Array 15 (2022) 100187

Contents lists available at ScienceDirect

Array

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

How statistical modeling and machine learning could help in the calibration of numerical simulation and fluid mechanics models? Application to the calibration of models reproducing the vibratory behavior of an overhead line conductor

Hamdi Amroun [a,*], Fikri Hafid [b,c,d], Ammi Mehdi [a]

[a] Department of Computer Sciences, Artificial Intelligence, LIASD Lab, University of Paris 8, Saint-Denis, France
[b] RTE-R&D: 7C place du Dome, 92800 Puteaux, France,
[c] MSDA UM6P (Lot 660, Hay Moulay Rachid, Ben Guerir 43150, Morocco
[d] ENS Paris-Saclay 4, avenue des Sciences 91190 Gif-sur-Yvette, France

ARTICLE INFO

Keywords:
Machine learning
Numerical simulation
Fluid mechanics
Model calibration
Overhead line conductor

ABSTRACT

The world of fluid mechanics is increasingly generating a large amount of data, thanks to the use of numerical simulation techniques. This offers interesting opportunities for incorporating machine learning methods to solve data-related problems such as model calibration. One of the applications that machine learning can offer to the world of Engineering and Fluid Mechanics in particular is the calibration of models making it possible to approximate a phenomenon. Indeed, the computational cost generated by some models of fluid mechanics pushes scientists to use other models close to the original models but less computationally intensive in order to facilitate their handling. Among the different approaches used: machine learning coupled with some optimization methods and algorithms in order to reduce the computation cost induced. In this paper, we propose a framework which is a new flexible, optimized and improved method, to calibrate a physical model, called the wake oscillator (WO), which simulates the vibratory behaviors of overhead line conductors. an approximation of a heavy and complex model called the strip theory (ST) model. OPTI-ENS is composed of an ensemble machine learning algorithm (ENS) and an optimization algorithm of the WO model so that the WO model can generate the adequate training data as input to the ENS model. ENS model will therefore take as input the data from the WO model and output the data from the ST model. As a benchmark, a series of Machine learning models have been implemented and tested. The OPTI-ENS algorithm was retained with a best Coefficient of determination (R2 Score) of almost 0.7 and a Root mean square error (RMSE) of 7.57e−09. In addition, this model is approximately 170 times faster (in terms of calculation time) than an ENS model without optimization of the generation of training data by the WO model. This type of approach therefore makes it possible to calibrate the WO model so that simulations of the behavior of overhead line conductors are carried out only with the WO model.

1. Introduction

Overhead line operators are subject to environmental interactions, especially those that result from interaction with wind at varying speeds. Wind movements, for example, produce vibrations which can pose problems and have consequences on, for example, the aging of these conductors and physical assets. Having become aware of this observation, the research teams of Rte (Réseau de Transport d'Electricité) the French TSO (Transmission System Operator) launched several research projects aimed at establishing physical models capable of reproducing the vibratory behavior of conductors under different conditions (wind speed, etc.). This research has resulted in the development of two types of models, making it possible to have the movements of a given conductor in time and space for a given wind speed [1]. The first physical model, called the strip theory model (ST), is based on Computational fluid dynamics calculations (CFD), which are costly in calculation time and heavy in their handling (business software, control of outputs, storage, etc.). The second model, called the wake oscillator model (WO). This model was implemented in an ad hoc tool, is based on simplifications of the coupling terms, avoiding all of the

* Corresponding author.
E-mail addresses: amroun@math.univ-paris13.fr (H. Amroun), fikri.hafid@rte-france.com (F. Hafid), mehdi.ammi@univ-paris8.fr (A. Mehdi).

https://doi.org/10.1016/j.array.2022.100187
Received 3 February 2022; Received in revised form 10 May 2022; Accepted 11 May 2022
Available online 30 May 2022

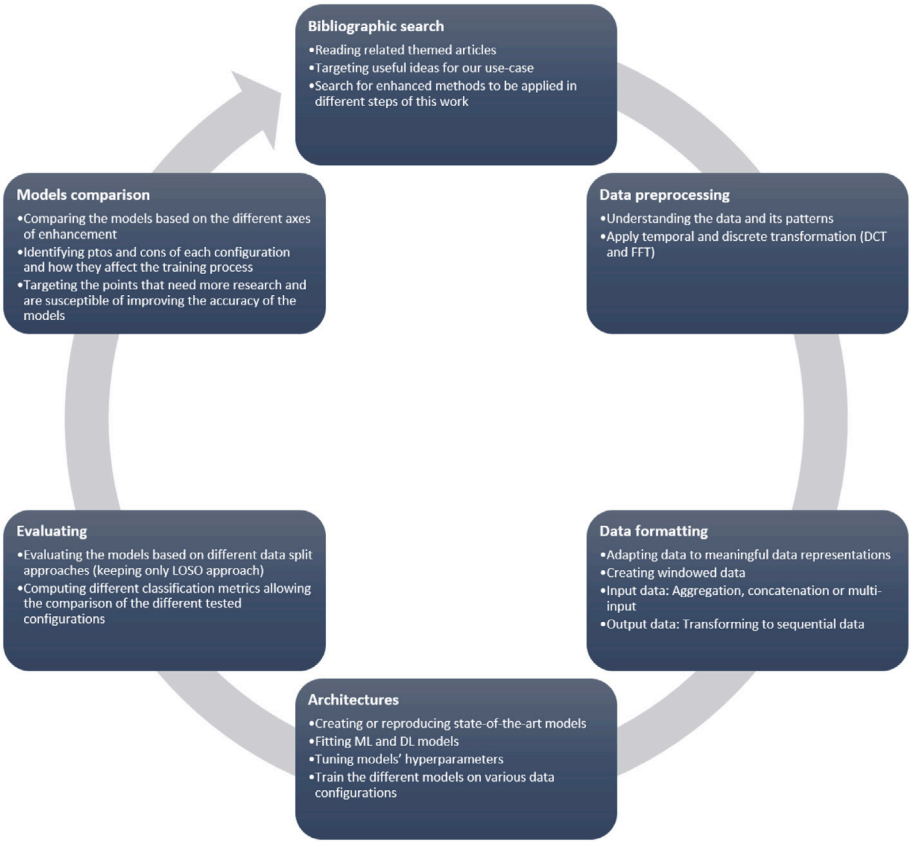

Fig. 1. Flowchart of the project.

cumbersome associated with the first physical model. Using the simplified wake oscillator model involves a choice of modeling parameters (i.e."hyper-parameters"), which are calibrated via benchmark results (the Strip Theory model results). The calibration of such a model is therefore done by comparing the output results provided by the two models, knowing that here we consider the strip theory model as the reference to reproduce. The problem with this step is that it can prove to be costly in number of simulations without proper optimization. This has so far been little covered in Rte's research on the subject, except through basic grid search methods, analogous to [2].

Academically, the revolution brought by machine learning models in many fields, particularly in fluid mechanics and engineering, has changed the practices and the way physicists and researchers approach a problem in this field. Indeed, these different methods of numerical modeling and scientific computing generate an enormous amount of data, which must be regularly analyzed and synthesized in order to be able to process them in order to generate insights. In this context, and particularly in the field of numerical simulation for fluid mechanics, different strategies for combining physical and ML approaches have been established very recently in the literature in order to match an approximate or simplified model to a certain model considered. how being a Ref. [1–3].

Machine learning algorithms incorporated here will require additional work in order to have results in an acceptable time. Indeed, the Wake Oscillator model used here will require optimization in order to avoid having it run several times for the same time series. It is therefore a matter of offering techniques making it possible not only to lighten the calculations of the WO but also not to have to do it manually by incorporating a method for optimizing the WO Model. Therefore, the incorporation of an optimization model will make it possible to generate the "good data" of training that the ENS model will recover to be trained and predict the values of y/d (the behavior of the cable) and thus avoid an interminable wait for manual generation of training data [4–12,12–39]. (see Fig. 1).

The objective of this paper is to propose a new strategy for using ML coupled with an optimization algorithm to calibrate hyper-parameters of a wake oscillator model on simulations from a strip theory model. The idea is to allow the WO model to self-adapt thanks to the use of a machine learning model which will take the data of the WO as input and will produce the corresponding predictions as an output, i.e. ST model data. In the end, the WO model will be able to produce the expected outputs without manual calibration as well as in an acceptable time. The idea here is not only to produce data that could have been obtained with the ST model with the WO model but also to be able to do it in an automated and rapid manner while guaranteeing a better approximation result.

To achieve this, we propose a processing pipeline consisting of three stages: (i) Data collection, (ii) data modeling and (iii) data visualization. The first step is to collect the necessary data by running the WO model to build a Machine learning model. Here, we will use several optimization algorithms in order to find the best values of the WO parameters and build the training/test/validation sets. The HyperOpt optimization algorithm will be privileged here because of its adaptation to our multi-parameter and multidimensional problem. Then, we will propose a methodology allowing to build our ENS model. Finally, for comparison, a series of models will be tested in order to compare them to our model and choose the best.

In this paper, we explore an approach based on the use of an ensemble learning approach (ENS) coupled with an optimization algorithm to calibrate the WO model so that it can be used to approximate the ST model. Our approach is subdivided into three stages: The first step is to collect data that will be used to build our ENS model. The data collection requires the use of the WO model using the correct values of the hyper parameters in order to have data representative of the problem, that is, close to the ST model. The second step consists of

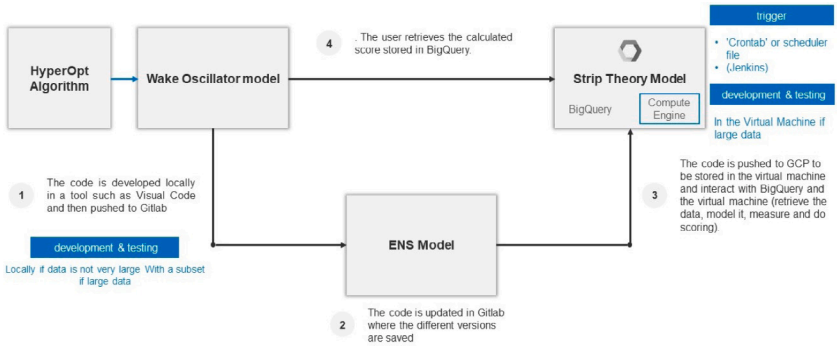

Fig. 2. OPTI-ENS framework.

modeling: the data thus generated will be used to build our ENS model. The third step is to evaluate the model built using data that did not participate in its design.

The rest of this paper is organized as follows: Section 3 presents the methodology followed to calibrate this prediction model. Section 4 presents the results of this study. Section 5 presents a discussion of these results and finally we end with a conclusion and perspectives as well as some references.

2. Methodology

Fig. 3 summarizes the framework of our approach. In this part, we present our methodology allowing us to optimally predict the vibratory behavior of an overhead line conductor. Our approach is composed of two main steps: (i) data collection via the WO model and (ii) the build of a machine learning model allowing to take the WO data as input and predict the values of y/d output. The theoretical foundations of ST models and its WO approximation have been discussed in a previous work [40]. In this work, it was a question of defining the theoretical frameworks of the WO and ST models. A first attempt to predict the values of y/d from this WO model has been established. In this paper, it is much more about improving the work already done previously by us. One of the limitations noted previously is the heaviness of the calculations to reach and build a training database. The second limitation is being able to run the learning model locally, which prompted us to distribute it in the cloud. In the following, this methodology part will mainly comprise two stages: (i) the first is to incorporate a model for optimizing the collection of data coming from the WO. (ii) the second step consists in training a Machine learning model making it possible to predict the values of y/d. The framework proposed here is called OPTI-ENS. OPTI to refer to the optimization algorithm incorporated in the training data collection phase and ENS to refer to our training model (Ensemble model).

2.1. OPTI-ENS model for the calibration of the Wake Oscillator model

We initiated this methodology by saying that it is about developing an optimized approach allowing to calibrate the WO model from the data of the ST model. This approximation was only possible by playing on the values of the hyperparameters of the WO model. This causes problems in particular of cumbersome handling and difficulties in finding the optimum. Indeed, a single data configuration (time series coming from the ST model) would require hundreds and hundreds of executions of the WO model and each execution would require between 2 min and 40 min depending on the length of the time series and according to the types of machines on which we choose to do the simulations. An optimization of the simulation and the recovery of the data is therefore necessary (see Fig. 2).

We propose here a framework allowing to automatically and optimally collect the data allowing the calibration of the Wake Oscillator model, to create a training model and then to predict the values of y/d. The processing pipeline is made up of three phases: (i) data collection using the Wake Oscillator model, (ii) data generation via the Hyperopt model and (iii) the build of the ENS predictive model to predict the values of y/d.

2.1.1. Optimization of the data collection process

In this study, our OPTI-ENS model operates directly on time-series WO data in order to predict the values of y/d. WO data can be obtained from the executions of the WO model, following the different values of its hyper parameters. Our dataset is composed of the following elements: 10 variables, including the different values of the hyperparameters of the model, namely (md, U[m/s], d[m], m[kg/m], L[m], H[N], Nt, Dt[s], tf[s],ymax[m]) and the y/d values. The values of y/d are the values to be calculated using the model of WO

Figs. 12, 13 and 14 show an overview of the values of y/d of the two models (WO and ST) according to the values of the hyperparameters. Adjusting these values makes it possible to improve the approximation between the two models, as we can see in Fig. 13, and 14. Fine tuning of the hyper-parameters makes it possible to achieve an optimal configuration as in figure 14. This is an example of a single time series from out database. This consists of a total of 60 time series. This represents a total recording of nearly 5620 s. We need a suitable metric to compare 2 time series. One of the widely used metrics is the RMSE metric. This method requires having 2 time series of the same length, ie with the same number of points. In order to obtain time series with the same number, we apply a transformation on the reference model, basing the definition of the timestep (dt) and the output step (dr) on the final time of the WO model and on the number of total points.

$$tf(finaltime) = Dt/Dr \tag{1}$$

This formula makes it possible to obtain a time series of the ST model having the same number of points as the WO model. This subsequently allows the use of methods to compare 2 time series including the RMSE.

Now that the data is preprocessed, the next step discusses the modeling. The use of the learning framework as it is will cause a problem of flexibility despite the effectiveness of the method. Indeed, the learning pipeline will always feed on the data of the WO generated manually. In order to lighten and improve this approach, a method for optimizing the generation of WO data has been implemented. The study of different optimization methods already widely used in the state of the art, in particular for the Operational Research community, led us to explore one of them, namely HyperOpt.

The search space in our case is the list of the parameters used in the WO model with their respective value ranges. One of the advantages provided by Hyperopt is the possibility to define the ranges of values according to particular distributions (according to a uniform or log-normal distribution for example), to define at the same time distributions of discrete values for certain parameters and floating values for others.

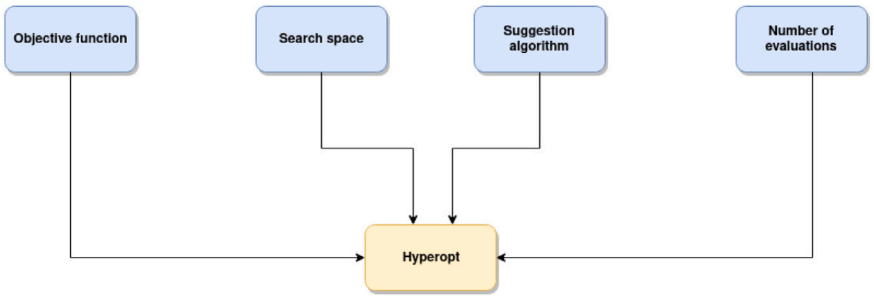

Fig. 3. Hyperopt arguments.

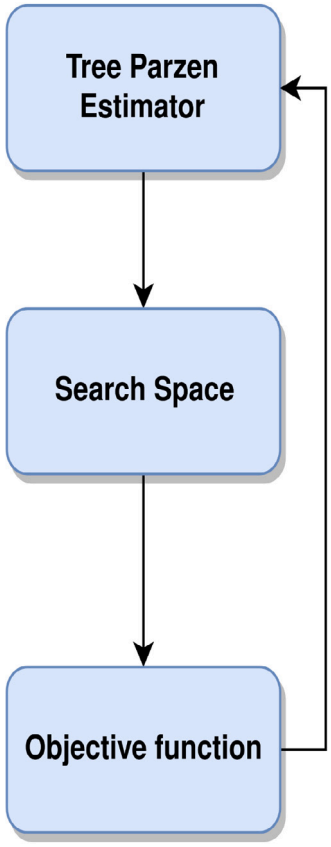

Fig. 4. Hyperopt algorithm.

The search space also accepts a nested search space within another. The suggestion algorithm is the heart of the Hyperopt algorithm. It makes it possible to assess which part of the search space is promising by suggesting combinations of promising parameters to pass to the objective function. The mainly used suggestion algorithm is the Tree-Parzen Estimator (TPE). Finally, the number of evaluations is used to define the number of times the objective function will be evaluated. The higher the number of evaluations, the more the algorithm can explore the ranges of values to find the optimal combination of parameters (see Fig. 4).

The general operation is based on the suggestion algorithm which evaluates the search space in order to optimize the search towards sub-ranges of parameters which may be interesting to explore. These parameters are then supplied as input to the cost function, the output of which must be minimized. The algorithm keeps in memory the best result obtained.

The TPE algorithm is based on a search history that the algorithm builds up over the course of Hyperopt iterations. This history is composed of tuples composed on one side of the parameters and on the other side the output obtained by the cost function. First, TPE sorts the tuples according to the value obtained by the objective function and then separates them into 2 sets. Subsequently, TPE applies an estimation by kernel making it possible to obtain a so-called density probability of the parameters in the 2 sets. Finally, we recover the minimum of the ratio between the 2 sets. The combination of parameters obtained is stored in the search history and passed to the cost function.

We have therefore created a search space that the Hyperopt algorithm can explore. Nevertheless, we encountered difficulties related to the RAM consumption which increased exponentially. This was due to the fact that the generated simulations accumulated in the RAM as well as the pp parameter which was set to "True" made that the display of the results on the execution console overloaded the programming tool. The systematic suppression of the simulation after its use for calculations as well as the definition of the pp parameter to "False" made it possible to solve this problem. We then applied the Hyperopt algorithm to our Time-Series. For this, we used iterations ranging from 10 to 550, variations in the search range of the parameters, as well as the use of the Tree-Parzen suggestion and Simulated annealing algorithms. With this configuration of parameters, we have not succeeded in obtaining results which are sufficient to be integrated into the training set. In addition, the execution time of the Hyperopt algorithm did not allow to perform more iterations to refine the result. Among the exploitable optimization methods that we have listed, is the method based on simulated annealing (Fig. 8). We therefore tried to use this algorithm to optimize the parameters of the WO model [41].

Simulated annealing is an algorithm inspired by the cooling of metals. The principle consists in defining a starting temperature value, a temperature decrease rate, as well as the minimum limit that the temperature can reach. The algorithm starts by selecting an initial point and evaluates the neighboring points using a cost function. If the neighboring point is better than the current point, the algorithm keeps it in memory as the best result. The heart of the algorithm is based on the case where the neighboring point is worse than the current point. In the following case, there is always a chance that the result will be accepted according to an acceptance probability which decreases with the temperature value. At each iteration, the temperature decreases according to the rate of decrease. This system, called the Metropolis criterion [42], first of all makes it possible to explore the results, because for a high temperature, the probability of acceptance is high. Subsequently, as the temperature decreases, the probability of acceptance decreases, allowing the results to be exploited in the most promising region. The application of simulated annealing to the WO model did not allow to obtain interesting results. The parameters found were subject to large fluctuations and showed no signs of convergence. However, a more robust variant of simulated annealing, called Dual Annealing, seemed to offer increased robustness compared to simulated annealing. This approach is based on the combination of 2 versions of simulated annealing by combining the classic version of simulated

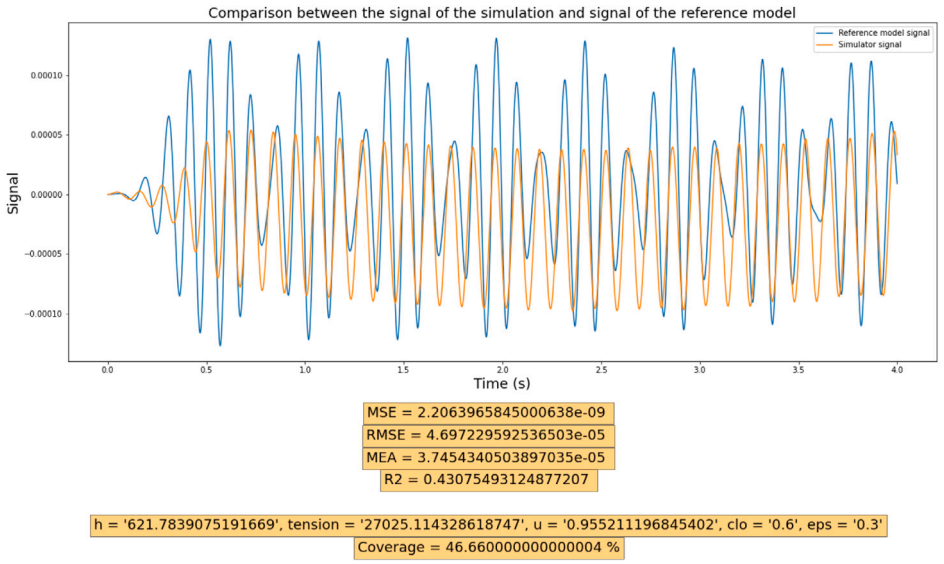

Fig. 5. Hyperopt results with 1 - R2 Score cost function.

annealing with a "fast" version which allows the selection of the parameters to be explored with a change in the method of probability of acceptance of results and a modified temperature decrease method. We therefore tried to apply the Dual Annealing algorithm to optimize the parameters of our simulator.

The results obtained are clearly more interesting than those obtained via simple simulated annealing, but are not for all that better than the results obtained via Hyperopt. In addition, the execution time of this method makes it difficult to perform many iterations necessary to assess the accuracy of the algorithm in our use case and makes the use of Hyperopt on these iteration ranges more interesting in terms of results.

Since the Hyperopt runs did not provide actionable results for a limited number of iterations, we decided to migrate the runs to a distributed instance. This instance, which has greater computational resources, makes it possible to perform calculations more quickly and thus to use the Hyperopt algorithm for a number of iterations greater than an execution on local machines. The architecture chosen for this project is an AWS Sagemaker with a distributed ml.c4.2xlarge instance. The code used in this instance is loaded from the Github (Figs. 8 and 9).

Until now, the Hyperopt algorithm has used an RMSE cost function. We tried to use a cost function based on the R2 score. The R2 Score is a metric for evaluating the dispersion of results between 2 sets of data. A negative result indicating a less good regression than a straight line, a zero result indicating a clear difference and a result close to 1 indicating a resemblance between the 2 curves. Since Hyperopt only takes functions to minimize and not to maximize, we have chosen to define a cost function defined by the formula:

$$F(x) = 1.0 - R2Score \tag{2}$$

The application of this cost function to Hyperopt made it possible to significantly improve the quality of the results obtained. The results obtained are much more interesting visually than the results obtained with the RMSE cost function. It also helps to establish a more mathematically robust metric for evaluating the resemblance between Time-Series which until now was mainly evaluated on their visual tendencies and appearance (see Fig. 5).

The optimization method being chosen, Hyperopt will provide the training data to a train a pipeline in order to predict the values of y/d. In the next section, we propose a new approach for predicting ST data with the best precision.

2.1.2. Structure of OPTI-ENS model

Our OPTi-ENS model is made up of two bricks: a Hyperopt type optimization algorithm followed by a learning algorithm. The training data were therefore generated thanks to the use of Hyperopt on the WO model. We are at this stage ready to train our ENS model with this training data. The technical architecture of our ENS model is composed of 10 XGboost type models (to form an XGboost drill) and a deep neural network (DNN). All the XGboost and DNN models take as inputs the data of the WO model, namely the values of y/d as well as the values of the hyper-parameters of the WO model which made it possible to obtain these y/d values. These models individually produce the predictions that serve as metadata to feed the Ridge Regression model. The final output of the OPTI-ENS model therefore takes the form of a weighted average of all the outputs of all the models (XGboost and DNN) taken individually.

For the given input dataset $B = \left[H_1, H_2, \ldots . H_n\right]$, where H_i denotes the input vector consists of WO input time series data and hyper-parameters values $\left(H_i\right), H_i = \langle H_1, H_2 \ldots H_n \rangle$, we define a function to predict y/d of ST model at time $t + 1$ on the basis of WO hyper parameters values.

It can be given as follows:

$$\hat{W}_{w+1} = f(H)_w = S\left(H_1, H_2, \ldots . H_{n+1}\right)_{w+1} \tag{3}$$

where $\hat{W}_{w+1}$ denotes the predicted y/d, $S(.)$ represents the function of proposed ENS model which is a combination of XGBoost and DNN models as given below:

$$S(H) = \left[S_1(H), S_2(H), \ldots, S_{m+1}(H)\right] \tag{4}$$

where $S_1(H), S_2(H) \ldots, S_m(H)$ denotes the m XGBoost models (i.e. in XGBoost forest as explained in the previous section) and $S_m(H)$ denotes the DNN in the ENS model. These base models are trained to predict y/d as follows:

$$\begin{bmatrix} \hat{w}_1 = S_1(H) \\ \hat{w}_2 = S_2(H) \\ \cdot \\ \cdot \\ \hat{w}_{m+1} = S_{m+1}(H) \end{bmatrix} \tag{5}$$

where $\hat{w}_i$ is the y/d predicted by individual base models (i.e. XGboost and DNN). The outputs of these base models are given as input to the Ridge Regression, which assigns weights to each base model to give the final y/d prediction as explained here:

$$\hat{u} = k_0 + k_1 * \hat{u}_1 + k_2 * \hat{u}_2 + \cdots \cdots + k_{m+1} * \hat{u}_{m+1} \tag{6}$$

where k_i denotes the weights assigned to each base model (XGboost and DNN). These weights are optimized using the square of the sum of the difference between the actual y/d and predicted y/d, as follows:

$$C(k) = \frac{1}{2n}\sum_{i=1}^{n}\left(u_i - \hat{u}_i - 1\right)^2 + (\lambda - 1)\sum_{j=1}^{m} k_j^2 \tag{7}$$

where C(k) is the function which is used to optimize the weights of the Ridge Regression model and z_i is the actual y/d.

Each XGBoost sub-model is trained with 80% of randomly selected training data to ensure that the data and each model is trained on separate hyperparameters. The Regression Ridge model is added in order to integrate each prediction result of the y/d value of each submodel independently. The literature suggests that to determine the generalization performance of models, our dataset is divided into three subsets: training, validation, and test set, having 80%, 10%, and 10% of total data, respectively.

- Training set is used to train and build the sub models XGBoost and DNN. - Validation set is used to determine the performance of different sub-model architectures (number of regression trees in the XGBoost, number of hidden layers and units or neurons for the DNN architecture).‘

- Test set is used to determine the error rate of the generalization by the mean of the root mean square error (RMSE) and mean bias error (MBE).

2.1.3. Implementation of our approach

In this section, we propose to approach the fine tuning of our model. This process is the only guarantee of the quality of the model, its capacity for generalization and its robustness which, according to the literature, remain important elements to define especially when it comes to creating machine learning models for a field such as fluid mechanics.

(a) *X*GBoost

We start by addressing and explaining the foundation of the best XGboost model structure. A single XGboost model is firstly evaluated on the validation set using a Grid search technique. this grid search technique must find the best combination between two parameters: (a) number of regression trees (XGBoost) $(n\_$ estimators, $k)$ and (b) maximum depth of a regression tree (d). A set of XGBoost models are used to build an XGBoost forest. An optimal number was found by evaluating the model in the validation set.

(b) Deep neural network

The structure of this model which was built using Tensorflow library is composed of (a) input layer taking into account the different values of the hyper-parameters of the WO model, the values of y/d as a function of time, (b) several separate layers with several neurons and having as an activation function the “Relu” function and finally (c) an output layer with a single neuron representing the value of y/d having a linear activation function. The number of hidden layers was calculated as follows: we start with a single hidden layer with a number of epochs ranging from 120 to 380 in steps of 10. Then gradually we add more hidden layers and RMSE values are calculated in the training, test and validation sets respectively. We come to 8 hidden layers that produced the best result in terms of RMSE. Beyond this step, the model enters an overfitting phase. To this end, a dropout layer is added in the network as regularization method to prevents the over-fitting, improves the generalization performance as well. The particularity of the model that we used is to allow a better predictive capacity when the DNN model is called upon in the choice and the decision-making process. Indeed, we introduce here a criterion allowing to increase the number of neurons by the modification of its activation function ReLU between two limits as follows:

$$\mathrm{f(y)} = \max\left(t_{\min+1}, y\right) \tag{8}$$

Where $t_{\min}$ is the constant value for the new ReLu In the DNN, $t_{\min-1} = 0$, while $t_{\min-1} < 0$.

The other value of ReLu is written as follows:

$$\mathrm{f(y)} = \min\left(t_{\max-1}, \max\left(t_{\min+1}, y\right)\right) \tag{9}$$

Where $t_{\min+1}$ and $t_{\max-1}$ are the constant values for the ReLu. In the ReLU used in our DNN model, $t_{\min} = 0$ and there is no $t_{\max}$.

Our DNN was used as a regressor, which can extract features by itself and without any domain specific knowledge about the simulation time series data. By skipping the procedure of extracting features, the model could become more responsive. The training process is divided into two steps: pre-training and fine-tuning. Pretraining is unsupervised, and an initial network will be obtained using a greedy layer-wise training algorithm. Finetuning is supervised and the parameters of all the layers will be updated using back propagation algorithm. The following notations are used to denote the components of the network:

- $Input = h_0$ Input layer.
- hid_i (i = 1, 2, … , τ -1), the i^{th} hidden layer.
- O = h_τ output layer.
- w_i(i = 1, … , τ): connection weight matrix between hid_i and hid_{i+1}.
- ρ_i(i = 1, … , τ): biases for neurons of layer hid_i when they are activated by the hid_{i+1} layer.
- ζ_i(i = 1, … , τ) : biases for neurons of layer h_i when they are activated by the hid_{i-1} layer.
- Θ: all the network settings
- Tr: training dataset - $\left[f_{\theta(x)}\right]_i$: The score associated with the i^{th} label by our parameter network. - According to [15], for two adjacent layers: h_{i-1} and hid_i activation functions are defined as $\mathrm{p}\left(h_{i-1,s} = 1 \mid hid_i\right) = \Gamma\left(\rho_{i,s} + \sum_j w_{i,j}, hid_{i,j}\right)$

$$\mathrm{p}\left(hid_{i,t} = 1 \mid hid_{i-1}\right) = \Gamma\left(\varsigma_{i,t} + \sum_j w_{i,j} h_{i,j}\right) \Gamma(\mathrm{x}) = \frac{1}{(1 + \mathrm{e}^{-\mathrm{x}})} \tag{10}$$

Where Γ (.) is the logistic function. The training of our DNN model was done in two parts: pretraining and fine-tuning.

All the implementations were carried out in an AWS environment under SageMaker under an instance of type “p4d.24xlarge”.

In the next section (results), we also compared our results with the results of other competitive models (Random Forest, Linear Regression, Adaboost, SVR, XGboost, DNN). For the DNN model we tested, is a neural network type model with a single hidden layer, an input layer representing the data and an output layer with a single neuron representing the prediction value y/d (see Fig. 6).

3. Results

In this section, we will present the different results of the prediction of the values of y/d according to different machine learning models as well as our OPTI-ENS model. This benchmark mainly concerns RMSE type errors and the R2 Score, because these two metrics are the most significant, often widely used in the state of the art, especially in the validation of machine learning models whether in regression or in classification. This section is divided into two sections: (i) The visualization of the data generated by the Hyperopt optimization algorithm which are the training data and (ii) The creation and training of our ENS model which is composed of several XGBoost models and an optimized DNN model by modifying its ReLu activation function. At the end we will also make available the prediction results, the RMSE errors, the differences in execution speed and finally some settings that we had adopted in the process of building our OPTI-ENS model.

3.1. Data visualization

The creation of the training set required the use of the model WO in order to match the data of the model ST with the data of the model WO. Figs. 13, 14 and 15 make it possible to visualize a sample of a single time series of simulation data of the data of the WO model compared to the ST model. Notice that by changing the values of the

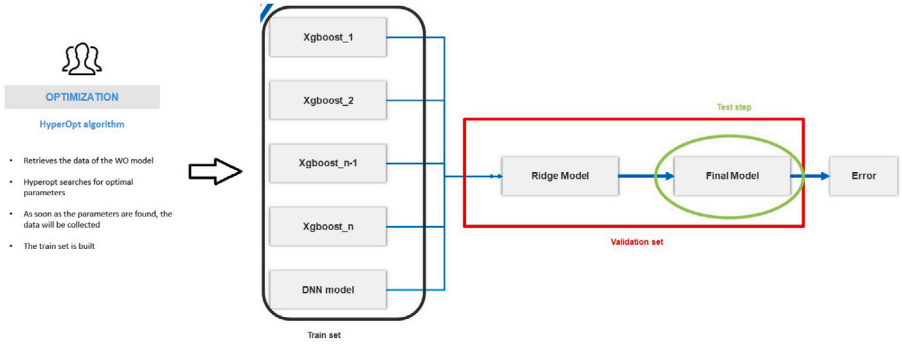

Fig. 6. The architecture of the proposed ENS model.

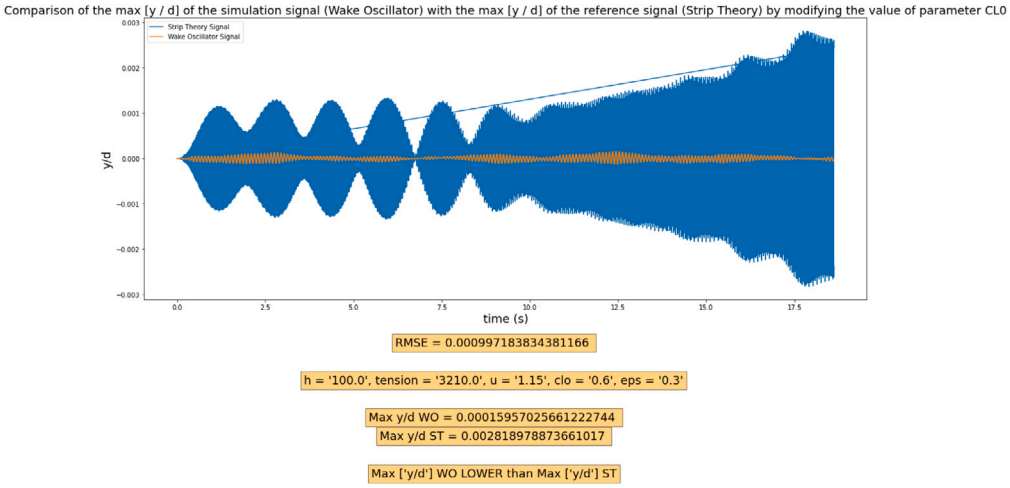

Fig. 7. Visualization of simulation data WO Vs ST models.

hyper-parameters of the model WO, the two curves come closer until a better result is reached (Fig. 15). According to our previous work, we have indicated the most promising parameters in order to converge the WO simulation model to the ST model. Among these parameters the parameter u (Wind Speed), and the modifications made were tested for a range of values going from 0.1 to 7.0. Changing this parameter accentuates the general curvature of the time series, the points are closer together, The maximum amplitude taken by the points is greater. Next, we evaluated the impact of changing the "Tension" parameter on the time series. The range of values of the voltage parameter is 1 to 40,000. The range of values being large, it has been established that the voltage has a large impact and the step to explore the interval should be small enough not to miss interesting time series to exploit. In the end, an exploration of the ranges of values with a step of 100 gives results which are more easily exploitable and transposable to other time series. Furthermore, a definition of the voltage parameter = 100 causes a significant increase in the maximum amplitude.

Following the study of the impact of the parameters u and h on the simulation time series, we extended the study to the parameters cl0 and eps defined in the configuration of the simulator. causes a linear increase in the maximum amplitude of the time series. Changing the value of eps has no impact on most time series, some time series shows an increase in amplitude for eps = 0.85. This is of course done as an example before calling on the Hyperopt optimization algorithm in order to allow automation and search for these optimal parameters quickly (see Figs. 7, 10 and 11).

3.2. Modeling

The training set created in the "Data Visualization" part was used to train a series of training models. For comparison, we compared the Y/D prediction results of our OPTI-ENS model by a series of models. Table 1 summarizes the results obtained by understanding some Machine learning algorithms in terms of R2 Score and RMSE. The R2 score gives an indication of how close the ST data is to the prediction data. The higher the R2 Score, the better the prediction quality. Since the R2 Score alone does not whistle, the RMSE was calculated (as an example) to show that the best model must also have the weakest RMSE. Figures 15 and 16 show an example of some prediction results of the Linear Regression model compared to our ENS model. The Linear Regression model (as an example) has a very high RMSE compared to the ENS model. Same remark for the R2 Score, the ENS model presents an R2 Score which is close to double that of the Linear Regression model. From the results of Figures 15 and 16, as well as the results of benchmark in Table 1, we can see that the predictive capacity of our ensemble model confirms the results of the state of the art, namely that the set models produce

Table 1
Results of our benchmark: Here, we took only two types of metrics: the RMSE to demonstrate the error made by the different models and the R2 Score to illustrate their predictive capacity (as long as R2 Score close to 1, this means that the model in question has 'better predictive ability).

Algorithm	R2Score	RMSE
Random Forest (RF)	0.322	1.1334678543987654e−05
LR	0.234	9.1254678543987654e−06
AdaBoost	0.299	6.3334678543987654e−06
SVR	0.294	7.8284678543987654e−06
XGBoost	0.390	5.6565898765678943e−06
DNN	0.400	1.7676789874657436e−08
ENS	0.699	7.5676543245673839e−09
Opti-ENS	0.697	6.0773736543334343e−09

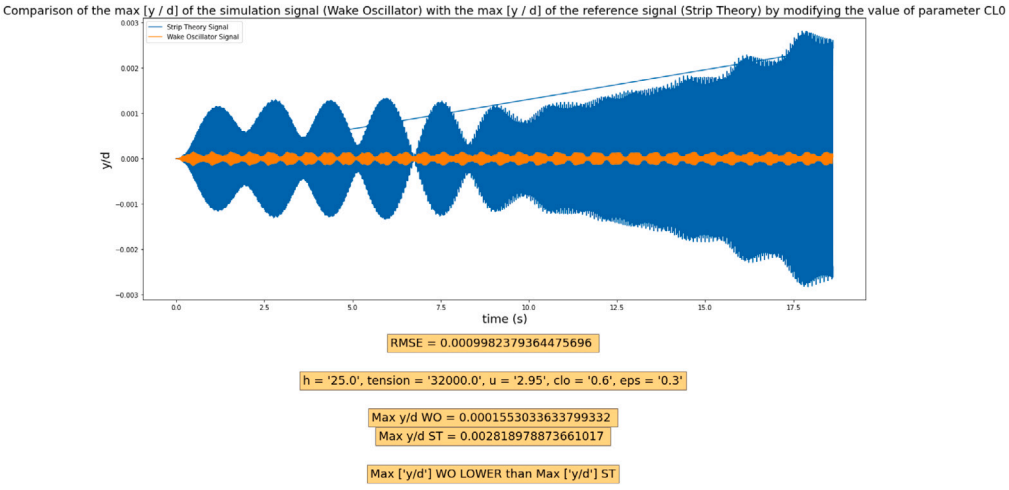

Fig. 8. Visualization of simulation data WO Vs ST.

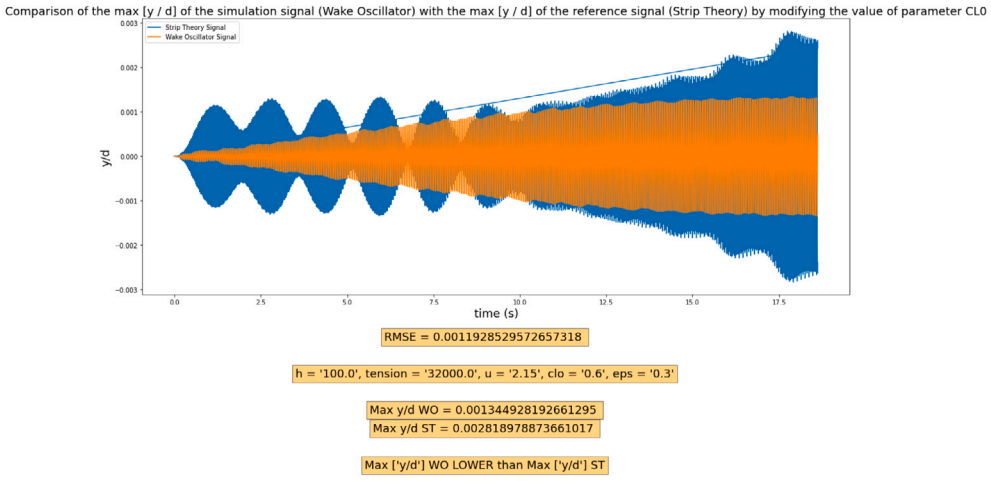

Fig. 9. Visualization of simulation data WO Vs ST.

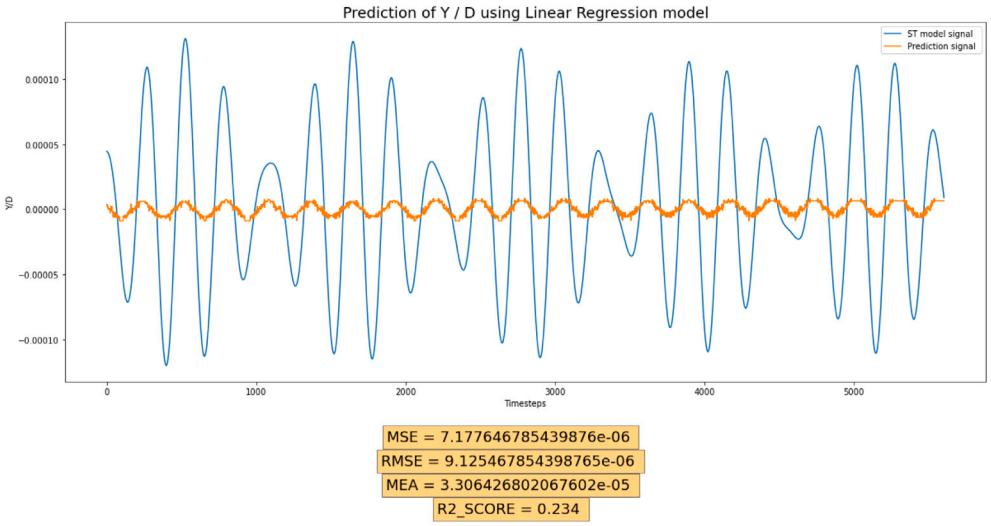

Fig. 10. Predicting ST model data by training WO data with the Linear Regression model. As you can see here, with this training model, the quality of prediction of the values of y/d is poor.

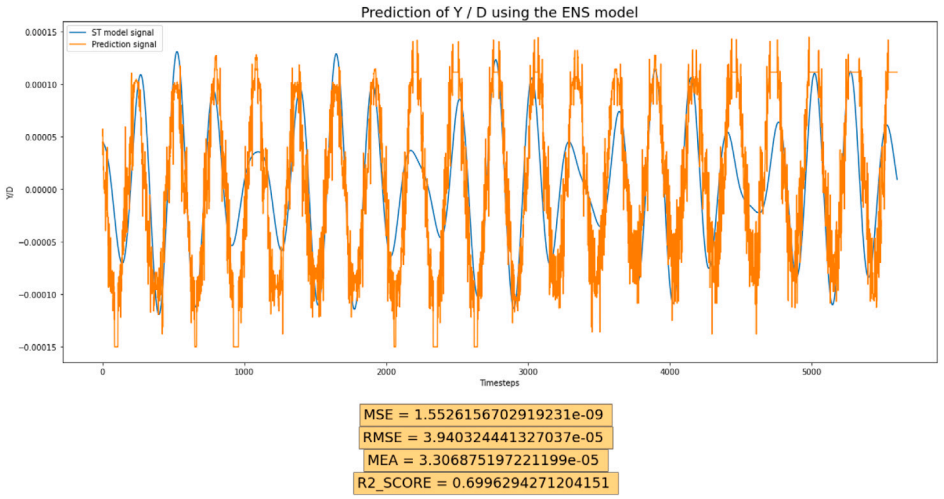

Fig. 11. Predicting ST model data by training WO data with the OPTI-ENS model. Here clearly as you can see, the prediction quality of the y/d values is clearly excellent.

Type	Evaluation	OPTI-ENS Vs ENS
Infrastructure	• GPU characteristics • CPU characteristics • RAM size	• The sizes are the same for performing both approaches.
Architecture	• Number of layers • Number of trainable parameters	• A neural network with 12 hidden layers • 48k Parameters
Training	• Number of epochs • Batch size • Data split • Search space • Data size on disc • Model size during training • Size of the trained and exported model	• 45.78k epochs • 32 • 80-20 • Grid search • 5.5 Go • 2 Mo • 1.7 Mo • 2 Mo
Indicators	• Data reading time (and preprocessing if it exists) • Training time • Prediction time • Use (in%) of GPU, CPU and RAM	• Between 20 s and 1 min • 2 min to 4 min • 5 s average • 80% average
Other	• Evolution of the loss • Evolution of metrics • Comparison of execution speeds	• ENS slower than OPTI-ENS • Same level • Faster execution speed for OPTI-ENS

Fig. 12. Here we compared our processing pipeline namely OPTI-ENS versus ENS alone without optimization. Overall, we can say that the OPTI-ENS approach is much more flexible in terms of execution speed, GPU/CPU usage, and training. This can be explained by the fact that most of the data collection processing is done via the optimization algorithm which facilitates the task afterwards for the construction phase of the learning model. It must also be said that OPTI-ENS offers much more flexibility compared to ENS in terms of the possibility of using computational resources which is more advantageous for example when one is forced to work in a "big" environment. data ", which is the case in most of the work in numerical simulation or fluid mechanics.

Table 2
Result of using Hyperopt with ENS in terms of precision and execution time.

Algorithm	R2Score	Execution time (s)
ENS+Hyperopt(OPTI-ENS)	0.697	13.76 s
ENS	0.699	2465 s

better results than models used individually. The predictive capacity of the ENS model is mainly due to the characteristics of XGBoost models of producing results on not too high quantities of data as well as the predictive power of DNN type models on time series especially when learning is done on data. almost raw time series.

Table 2 shows that the predictive capacity of the OPTI-ENS model is approximately more than 170 times greater than a conventional ENS model without the incorporation of the Hyperopt model. Fig. 17 summarizes all the elements that were taken into account (in addition to the speed of execution) in order to prove the effectiveness of our approach. Fig. 18 presents the elements necessary to test the reliability of the proposed approach. Given that the proposed model is a machine learning model, the reliability of our approach should be verified through the traditional method already recommended by most works in the state of the art, namely: the subdivision of the set of data in three sets: (i) a training set to build the model, (ii) a validation set to validate the model and finally (iii) a test set to test its ability to generalize. As part of our paper, the three points above have been checked and validated. Thus, we want to focus on a little more extensive validation through the elements given in the table of Figs. 12 & 13.

4. Discussion

The choice of the Hyperopt optimization algorithm came by experimenting with several approaches. As we explained in the "methodology" part, Given the number of parameters, we decided to apply an optimization method by gradient descent in order to converge the

Type	Test type
Ethics	• Confidentiality - data leakage o Anonymization o Pseudonymisation • Security: o Robust against cyber attacks • Reliability: o Know the strengths and weaknesses of the model o Detect biases in decision-making by the model o Investigate the advancement of classes • Explainability o Explain the nature of the outputs of the models according to the inputs chosen
Pipeline	• Data collection and preprocessing • Feature engineering • Experiment with several models and choose the best ones according to the adapted metrics • Test, optimize and validate the models and choose the best one according to its robustness • Put the model in production • Monitor model compliance and performance • Go back to the first tasks for improvement purposes
Techniques	• K-fold cross-validation : o Train the model on k-1 folds and test on 1 fold for k times • Leave-one-out Cross-Validation (LOOCV): o Train the model on n-1 observations / subjects and test on the only remaining one for n times • Leave-one-group-out Cross-Validation (LOGOCV): o Train the model on n-1 groups / populations and test on the only remaining one for n times • Nested cross-validation: o Use the same technique as the classic CV o For each iteration, we add a nested tuning loop by transforming the basic set train into a train and validation sets • Time series CV: o Each step consists of feeding the training data with more data and shifting the test set to the end of the train set
Model comparison tests	• Wilcoxon signed-rank test • McNemar's test • Paired t-test

Fig. 13. Machine learning approaches such as what we are proposing here in this paper are validated through one of the techniques that we mentioned in this table. Here in our paper, we validated the effectiveness of our two-step approach: (i) first we used a dataset subdivided into three datasets: a training dataset, a validation dataset and a test dataset for test its capacity for generalization. The state of the art therefore suggests this mode of validation. On the other hand, in a more general framework, we could verify the effectiveness of an approach such as ours by verifying at least four essential points, namely; ethics, treatment pipeline, validation techniques and test models. There are also other validation modes that we had also tested which is based on adding another dataset and using the model as is. The results are not far from what we proposed here. As an indication, we had obtained an RMSE 8.97e−09 which remains a little higher than what we obtained with our model during its construction which remains relatively normal because any machine learning model trained, during its use, the result obtained will be less than that obtained during of its creation then gradually the results improve because the model also learns from its mistakes and self-corrects.

results of the model of WO with the model of ST. The gradient descent method is a method for finding the minimum of a differential function. The gradient descent method is a method as the name suggests that is based on the gradient. The gradient of a function of several variables at a certain point is a vector which characterizes the variability of this function in the vicinity of this point. The gradient is calculated from the derivative of the function. If the gradient is high, the function is far from the minimum, if the gradient is low or zero, then we are close to the minimum. The cost function must subsequently be minimized via the gradient descent algorithm in order to obtain a WO model close to the ST model. In our case, and according to the work previously carried out internally at RTE, the cost function is not accessible and therefore not differentiable. This prevents the use of the gradient descent method. For the cost function, we looked at the root of the mean squared error (RMSE). The RMSE function being perfectly defined within the framework of the differences between the values between 2 series of points. This method evaluates the dispersion of the residues. We found that the WO time series and the ST time series were not made up of the same number of points. The time series of ST and WO had identical final time (tf), but not the same time step, therefore the number of points between the ST model and the WO model was not identical. The difference in terms of the number of points made it impossible to apply a method such as RMSE to compare the 2 models. The RTE team had indicated the most promising parameter in order to converge the WO model towards the ST model. This parameter is the u (Wind Speed). Changes to this parameter have been tested for a value range from 0.1 to 7.0. The modification of this parameter accentuates the general curvature of the time series, the points are closer together. The maximum amplitude taken by the points is greater. We then evaluated the impact of changing the voltage parameter (H) on the time series. The value range of the voltage parameter is 1 to 40,000. Since the value range is large, it has been established that the voltage has a large impact and the step used to explore the interval should be small enough not to not miss interesting time series to exploit. In the end, an exploration of the ranges of values with a step of 100 gives results that are more easily exploitable and transposable to other time series. Furthermore, a definition of the voltage parameter = 100 causes a significant increase in the maximum amplitude. Following the study of the impact of the parameters u and h on the simulation time series, we extended the study to the parameters cl0 and eps defined in the configuration of

the WO. The increase in the value of the parameter cl0 causes a linear increase in the maximum amplitude of the time series. Changing the value of eps has no impact on most time series, however some time series show a increase in amplitude for eps = 0.85. The manual approach made it possible to obtain interesting results. By iterations and using a dichotomy exploration method, we arrived at simulation results that visually approximate the WO model. However, the manual approach did not give conclusive results on the other Time-Series. This is mainly due to the WO execution time for the corresponding Time-Series. The 2 Time-Series on which we managed to obtain promising results were Time-Series which in terms of length (final time) were the shortest, this induces a lower number of points and therefore were those which require the lower execution time. This made it possible to reduce the search ranges quickly. However, for other Time-Series, this approach turns out not to be effective enough. We realized that for each time step, the WO produces a value, while the model of ST as for it, produces for each time step 2 values: a positive value and a negative value. In order to be able to create a model, we first focused on the positive values of the reference model. We retrieve the data from the WO to train the model and the model data from ST as test data. We apply this procedure by first concatenating the time series and then separating the simulation and reference data with 80% of the data for training and 20% for validation.

In some cases, we get non-exploitable values in the time series of ST such as missing values (NA). These values are redacted from the learning set. Finally, we make sure that the length of the vector containing the reference data is the same length as that for the training. We have also tried a new approach based on simulation generation whose maximum amplitude is closest to the maximum amplitude of the ST model. However, this approach did not provide better models than the classical approach based on visual resemblance.

The results of prediction of the values of y/d made it possible to show the predictive capacity of our OPTI-ENS model compared to models well known in the state of the art [43–46]. Basic models such as Linear Regression showed very low prediction as well as very high RMSE. A small improvement was observed using the Random Forest model but this remains insufficient in terms of prediction error which remains very high. The DNN-type model also showed an improvement in the predictive quality in terms of RMSE and R2 Score but remains below 0.5 in terms of R2 Score. The OPTI-ENS model has surpassed all the models tested (Table 1) and obtains better results in terms of R2 Score and a better error (RMSE). This explains the power and predictive capacity of the ENS model because coupled with the DNN (which was designed to have a better quantity of neurons thanks to the modification of the ReLu function that we had proposed as well as the proposed pre-processing method) and a series of XGBoost, the DNNs participated in better managing the raw data coming from the WO model. In addition, the XGBoost models have brought a predictive capacity which has improved the prediction results. These results are in line with the state of the art in which several studies have shown the predictive power of this type of models [47,48]. Our model in particular took its power from its structure which is composed of XGBoost models which are known in regression spots but also the power of models with a neural architecture (DNN) added more flexibility in the implementation of the model which positively influenced the quality of the prediction [49–52].

5. Conclusion

In this paper, we have proposed an approach allowing to calibrate in an optimal and automatic way a model allowing to reproduce the vibratory behavior of an overhead line conductor. The Wake Oscillator model is actually the approximate representation of a physical Strip Theory model. The latter being complicated to implement in order to generate simulation data, the WO model made it possible to approximate the ST model. In order to be able to use WO's model in this approximation process, it was first necessary to do it manually and time series by time series. This is obviously not conceivable given the complexity of computation and the heaviness induced by this manipulation. The WO model therefore required an additional intervention or modification in order to efficiently generate the data of the WO model. For this, we used a model for optimizing the computation of the WO by the use of machine learning. Indeed, the learning model takes as input the time series generated by the WO model by means of an optimization model: Hyperopt and produces as output the outputs that would have produced an ST model. Our optimization and then machine learning framework made it possible to predict the output values of an ST model with a very low error and with a very acceptable data generation speed. This study will make it possible in the near future to set up a tool directly allowing to have simulation data close to those that would have been obtained using the ST model alone and without optimization.

CRediT authorship contribution statement

Hamdi Amroun: Conceptualization, Methodology/Study design, Software, Formal analysis, Investigation, Resources, Data curation, Writing – original draft, Writing – review and editing, Visualization. **Fikri Hafid:** Methodology/Study design, Validation, Formal analysis, Writing – review and editing, Supervision, Project administration. **Ammi Mehdi:** Methodology/Study design, Validation, Formal analysis, Writing – review and editing, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

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

Acknowledgments

We would like to thank the entire RTE (Réseau de Transport d'Electricité) team for their trust in us by entrusting us with this very ambitious and interesting project. I would also like to thank the entire EuroBios team in particular John Redford for his valuable explanations and his time, Maxime Guéguin for his involvement. Without forgetting Emmanuel for the documentation on the Wake Oscillator and its explanations. We would also like to thank Youness El Marhraoui for his precise help in the use and generation of WO data. I also thank our two interns: Massil Ksouri and Nithushan for their involvement.

References

[1] Zhang L, Redford J, Hafid F, Ghidaglia J-M, Gueguin M. VIV modelled using simplified cable dynamics coupled to sub-critical cylinder flow simulations in a moving reference frame. Eur J Mech B Fluids 2021;85:214–31.
[2] Violette R, De Langre E, Szydlowski J. Computation of vortex-induced vibrations of long structures using a wake oscillator model: comparison with DNS and experiments. Comput Struct 2007;85(11–14):1134–41.
[3] Ge Z, Song Z, Ding SX, Huang B. Data mining and analytics in the process industry: The role of machine learning. IEEE Access 2017;5:20590–616.
[4] Chang JC, Amershi S, Kamar E. Revolt: Collaborative crowdsourcing for labeling machine learning datasets. In: Proceedings of the 2017 CHI conference on human factors in computing systems. 2017, p. 2334–46.
[5] Oliynyk AO, Buriak JM. Virtual issue on machine-learning discoveries in materials science. ACS Publications; 2019.
[6] Fan Y, Li W, Chen N, Ahn J-H, Park Y-J, Kratzer S, et al. OC-SMART: A machine learning based data analysis platform for satellite ocean color sensors. Remote Sens Environ 2021;253:112236.
[7] Romagnoni A, Jégou S, Van Steen K, Wainrib G, Hugot J-P. Comparative performances of machine learning methods for classifying Crohn Disease patients using genome-wide genotyping data. Sci Rep 2019;9(1):1–18.
[8] D'Amour A, Heller K, Moldovan D, Adlam B, Alipanahi B, Beutel A, et al. Underspecification presents challenges for credibility in modern machine learning. 2020, arXiv preprint arXiv:2011.03395.

[9] Ludwig J, Mullainathan S, Spiess J. Augmenting pre-analysis plans with machine learning. In: AEA papers and proceedings, vol. 109. 2019, p. 71–6.

[10] Fung R, Villar J, Dashti A, Ismail LC, Staines-Urias E, Ohuma EO, et al. Achieving accurate estimates of fetal gestational age and personalised predictions of fetal growth based on data from an international prospective cohort study: a population-based machine learning study. Lancet Digit Health 2020;2(7):e368–75.

[11] Abrell J, Kosch M, Rausch S. How effective was the UK carbon tax?-A machine learning approach to policy evaluation. In: A machine learning approach to policy evaluation (April 15, 2019). CER-ETH–center of economic research at ETH Zurich working paper, vol. 19. 2019, p. 317.

[12] Hughes GL, Lones MA, Bedder M, Currie PD, Smith SL, Pownall ME. Machine learning discriminates a movement disorder in a zebrafish model of Parkinson's disease. Dis Models Mech 2020;13(10).

[13] Malik MM. A hierarchy of limitations in machine learning. 2020, arXiv preprint arXiv:2002.05193.

[14] Zhao H, Hua Q, Chen H-B, Ye Y, Wang H, Tan SX-D, et al. Thermal-sensor-based occupancy detection for smart buildings using machine-learning methods. ACM Trans Des Autom Electron Syst (TODAES) 2018;23(4):1–21.

[15] Probst DM, Raju M, Senecal PK, Kodavasal J, Pal P, Som S, et al. Evaluating optimization strategies for engine simulations using machine learning emulators. J Eng Gas Turbines Power 2019;141(9).

[16] Hepler NL, Scheffler K, Weaver S, Murrell B, Richman DD, Burton DR, et al. Idepi: rapid prediction of HIV-1 antibody epitopes and other phenotypic features from sequence data using a flexible machine learning platform. PLoS Comput Biol 2014;10(9):e1003842.

[17] Yazdani M. The ideal teaching machine. In: Computers and modern language studies. 1986, p. 144–53.

[18] Klaib AF, Alsrehin NO, Melhem WY, Bashtawi HO, Magableh AA. Eye tracking algorithms, techniques, tools, and applications with an emphasis on machine learning and internet of things technologies. Expert Syst Appl 2020;114037.

[19] Kinn D. Reducing estimation risk in mean-variance portfolios with machine learning. 2018, arXiv preprint arXiv:1804.01764.

[20] Srivastava A, Karpievitch YV, Eichten SR, Borevitz JO, Lister R. HOME: a histogram based machine learning approach for effective identification of differentially methylated regions. BMC Bioinformatics 2019;20(1):1–15.

[21] Smith ASG. Application of machine learning algorithms in adaptive web-based information systems [Ph.D. thesis], Middlesex University; 1999.

[22] Zakutayev A, Wunder N, Schwarting M, Perkins JD, White R, Munch K, et al. An open experimental database for exploring inorganic materials. Sci Data 2018;5(1):1–12.

[23] Bhattacharjee A, Soni B, Verma G, Borgohain SK, Gao X-Z. Machine learning, image processing, network security and data sciences: Second international conference, MIND 2020, Silchar, India, July 30–31, 2020, proceedings, part II, vol. 1241. Springer Nature; 2020.

[24] Cooper K, Baddeley C, French B, Gibson K, Golden J, Lee T, et al. Novel development of predictive feature fingerprints to identify chemistry-based features for the effective drug design of SARS-CoV-2 target antagonists and inhibitors using machine learning. ACS Omega 2021;6(7):4857–77.

[25] Kim S, Tasse D, Dey AK. Making machine-learning applications for time-series sensor data graphical and interactive. ACM Trans Interact Intell Syst (TiiS) 2017;7(2):1–30.

[26] Norouzzadeh MS, Morris D, Beery S, Joshi N, Jojic N, Clune J. A deep active learning system for species identification and counting in camera trap images. 2019, arXiv preprint arXiv:1910.09716.

[27] Pillai S, Good B, Richman D, Corbeil J. A new perspective on V3 phenotype prediction. AIDS Res Hum Retroviruses 2003;19(2):145–9.

[28] Vento NFR. Hypothesis-based machine learning for deep-water channel systems [Ph.D. thesis], Colorado State University; 2020.

[29] Chandorkar M. Machine learning in space weather [Ph.D. thesis], Université of Eindhoven; 2019.

[30] Agajanian S. Development of integrated machine learning and data science approaches for the prediction of cancer mutation and autonomous drug discovery of anti-cancer therapeutic agents. 2020.

[31] Yaganapu A. Detection of SNPS associated with bone loss rate by using machine learning approaches. 2020.

[32] Nam S. Understanding and supporting vocabulary learners via machine learning on behavioral and linguistic data [Ph.D. thesis], 2020.

[33] Bose S. Towards explainability in machine learning for malware detection [Ph.D. thesis], The Florida State University; 2020.

[34] Hayes TF. A qualitative exploratory study of emergency medicine clinician perspectives on clinical decision support systems (CDSS) rooted in machine learning in England [Ph.D. thesis], 2020.

[35] Shi L. Leveraging big data and machine learning technologies for accurate and scalable genomic analysis [Ph.D. thesis], The Florida State University; 2020.

[36] Caley JA. A survey of systems for predicting stock market movements, combining market indicators and machine learning classifiers. 2013.

[37] Araci D. Finbert: Financial sentiment analysis with pre-trained language models. 2019, arXiv preprint arXiv:1908.10063.

[38] Shah N, Zhou D, Peres Y. Approval voting and incentives in crowdsourcing. In: International conference on machine learning. PMLR; 2015, p. 10–9.

[39] Fletcher T. Polarizable multipolar electrostatics driven by kriging machine learning for a peptide force field: Assessment, improvement and up-scaling [Ph.D. thesis], University of Manchester; 2014.

[40] Amroun H, Ammi M, Hafid F. Proof of concept: Calibration of an overhead line conductors' movements simulation model using ensemble-based machine learning model. IEEE Access 2021.

[41] Dowsland KA, Thompson J. Simulated annealing. Handbook of natural computing. Springer-Verlag; 2012, p. 1623–55.

[42] Wu X, Bai W, Xie Y, Sun X, Deng C, Cui H. A hybrid algorithm of particle swarm optimization, metropolis criterion and RTS smoother for path planning of UAVs. Appl Soft Comput 2018;73:735–47.

[43] Loubere P. Deep-sea benthic foraminiferal assemblage response to a surface ocean productivity gradient: a test. Paleoceanography 1991;6(2):193–204.

[44] Gregor L, Lebehot AD, Kok S, Scheel Monteiro PM. A comparative assessment of the uncertainties of global surface ocean CO 2 estimates using a machine-learning ensemble (CSIR-ML6 version 2019a)–have we hit the wall? Geosci Model Dev 2019;12(12):5113–36.

[45] Chen S, Hu C, Barnes BB, Wanninkhof R, Cai W-J, Barbero L, et al. A machine learning approach to estimate surface ocean pCO2 from satellite measurements. Remote Sens Environ 2019;228:203–26.

[46] Gregor L, Gruber N. Oceansoda-ETHZ: a global gridded data set of the surface ocean carbonate system for seasonal to decadal studies of ocean acidification. Earth Syst Sci Data 2021;13(2):777–808.

[47] Pesantez-Narvaez J, Guillen M, Alcañiz M. Predicting motor insurance claims using telematics data—Xgboost versus logistic regression. Risks 2019;7(2):70.

[48] Zhang X, Yan C, Gao C, Malin BA, Chen Y. Predicting missing values in medical data via xgboost regression. J Healthc Inform Res 2020;4(4):383–94.

[49] Chen T, He T, Benesty M, Khotilovich V, Tang Y, Cho H, et al. Xgboost: extreme gradient boosting. In: R package version 0.4-2, vol. 1, no. 4. 2015, p. 1–4.

[50] Aggarwal K, Kirchmeyer M, Yadav P, Keerthi SS, Gallinari P. Benchmarking regression methods: a comparison with CGAN. 2019, arXiv preprint arXiv:1905.12868.

[51] Devan P, Khare N. An efficient XGBoost–DNN-based classification model for network intrusion detection system. Neural Comput Appl 2020;1–16.

[52] Amroun H, Ouarti N, Ammi M. Recognition of human activity using internet of things in a non-controlled environment. In: 2016 14th international conference on control, automation, robotics and vision. IEEE; 2016, p. 1–6.