of the uncertainty or degree of belief of the predictions, the lack of real and valuable data, and the ability to handle time-series data. There exist different BNN that provide a quantification of the prediction uncertainty, but this is rigid and limited by

predefined likelihood functions (e.g., Gaussian). Hybrid models that could tackle the data scarcity problems have also been developed, but they often ignore the uncertainty and include the physics only in the loss function, through differential equations and boundary conditions. Hybrid RNN have been used to process sequential data, but these algorithms are trained with the backpropagation algorithm, which raises multiple gradient-related problems. Therefore, a new family of models is needed, one able to quantify the uncertainty in a flexible manner without being constrained to predefined probability models for the likelihood function; that includes physics-based models in the forward pass to mitigate data scarcity and improve the extrapolation capacity; and one that can handle sequential data without being subjected to the issues arising from the use of backpropagation.

2
Research objectives

The application of ANN to civil engineering is often limited to research [11, 12, 13], and not commonly found in real world applications. This is mostly due to a series of drawbacks in this technology that may prevent its deployment at the scale of civil infrastructure. While ANN have demonstrated the ability to make accurate predictions in other fields, these are normally deterministic, and do not take into consideration the uncertainty inherent in the observed data. Some methods, such as the state-of-the-art BNN, have already addressed this problem [29, 1, 31], but they are limited by predefined likelihood functions, which results in a constrained quantification of the uncertainty [20]. The scarcity of quality data is another challenge that the civil engineering industry faces, and that is a major limitation for the application of ANN. Hybrids methods that fuse physics and data-driven algorithms have been developed to mitigate this situation, however, they normally use partial differential equations which are embedded in the loss functions [42], whilst again ignoring the uncertainty in the data. And even if quality data is available, this is normally sequential in nature and needs to be processed by other types of neural networks, such as RNN. But these are not without drawbacks either, as they use the backpropagation algorithm to train the weights, which is prone to gradient-related problems [34, 34, 35]. Therefore, the main goal of the research presented in this thesis is to develop a new methodology to train neural networks that is suitable for civil and structural engineering problems, can quantify the uncertainty, performs well in conditions where data is scarce, and is not subject to gradient-related problems when handling sequential data. To that end, several hypothesis are formulated below, leading to more specific objectives:

1. Despite the numerous applications of standard ANN, including in civil engineering, most of them are deterministic and do not quantify the uncertainty in their predictions. This problem has been addressed by BNN, which produce probabilistic outputs, or in other words, a range of plausible prediction outputs. While they use probability density functions to define their parameters (weights and bias) and likelihood functions, these are predefined by the user, normally Gaussian. This translates into a rigid quantification of the uncertainty [20], given that both the parameters and likelihood functions are being forced to follow a specific shape.
   Hypothesis 1: The value of the weights and bias of ANN could be optimised using Approximate Bayesian Computation by Subset-Simulation (ABC-SS).
   ⇒**Research objective 1:** *Develop a new training method for neural networks based on ABC-SS, where the likelihood function, and the parameters do not need to follow a predefined shape.*

2. Quality data are a rare sight in civil engineering applications [79, 80]. The reasons behind this situation are diverse: the life span of infrastructure assets is too long to have records about their whole life cycle, inspections and maintenance operations are not always recorded in a usable manner for data-driven algorithms, the digitalization of structural elements through sensorization is just beginning to occur, and lack of knowledge between fellow civil engineers about the potentials of data-driven methods, just to name but a few. And when enough data is not available, then the algorithms need to be able to extrapolate beyond the domain of the training data, which is something that standard ANN are not good at [40]. However, hybrid methods that include physics in their algorithm have demonstrated good performance in situations where data is scarce. Even so, they also present some issues and limitations, as they normally work with differential equations inside the loss function [42], and ignore the quantification of the uncertainty. In civil engineering, a great part of the knowledge gathered along centuries is in the form of mathematical or empirical formulations, and the uncertainty plays a major role given the serious consequences of miscalculations. At the moment, the uncertainty in physics-based models is often mitigated through safety coefficients.
   Hypothesis 2: The limitations related to data scarcity, extrapolation and quantification of the uncertainty could be mitigated if physics-based models are included in the forward pass of the neural network in the form of mathematical formulations, and Bayesian training is used.
   ⇒**Research objective 2:** *Propose different ways of introducing the physics-based models in the forward pass of the neural network, and use ABC-SS as Bayesian training algorithm.*

3. When the records of a data set are stored consecutively, and there exists a dependence between consecutive points, this data set is considered to be sequential. This type of data is common in civil engineering, and can be found in fatigue damage, degradation processes, seismic events, and so forth [80]. Recurrent neural networks (RNN)

are a special type of ANN, which can analyse the temporal dependencies between the data points, and find patterns in them [81, 82, 83]. Moreover, they provide the best performance when making predictions about future events based on historical data. But they are also subject to problems such as the ones mentioned before about data scarcity and the quantification of the uncertainty, although the same proposed solutions would apply to them. However, this type of neural networks are very sensitive to gradient-related problems, given that the derivatives need to be propagated backwards through many time steps [78]. While this has been mitigated through other types of RNN that include complex gates inside the recurrent cell, such as Long Short Term Memory (LSTM) neural networks [63], they comprise a larger number of parameters and the backpropagation of the gradients may still cause instabilities due to finding a different local minima of the loss function on each run of the algorithm, reducing the reliability of the model.

Hypothesis 3: A RNN that includes physics-based models in its forward pass and is trained using Bayesian methods can avoid the aforementioned limitations and may provide a useful methodology for prognostics. Additionally, gated units could be avoided, reducing the number of weights and bias required.

⇒**Research objective 3:** *Propose a RNN that includes physics-based models in their forward pass, and train them with ABC-SS.*

3

Completion of objectives

The main research objectives of this thesis have been presented previously in Chapter 2. The work undertaken towards the completion of such objectives, including the methodologies developed, and the experiments carried out to demonstrate such methodologies, are briefly outlined in this chapter. Also, references to where they can be found in this document are included in the text below. Most of this information has already been published in journals or conference proceedings, which are listed in Appendix A.

Research objective 1:

Develop a new training method for neural networks based on ABC-SS, where the likelihood function, and the parameters do not need to follow a predefined shape.

The Bayesian inference algorithm ABC-SS was published in 2014 [84], and since then it has been applied to a wide range of Bayesian inference problems [85]. However, this algorithm has never been used before for training neural networks, thus a number of modifications are required to achieve such purpose. The adjustments made to ABC-SS to transform it into a training algorithm for neural networks can be found in Chapter 5. The result is a Bayesian gradient-free training method, where the weights are not defined by parametric functions, and the outputs reflect the uncertainty in the observed data. This new methodology, henceforth called BNN by ABC-SS, has been applied to two illustrative problems, also shown in Chapter 5 (Sections 5.2 and 5.3), where it can be seen that the weights are defined by free-form probability density functions, and the uncertainty in the predictions is quantified faithful to the observed reality. The proposed BNN by ABC-SS has also been applied to a real case study about fatigue in

composite materials. The propagation of damage in this type of materials is not yet well understood, given the complexity of the process with several modes of failure interacting among them [86], and therefore, the quantification of the uncertainty could be critical [87]. The performance of BNN by ABC-SS in this case study has been compared against the state-of-the-art BNN, including Hamiltonian Mote Carlo [31, 88], Variational Inference (Bayes by backprop) [29] and Probabilistic Backpropagation [1]. The results and their discussion can be found in Chapter 9.

Research objective 2:

Propose different ways of introducing the physics-based models in the forward pass of the neural network, and use ABC-SS as Bayesian training algorithm.

The so-called physics-informed neural networks normally include the physics in the loss function, through differential equations and boundary conditions [42]. However, the objective of this thesis is to introduce physics-based models, represented by mathematical formulations, in the forward pass of the neural network. Thus, the physics are not only used during the training phase, but also afterwards when making predictions in a real-world scenario. Three variants have been proposed in Chapter 6, introducing the physics-based models in the metric function of ABC-SS, through the input neurons, and through the output neurons. All three variants exhibit a different behaviour, from regularization properties to an improved extrapolation capability. This new family of algorithms, from now on called physics-guided Bayesian neural network by ABC-SS (PG-BNN by ABC-SS), fuse the knowledge embedded in physics-based models, with the non-linearity of ANN and the flexibility of ABC-SS to quantify the uncertainty. An illustrative problem is presented in Chapter 6 (Section 6.2), where a set of synthetic data about projectile motion under different wind conditions is created in a simplistic manner, so the concepts can be easily understood. An engineering case study is also provided in Chapter 10, where PG-BNN by ABC-SS is applied to experimental data from a reinforced concrete column subjected to lateral forces. The performance of the proposed algorithms are compared against their data-driven counterparts and the state-of-the-art physics-guided neural networks.

Research objective 3:

Propose a RNN that includes physics-based models in their forward pass, and train them with ABC-SS.

RNN have excelled at handling sequential data, but like standard neural networks, they cannot quantify the uncertainty in their predictions or extrapolate outside the domain of the training data [40]. Moreover, they are specially sensitive to gradient-related problems [78]. By applying the concepts explained in research objectives 1 and 2 to RNN, a new type of physics-guided Bayesian recurrent neural network (PG-BRNN by ABC-SS) has been developed. The training algorithm of BNN by ABC-SS has

been adapted to accommodate the weights and bias matrices found in standard RNN, and the physics-based model has been included in the recursive forward pass. This is explained in Chapter 7, specifically in Algorithm 3. Thanks to ABC-SS, the hidden state of each time-step does not need to be stored, nor their gradient backpropagated through time. As a result, complex gates inside the recurrent cell are also avoided. This new algorithm has been applied to a case study about accelerations in concrete buildings during seismic events, and its performance compared against the state-of-the-art RNN, such as LSTM [63], Gated Recurrent Unit (GRU) [89], Bidirectional LSTM [90] and Monte Carlo Dropout LSTM [91]. The results are presented in Chapter 11.

4

Theorical fundamentals

This chapter aims at providing the theoretical framework that supports the methodologies proposed in this thesis. From Section 4.1 to Section 4.5, the following concepts are explained: ANN and their formulation, BNN and their probabilistic nature, ABC-SS as inference engine, the combination of data-driven approaches and physics-based models into hybrid methods, and finally, the principles of RNN.

4.1 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 [92]. 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 [93]. 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. Figure 4.1 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 Figure 4.1, also known as forward propagation, is formulated as follows:

$$\hat{y}_k = f(x; w, b) = g(\sum_{j=1}^{m} w_{jk}^{(2)} h(\sum_{i=1}^{n} w_{ij}^{(1)} x_i + b_j^{(1)}) + b_k^{(2)}) \tag{4.1}$$

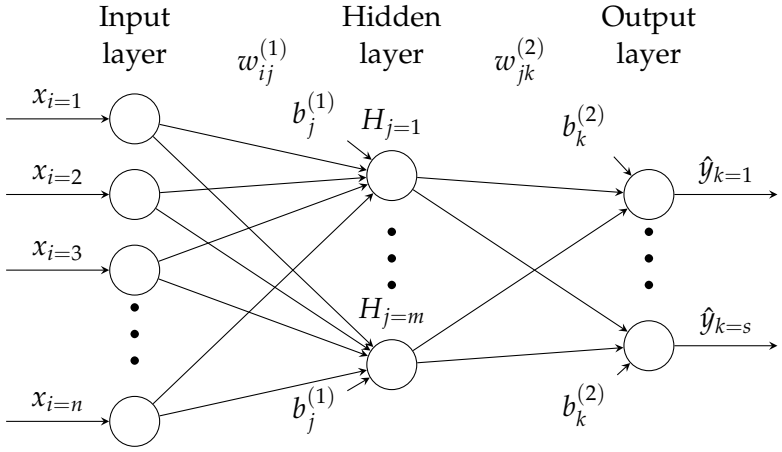


Figure 4.1: Generic example of a basic FNN

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 [94] 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 [33].

4.2 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 4.1, 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 [95, 96, 97, 98].

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; and 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:

$$p\left(\theta|\mathcal{D}, \mathcal{M}\right) = \frac{p\left(\mathcal{D}|\theta, \mathcal{M}\right) p\left(\theta|\mathcal{M}\right)}{p\left(\mathcal{D}|\mathcal{M}\right)} \tag{4.2}$$

where $p\left(\theta|\mathcal{D}, \mathcal{M}\right)$ is the posterior PDF of the model parameters given the data, and $p\left(\mathcal{D}|\theta, \mathcal{M}\right)$ 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 $p\left(\theta|\mathcal{M}\right)$ 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 $p\left(\mathcal{D}|\mathcal{M}\right)$ is known as the *evidence* and represents how likely the data set $\mathcal{D}$ 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) [99, 100] 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.

4.3 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 [101]. 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, Equation (4.2) can be adapted when applied to the pair $(\theta, \hat{y}) \in \Theta \times \mathcal{O} \subset \mathbb{R}^{d+l}$ as follows:

$$p(\theta, \hat{y}|\mathcal{D}) \propto p(\mathcal{D}|\hat{y}, \theta)\, p(\hat{y}|\theta)\, p(\theta) \tag{4.3}$$

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 [102]. 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 $\eta(\cdot)$ [103] which, if required, allows the comparison between both vectors in a weak manner. Thus, under the ABC perspective, Equation (4.3) can be rewritten as [84]:

$$p_\epsilon(\theta, \hat{y}|\mathcal{D}) \propto P(\hat{y} \in \mathcal{B}_\epsilon(y)|\theta)\, p(\hat{y}|\theta) p(\theta) \tag{4.4}$$

where $P(\hat{y} \in \mathcal{B}_\epsilon(y)|\theta)$ 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:

$$p_\epsilon(\theta|\mathcal{D}) \propto P(\hat{y} \in \mathcal{B}_\epsilon(y)|\theta)\, p(\theta) \tag{4.5}$$

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, several techniques have been proposed to address this trade-off by combining the ABC principles with sampling methods like Markov Chain Monte Carlo [104], Parallel Tempering [105], or Population Monte Carlo [106]. 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 [107]. Among those, the so-called Approximate Bayesian Computation by Subset Simulation algorithm, namely ABC-SubSim (ABC-SS) algorithm [84], has been 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 [108] and Pi4U [109].

ABC-SS exploits the ABC principles with the Subset Simulation method [110] which transforms a rare event simulation problem into a sequence of simulations with larger probabilities, resulting in a reduction of the computational cost [111, 112]. 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:

$$\mathcal{S}_j = \{(\theta, \hat{y}) : \rho(\eta(\hat{y}, \eta(y)) \leqslant \epsilon_j\}, \text{ and } \epsilon_{j+1} < \epsilon_j \quad \forall j = 1, \ldots, j \tag{4.6}$$

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:

$$P(\mathcal{S}) = P(\mathcal{S}_1) \prod_{j=2}^{\ell} P\left(\mathcal{S}_j | \mathcal{S}_{j-1}\right) \tag{4.7}$$

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.

4.4 Physcics-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 data. However, despite their good performance in many fields, it is fair to say that they are just a representation of a hypothesised 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 [113], reaching computational inefficiency and becoming unsuitable for large-scale real-time prediction activities. In addition, the amount

and frequency of data collection nowadays is beyond the processing capabilities of most 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 sensible to make use of both approaches and create hybrid models [114, 115, 116] 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)* [42], *Neural Networks Augmented Physics Models (NNAP)* [51] or *Physics-Guided Neural Networks (PGNN)* [117]. The way physics and ANN are combined mostly depends on the physics-based models and their formulation, as they can be introduced in different parts of the hybrid model, such as the loss function or the forward pass. In this thesis 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. Figure 4.2 illustrates the concept of physics-guided ANN.

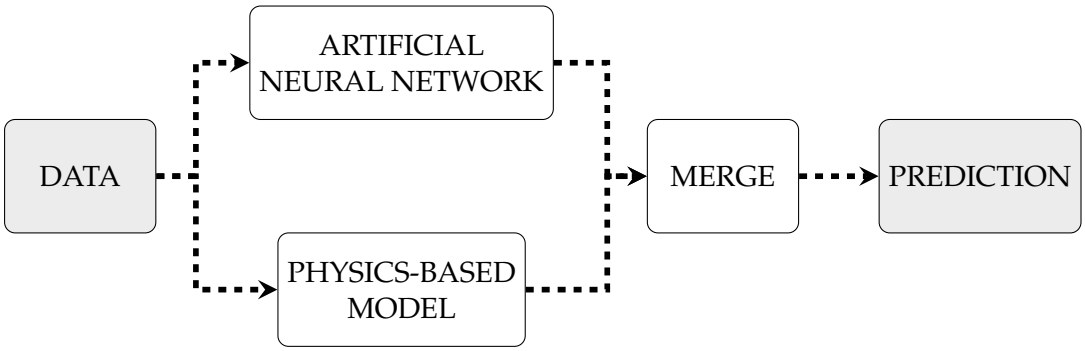


Figure 4.2: Generic diagram of physics-guided artificial neural networks

4.5 Recurrent Neural Networks

The concept of RNN was born in the 1980s [81, 82, 83], 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 Figure 4.3, and the formulation of the forward pass for the most basic form of RNN is shown in Equation 4.8 (note that *tanh* and *sigmoid* are interchangeable by any other activation function, and b and c are bias parameters).

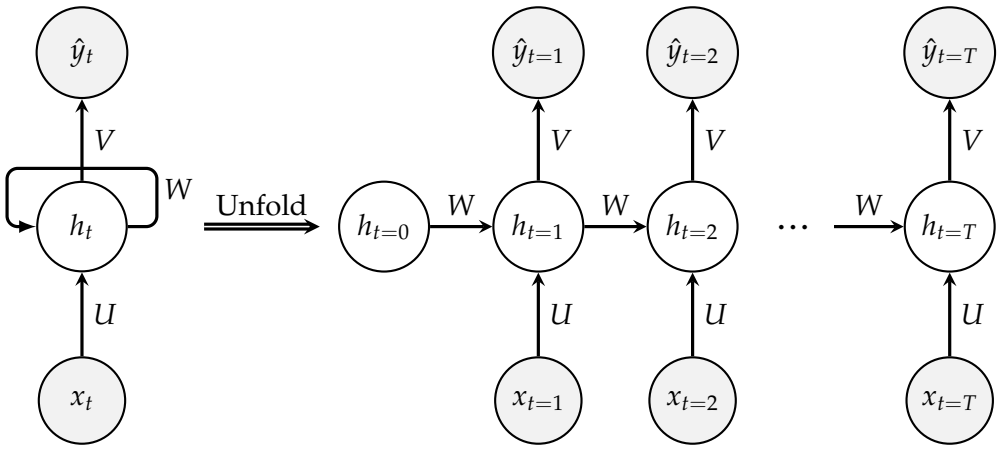


Figure 4.3: Schematic representation of the folded and unfolded RNN

$$
\begin{aligned}
a_t &= b + Wh_{t-1} + Ux_t \\
h_t &= tanh(a_t) \\
o_t &= c + Vh_t \\
\hat{y}_t &= sigmoid(o_t)
\end{aligned}
\tag{4.8}
$$

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* [118]. 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 [90], Gated Recurrent Unit (GRU) [89] or Long Short Term Memory (LSTM) [63, 64]. The interested reader is referred to [118] (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 [82], 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 structural health monitoring (SHM) and prognostics and health management (PHM) [119], 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, Chapter 7 will present a different approach to avoid such drawbacks.

4.6 Long Short Term Memory

As mentioned in Section 4.5, LSTM RNN have solved, or at least mitigated, some of the difficulties found in the successful implementation of vanilla RNN, specially those related to the backpropagation of the gradient. Hence, they became the golden standard of RNN in both, research and industry. Moreover, tasks like speech recognition and object detection/classification were enabled for use in real case scenarios only after the introduction of this methodology.

In theory, vanilla RNN are able to learn long-term dependencies between input and output information, even when there exists a significant number of time steps between them. However, this is not always true in practise, as demonstrated by Sepp Hochreiter in his thesis [120], where he studied in depth the vanishing gradient problem in artificial neural networks and found solid theoretical reasons. In 1995 they released a technical report, to then publish the first paper about LSTM in 1997 [63]. Many different variants and upgrades have been published since then, but they are all based on the original principles.

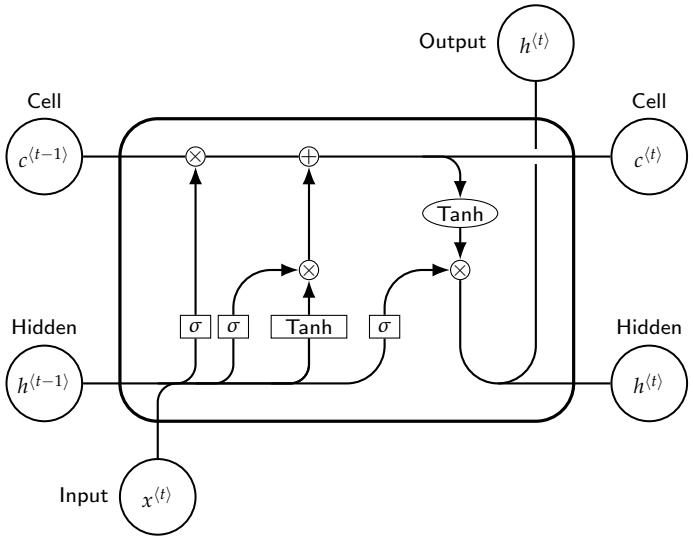


Figure 4.4: Generic diagram of Long Short Term Memory neural networks (by J. Leon).

LSTM RNN are designed to store those long-term dependencies between the input and output information. This is achieved thanks to the *cell state*, which passes the information through the different time steps with minimal interaction. The addition or removal of information from the *cell state* is managed by a series of gates. The piece of information to be discarded is decided by the *forget gate*, where a sigmoid function acts as an outlet, regulating what goes through and what goes away depending on its value (0 to 1). The *input gate* controls what new information is stored in the *cell state*, again using a sigmoid function, but also a tangent hyperbolic function. Then, the old and new information are combined into the next *cell state* using the *update gate*. Finally, the output information for each time step is obtained by passing the new *cell state* through a tangent hyperbolic function, to keep values

between -1 and 1, and then multiplying it by the output of the input gate. Figure 4.4 shows a schematic representation of the process inside an LSTM cell.

$$
\begin{aligned}
i_t &= \sigma(x_t U^i + h_{t-1} W^i) \\
f_t &= \sigma(x_t U^f + h_{t-1} W^f) \\
o_t &= \sigma(x_t U^o + h_{t-1} W^o) \\
\tilde{C}_t &= \tanh(x_t U^g + h_{t-1} W^g) \\
C_t &= \sigma(f_t C_{t-1} + i_t \tilde{C}_t) \\
h_t &= \tanh(C_t) o_t
\end{aligned}
\tag{4.9}
$$

The formulation for one time-step can be found in Equation 4.9, where C represents the *cell state*, U and W are the weights matrices, h are the hidden state for each time step, f is the output of the *forget gate*, i is the output of the *input gate*, $\tilde{C}$ represents the candidate input information and o regulates the output of the overall cell.

II

CONTRIBUTIONS

"Nothing in life is to be feared, it is only to be understood. Now is the time to understand more, so that we may fear less."

— MARIE CURIE

5

BNN by ABC-SS

In this Chapter, the ABC-SS algorithm is adapted to train the weights and bias parameters in ANN. Following the principles explained in Chapter 4 (Section 4.3), the forward model class $\mathcal{M}$ now represents the architecture of the ANN, and $\theta = \{w, b\}$. While the state-of-the-art BNN update their parameters based on the evaluation of the gradient of a cost function $\nabla_\theta C(\theta)$ using the backpropagation algorithm, ABC-SS aims at obtaining the posterior distribution function of the parameters θ through statistical simulation. Thus, 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 methodology, including a chart and a pseudocode, is explained in Section 5.1, and the capabilities of this training algorithm are illustrated in Sections 5.2 and 5.3 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.

5.1 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 $\left({}^1/_{P_0} - 1\right)$ new samples from each seed within the region $\mathcal{S}_1$, until the total population in $\mathcal{S}_1$ reaches N samples. The generation of samples is done by the

Modified Metropolis Algorithm (MMA) [110, 121], 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 $p\left((\theta, \hat{y})|\mathcal{S}\right)$ is defined. Note that the final subset $\mathcal{S}$ constitute a set of N parameter configurations $\theta_{\mathcal{S}}^{(1)}, \theta_{\mathcal{S}}^{(2)}, ..., \theta_{\mathcal{S}}^{(n)}, ..., \theta_{\mathcal{S}}^{(N)}$, 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 marginalised posterior $p_\epsilon(\theta|y)$ whose information is used to produce robust predictions and quantify their uncertainty. A conceptual scheme is provided in Figure 5.1 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, it should be noted that special care is 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 $\mathcal{S}_j$. Recommendations for the selection of those values can be found in [84], while some advances in the scaling of the Subset Simulation algorithm is covered in [121].

5.2 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 ℓ=6. Mean Squared Error (MSE) has been used as the 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.

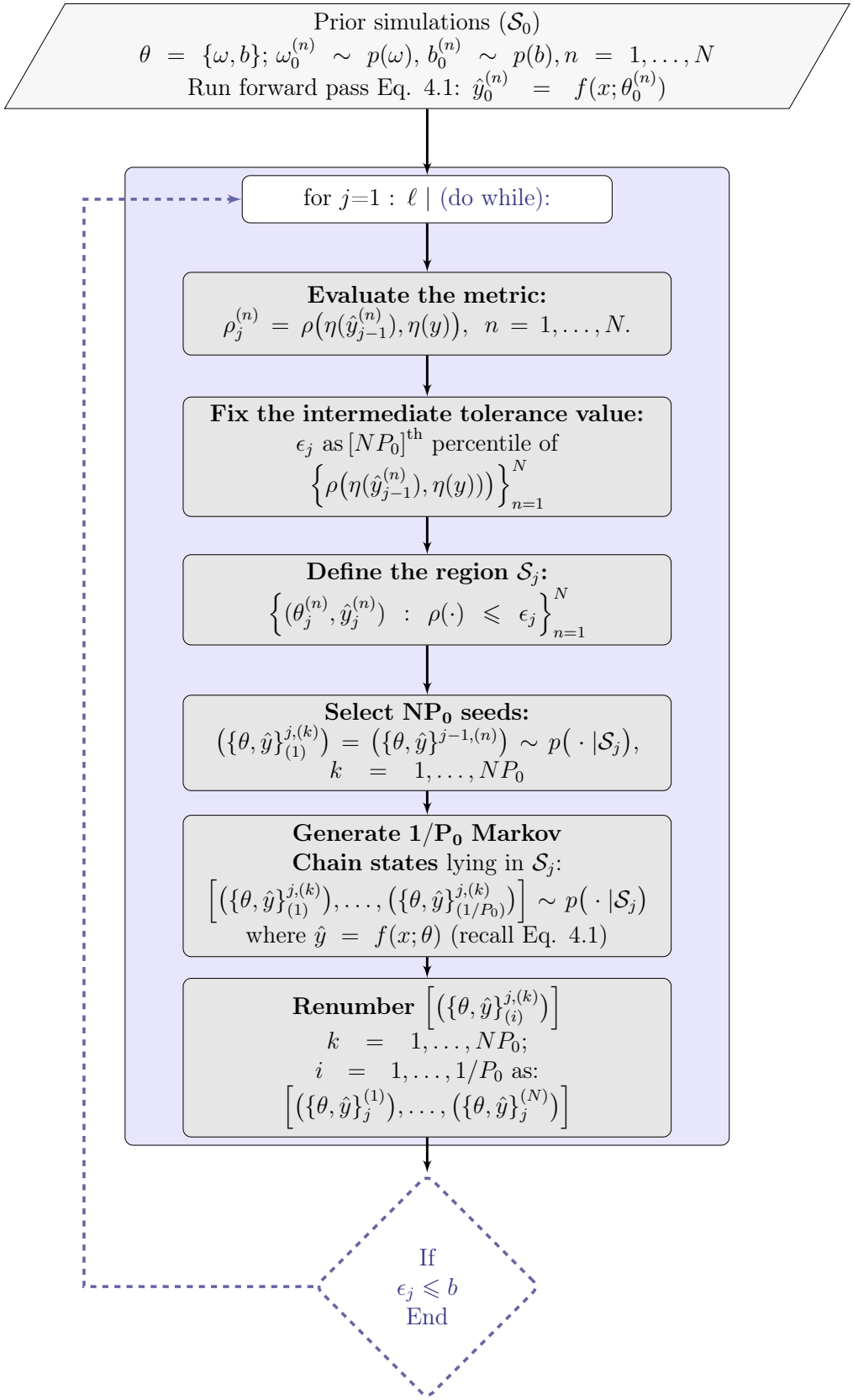


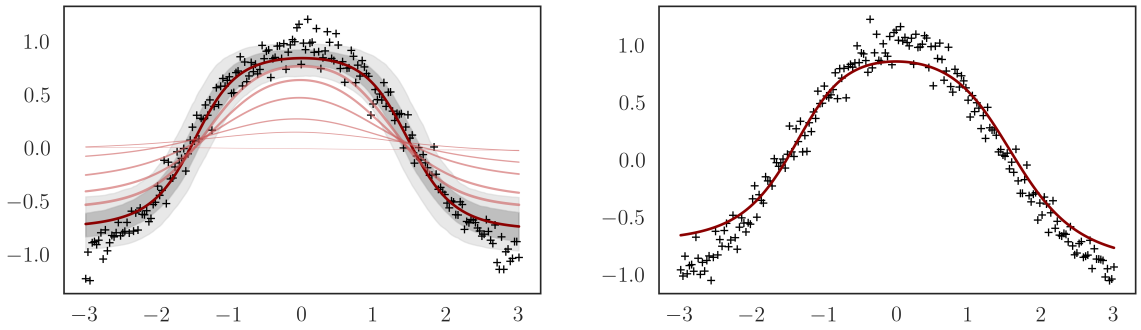
Figure 5.1: Conceptual scheme of the main steps of the BNN by ABC-SS method. The steps corresponding to the while-loop option are depicted using dashed-blue line.

Algorithm 1 *BNN by ABC-SS*

Inputs:
FNN architecture and activation functions
$P_0 \in [0, 1]$
ℓ {number of simulation levels}
N {samples per simulation level}
$\rho()$ {metric/cost function, such as MSE}
$\eta()$ {summary statistic, such as median}
$\sigma_0 \leftarrow (\ell + 1)0.1$
Begin:
for $n : 1, ..., N$ **do**
  Sample initial parameters $\theta_0^{(n)}$, where θ_0 includes all weights w and bias b of the BNN, from priors $p(w)$ and $p(b)$, such as $\mathcal{N}(0, I)$
  $\hat{y}_0^{(n)} \leftarrow$ Use Equation (4.1) to run a forward pass with parameters $\theta_0^{(n)}$ and input x from D
  $\rho_0^{(n)} \leftarrow \rho(\eta(\hat{y}_0^{(n)}), \eta(y))$
  Set $M = \{\theta_0^{(n)}, \hat{y}_0^{(n)}, \rho_0^{(n)}\}_{n=1}^{N}$
end for
for $j : 1, ..., \ell$ **do**
  Renumber $[\theta_{j-1}^{(n)}, n : 1, \ldots, N]$ so that $\rho_{j-1}^{(1)} \leqslant \ldots \leqslant \rho_{j-1}^{(n)} \leqslant \ldots \leqslant \rho_{j-1}^{(N)}$
  $\epsilon_j \leftarrow \rho_{j-1}^{NP_0}$
  $\sigma_j \leftarrow \sigma_0 - 0.1\ell$ {proposed standard deviation decreases in each simulation level}
  $C \leftarrow 1$ {set counter to 1}
  for $i : 1, \ldots, NP_0$ **do**
    $\theta_j^{(i)} \leftarrow \theta_{j-1}^{(i)}$ {select seeds}
    $\rho_j^{(i)} \leftarrow \rho_{j-1}^{(i)}$
  end for
  for $k : 1, \ldots, NP_0$ **do**
    $\mu \leftarrow \theta_j^{(k)}$
    for $i : 1, \ldots, ({}^1/_{P_0}) - 1$ **do**
      $\theta^* \sim \mathcal{N}(\mu, \sigma_j)$
      $\hat{y}^* \leftarrow$ Use Equation (4.1) to run a forward pass with parameters θ^*
      $\rho^* \leftarrow \rho(\eta(\hat{y}^*), \eta(y))$
      if $\rho^* \leqslant \epsilon_j$ **then**
        $\theta_j^{(NP_0+C)} \leftarrow \theta^*$, and $\rho_j^{(NP_0+C)} \leftarrow \rho^*$
        $\mu \leftarrow \theta^*$
      else
        $\theta_j^{(NP_0+C)} \leftarrow \theta_j^k$, and $\rho_j^{(NP_0+C)} \leftarrow \rho_j^k$
      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**

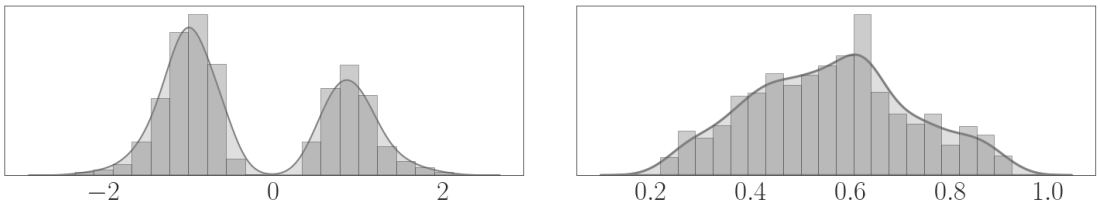
As can be seen in Figure 5.2, both algorithms have obtained similar predictions, however, BNN by ABC-SS consistently reached similar outcomes while those from the FNN, trained with ordinary gradient descent, experience more variability between different runs of the algorithm. This suggests that BNN by 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 Figure 5.3. Of special interest is Figure 5.3(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 Figure 5.2(a) shows the learning process throughout the intermediate simulation levels.



(a) BNN by ABC-SS

(b) FNN trained with Batch Gradient Descent

Figure 5.2: BNN by ABC-SS (left) and Batch Gradient Descent (right), 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.



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

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

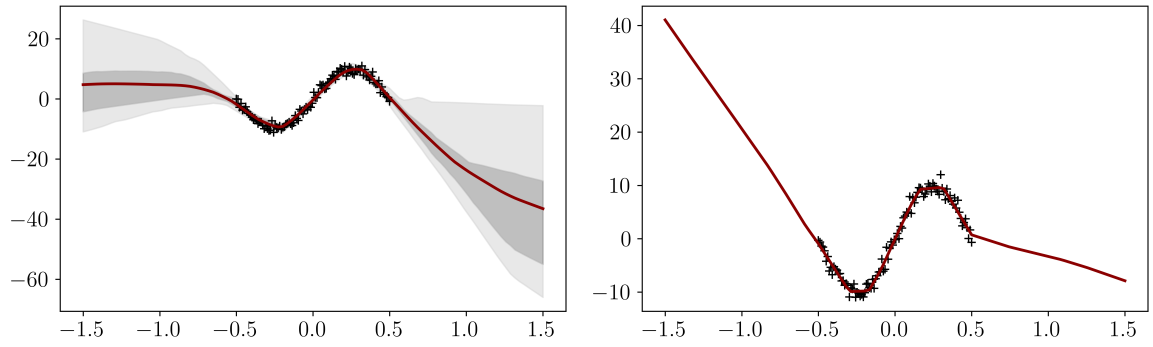
Figure 5.3: BNN by ABC-SS, illustration of the uncertainty about the trained parameters (left) and predictions (right), Illustrative Problem 1.

5.3 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 ℓ=8. Again, the MSE function has been used for the metric ρ.

As shown in Figures 5.4(a) and 5.5(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, Figure 5.4(b), is equally confident both inside and outside the domain of the training data.

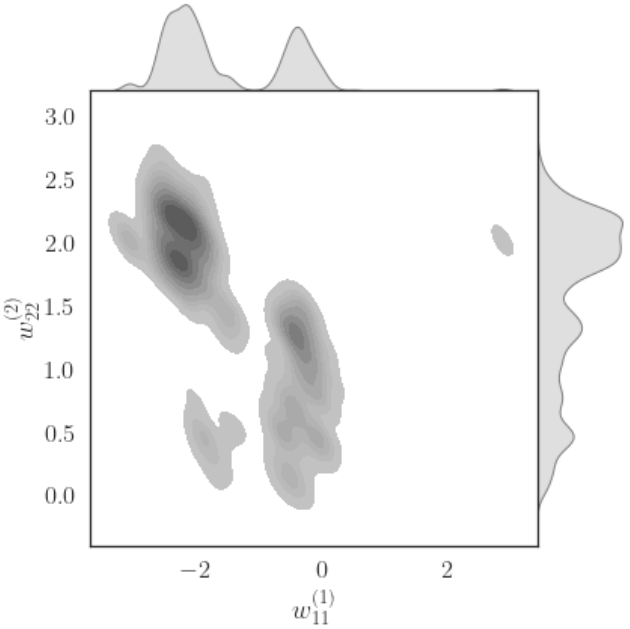
Regarding the posterior PDF of the parameters, Figure 5.5(a) shows a more complex function, which varies between different runs of the BNN by ABC-SS algorithm. This confirms that, given a tolerance value ϵ, the number and diversity of valid sets of parameters $\theta = \{w, b\} \in \Theta \subseteq \mathbb{R}^d$, under tolerance ϵ, increases with the dimension d of the parameter space. Figure 5.5(a) also shows a statistical correlation in the weights which is taken into account by the proposed algorithm when making predictions.



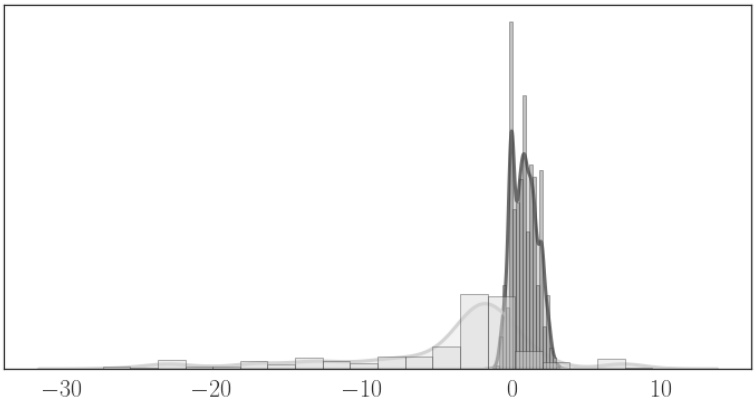
(a) BNN by ABC-SS

(b) FNN trained with Batch Gradient Descent

Figure 5.4: BNNs by ABC-SS (left) and Batch Gradient Descent (right), 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.



(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.

Figure 5.5: Illustrations of the uncertainty about the trained parameters (a), and predictions made by the BNN trained with ABC-SS (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.

6
PG-BNN by ABC-SS

This chapter aims at providing different ways of introducing physics-based models into the forward pass of ANN. Building upon BNN by ABC-SS explained in Chapter 5, a new physics-guided Bayesian neural network is proposed in Section 6.1. This new algorithm benefits from both, the advantages of ABC-SS as inference engine, and the extrapolation capabilities of physics-based models when data is scarce. An illustrative example is also included in Section 6.2 to ease the understanding of the concepts.

6.1 BNN by ABC-SS guided by physics-based models

Three different methods are proposed to combine physics-based models with BNN by ABC-SS to obtain hybrid Bayesian neural networks, the so-called PG-BNN by ABC-SS. Details of the implementation, changes to the original BNN by ABC-SS algorithm and a description of the expected behaviour of the hybrid models are provided in Sections 6.1.1, 6.1.2 and 6.1.3. The proposed neural network architectures are also shown graphically so they can be compared against the standard ANN shown in Chapter 4 (Figure 4.1 and Equation 4.1). It should be noted that, depending on the nature of the problem to be solved, the physics-based models described below may be substituted by any model $M(x) = \hat{y}$, where M represents the model class, x is the input information and $\hat{y}$ the output of the model.

6.1.1 Physics learnt through the metric function

ANN base their training and updating of parameters on a loss function, while BNN often use a predefined likelihood function chosen by the user. BNN by ABC-SS lacks both of them, and instead approximates the likelihood function to $P\left(\hat{y} \in \mathcal{B}_\epsilon(y)|\theta\right)$, which can be interpreted as the probability of $\hat{y}$ to fall within a region $\mathcal{B}_\epsilon(y) = \{\hat{y} \in \mathcal{O} : \rho(\eta(\hat{y}), \eta(y)) \leqslant$

$\epsilon\}$, where a metric function $\rho(\cdot)$ assesses the distance between prediction $\hat{y}$ and data $y \in \mathcal{D}(x, y)$, based on a summary statistics $\eta(\cdot)$ chosen by the user. Thus, a set of parameters $\theta = \{w, b\}$ will be accepted only if $\rho(\eta(\hat{y}), \eta(y)) \leqslant \epsilon$, in other words, only if prediction $\hat{y}$ is close enough to the data y. As mentioned in Section Chapter 5, this lack of likelihood function provides an enhanced flexibility, which results in a fairer representation of the uncertainty.

In this thesis, a new metric function ρ_p based on the laws of physics, is proposed in addition to the current data-driven metric. This will ensure that during training a set of parameters θ is accepted only if, given an input $x \in \mathcal{D}(x, y)$, the prediction $\hat{y}$ is also close enough to the prediction y_p made by the physics-based model. Moreover, two hyperparameters α and β are included in the final metric function, so the user can control how much weight is given to the physics-based model and to the data-driven approach. Algorithm 2 shows the necessary adaptation of BNN by ABC-SS, and Figure 6.1 a schematic representation of the concept.

Algorithm 2 *Physics learnt through the metric function*

1: Every time $\rho()$ needs to be calculated in Algorithm 1 of Chapter 5, replace it as follows:
2: **New terms:**
3: $\rho_d()$ {data-driven metric}
4: $\rho_p()$ {physics-based metric}
5: $\rho_f()$ {overall or final metric}
6: y_d {data y from training data $\mathcal{D}(x, y)$}
7: y_p {prediction from physics-based model}
8: α {weight given to the data-driven approach}
9: β {weight given to the physics-based approach}
10: **Begin:**
11: $\rho_d \leftarrow \rho(\eta(\hat{y}), \eta(y_d))$
12: $\rho_p \leftarrow \rho(\eta(\hat{y}), \eta(y_p))$
13: $\rho_f \leftarrow \alpha\rho_d + \beta\rho_p$

The proposed physics-based metric is expected to provide a regularization effect, which will be added to the natural regularization of BNN thanks to the prior information. This may translate into better generalization, helping to avoid overfitting. Extrapolation might also improve slightly in some cases as the BNN is encouraged to comply with the actual physics, however, since the physics-based model is not implicitly included in the forward pass, accurate extrapolation is not anticipated nor should it be considered as a goal. The uncertainty quantified by the proposed algorithm might also be reduced, given that the BNN now has more information and will discard those data points that depart significantly from the laws of physics.

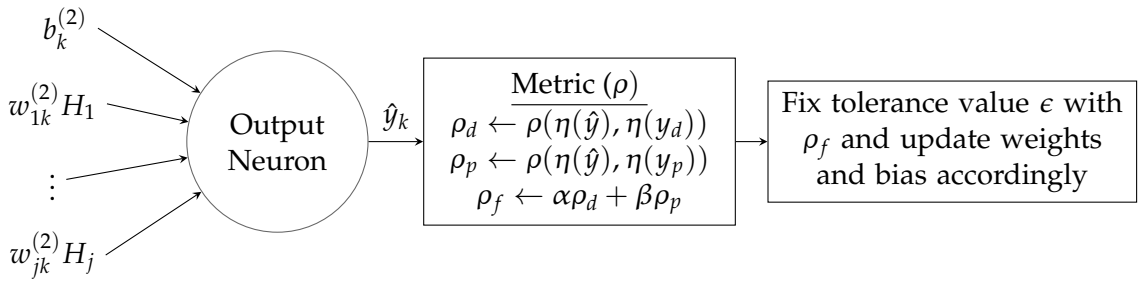


Figure 6.1: Schematic representation of proposed PG-BNN by ABC-SS

6.1.2 Physics learnt through input neurons

Principal Components Analysis (PCA) is an interesting method to reduce the dimensionality of big data sets where there exists a large number of variables [122]. Briefly, PCA aims to find any existing relationship between variables, and then uses linear combinations representing such relationships as new variables. Although reducing the dimensionality of data sets is not the objective of this thesis, the conceptual idea of PCA about identifying correlations between variables and use them as new inputs could serve as an analogy and inspiration for introducing the laws of physics in neural networks. After all, in engineering applications the input and output variables of the neural network are related by the laws of physics. Figure 6.2 shows how physics could be embedded into the architecture of the neural network via the input layer. The term *Physics* refers to the output of the physics-based model, and the subscript $p = 1, .., s$ in $Physics_p$ indicates the number of outputs in the physics-based model, in case there are multiple outputs.

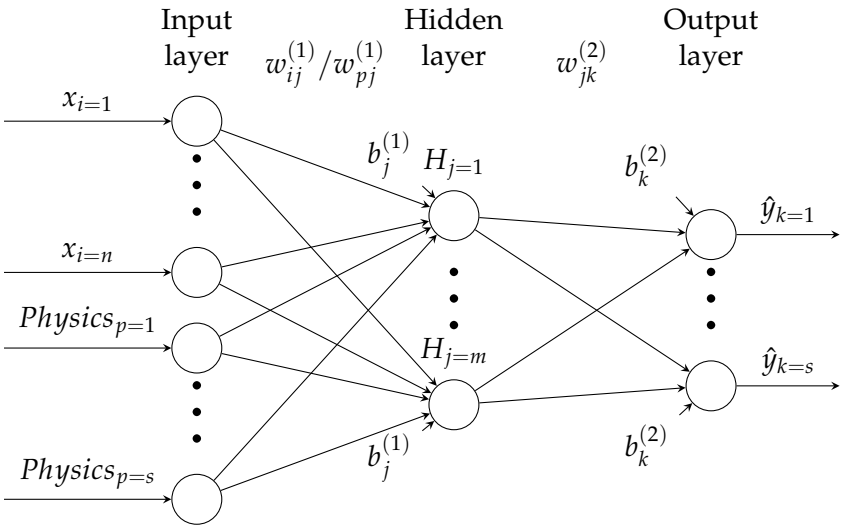


Figure 6.2: Architecture of PG-BNN by ABC-SS via the input layer

When the output of the physics-based model is used as a new input to the neural network, as in Figure 6.3, the latter is informed about a relationship between the input and

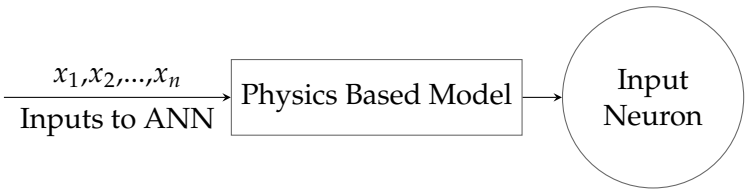


Figure 6.3: Physics-based model fed into the input neuron

output variables. That information could be comprehensive or incomplete, but it will contribute to the learning process in all cases. Moreover, during training the weights and bias of the neural network will learn how to manipulate and change the physics, based on the inputs, so the predictions of the neural network match the observed data.

The forward pass now includes the laws of physics, as shown in Equation 6.1. Therefore, this knowledge is also applied to predictions made outside the domain of the training data, thus improving the extrapolation capacities of the neural network. The uncertainty is also expected to reduce, given that the neural network is now better informed. Apart from the forward pass, the algorithm of BNN by ABC-SS remains unchanged.

$$\hat{y}_k = g(\sum_{j=1}^{m} w_{jk}^{(2)} h(\sum_{i=1}^{n} w_{ij}^{(1)} x_i + \sum_{p=1}^{s} w_{pj}^{(1)} Physics_p + b_j^{(1)}) + b_k^{(2)}) \tag{6.1}$$

6.1.3 Physics learnt through output neurons

The laws of physics should be able to explain most processes and actions that occur in the real world, however, we see that this is not the case nowadays. Quoting the British statistician George E. P. Box, "All models are wrong, but some are useful". There is much truth in that sentence, although it does not mean that the real world is not governed by physics, but that we are not able to precisely model all the complexities of the real world. In fact, the more complex models are, the more hyperparameters they include and more prone they are to overfitting. Conversely, simple models tend to generalize better and are less sensible to the tuning of the hyperparameters, probably at the expense of making less accurate predictions. This is where ANN may help, given their capacity to learn non-linear patterns from observed data. Therefore, it seems sensible to use ANN to identify and learn those complexities in the real world that simple physics-based models cannot.

As shown in Figure 6.4, this idea can be materialized by adding the outputs of the physics-based model to the output layer of the neural network, just like another bias parameter (Figure 6.5). With this new architecture the weights and bias of the neural network are forced to adjust to compensate the information coming from the laws of physics, learning those complexities and patterns that are missing in the physics-based model, such as environmental factors. Furthermore, those complexities do not need to be identified or defined in advance, given that BNN by ABC-SS provides great flexibility to adapt to different patterns and capture the uncertainty present in the observed data as a whole, no matter their nature. The physics-based model is therefore included in the forward pass as per Equation

6.2, which will improve the extrapolation capacities of the neural network significantly. Besides, the patterns learnt during training to compensate the physics will be propagated to those unexplored regions of the output variable space. The uncertainty is expected to reduce greatly within the domain of the training data, but also outside of it. Apart from the forward pass, the algorithm of BNN by ABC-SS remains unchanged.

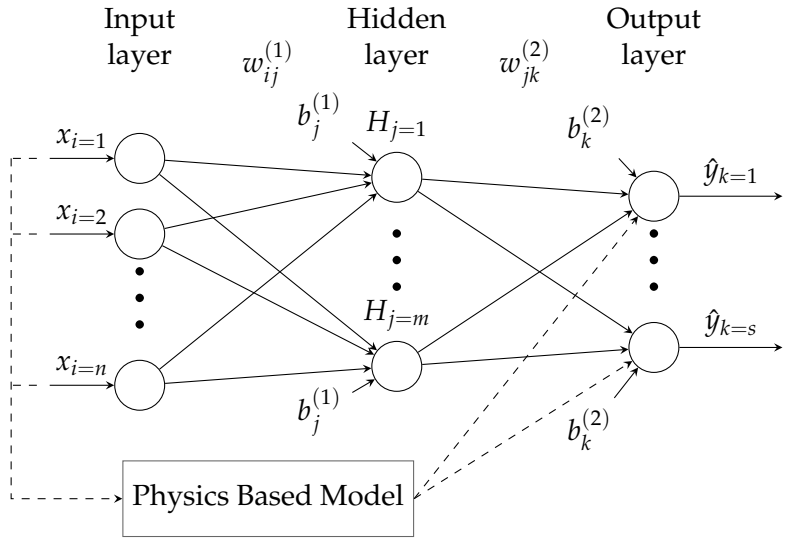


Figure 6.4: Architecture of PG-BNN by ABC-SS via the output layer

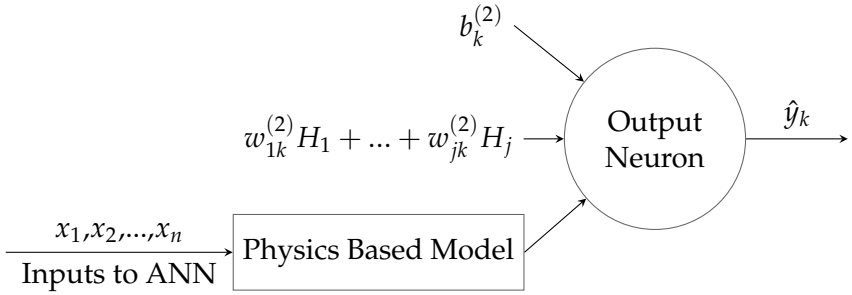


Figure 6.5: Physics-based model fed into the output neuron like a bias parameter

$$\hat{y}_k = g(\sum_{j=1}^{m} w_{jk}^{(2)} h(\sum_{i=1}^{n} w_{ij}^{(1)} x_i + b_j^{(1)}) + b_k^{(2)} + Physics_k) \tag{6.2}$$

6.2 Illustrative problem: projectile motion

In this section, the illustrative problem is described, along with an explanation about how the experimental data set has been created. Then, the results are presented and discussed.

6.2.1 Description and data processing

A projectile motion problem, using synthetic data generated in Python, has been chosen to illustrate the concepts presented in Section 6.1. In this case, a two-dimensional problem is considered where there is no lateral movement, and therefore, the motion along the

perpendicular axis 'x' (horizontal) and 'y' (vertical) can be studied independently. The variable of interest (output) is the distance travelled by the projectile d_t, which depends on the initial velocity v_0 of the projectile, the initial angle λ_0 relative to the horizontal assuming a level ground, and some unknown environmental conditions. The non-linear relationship between the independent variables v_0 and λ_0 is given by the following physics-based model:

$$\mathcal{R} = \frac{v_0^2 \sin 2\lambda_0}{g} \tag{6.3}$$

where g represents the vertical acceleration due to gravity, which is approximated to 9.81 m/s^2.

Finally, some unknown or environmental conditions need to be modelled so the synthetic observed data differs from the pure physics. For this example, some headwind has been added so that the distance travelled by the projectiles is reduced depending on the initial angle λ_0. It has been assumed that, when such angle is less than 45° the distance d_t is reduced by $\mathcal{N}(0.02\mathcal{R}, 1)$, and $\mathcal{N}(0.04\mathcal{R}, 1)$ otherwise. This is to simplistically simulate the surface friction near the ground which forces the wind to slow. That results in projectiles with high initial angle λ_0 and long range $\mathcal{R}$ to be more affected by the wind ($\sim$4% of $\mathcal{R}$) than those with flatter angles and short ranges ($\sim$2% of $\mathcal{R}$). It is worth mentioning that the observed data has been created with the only purpose of illustrating the concept of PG-BNN by ABC-SS and its potential, hence the headwind is just a non-complex pattern that is added to account for some unknown conditions.

Three data sets have been created, one training data set and two test data sets, thus the interpolation (Test Set 1) and extrapolation (Test Set 2) capabilities of the different algorithms can be evaluated. Therefore, the synthetic data is generated as follows:

$$\text{Training Data Set} \begin{cases} \text{300 data points} \\ \text{Inputs: } \lambda_0 \in [30, 60] \text{ and } v_0 \in [30, 60] \\ \text{Outputs: } d_t \begin{cases} \frac{v_0^2 \sin 2\lambda_0}{g} - \mathcal{N}(0.02\mathcal{R}, 1) \text{ when } \lambda_0 \leq 45° \\ \frac{v_0^2 \sin 2\lambda_0}{g} - \mathcal{N}(0.04\mathcal{R}, 1) \text{ when } \lambda_0 > 45° \end{cases} \end{cases}$$

$$\text{Test Data Set 1} \begin{cases} \text{150 data points} \\ \text{Inputs: } \lambda_0 \in [30, 60] \text{ and } v_0 \in [30, 60] \\ \text{Outputs: } d_t \begin{cases} \frac{v_0^2 \sin 2\lambda_0}{g} - \mathcal{N}(0.02\mathcal{R}, 1) \text{ when } \lambda_0 \leq 45° \\ \frac{v_0^2 \sin 2\lambda_0}{g} - \mathcal{N}(0.04\mathcal{R}, 1) \text{ when } \lambda_0 > 45° \end{cases} \end{cases}$$

$$\text{Test Data Set 2} \begin{cases} \text{150 data points} \\ \text{Inputs: } \lambda_0 \in [10,30] \cup [60,80] \text{ and } v_0 \in [10,30] \cup [60,80] \\ \text{Outputs: } d_t \begin{cases} \frac{v_0^2 \sin 2\lambda_0}{g} - \mathcal{N}(0.02\mathcal{R},1) \text{ when } \lambda_0 \leq 45^\circ \\ \frac{v_0^2 \sin 2\lambda_0}{g} - \mathcal{N}(0.04\mathcal{R},1) \text{ when } \lambda_0 > 45^\circ \end{cases} \end{cases}$$

The input vectors are organised in two-dimensional arrays including the initial angle and velocity [λ_0,v_0], while the output vectors are one-dimensional arrays containing the distance travelled by the projectile [d_t]. The physics-based information is arranged in one-dimensional arrays [$\mathcal{R}$].

6.2.2 Algorithmic details

The algorithms used in the illustrative problem are listed below. Their architecture comprises two hidden layers with Rectified Linear Units (ReLU) as the activation function, and the output layer with one neuron and a linear activation function. Note that the physics-guided neural networks, both the proposed models PG-BNN by ABC-SS and the benchmark models state-of-the-art (SOTA) PGNN, have an index number from (1) to (3) depending on where the physics are introduced in the ANN architecture, being (1) through the metric/loss function, (2) through the input neurons, and (3) through the output neurons.

- BNN by ABC-SS: A BNN trained with ABC-SS as per Algorithm 1 in Chapter 5, to be used as a data-driven Bayesian benchmark. The neural network structure comprises two input neurons, 15 neurons per hidden layer, and one output neuron. The hyperparameters chosen are P_0=0.1, N=100,000, σ_0=0.9, p=0.50 and tolerance value (normalized) ϵ=0.0007.
- Standard ANN with L2 regularization: A standard neural network using *TensorFlow*, to serve as a deterministic data-driven benchmark. *Adam* optimizer [123] with *early stopping* and L2 regularization are used during training. The neural network structure comprises two input neurons, 15 neurons per hidden layer, and one output neuron. The hyperparameters used are L2=0.01, *epochs*=20,000 and *patience*=100.
- PbM: Physics-based model to be used as a physics-based benchmark. The model formulation can be found in Equation 6.3.
- PG-BNN by ABC-SS: The proposed hybrid BNN trained with ABC-SS as per Section 6.1. Three variants are used as follows:
  - (1): A hybrid BNN as per Section 6.1.1. The architecture comprises 2 input neurons, 15 neurons per hidden layer, and one output neuron. The hyperparameters chosen are P_0=0.1, N=100,000, σ_0=0.9, p=0.50 and tolerance value (normalized) ϵ=0.0007. Also, three values of α (0.25, 0.5 and 0.75) have been tested.
  - (2): A hybrid BNN as per Section 6.1.2. The neural network structure comprises 3 input neurons, 5 neurons per hidden layer and one output neuron. The hyperparameters chosen are P_0=0.2, N=10,000, σ_0=0.9, p=0.50 and tolerance value (normalized) ϵ=0.0007.

  - (3): A hybrid BNN as per Section 6.1.3. In this case, the same network structure and hyperparameters as for (2) are used, but with 2 input neurons.
- SOTA PGNN: A physics-guided neural network trained with the state-of-the-art backpropagation algorithm using *TensorFlow*, to be used as a physics-guided benchmark. Three variants are tested as follows:
  - (1): A PGNN which follows the present-day approach of introducing the physics in the loss function. The training process uses the backpropagation algorithm to minimize a hybrid loss function, which includes a standard data-driven term ($Loss_d$) and a physics-based one ($Loss_p$), as follows:

$$\arg\min_{(w,b)} Loss_d(\hat{y}, y) + \lambda_p Loss_p(\hat{y}, y_p) \tag{6.4}$$

    where: $Loss_d(\hat{y}, y) = \frac{1}{N}\sum_{n=1}^{N}(\hat{y}_n - y_n)^2$; $Loss_p(\hat{y}, y_p) = \frac{1}{N}\sum_{n=1}^{N}(\hat{y}_n - y_{p,n})^2$; $\hat{y}$ is the output of the neural network; y is the training data; and y_p is the output of the physics-based model described in Equation 6.3. The neural network architecture is the same as that of BNN by ABC-SS, with 15 neurons per hidden layer. *Adam* optimizer [123] with *early stopping* is used for training, and the values of the hyperparameters are λ_p=0.5, *epochs*=20,000 and *patience*=100.
  - (2): A PGNN with the architecture presented in Figure 6.2, where the physics are introduced through the input layer. The number of neurons per layer are the same as PG-BNN by ABC-SS (2). *Adam* optimizer [123] with *early stopping* is used for training, and the values of the hyperparameters are *epochs*=10,000 and *patience*=80.
  - (3): A PGNN with the architecture presented in Figure 6.4, where the physics are introduced through the output neurons. The number of neurons per layer are the same as PG-BNN by ABC-SS (3). *Adam* optimizer [123] with *early stopping* is used for training, and the values of the hyperparameters are *epochs*=10,000 and *patience*=60.

6.2.3 Results and discussion

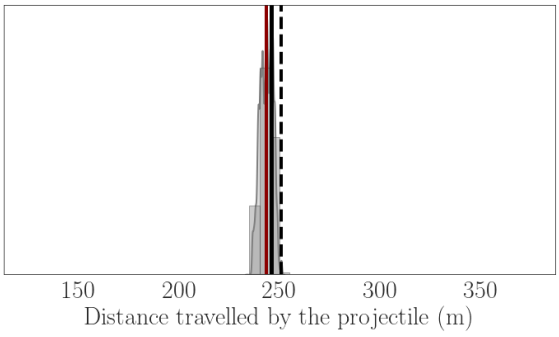
As explained in Section 6.2, a projectile motion problem is adopted to illustrate the proposed concepts, evaluate the performance of the proposed hybrid algorithms, and compare them against purely data-driven and physics-based models, as well as the state-of-the-art (SOTA) PGNN trained with backpropagation. All algorithms presented in Section 6.2.2 have been trained and tested with the data sets presented in Section 6.2.1 through 50 independent runs. The RMSE from those different runs has been recorded and the results are shown in Table 9.1. It can be seen that the proposed hybrid models where the laws of physics are introduced in the metric ρ, as per PG-BNN by ABC-SS (1), neither provide better results than the data-driven approach with BNN by ABC-SS, nor seem to improve extrapolation. However, the new metric ρ_p may be understood as a regularization tool, which may force the neural network to ignore those training data points that differ significantly from the physics. This

suggests that this hybrid model might be useful in those cases where there is a significant amount of noise in the observed data. PG-BNN by ABC-SS (2) has provided better results than the purely data-driven approaches but, even though its predictions on Test Data Set 2 have outperformed those from BNN by ABC-SS and Standard ANN, it does not extrapolate better than the physics-based model. The best results are given by PG-BNN by ABC-SS (3) and SOTA PGNN (3), especially when extrapolating in Test Data Set 2, outperforming the physics-based model, the data-driven algorithms, and the the other variants of physics-guided neural networks. The neural network in PG-BNN by ABC-SS (3) seems to find a pattern in the discrepancy between the physics-based model and the observed data which could be, for instance, some environmental conditions not included in the model, like the headwind in our case. Then, when asked to extrapolate, it applies that pattern to the physics included in the overall hybrid model, thus improving the predictions of the purely physics-based model. But most importantly, ABC-SS allows for an accurate quantification of the uncertainty as shown in Figure 6.6, where the predictions made outside the domain of the training data (extrapolation) are more disperse. That also provides us with valuable information about the degree of belief on the predictions made by the hybrid model. Finally, a mixed model where the physics-based model is introduced in both the input and output neurons was tested, however, it did not provide better results than PG-BNN by ABC-SS (3).

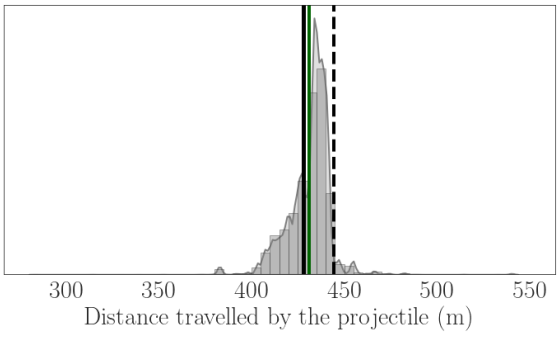
This illustrative experiment has shown that neural networks can help physics-based models to consider complex aspects that were not included in the original model, such as environmental conditions, in the same way that physics-based models can help neural networks to extrapolate outside the domain of the training data. This last aspect is graphically explained in Figure 6.7, where we see that the hybrid model benefits from both, the data-driven approach to improve the physics-based predictions, and especially from the physics-based model when extrapolating (panel b). That symbiosis brings to light the fact that hybrid models are specially useful when solving engineering problems where data is scarce but there exist relatively simple physics-based models, or at least, some prior knowledge of the task in hand. Also, the use of ABC-SS as learning engine has provided more flexibility and accuracy than standard backpropagation. This Bayesian training is the main advantage of the proposed PG-BNN by ABC-SS over the state-of-the-art methods, as it provides the user with valuable information about the uncertainty present in the observed data. Lastly, it should be noted that the computational cost of the hybrid algorithms in this experiment is comparable to that of their data-driven counterparts. However, if very complex physics-based models with high computational costs are used, then the running time of the hybrid algorithms could be impacted.

Table 6.1: Comparison between PG-BNN by ABC-SS, BNN by ABC-SS, standard ANN, the physics-based model and the state-of-the-art PGNN, for the illustrative problem in Section 6.2. The results, expressed in terms of RMSE, were obtained after 50 independent runs of each algorithm.

Statistics of RMSE obtained in 50 independent runs of the training algorithm							
	Neurons per Hidden Layer	Test Data Set 1 (Interpolation)			Test Data Set 2 (Extrapolation)		
		Q1 (P_{25})	Median (P_{50})	Q3 (P_{75})	Q1 (P_{25})	Median (P_{50})	Q3 (P_{75})
PG-BNN by ABC-SS (1) (α=0.25)	15	11.618	12.319	13.423	124.756	136.819	149.348
PG-BNN by ABC-SS (1) (α=0.5)	15	10.361	11.288	12.560	116.899	137.623	152.801
PG-BNN by ABC-SS (1) (α=0.75)	15	10.191	11.209	12.294	120.976	131.285	147.468
PG-BNN by ABC-SS (2)	5	5.249	5.909	6.588	21.271	32.211	39.404
PG-BNN by ABC-SS (3)	5	**3.670**	3.856	3.985	**5.253**	**5.919**	**6.833**
BNN by ABC-SS	15	10.254	11.376	12.529	121.682	133.947	146.087
Physics-based Model	N/A	8.780	8.780	8.780	10.527	10.527	10.527
SOTA PGNN (1)	15	23.258	23.945	24.704	113.396	115.289	117.182
SOTA PGNN (2)	5	3.755	**3.766**	**3.788**	31.938	34.780	38.839
SOTA PGNN (3)	5	3.875	3.930	3.967	5.990	6.703	8.407
Standard ANN with L2 Reg	15	5.540	7.905	20.093	126.270	134.994	140.120

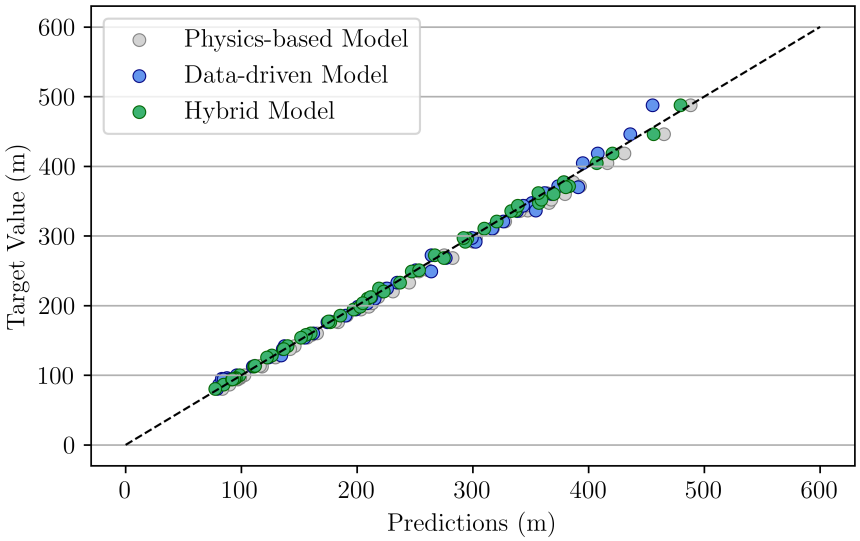


(a) Prediction inside the domain of the training data (interpolation)

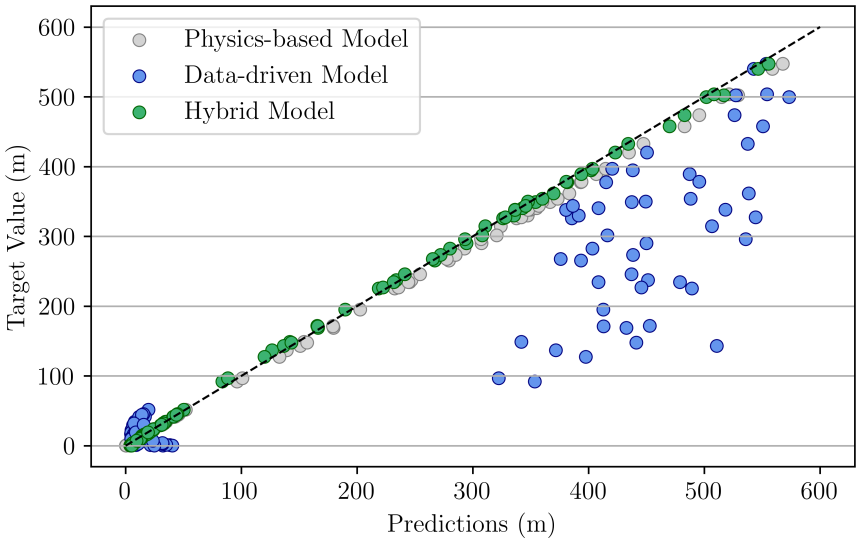


(b) Prediction outside the domain of the training data (extrapolation)

Figure 6.6: Illustrative Problem. Probabilistic predictions made by PG-BNN by ABC-SS (3) shown as a light grey density function, within the domain of the training data (interpolation) and outside of it (extrapolation). The mean predictions of the hybrid model are shown in red and green respectively. The predictions made by the purely physics-based model are shown in dashed black line and the true value of the projectile range in continuous black line.



(a) Test Data Set 1 (interpolation)



(b) Test Data Set 2 (extrapolation)

Figure 6.7: Illustrative problem. Scatter plot of target values against predicted values by the hybrid model PG-BNN by ABC-SS(3) in green, data-driven model BNN by ABC-SS in blue and the physics-based model in grey, for Test Data Set 1 (interpolation) in panel (a) and for Test Data Set 2 (extrapolation) in panel (b).

7

PG-BRNN by ABC-SS

In this chapter, the concepts and methodologies developed in Chapters 5 and 6 are applied to RNN. This type of neural network is specifically designed to process sequential data, which is very common in civil engineering. Moreover, making predictions about future events, such as time of failure, based on historical data is one of the main objectives in SHM. This new algorithm is in line with such objectives, and constitutes a new Bayesian prognostic tool. A data-driven illustrative problem is also provided, however, it is built upon the case study presented in Chapter 9, thus it is recommended that the reader explores such case study before proceeding with this illustrative example.

7.1 RNN trained with ABC-SS and guided by physics-based models

Based on the principles explained in Chapter 4 (Sections 4.3, 4.4 and 4.5), a novel RNN is proposed in this section, which combines the methods described in Chapters 5 and 6, 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 Equation 4.8. However, the hidden states no longer need to be stored anywhere, 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. This Bayesian learning engine will allow for long-term dependencies to be learnt, without the need for more complex architectures comprising a greater number of parameters. A detailed implementation of ABC-SS to train RNN can be found in Algorithm 3 below. So far in this section, it has been discussed how the principles of ABC-SS and RNN may be combined into a data-driven Bayesian RNN, hereafter called BRNN by ABC-SS.

Algorithm 3 *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 4.8 (or Equation (7.1) 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 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} $[\theta_{j-1}^{(n)}, n : 1, \ldots, N]$ so that $\rho_{j-1}^{(1)} \leqslant \ldots \leqslant \rho_{j-1}^{(n)} \leqslant \ldots \leqslant \rho_{j-1}^{(N)}$
  $\epsilon_j \leftarrow \rho_{j-1}^{NP_0}$
  $\sigma_j \leftarrow \sigma_{j-1} * p$ {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 $i : 1, \ldots, NP_0$ **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 $k : 1, \ldots, NP_0$ **do**
    $\mu \leftarrow \theta_j^{(k)}$
    for $i : 1, \ldots, ({}^1/_{P_0}) - 1$ **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 (4.8), or (7.1) 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

The next step consists of introducing the physics-based model into the proposed BRNN by ABC-SS. In light of the results obtained in the illustrative problem in Chapter 6 (Section 6.2.3), the approach proposed in the third variant (Section 6.1.3), where the physics are introduced in the forward pass through the output neurons, is followed and applied to BRNN by ABC-SS. Figure 7.1 illustrates the proposed physics-guided recurrent forward pass, and Equation 7.1 provides its mathematical formulation (note that f_1 and f_2 represent the activation functions chosen by the user). It can be seen that the physics-based model now acts like an extra bias parameter.

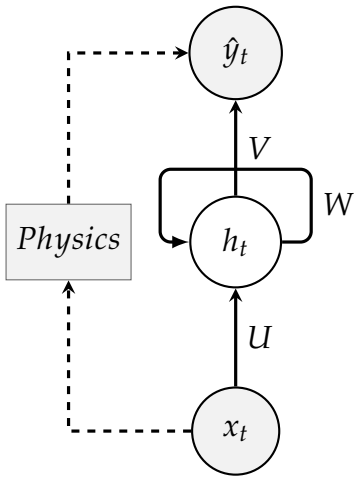


Figure 7.1: Schematic representation of the folded Physics-guided BRNN by ABC-SS

$$
\begin{aligned}
a_t &= b + Wh_{t-1} + Ux_t \\
h_t &= f_1(a_t) \\
o_t &= c + Vh_t + Physics(x_t) \\
\hat{y}_t &= f_2(o_t)
\end{aligned}
\tag{7.1}
$$

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 potential benefits are evaluated and discussed in Chapter 11.

7.2 LSTM trained with ABC-SS and guided by physics-based models

As explained in Chapter 4, LSTM neural networks were fundamentally developed to avoid gradient-related issues in RNN, and more specifically, the vanishing gradient problem. This is achieved at the expense of a more complex algorithm with more parameters and activation functions. Since ABC-SS completely avoids this drawback thanks to its gradient-free nature, it may seem that LSTM trained with ABC-SS would bring little benefit over BRNN by ABC-SS previously discussed.

Notwithstanding the foregoing, a physics-guided LSTM trained with ABC-SS (PG-LSTM by ABC-SS) is proposed in this section. Thus, it can be compared against PG-BRNN by ABC-SS, and evaluate whether it provides any further benefit. The implementation of the proposed algorithm follows the same principles as PG-BRNN by ABC-SS, however, the forward pass and the weights matrix M_{RNN} needs to be adjusted to accommodate the *forget*, *input*, *update* and *output gates*, including the extra parameters required.

The forward pass of PG-LSTM by ABC-SS can be found in equation 7.2, and the pseudocode of the ABC-SS based training is given in Algorithm 4. Sigmoid functions are represented by σ, and function f could be any activation function chosen by the user. As can be seen, the physics are introduced in the neural network through the output neuron, as per Figure 7.1, but with the recurrent cell structure shown in Figure 4.4 of Chapter 4.

$$
\begin{aligned}
i_t &= \sigma(x_t U^i + h_{t-1} W^i + b_i) \\
f_t &= \sigma(x_t U^f + h_{t-1} W^f + b_f) \\
o_t &= \sigma(x_t U^o + h_{t-1} W^o + b_o) \\
\tilde{C}_t &= \tanh(x_t U^g + h_{t-1} W^g + b_c) \\
C_t &= \sigma(f_t C_{t-1} + i_t \tilde{C}_t) \\
h_t &= \tanh(C_t) o_t \\
\hat{y}_t &= f(V h_t + Physics(x_t) + b_y)
\end{aligned}
\tag{7.2}
$$

As shown in Equation 7.2, and Algorithm 4, the amount of parameters to be adjusted in PG-LSTM by ABC-SS is considerably higher than in PG-BRNN by ABC-SS, hence the

Algorithm 4 *Training algorithm for LSTM with ABC-SS*

The pseudocode in Algorithm 3 needs to be adjusted to train LSTM as follows:

Line 2: Create ABC-SS matrix M_{LSTM}, with N rows and as many columns as the total number of parameters (θ) in LSTM matrices U^i, W^i, b_i,U^f, W^f, b_f, U^o, W^o, b_o, U^g, W^g, b_c, V, and b_y, plus one additional column to store the metric

Line 5 and 28: Parameters vector $\theta_j^{(n)}$ need to be rearranged in LSTM matrices U^i, W^i, b_i,U^f, W^f, b_f, U^o, W^o, b_o, U^g, W^g, b_c, V, and b_y.

Line 8 and 31: $\hat{y}_0^{(n)} \leftarrow$ Forward pass as per Equation 4.9 for LSTM by ABC-SS, or Equation 7.2 for PG-LSTM by ABC-SS, with LSTM matrices.

Line 43: Matrix M_{LSTM} to be updated as the concatenation of M_{Seeds} and M_{SubSet}.

computation time is expected to escalate. A significant improvement in performance is then required to justify such increase in the complexity of the neural network architecture.

7.3 Data-driven illustrative example: Fatigue in composite materials

This illustrative example aims at evaluating the performance of the proposed Bayesian RNN, and compare it against the state-of-the-art data-driven algorithms. The same data set as in Chapter 9 is used on purpose, so the superiority of RNN over standard feed forward neural networks can be demonstrated. Therefore, it is recommended that such case study is explored before proceeding to the illustrative example below.

7.3.1 Description and data processing

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 strenght-to-weight and stiffness-to-weight ratios up to 5 times larger than grade steel, and they are corrosion resistant [124]. However, they are heterogeneous materials and their damage response is very complex to understand [86], let alone predicting with accuracy the damage behaviour 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. This uncertainty about the long term behaviour of CFRP materials under fatigue conditions and our inability to accurately predict their remaining useful life, are among their main obstacles to be massively used as main structural materials in industries such as aerospace engineering [87]. 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 illustrative example, 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) [125], corresponding to the cross-ply laminates TD19 and TD21 ($[0_2/91_4]_s$). 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.

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

As explained before, this data set is used in Chater 9 as a case study to evaluate the performance of BNN by ABC-SS. Therefore, this illustrative example will also provide a fair evaluation of the benefits of using RNN to process sequential data, over standard feedforward architectures.

7.3.2 Algorithmic details

The proposed algorithms have been applied to the illustrative example described in Section 7.3.1. 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 7.3.3.

- BRNN by ABC-SS: A vanilla RNN trained with ABC-SS, as detailed in Section 7.1. The values of the hyperparameters are: P_0=0.2, N=10,000, σ_0=0.9, p=0.50, ϵ=0.003 and the activation function for the hidden layer is *tanh*.
- LSTM RNN: As explained in Chapter 4, this advanced type of RNN 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* [126] and the hyperparameters used are: minimum squared error (MSE) as the loss function, the optimizer *Adam* [127], batchsize=1 and 200 epochs.
- LSTM by ABC-SS: A LSTM RNN trained with ABC-SS as detailed in Section 7.2. The values of the hyperparameters are: P_0=0.2, N=10,000, σ_0=0.9, p=0.50, ϵ=0.003 and the activation function for the hidden layer is *tanh*.

- Bidirectional LSTM RNN: First proposed in 2005 [128], 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 and 25 epochs.
- GRU RNN: This type of recurrent neural network first appeared in 2014 [64], 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* batchsize=1 and 200 epochs.
- Monte Carlo Dropout RNN (MC Dropout): The *Dropout* method, published in 2014 [129], meant to mitigate the overfitting problem. This was the origin of a new Bayesian training algorithm for neural networks, MC Dropout [130]. This algorithm provides probabilistic outputs and a quantification of the uncertainty, and has demonstrated better performance than other state-of-the-art Bayesian methods such as *Variational Inference* [131]. In our case, MC Dropout LSTM has been implemented following [91] and the code in *GitHub*[1], with *Pytorch* [132]. The dropout method is based on some neurons being dropped on each 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 is 150 and the learning rate lr=0.01.

The performance of the different algorithms has been evaluated after 30 independent runs. 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 graphically.

7.3.3 Results and discussion

In this experiment the proposed BRNN by ABC-SS and LSTM by ABC-SS, along with the state-of-the-art RNN detailed in Section 7.3.2, have been trained using full sequences about fatigue damage evolution, and then tested on unseen data, as per Section 7.3.1. During testing, the RNN were provided with the first 4 time-steps of the sequence, corresponding

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

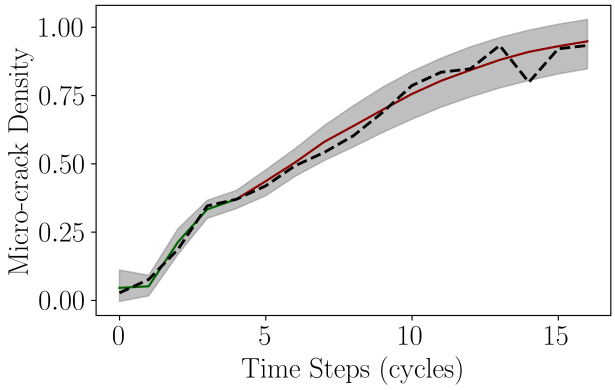
to 22% of such loading sequence, and were then asked to make predictions about future time-steps, taking as inputs their own output from the previous time-step, also called recursive multi-step forecasting. That way, the ability of the different RNN to make long-term predictions about the progression of micro-crack density can be assessed. The results are shown in Table 7.1. In terms of accuracy, it can be seen that BRNN by ABC-SS and LSTM by ABC-SS have comparable performance to the state-of-the-art LSTM and MC Dropout, although the later 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 in the observed data is graphically assessed in Figure 9.1, where the proposed BRNN by ABC-SS is compared against the state-of-the-art Bayesian RNN, namely MC Dropout. The uncertainty bounds of the algorithms trained with ABC-SS naturally increases as we move rightwards along the time-steps axis. Which makes sense given that, in a real world scenario, predictions about the structural behaviour of an asset in the far future tend to be more uncertain than those ones about the near future. In addition, the uncertainty bounds of RBNN by ABC-SS encapsulates all the target values.

Also, the results obtained in this example improve those in in Chapter 9, which confirms that RNN generally perform better on sequential data than feedforward neural networks.

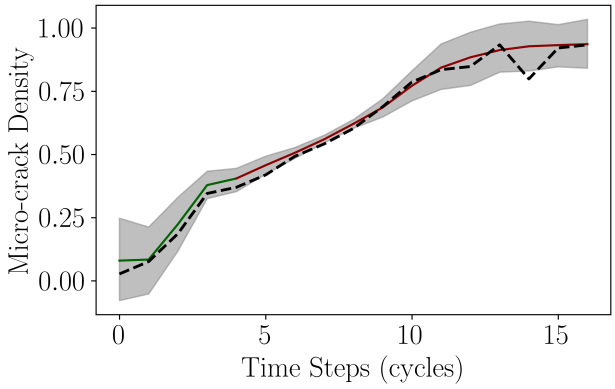
Overall, RBNN by ABC-SS has demonstrated great accuracy, comparable to that of LSTM and MC Dropout, while showing more stability, becoming a more reliable option. The absence of gradient evaluation is behind such stability, given that ABC-SS does not rely on finding a local minima of a loss function 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. Finally, LSTM by ABC-SS does not provide better results than BRNN by ABC-SS, which confirms that the gradient-related issues in RNN are solved by the use of ABC-SS. Therefore, combining ABC-SS with complex RNN architectures, such as LSTM neural networks, do not seem to provide any additional benefit, and may only result in an increase of the computational time due to the higher number of parameters to be trained. Providing that enough data was available, BRNN by ABC-SS might become a useful option for an onboard structural health monitoring system.

Table 7.1: 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**
LSTM by ABC-SS	0.00195	0.00177	0.00191	0.00208	0.00030
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

Figure 7.2: Predictions (normalized) 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}).

III

CASE STUDIES

"See now the power of truth; the same experiment which at first glance seemed to show one thing, when more carefully examined, assures us of the contrary."

— GALILEO GALILEI

8

Hyperparameter sensitivity analysis

In Chapters 9, 10 and 11, a series of cases studies are presented, where different algorithms are used. The performance of those algorithms depend on a certain number of hyperparameters that need to be optimised. As per most data-driven methods, hyperparameters in ANN are tuned through a trial and error process, testing the algorithms with different combinations of hyperparameters and evaluating their performance. This is also known as a hyperparameter sensitivity analysis, given that during the tuning process the impact of each hyperparameter in the overall performance of the ANN is exposed, along with its sensitivity to small changes. While every case study includes different algorithms with different hyperparameters, the methodology for finding the optimum value of such hyperparameters is the same, and it is briefly explained below:

- Model architecture: Different architectures have been tested, from multi-layer perceptrons with one single hidden layer, to more complex configurations with 4 hidden layers and sophisticated gated cells. The number of units tested per hidden layer varied from 1 to 400 in some cases. The data used in all case studies are not significantly complex, in terms of size and number of features. So in most cases, a low number of units was enough to provide a good performance. When a comparable performance was found between two different architectures, the simplest one was selected. Regarding the activation functions, the following types were used: ReLU, leaky ReLU, sigmoid, tangent hyperbolic and linear. Generally, ReLU provided the best results in feedforward neural networks, while tangent hyperbolic was the best for RNN. In the case of gated algorithms, such as LSTM or GRU, sigmoid and tangent hyperbolic are always used in the recurrent cell. When choosing the best combination of hidden layers, units

per layer and activation functions, different validation hold-out sets from the training data were used.

- Model hyperparameters:
  - Training based on backpropagation: The hyperparameters to be tuned vary from algorithm to algorithm, and these are: the number of epochs, the learning rate, the patience of the early stopping optimizer, the dropout probability, the λ_{L2} parameter in L2 regularization, and λ_p in the state-of-the-art physics guided neural network (SOTA PGNN (1)). As explained previously, different hold-out data sets within the training set are used. The number of epochs is selected by monitoring the training loss and validation loss, trying to find the inflexion point where more training epochs do not provide a lower validation loss. The learning rate dictates how much to change the model parameters on each epoch, and it is a sensitive hyperparameter with high impact in the search of the global minima. The *patience* is fixed to a value which avoids overfitting without compromising on model accuracy. Low values of *patience* may lead to underfitting, while higher numbers may stop the training too late, leading to overfitting. The λ_{L2} penalty parameter is also used to avoid overfitting, and needs to be adjusted iteratively to find the balance between model complexity and closely fitting the model to the training data. The dropout probability influences the number of neurons being activated in each forward pass of the algorithm. This value represents a trade-off between convergence rate and generalization. Finally, λ_p in SOTA PGNN (1) regulates the importance of the physics-based model during training. When low values are used the physics are not strongly considered, leading to better fit of the model to training data, however, higher values penalise data fitting and prioritise the physics, which may improve extrapolation.
  - Training based on ABC-SS: The hyperparameters to be optimised are P_0, N, σ_0, p and ϵ. A similar process was followed, using validation hold-out data sets. The value of P_0 dictates the proportion of samples that are kept on every simulation level as seeds, and N provides the total number of samples per simulation level. In terms of sensitivity, for more complex architectures P_0 needs to be set to a smaller value, while a bigger number of samples N are required. The values of σ_0 and p, which refer to how new samples are drawn from the proposal PDF, are more sensitive and need to be adjusted simultaneously. The value of the tolerance ϵ has a similar effect to the *patience* parameter, as it stops the simulation when a certain error is reached. This value should be set to avoid overfitting, but without compromising the accuracy of the model. A full sensitivity analysis about ABC-SS hyperparameters can be found in [84].

As a final remark, test data sets are not used during training or for hyperparameter tuning, but always reserved for the testing stage only.

9

Fatigue damage in composite materials

This chapter presents a case study about micro-crack density prediction in composite materials using the proposed BNN by ABC-SS in Chapter 5. The performance of the proposed algorithm is compared against the state-of-the-art BNN, and a probability safety assessment is also provided. The experiment is first described in Section 9.1, the probability safety assessment is explained in Section 9.2, the algorithms used are shown in Section 9.3, and finally the results and their discussion are provided in Section 9.4.

9.1 Description and data processing

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 [86]) 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 [87]. 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) [125] 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 [133]. 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 $\hat{y_1}$ 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.

9.2 Probabilistic Safety Assessment

Safety is critical in aerospace engineering, and it is the primary driver for all decisions about materials, designs and technologies to be implemented. As discussed in Section 9.1, the behaviour of composite structures under fatigue, and the interaction between their different modes of failure, are not yet well understood, which limits their implementation. Therefore, a reliable evaluation of their probability of failure is an important step towards a large scale application.

The proposed methodology starts by setting a failure threshold for the target variable, micro-crack density in our case study. This a value which, if exceeded, the composite structure will perform below a required safety standard, and does not necessarily mean material breakage. In this context, it is case specific and may differ depending on the particular application. In the experiment described in this chapter, the threshold has been set to 0.8 (normalized). Next, the different BNN are trained, so we can make predictions on the test data. These neural networks are probabilistic by nature, so their outputs are not deterministic values but a density function. The number of samples that are drawn from this output is chosen by the user, and in our case they can be found in Section 9.3. Finally, the probability of failure, being 0 very unlikely and 1 very certain, is calculated based on the proportion of samples that fall beyond the failure threshold, as follows:

$$P_{failure} = \frac{Number\,of\,Samples >= threshold}{Total\,Number\,of\,Samples} \tag{9.1}$$

The experimental data is also used to calculate the real observed probability of failure, so it can be compared against the predictions obtained from the Bayesian neural networks and check if they are consistent.

9.3 Algorithmic details and performance metrics

Refer to Chapter 8 for information about the hyperparameter sensitivity analysis. The baseline architecture used 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 in Chapter 5, adapted with a *while* loop and $\sigma_j = \sigma_0 p$. Two different architectures are used, the baseline architecture for

the comparison with the state-of-the-art BNN in Section 9.4.2, 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. The hyperparameters chosen are P_0=0.1, N=100,000, σ_0=0.75, p=0.58 and tolerance value ϵ=0.012 (ϵ=0.007 is used in the comparison exercise).

- Variational Inference, Bayes by Backprop (BBP) [29]: A BNN with the baseline architecture, trained with an open source algorithm[1] implemented in Keras [134]. 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 [123], $lr = 0.001$ and $epochs = 100,000$. The same BNN has also been trained with LeakyReLU [135] as the activation function in the hidden layers.
- Probabilistic Backpropagation (PBP) [1]: A BNN with the baseline architecture, trained with the open source algorithm[2] provided in [1]. 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) [88]: 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 [31] as follows; step size $\varepsilon = 0.001$, 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.

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 Figure 9.3, Figure 9.4, Figure 9.1(a) and Table 9.1, 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) and IQR; stability, measured by the number of outliers; computation time (Intel® Core™ i7-10510U CPU @ 1.80GHz (8 Threads) ~2.3GHz, 8GB RAM); and the capacity of

[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

the algorithms to quantify the uncertainty, evaluated graphically by the ability of the uncertainty band to capture the variability observed in the data.

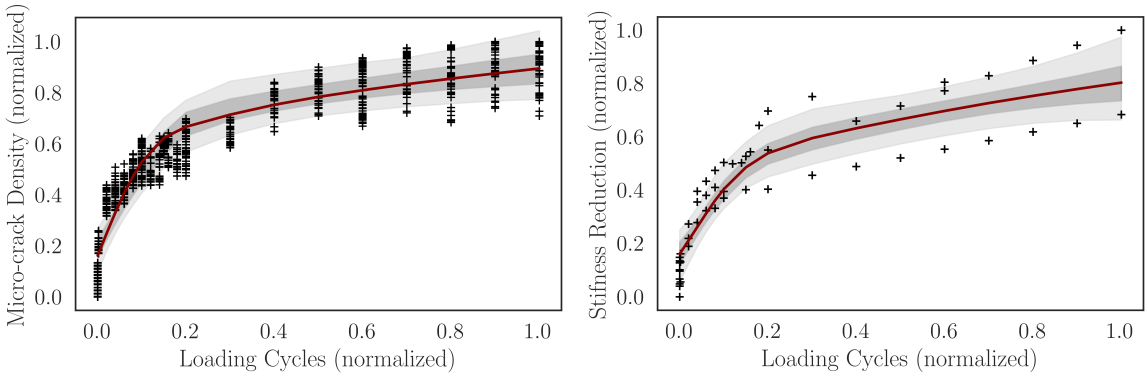
9.4 Results and discussion

As shown in Figure 9.1, 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 microcracks 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 [86]. 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 increase under real-life conditions, making the adoption of deterministic physics-based models unfeasible [136]. Failing to account this uncertainty results in high safety factors, which undermines the high utilisation 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 Figure 9.2. 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.

9.4.1 Comparison with the state-of-the-art BNN

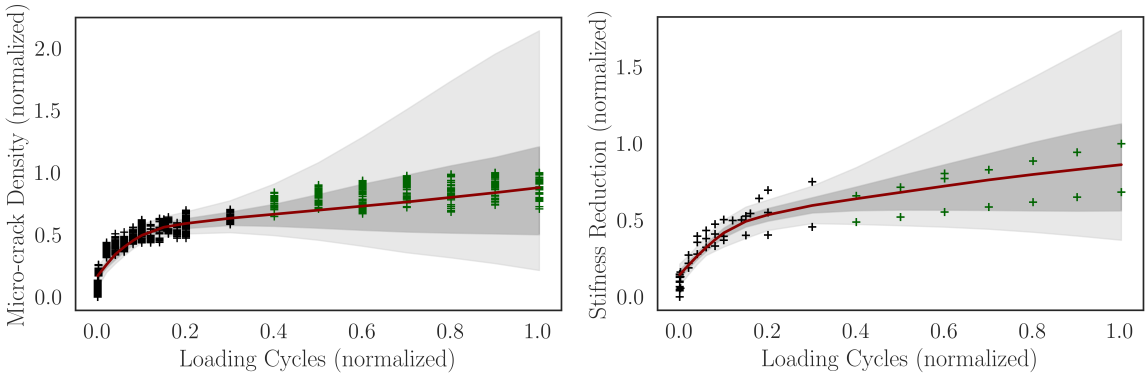
A comparative assessment has also been carried out. In particular, the results were compared with those obtained using the Variational Inference (VI) method, more specifically Bayes by Backprop (BBP) with Keras [134], Hamiltonian Monte carlo (HMC) and Probabilistic Backpropagation (PBP). The architecture selected for the chosen BNN has been presented in Section 9.3. Figure 9.3 provides a box plot of the MSE obtained after training each BNN 50 times independently, and the numerical values are shown in Table 9.1. 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,



(a) Predictions of micro-cracks density.

(b) Predictions of stiffness reduction.

Figure 9.1: 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.



(a) Predictions of micro-cracks density.

(b) Predictions of stiffness reduction.

Figure 9.2: 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.

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 [35]. 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 Figure 9.3 and Table 9.1, 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 illustrated in Figure 9.4 and Figure 9.1(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 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 [1]. 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* [137] without using optimised libraries, unlike HMC and PBP which are implemented in *Pytorch* [132] and *Theano* [138] respectively. It is presumed that the computation time of BNN by ABC-SS may be improved by using libraries based on graphs like *Tensorflow* [126], and possibly with parallel computation, remaining both options as a potential continuation of this research. By looking at the results obtained, it can 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 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.

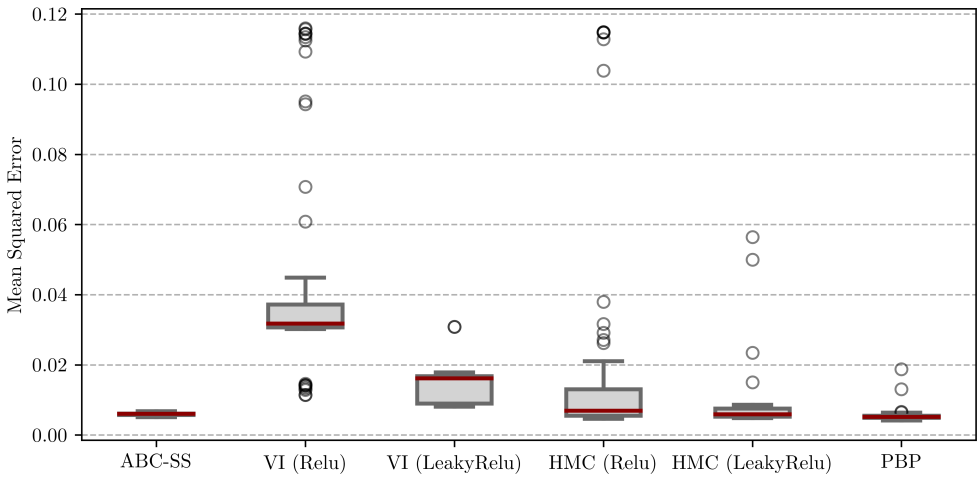


Figure 9.3: Analysis of the MSE achieved in 50 independent simulations of BNN by ABC-SS (ReLU) and Variational Inference with Bayes by Backprop (ReLU and LeakyReLU). The MSE achieved with each neural network throughout the 50 simulations is represented by their minimum, first quartile, median, third quartile, maximum and outliers.

Table 9.1: 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 summarised in this table.

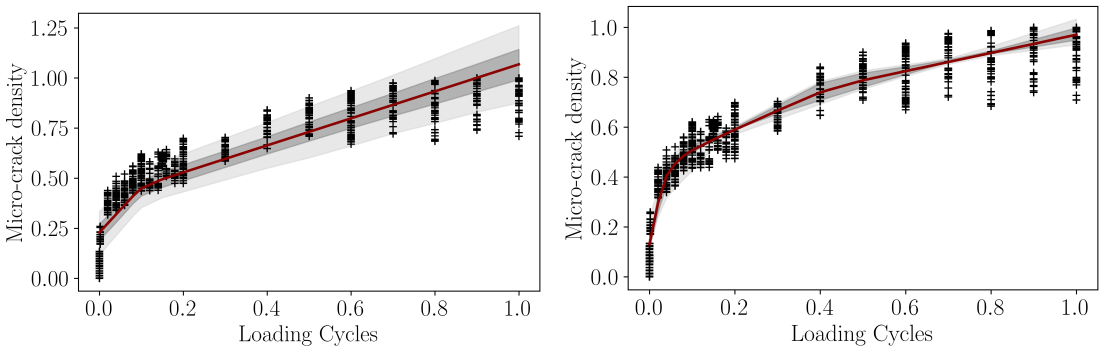
Statistics of MSE obtained in 50 independent runs of the training algorithm							
	Q1 (P_{25})	Median (P_{50})	Q3 (P_{75})	IQR (Q3-Q1)	Absolute Min-Max	Outliers	Comput. Time
BNN by ABC-SS	0.0057	0.0060	0.0062	**0.0005**	0.0052-**0.0068**	**0**	144s
VI ReLU	0.0307	0.0317	0.0394	0.0087	0.0114-0.1159	21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

9.4.2 Probability safety assessment

The probability of failure has been calculated for the last cycles of the experiment, following the methodology explained in Section 9.2, and the results are shown in Table 9.2. It can be seen that BNN by ABC-SS provides the closest probabilities to the observed data. This is clear when comparing the average difference (root mean squared error) between the probabilities given by the different algorithms and the observed data, which are: BNN by ABC-SS (0.15), HMC (0.29), PBP (0.31) and VI (0.24). The results in Table 9.2 have also been illustrated in Figure 9.5, where we can see that the green line is the best fit. Moreover, those data suffer from noise, which is most likely responsible for the negative slope in some parts of the dashed grey curve. This issue is solved by all four algorithms, as they are monotonically increasing, however, HMC and PBP seem to provide a more simple approximation, going from 0 to 1 in just a few loading cycles.

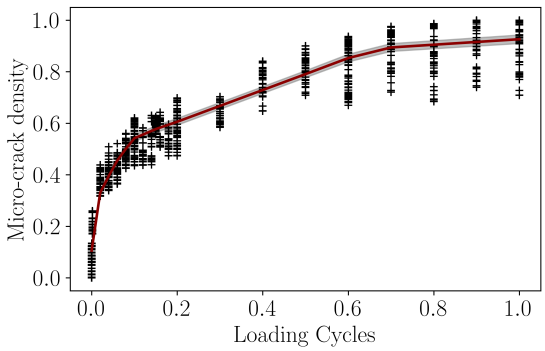
Table 9.2: Probability of failure, based on the probabilistic predictions made by the proposed algorithms. The failure threshold is set at 0.80 micro-crack density (normalized).

Probability of failure, from 0 (very improbable) to 1 (certain)									
	Number of cycles								
	100,000	150,000	200,000	250,000	300,000	350,000	400,000	450,000	500,000
Observed	0.00	0.00	0.39	0.80	0.58	0.90	0.81	0.89	0.86
BNN by ABC-SS	0.01	0.05	0.17	0.37	0.63	0.82	0.88	0.89	0.88
HMC	0.00	0.00	0.00	0.04	0.97	0.98	0.98	0.98	0.98
PBP	0.00	0.00	0.00	0.00	1.00	1.00	1.00	1.00	1.00
VI	0.00	0.00	0.02	0.15	0.48	0.78	0.93	0.98	0.99



(a) Variational Inference (Bayes by Backprop)

(b) Hamiltonian Monte Carlo



(c) Probabilistic Backpropagation

Figure 9.4: Illustrative comparison between the state-of-the-art BNN on uncertainty quantification (axes normalized). Black crosses are training samples, dark red lines are median predictions, dark grey region is the interquartile 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 manuscript [1].

Finally, the predictions made by BNN by ABC-SS during the last cycles of the experiment are shown in Figure 9.6 (green PDF), and compared against the given data (grey PDF). While the shape of those density functions are not a perfect match, the overall estimation about the probability of failure, meaning the area of the PDFs located to the right of the threshold line (red), are acceptably accurate. Again, this is thanks to the flexibility of BNN by ABC-SS to capture the uncertainty and variability in the observed data.

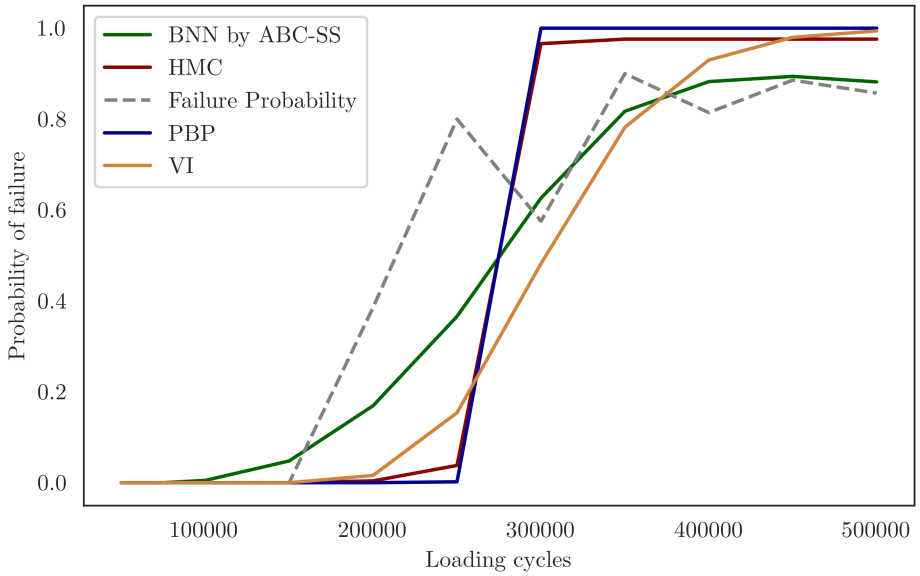


Figure 9.5: Evaluation of the probability of failure (0 to 1), based on the predictions made by the different Bayesian Neural Networks. The threshold for plausible failure was set at 0.8 micro-crack density (normalized).

.

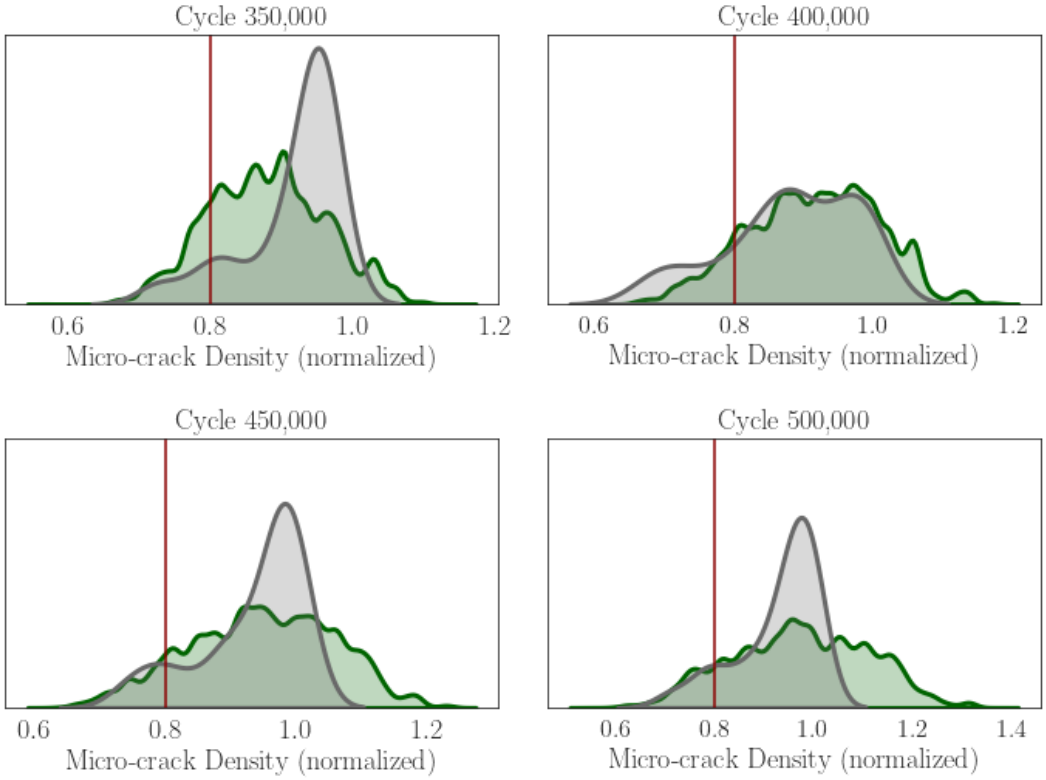


Figure 9.6: Probability density function (PDF) of predictions made by BNN by ABC-SS at different loading cycles. Those predictions, shown in green, are compared against the observed data, which are shown in light grey. The red line represents the failure threshold, and the probability of failure is given by the area of the PDF located to the right of this line.

.

10

Reinforced concrete column during seismic events

This chapter presents a case study about displacement in a reinforced concrete column during seismic events, using PG-BNN by ABC-SS as explained in Chapter 6 . The proposed hybrid models have been compared and benchmarked against their data-driven and physics-based counterparts, as well as the state-of-the-art PGNN. The experimental framework and the physics-based model used are first described in Section 10.1, the algorithms and performance metrics are shown in Section 10.2, and finally the results and their discussion are provided in Section 10.3.

10.1 Description, data processing and physics-based model

The engineering application of the proposed PG-BNN by ABC-SS consists of a cantilever reinforced concrete beam-column, subjected to constant axial load and variable cyclic lateral deformation. The lateral force F (shear strength) of the column is the variable of interest in this case, as it was the distance d_t in the illustrative problem in Chapter 6. The data used in this experiment is publicly available and were taken from [139]. In particular, the test No. 1 performed by [2] is used. This data set comprises 626 data points, which are sequential in nature, given that the displacement and shear strength were recorded continuously throughout the loading cycles. The specimen consisted of a double-ended beam column of 3300 [mm] length and 550x550 [mm] cross section (see Figure 10.1), with 12 reinforcing bars with a nominal diameter of 24 [mm] as longitudinal reinforcement, symmetrically distributed in the cross section. Lateral reinforcement comprised two 10 [mm] diameter stirrups spaced every 80 [mm]. The average concrete compressive strength was measured as 23.1 [MPa].

The yield strength of the longitudinal and transverse reinforcement was 375 [MPa] and 297 [MPa], respectively. The specimen was subjected to a constant axial compressive load of 1815 [kN]. According to [139], the data of the specimen have been adapted to the case of an equivalent cantilever column by means of an equivalent cantilever length. Accordingly, the equivalent length for the selected specimen is set equal to 1200 [mm].

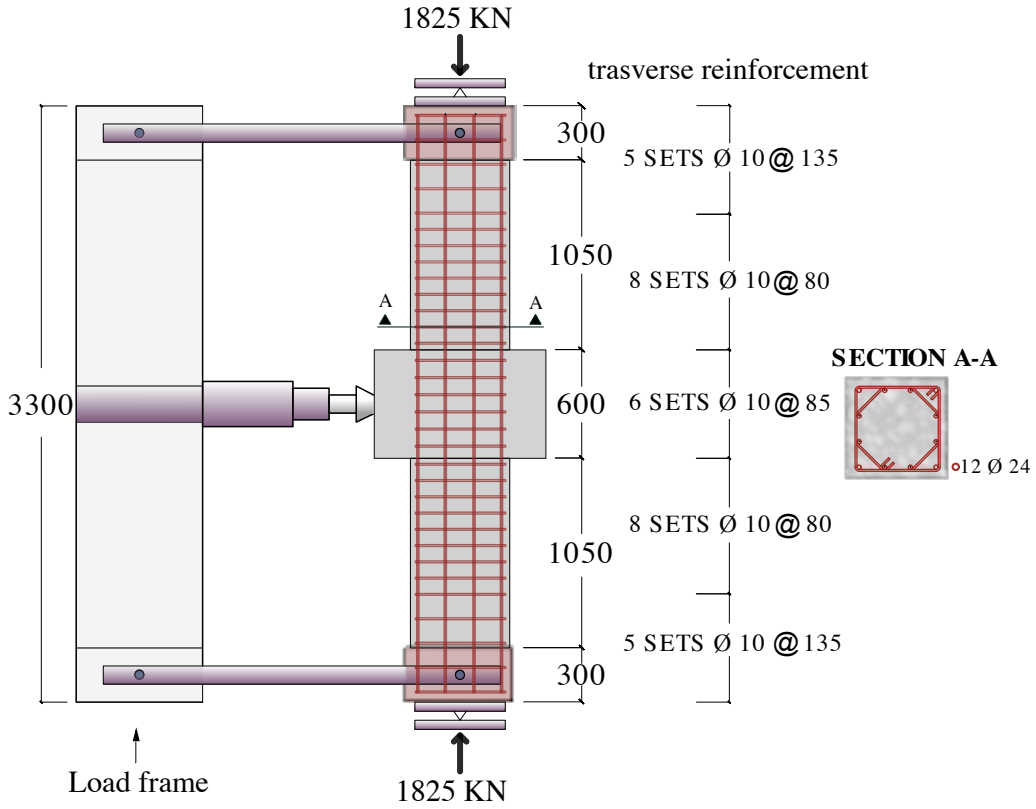


Figure 10.1: Double-ended reinforced concrete beam-column specimen details, adapted from [2].

The physics-based model used consists of a force-based formulation of a beam-column nonlinear element, fed with fiber sections. This model outputs the shear strength F_m of the column based on: (1) the lateral displacement of the free side of the cantilever, (2) the stiffness and constitutive behavior of the materials, and (3) the geometric characteristics of the element. OpenSeespy software [140] is used to construct the numerical model. The beam-column element deformations are solved using 5 Newton-Cotes integration points, each with the same fiber section. A discretization is done to model the axial and flexure behaviour of the section by means of uni-axial constitutive models, where *Concrete01* and *Steel02* models are used to represent the concrete and steel uni-axial behaviour, respectively. The input parameters of the uni-axial models are defined according to the recommendations given in [141]. Note that the concrete inside the stirrup cage is subjected to lateral pressure

due to Poisson's effect and the passive action of the stirrups; and that the lateral pressure affects the uni-axial behaviour of the concrete, giving additional strength and deformation capacity. This behaviour is considered by the confinement factor, which is estimated using the recommendations of [142]. Table 10.1 summarizes the input data of the numerical model, whereas Figure 10.2 depicts the configuration of the numerical model and the uni-axial constitutive models of the steel reinforcement and concrete.

The experimental input data fed into the neural networks are three: the lateral displacement d_l, the direction of the displacement d_d (positive or negative), and the number of cycles n_c (where one cycle is a full lateral displacement on each direction). All three inputs are obtained by processing the displacement data in [2]. Therefore, the objective is to predict the lateral force F (shear strength) at a certain time of the experiment given the lateral displacement, the direction of such displacement and the number of cycles that the column has experienced at that point. Moreover, only the first cycles of the experimental data will be used for training, and the rest will be used as test data. Thus, we can evaluate the extrapolation capabilities of the algorithms, based on their ability to make predictions about future cycles.

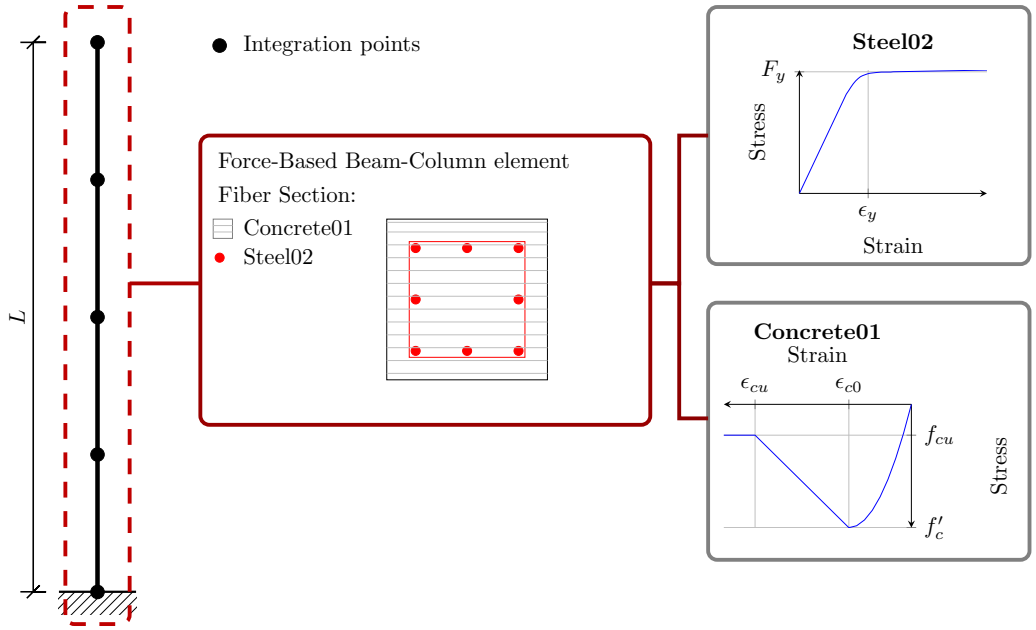


Figure 10.2: Schematic view of the nonlinear model of a cantilever reinforced concrete beam-column modelled using OpenSeespy. On the right-hand side, plots of the constitutive material monotonic behavior are presented.

The input vectors are organised in three-dimensional arrays including lateral displacement, the direction of such displacement, and the number of cycles [d_l,d_d,n_c], while the output vectors are one-dimensional arrays containing the observed force (shear strength) [F]. The physics-based information coming from the OpenSeespy model is arranged in one-dimensional arrays [F_m].

Table 10.1: Input parameters values of the reinforced concrete model in the engineering case study of Section 10.1

Axial Force	Steel Yield Strength	Concrete Compressive Strength	Cross Section	Length	Confinement Factor	Longitudinal reinforcement ratio	Strain hardening ratio
1815	375	23.10	550x550	1200	1.70	0.0179	0.0013

A data sensitivity analysis has also been carried out, where different ratios of training/test data have been used, namely 20/80, 40/60, 60/40 and 80/20. Thus the performance of the physics-guided and data-driven algorithms under different conditions of availability of data can be evaluated. It can be seen from Table 10.3, Figure 10.5 and discussion in Section 10.3, that the amount of data used for training has a significant effect on the performance of all algorithms, as could be expected.

10.2 Algorithmic details and performance metrics

The proposed hybrid models have been compared and benchmarked against their data-driven and physics-based counterparts, as well as the state-of-the-art PGNN and a standard ANN, both trained with the backpropagation algorithm using *TensorFlow*, so their performance and potential benefits can be evaluated. The results from this comparison can be found in Tables 10.2 and 10.3, and Figures 10.3-10.5. The methodology used for selecting the optimal architecture and tuning the hyperparameters is explained in Chapter 8.

The architecture used in all algorithms comprise two hidden layers with Rectified Linear Units (ReLU) as the activation function, and the output layer with one neuron and a linear activation function. The number of neurons in the input layer varies between the algorithms. Note that the physics-guided neural networks, both the proposed models PG-BNN by ABC-SS and the benchmark models state-of-the-art (SOTA) PGNN, have an numerical index depending on where the physics are introduced in the ANN architecture, being (2) through the input neurons and (3) through the output neurons. Thus, the proposed algorithms can be easily compared against their correspondent state-of-the-art algorithms. Also, in light of the poor results they provide and in line with the discussion in the illustrative problem in Chapter 6, algorithms PG-BNN by ABC-SS (1) and SOTA PGNN (1) are not included in this case study.

- BNN by ABC-SS: A BNN trained with ABC-SS as per Algorithm 1 in Chapter 5, to serve as a Bayesian data-driven benchmark. The neural network structure comprises 3 input neurons, 15 neurons per hidden layer, and one output neuron. The hyperparameters chosen are P_0=0.1, N=100,000, σ_0=0.9, p=0.50 and tolerance value (normalized) ϵ=0.0009.
- Standard ANN with L2 regularization: A standard neural network using *TensorFlow*, to serve as a deterministic data-driven benchmark. *Adam* optimizer [123] with *early stopping* and L2 regularization are used during training. The neural network structure

comprises 3 input neurons, 15 neurons per hidden layer, and one output neuron. The hyperparameters used are L2=0.01, *epochs*=20,000 and *patience*=100.

- PbM: Physics-based model to be used as a physics-based benchmark. The model formulation can be found in Section 10.1.
- PG-BNN by ABC-SS: The proposed hybrid BNN trained with ABC-SS as per Chapter 6. Two variants are used as follows:
  - (2): A hybrid BNN as per Section 6.1.2 in Chapter 6. The neural network structure comprises 4 input neurons, 5 neurons per hidden layer, and one output neuron. The hyperparameters chosen are P_0=0.2, N=10,000, σ_0=0.9, p=0.50 and tolerance value (normalized) ϵ=0.009.
  - (3): A hybrid BNN as per Section 6.1.3 in Chapter 6. In this case, the same network structure and hyperparameters as for (2) are used, but with 3 input neurons.
- SOTA PGNN: A physics-guided neural network trained with the state-of-the-art back-propagation algorithm using *TensorFlow*, to be used as a physics-guided benchmark. Three variants are tested as follows:
  - (2): A PGNN with the architecture presented in Figure 6.2 in Chapter 6, where the physics are introduced through the input layer. The number of neurons per layer are the same as PG-BNN by ABC-SS (2). *Adam* optimizer [123] with *early stopping* is used for training, and the values of the hyperparameters are *epochs*=10,000 and *patience*=80.
  - (3): A PGNN with the architecture presented in Figure 6.4 in Chapter 6, where the physics are introduced through the output neurons. The number of neurons per layer are the same as PG-BNN by ABC-SS (3). *Adam* optimizer [123] with *early stopping* is used for training, and the values of the hyperparameters are *epochs*=10,000 and *patience*=60.

The metric proposed to evaluate the performance of the algorithms is chosen taking into account the magnitude of the target variable F. This is expressed in Newtons [N], and takes significantly large values. Therefore, mean-square-error (MSE) of the normalized data is used. The precision and stability of the algorithms is measured by the value of quartiles $Q1(P_{25})$, $Q2(P_{50})$ and $Q3(P_{75})$. The ability to quantify the uncertainty is evaluated graphically.

10.3 Results and discussion

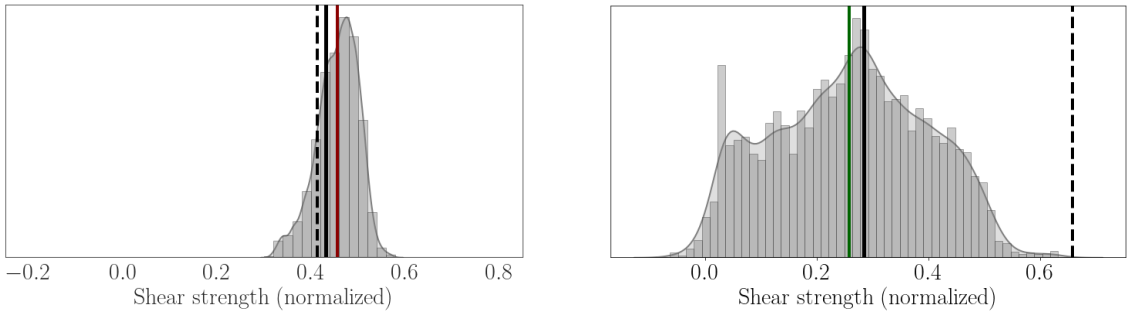
The proposed algorithms have been applied to one of the column tests recorded in the The PEER Structural Performance Database [139] as explained in Section 10.1, and benchmarked against the purely data-driven methods, such as BNN by ABC-SS and Standard ANN, the physics-based model described in that same section, and the state-of-the-art physics-guided neural networks. The algorithms, along with the choice of architecture and hyperparameters, are explained in Section 10.2 and the results of the experiment can be found in Table 10.2. Overall, the results of this experiment are similar to those obtained in the illustrative

problem in Chapter 6. When evaluated on test data, PG-BNN by ABC-SS (3) and SOTA PGNN (3) outperform the other physics-guided neural networks, the physics-based model, BNN by ABC-SS and Standard ANN, even when these purely data-driven approaches required a more complex architecture with more neurons in the hidden layers. Once again, the neural network has been able to learn a pattern in the difference between the physics and the data, so when asked to make a prediction about unseen data it compensates the information coming from physics-based model with such pattern, thus it closely matches reality. Also, the time of computation of the hybrid models is significantly lower, given its simpler architecture and relatively small number of samples N required. Interestingly, SOTA PGNN (2) and PG-BNN by ABC-SS (2) achieve low MSE values when evaluated on training data, which may suggest that introducing the physics through the input neurons is more prone to overfitting. This might be because the neural network also manipulates the physics introduced through the input layer to match the observed data. For that same reason, the performance of both SOTA PGNN (2) and PG-BNN by ABC-SS (2) seem to be worse on test data. The quantification of the uncertainty is the main advantage that the proposed hybrid models share with BNN by ABC-SS, given that both are trained with approximate Bayesian computation [84]. This is shown in Figure 10.3, where we see that PG-BNN by ABC-SS (3) not only make better predictions than the physics-based model, especially on test data, but also quantifies the uncertainty realistically. It seems natural that such uncertainty (light grey density function), translated into the width of range of plausible values, grows as we move away from the training data, as in panel (b) of Figure 10.3 and Figure 10.4. Lastly, and in line with the results obtained in the illustrative problem, the good performance of PG-BNN by ABC-SS (3) outside the domain of the training data (extrapolation) is notable, as can be seen again in Table 10.2, Figure 10.3 panel (b) and Figure 10.4, where the predictions about future cycles (green line) are acceptably accurate. As a final remark, from the results provided by both PG-BNN by ABC-SS (3) and the physics-guided SOTA PGNN (3) used as benchmark, it may be concluded that introducing the physics through the output neuron provides the best performance. Moreover, PG-BNN by ABC-SS (3) also allows for a flexible quantification of the uncertainty, which will improve the subsequent decision making process. In terms of efficiency, the proposed hybrid models showed comparable running times to that of their data-driven counterparts, as per in the illustrative example.

A sensitivity analysis about the performance of the algorithms based on the availability of data has also been carried out, and the results are shown in Table 10.3. When data is very scarce, such as 20%, the hybrid models do not seem to benefit from them significantly, as their accuracy on test data is in the same order of magnitude than the purely physics-based model. However, when a greater amount of data is available, such as 40% or 60%, the hybrid models benefit considerably from them and outperform both the data-driven methods and the purely physics-based model. This becomes more evident when 80% of the total data is available for training, as both PG-BNN by ABC-SS (3) and SOTA PGNN (3) provide very accurate predictions in comparison with all other methods. Regarding the quantification

Table 10.2: Detailed comparison, based on a training/test data ratio of 60/40, between PG-BNN by ABC-SS, BNN by ABC-SS, Standard ANN, the purely physics-based model, and the state-of-the-art PGNN. The results, expressed in terms of MSE, were obtained after 50 independent runs of each algorithm.

Statistics of MSE obtained in 50 independent runs of the training algorithm							
	Neurons per Hidden Layer	Training Data Set			Test Data Set		
		Q1 (P_{25})	Median (P_{50})	Q3 (P_{75})	Q1 (P_{25})	Median (P_{50})	Q3 (P_{75})
PG-BNN by ABC-SS (2)	5	0.0042	0.0045	0.0047	0.0103	0.0129	0.0160
PG-BNN by ABC-SS (3)	5	0.0051	0.0054	0.0056	0.0052	**0.0056**	**0.0073**
BNN by ABC-SS	15	0.0050	0.0054	0.0057	0.0157	0.0184	0.0299
Physics-based Model	N/A	0.0308	0.0308	0.0308	0.0521	0.0521	0.0521
SOTA PGNN (2)	5	0.0023	0.0030	0.0038	0.0118	0.0144	0.0243
SOTA PGNN (3)	5	0.0009	0.0011	0.0057	**0.0046**	0.0088	0.0127
Standard ANN with L2 Reg	15	0.0047	0.0052	0.0079	0.0357	0.0494	0.0745



(a) Prediction on training data

(b) Prediction on test data (extrapolation)

Figure 10.3: Engineering case study. Mean predictions made by PG-BNN by ABC-SS (3) on training data (red) and test data (green). The uncertainty is represented by the light grey PDF, the prediction of the physics-based model is given by the dashed line and the target value is the black continuous line.

of the uncertainty, Figure 10.5 shows how it evolves with the percentage of data available for training. The uncertainty reduces drastically as more data is gathered, and the biggest difference appears between 20% and 40%.

Regarding the applicability of this experiment to a real world scenario, the seismic structural engineering field could become a good candidate. One of the problems that arise in a post-earthquake scenario is the difficulty in deciding if a structure remains safe and can still be used [143], in relation to the capability of that structure to withstand the aftershocks, all aggravated by the significant uncertainty inherent to this type of phenomena. This can be of special interest for healthcare facilities, where the evacuation (or closure) of the building is not a straightforward decision during an emergency. The combination of visual inspections

with the proposed hybrid model framework could become an effective tool for fast evaluation, which is required to take an informed decision during this kind of critical scenarios. Moreover, the presented tool aligns with the current tendency in seismic structural engineering, about the need to account for uncertainties on the behaviour of structural elements [144].

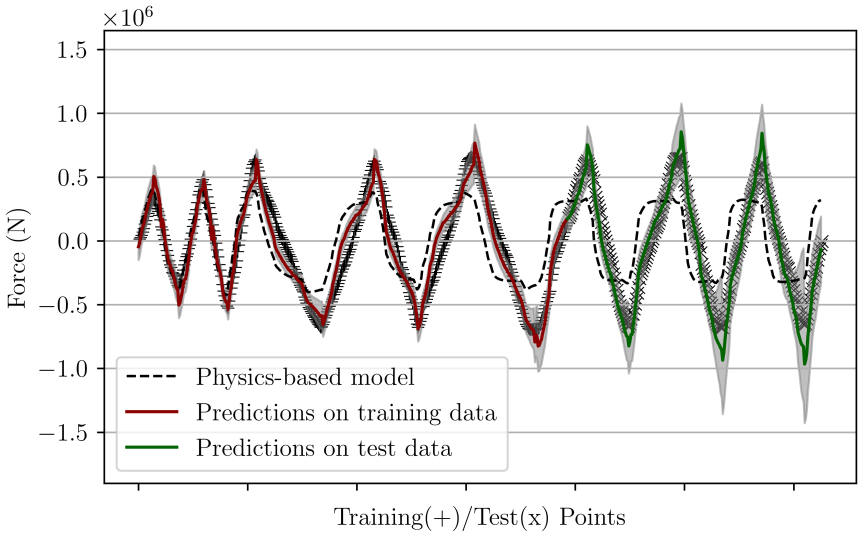
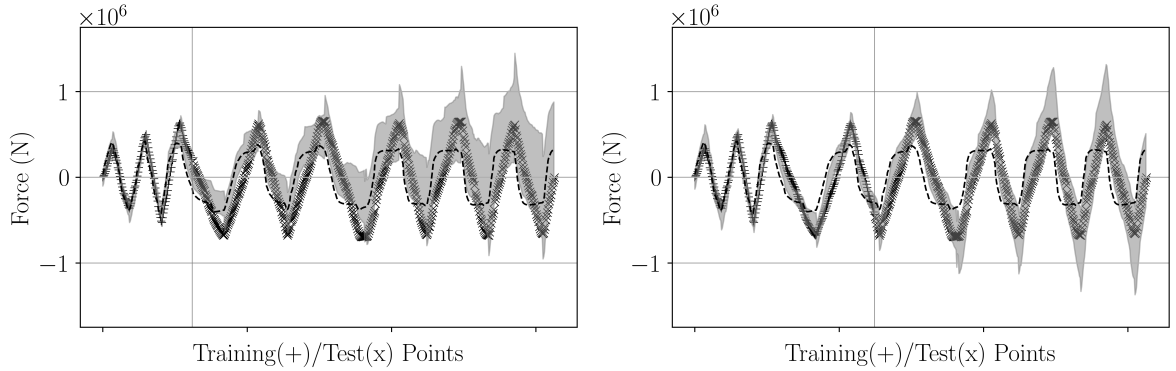


Figure 10.4: Predictions about lateral force made by PG-BNN by ABC-SS (3) on training data (red) and on test data (green). The uncertainty is represented by the grey hatch, the prediction of the physics-based model is given by the dashed line, the training data set is represented by '+' and the test data set is represented by 'x'.

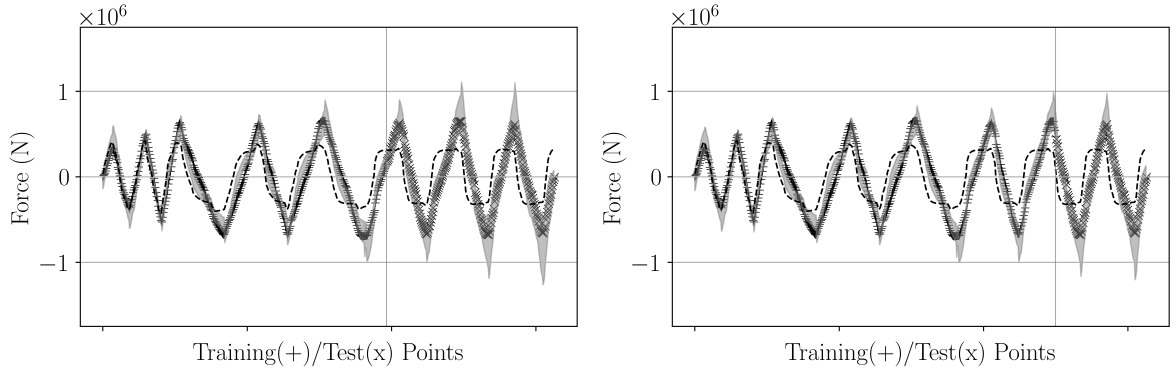
Table 10.3: Sensitivity analysis about different ratios of training/test data and the accuracy of the algorithms. The results, expressed in terms of MSE, refer to the median value (P_{50}) of the error obtained on test data after 50 independent runs of each algorithm, based on different ratios of training/test data.

Median value (P_{50}) of MSE obtained on test data after 50 independent runs					
	Neurons per Hidden Layer	Percentage of data used for training			
		20%	40%	60%	80%
PG-BNN by ABC-SS (2)	5	0.0392	0.0186	0.0129	0.0112
PG-BNN by ABC-SS (3)	5	0.0460	0.0083	0.0056	0.0031
BNN by ABC-SS	15	0.1947	0.1616	0.0184	0.0162
SOTA PGNN (2)	5	0.0740	0.0540	0.0144	0.0107
SOTA PGNN (3)	5	0.0481	0.0362	0.0088	0.0030
Standard ANN with L2 Reg	15	0.1250	0.1244	0.0494	0.0334
Physics-based Model	NA	0.0459	0.0512	0.0521	0.0587



(a) 20% of data used for training

(b) 40% of data used for training



(c) 60% of data used for training

(d) 80% of data used for training

Figure 10.5: Data sensitivity analysis. The uncertainty quantified PG-BNN by ABC-SS (3) is represented by the grey hatch, the prediction of the physics-based model is given by the dashed line, the training data set is represented by '+' and the test data set is represented by 'x'. The vertical grey lines divide the training data domain from the test data domain. It can be seen how the uncertainty reduces gradually as more data is available for training.

11

Accelerations in concrete structures

This chapter presents a case study about accelerations in a concrete building during a seismic event, using PG-BRNN by ABC-SS as explained in Chapter 7 . The proposed hybrid models have been compared and benchmarked against their data-driven and physics-based counterparts, as well as the state-of-the-art PG-RNN. The experimental framework and the physics-based model used are first described in Section 11.1, the algorithms and performance metrics are shown in Section 11.2, and finally the results and their discussion are provided in Section 11.3.

11.1 Description, data processing and physics-based model

The potential of PG-BRNN by ABC-SS to become part of a wider SHM system is explored in this chapter. The application of machine learning algorithms to SHM has already been studied in the existing literature [145, 146], 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 [147] (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*, 9^{th} floor. The input signal at the base of the building is also available in that data set.

Unlike the illustrative data-driven example in Chapter 7, 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 9^{th} 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 only used to calculate the metric, and evaluate the performance of the algorithms.

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

The physics-based model used in the hybrid neural networks is based on the computational model proposed in [3]. 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. [3] 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 11.1 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.

Table 11.1: Values of the model parameters proposed in [3]

	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	390594	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	895961	5.07E+06	3.18E+06	4.28E+06	4.84E+06	697790	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.

11.2 Algorithmic details and performance metrics

The details of the algorithms used in this case study, along with 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 11.3.

- BRNN by ABC-SS: A vanilla RNN trained with ABC-SS, as detailed in Chapter 7. The values of the hyperparameters are: P_0=0.2, N=10,000, σ_0=0.9 and p=0.50. The activation function for the hidden layer is *ReLU* and the tolerance value is ϵ=0.005.
- LSTM RNN: As explained in Section 7.3.2 of Chapter 7, this algorithm has been implemented using *Tensorflow* [126] and the hyperparameters used are: minimum squared error (MSE) as the loss function, the optimizer *Adam* [127], batchsize=1 and 100 epochs.
- LSTM by ABC-SS: A LSTM RNN trained with ABC-SS as detailed in Section 7.2 of Chapter 7. The values of the hyperparameters are: P_0=0.2, N=10,000, σ_0=0.9, p=0.50, ϵ=0.005 and the activation function for the hidden layer is *tanh*.
- Bidirectional LSTM RNN: As explained in Section 7.3.2 of Chapter 7, this algorithm has been implemented using *Tensorflow* and the hyperparameters used are: minimum

squared error (MSE) as the loss function, the optimizer *RMSprop*, batchsize=1 and 50 epochs.

- GRU RNN: As explained in Section 7.3.2 of Chapter 7, this algorithm has been implemented using *Tensorflow* [126] and the hyperparameters used are: minimum squared error (MSE) as the loss function, the optimizer *Adam* [127], batchsize=1 and 100 epochs.
- Monte Carlo Dropout RNN (MC Dropout):As explained in Section 7.3.2 of Chapter 7, MC Dropout LSTM has been implemented following [91] and the code in *GitHub*[1], with *Pytorch* [132]. Regarding the hyperparameters, the dropout probability is 0.5, the loss function is MSE and batch size=8. The hyperparameters used are are 50 epochs and lr=0.0001.
- PG-BRNN by ABC-SS: A physics-guided RNN trained with ABC-SS as per Chapter 7, Section 7.1. The physics-based model, specified in Section 11.1, 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 figure 7.1 in Chapter 7. The hyperparameters are: MSE as the loss function, the optimizer *Adam*, batchsize=1 and 10 epochs.
- PG-LSTM by ABC-SS: A physics-guided LSTM trained with ABC-SS as per Chapter 7, Section 7.2. The physics-based model, specified in Section 11.1, 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-Bidirectional LSTM RNN: A physics-guided Bidirectional LSTM RNN, with the physics-based model introduced in the output neuron, as per figure 7.1 in Chapter 7. 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 figure 7.1 in Chapter 7. 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 figure 7.1 in Chapter 7. 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 11.1.

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

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

interquartile range (IQR). The ability to quantify the uncertainty, in those algorithms trained with Bayesian methods, is evaluated graphically.

11.3 Results and discussion

As explained in Section 11.1, this 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 Chapter 4 Section 4.4, given their capacity to generalize well throughout 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 11.2. 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 Figure 11.1, where the left side of each plot is a density function of the MSE obtained by the physics-guided versions, and the right side refers to the data-driven versions of each RNN. 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 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 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. Regarding PG-LSTM by ABC-SS, this has not provided better results than PG-BRNN by ABC-SS, even when the LSTM architecture is more complex. This fact confirms that ABC-SS solves the gradient-related issues in vanilla RNN, and therefore, combining this Bayesian training method with more complex RNN may not present any further benefit. Finally, the quantification of the uncertainty of PG-BRNN by ABC-SS is more realistic than that of its Bayesian competitor PG-MC Dropout LSTM, as shown in Figure 11.2. PG-BRNN by ABC-SS seems slightly more confident as well as precise. Another reading from Figure 11.2 is that both Bayesian RNN are quite confident

about the point when accelerations change in direction, but not so much about when they reach their highest levels. This suggest that the inflexion point between accelerating and decelerating is the most difficult part to predict.

Table 11.2: 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-BRNN 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-LSTM by ABC-SS	0.00053	0.00046	0.00049	0.00060	0.00014
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
BRNN 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
LSTM by ABC-SS	0.09772	0.04711	0.07268	0.14234	0.09523
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		

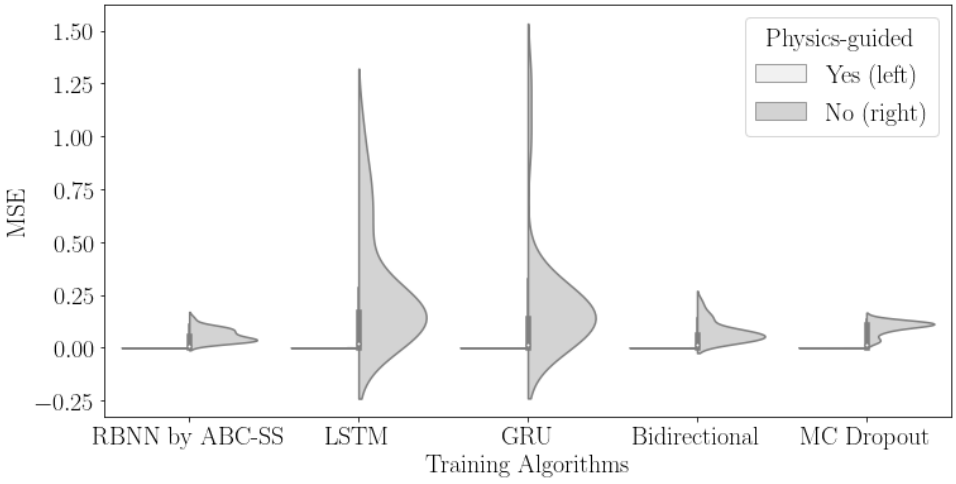
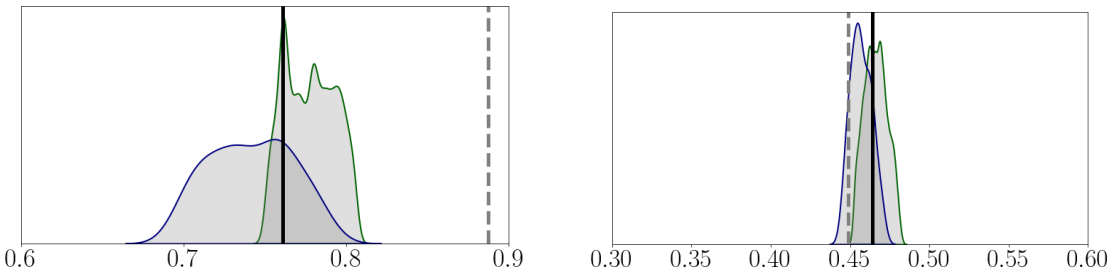


Figure 11.1: Violin plot of the MSE obtained for each algorithm after 30 independent runs.



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

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

Figure 11.2: PDF (P_5 - P_{95}) of the predictions made by PG-BRNN 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.

The predictions provided by PG-BRNN by ABC-SS have also been evaluated in the frequency domain after applying the fast Fourier transform (FFT) [148], as shown in Figure 11.3. 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.

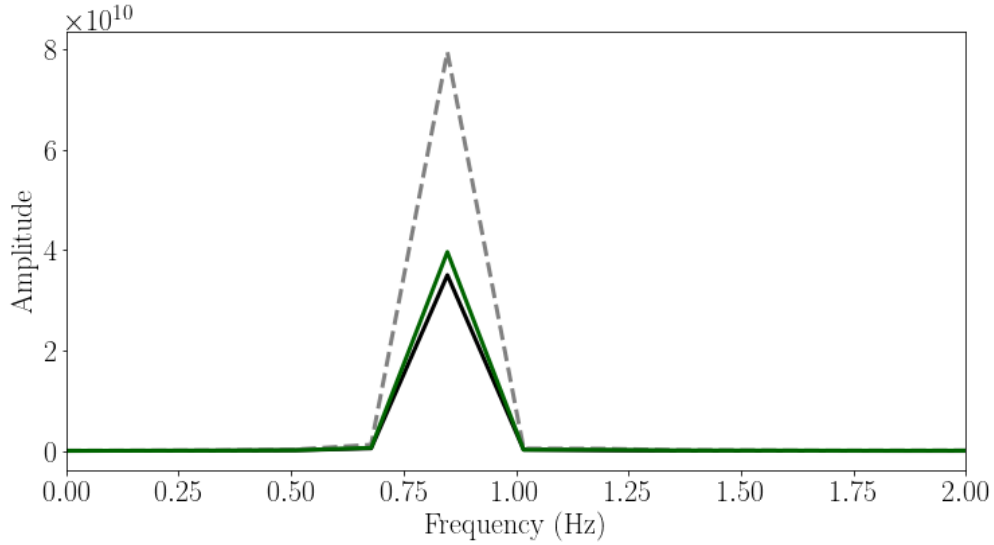


Figure 11.3: Comparison of the Fourier amplitudes provided by the physics-based model [3] (dashed-grey), PG-BRNN by ABC-SS (green), and experimental measurements (black).

IV

CONCLUSIONS

"An expert is a person who has made all the mistakes that can be made in a very narrow field."

— NIELS BOHR

12

Conclusions and future work

Standard artificial neural networks need a considerable amount of data for training, and even if these are available, extrapolation is beyond their capabilities. Moreover, their training is based on the backpropagation algorithm, which is the cause of gradient-related problems. These drawbacks hinder the implementation of artificial neural networks in many areas of engineering, where data is scarce and safety is the number one priority.

In this doctoral thesis, a new training algorithm based on approximate Bayesian computation by Subset Simulation was presented. While this methodology provides comparable accuracy to that of the state-of-the-art neural networks, it also quantifies the uncertainty inherent in the observed data. This translates into valuable information for the subsequent decision-making process, and may contribute to build trust around the use of artificial neural networks in engineering. Furthermore, this new Bayesian training method was combined with physics-based models into a hybrid algorithm, reducing the amount of data required and improving significantly the extrapolation capabilities. Finally, the aforementioned principles were applied to recurrent neural networks, so sequential data could be exploited using the proposed methodology. The probabilistic nature of approximate Bayesian computation avoids the gradient-related problems of backpropagation, allowing for long-term dependencies between distant data points to be learnt without the use of complex architectures.

These new concepts were explained using illustrative examples, and the performance of the proposed algorithms were evaluated in three different engineering case studies, mainly related to the structural integrity of different elements and materials. The results demonstrated the potential of the methodology to become a part of PHM systems, given its ability

to quantify the uncertainty and provide real-time predictions based on historical data and physics-based models.

More specifically, the following conclusions could be extracted from each of the hypothesis and research objectives described in Chapter 2. The limitations of the proposed methodologies and potential future works are also summarised below:

Hypothesis 1: The value of the weights and bias of ANN could be optimised using Approximate Bayesian Computation by Subset-Simulation (ABC-SS).

Most 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.

In Chapter 5, a novel training method for Bayesian neural networks was developed using approximate Bayesian computation combined with Subset Simulation as inference engine. The resulting methodology, named in this thesis as BNN by ABC-SS, was illustrated in Sections 5.2 and 5.3 of Chapter 5 using two academic examples with synthetic data created from sine and cosine functions with added noise, and then applied to an engineering case study about fatigue damage in composite structures in Chapter 9. The results revealed that the non-parametric formulations of the likelihood function and the PDF of the weights provide a realistic uncertainty quantification according to the training data. Besides, through comparison with the Variational Inference, Hamiltonian Monte Carlo and Probabilistic Backpropagation methods, BNN by ABC-SS showed more stability when making predictions, presumably due to absence of gradient. Particularly for the case study about damage propagation in composites, the proposed data-driven methodology can be seen as an alternative to purely physics-based models, which fail at quantifying the real amount of prediction uncertainty.

In addition, a probabilistic safety assessment was carried out in Section 9.2 of Chapter 9, using the predictions made by the different Bayesian neural networks. The objective was to model the probability of failure at a given loading cycle based on the probabilistic output of the algorithm and a predefined damage threshold. BNN by ABC-SS provided the best results, demonstrating flexibility to capture the variability in the data. Thereby, its predictions about the probability of failure approximated the observed data significantly well.

Finally, it may be concluded that this new training algorithm becomes specially useful when applied to problems where a decision is significantly dependent on the amount of uncertainty. Moreover, their predictions can be used in subsequent probabilistic safety assessments, which in turn also helps to make informed decisions regarding maintenance, or the potential replacement of the structural element.

Regarding future works, 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. Also, a more directed sampling method should be investigated, so the number of samples required can be reduced and the efficiency of the overall method improved. Both paths are closely related and establishes a natural continuation to this line of research.

Hypothesis 2: The limitations related to data scarcity, extrapolation and quantification of the uncertainty could be mitigated if physics-based models are included in the forward pass of the neural network in the form of mathematical formulations, and Bayesian training is used.

Chapter 6 of this thesis presented a new algorithm which combines BNN by ABC-SS with physics-based models, the so-called PG-BNN by ABC-SS. Unlike other physics-guided neural networks where the physics are often introduced in the loss function or through boundary conditions, and then backpropagated during training, the proposed algorithm inserts the physics directly in the forward pass, which improves the extrapolation capabilities. Moreover, ABC-SS is a Bayesian gradient-free training method that provides the proposed algorithm with stability, flexibility and the ability to quantify the uncertainty. Those properties were evaluated in a case study about the behaviour of a reinforced concrete column during a seismic event, where the accuracy and reliability of PG-BNN by ABC-SS surpassed those of the state-of-the-art physics-guided neural networks trained with backpropagation, and outperformed significantly the purely physics-based and data-driven approaches.

The two main advantages of PG-BNN by ABC-SS, namely its ability to extrapolate outside the domain of the training data set and to quantify the uncertainty in the outputs, significantly improve the accuracy of predictions about future events, and limit the risk in the subsequent decision making process. The results in the engineering case study showed the potential of the proposed algorithm to become, if combined with visual inspections, an effective and fast tool to evaluate and diagnose the condition of structural elements after seismic events. Certainly, a tool that can anticipate the outcome of an event of which there is little data, with a defined degree of confidence, could be particularly useful in different engineering fields.

Future research should focus on different ways of introducing the physics-based models within the architecture of the artificial neural network, so the parameters of both data-driven and physics-based models are closely interconnected. Also, the use of adaptive activation functions should be explored.

Hypothesis 3: A RNN that includes physics-based models in its forward pass and is trained using Bayesian methods can avoid the aforementioned limitations and may provide a useful methodology for prognostics. Additionally, gated units could be avoided, reducing the number of weights and bias required.

When the observed data is sequential, 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 Chapter 7, 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 in prognostics, where 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, absence of gradient evaluation, and probabilistic outputs with accurate quantification of the uncertainty. The same principles were applied to LSTM neural networks, the so-called LSTM by ABC-SS, to assess if there is any additional benefit in using complex gated architectures even when there is no gradient evaluation.

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 physics-guided 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. Finally, PG-LSTM by ABC-SS did not present any improvement over PG-BRNN by ABC-SS in terms of accuracy or stability, while the computation time increased significantly due to the higher number of parameters to be trained. This demonstrates that ABC-SS allows the basic RNN architecture to capture long term dependencies, without the need for gated units.

In relation to future works, BRNN by ABC-SS could be explored as a quasi-real time predictor for onboard PHM 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 rapidly evaluate its structural integrity and the safety of the utility systems. Lastly, and in line with the conclusions in Hypothesis 1, ABC-SS is limited by the dimension of the parameter space, and may not be suitable for training RNN with a very large number of neurons, such as those used for complex video activity recognition. While engineering applications do not often require such

high-dimensional neural networks, extending the training capacity of ABC-SS to more complex architectures needs to be explored. The use of parallel computing could also accelerate the sampling method in ABC-SS, helping to optimize the computational resources available.

13

Conclusiones y trabajos futuros

Las redes neuronales artificiales necesitan una cantidad considerable de datos para su entrenamiento, e incluso si estos están disponibles, la extrapolación va más allá de sus capacidades. Además, su entrenamiento se basa en el algoritmo de retropropagación ("*backpropagation*"), que es la causa de los problemas relacionados con el gradiente. Estos inconvenientes dificultan la implementación de las redes neuronales artificiales en muchas áreas de la ingeniería, donde los datos son escasos y la seguridad es la prioridad número uno.

En esta tesis doctoral se presentó un nuevo algoritmo de entrenamiento basado en computación Bayesiana aproximada por simulación de subconjuntos. Si bien esta metodología proporciona una precisión comparable a la de las redes neuronales actuales, también cuantifica la incertidumbre inherente a los datos observados. Esto se traduce en información valiosa para la posterior toma de decisiones y puede contribuir a generar confianza en torno al uso de redes neuronales artificiales en ingeniería. Este nuevo método de entrenamiento Bayesiano también se combinó con modelos basados en física para crear un algoritmo híbrido, lo que redujo la cantidad de datos necesarios y mejoró significativamente las capacidades de extrapolación. Finalmente, los principios antes mencionados se aplicaron a redes neuronales recurrentes, por lo que los datos secuenciales pueden explotarse utilizando la metodología propuesta. La naturaleza probabilística de la computación Bayesiana aproximada evita los problemas relacionados con el gradiente, lo que permite aprender las relaciones de dependencia existentes entre datos de entrenamiento alejados en el tiempo sin el uso de arquitecturas complejas.

Estos nuevos conceptos se explicaron con ejemplos ilustrativos, para luego evaluar el rendimiento de los algoritmos propuestos con tres casos de estudio diferentes, principalmente relacionados con la integridad estructural de diferentes elementos y materiales. Los resultados demostraron el potencial de la metodología propuesta para formar parte de sistemas de pronóstico y gestión de la salud estructural (PHM), dada su capacidad para cuantificar la incertidumbre y realizar predicciones en tiempo real basadas en datos históricos y modelos físicos.

Más específicamente, las siguientes conclusiones se pueden extraer de cada una de las hipótesis y objetivos de investigación descritos en el Capítulo 2. Las limitaciones de las metodologías propuestas y los posibles trabajos futuros también se resumen a continuación:

Hipótesis 1: El valor de los pesos y sesgos en las redes neuronales artificiales podría optimizarse utilizando computación Bayesiana aproximada por simulación de subconjuntos (ABC-SS).

La mayoría de los algoritmos de entrenamiento Bayesiano actuales usan funciones de densidad de probabilidad (FDP) paramétricas rígidas para describir la función de verosimilitud y/o los pesos y sesgos, generalmente una función gaussiana definida por su media y desviación típica, lo que condiciona su capacidad para representar la incertidumbre en los datos observados. Y no es la única limitación, ya que a menudo están sujetos a los inconvenientes del gradiente y la retropropagación.

En el capítulo 5, se desarrolló un nuevo método de entrenamiento para redes neuronales Bayesianas utilizando computación Bayesiana aproximada por simulación de subconjuntos como motor de inferencia. La metodología resultante, denominada en esta tesis como BNN by ABC-SS, se ilustra en las secciones 5.2 y 5.3 del capítulo 5 usando ejemplos académicos con datos sintéticos creados a partir de funciones de seno y coseno con ruido agregado, y luego aplicados a un caso de estudio sobre fatiga en estructuras compuestas en el Capítulo 9. Los resultados revelaron que la formulación no paramétrica de la función de verosimilitud y la FDP de los pesos proporciona una cuantificación realista de la incertidumbre acorde a los datos de entrenamiento. Así mismo, mediante la comparación con el método "*Variational Inference*", "*Hamiltonian Monte Carlo*" y "*Probabilistic Backpropagation*, BNN by ABC-SS demostró mayor estabilidad al realizar predicciones, en gran parte gracias a la ausencia de gradiente. Particularmente para el caso de estudio sobre la propagación de daño en materiales compuestos, la metodología propuesta puede verse como una alternativa a aquellos modelos puramente físicos que no logran cuantificar la cantidad real de incertidumbre en las predicciones.

Asimismo, se realizó una evaluación probabilística de seguridad en la Sección 9.2 del Capítulo 9, utilizando las predicciones llevadas a cabo por las diferentes redes neuronales

Bayesianas. El objetivo era modelar la probabilidad de fallo en un ciclo de carga determinado en función de la predicción probabilistica del algoritmo y un umbral de daño predefinido. BNN by ABC-SS proporcionó los mejores resultados, demostrando flexibilidad para capturar la variabilidad en los datos. Por lo tanto, sus predicciones sobre la probabilidad de fallo se aproximaron significativamente bien a los datos observados.

Finalmente, se puede concluir que este nuevo algoritmo de entrenamiento es especialmente útil cuando se aplica a problemas donde se requiere tomar una decisión teniendo en cuenta la cantidad de incertidumbre existente. Por la misma razón, sus predicciones también se pueden utilizar en evaluaciones de seguridad, lo que a su vez ayuda a tomar decisiones mejor informadas con respecto al mantenimiento, o la posible sustitución del elemento estructural.

En cuanto a trabajos futuros, la escalabilidad del método propuesto para entrenar redes neuronales profundas, con espacios de parámetros de alta dimensión y grandes conjuntos de datos de entrenamiento, podría ampliar aún más el rango de aplicaciones potenciales de esta metodología. Igualmente, se debe investigar un método de muestreo más dirigido, de modo que se pueda reducir el número de muestras requeridas y mejorar la eficiencia del método en general. Ambos caminos están estrechamente relacionados y establecen una continuación natural a esta línea de investigación.

Hipótesis 2: Las limitaciones relacionadas con la escasez de datos, la extrapolación y la cuantificación de la incertidumbre podrían mitigarse si modelos basados en física se incluyen en la ecuación de la red neuronal en forma de formulaciones matemáticas, y se usa un entrenamiento Bayesiano.

El capítulo 6 de esta tesis presentó un nuevo algoritmo que combina BNN by ABC-SS con modelos basados en física, el llamado PG-BNN by ABC-SS. A diferencia de otras redes neuronales guiadas/informadas por física, en las que esta a menudo se introduce en la función de coste o a través de ciertas condiciones de contorno, y luego se retropropaga durante el entrenamiento, el algoritmo propuesto inserta la física directamente en la ecuación de la red neuronal Bayesiana, lo que mejora las capacidades de extrapolación. Adicionalmente, ABC-SS es un método de entrenamiento sin evaluación del gradiente que dota al algoritmo de estabilidad, flexibilidad y capacidad de cuantificar la incertidumbre. Esas propiedades fueron evaluadas utilizando un caso de estudio sobre el comportamiento de una columna de hormigón armado durante un evento sísmico, donde la precisión y fiabilidad de PG-BNN by ABC-SS superó las de las actuales redes neuronales guiadas por física y entrenadas con retropropagación, y mejoró significativamente a los modelos basados puramente en física y datos.

Las dos ventajas principales de PG-BNN by ABC-SS, a saber, su capacidad para extrapolar fuera del dominio del conjunto de datos de entrenamiento y para cuantificar la incertidumbre en los resultados, mejoran significativamente la precisión de las predicciones sobre

eventos futuros y limitan el riesgo en el posterior proceso de toma de decisiones. Los resultados del caso de estudio mostraron el potencial del algoritmo propuesto para convertirse, si se combina con inspecciones visuales, en una herramienta eficaz y rápida para evaluar y diagnosticar el estado de los elementos estructurales después de eventos sísmicos. Ciertamente, una herramienta que pueda anticipar el resultado de un evento del que existen pocos datos, con un grado definido de confianza, podría ser particularmente útil en diferentes campos de la ingeniería.

Los futuros esfuerzos de investigación deben centrarse en diferentes formas de introducir los modelos físicos dentro de la arquitectura de la red neuronal artificial, de modo que los parámetros de los modelos basados en datos y en física estén estrechamente interconectados. También se debe explorar el uso de funciones de activación adaptativas.

Hipótesis 3: Una red neuronal recurrente (RNN) que incluye modelos basados en física en su ecuación y es entrenada usando métodos Bayesianos puede evitar las limitaciones ya mencionadas, proporcionando una metodología útil para el pronóstico. Del mismo modo, se pueden evitar las *"gated units"*, reduciendo la cantidad de pesos y sesgos requeridos.

Cuando los datos observados son secuenciales, el éxito de las RNN en todas sus diferentes versiones es incuestionable; sin embargo, su rendimiento depende en gran medida de la disponibilidad de grandes cantidades de datos de entrenamiento, y esto no es algo frecuente en ingeniería civil. Por otro lado, el proceso de entrenamiento de las RNN actuales se basa en la evaluación de una función de coste y el uso del algoritmo "*backpropagation*", lo que implica algunos inconvenientes bien conocidos como "*vanishing gradient*" y "*exploding gradient*", o el estancamiento en mínimos locales, proporcionando resultados variables.

En el Capítulo 7, se propuso una nueva RNN Bayesiana guiada por física y entrenada con ABC-SS. Los modelos basados en física se introducen en la ecuación de la RNN, lo que mitiga los problemas relacionados con la falta de datos y permite la extrapolación. Esto es especialmente importante en la realización de pronóstico sobre horizontes lejanos. Al mismo tiempo, el uso de ABC-SS como motor de aprendizaje se traduce en pesos y sesgos probabilísticos no paramétricos, regularización Bayesiana, ausencia de evaluación del gradiente y predicciones probabilísticas con cuantificación de incertidumbre. Los mismos principios se aplicaron a las redes neuronales LSTM, denominadas LSTM by ABC-SS en esta tesis, para evaluar si existe algún beneficio adicional en el uso de arquitecturas más complejas, incluso cuando no se lleva a cabo la evaluación del gradiente.

La RNN bayesiana propuesta se ha aplicado a dos experimentos de ingeniería estructural diferentes sobre la progresión del daño por fatiga en materiales compuestos y las aceleraciones en edificios de hormigón armado durante eventos sísmicos. Los resultados han demostrado que, si bien PG-BRNN by ABC-SS proporciona una precisión comparable a las RNN guiadas por física actuales, sus predicciones en diferentes ejecuciones del algoritmo

presentan muy poca desviación, lo que resulta en una opción más fiable. Además, en comparación con su competidor Bayesiano MC Dropout, el algoritmo propuesto mostró una cuantificación más precisa y realista de la incertidumbre. Finalmente, PG-LSTM by ABC-SS no presentó ninguna mejora sobre PG-BRNN by ABC-SS en términos de precisión o estabilidad, mientras que el tiempo de computación aumentó significativamente debido a la mayor cantidad de parámetros a entrenar. Esto demuestra que ABC-SS permite que la arquitectura básica de RNN capture dependencias a largo plazo, sin necesidad de *"gated units"*.

En relación a trabajos futuros, BRNN by ABC-SS podría explorarse como método de predicción en tiempo "*quasi-real*" para sistemas PHM a bordo, siempre que haya suficientes datos reales disponibles. Asimismo, PG-BRNN by ABC-SS ha demostrado potencial para convertirse en una herramienta de predicción "*in-situ*" para eventos sísmicos y/o réplicas en edificios, ayudando así a evaluar rápidamente su integridad estructural y la seguridad de los sistemas de servicios públicos. Por último, y en línea con las conclusiones de la Hipótesis 1, ABC-SS está limitado por la dimensión del espacio de parámetros y puede no ser adecuado para entrenar RNN muy grandes con una gran cantidad de neuronas, como las que se usan para el reconocimiento de video. Si bien las aplicaciones de ingeniería no suelen requerir redes neuronales con un elevado número de pesos y sesgos, es necesario explorar la posibilidad de ampliar la capacidad de entrenamiento de ABC-SS a arquitecturas más complejas. El uso de computación paralela también podría acelerar el método de muestreo en ABC-SS, ayudando a optimizar los recursos computacionales disponibles.

A

Research records

A.1 Journal articles

The methodologies and results presented in this thesis, along with other related contributions of the author, have been partially reflected in the following publications:

- Juan Fernández, Manuel Chiachío, Juan Chiachío, Rafael Muñoz, Francisco Herrera. Uncertainty quantification in Neural Networks by Approximate Bayesian Computation: Application to fatigue in composite materials. *Engineering Applications of Artificial Intelligence (2022), vol. 107, p. 104511* (I.F: 7.802, Rank:19-190=10%, 10 cites)
- Juan Fernández, Juan Chiachío, Manuel Chiachío, José Barros, Matteo Corbetta. Physics-guided Bayesian neural networks by ABC-SS: Application to reinforced concrete columns. *Engineering Applications of Artificial Intelligence (2023), vol. 119, p. 105790* (I.F: 7.802, Rank:19-190=10%)
- Manuel Chiachío, María Megía, Juan Chiachío, Juan Fernández, María L. Jalón. Structural digital twin framework: Formulation and technology integration. *Automation in Construction (2022), vol. 140, p. 104333* (I.F: 10.517, Rank:5-175=2.857%, 4 cites)
- Ali Saleh, Manuel Chiachío, Juan Fernández, Athanasios Kolios. Self-adaptive optimized maintenance of offshore wind turbines by intelligent Petri nets. *Reliability Engineering System Safety (2023), vol. 231, p. 109013* (I.F: 7.247, Rank:12-102=11.765%)

A.2 International conference and article

- 14th Annual Conference of the Prognostics and Health Management Society (Nov 1 – 4, 2022), Nashville, Tennessee, USA.
  - Juan Fernández, Juan Chiachío, Manuel Chiachío, Ali Saleh. Probabilistic Safety Assessment in Composite Materials using BNN by ABC-SS. *In Annual Conference of the PHM Society (2022, Vol. 14, No. 1).*

A.3 International research stay

- Research stay and collaboration with the Diagnostics & Prognostics Group in the Intelligent Systems Division at NASA Ames Research Center, Moffet Field, California, USA.
  - Application of physics-informed neural networks to predictions about the performance of real engineering components in aviation (unmanned aerial vehicles), using approximate Bayesian computation as the training method to allow for uncertainty quantification.

A.4 Open Access Code

A basic implementation in Python of Bayesian Neural Networks trained with Approximate Bayesian Computation by SubSet Simulation was uploaded to GitHub, and the link can be found below. Furthermore, two illustrative examples are provided to explain the learning process through the simulation levels, and also how the uncertainty varies when interpolating and extrapolating. Similar examples are also described in this thesis, specifically Illustrative Problem 1 and Illustrative Problem 2 in Chapter 5.

Link → https://github.com/J-Fdez/BNN-by-ABC-SS.git

References

[1] Jose Miguel Hernandez-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of Bayesian neural networks. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1861–1869, Lille, France, 07–09 Jul 2015. PMLR.

[2] Wayne Gill. *Ductility of rectangular Reinforced Concrete Columns with axial load*. PhD thesis, University of Canterbury, 1979.

[3] José Barros, Manuel Chiachío, Juan Chiachío, and Frank Cabanilla. Adaptive approximate Bayesian computation by subset simulation for structural model calibration. *Computer-Aided Civil and Infrastructure Engineering*, 37(6):726–745, 2022.

[4] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, Eftychios Protopapadakis, and Diego Andina. Deep learning for computer vision: a brief review. *Computational Intelligence and Neuroscience*, 2018:7068349, 2018.

[5] Shipra Arora and Rishi Singh. Automatic speech recognition: a review. *International Journal of Computer Applications*, 60:34–44, 12 2012.

[6] Oliver Sturman, Lukas von Ziegler, Christa Schläppi, Furkan Akyol, Mattia Privitera, Daria Slominski, Christina Grimm, Laetitia Thieren, Valerio Zerbi, Benjamin Grewe, et al. Deep learning-based behavioral analysis reaches human accuracy and is capable of outperforming commercial solutions. *Neuropsychopharmacology*, 45(11):1942–1952, 2020.

[7] W.A. Awad and S.M. Elseuofi. Machine learning methods for spam e-mail classification. *International Journal of Computer Science Information Technology*, 3(1):173–184, 2011.

[8] I. Sadgali, N. Sael, and F. Benabbou. Performance of machine learning techniques in the detection of financial frauds. *Procedia Computer Science*, 148:45–54, 2019.

[9] Laura Rodríguez Carlos-Roca, Isabelle Hupont Torres, and Carles Fernández Tena. Facial recognition application for border control. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2018.

[10] Hojjat Adeli. Neural networks in civil engineering: 1989–2000. *Computer-Aided Civil and Infrastructure Engineering*, 16(2):126–142, 2001.

[11] Dipali Pandya and Dhaval Shah. Experimentation and its prediction of process parameters effects on elongation in tensile test of aisi 1008 steel using ann model. *Procedia Technology*, 14:282–289, 2014.

[12] Abd Rashid Abd Aziz and Kau-Fui Vincent Wong. A neural-network approach to the determination of aquifer parameters. *Groundwater*, 30(2):164–166, 1992.

[13] Semet Çelik and Özcan Tan. Determination of preconsolidation pressure with artificial neural network. *Civil Engineering and Environmental Systems*, 22(4):217–231, 2005.

[14] Margaret W Emsley, David J Lowe, A Roy Duff, Anthony Harding, and Adam Hickson. Data modelling and the application of a neural network approach to the prediction of total construction costs. *Construction Management & Economics*, 20(6):465–472, 2002.

[15] Kush R Varshney and Homa Alemzadeh. On the safety of machine learning: Cyber-physical systems, decision sciences, and data products. *Big data*, 5(3):246–255, 2017.

[16] Xiaoge Zhang and Sankaran Mahadevan. Bayesian neural networks for flight trajectory prediction and safety assessment. *Decision Support Systems*, 131:113246, 2020.

[17] Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning*, 110(3):457–506, 2021.

[18] James L Beck. Bayesian system identification based on probability logic. *Structural Control and Health Monitoring*, 17(7):825–847, 2010.

[19] Stefan Depeweg, Jose-Miguel Hernandez-Lobato, Finale Doshi-Velez, and Steffen Udluft. Decomposition of uncertainty in Bayesian deep learning for efficient and risk-sensitive learning. In *International Conference on Machine Learning*, pages 1184–1193. PMLR, 2018.

[20] Z Ghahramani. Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553):452–459, 2015.

[21] Wray L. Buntine and Andreas S. Weigend. Bayesian back-propagation. *Complex Systems*, 5:603–643, 1991.

[22] David J. C. MacKay. A practical Bayesian framework for backpropagation networks. *Neural Computation*, 4(3):448–472, 1992.

[23] R. M. Neal. Bayesian training of backpropagation networks by the hybrid Monte Carlo method. Technical Report CRG-TR-92-1, Department of Computer Science, U. of Toronto, April 1992.

[24] Radford M. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag, Berlin, Heidelberg, 1996.

[25] Jouko Lampinen and Aki Vehtari. Bayesian approach for neural networks—review and case studies. *Neural Networks*, 14(3):257 – 274, 2001.

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

[27] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 14(5), 2013.

[28] H. Wang, X. Bai, and J. Tan. Uncertainty quantification of bearing remaining useful life based on convolutional neural network. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 2893–2900, 2020.

[29] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1613–1622, Lille, France, 07–09 2015. PMLR.

[30] Shaocheng Jia, Yun Yue, Zi Yang, Xin Pei, and Yashen Wang. Travelling modes recognition via bayes neural network with bayes by backprop algorithm. In *CICTP 2020*, pages 3994–4004. 2020.

[31] Maximilian Benker, Lukas Furtner, Thomas Semm, and Michael F. Zaeh. Utilizing uncertainty information in remaining useful life estimation via Bayesian neural networks and Hamiltonian Monte Carlo. *Journal of Manufacturing Systems*, In Press, 2020.

[32] Daniel Levy, Jascha Sohl-dickstein, and Matt Hoffman. Generalizing Hamiltonian Monte Carlo with neural networks. In *ICLR 2018 Conference*, 2018.

[33] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[34] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. PMLR, 2013.

[35] Lu Lu. Dying relu and initialization: Theory and numerical examples. *Communications in Computational Physics*, 28(5):1671–1706, 2020.

[36] Steven L Brunton, Bernd R Noack, and Petros Koumoutsakos. Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics*, 52:477–508, 2020.

[37] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[38] Rachna Vaish, UD Dwivedi, Saurabh Tewari, and SM Tripathi. Machine learning applications in power system fault diagnosis: Research advancements and perspectives. *Engineering Applications of Artificial Intelligence*, 106:104504, 2021.

[39] Haibo He and Edwardo A Garcia. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284, 2009.

[40] Pamela J Haley and DONALD Soloway. Extrapolation limitations of multilayer feedforward neural networks. In *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, volume 4, pages 25–30. IEEE, 1992.

[41] Andrew White, Malachi Tolman, Howard Thames, Hubert Withers, Kathy Mason, and Mark Transtrum. The limitations of model-based experimental design and parameter estimation in sloppy systems. *PLOS Computational Biology*, 12, 2016.

[42] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[43] Guofei Pang, Lu Lu, and George Em Karniadakis. fpinns: Fractional physics-informed neural networks. *SIAM Journal on Scientific Computing*, 41(4):A2603–A2626, 2019.

[44] Navid Zobeiry and Keith D Humfeld. A physics-informed machine learning approach for solving heat transfer equation in advanced manufacturing and engineering applications. *Engineering Applications of Artificial Intelligence*, 101:104232, 2021.

[45] Florian Arnold and Rudibert King. State–space modeling for control based on physics-informed neural networks. *Engineering Applications of Artificial Intelligence*, 101:104195, 2021.

[46] Hongwei Guo, Xiaoying Zhuang, and Timon Rabczuk. A deep collocation method for the bending analysis of kirchhoff plate. *arXiv preprint arXiv:2102.02617*, 2021.

[47] Mohammad Amin Nabian, Rini Jasmine Gladstone, and Hadi Meidani. Efficient training of physics-informed neural networks via importance sampling. *Computer-Aided Civil and Infrastructure Engineering*, 2021.

[48] Hongyu Sun, Lisha Peng, Junming Lin, Shen Wang, Wei Zhao, and Songling Huang. Microcrack defect quantification using a focusing high-order sh guided wave emat: the physics-informed deep neural network guwnet. *IEEE Transactions on Industrial Informatics*, 18(5):3235–3247, 2021.

[49] Hongyu Sun, Lisha Peng, Songling Huang, Shisong Li, Yue Long, Shen Wang, and Wei Zhao. Development of a physics-informed doubly fed cross-residual deep neural network for high-precision magnetic flux leakage defect size estimation. *IEEE Transactions on Industrial Informatics*, 18(3):1629–1640, 2021.

[50] Russell Stewart and Stefano Ermon. Label-free supervision of neural networks with physics and domain knowledge. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[51] Wannes De Groote, Edward Kikken, Erik Hostens, Sofie Van Hoecke, and Guillaume Crevecoeur. Neural network augmented physics models for systems with partially unknown dynamics: Application to slider-crank mechanism. *IEEE/ASME Transactions on Mechatronics*, 2021.

[52] Jinjiang Wang, Yilin Li, Rui Zhao, and Robert X Gao. Physics guided neural network for machining tool wear prediction. *Journal of Manufacturing Systems*, 57:298–310, 2020.

[53] Ruiyang Zhang, Yang Liu, and Hao Sun. Physics-guided convolutional neural network (phycnn) for data-driven seismic response modeling. *Engineering Structures*, 215:110704, 2020.

[54] Uduak Inyang-Udoh and Sandipan Mishra. A physics-guided neural network dynamical model for droplet-based additive manufacturing. *IEEE Transactions on Control Systems Technology*, 2021.

[55] Anuj Karpatne, William Watkins, Jordan Read, and Vipin Kumar. Physics-guided neural networks (pgnn): An application in lake temperature modeling. *arXiv preprint arXiv:1710.11431*, 2017.

[56] Xiaowei Jia, Jared Willard, Anuj Karpatne, Jordan Read, Jacob Zwart, Michael Steinbach, and Vipin Kumar. 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*, pages 558–566. SIAM, 2019.

[57] Liu Yang, Xuhui Meng, and George Em Karniadakis. B-pinns: Bayesian physics-informed neural networks for forward and inverse pde problems with noisy data. *Journal of Computational Physics*, 425:109913, 2021.

[58] Dongkun Zhang, Lu Lu, Ling Guo, and George Em Karniadakis. Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems. *Journal of Computational Physics*, 397:108850, 2019.

[59] Falih N Jabir B. Dropout, a basic and effective regularization method for a deep learning model: a case study. *Indonesian Journal of Electrical Engineering and Computer Science*, 24(2):1009–1016, 2021.

[60] Arka Daw, R Quinn Thomas, Cayelan C Carey, Jordan S Read, Alison P Appling, and Anuj Karpatne. 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*, pages 532–540. SIAM, 2020.

[61] Muhammed A Hassan, Nadjem Bailek, Kada Bouchouicha, and Samuel Chukwujindu Nwokolo. Ultra-short-term exogenous forecasting of photovoltaic power production using genetically optimized non-linear auto-regressive recurrent neural networks. *Renewable Energy*, 171:191–209, 2021.

[62] Arsalan Lambay, Ying Liu, Phillip Morgan, and Ze Ji. A data-driven fatigue prediction using recurrent neural networks. In *2021 3rd International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, pages 1–6. IEEE, 2021.

[63] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[64] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[65] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649. Ieee, 2013.

[66] Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 606–615, 2016.

[67] Daopeng Wang, Jifei Fan, Hanliang Fu, and Bing Zhang. Research on optimization of big data construction engineering quality management based on rnn-lstm. *Complexity*, 2018, 2018.

[68] Yuejian Chen, Meng Rao, Ke Feng, and Ming J Zuo. Physics-informed lstm hyperparameters selection for gearbox fault detection. *Mechanical Systems and Signal Processing*, 171:108907, 2022.

[69] Manu Lahariya, Karami Farzaneh, Chris Develder, and Guillaume Crevecoeur. Physics informed lstm network for flexibility identification in evaporative cooling system. *IEEE Transactions on Industrial Informatics*, 2022.

[70] Benjamin Wu, Oliver Hennigh, Jan Kautz, Sanjay Choudhry, and Wonmin Byeon. Physics informed rnn-dct networks for time-dependent partial differential equations. *arXiv preprint arXiv:2202.12358*, 2022.

[71] Parisa Shokouhi, Vikas Kumar, Sumedha Prathipati, Seyyed A Hosseini, Clyde Lee Giles, and Daniel Kifer. Physics-informed deep learning for prediction of co2 storage site response. *Journal of Contaminant Hydrology*, 241:103835, 2021.

[72] Pu Ren, Chengping Rao, Yang Liu, Jian-Xun Wang, and Hao Sun. Phycrnet: Physics-informed convolutional-recurrent network for solving spatiotemporal pdes. *Computer Methods in Applied Mechanics and Engineering*, 389:114399, 2022.

[73] Renato G Nascimento, Kajetan Fricke, and Felipe AC Viana. A tutorial on solving ordinary differential equations using python and hybrid physics-informed neural network. *Engineering Applications of Artificial Intelligence*, 96:103996, 2020.

[74] Renato G Nascimento, Matteo Corbetta, Chetan S Kulkarni, and Felipe AC Viana. Hybrid physics-informed neural networks for lithium-ion battery modeling and prognosis. *Journal of Power Sources*, 513:230526, 2021.

[75] Renato Giorgiani Nascimento and Felipe AC Viana. Cumulative damage modeling with recurrent neural networks. *AIAA Journal*, 58(12):5459–5471, 2020.

[76] Renato Giorgiani Nascimento and Felipe AC Viana. Fleet prognosis with physics-informed recurrent neural networks. *arXiv preprint arXiv:1901.05512*, 2019.

[77] Xiaowei Jia, Jacob Zwart, Jeffrey Sadler, Alison Appling, Samantha Oliver, Steven Markstrom, Jared Willard, Shaoming Xu, Michael Steinbach, Jordan Read, et al. Physics-guided recurrent graph networks for predicting flow and temperature in river networks. *arXiv preprint arXiv:2009.12575*, 2020.

[78] Marco Gori and Alberto Tesi. On the problem of local minima in backpropagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(1):76–86, 1992.

[79] Lei Wang, Yafei Ma, Jianren Zhang, Xuhui Zhang, and Yongming Liu. Uncertainty quantification and structural reliability estimation considering inspection data scarcity. *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part A: Civil Engineering*, 1(2):04015004, 2015.

[80] Murad Al Qurishee, Weidong Wu, Babatunde Atolagbe, Joseph Owino, Ignatius Fomunung, and Mbakisya Onyango. Creating a dataset to boost civil engineering deep learning research and application. *Engineering*, 12(3):151–165, 2020.

[81] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

[82] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[83] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[84] Manuel Chiachio, James L. Beck, Juan Chiachio, and Guillermo Rus. Approximate Bayesian computation by subset simulation. *SIAM Journal on Scientific Computing*, (3):A1339—-A1358, 2014.

[85] Anis Ben Abdessalem, Nikolaos Dervilis, David Wagg, and Keith Worden. Model selection and parameter estimation in structural dynamics using approximate Bayesian computation. *Mechanical Systems and Signal Processing*, 99:306–325, 2018.

[86] Ramesh Talreja. Damage and fatigue in composites–a personal account. *Composites Science and Technology*, 68(13):2585–2591, 2008.

[87] Juan Chiachío, Manuel Chiachío, Abhinav Saxena, Shankar Sankararaman, Guillermo Rus, and Kai Goebel. Bayesian model selection and parameter estimation for fatigue damage progression models in composites. *International Journal of Fatigue*, 70:361–373, 2015.

[88] Michael Betancourt. A conceptual introduction to Hamiltonian Monte Carlo. *arXiv preprint arXiv:1701.02434*, 2017.

[89] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[90] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

[91] Lingxue Zhu and Nikolay Laptev. Deep and confident prediction for time series at uber. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 103–110. IEEE, 2017.

[92] Warren Mcculloch and Walter Pitts. A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:127–147, 1943.

[93] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[94] Ian Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28, pages 1319–1327, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[95] T. Bayes. An essay towards solving a problem in the doctrine of chances. *Phil. Trans. of the Royal Soc. of London*, 53:370–418, 1763.

[96] Pierre-Simon Laplace. *Théorie analytique des probabilités*. Courcier, Paris, 1812.

[97] H. Jeffreys. *Theory of Probability*. Oxford, Oxford, England, third edition, 1961.

[98] Richard T. Cox. Probability, frequency, and reasonable expectation. *American Journal of Physics*, 14(2):1–13, 1946.

[99] Radford M. Neal. Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, Department of computer science. University of Toronto, 1993.

[100] W.R. Gilks, S. Richardson, and D.J. Spiegelhalter. *Markov chain Monte Carlo in practice*. Chapman and Hall/CRC, Boca Raton, 1996.

[101] Jean Michel Marin, Pierre Pudlo, Christian P. Robert, and Robin Ryder. Approximate Bayesian computational methods. *Statistics and Computing*, pages 1167—-1180, 2012.

[102] A.M. Santoso, K.K. Phoon, and S.T. Quek. Modified metropolis–hastings algorithm with reduced chain correlation for efficient subset simulation. *Probabilistic Engineering Mechanics*, 26(2):331 – 341, 2011.

[103] Paul Fearnhead and Dennis Prangle. Constructing summary statistics for approximate Bayesian computation: Semi-automatic approximate Bayesian computation. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 74(3):419–474, 2012.

[104] Paul Marjoram, John Molitor, Vincent Plagnol, and Simon Tavaré. Markov chain Monte Carlo without likelihoods. *Proceedings of the National Academy of Sciences*, 100(26):15324–15328, 2003.

[105] Meïli Baragatti, Agnès Grimaud, and Denys Pommeret. Likelihood-free parallel tempering. *Statistics and Computing*, 23(4):535–549, 2013.

[106] Mark A Beaumont, Jean-Marie Cornuet, Jean-Michel Marin, and Christian P Robert. Adaptive approximate Bayesian computation. *Biometrika*, 96(4):983–990, 2009.

[107] Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. An adaptive sequential Monte Carlo method for approximate Bayesian computation. *Statistics and Computing*, 22(5):1009–1020, 2012.

[108] Ritabrata Dutta, Marcel Schoengens, Lorenzo Pacchiardi, Avinash Ummadisingu, Nicole Widmer, Jukka-Pekka Onnela, and Antonietta Mira. Abcpy: A high-performance computing perspective to approximate Bayesian computation. *arXiv preprint arXiv:1711.04694*, 2017.

[109] Panagiotis Hadjidoukas. Accessed Feb 25, 2021.

[110] Siu Kui Au and James L. Beck. Estimation of small failure probabilities in high dimensions by subset simulation. *Probabilistic Engineering Mechanics*, 16(4):263 – 277, 2001.

[111] J. Ching, S.K. Au, and J.L. Beck. Reliability estimation for dynamical systems subject to stochastic excitation using subset simulation with splitting. *Computer Methods in Applied Mechanics and Engineering*, 194(12):1557 – 1579, 2005.

[112] S.K. Au, J. Ching, and J.L. Beck. Application of subset simulation methods to reliability benchmark problems. *Structural Safety*, 29(3):183–193, 2007.

[113] Andrew White, Malachi Tolman, Howard D Thames, Hubert Rodney Withers, Kathy A Mason, and Mark K Transtrum. The limitations of model-based experimental design and parameter estimation in sloppy systems. *PLoS Computational Biology*, 12(12):e1005227, 2016.

[114] Jared Willard, Xiaowei Jia, Shaoming Xu, Michael Steinbach, and Vipin Kumar. Integrating physics-based modeling with machine learning: A survey. *arXiv preprint arXiv:2003.04919*, 1(1):1–34, 2020.

[115] Rahul Rai and Chandan K Sahu. Driven by data or derived through physics a review of hybrid physics guided machine learning techniques with cyber-physical system (cps) focus. *IEEE Access*, 8:71050–71073, 2020.

[116] Seokyoung Hong, Nahyeon An, Hyungtae Cho, Jongkoo Lim, In-Su Han, Il Moon, and Junghwan Kim. A dynamic doft sensor based on hybrid neural networks to improve early off-spec detection. *Engineering with Computers*, pages 1–11, 2022.

[117] Arka Daw, Anuj Karpatne, William Watkins, Jordan Read, and Vipin Kumar. Physics-guided neural networks (pgnn): An application in lake temperature modeling. *arXiv preprint arXiv:1710.11431*, 2017.

[118] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[119] Roozbeh Razavi-Far, Shiladitya Chakrabarti, and Mehrdad Saif. Multi-step-ahead prediction techniques for lithium-ion batteries condition prognosis. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 004675–004680. IEEE, 2016.

[120] S. Hochreiter. Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut für Informatik, Lehrstuhl Prof. Brauer, Technische Universität München, 1991.

[121] Konstantin M. Zuev, James L. Beck, Siu-Kui Au, and Lambros S. Katafygiotis. Bayesian post-processor and other enhancements of subset simulation for estimating failure probabilities in high dimensions. *Computers Structures*, 92-93:283–296, 2012.

[122] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 2(1-3):37–52, 1987.

[123] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[124] Lenka Markovičová, Viera Zatkalíková, and Patrícia Hanusová. Carbon fiber polymer composites. In *Conference Quality Production Improvement–CQPI*, volume 1, pages 276–280, 2019.

[125] Abhinav Saxena, Kai Goebel, Cecilia Larrosa, and Fu-Kuo Chank. CFRP Composites Data Set, NASA Ames Prognostics Data Repository.

[126] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal

Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[127] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[128] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5-6):602–610, 2005.

[129] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[130] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pages 1050–1059. PMLR, 2016.

[131] Junfu Chen, Dechang Pi, Zhiyuan Wu, Xiaodong Zhao, Yue Pan, and Qiang Zhang. Imbalanced satellite telemetry data anomaly detection model based on Bayesian lstm. *Acta Astronautica*, 180:232–242, 2021.

[132] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[133] Cecilia Larrosa Wilson and F.-K Chang. Real time in-situ damage classification, quantification and diagnosis for composite structures. *19th International Congress on Sound and Vibration 2012, ICSV 2012*, 4:2696–2704, 01 2012.

[134] François Chollet et al. Keras. `https://keras.io`, 2015. Accessed Mar 6, 2021.

[135] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[136] Srinivas Sriramula and Marios K Chryssanthopoulos. Quantification of uncertainty modelling in stochastic analysis of frp composites. *Composites Part A: Applied Science and Manufacturing*, 40(11):1673–1684, 2009.

[137] Guido Van Rossum and Fred L Drake Jr. *Python Reference Manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[138] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, 2016.

[139] Michael Berry, Myles Parrish, and Marc Eberhard. Peer structural performance database user's manual (version 1.0). *University of California, Berkeley*, 2004.

[140] Yinhao Zhu and Nicholas Zabaras. Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification. *Journal of Computational Physics*, 366:415–447, 2018.

[141] Madhu M Karthik and John B Mander. Stress-block parameters for unconfined and confined concrete based on a unified stress-strain model. *Journal of Structural Engineering*, pages 270–273, 2011.

[142] John B Mander, N. Priestley, and Robert Park. Theoretical stress-strain model for confined concrete. *Journal of Structural Engineering*, 1988.

[143] FEMA. *FEMA P-58-1: Seismic performance assessment of buildings. Volume 1–methodology*, volume 10. Federal Emergency Management Agency, 2012.

[144] Eyitayo A. Opabola and Kenneth J. Elwood. Collapse performance of nominally identical nonductile circular columns susceptible to failure-mode variability. *Journal of Structural Engineering*, 147(6):04021069, 2021.

[145] Chia-Ming Chang, Tzu-Kang Lin, and Chih-Wei Chang. Applications of neural network models for structural health monitoring based on derived modal properties. *Measurement*, 129:457–470, 2018.

[146] Yang Yu, Chaoyue Wang, Xiaoyu Gu, and Jianchun Li. A novel deep learning-based method for damage identification of smart building structures. *Structural Health Monitoring*, 18(1):143–163, 2019.

[147] Prateek Pratap and Santiago Pujol. Dynamic tests of an idealized long-period structure. *DEEDS*, 2021.

[148] E. O. Brigham and R. E. Morrow. The fast fourier transform. *IEEE Spectrum*, 4(12):63–70, 1967.