Soft Computing (2019) 23:9937–9953
https://doi.org/10.1007/s00500-018-3549-3

METHODOLOGIES AND APPLICATION

Deep evolutionary modeling of condition monitoring data in marine propulsion systems

Alberto Diez-Olivan[1] · Jose A. Pagan[2] · Ricardo Sanz[3] · Basilio Sierra[4]

Published online: 5 October 2018
© Springer-Verlag GmbH Germany, part of Springer Nature 2018

Abstract
In many complex industrial scenarios where condition monitoring data are involved, data-driven models can highly support maintenance tasks and improve assets' performance. To infer physical meaningful models that accurately characterize assets' behaviors across a wide range of operating conditions is a difficult issue. Usually, data-driven models are in black-box format, accurate but too complex to intelligibly explain the inherent physics of the process and lacking in conciseness. This study presents a deep evolutionary-based approach to optimally model and predict physical behaviors in industrial assets from operational data. The evolutionary modeling process is combined with long short-term memory networks, which are trained on estimations made by the evolutionary physical model and then used to predict sequences of data over a number of time steps. The likelihood of behaviors of interest is assessed by means of the resulting sequences of residuals, and a resulting score is computed over time. The proposed approach is applied to model and predict a set of temperatures related to a marine propulsion system, anticipating anomalies and changes in operating conditions. It is demonstrated that deep evolutionary modeling results are quite satisfactory for prognostics and obtained physical models are practical and easy to understand.

Keywords Evolutionary modeling · LSTM networks · Behavior characterization · Condition monitoring · Genetic programming · Marine propulsion systems · Machine learning

1 Introduction

In many complex industrial scenarios, physical models are used to measure assets' performance and to estimate their health status. Nevertheless, they are complex, time-consuming and costly. To define a physical model from intuition requires a wide experience in the underlying physics of the process to be described. Once a model is proposed, it has to be experimentally evaluated. When the obtained deviation between the model prediction and the experimental data is within reasonable error limits (arbitrarily determined), the model is considered valid. This inductive process assumes the model validity, while there are no contradictory experimental cases found.

Physical formulation and simulation of systems and processes is usually accomplished when no data are available, e.g., to support system design (Tian and Voskuijl 2015). When condition monitoring data are available, data-driven models can be used to automatically establish linear and non-linear relations between variables involved in the physical process under study by means of a mathematical language (Yin et al. 2015). Once the empirical model is built, it can be employed to predict the dependent variable that characterizes the asset state in the future whenever the initial conditions are known. There are several techniques based on this approach:

Communicated by V. Loia.

✉ Alberto Diez-Olivan
alberto.diez@tecnalia.com

Jose A. Pagan
japagan@navantia.es

Ricardo Sanz
ricardo.sanz@upm.es

Basilio Sierra
b.sierra@ehu.eus

[1] Tecnalia Research and Innovation, Donostia - San Sebastián, Guipúzkoa, Spain

[2] Diesel Engine Factory, Diagnose Engineering and Product Development, Cartagena, Spain

[3] Autonomous Systems Laboratory, Universidad Politécnica de Madrid, Madrid, Spain

[4] Department of Computer Sciences and Artificial Intelligence, UPV/EHU, Donostia - San Sebastián, Spain

finite-element method, regression, interpolation, etc. (Chapra and Canale 2012).

Advanced mathematical modeling from data can be accomplished in many different ways, usually applying machine learning algorithms and soft computing techniques (Azar and Vaidyanathan 2015; Zhu and Azar 2015). They provide a very powerful statistical tool to solve complex real-world problems where data are involved, i.e., the prediction of fire in forests by using different neural network models (Al_Janabi et al. 2017), the detection and prevention of cyber-attacks against information systems by means of fuzzy logic and a risk analysis model (Al-Janabi 2017) or the analysis of asset operational data in industrial scenarios by means of predictive models (Rousseaux 2016), as, for instance, kernel methods and support vector machines (Diez-Olivan et al. 2018) or a combination of unsupervised methods and fuzzy logic (Diez-Olivan et al. 2017). However, this kind of techniques usually provides black-box solutions, lacking in conciseness and clarity, they require a deep expert knowledge to successfully address complex problems, or they can only be applied to relatively simple physical processes.

From existing methods, evolutionary-based algorithms provide a flexible, efficient and robust optimization and search strategy to infer physical models from data (Meerschaert 2013). There are four main types of evolutionary-based algorithms: genetic algorithms (GA), genetic programming (GP), evolutionary programming (EP) and evolutionary strategies (ES) (Espinosa and Ayala-Solares 2016; Brabazon et al. 2015). In addition to these methods, some combinations and extensions of previous ones can be found in the literature. Memetic algorithms (MA), for instance, emerged as a combination of GA and local search (LS) methods and are now becoming very popular to solve a broad range of different problems. Another interesting example is grammar-guided genetic programming (GGGP), which is an extension of traditional GP systems and it always generates valid solutions, represented as individuals of the population or points that belong to the search space.

Time series prediction is another big issue when detecting anomalies in condition monitoring data. Traditional strategies use statistical measures such as moving average over a time window, ARIMA, kalman filter and cumulative sum (Box et al. 2015). Regression models fitted to nonstationary data can better represent more complex, nonlinear dependencies with other related features. To that concern, Gaussian process regression (Rasmussen and Williams 2006) and multilayer perceptron (MLP) networks for regression (Rojas 2013) are two very popular examples of prognostics models. In a similar way to which it is done in MLP networks, deep learning methods can be seen as a cascade of many layers of processing units that combine the predictor features to approximate the target feature (LeCun et al. 2015). In convolutional neural networks (CNNs), for instance, the convolutional layer is the core building block, which mainly consists of a set of learnable filters (or kernels) that compute dot products between the input at any position of the data matrix and the entries of the filter, usually extended through the full depth of the input volume (Munteanu et al. 2017). Recurrent neural networks (RNNs) present some interesting properties for time series forecasting like loops in them, allowing information to persist (Mikolov et al. 2010). They are powerful and increasingly popular models for learning from varying-length sequence data, particularly those using LSTM hidden units (Hochreiter and Schmidhuber 1997). LSTM networks for anomaly/fault detection in time series have demonstrated very good accuracy (Hayton et al. 2007; Guo et al. 2017).

Temporal anomaly detection approaches usually learn models that best fit time series to compute errors when comparing new, incoming data to predicted values. Some recent works have dealt with LTSM networks for anomaly detection in time series (Malhotra et al. 2015; Shipmon et al. 2017). However, they have not yet been combined with an understandable physical modeling of condition monitoring data for prognostics, anticipating anomalous data sequences over time.

This work is an effort to provide a deep evolutionary method able to efficiently and accurately model physical behaviors and to predict anomalous sequences of data. Models learned can be used for condition monitoring of the asset and to optimize its performance. Moreover, they can highly improve knowledge about the behavior and physics of the process or asset under study, providing mathematical expressions easy to interpret and to apply. One main advantage of the proposed learning framework is that the comprehensive evolutionary modeling process is combined with accurate deep learning strategies. Another key contribution is the method used to finally estimate the health score of the asset under study, which is easily interpretable for maintainers and it offers a real-time tool for maintenance optimization. The proposed approach is applied over real operational data acquired from a reduction gear of a CODOG (combined diesel or gas) marine propulsion system.

The rest of the article is organized as follows. Section 2 presents the deep evolutionary modeling approach used to model physical behaviors from condition monitoring data, including the anomaly prediction strategy. In Sect. 3, the experimental setup is described and results obtained are discussed. Finally, the conclusions achieved in this study are given in the last section.

2 Deep evolutionary modeling approach

2.1 Evolutionary physical modeling

By defining a simple mathematical grammar, a great number of physical models can be inferred from data (Whigham et al. 1995). Thanks to both linear and nonlinear mathematical functions, a wide variety of industrial processes can be modeled from operational data: thermodynamic, kinematics, dynamics, chemical models, etc. Furthermore, by means of exponential functions, the representation of several linear differential equation solutions can be accomplished.

The technique that allows searching both the coefficients that best fit data and the model itself is known as symbolic regression. Symbolic regression is based on concepts known some decades ago, although the symbolic regression concept is relatively recent (Koza 1992). Technological revolution experimented by computational systems made it possible to envisage mathematical expressions search and fitting, which are computationally expensive, with a progressive success.

The first symbolic regression approaches are based on extensions of classical regression methods. They are called stepwise regression methods. In these methods, the targeted mathematical function is iteratively modified by adding and removing mathematical terms (Dallemand 1958; Westervelt 1960). Stepwise regression methods are supported by statistical concepts like regression analysis. Therefore, initial data need to satisfy statistical requirements imposed by the statistical tests used to validate generated models.

Other symbolic regression methods are the so-called data-driven heuristic model generation methods. These methods appeared later and are inspired by the human cognitive process responsible for the physical laws discovery. To do this, these methods employ heuristic rules in order to drive the search process to find the mathematical expression that best fit data (Langley and Zytkow 1989).

Both stepwise regression methods and data-driven heuristic approaches are only able to obtain limited models in terms of mathematical complexity. A priori information about the model to be constructed is needed in order to state the more complex mathematical expressions search (Harrell 2001). Moreover, the most of these techniques require an exhaustive initial data set, coming from a factorial experiment design, where all the possible coupled variable modifications have to be considered. This restriction makes impossible their use on initial data sets with other formats.

The raise of the information technologies has allowed the use of more sophisticated search techniques like evolutionary computation (Gong et al. 2015). By means of evolutionary computation, the search problem, inherent in symbolic regression, can be faced avoiding the existing constraints in other approaches. Nowadays, evolutionary computation employment has been a revolution into the symbolic regression domain (Haeri et al. 2017). Its use has been extended to a great number of problems, from purely mathematical to engineering and industrial applications. Thus, different GP-based modeling approaches have been recently proposed by different authors, e.g., using vibro-acoustic condition monitoring data in machining processes (Garg et al. 2015), operational data to model wind turbine dynamics (La Cava et al. 2016), time series to detect structural breaks (Doerr et al. 2017) or to model gas compressor performance in oil and gas industry for maintenance optimization (Safiyullah et al. 2016) and addressing a pharmaceutical problem by optimizing a zero-order release matrix tablet (Barmpalexis et al. 2011). Symbolic regression can be therefore considered as a very useful tool to assist scientists in the physical laws deduction process.

The symbolic regression method for modeling different behaviors of complex systems by means of their operational parameters is based on the work proposed by Carrascal et al. (2009, 2010), addressing the automatic physical model design problem by an evolutionary approach. Evolution-based algorithms are considered to be the most flexible, efficient and robust of all optimization and search algorithms known to computer science. Therefore, they are becoming widely used to solve a broad range of problems of different nature and characteristics (Bentley 1999).

The overlying idea consists on inferring mathematical models that characterize behaviors of interest. Given a target feature, $\boldsymbol{y} = (y_1, \ldots, y_n)$, a set of operands, {sine, tangent, cosine, exp, log, abs, pow, sqrt, $*$, /, $+$, $-$}, a set of m input features, $X = \{X_1, \ldots, X_m\}$ where each feature X_i can take a value from its own set of possible values χ_i, and n feature vectors or instances, $\boldsymbol{x_i} = (x_1, \ldots, x_m) \in \chi = (\chi_1, \ldots, \chi_m)$, with $i = 1, \ldots, n$, the evolutionary process finds the derivation tree that best fits the model, $f(X) = \hat{\boldsymbol{y}} \approx \boldsymbol{y}$, by optimally combining features and operands and considering a fitness value that is the mean squared error (MSE, see Eq. 7) computed by the formula that is being tested. Each individual in the population codifies a valid derivation tree, which represents a mathematical expression. It is important for two individuals representing similar solutions to have similar codifications. This principle avoids a strictly random searching process. Consider, for instance, the formula $f(X) = X_1 + \sqrt{X_2} - \frac{X_3}{\sin X_4}$, which can be represented by the derivation tree shown in Fig. 1.

During the evolutionary process, previously selected individuals are crossed with probability 0.9. Data are split into training (60% of the total data size) and test (30% of the total data size, holding the other 10% for validating the LSTM-based model) sets, aiming to avoid a possible overfitting problem. A maximum number of nodes to be used in mathematical expressions are also considered. From a mathematical point of view, this restriction avoids obtaining too complex expressions.

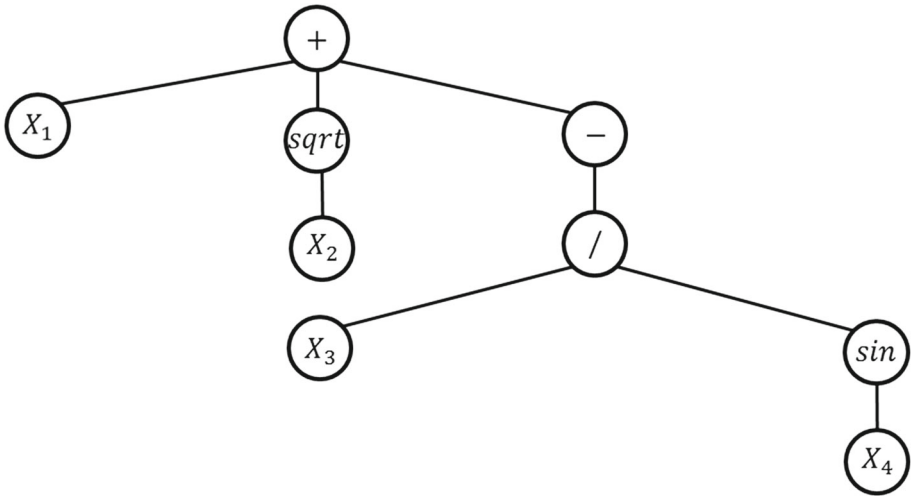

Fig. 1 An example of evolutionary modeling derivation tree that represents the formula $f(X) = X_1 + \sqrt{X_2} - \frac{X_3}{\sin X_4}$

The mathematical context-free grammar defined allows combining a great number of mathematical operands to create behavioral models from data. Thus, the automatic generation of complex mathematical expressions that represent behaviors of interest can be accomplished. The evolutionary modeling process outcome is a mathematical model in which all the constants have been solved. The corresponding flowchart can be seen in Fig. 2.

The GP module starts generating an initial population of mathematical expressions that are evaluated by the GA module. The evaluation of candidate expressions is performed by computing their fitness value, which measures how accurately the expression solves the problem. After selecting the best candidates of the current population, a new population is generated applying the following operations:

- *Reproduction* an existing expression is copied to next population.
- *Crossover* different expressions are combined by crossover to create a new expression.
- *Mutation* a component in an existing expression is changed to create a new expression.

The evolutionary process finishes when the maximum number of evolutions is reached.

The details of the configuration parameters of the modeling process are presented in Table 1. Different values were tested prior to establishing the proposed parameter values, concluding that those were the most appropriate for the problem under study in terms of accuracy and performance of the resulting models.

Table 1 Evolutionary modeling configuration parameters

Parameter	Value
Population size	500
Number of evolutions	100
Constant range	0–100
Operands	{sine, tangent, cosine, exp, log, abs, pow, sqrt, ∗, /, +, −}
Maximum nodes	21
Reproduction rate	0.1
Mutation rate	0.1
Crossover rate	0.9

2.2 LSTM-based anomaly prediction

RNNs are networks with loops in them, allowing information to persist (Mikolov et al. 2010). The idea is to let every step of a RNN pick information to look at from some larger collection of information. Therefore, they can be thought of as multiple copies of the same network, each passing a message to a successor. This chain-like nature reveals that RNNs are intimately related to sequences and lists. They are powerful and increasingly popular models for learning from varying-length sequence data, particularly those using Long Short-Term Memory (LSTM) hidden units (Hochreiter and Schmidhuber 1997). LSTM neural networks overcome the vanishing gradient problem that is present in RNNs.

The key to LSTMs is the memory cells, whose states are carefully regulated by structures called gates. They are composed out of a sigmoid neural net layer, which outputs numbers between zero and one to describe how much of each component should be let through, and a pointwise multiplication operation. An LSTM has three of these gates to protect and control the cell state, deciding what information is going to be thrown away, what new information is stored in the cell state and what will be the output. More precisely, the input, I_G, output, O_G, and forget, F_G, gates prevent memory contents from being perturbed by irrelevant inputs and outputs and thus allowing for long term memory storage.

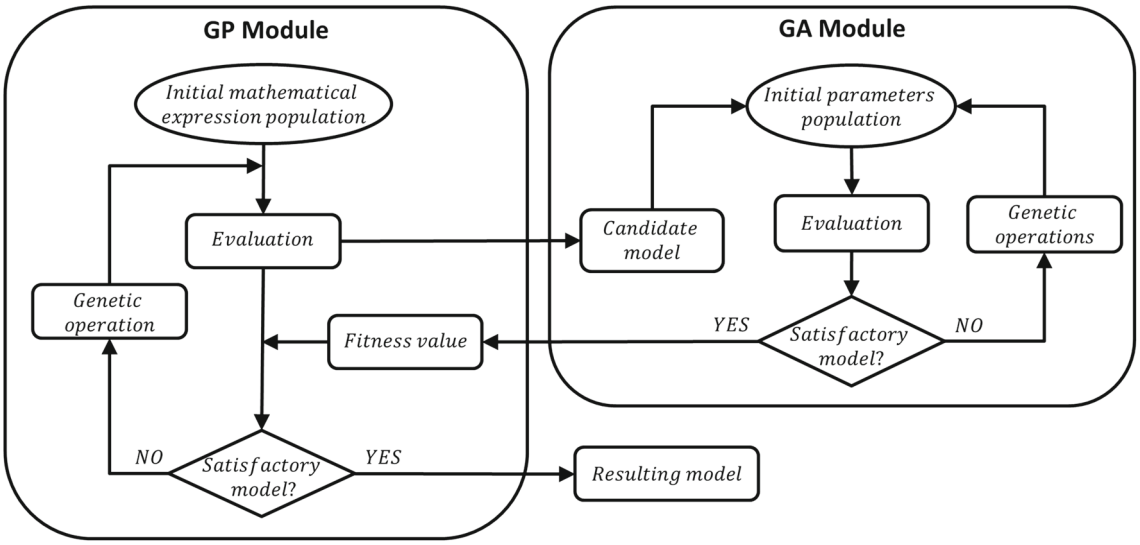

Fig. 2 Proposed evolutionary physical modeling flowchart composed of a GP Module, which starts generating an initial population of mathematical expressions, and a GA Module that evaluates them

The first step is to decide what information is going to be thrown away from the cell state. This decision is made by a sigmoid layer that looks at h_{t-1} and x_t, and outputs a number between 0 and 1 for each number in the cell state C_{t-1}, as it can be seen in Eq. 1.

$$F_t = \sigma\left(W_F \cdot [h_{t-1}, x_t] + b_F\right) \tag{1}$$

where F_t is the forget gate at time step t, W_F is the weight matrix that modifies the input at the same time step, x_t, and the hidden state at previous time step, h_{t-1}, and b_f is the bias term.

The next step is to decide what new information is going to be stored in the cell state. To do so, a sigmoid layer decides which values will be updated. In addition to that, a tanh layer creates a vector of new candidate values, $\tilde{C}_t$, that could be added to the state. Then, these two inputs will be combined to create an update to the state. The corresponding equations are shown as follows (see Eq. 2).

$$\begin{aligned} I_t &= \sigma\left(W_I \cdot [h_{t-1}, x_t] + b_I\right) \\ \tilde{C}_t &= \tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right) \end{aligned} \tag{2}$$

where I_t is the input gate at time step t, $\tilde{C}_t$ is the vector of new candidate values, W_I and W_C are the weight matrix that modifies the input at the same time step, x_t, and the hidden state at previous time step, h_{t-1}, and b_I and b_C are the bias terms.

The old cell state, C_{t-1}, is updated into the new cell state C_t according to the decisions made in previous steps, $C_t = F_t * C_{t-1} + I_t * \tilde{C}_t$. These are the new candidate values, scaled by how much each state value is updated.

Finally, the output is computed on the basis of the cell state, but filtered. A sigmoid layer is run to determine what parts of the cell state are going to be produced as output, and the output of the sigmoid gate is multiplied by the tanh of the cell state in order to produce values between -1 and 1 (see Eq. 3).

$$\begin{aligned} O_t &= \sigma\left(W_O \cdot [h_{t-1}, x_t] + b_O\right) \\ h_t &= O_t * \tanh(C_t) \end{aligned} \tag{3}$$

where O_t is the output gate at time step t, W_O is the weight matrix that modifies the input at the same time step, x_t, and the hidden state at previous time step, h_{t-1}, b_O is the bias term, C_t is the cell state and h_t is the computed output.

The LSTM units in a hidden layer are fully connected through recurrent connections. Layers are stacked so each unit in a lower LSTM hidden layer is fully connected to each unit in the LSTM hidden layer above it through feedforward connections. An schema of a typical LSTM unit is presented in Fig. 3a. In the schema, σ and tanh refer to sigmoid and tangent hyperbolic neural network layers, respectively, whereas X and Σ correspond to product and addition pointwise operations, respectively, over vectors.

In this study, a stacked LSTM network with two hidden layers and sigmoid activation function is used. Considering the time series that results from the evolutionary physical model predictions, $f(X) = \hat{\mathbf{y}} = (\hat{y}_1, \ldots, \hat{y}_n)$, data standardization is applied to $\hat{\mathbf{y}}$ as $\frac{\hat{\mathbf{y}} - \mu(\hat{\mathbf{y}})}{\sigma(\hat{\mathbf{y}})}$, being $\mu(\hat{\mathbf{y}})$ and $\sigma(\hat{\mathbf{y}})$ the mean and standard deviation of $\hat{\mathbf{y}}$, respectively. Then, data sequences are defined as m-dimensional vectors whose elements correspond to the input variables. The LSTM-based prediction model learns to predict the next l values for d of the

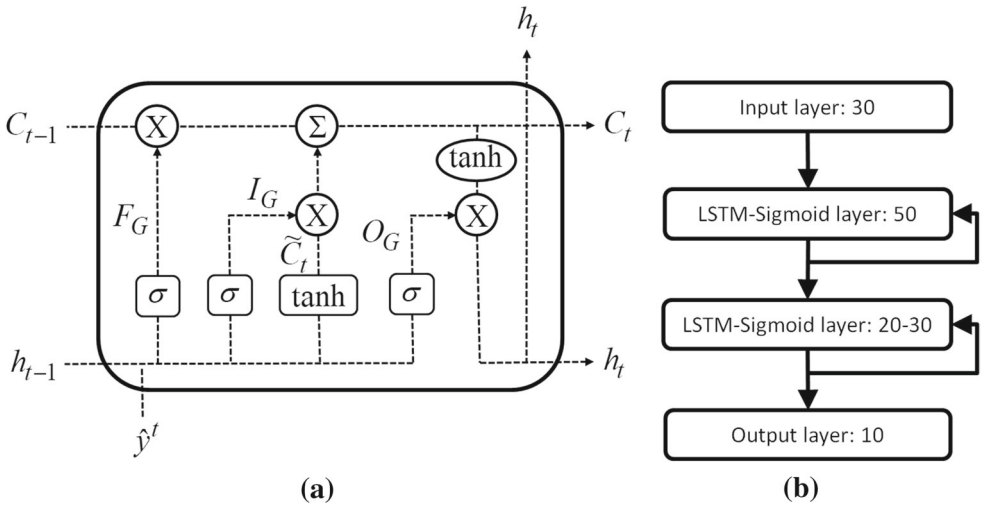

Fig. 3 Illustration of **a** a typical long short-term memory unit and **b** stacked LSTM-based network architecture used in this study, indicating the number of units (dimensionality of the output space) in each layer

input values s.t. $1 \leq d \leq m$. Given the asset under study and the monitoring time frequency (one value per minute), d is set to 30 time steps and l is set to 10 time steps, which means that using the data corresponding to the last 30 min of data the LSTM network will estimate the next 10 min of data. Although other complementary tests were performed with different l values (e.g., $l = 20$ and $l = 30$), results obtained were less accurate in terms of error rates and established time parameters were considered as good enough by the domain experts in order to anticipate behaviors of interest.

The network architecture that serves as the basis to model the performance of the reduction gear can be seen in Fig. 3b. Fifty process units (dimensionality of the output space) are considered in the first layer and 20–30 in the second layer, which indicates the number of units regarding the scenarios under consideration, S1 and S2, respectively. They will be further described in Sect. 3.1. Different candidate network architectures and configuration parameters were tested against validation data (the last 10% of the data set), and the resulting model error (MAE) was calculated in each trial. The configuration parameters and the architecture of the model with lowest error were considered as the optimal in each scenario (Çelik and Arcaklioğlu 2005). All of the networks used in this study were trained with an epoch (number of iterations) of 100, given the fact that from that iteration the loss function (MSE) and the network error (MAE) did not improve.

All details regarding the configuration parameters used in this study are presented in Table 2.

The training process finishes when the error of the model (MAE) for all training data in an epoch is less than 0.01 or when the maximum number of iterations is reached.

Having a prediction length of l, each of the selected d dimensions of $\hat{y}_t \in \hat{\boldsymbol{y}}$ for $l < t \leq n - l$ is predicted l times. The sequences of residuals are computed for every $\hat{y}_i$

Table 2 LSTM network configuration parameters

Parameter	Value
Type of network	Stacked LSTM
Number of layers	2
Units in the first hidden layer	50
Units in the second hidden layer	20–30
Activation function in hidden layer	Sigmoid
Activation function in output layer	Linear
Training method	Feedforward
Number of iterations	100
Time sequence size used as input values, d	30
Prediction length, l	10

as $\boldsymbol{r_i^t} = (r^t_{i,11}, \ldots, r^t_{i,1l}, \ldots, r^t_{i,d1}, \ldots, r^t_{i,dl})$, being $r^t_{i,kj}$ the difference between $\hat{y}_t$ and the predicted value at time $t-j$, h_t, and $k = 1, \ldots, d$. Then, the log likelihood of each sequence is calculated as it is shown in Eq. 4.

$$p_i = f(\overline{\boldsymbol{r}}_{\boldsymbol{i}} | \mu(\log \boldsymbol{r}), \sigma(\log \boldsymbol{r})) = \frac{1}{\overline{\boldsymbol{r}}_{\boldsymbol{i}} \sigma(\log \boldsymbol{r}) \sqrt{2\pi}} \exp^{\frac{-(\ln \overline{\boldsymbol{r}}_{\boldsymbol{i}} - \mu(\log \boldsymbol{r}))^2}{2\sigma(\log \boldsymbol{r})^2}} \tag{4}$$

where $\overline{\boldsymbol{r}}_{\boldsymbol{i}}$ is the average value of sequence of residuals $\boldsymbol{r_i}$ and parameters $\mu(\log \boldsymbol{r})$ and $\sigma(\log \boldsymbol{r})$ stand for the mean and standard deviation of $\log \boldsymbol{r}$, respectively, being $\boldsymbol{r} = (\overline{\boldsymbol{r}}_{\boldsymbol{1}}, \ldots, \overline{\boldsymbol{r}}_{\boldsymbol{n}})$ the list of average values of sequences of residuals over time.

Finally, a moving average filter of time window size $d = 30$ is applied to $\boldsymbol{p} = (p_1, \ldots, p_n)$ in order to avoid detecting transient behaviors that concern noise and false alarms. The smoothed score of p_i, denoted by score_{p_i}, is computed as follows:

$$\text{score}_{p_i} = \frac{\sum_{j=i-d}^{i+d} p_j}{d} \tag{5}$$

where p_j, with $j = i-d, \ldots, i+d$, is the log likelihood computed at time j and d is the window size.

Figure 4 shows the evolution of the log likelihood score over time, with and without applying the moving average smoothing (with $d = 30$). Without smoothing, it can be seen a decrease in the value that could be interpreted as an anomalous behavior. However, the immediate increase in the score indicates that it is not a real anomaly. Therefore, the smoothed score is an efficient method for restricting this phenomenon.

An anomaly threshold is calculated over the training data as the lowest score value, $\min(\text{score}_{\boldsymbol{p}})$, after smoothing the resulting set of log likelihoods for every sequence of residuals in the training data set, which corresponds to the predictions made by the evolutionary physical model. Then, when checking new data, $\hat{y}_{\text{new}}$, if the corresponding score value is below this threshold, the sequence will be considered as a potential anomaly to arise in l time steps. Similarly, a threshold to predict changes in the operating condition of the asset is also provided, by computing $\mu(\text{score}_{\boldsymbol{p}}) - 3\sigma(\text{score}_{\boldsymbol{p}})$.

The definition of these thresholds allows performing an intelligent online monitoring of the asset, anticipating possible faulty behaviors and changes in the operating condition of the asset by considering the predictions made by the LSTM network.

2.3 Overview of the algorithm

The overall algorithm combines the evolutionary physical modeling of condition monitoring data with the prediction of time sequences given by the LSTM network. The overall algorithm can be seen in Algorithm 1.

Algorithm 1 Deep evolutionary modeling for anomaly detection

Inputs: a target feature $\boldsymbol{y} = (y_1, \ldots, y_n)$, a set of operands, {sine, tangent, cosine, exp, log, abs, pow, sqrt, $*$, /, $+$, $-$}, a set of m input features, $X = \{X_1, \ldots, X_m\}$, the time sequence size used as input values d and the prediction lenght l

1: Compute $f(X) = \hat{\boldsymbol{y}} \approx \boldsymbol{y}$ using the evolutionary physical modeling presented in 2.1
2: Fit standardize $\hat{\boldsymbol{y}}$ using the LSTM network presented in 2.2
3: Compute $\boldsymbol{p}$ using Eq. 4
4: Compute $\text{score}_{\boldsymbol{p}}$ using Eq. 5
5: anomaly threshold $= \min(\text{score}_{\boldsymbol{p}})$
6: operational threshold $= \mu(\text{score}_{\boldsymbol{p}}) - 3\sigma(\text{score}_{\boldsymbol{p}})$
7: **for all** $\hat{y}_{\text{new}}$ **do**
8: Compute $\text{score}_{p\text{new}}$ using Eq. 5
9: **if** $\text{score}_{p\text{new}} <$ anomaly threshold **then**
10: $\hat{y}_{\text{new}} \rightarrow$ anomaly
11: **end if**
12: **if** $\text{score}_{p\text{new}} <$ operational threshold **then**
13: $\hat{y}_{\text{new}} \rightarrow$ operating condition change
14: **end if**
15: **end for**

Besides the interpretability of the physical model obtained and the prediction accuracy given by the deep neural network, one main advantage of this approach is that the target variable to be modeled and predicted, normally corresponding to the control parameter of the asset, can be fully approximated by other operational parameters. Once the modeling phase is completed, in a test bench under laboratory conditions, for instance, the deep evolutionary model can be deployed for online condition monitoring with no need of measuring the asset control parameter. On the other hand, one main drawback is that the evolutionary modeling generates a valid formula, but not necessarily the one that best fits the data. And in relation to the LSTM-based anomaly prediction, and despite of its accuracy for sequence prediction, the model training stage becomes slower than other similar approaches due to the complexity of the activation function.

3 Test scenario

3.1 Description of the experimental setup

The main goal of the present study is to provide a deep evolutionary-based approach to efficiently model and predict the physical behavior of complex assets and processes. In order to evaluate the proposed approach, several experiments were conducted on a real scenario, processing operational data acquired from a CODOG (combined diesel or gas) marine propulsion system. It is designed for using diesel engines for cruising at economical speed or using gas turbines for dashing at high speed.

The heart of a combined propulsion system is a reduction gear, which is an arrangement that allows to lower an input speed for a requirement of slower output speed, with same or more output torque. Its main function is to turn propellers within an efficient speed range, optimizing load conditions and fuel consumption.

The reduction gear assembly is usually known as reduction gear box, and it consists of a set of rotating gears connected to a wheel work. The high-speed incoming motion from the wheel work is transmitted to the set of rotating gears, wherein the motion or torque is changed. Depending on the output speed requirement of the application, the number of gears used in the reduction gear may vary.

A drive system in a multi-engine marine vessel having port and starboard propellers combines gear systems for drive by electric motors, gas turbines and/or diesel engines. Gears between starboard and port side propellers are connected for high output. An illustration of 1st and 2nd reduction gears, located in starboard and port side, respectively, can be seen in Fig. 5. TP stands for the power turbine and GG corresponds to the gas generator.

The proposed deep evolutionary modeling approach has been tested on 1 year time operational data, from January to December 2016, of a port side marine vessel reduction gear.

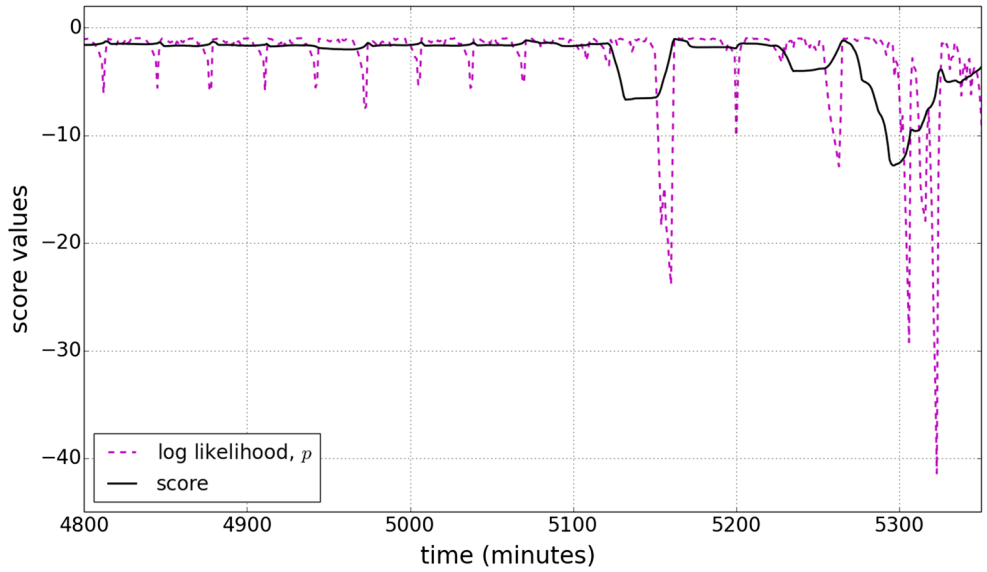

Fig. 4 Example of the evolution of the log likelihood score over time, with and without applying the moving average smoothing with $d = 30$

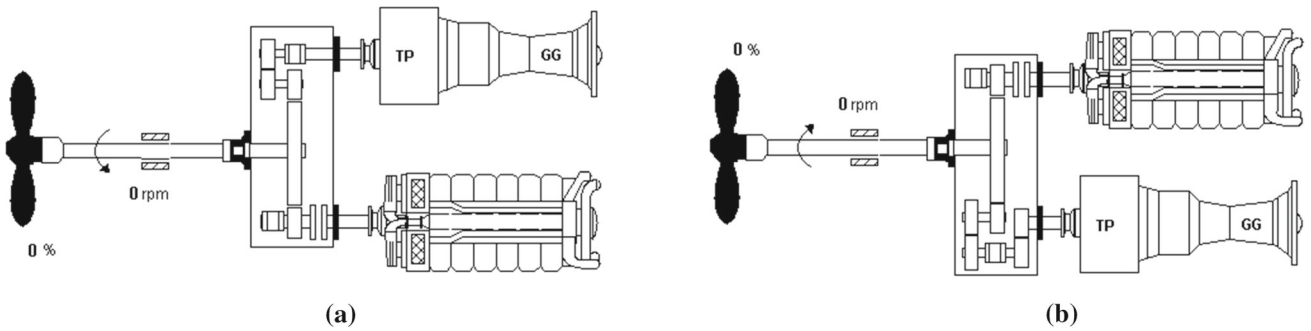

Fig. 5 Illustration of **a** 1st reduction gear (starboard) and **b** 2nd reduction gear (port side) of a CODOG (combined diesel or gas) marine propulsion system

Table 3 Reduction gear scenarios considered in this study

Scenario	Code	Diesel engine speed	Gas turbine speed	Data set size
Diesel engine in stable operation	S1	350–1500 rpm	–	109,284 events
Gas turbine in stable operation	S2	–	4000–8900 rpm	28,688 events

A total of 142,702 events are analyzed. An event is collected approximately every minute, and it is composed by values of all monitored parameters at that specific time instant. Events in time regions where the engine is not in stable operating conditions are not considered, since they have no significance regarding physical models. Seasonality is controlled by using environmental features in the mathematical expressions when needed.

On the basis of diesel engine speed and gas turbine speed, analyzed events are distributed as it is presented in Table 3.

Target variables were identified as characteristic parameters of reduction gear physical behavior regarding considered scenarios: diesel engine in stable operation (S1) and gas turbine in stable operation (S2). They are the engine pinion bearing and the gas turbine thrust bearing temperatures, and they present different range of values and variability depending on reduction gear operating condition. A sensitivity analysis based on correlation coefficient was used to determine what set of operational parameters was used as input for each target variable, to select the most sensitive for use in the modeling process. The corresponding parameters used in this study are listed in Table 4.

The software platforms used are Java Platform Enterprise Edition from Oracle, based on the Java programming language, and Spyder 2.3.0, based on Python. Experiments were conducted in the Java-based Eclipse RCP (Rich Client

Table 4 Reduction gear parameters used in model construction

Type	Parameter	Code
Output	Engine pinion bearing temp	y_{S1}
	Gas turbine thrust bearing temp	y_{S2}
Input	First reduction pinion axial bearing temp	$x_{1,S1}$
	First reduction bull gear temp	$x_{2,S1}$
	Main back thrust bearing temp	$x_{1,S2}$
	Engine pinion bearing temp	$x_{2,S2}$
	First reduction pinion bearing temp	$x_{3,S2}$

Platform, Kepler version) environment and in Python 3.4.3 (keras package was used, available at https://keras.io/), on Ubuntu-Linux 14.04 (64bits) on a CORE i5 desktop PC with 6 GB of RAM memory. Using mentioned software and hardware processing specifications, computational time for conducted experiments is presented in Table 5. This includes the required time of the proposed deep evolutionary approach regarding each learning phase of the modeling process in relation to scenarios S1 and S2 and the size of the resulting models, once they are serialized for further use. Having constructed the deep evolutionary models, a random event was investigated to calculate the required computational time for online anomaly prediction. The result of this exercise is tabulated in the last row of Table 5. According to the obtained results, it can be seen that the required computational time for online warning is considerably small which demonstrates the fact that combination of off-line learning and online testing using the proposed approach can be applied for real-time anomaly prediction.

In the following subsection, the analysis process followed by proposed methodology is shown and discussed over described scenario.

3.2 Results and discussion

Data taken from the experimental setup were used as training (60%), testing (30%) and validation (10%) sets for the modeling step to achieve generalization. Training set was used for learning the evolutionary physical model, whereas testing set was used for testing it. Predictions made by the evolutionary physical model were used for training the LSTM network, which was tested over the validation set. The resulting score is provided, and the capability of the approach to anticipate potential anomalies is demonstrated.

The overall performance is quantified by computing the mean absolute error (MAE), mean squared error (MSE) and coefficient of determination (R^2) given by:

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n} \mathrm{abs}(y_i - \hat{y}_i) \tag{6}$$

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{7}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \overline{y})^2} \tag{8}$$

where $\overline{y}$ is the average value of the observed data, $\boldsymbol{y} = (y_1, \ldots, y_n)$, and $\hat{\boldsymbol{y}} = (\hat{y}_1, \ldots, \hat{y}_n)$ are the predicted values for each $i = 1, \ldots, n$ by the model. In the case of the LSTM network predictions, the produced sequences of size l are compared to the estimations made by the evolutionary model learned and the validation data during training and testing phases, respectively.

Test data, validation data and the estimations made by generated models regarding each step of the learning process are graphically shown to support the above mentioned performance metrics. They can be easily interpretable for maintainers and system experts, offering a very simple tool to visually validate the proposed approach at each stage of the learning process.

In the following subsections, the results obtained by the proposed approach when modeling control parameters and anticipating behaviors of interest in scenarios S1 and S2 are discussed. In order to evaluate the performance achieved in the modeling stage in relation to other similar and widely used algorithms, results are compared to those obtained by an optimized version of the classification and regression trees, CART, algorithm (Rokach and Maimon 2005) and MLP networks for sequence prediction (Zhao et al. 2017). Tests were conducted setting the maximum trees depth to 20 and using the same LSTM network configuration parameters as the basis for the MLP architecture regarding each scenario, since adding further layers into the MLP does not improve the results but slows down training. It must be noted that generated models are valid for the ranges of data sets given in Table 3.

A tenfold cross-validation, Shapiro–Wilk and Levene tests are also executed on test and validation datasets, which are the training and test stages in relation to the MLP and LSTM networks, aiming to check whether there are significant differences between the proposed approach and the DT-MLP methods (Rodriguez et al. 2010; Garson 2012). The focus is on the most critical part of the process: the prediction stage.

3.2.1 Scenario S1: diesel engine in stable operation

The details of the experimental parameters including range, mean (μ) and standard deviation (σ) used in this study with diesel engine in stable operation are presented in Table 6.

Formula obtained regarding the engine pinion bearing temperature in scenario S1 can be seen in Eq. 9.

$$y_{S1} = \cos\left(|x_{2,S1}|\right) + |x_{2,S1}| - x_{1,S1} - 43.0 \tag{9}$$

Table 5 Computational time of the proposed deep evolutionary approach regarding each learning phase of the modeling process and size of the resulting models in scenarios S1 and S2

	Scenario			
	S1		S2	
	Time (s)	Size	Time (s)	Size
Evolutionary physical modeling	181.51	211 bytes	81.59	336 bytes
LSTM network	6592.714	152.2 kB	2072.538	187.0 kB
Total	6774.224	152.411 kB	2154.128	187.336 kB
Instant prediction	0.004		0.003	

Table 6 Range of values of reduction gear parameters used in this study regarding scenario S1

Target variable	Range (°C)	$\mu \pm \sigma$	Input variable	Range (°C)	$\mu \pm \sigma$
y_{S1}	17–63	54.61 ± 4.17	$x_{1,S1}$	22–59	53.91 ± 2.27
			$x_{2,S1}$	17–46	43.13 ± 1.11

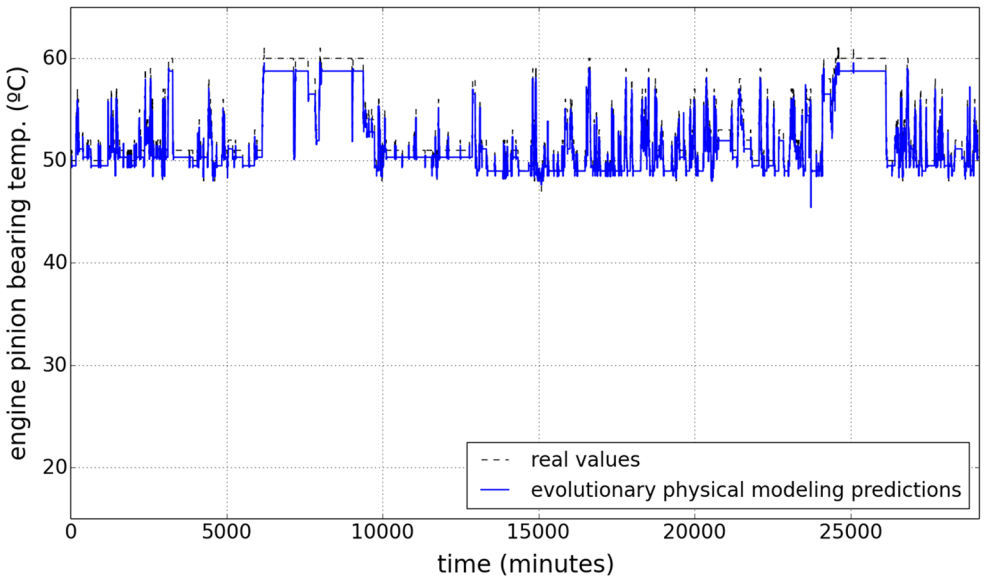

Fig. 6 Evolutionary physical modeling results on engine pinion bearing temperature test data, real and predicted values over time

where y_{S1} is the engine pinion bearing temperature, $x_{1,S1}$ is the 1st reduction pinion axial bearing temperature and $x_{2,S1}$ is the 1st reduction bull gear temperature.

The prediction undertaken by the evolutionary model learned, together with sensor readings for the same input test data, is shown in Fig. 6.

Then, the LSTM network is trained over evolutionary physical modeling predictions. The resulting LSTM network estimations and the real engine pinion bearing temperature values on validation data set can be seen in Fig. 7.

The statistical performance of the method regarding each learning phase of the modeling process in scenario S1 and in relation to training, testing and validation data sets is given in Table 7.

The training results proved that the proposed evolutionary bearing model in scenario S1 obtained good accuracy ($R^2_{\text{Evol}-\text{train}} = 0.917$), and low error rates when fitting the training data ($\text{MAE}_{\text{Evol}-\text{train}} = 0.935$ and $\text{MSE}_{\text{Evol}-\text{train}} = 1.385$). A high generalization can be also observed when comparing the evolutionary model predictions with the experimental data used for the test stage ($R^2_{\text{Evol}-\text{test}} = 0.938$), and once again rather low error rates ($\text{MAE}_{\text{Evol}-\text{test}} = 0.804$ and $\text{MSE}_{\text{Evol}-\text{test}} = 0.898$). For decision trees regression-based modeling, results are, in general, slightly better than those given by the evolutionary model on the training set (MAE = 0.383, MSE = 0.249 and $R^2 = 0.981$), but less accurate on the test set (MAE = 0.669, MSE = 1.185 and $R^2 = 0.877$). The evolutionary model generalizes better on unseen data, and therefore, it is more meaningful and valid to capture the physics of the asset and its dynamics.

LSTM network also performed quite satisfactorily, giving $R^2_{\text{LSTM}-\text{train}} = 0.931$ and $R^2_{\text{LSTM}-\text{test}} = 0.917$, and keeping low error rates when approximating the predictions made by the evolutionary bearing model and predicting the validation

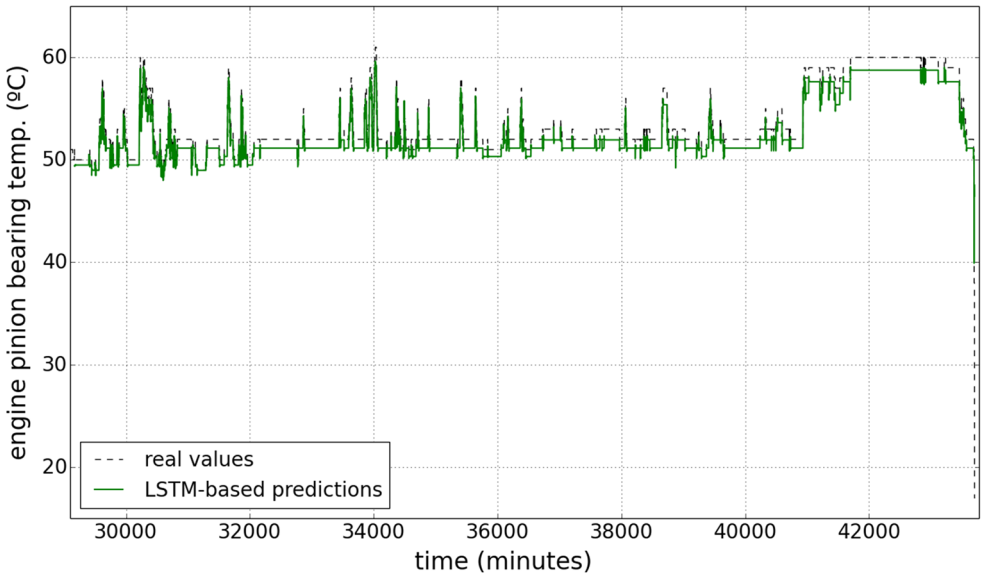

Fig. 7 LSTM results on engine pinion bearing temperature validation data, real and predicted values over time

Table 7 Results of the deep evolutionary modeling for engine pinion bearing temperature training, testing and validation sets with diesel engine in stable operation

Parameter	Data used	Learning stage	MAE	MSE	R^2
$\boldsymbol{y_{S1}}$	Training set	Evolutionary physical modeling (training)	0.935	1.385	0.917
	Testing set	Evolutionary physical modeling (testing)	0.804	0.898	0.938
	$f(X_{S1}) = \hat{\boldsymbol{y}_{S1}} \approx \boldsymbol{y_{S1}}$	LSTM network (training)	0.723	0.908	0.931
	Validation set	LSTM network (testing)	0.573	0.759	0.917

data values ($\text{MAE}_{\text{LSTM-train}} = 0.723$, $\text{MSE}_{\text{LSTM-train}} = 0.908$ and $\text{MAE}_{\text{LSTM-test}} = 0.573$, $\text{MSE}_{\text{LSTM-test}} = 0.759$, respectively). Error rates decreased over time, which could be due to the fact that data set used for testing and validation presented a more stable behavior regarding the propulsion system operating conditions. In comparison with the MLP network, it turned out that this latter performed slightly better than the former, obtaining MAE = 0.393, MSE = 0.578, R^2 = 0.956 and MAE = 0.319, MSE = 0.526, R^2 = 0.942 on training and test sets, respectively. This yields to believe that under certain circumstances, a well-trained MLP network is able to very accurately forecast time series data. Results obtained when performing the tenfold cross-validation procedure demonstrate a more robust generalization in the case of the LSTM network, as it can be seen in Table 8, which provides the mean value of every fold for each performance metric results regarding the MLP and LSTM networks.

A Shapiro–Wilk statistical test is performed for each performance metric regarding the predictions made by both methods. Obtained results are shown in Table 9. Only in a few cases the p value is less than 0.05 (e.g., MLP–MAE and MLP–MSE metrics computed over the training set), then the null hypothesis is not rejected and there is evidence that the data are drawn from a normally distributed population in most of the cases.

A Levene test is also applied to check that each pair of sets of results are from populations with equal or similar variances. Resulting p values can be seen in Table 10. Given the Levene test results, it can be concluded that the variances in the sets of results for every pair of performance metrics are homogeneous.

The anomaly threshold computed over the smoothed log likelihoods of the residuals, $\boldsymbol{r_{S1}}$, and the resulting score over the validation data are presented in Fig. 8. The real sensor readings are also shown in the same figure, above.

The sudden drop in engine pinion bearing temperature at the end of the time series is successfully anticipated by the resulting score, which is significantly lower than the anomaly threshold at time $a_{t,S1} - l$. In this case, the problem could have been caused by a faulty sensor. Additionally, several operational changes are predicted at times $b_{t,S2} - l$, $t = (1, \ldots, 9)$.

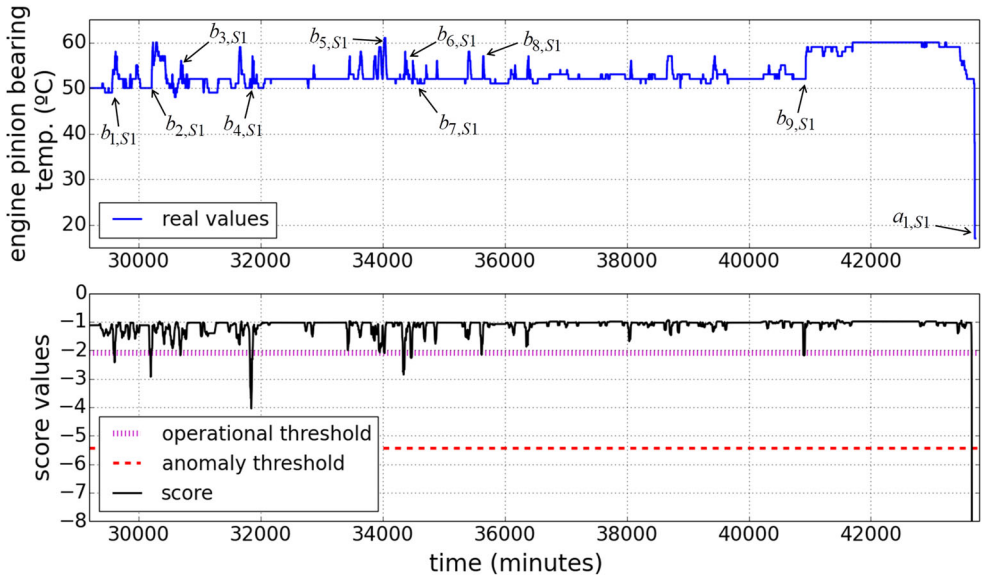

Fig. 8 The real sensor readings (above) and the resulting operational and anomaly thresholds and score values on engine pinion bearing temperature (below) over the validation set

3.2.2 Scenario S2: gas turbine in stable operation

The details of the experimental parameters including range, mean (μ) and standard deviation (σ) used in this study with gas turbine in stable operation can be seen in Table 11.

Formula obtained regarding the gas turbine thrust bearing temperature in scenario S2 is presented in Eq. 10.

$$y_{S2} = \log x_{1,S2} + 2x_{2,S2} - \frac{x_{3,S2}}{\log(2x_{1,S2})} \tag{10}$$

where y_{S2} is the gas turbine thrust bearing temperature, $x_{1,S2}$ is the main back thrust bearing temperature, $x_{2,S2}$ is the engine pinion bearing temperature and $x_{3,S2}$ is the 1st reduction pinion bearing temperature.

The values predicted by the formula given by the evolutionary physical modeling process are presented in Fig. 9. The real gas turbine thrust bearing temperature values are also provided.

The LSTM network predictions made in relation to scenario S2, in addition to the real engine pinion bearing temperature values on validation data set, are shown in Fig. 10. As in the previous case, the resulting LSTM network was trained using the values obtained by the evolutionary physical model, $f(X_{S2}) = \hat{y}_{S2} \approx y_{S2}$.

The results achieved by the proposed approach in relation to each learning phase of the modeling process in scenario S2, in terms of error rates and R^2, are shown in Table 12.

As it was found in relation to the bearing model in scenario S1 when the diesel engine was working in stable operation, the training results regarding scenario S2 also proved that the proposed deep evolutionary bearing model obtained good accuracy ($R^2_{\text{Evol}-\text{train}} = 0.949$ and $R^2_{\text{LSTM}-\text{train}} = 0.843$), and low error values ($\text{MAE}_{\text{Evol}-\text{train}} = 0.97$, $\text{MSE}_{\text{Evol}-\text{train}} = 1.717$ and $\text{MAE}_{\text{LSTM}-\text{train}} = 1.44$,

Table 8 Tenfold cross-validation results for each performance metric regarding MLP and LSTM networks with diesel engine in stable operation

	MAE		MSE		R^2	
	Train	Test	Train	Test	Train	Test
MLP	0.457	0.437	0.604	0.588	0.948	0.882
LSTM	0.420	0.456	0.563	0.617	0.951	0.882

Table 9 Shapiro–Wilk normality test p values computed on a tenfold cross-validation basis for the MLP and LSTM networks with diesel engine in stable operation

	MAE		MSE		R^2	
	Train	Test	Train	Test	Train	Test
MLP	0.037	0.387	0.001	0.825	0.102	0.228
LSTM	0.486	0.146	0.272	0.467	0.294	0.354

Table 10 Levene test p values computed on a tenfold cross- validation basis to check the null hypothesis that all input samples for each performance metric values regarding MLP and LSTM networks with diesel engine in stable operation are from populations with equal or very similar variance

	MAE		MSE		R^2	
	Train	Test	Train	Test	Train	Test
MLP–LSTM	0.568	0.557	0.895	0.842	0.866	0.696

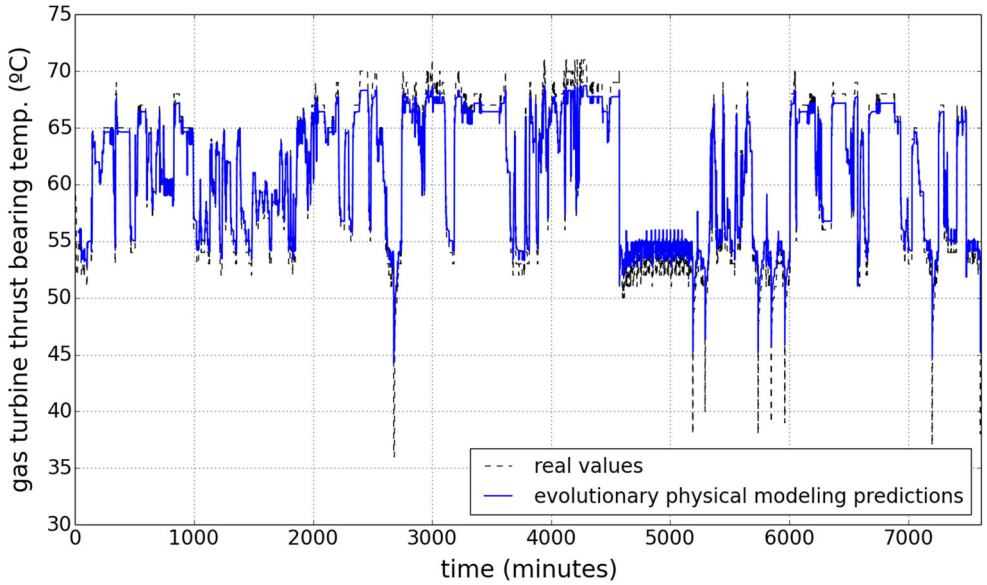

Fig. 9 Evolutionary physical modeling results on gas turbine thrust bearing temperature test data, real and predicted values over time

Table 11 Range of values of reduction gear parameters used in this study regarding scenario S2

Target variable	Range (°C)	$\mu \pm \sigma$	Input variable	Range (°C)	$\mu \pm \sigma$
y_{S2}	33–80	60.42 ± 6.56	$x_{1,S2}$	30–48	43.53 ± 1.75
			$x_{2,S2}$	34–70	57.78 ± 5.60
			$x_{3,S2}$	36–74	56.34 ± 5.12

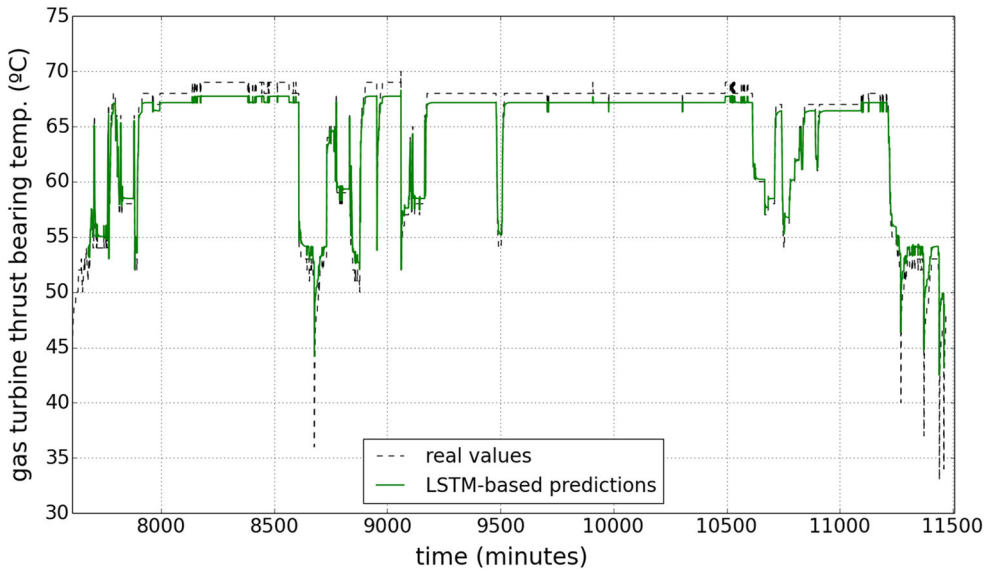

Fig. 10 LSTM results on gas turbine thrust bearing temperature validation data, real and predicted values over time

$\text{MSE}_{\text{LSTM-train}} = 6.690$) when fitting the training data. A high generalization can be also observed when comparing the evolutionary model predictions with the experimental data used for the test stage ($R^2_{\text{Evol-test}} = 0.959$), and once again rather low error values ($\text{MAE}_{\text{Evol-test}} = 1.01$ and $\text{MSE}_{\text{Evol-test}} = 1.731$). LSTM network was also quite accurate when predicting the sequences of data on validation set, giving $R^2_{\text{LSTM-test}} = 0.88$, $\text{MAE}_{\text{LSTM-test}} = 1.075$ and $\text{MSE}_{\text{LSTM-test}} = 4.354$. However, a slight decrease on performance can be appreciated when fitting the deep network to the evolutionary physical modeling values, also producing worse results than those obtained in scenario S1. This can

Table 12 Results of the deep evolutionary modeling for gas turbine thrust bearing temperature training, testing and validation sets with gas turbine in stable operation

Parameter	Data used	Learning stage	MAE	MSE	R^2
y_{S2}	Training set	Evolutionary physical modeling (training)	0.97	1.717	0.949
	Testing set	Evolutionary physical modeling (testing)	1.01	1.731	0.959
	$f(X_{S2}) = \hat{y}_{S2} \approx y_{S2}$	LSTM network (training)	1.44	6.690	0.843
	Validation set	LSTM network (testing)	1.075	4.354	0.88

Table 13 Tenfold cross-validation results for each performance metric regarding the MLP and LSTM networks with gas turbine in stable operation

	MAE		MSE		R^2	
	Train	Test	Train	Test	Train	Test
MLP	1.374	1.391	6.246	6.321	0.857	0.774
LSTM	1.458	1.526	5.998	6.745	0.863	0.765

Table 14 Shapiro–Wilk normality test p values computed on a tenfold cross-validation basis for the MLP and LSTM networks with gas turbine in stable operation

	MAE		MSE		R^2	
	Train	Test	Train	Test	Train	Test
MLP	0.083	0.002	0.519	0.712	0.518	0.855
LSTM	0.240	0.504	0.913	0.402	0.635	0.619

be due to the more variable nature of the control parameter under study, the gas turbine thrust bearing temperature.

Decision trees-based modeling obtained good training results, largely similar to those obtained by the evolutionary model, namely MAE = 0.561, MSE = 0.611 and R^2 = 0.986. It also generalized quite satisfactorily on the test set, giving MAE = 0.792, MSE = 0.992 and R^2 = 0.974. However, the resulting model is not as understandable and physically meaningful as the corresponding evolutionary formula. On the other hand, regarding the MLP network-based modeling on the test data, the overall performance was slightly less accurate than that achieved by the LSTM network, obtaining MAE = 1.817, MSE = 8.026 and R^2 = 0.812 when fitting the data generated by the evolutionary model and MAE = 1.224, MSE = 4.996 and R^2 = 0.863 on the validation set.

When comparing results given by the tenfold cross-validation procedure, as it can be seen in Table 13, MLP and LSTM networks obtained very similar performance and generalization capacity. The R^2 metric produced less accurate results in both methods.

Following the same methodology as in the case of the scenario S1, the Shapiro–Wilk statistical test is performed for each performance metric with gas turbine in stable operation and regarding the predictions made by MLP and LSTM networks. Obtained results are shown in Table 14. Only in the case of the MLP–MAE metric computed over the test set the corresponding p value is less than 0.05, then the null hypothesis is not rejected and there is evidence that the data are drawn from a normally distributed population in most of the cases.

Given the p values obtained by the Levene test (see Table 15), it can be concluded that the variances in the sets of results for every pair of performance metrics regarding the MLP and LSTM networks are homogeneous, similarly as it was found in relation to scenario S1 only in the case of the R^2 metric computed over the training set the corresponding p value was slightly less than 0.05.

Table 15 Levene test p values computed on a tenfold cross- validation basis to check the null hypothesis that all input samples for each performance metric values regarding MLP and LSTM networks with gas turbine in stable operation are from populations with equal or very similar variance

	MAE		MSE		R^2	
	Train	Test	Train	Test	Train	Test
MLP–LSTM	0.404	0.546	0.173	0.410	0.043	0.958

The resulting score over the validation set, along with the anomaly threshold computed as the min of the smoothed likelihood of the differences between the predictions made by the evolutionary physical model learned and the LSTM network estimations, is shown in Fig. 11. The real gas turbine thrust bearing temperature values over the validation set are also provided, above.

No anomalous sequences are detected. However, heavy changes in the computed score over time can be appreciated. They mainly correspond to changes in the operating behavior of the propulsion system, as it can be appreciated in the sensor readings. They can be also anticipated by the proposed approach, namely at times $b_{t,S2} - l, t = (1, \ldots, 7)$, by means of the operational threshold calculated to predict changes in the operating conditions of the asset. The corresponding warnings can be used for obtaining higher efficiency, optimizing fuel consumption and reducing emissions.

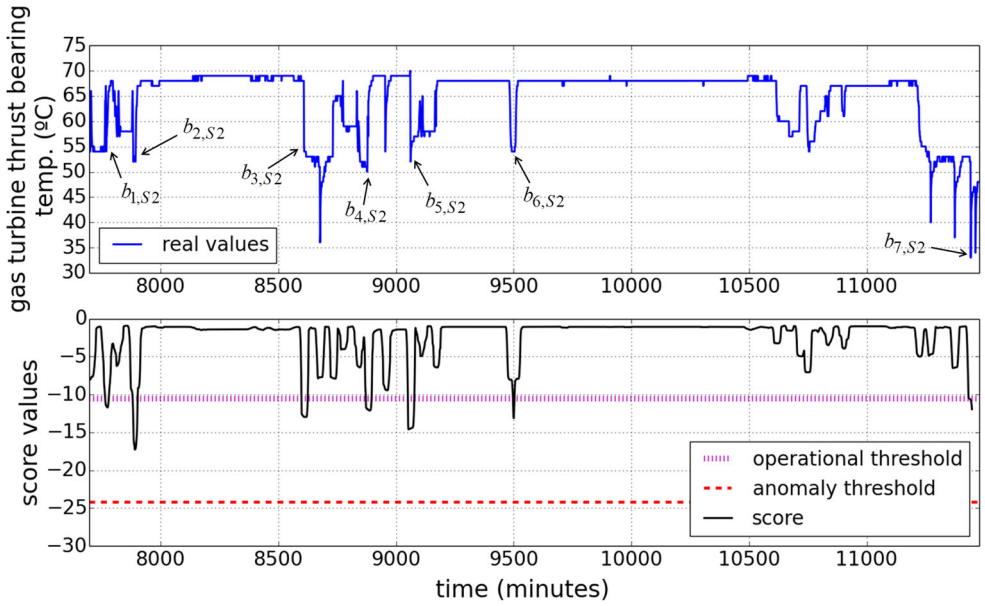

Fig. 11 The real sensor readings (above) and the resulting operational and anomaly thresholds and score values on gas turbine thrust bearing temperature (below) over the validation set

4 Conclusions

This work presents a comprehensive yet accurate data-driven deep evolutionary approach to efficiently estimate and predict underlying physical behaviors from condition monitoring data. The objective of the study is to provide a valid solution focused on interpretability, in contrast to 'black-box' models, but also on the prediction accuracy of the resulting model. Experimental results show the potential of the proposed approach to successfully model and anticipate behaviors from a reduction gear of a CODOG (combined diesel or gas) marine propulsion system, providing a finer and easier to interpret technique for experts to study and online monitoring its performance. Besides the interpretability of the physical model obtained and the prediction accuracy given by the deep neural network, one main advantage of this approach is that the control parameter to be modeled and predicted can be fully approximated by other operational parameters, which can be much easier to monitor in a real industrial environment. Nevertheless, it must be noted that the formula generated by the evolutionary modeling is not necessarily the one that best fits the data under study. The computational time needed for training the LSTM-based anomaly prediction model is also an issue that should be considered.

Training, testing and validation results proved that the models obtained by the proposed approach were able to successfully learn the nonlinear relationship between the input and target variables of the reduction gear in both scenarios, with high coefficient of determination, being $R^2_{\text{Evol}-\text{train}} = 0.917$, $R^2_{\text{Evol}-\text{test}} = 0.938$ in evolutionary physical modeling stage, and $R^2_{\text{LSTM}-\text{train}} = 0.931$, $R^2_{\text{LSTM}-\text{test}} = 0.917$ in LSTM network fitting stage, respectively, in scenario S1, when the diesel engine was in stable operation. In this same scenario, it was found that the sudden drop in engine pinion bearing temperature at the end of the time series was successfully anticipated at time $a_{1,S1} - l$, together with several operational changes at times $b_{t,S1} - l$, $t = (1, \ldots, 9)$. Regarding scenario S2, when the gas turbine was in stable operation, the proposed deep evolutionary bearing model also obtained good accuracy rates ($R^2_{\text{Evol}-\text{train}} = 0.949$, $R^2_{\text{Evol}-\text{test}} = 0.959$ and $R^2_{\text{LSTM}-\text{train}} = 0.843$, $R^2_{\text{LSTM}-\text{test}} = 0.88$). In this case, the LSTM network performed slightly worse than in scenario S1 probably due to the more variable nature of the control parameter under study. Similarly as in scenario S1, the most important changes in the operating condition of the asset were predicted, namely at times $b_{t,S2} - l$, $t = (1, \ldots, 7)$.

The performance achieved by the deep evolutionary approach at each step of the modeling stage was compared to that obtained by decision trees for regression and MLP networks for sequence prediction. Overall, results were largely similar in all cases. However, although decision trees regression performed quite satisfactorily, evolutionary models are more meaningful regarding the physics of the asset under study. In addition to that, it can be also concluded that both the LSTM and well-trained MLP networks are able to very accurately forecast time series data.

All these findings show that deep evolutionary models learned are fast and practical, yet accurate representations of

physical behaviors in data. From such mathematical formulation, different asset behaviors can be modeled and anticipated in l time steps for a wide variety of purposes: health status estimation, work orders scheduling, energy saving, working conditions optimization, etc. The deep evolutionary models can be easily deployed for online asset condition monitoring and predictions made can be further compared to real asset conditions. This will allow detecting deviations from optimal operation and thus obtaining higher efficiency, optimizing fuel consumption and reducing emissions.

Compliance with ethical standards

Conflict of interest All authors declare that they have no conflict of interest.

Ethical standard This article does not contain any studies with human participants or animals performed by any of the authors.

References

Al-Janabi S (2017) Pragmatic miner to risk analysis for intrusion detection (pmraid). In: International conference on soft computing in data science. pp 263–277

Al_Janabi S, Al_Shourbaji I, Salman MA (2017) Assessing the suitability of soft computing approaches for forest fires prediction. Appl Comput Inform 14:214–224

Azar AT, Vaidyanathan S (2015) Computational intelligence applications in modeling and control. Springer, Berlin

Barmpalexis P, Kachrimanis K, Tsakonas A, Georgarakis E (2011) Symbolic regression via genetic programming in the optimization of a controlled release pharmaceutical formulation. Chemometr Intell Lab Syst 107(1):75–82

Bentley P (1999) Evolutionary design by computers. Morgan Kaufmann, Burlington

Box GE, Jenkins GM, Reinsel GC, Ljung GM (2015) Time series analysis: forecasting and control. Wiley, Hoboken

Brabazon A, O'Neill M, McGarraghy S (2015) Natural computing algorithms. Springer, Berlin

Carrascal A, Díez Oliván A, Font Fernández JM, Manrique Gamo D (2009) Evolutionary generation of fuzzy knowledge bases for diagnosing monitored railway systems. In: 22nd International congress on condition monitoring and diagnostic engineering management (COMADEM 2009), pp 191–198, San Sebastián, España

Carrascal A, Font J, Pelta D (2010) Evolutionary physical model design. In: Research and development in intelligent systems, vol xxvi. Springer, pp 487–492

Çelik V, Arcaklioğlu E (2005) Performance maps of a diesel engine. Appl Energy 81(3):247–259

Chapra SC, Canale RP (2012) Numerical methods for engineers, vol 2. McGraw-Hill, New York

Dallemand JE (1958) Stepwise regression program on the ibm 704. In: GMR-I99, Research Laboratories, GeneralMotor Corporation (November 1958)

Diez-Olivan A, Pagan JA, Khoa NLD, Sanz R, Sierra B (2018) Kernelbased support vector machines for automated health status assessment in monitoring sensor data. Int J Adv Manuf Technol 95(1–4):327–340

Diez-Olivan A, Pagan JA, Sanz R, Sierra B (2017) Data-driven prognostics using a combination of constrained k-means clustering, fuzzymodeling and lof-based score. Neurocomputing 241:97–107

Doerr B, Fischer P, Hilbert A, Witt C (2017) Detecting structural breaks in time series via genetic algorithms. Soft Comput 21(16):4707–4720

Espinosa HEP, Ayala-Solares JR (2016) The power of natural inspiration in control systems. In: Nature-inspired computing for control systems. Springer, pp 1–10

Garg A, Vijayaraghavan V, Tai K, Singru PM, Jain V, Krishnakumar N (2015) Model development based on evolutionary framework for condition monitoring of a lathe machine. Measurement 73:95–110

Garson GD (2012) Testing statistical assumptions. Statistical Associates Publishing, Asheboro

Gong YJ, Chen WN, Zhan ZH, Zhang J, Li Y, Zhang Q, Li JJ (2015) Distributed evolutionary algorithms and theirmodels: a survey of the state-of-the-art. Appl Soft Comput 34:286–300

Guo L, Li N, Jia F, Lei Y, Lin J (2017) A recurrent neural network based health indicator for remaining useful life prediction of bearings. Neurocomputing 240:98–109

Haeri MA, Ebadzadeh MM, Folino G (2017) Statistical genetic programming for symbolic regression. Appl Soft Comput 60:447–469

Harrell FE (2001) Regression modeling strategies: with applications to linear models, logistic regression, and survival analysis. Springer, Berlin

Hayton P, Utete S, King D, King S, Anuzis P, Tarassenko L (2007) Static and dynamic novelty detection methods for jet engine health monitoring. Philos Trans R Soc Lond A Math Phys Eng Sci 365(1851):493–514

Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780

Koza JR (1992) Genetic programming: on the programming of computers by means of natural selection, vol 1. MIT Press, Cambridge

La Cava W, Danai K, Spector L, Fleming P, Wright A, Lackner M (2016) Automatic identification of wind turbine models using evolutionary multiobjective optimization. Renew Energy 87:892–902

Langley P, Zytkow JM (1989) Data-driven approaches to empirical discovery. Artif Intell 40(1):283–312

LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436–444

Malhotra P , Vig L, Shroff G, Agarwal P (2015). Long short term memory networks for anomaly detection in time series. In: Proceedings. p 89

Meerschaert MM (2013) Mathematical modeling. Academic Press, Cambridge

Mikolov T, Karafiát M, Burget L, Cernockỳ J, Khudanpur S (2010) Recurrent neural network based languagemodel. In: Interspeech, vol 2, p 3

Munteanu MC, Caliman A, Zaharia C (2017, May 30) Convolutional neural network. In: Google patents. US Patent 9,665,799

Rasmussen CE, Williams CK (2006) Gaussian processes for machine learning, vol 38, pp 715–719. The MIT Press, Cambridge, MA, USA

Rodriguez JD, Perez A, Lozano JA (2010) Sensitivity analysis of k-fold cross validation in prediction error estimation. IEEE Trans Pattern Anal Mach Intell 32(3):569–575

Rojas R (2013) Neural networks: a systematic introduction. Springer, Berlin

Rokach L, Maimon O (2005) Top-down induction of decision trees classifiersa survey. IEEE Trans Syst Man Cybern Part C (Appl Rev) 35(4):476–487

Rousseaux F (2016) Big data and data-driven intelligent predictive algorithms to support creativity in industrial engineering. Comput Ind Eng 112:459–465

Safiyullah F, Sulaiman SA, Zakaria N, Jasmani MS, Ghazali SMA (2016) Modeling the isentropic head value of centrifugal gas com-

pressor using genetic programming. In: Matec web of conferences, vol 38

Shipmon DT, Gurevitch JM, Piselli PM, Edwards ST (2017) Time series anomaly detection; detection of anomalous drops with limited features and sparse examples in noisy highly periodic data. arXiv preprint arXiv:1708.03665

Tian F, Voskuijl M (2015) Automated generation of multiphysics simulation models to support multidisciplinary design optimization. Adv Eng Inf 29(4):1110–1125

Westervelt FH (1960) Automatic system simulation programming (Unpublished doctoral dissertation). Ph.D. Dissertation, University ofMichigan

Whigham PA, et al (1995) Grammatically-based genetic programming. In: Proceedings of the workshop on genetic programming: from theory to real world applications, vol 16, pp 33–41

Yin S, Li X, Gao H, Kaynak O (2015) Data-based techniques focused on modern industry: an overview. IEEE Trans Ind Electron 62(1):657–667

Zhao Y, Chu S, Zhou Y, Tu K (2017) Sequence prediction using neural network classiers. In: International conference on grammatical inference, pp 164–169

Zhu Q, Azar AT (2015) Complex system modelling and control through intelligent soft computations. Springer, Berlin

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.