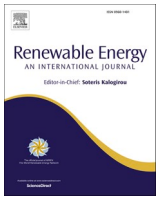
Renewable Energy 221 (2024) 119700

Contents lists available at ScienceDirect

Renewable Energy

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



Advanced hyperparameter optimization of deep learning models for wind power prediction

Shahram Hanifi [a,*], Andrea Cammarono [a], Hossein Zare-Behtash [a]

[a] James Watt School of Engineering, University of Glasgow, G12 8QQ, Glasgow, UK

ARTICLE INFO

Keywords:
Hyperparameter optimization
Bayesian optimization
Gaussian process
Acquisition function
Wind power forecasting
Random initialization

ABSTRACT

The uncertainty of wind power as the main obstacle of its integration into the power grid can be addressed by an accurate and efficient wind power forecast. Among the various wind power forecasting methods, machine learning (ML) algorithms, are recognized as a powerful wind power forecasting tool, however, their performance is highly dependent on the proper tuning of their hyperparameters. Common hyperparameter tuning methods such as grid search or random search are time-consuming, computationally expensive, and unreliable for complex models such as deep learning neural networks. Therefore, there is an urgent need for automatic methods to discover optimal hyperparameters for higher accuracy and efficiency of prediction models. In this study, a novel investigation is contributed to the field of wind power forecasting by a comprehensive comparison of three advanced techniques – Scikit-opt, Optuna, and Hyperopt – for hyperparameter optimization of Convolutional Neural Network (CNN) and Long Short-Term Memory Network (LSTM) models, a facet that, to our knowledge, has not been systematically explored in existing literature. The impact of these optimization techniques on the accuracy and efficiency of the CNN and LSTM models are assessed by comparing the root mean square error (RMSE) of the predictions and the required time to tune the models. The results show that the Optuna algorithm, using a Tree-structured Parzen Estimator (TPE) search method and Expected Improvement (EI) acquisition function, has the best efficiency for both CNN and LSTM models. In terms of accuracy, it is demonstrated that while for the CNN model all the optimization methods achieve similar performances, the LSTM model optimized by the Hyperopt algorithm, based on the annealing search method, results in the highest accuracy. In addition, for the first time in this research, the impact of the random initialization features on the performance of the forecasting models with neural networks is investigated. The proposed structures for deep learning models were examined to determine the most robust structure with the minimal sensitivity to the randomness. What we have discovered from the comparison of advanced hyperparameter optimization methods can be used by researchers to tune the time series-based forecasting models.

1. Introduction

Forecasts show that by 2050, due to the growing demand for products and services, electricity generation needs to be increased by a factor of 2.5 [1], even considering the improvement in energy efficiency in various sectors, (e.g., transportation and heating) demanded by the regulations. Currently, the majority (42 %) of electricity generation in the UK comes from gas-fired power plants, which causes many environmental issues such as global warming and increased greenhouse gas (GHG) emissions [1]. To prevent these adverse effects, and take steps towards decarbonization, the share of electricity produced from fossil fuels needs to be gradually reduced and replaced with renewable energies in a way that by the end of 2050 three-quarters of the total electricity is supplied from clean sources of energy. Fig. 1 shows the trend of electricity production of different power plants until 2050 in the UK [2].

As demonstrated in Fig. 1, the main renewable energy source considered to replace fossil fuels is wind energy, either onshore or offshore. Wind energy is the fastest-growing renewable energy with the ability to guarantee national security against global threats caused by the reduction of fossil fuel reserves and their price increase [1].

Building wind power plants has some negative environmental effects on communities and wildlife including noise, visual effects, etc [1]. However, the main obstacle against the integration of wind power into

* Corresponding author.R703C Level 7, School of Engineering, James Watt South Building, Glasgow G12 8QQ, UK.
E-mail address: Shahram.hanifi@Glasgow.ac.uk (S. Hanifi).

https://doi.org/10.1016/j.renene.2023.119700
Received 15 May 2023; Received in revised form 14 November 2023; Accepted 19 November 2023
Available online 2 December 2023

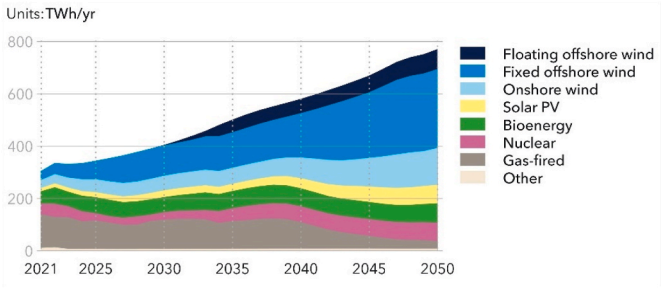


Fig. 1. UK grid-connected power generation by different power stations [2].

the power grid comes from its inherent characteristic, variability, and unpredictability [3].In the power grid, a balance between the electricity consumption and production must be maintained. Failing to do so would result in the impossibility to supply electric power in a stable manner. The provision of a high share of grid power by non-dispatchable wind energy makes this balance more difficult unless an accurate forecast of generated power is provided. The inability to provide accurate wind power prediction has various consequences. Technically, forecast divergence impedes the ability of transmission system operators (TSO) to plan the fulfilment of demand based on the available power capacity. Furthermore, from an economic point of view, the uncertainty of the total wind power forecast leads to the doubt in day-ahead and balancing market costs, which will bring financial losses to electricity providers [4].

The TSO is responsible for balancing the electricity production and consumption based on the price-quantity bids of electricity producers. If the power delivery of each supplier is not equal to its committed level, changes in electricity supply will be made, either through TSO's own facilities or purchases from other producers (through bilateral contracts or power pools). The cost of these changes is usually covered by penalties to the electricity producers in breach of their contractual obligations. Sometimes these penalties are so high that erode a large percentage of the production income.

By increasing the accuracy of forecasting, these penalties can be avoided, and production costs can be reduced. For example, in the sole island of Ireland, an improvement of 1 % in the accuracy of wind power forecast results in 0.27 % saving of the total generation costs which equates to €4.1 million [5].

Wind power forecasting also provides the possibility of saving costs during the maintenance of wind turbines. Postponing the maintenance to the times when the lowest amount of energy is produced, would greatly increase the profits [3].

Over the last decade, various methods have been developed to forecast wind power [3]. One of the most common prediction methods in this field, are machine learning (ML) algorithms. ML algorithms are mathematical functions that represent the relationship between different aspects of data. These algorithms, which spend a significant amount of time training from data, must be configured before training by setting some variables known as parameters and hyperparameters [6].

Parameters, like the weight at each neuron in a Neural Network (NN), are intrinsic to the model equation and can be determined while the algorithm is being trained. But the hyperparameters, in contrast, are not directly learnt by the learning algorithm. They are specified outside of the training procedure and their role is to control the capacity of the models and increase their flexibility to fit the data. A correct choice of hyper parameter is important to prevent overfitting and improve the generalization of the algorithm.

Hyperparameters have a large impact on the performance of the learning algorithms and their values vary depending on the specific problem domain where the algorithm is used. As a result, they need to be optimized for each dataset [7]. The process of finding the best hyperparameters for a given dataset is called hyperparameter optimization or hyperparameter tuning. Hyperparameter tuning consists of defining the hyperparameter space, a method for sampling candidate hyperparameters, and a metric that is required to be minimized or maximized. As it is not possible to define a formula to find the hyperparameters, different combinations of hyperparameters need to be tried and the model performance evaluated at each stage. But the critical step is to choose how many different hyperparameter combinations are going to be tested. Intuitively, the higher the number of hyperparameter combinations, the greater the chance to get a better performing model. But at the same time, it leads to greater computational cost, because we will end up training a large number of models at the same time. It is also important to determine which hyperparameters have a greater effect on the performance of the ML models [6].

The hyperparameters of simple ML models, such as linear models or tree-based algorithms, can be estimated manually through iterative trial and error. This approach can be very challenging for users who do not have enough professional background and practical experience. To overcome the disadvantages of manual search, automatic search algorithms such as grid search and random search have been proposed.

Grid search does an exhaustive search through training the ML model with all possible combinations of hyperparameters. Then, after evaluating the performance of the model based on a predefined metric, identifies hyperparameters that achieve the best performance [8]. This search method is used extensively in the literature. For example in the developed wind speed prediction model by Zhou et al. [9], the hyperparameters of support vector regression (SVR) were selected by grid search. In another research, Kisvari et al. [10] applied the grid search to tune the hyperparameters of the Gated Recurrent Unit (GRU) and Long Short-Term Memory (LSTM) models. Although this method automatically handles the optimization process, for more complex models such as NNs, it quickly loses its efficiency by increasing the number of hyperparameters and widening the range of their values [8]. Because in these cases the training of the models becomes very costly both in time and required computing facilities, trying all combinations of hyperparameters is not an option.

Random search, on the other hand, tries random combinations of a range of values to increase the efficiency in a high-dimensional space. Bergstra et al. [11] showed that random search, while having the same advantages as grid search, has a much higher efficiency, especially in a high-dimensional space. Nonetheless, in the random selection of combinations, there is the possibility of not considering the best combination, especially in complex models [12]. In both random and grid search methods, all the candidate points are generated upfront and evaluated in parallel, and once all evaluations are done, the best hyperparameters will be selected. Selecting and evaluating candidates without considering the past evaluated hyperparameters leads to the inefficiency of these methods. Because considerable time is spent evaluating inappropriate hyperparameters. As a result, there has always been a high demand for smarter tunning methods that achieve higher accuracy and efficiency by considering the results of the previously assessed hyperparameters.

Sequential model-based optimization (SMBO) which is also known as Bayesian optimization, is an effective algorithm for solving the optimization problem of functions with high-dimensional space [6]. In sequential search, a few hyperparameters are selected and after evaluation of their quality, where to sample next will be decided. SMBO methods have been used in the literature for hyperparameter optimization of ML models. For example, Masum et al. [7] for detecting network intrusion, used Bayesian optimization to find the best hyperparameters for deep neural networks. In the field of wind power forecasting, Zha et al. [13] utilized Tree-structured Parzen Estimator (TPE) algorithm to obtain the best hyperparameters of the temporal convolution network (TCN), but they did not compare the optimization performance of the applied algorithm with other optimization methods. In another study, Hanifi et al. [14] used the TPE search method to optimize

the LSTM model for wind power prediction. Although in the comparison of the accuracy and efficiency of the proposed method with the conventional grid search method, the authors proved the better optimization performance of the TPE, they did not investigate other smart and advanced hyperparameter tuning methods.

In this study, three advanced hyperparameter optimization algorithms including Scikit-opt, Hyperopt, and Optuna are investigated in detail and their performance in the hyperparameter optimization of two widely used deep learning methods for time series-based predictions, the CNN and LSTM models are assessed. As these models can solve various prediction issues with short-term and long-term dependencies, it is useful to study ways by which their performance can be improved. The accuracy and calculation time of CNN and LSTM models are compared on power data from a real offshore wind turbine. To the authors' knowledge, this comparison of advanced hyperparameter optimization techniques has not been attempted in the literature.

The application of the advanced optimization methods in this research provides the possibility to overcome the most important challenges and difficulties of hyperparameter optimization of deep learning models including the high dimensions of their search space and the sensitivity of deep learning models to the selection of hyperparameters.

Regardless of the hyperparameter optimization issue, most artificial neural networks (ANN) rely on randomness in their training process. This means that, despite training with the same data, they can have different results. To endure the efficient performance of the prediction models and achieve consistent results, it is necessary to take this randomness into consideration. In this study, for the first time in wind power forecasting field, the impact of the random initialization on the accuracy of the forecasting models is assessed. For this purpose, different structures of deep learning models, proposed by the most efficient optimization technique, are examined to determine the most robust structure with the minimal sensitivity to the randomness. As the contribution of this study, it is empirically demonstrated that the Optuna optimization algorithm using the TPE search method and Expected Improvement (EI) acquisition function, is the most effective method of tuning the CNN and LSTM, two commonly used deep learning methods.

The key contributions of this research to current knowledge gaps are as follows.

- Going beyond simple grid or random search methods, which evaluate all the pre-generated candidate points in parallel, this paper explores utilizing the SMBO optimization methods, where hyperparameter selection in each stage depends on their quality evaluation in the previous stages.
- Some SMBO methods have been used and compared with grid search and random search methods in previous papers; however, a comprehensive comparison between various advanced hyperparameter optimization methods appears understudied. This comparison enables wind power prediction models to be tuned using the most appropriate SMBO method. This comparison is introduced and applied here.
- The impact of randomness in the training process of deep learning models and its effect on the prediction performance was found to be understudied in the literature. This paper investigates this to determine the most robust structure for both LSTM and CNN models.
- Finally, this study evaluates a greater number of hyperparameters with a wider range than previous studies. Consequently, prediction accuracy was enhanced compared to previous attempts.

2. Methodology

As previously mentioned, in sequential search methods, several hyperparameters are selected and after evaluating their quality, the next sampling location is decided. The goal of sequential search is to make fewer evaluations of the models with various hyperparameters and evaluate only those that have the most promising candidate hyperparameter. The trade-off here is between less ML model training time and the time to estimate where to sample next. Hence, this model is the preferred option when the evaluation procedure (training the model and evaluating its performance) takes much longer than the process of evaluating where to sample next.

Sequential models integrate sample information with previous information about the unknown function to achieve posterior information about the function distribution. This posterior information is then used to determine the location of the optimal performance. Bayesian optimization is one of the effective sequential models that can solve the optimization problem of unknown functions.

2.1. Bayesian optimization

Bayesian optimization is a powerful sequential tool for optimization of functions that do not presume any functional forms [15]. In Bayesian optimization, contrary to the grid or random search, the results of past evaluations are employed for building a probabilistic model that maps hyperparameters to the probability of a score in the objective function. In this way, the optimization process is more efficient as the next set of hyperparameters are selected in an informed manner. This efficiency comes from considering promising hyperparameters in past results which make fewer calls to the objective function. During the optimization, the aim is finding the maximum value of an unknown objective function f:

$$x^* = arg\max_{x\in\chi} f(x) \tag{1}$$

where χ is the search space of hyperparameters, x.

In Bayesian optimization f is treated as a random function and a prior is placed over it. The prior captures the behaviour of the objective function f. Following the collection of the function evaluations, the prior is updated to build the posterior distribution over the objective function. The posterior distribution is then used to construct an acquisition function to determine the next query point.

As can be seen in this optimization process two functions are required, a function that acts as the prior of the optimization functions, and a posterior function which is an acquisition function that determines where to sample next. The prior function can be estimated by Gaussian processes or other more specific algorithms such as TPE or random forests (RF). On the other hand, EI or Upper Confidence Bound (UCB) can be used as the acquisition functions.

Some of these functions and their performance are discussed in the following sections.

2.1.1. Acquisition function

After finding the posterior distribution of the objective function, the next challenge is to know how to search for the next points which derivates the maximum of the function f. In Bayesian optimization, the acquisition function u is used to achieve this goal. Assuming that the high value of the acquisition function corresponds to the large value for the objective function f, maximizing the acquisition function will maximize the objective function f.

The common acquisition functions that are used in Bayesian optimization include the probability of improvement (PI) and EI functions.

Function PI tries to search around the current optimum sample to find points that may exceed the current optimum value. This function can be described as follows:

$$PI(x) = \varphi\left(\frac{\mu(x) - f(x^+)}{\sigma(x)}\right) \tag{2}$$

where φ represents the cumulative distribution function of the standard Gaussian distribution. As evident, the main drawback of the PI acquisition function is that it selects samples only from regions close to the current optimal solution (exploration). Therefore, the possible better

points that are far away from the local optimal points may not be investigated [8].

To solve the problem of falling into the local optimum solution, the EI acquisition function is employed [16]. The EI function while exploring the region of the current optimum value, calculates the expected improvement of the new point. If the improvement value is less than the desired value after running the algorithm, it will be assumed that the current optimal point is the best solution available in that area, and therefore the algorithm searches for the optimal point in other points of the domain (exploitation).

The difference between the function value at the new selected point and the current optimal value is called the degree of improvement I. If the value of the function at the new point is lower than the current optimal value, the improvement function is considered 0:

$$I(x) = max\{0, f_{t+1}(x) - f(x^+)\} \tag{3}$$

Based on the assumption that the distribution of the function value at the new sampling point ($f_{t+1}(x)$) obeys the normal distribution with mean $\mu(x)$ and standard deviation $\sigma^2(x)$, the random variable I obeys the normal distribution too with the mean $\mu(x) - f(x^+)$ and standard deviation $\sigma^2(x)$. Equation (4) shows the probability density of I:

$$f(I) = \frac{1}{\sqrt{2\pi}\sigma(x)} \exp\left(-\frac{(\mu(x) - f(x^+) - I)^2}{2\sigma^2(x)} \right), I \geq 0. \tag{4}$$

Now, the EI can be defined as:

$$E(I) = \int_{\infty}^{\infty} I\, f(I) \mathrm{d}I = \int_{I=0}^{I=\infty} I\, \frac{1}{\sqrt{2\pi}\sigma(x)} \exp\left(-\frac{(\mu(x) - f(x^+) - I)^2}{2\sigma^2(x)} \right) \mathrm{d}I = \sigma(x)[Z\varphi(Z) + \varphi(Z)] \tag{5}$$

where φ is the probability density function of the standard normal distribution and:

$$Z = \frac{\mu(x) - f(x^+)}{\sigma(x)} \tag{6}$$

As can be seen from these equations the EI can be calculated from σ, μ and the current optimal point $f(x^+)$. In this study, the EI function is used as the acquisition function for all hyperparameter optimization techniques because it can address the exploration and exploitation trade-off [7]. This function is proven to have a strong theoretical guarantee [17] and empirical effectiveness [18].

2.1.2. Tree-structured Parzen Estimator (TPE)

In cases of hyperparameter optimization with higher dimensions and a small fitness evaluation budget, an alternative to the Gaussian process (GP) approach is required. In GP, the aim was to approximate f(x) or the probability of score given the hyperparameter, $P(y|x)$, based on the marginals including the probability of each one of the hyperparameters and the probability of the hyperparameters given the score, p(x/y):

$$P(y|x) = \frac{P(x|y) \times P(y)}{P(x)} \tag{7}$$

where y is the score f(x) and x represent the hyperparameters.

But in TPE, instead of approximating the left side of equation (7), the probability of hyperparameters given the score that is obtained when sampling some of the values of the hyperparameters, $P(x|y)$ is attempted to be approximated. This conditional probability ($P(x|y)$) is approximated utilizing two different functions: function $\ell(x)$ for the cases where the performance is smaller than a certain value of performance, and the function $g(x)$ for cases where the performance is bigger than the certain value of performance:

$$P(x|y) = \begin{cases} \ell(x) & \text{if } y < y^* \\ g(x) & \text{if } y \geq y^* \end{cases} \tag{8}$$

these two densities of *l(x)* and *g(x)* will then be used in the EI function, and after some derivations will end up at the following equation:

$$EI_{y^*}(x) \propto \left(\gamma + \frac{g(x)}{\ell(x)}(1 - \gamma) \right)^{-1} \tag{9}$$

This expected improvement can determine where to sample the next for hyperparameters.

2.2. Optimization algorithms

2.2.1. Scikit optimize (GP-EI)

Scikit optimize is an open-source Python package that can perform various forms of Bayesian optimization. It implements several search algorithms including Bayesian optimization with gaussian processes (through the *gp_minimize* function), Bayesian optimization with random forest (through the *forest_minimize* function), and Bayesian optimization with Gradient Boosting Trees (through the *gbrt_minimize* function).

To perform the optimization, the first step is to define the objective function that needs to be minimized. The objective function usually takes the ML model and the hyperparameters and outputs a performance metric.

Scikit optimize comes with a built-in module to create hyperparameter spaces to sample from. The samples can be integers, reals, and categories. In addition, a variety of acquisition functions are available to choose from including the EI, PI and Lower Confidence Bound (LCB).

In this project the Scikit optimize is implemented to use Bayesian optimization with Gaussian process as the surrogate. To this end, the *gp_minimize* package is imported and two prediction models (CNN and LSTM) were defined as the objective function. Then the hyperparameter space is defined as well as the initial number of points at which to evaluate the objective function before starting to guide the Bayesian optimization search. As mentioned earlier in this study the EI is used due to its capability of exploration and exploitation to guide the search in Bayesian optimization.

2.2.2. Optuna (TPE-EI)

As second tuning method in this study, the Optuna optimization package with its define-by-run design is employed. Optuna is a recently developed optimization algorithm [19] that has been successfully used to tune the hyperparameters of LSTM model for wind power prediction [14]. The selection of Optuna as one of the advanced optimiation algorithms in this research is based on its three specific features, define-by-run context, efficient sampling, and ease of setup. The define-by-run context enables building the search space dynamically. In the optimization process, different hyperparameter combinations are considered as input to maximize a function and respond a validation score as an output. The objective function formulates the search space dynamically through interacting with the trial object [20]. Efficient sampling of Optuna allows handling of both types of sampling, relational sampling to benefit the parameters correlations and independent sampling to consider each sample separately [20]. Another advantage of Optuna is its easy setup, which allows it to be used for a variety of tasks, from lightweight experiments performed through interactive interfaces to heavyweight distributed computing [19]. There are various features in Optuna that need to be set up. Sampler as the hyperparameter search algorithm can be selected among different options including the basic ones such as Grid Search, Random Search and CMA-ES. In this study the TPE algorithm is used to consider each hyperparameter independently. The next option is the Pruner that stops testing trials that do not offer promising performances. The default pruner which is used in this study is the Median Pruner, however other pruners such as the Percentile Pruner, Successive Halving, Hyperband and Threshold pruner are also

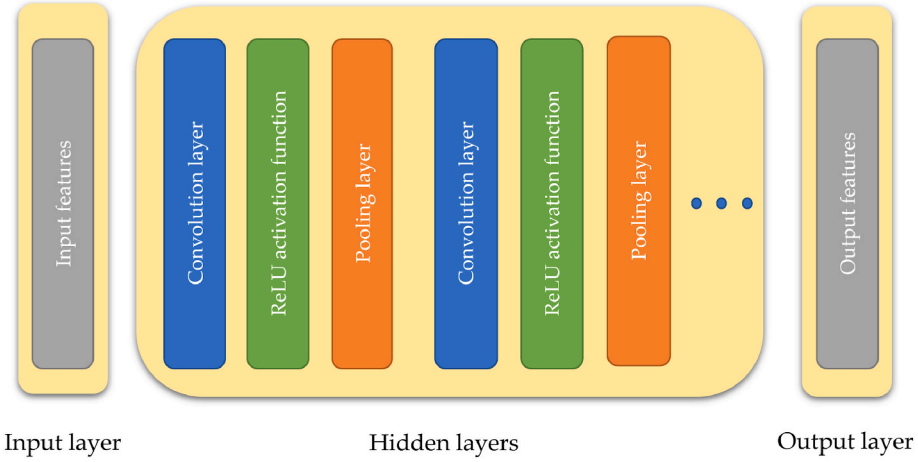


Fig. 2. Basic architecture of a CNN model.

available. It is also necessary to set the direction of the tuning process, which indicates whether we want to minimize or maximize the objective function. As mentioned earlier, the aim of this study is to minimize the RMSE value of the wind power forecasting. As a result, the minimize direction is selected. Further details of the optimization by this package can be found in Ref. [19].

2.2.3. Hyperopt (Annealing-EI)

Hyperopt is a python library that enables implementing Bayesian optimization. Hyperopt has already been used for hyperparameter optimization of deep neural networks and convolutional neural networks [12]. Hyperopt provides three search algorithms: random search (*rand.suggest*), annealing (which is another sequential mode-based optimization with the gaussian process alternative in order to be able to sample nested hyper parameters), and TPE. In this study, the annealing search method is selected to compare its performance with the TPE search method used in the Optuna package and the GP search method used in the Scikit optimize algorithm.

2.3. Deep learning neural networks (CNN and LSTM models)

In this research, to measure the performance of the optimization methods, two widely used deep learning models, CNN and LSTM models are used.

2.3.1. Convolutional neural network (CNN)

CNN is a particular type of NNs that uses a mathematical function called convolution instead of general matrix multiplication. The basic structure of a CNN model is shown in Fig. 2 This deep learning model like any feed-forward neural network (FFNN) consist of three main layers including input layer, hidden layers, and output layer. The hidden layer is where the convolution function is performed based on a dot product of the convolution kernel with the matrix of its input layer. The convolution process can be presented as follows:

$$h_{ij}^{k} = f\left(\left(W^{k} \otimes x\right)_{ij} + b_{k}\right) \tag{10}$$

where f, x and b represent the activation function, the vector of input series and the vector of bias, respectively and W^{k} denotes the connected kernel weights to the kth feature map. In this equation, $\otimes$ symbol indicates the convolution function, and the common activation function f is the Rectified Linear Unit (ReLU). Following the convolution operation on the input matrix of each convolution layer, a feature map is created as the input of the next layer.

The convolution layer in a CNN is cascaded by the pooling layer which is a shape of non-linear down-sampling. The main purpose of the application of the pooling layer is to gradually decrease the size of the data representation to control overfitting.

CNN can effectively extract the hidden non-linear dependencies of time series through automatic creation of filters [21]. As a result it has been used widely for time series based predictions [17–19]. The CNN model in this study consists of two convolutional layers. It also can have up to four fully connected dense layers. The channel number of each convolution layer and the quantity of the dense layers will be suggested by hyperparameter optimization algorithm.

2.3.2. Long Short-Term Memory (LSTM)

Recurrent neural network (RNN) is another type of NNs where connection between nodes can create cycles. As a result, it has a high ability to represent almost all dynamics [22]. However, due to the limitations in learning process, in cases that the time interval between the input signal and the target signal increases, the backpropagation error either vanishes or explodes and its effectiveness decreases [14]. To address this drawback, the LSTM model based on a memory cell as the core is presented. LSTM, while having the advantages of RNNs, provides a suitable solution for the vanishing gradient problem. The description and computation of the LSTM components is explained in detail in Ref. [14]. LSTM has an excellent ability to learn non-linear short-term and long-term dependencies. As a result, it has been used widely for time series-based predictions [20,21,23,24].

LSTM model has several hyperparameters that are required to be optimized. These hyperparameters have a significant impact on the total computational cost and determine the ability of the network to generalize well over the unseen data domains. In this study, hyperparameters such as the number of neurons, the batch size (the size of provided data to network before updating the weights), the epochs number (the number of iterations completed during training the model), the activation function and optimizers were optimized using three different optimization approaches.

2.4. Prediction performance criteria

In this study the best hyperparameter combinations found by different optimization methods were used to create the deep learning wind power forecasting methods. To evaluate the performance of the optimization algorithms, the root mean square error (RMSE) which can be computed as described in equation (11), is used.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\hat{Y}_i - Y_i\right)^2} \tag{11}$$

Table 1
search space for hyperparameters of the CNN and LSTM models.

CNN Model		LSTM Model	
Hyperparameter	Search Space	Hyperparameter	Search Space
Units of first conv. Layer	30, 31, …, 130	Units (neurons) of first LSTM Layer	10, …, 100
Units of second conv. Layer	30, 31, …, 130	Number of dense layers	1, 2, 3, 4
Number of dense layers	1, 2, 3, 4	Units (neurons) in dense layer	10, 11, …, 100
Units (neurons) in dense layer	10, 11, …, 100	Activation function	Sigmoid, tanh, ReLU
Activation function	Sigmoid, tanh, ReLU	Loss function	MAE, MSE
Loss function	MAE, MSE	Optimizer	ADAM, Adadelta
Optimizer	ADAM, Adadelta	Epochs	50, 51, …, 150
Epochs	50, 51, …, 100		

where Y_i represents the recorded offshore wind power at time step t, $\widehat{Y}_i$ is the wind power prediction of deep learning forecasting models (for the identified time step), and N represents the number of data points. The RMSE is the most common evaluation metric used in the field of wind power forecasting [3].

2.5. Wind power dataset

The source SCADA data used in this study is provided the Leven mouth Demonstration Turbine (LDT) in Scotland. The key parameters and configuration of this offshore wind turbine is described in detail in Refs. [14,25]. The dataset covers the first four months of 2019, and it includes more than 500 different observations including wind speed, wind power, nacelle orientation, etc.

During the pre-processing stage, the negative wind power values within the dataset were replaced with zero based on the recommendations issued in Ref. [26]. In addition to diminishing the negative impact of wind turbulence on the correlation between the generated wind power and measured wind speed, the time series resolution averaged 10 min in coordination with the approved average time by the international standard for power performance measurements of electricity-producing wind turbines (IEC 61400-12-1) [27]. The dataset was then divided into two parts, the first 90 % for training and the rest 10 % for testing both CNN and LSTM deep learning models, which were optimized by various hyperparameter optimization techniques.

3. Experimental results and discussions

The hyperparameters and their search spaces for the CNN and LSTM models for all three optimization methods are determined according to Table 1. In this research, a number of hyperparameters, with significant impact on prediction performance in similar studies ([28,29]), as well as a number of other parameters that have not been investigated, are selected for tuning. Additionally, experiments were conducted to modify the ranges of each hyperparameter in order to improve the overall prediction accuracy. For example, in the case of activation function for LSTM model, the Softmax activation function that leads to an increase in prediction error is not considered. Or, a new range of epoch numbers for CNN model has been selected compared to our previous research (), which increases the accuracy of the prediction. Considering the ranges of values shown in Table 1 for the different hyperparameters, there are more than 2000 million hyperparameter combinations for CNN models and more than 38 million combinations for LSTM models.

To carry out all the simulations and experiments, the Python programming language with Scikit-opt, Optuna, Hyperopt packages, and Scikit-learn libraries are employed on a PC with Intel Core™ i7–11850H 2.5 GHz CPU and 16 GB RAM (without GPU processing). In addition, for better investigation of the performance of the various hyper parameter optimization and forecasting models, similar parameters were set for all selected models; for example, the selected time lag (input layer length) was set at 10 in all simulations.

3.1. Optimization by scikit-optimize algorithm

As the first optimization method in this research, the Scikit optimize is implemented. To utilize optimization in this algorithm, except defining the hyperparameter space, it is required to pass the objective function, the initial number of points at which to evaluate the objective function before starting to guide the Bayesian optimization search, the acquisition function, and the number of times that we want to sample the hyper parameter space subsequently. The Gaussian process is selected as the search method and EI as the acquisition function. The RMSE values of the wind power predictions by forecasting models which were built with selected hyperparameters are considered as the objective function.

The Scikit optimize algorithm for tuning the CNN model, launched for 300 tests. The best ten hyperparameter combinations with the least RMSE values are obtained according to Table 2 while the whole optimization process took approximately 455 min. As can be seen in this table, the minimum RMSE value of 533.74 is obtained for trial number 204.

The Scikit optimize optimization method with similar settings is also used to tune the LSTM prediction model. In this case, the entire tuning process for 300 trials, took approximately 1740 min and a minimum RMSE value of 549.07 was obtained for trial number 208. Table 3 shows the best ten hyperparameter combinations for the LSTM model that lead to the lowest RMSE values.

As can be seen from Tables 2 and 3, the values of some of the hyperparameters such as the activation function, loss function, and optimizer are equal for all proposed hyperparameter combinations. This shows that the optimization algorithm has quickly reached a level of certainty regarding these parameters. On the other hand, for other hyperparameters such as the units of convolution layers, small changes

Table 2
Best hyperparameter combinations of CNN model found by Scikit-Optimize.

Rate	Trial number	Units of first conv. Layer	Units of second conv. Layer	No. of dense layers	No. of dense units	Activation function	Loss function	optimizer	Epochs	RMSE
1	204	87	89	4	100	ReLU	MAE	ADAM	104	553.74
2	219	128	12	4	100	ReLU	MAE	ADAM	101	560.29
3	210	128	12	4	100	ReLU	MAE	ADAM	104	562.66
4	257	128	12	1	100	ReLU	MAE	ADAM	132	566.65
5	217	128	12	4	100	ReLU	MAE	ADAM	101	569.20
6	88	12	12	4	100	ReLU	MAE	ADAM	87	569.57
7	230	128	31	2	100	ReLU	MAE	ADAM	98	572.18
8	138	12	91	4	100	ReLU	MAE	ADAM	89	572.47
9	162	12	104	4	100	ReLU	MAE	ADAM	88	572.52
10	78	128	128	4	100	ReLU	MAE	ADAM	90	572.57

Table 3
Best hyperparameter combinations of LSTM model found by Scikit-Optimize.

Rate	Trial number	Units of first LSTM layer	dense layers	Units of dense layer	Activation function	Loss function	optimizer	Epochs	RMSE
1	146	61	1	68	ReLU	MSE	ADAM	150	549.07
2	177	27	1	10	ReLU	MSE	ADAM	150	549.37
3	260	42	1	59	ReLU	MSE	ADAM	150	549.74
4	26	12	4	100	ReLU	MAE	ADAM	149	549.94
5	298	100	4	10	ReLU	MSE	ADAM	74	551.75
6	85	54	1	29	ReLU	MSE	ADAM	150	551.99
7	39	100	1	100	ReLU	MSE	ADAM	150	552.31
8	279	100	4	100	ReLU	MSE	ADAM	58	552.48
9	203	50	1	67	ReLU	MSE	ADAM	150	553.02
10	219	75	1	42	ReLU	MAE	ADAM	50	554.33

Table 4
Best hyperparameter combinations of CNN model found by Optuna.

Rate	Trial number	Units of first conv. Layer	Units of second conv. Layer	No. of dense layers	No. of dense units	Activation function	Loss function	optimizer	Epochs	RMSE
1	269	108	80	3	45	ReLU	MAE	ADAM	86	553.66
2	203	113	82	3	44	ReLU	MAE	ADAM	84	560.9
3	97	72	32	3	47	ReLU	MSE	ADAM	95	561.73
4	175	119	66	3	43	ReLU	MAE	ADAM	78	567.36
5	228	119	78	3	39	ReLU	MAE	ADAM	84	570.45
6	205	114	81	3	44	ReLU	MAE	ADAM	83	572.98
7	227	119	77	3	40	ReLU	MAE	ADAM	84	573.19
8	156	71	69	3	43	ReLU	MAE	ADAM	77	573.36
9	77	83	62	3	46	ReLU	MAE	ADAM	94	574.49
10	160	81	66	3	38	ReLU	MAE	ADAM	81	575.24

Table 5
Best hyperparameter combinations of LSTM model found by Optuna.

Rate	Trial number	Units of LSTM layer	No. of dense layers	No. of dense units	Activation function	Loss function	optimizer	Epochs	RMSE
1	91	45	3	60	ReLU	MSE	ADAM	128	538.14
2	123	43	2	61	ReLU	MSE	ADAM	124	541.04
3	81	34	2	73	ReLU	MSE	ADAM	125	542.27
4	272	46	2	59	ReLU	MSE	ADAM	120	542.34
5	255	42	2	59	ReLU	MSE	ADAM	108	542.39
6	69	37	1	54	ReLU	MSE	ADAM	104	543.49
7	82	37	2	73	ReLU	MSE	ADAM	127	545.09
8	248	50	2	63	ReLU	MSE	ADAM	105	545.44
9	241	42	2	61	ReLU	MSE	ADAM	103	546.29
10	258	52	2	63	ReLU	MSE	ADAM	123	547.53

can be seen until reaching the optimum value.

3.2. Optimization by Optuna algorithm

As the second tuning method in this study, the Optuna optimization package with its define-by-run design is employed. There are some features in the Optuna optimization package that need to be set up. Sampler, as the hyperparameter search algorithm, can be selected among different methods including the basic ones such as Grid Search, Random Search and CMA-ES. In this study the default search algorithm, TPE, is used. The next option is the Pruner that stops testing trials that are not offering promising performances. The default pruner which is used in this study is the Median Pruner, however other pruners including Percentile Pruner, Successive Halving, Hyperband and Threshold pruner are also available. The median pruner performs when the trial's best intermediate result is poorer than median of the intermediate results of former trials. It is also necessary to set the direction of the tuning process, which indicates whether we want to minimize or maximize the objective function. As mentioned in the previous section, the aim of this study is to minimize the RMSE value of the wind power forecasting. More details of the optimization process of this package can be found in Ref. [19].

The same hyperparameter spaces as what was considered for scikit optimize, are used again. For the CNN model, the minimum RMSE value of 553.66 is obtained for trial number 269 after about 492 min and regarding the LSTM model, the minimum RMSE value of 538.14 is obtained for trial number 91 after about 456 min. Tables 4 and 5 show the best ten hyperparameter combinations for CNN and LSTM models, found by the Optuna algorithm, respectively.

As can be seen from Tables 4 and 5, the values of two hyperparameters, namely the Activation function and optimizer, are equal for all the ten best combinations of CNN and LSTM models. While the best loss function selected in the ten most accurate hyperparameter combinations of the LSTM model is the mean square error (MSE), the results of experiments show that the value of this hyperparameter in the CNN model can be both the MSE and the mean absolute error (MAE).

3.3. Hyperopt

Hyperopt is the third advanced hyperparameter optimization algorithm utilized in this research. In the Hyperopt algorithm, first the configuration space is defined and then the *fmin* driver is used to determine the direction of the optimization. Hyperopt provides three search algorithms including the random search (*rand.suggest*), annealing, and the TPE. Annealing is another GP alternative SMBO model with the capability of sampling the nested hyper parameters. In this study the

Table 6
Best hyperparameter combinations of CNN model found by Hyperopt.

Rate	Trial number	Units of first conv. Layer	Units of second conv. Layer	No. of dense layers	No. of dense units	Activation function	Loss function	optimizer	Epochs	RMSE
1	203	105	33	2	21	ReLU	MSE	ADAM	80	556.57
2	237	105	33	2	58	ReLU	MAE	ADAM	80	567.44
3	270	105	33	2	21	ReLU	MSE	ADAM	80	567.88
4	204	105	33	2	58	ReLU	MAE	ADAM	80	569.32
5	200	105	33	2	21	ReLU	MAE	ADAM	80	571.45
6	257	105	33	2	21	ReLU	MAE	ADAM	80	571.72
7	172	105	33	2	21	ReLU	MAE	ADAM	80	571.85
8	285	105	34	2	14	ReLU	MAE	ADAM	80	572.39
9	220	105	33	2	21	ReLU	MAE	ADAM	80	573.22
10	66	105	33	1	21	ReLU	MAE	ADAM	80	574.06

Table 7
Best hyperparameter combinations of LSTM model found by Hyperopt.

Rate	Trial number	Units of LSTM layer	No. of dense layers	No. of dense units	Activation function	Loss function	optimizer	Epochs	RMSE
1	236	52	2	60	ReLU	MSE	ADAM	99	532.59
2	96	52	2	60	ReLU	MSE	ADAM	99	541.23
3	195	52	2	60	ReLU	MSE	ADAM	99	541.76
4	237	52	2	97	ReLU	MSE	ADAM	99	541.83
5	35	52	2	60	ReLU	MSE	ADAM	99	542.32
6	244	52	2	60	ReLU	MSE	ADAM	99	543.59
7	266	52	2	60	ReLU	MSE	ADAM	99	544.18
8	14	52	2	60	ReLU	MSE	ADAM	66	546.58
9	154	66	2	60	ReLU	MSE	ADAM	99	547.82
10	213	52	1	60	ReLU	MSE	ADAM	99	547.92

Table 8
The RMSE and processing time of the prediction models tuned in different method.

Optimization methods		RMSE (kW)		Calculation time (minutes)	
Algorithm	Search method	CNN	LSTM	CNN	LSTM
Scikit-Opt	GP	553.74	539.64	455	600
Optuna	TPE	553.66	538.14	492	456
Hyperopt	Annealing	556.57	532.59	368	892

annealing search method is selected to compare its performance with the TPE search method in the Optuna algorithm and the GP search method in Scikit optimize.

The best ten hyperparameter combinations of the CNN and LSTM models found by Hyperopt are shown in Tables 6 and 7. For the CNN model, the minimum RMSE value of 556.57 was obtained for trial number 203 after approximately 368 min. Regarding the LSTM model, the minimum RMSE value of 532.59 was obtained for trial number 236 after approximately 892 min.

3.4. Comparison of the hyperparameter optimization algorithms

By comparing the prediction accuracy of prediction models whose structures (hyperparameters) are proposed by optimization methods, the tuning methods can be compared. The processing speed is another factor in determining the performance of these optimization methods. Table 8 shows the performance of the prediction models tuned by the three optimization techniques, both in processing time and accuracy.

Considering the calculated RMSE values in Table 8 and Fig. 3(a), the first point that can be recognized is that the LSTM model is a better choice than the CNN for short term wind power prediction. This is believed to be due to the capability of the LSTM model in learning both long-term and short-term dependencies. On the other hand, while the prediction accuracy of different structures of the CNN model, proposed by different optimization methods are very similar, the LSTM model optimized by the Hyperopt algorithm based on the annealing search method results in the highest accuracy. In terms of prediction accuracy, the Optuna optimization method ranks second after the Hyperopt algorithm.

Based on the comparison of the required time for calculations/simulations in each optimization in Table 8 and Fig. 3(b), it can be

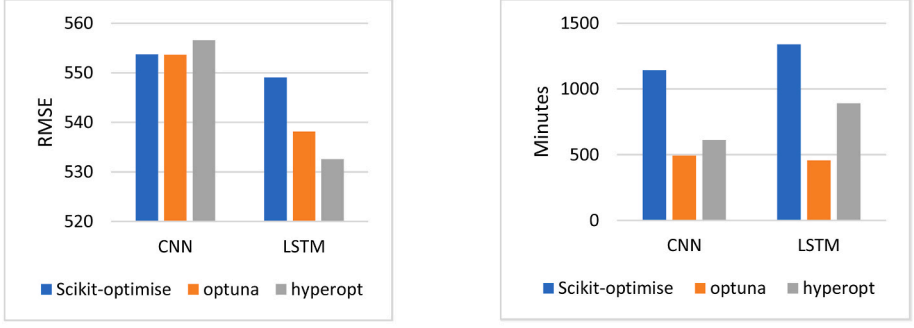


Fig. 3. RMSE (a) and calculation time (b) of the CNN and LSTM tuned by various optimization methods.

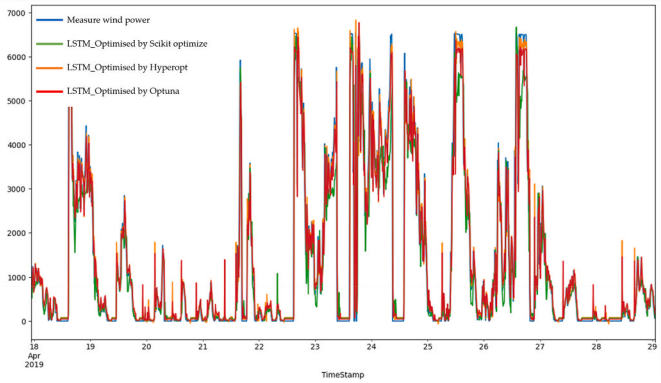


Fig. 4. Wind power predictions of LSTM models optimized by three different optimization algorithms.

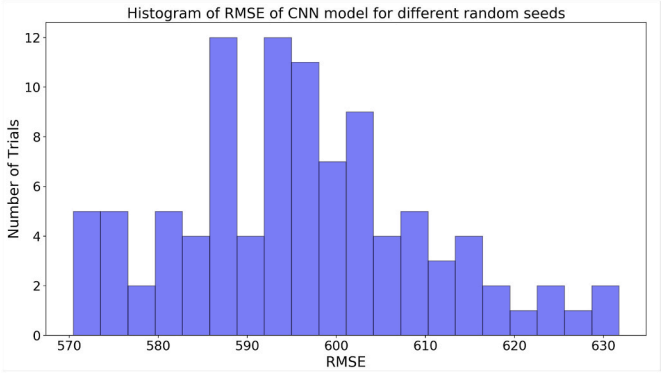


Fig. 5. Histogram of RMSE values of CNN model with different seeds.

Table 9
The statistical description of RMSE prediction results with different seeds.

RMSEs of the CNN prediction model	
Counts of trials	100
Average of RMSEs	595.96
Standard deviation	13.78
Minimum RMSE	570.45
Maximum RMSE	631.73

concluded that the Optuna optimization algorithm using the TPE search method and EI acquisition function, has the best efficiency for both CNN and LSTM models. Fig. 4 shows the prediction results of three individual LSTM prediction models tuned by the optimization methods. As can be in this figure, when the wind power generation encounters abrupt changes, the LSTM model optimized by Hyperopt and Optuna

Fig. 6. RMSE values of different hyperparameter combinations of CNN model with different seeds.

algorithms have better prediction performance than the LSTM model optimized by Scikit optimize.

3.5. Randomness impact on the optimization performance

As mentioned earlier, most artificial neural networks (ANN) have randomness in their training process. Randomness can have different reasons, the most important of which is the random initialization of weights and biases in NNs. Different initialization of weights causes changes in the training results that make the networks unstable and unreliable. As a result, to improve the performance of the prediction models and achieve consistent results, it is necessary to predict this randomness. In ML, for prediction of randomness the seed concept is used. Seed allows for the prediction of the randomness and makes the results reproducible [29]. Defining a specific seed value before training ensures consistent results of ANNs.

In this study, to investigate the impact of random initialization on the accuracy of deep learning models, an experiment is carried out consisting of one hundred trials of CNN model predictions with different seeds but constant hyperparameters. As can be seen in Fig. 5 and Table 9, the RMSE values of the predictions vary from 570.4 to 631.7 with an average value of 595.9 and standard deviation of 13.78. It is evident that the changes in random initialization, while all other variables are constant, can have a significant impact on the prediction accuracy of the forecasting models.

This means that without considering a wider range of random seeds it is difficult to trust the performance of a specific deep learning model. In other words, various initializations are required to see the robustness of the prediction model. In this project to investigate the impact of the randomness feature of NNs on the accuracy of the deep learning models built by the best hyperparameter combinations, 10 different random seeds [123, 951, 375, 435, 599, 54, 602, 325, 691, 36] are used for the best found hyperparameter combinations of the CNN and LSTM deep leaning models to find the best structure which is less sensitive to the

Table 10
RMSE values of different hyperparameter combinations of CNN model with different seeds.

Seeds		H.C 1	H.C 2	H.C 3	H.C 4	H.C 5	H.C 6	H.C 7	H.C 8	H.C 9	H.C 10
R.S 1	123	582.05	592.06	621.04	596.58	587.33	594.24	597.20	623.10	584.03	605.28
R.S 2	951	581.68	598.74	590.73	596.52	581.34	612.38	612.04	582.82	581.25	599.65
R.S 3	375	590.17	619.76	594.03	617.20	579.50	582.82	607.01	566.11	588.17	575.05
R.S 4	435	569.66	596.19	595.14	614.82	586.07	636.65	610.35	585.86	602.05	625.39
R.S 5	599	597.98	592.18	623.40	590.01	593.39	573.66	596.64	574.51	572.42	620.09
R.S 6	54	597.95	564.98	682.69	616.43	584.56	582.11	592.07	575.23	588.85	596.86
R.S 7	602	596.23	613.62	610.71	601.07	607.69	595.51	606.02	591.19	583.16	602.35
R.S 8	325	587.22	594.58	617.99	621.27	629.48	584.02	604.10	596.17	587.38	584.62
R.S 9	691	582.61	591.89	636.24	578.85	592.06	595.31	617.61	587.37	581.86	590.43
R.S 10	36	598.92	592.02	609.06	591.16	592.17	602.31	583.57	604.86	595.76	593.70
Averages		**588.45**	**595.60**	**618.10**	**602.39**	**593.36**	**595.90**	**602.66**	**588.72**	**586.49**	**599.34**
STD		**9.60**	**14.56**	**26.94**	**14.26**	**14.97**	**18.18**	**10.25**	**16.48**	**8.17**	**15.19**

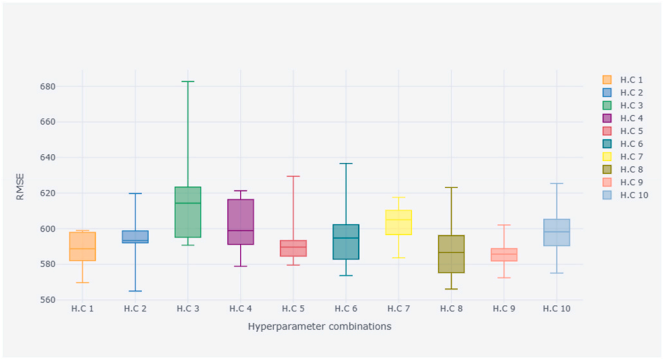


Fig. 7. Average and standard deviations of RMSE values for different hyperparameter combination.

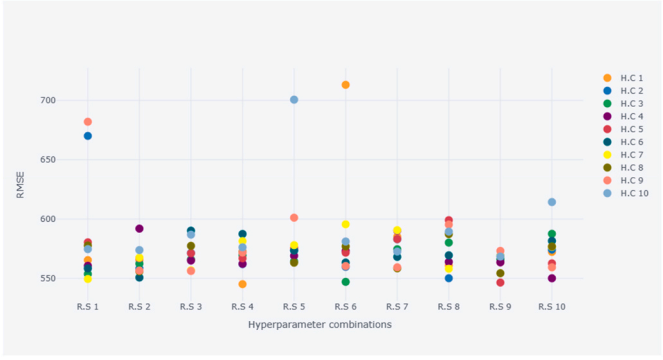


Fig. 8. RMSE values of different hyperparameter combinations of CNN model with different seeds.

randomness.

Table 10 shows the RMSE values related to the predictions made by the CNN models based on the 10 best hyperparameter combinations and 10 different random seeds.

For a better analysis of the test results presented in Table 10, the RMSE values are plotted in Fig. 6. In addition, the distribution of RMSE values for each hyperparameter combination are shown in Fig. 7.

As can be recognized from Table 10 and Figs. 6 and 7, considering all the set seed values, combination number 9 has the best performance, as it has the least average and standard deviation values of RMSEs. In another word, a CNN model built based on the hyperparameter of combination number 9 will be less sensitive to the randomness feature of NNs. A similar experiment/simulation was carried out for the LSTM prediction model. In a similar manner, Table 11 shows the RMSE values related to the predictions made by the LSTM models based on the 11 best hyperparameter combinations and 10 different random seeds.

Fig. 8 shows the RMSE values for different hyperparameter combinations and Fig. 9 demonstrates distribution of RMSE values for each hyperparameter combination.

As noticed from Table 11 and Figs. 8 and 9, considering all the set seed values, combination number 3 has the best performance, as it has the least average and standard deviation values of RMSEs. In other words, an LSTM model based on the hyperparameter of combination number 3 will be less sensitive to the randomness feature of NNs. To assess the robustness of the LSTM model based on the third hyperparameter combinations, an experiment is caried out in which two LSTM models base on the hyperparameter combination sets 1 and 3 were utilized for wind power prediction. In this experiment, no seeds value is set to see the performance of the proposed structures. As can be seen from Fig. 10, the forecasting model, which was built with the third hyperparameter combination, is more accurate. Better forecasting performance is clearly visible especially in higher wind power values, such as for example between hours 15:00 and 18:00 on 26th of April (Fig. 11) and during times when the wind power production experiences sudden

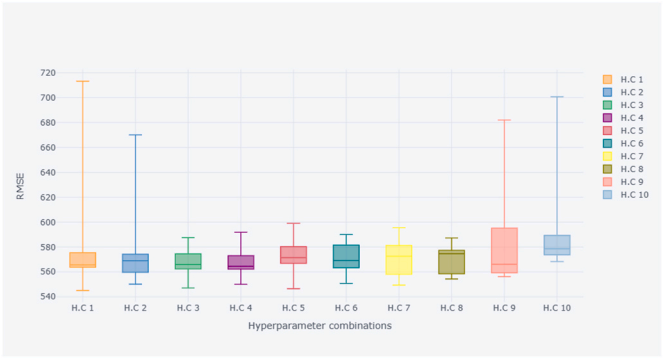


Fig. 9. Average and standard deviations of RMSE values for different hyperparameter combination.

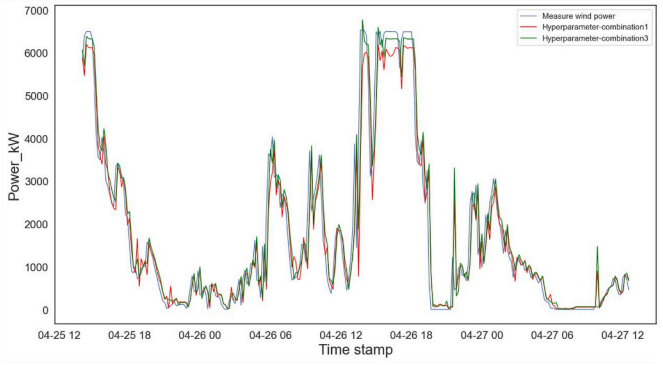


Fig. 10. wind power prediction of LSTM model built by two different hyperparameter combinations.

Table 11
RMSE values of different hyperparameter combinations of LSTM model with different seeds.

Seeds		H.C 1	H.C 2	H.C 3	H.C 4	H.C 5	H.C 6	H.C 7	H.C 8	H.C 9	H.C 10
R.S 1	123	565.41	670.06	553.7	560.58	580.36	558.4	549.45	577.62	681.99	574.64
R.S 2	951	565.62	558.31	562.39	591.9	567.04	550.72	567.53	555.90	556.53	573.81
R.S 3	375	565.64	571.43	565.93	565.04	571.15	590.16	556.83	577.39	556.23	586.87
R.S 4	435	545.13	562.01	569.52	562.27	566.86	587.41	581.36	572.37	571.77	576.19
R.S 5	599	575.46	573.69	564.41	569.04	575.63	573.44	578.02	563.22	601.12	700.64
R.S 6	54	713.16	559.72	547.07	573.1	571.66	563.3	595.56	576.98	560.43	581.04
R.S 7	602	585.11	582.99	574.58	590.25	583.1	568.11	590.65	558.49	559.33	572.58
R.S 8	325	560.84	550.14	580.14	563.89	599.04	569.42	558.06	587.30	595.30	589.38
R.S 9	691	563.77	566.52	565.97	563.51	546.47	568.94	568.75	554.34	573.16	568.38
R.S 10	36	572.29	574.24	587.55	550.10	562.65	581.54	576.65	576.86	559.29	614.30
Averages		**581.24**	**581.24**	**567.13**	**568.97**	**572.4**	**571.14**	**572.29**	**570.05**	**581.52**	**593.78**
STD		**47.49**	**47.49**	**11.85**	**13.07**	**13.85**	**12.46**	**14.95**	**11.23**	**38.77**	**39.78**

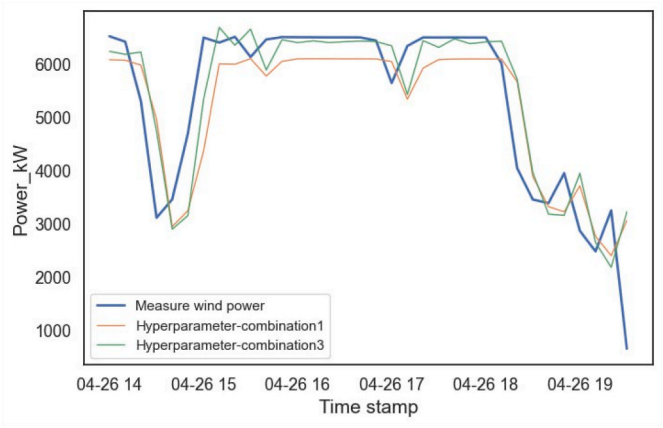


Fig. 11. wind power prediction of LSTM model built by two different hyperparameter combinations between hours 14:00 and 19:00 of 26th of April.

changes.

4. Conclusions

Since higher penetration of wind power into the power grid can be achieved by increasing wind power forecasting accuracy, the focus of this research has been on fine-tuning the hyperparameters of machine learning models to increase the accuracy and efficiency of forecasting. As opposed to grid search or random search which are time-consuming and unreliable, three advanced hyperparameter optimization techniques, Scikit-opt, Hyperopt, and Optuna, are used to tune CNN and LSTM prediction models, two widely used and strong deep learning models. The results showed that the Optuna optimization technique using Tree-structured Parzen estimator (TPE) search algorithm and Expected Improvement (EI) acquisition function, has the highest efficiency for both CNN and LSTM models. Also, regarding the improvement of the prediction accuracy, it has been demonstrated that while for the CNN model, all the optimization methods perform almost the same, the LSTM model optimized by the Hyperopt algorithm based on the annealing search method achieves the highest accuracy. In addition, in this research, the sensitivity of different structures of CNN and LSTM models to seed changes was investigated and the most resistant structure against randomness was selected for both models. The proposed models not only do not require initial settings of random seeds, but also provide the highest level of accuracy, efficiency, and robustness for the utilized offshore wind power dataset. This level of performance for short term wind power forecast can be used for regulation actions, real-time grid operations, market clearing and wind turbine control systems.

CRediT authorship contribution statement

Shahram Hanifi: Conceptualization, Methodology, Software, Investigation, Validation, Formal analysis, Writing – original draft, Visualization. **Andrea Cammarono:** Writing – review & editing, Methodology, Supervision. **Hossein Zare-Behtash:** Writing – review & editing, Methodology, Supervision.

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.

References

[1] U. Arinze, M. Eng, G. Mahmut, and M. Eng, "Challenges of Wind Power Integration to the Power Grid," no. December, pp. 0–8, 2020.
[2] DNV, "ENERGY TRANSITION OUTLOOK UK 2022 A National Forecast to 2050," 2022.
[3] S. Hanifi, X. Liu, Z. Lin, S. Lotfian, A critical review of wind power forecasting methods-past, present and Future, Energies 13 (15) (2020) 1–24, https://doi.org/10.3390/en13153764.
[4] R. Rogus, R. Castro, Comparative Analysis of Wind Energy Generation Forecasts in Poland and Portugal and Their Influence on the Electricity Exchange Prices, no. i, 2020.
[5] E.V.M. Garrigle, The Value of Accuracy in Wind Energy Forecasts, 2013, https://doi.org/10.1109/EEEIC.2013.6549572 no. May.
[6] J. Bergstra, B. Komer, C. Eliasmith, D. Yamins, D.D. Cox, Hyperopt: a Python library for model selection and hyperparameter optimization, Comput. Sci. Discov. 8 (1) (2015), https://doi.org/10.1088/1749-4699/8/1/014008.
[7] M. Masum, et al., Bayesian hyperparameter optimization for deep neural network-based network intrusion detection, in: Proceedings - 2021 IEEE International Conference on Big Data, Big Data 2021, 2021, pp. 5413–5419, https://doi.org/10.1109/BigData52589.2021.9671576, no. Ml.
[8] J. Wu, X.Y. Chen, H. Zhang, L.D. Xiong, H. Lei, S.H. Deng, Hyperparameter optimization for machine learning models based on Bayesian optimization, Journal of Electronic Science and Technology 17 (1) (2019) 26–40, https://doi.org/10.11989/JEST.1674-862X.80904120.
[9] J. Zhou, H. Liu, Y. Xu, W. Jiang, A hybrid framework for short term multi-step wind speed forecasting based on variational model decomposition and convolutional neural network, Energies 11 (9) (2018), https://doi.org/10.3390/en11092292.
[10] A. Kisvari, Z. Lin, X. Liu, Wind power forecasting – a data-driven method along with gated recurrent neural network, Renew. Energy 163 (2021) 1895–1909, https://doi.org/10.1016/j.renene.2020.10.119.
[11] J. Bergstra, Y. Bengio, Random search for hyper-parameter optimization, J. Mach. Learn. Res. 13 (2012) 281–305.
[12] J. Bergstra, R. Bardenet, Y. Bengio, B. Kégl, Algorithms for hyper-parameter optimization, in: Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011, NIPS 2011, 2011, pp. 1–9.
[13] W. Zha, J. Liu, Y. Li, Y. Liang, Ultra-short-term power forecast method for the wind farm based on feature selection and temporal convolution network, ISA Trans. (2022), https://doi.org/10.1016/j.isatra.2022.01.024.
[14] S. Hanifi, S. Lotfian, H. Zare-behtash, A. Cammarano, Offshore wind power forecasting—a new hyperparameter optimisation algorithm for deep learning models, Energies 15 (19) (2022) 6919.
[15] S. Putatunda, K. Rama, A Comparative Analysis of Hyperopt as against Other Approaches for Hyper-Parameter Optimization of XGBoost, ACM International Conference Proceeding Series, 2018, pp. 6–10, https://doi.org/10.1145/3297067.3297080.
[16] J. Mockus, V. Tiesis, A. Zilinskas, The application of Bayesian methods for seeking the extremum, Towards Global Optimization 2 (December) (1978).
[17] A.D. Bull, Convergence rates of efficient global optimization algorithms, J. Mach. Learn. Res. 12 (12) (2011) 2879–2904.
[18] J. Snoek, H. Larochelle, R.P. Adams, Practical Bayesian optimization of machine learning algorithms, Adv. Neural Inf. Process. Syst. 25 (2012) 1–9.
[19] T. Akiba, S. Sano, T. Yanase, T. Ohta, M. Koyama, Optuna: a next-generation hyperparameter optimization framework, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2019, pp. 2623–2631, https://doi.org/10.1145/3292500.3330701.
[20] P. Srinivas, R. Katarya, hyOPTXg: OPTUNA hyper-parameter optimization framework for predicting cardiovascular disease using XGBoost, Biomed. Signal Process Control 73 (December 2021) (2022), 103456, https://doi.org/10.1016/j.bspc.2021.103456.
[21] S. Oehmcke, O. Zielinski, O. Kramer, Input quality aware convolutional LSTM networks for virtual marine sensors, Neurocomputing 275 (2018) 2603–2615, https://doi.org/10.1016/j.neucom.2017.11.027.
[22] J. Zhang, J. Yan, D. Infield, Y. Liu, F. sang Lien, Short-term forecasting and uncertainty analysis of wind turbine power based on long short-term memory network and Gaussian mixture model, Appl. Energy 241 (March) (2019) 229–244, https://doi.org/10.1016/j.apenergy.2019.03.044.
[23] Y.Y. Hong, C.L.P.P. Rioflorido, A hybrid deep learning-based neural network for 24-h ahead wind power forecasting, Appl. Energy 250 (April) (2019) 530–539, https://doi.org/10.1016/j.apenergy.2019.05.044.
[24] S. Hanifi, H. Zare-Behtash, A. Cammarano, S. Lotfian, Offshore wind power forecasting based on WPD and optimised deep learning methods, Renew Energy 218 (2020), https://doi.org/10.1016/j.renene.2023.119241.
[25] N. Dethlefs, Deep Learning with Knowledge Transfer for Explainable Anomaly Prediction in Wind Turbines, 2020, pp. 1693–1710, https://doi.org/10.1002/we.2510, no. February.
[26] X. Shen, X. Fu, C. Zhou, A combined algorithm for cleaning abnormal data of wind turbine power curve based on change point grouping algorithm and quartile algorithm, IEEE Trans. Sustain. Energy 10 (1) (2019) 46–54, https://doi.org/10.1109/TSTE.2018.2822682.

[27] M. Jafarian, A.M. Ranjbar, Fuzzy modeling techniques and artificial neural networks to estimate annual energy output of a wind turbine, Renew. Energy 35 (9) (2010) 2008–2014, https://doi.org/10.1016/j.renene.2010.02.001.
[28] W. Zhang, Z. Lin, X. Liu, Short-term offshore wind power forecasting - a hybrid model based on discrete wavelet transform (DWT), seasonal autoregressive integrated moving average (SARIMA), and deep-learning-based long short-term memory (LSTM), Renew. Energy 185 (2022) 611–628, https://doi.org/10.1016/j.renene.2021.12.100.
[29] H. Peng, X. Bai, Artificial neural network-based machine learning approach to improve orbit prediction accuracy, J Spacecr Rockets 55 (5) (2018) 1248–1260, https://doi.org/10.2514/1.A34171.