Journal of Big Data

RESEARCH Open Access

Machine learning and deep learning models based grid search cross validation for short-term solar irradiance forecasting

Doaa El-Shahat[1], Ahmed Tolba[1], Mohamed Abouhawwash[2*] and Mohamed Abdel-Basset[1]

*Correspondence:
saleh1284@mans.edu.eg

[1] Department of Computer Science, Faculty of Computers and Informatics, Zagazig University, Zagazig 44519, Egypt
[2] Department of Mathematics, Faculty of Science, Mansoura University, Mansoura 35516, Egypt

Abstract

In late 2023, the United Nations conference on climate change (COP28), which was held in Dubai, encouraged a quick move from fossil fuels to renewable energy. Solar energy is one of the most promising forms of energy that is both sustainable and renewable. Generally, photovoltaic systems transform solar irradiance into electricity. Unfortunately, instability and intermittency in solar radiation can lead to interruptions in electricity production. The accurate forecasting of solar irradiance guarantees sustainable power production even when solar irradiance is not present. Batteries can store solar energy to be used during periods of solar absence. Additionally, deterministic models take into account the specification of technical PV systems and may be not accurate for low solar irradiance. This paper presents a comparative study for the most common Deep Learning (DL) and Machine Learning (ML) algorithms employed for short-term solar irradiance forecasting. The dataset was gathered in Islamabad during a five-year period, from 2015 to 2019, at hourly intervals with accurate meteorological sensors. Furthermore, the Grid Search Cross Validation (GSCV) with five folds is introduced to ML and DL models for optimizing the hyperparameters of these models. Several performance metrics are used to assess the algorithms, such as the *Adjusted R^2 score*, *Normalized Root Mean Square Error* (NRMSE), *Mean Absolute Deviation* (MAD), *Mean Absolute Error* (MAE) and *Mean Square Error* (MSE). The statistical analysis shows that CNN-LSTM outperforms its counterparts of nine well-known DL models with *Adjusted R^2 score* value of 0.984. For ML algorithms, gradient boosting regression is an effective forecasting method with *Adjusted R^2 score* value of 0.962, beating its rivals of six ML models. Furthermore, SHAP and LIME are examples of explainable Artificial Intelligence (XAI) utilized for understanding the reasons behind the obtained results.

Keywords: Solar radiation, Deep learning, Machine learning, Hybrid models, XAI

Introduction

Climate change (CC) [1, 2] has emerged as a global issue of paramount importance due to its devastating impact on the existence of all forms of life on Earth. It can pose significant threats to biodiversity loss as well as human health and food production [3] owing to increased heat and humidity [4]. Hence, Intergovernmental Panel on

CC (IPCC) has called for urgent decisions to mitigate the consequences of CC [5]. Fossil fuels encompass hydrocarbons and their derivatives, containing oil, coal and gas. Fossil fuels are one of the key factors of CC [6]. Until now, fossil fuels have been a primary source of non-renewable energy. Unfortunately, the combustion of fossil fuels results in greenhouse gases and carbon dioxide emissions into the atmosphere. When the sunlight penetrates the Earth's atmosphere, these emissions trap the heat and keep it from escaping back to space, which leads to global warming in the long term. Global warning can seriously destroy life on our planet.

Due to the aforementioned detrimental impacts of fossil fuels, about 197 countries at the 28th Conference of Parties (COP28) agreed to gradually shift from traditional fossil fuels to renewable energy sources [7]. Also, there has been a concerted effort to harness the power of renewable energy technologies, such as solar [8], wind [9], hydro [10], and geothermal energy [11], which offer cleaner and more sustainable options for energy production. Moreover, the field of renewable energies has received increasing political and research attention to overcome CC and the energy crisis. Solar energy is regarded as one of the most ubiquitous renewable energy sources, as the sun is abundant throughout the year. Solar energy has many uses [12], such as water heating [13], food drying [14], heating buildings [15], irrigation [16], water distillation [17] and waste recycling [18].

With the help of Photovoltaic (PV) systems [19, 20], the solar irradiance is transformed into electricity. Solar energy is considered a source of clean energy as it does not produce greenhouse gas emissions or other air pollutants during the electricity generation. The irregular presence of solar irradiance and the fluctuating environmental weather conditions (humidity, cloud cover, temperature, wind speed, and wind direction) affect the operational process, management, and stability of PV grids, which in turn affect the amount of electricity production. Therefore, predicting solar irradiance in the future based on historical observations to maintain the availability of solar energy is commonly known as Solar Irradiance Forecasting (SIF).

SIF is considered one of the most challenging issues. In businesses and industries, major decisions, including production, purchasing, and marketing can depend on forecasting. When the solar irradiance is not present, it is accumulated in batteries for future utilization. Hence, the more accurate prediction of solar irradiance helps to provide permanent electricity by using the solar energy stored in batteries during the intervals of solar irradiance absence. The forecast horizon [21] is the time period into the future over which the forecast is made, while the forecast resolution represents the scale or size of the frame at the forecast horizon. There are numerous categories of forecasting horizons [22, 23], as shown in Fig. 1. The figure presents some useful applications that are appropriate for each forecast horizon.

The domain of solar irradiance forecasting has garnered growing attention by many researchers over the past few decades. The literature outlines several methodologies that can be divided into three categories: deterministic [24], ML or statistical [25, 26], deep learning [27, 28], and hybrid methods [29, 30]. However, deterministic techniques have certain drawbacks when predicting short-term forecasts [31]. Unlike these deterministic methods, ML and DL algorithms are considered more accurate to predict solar irradiance.

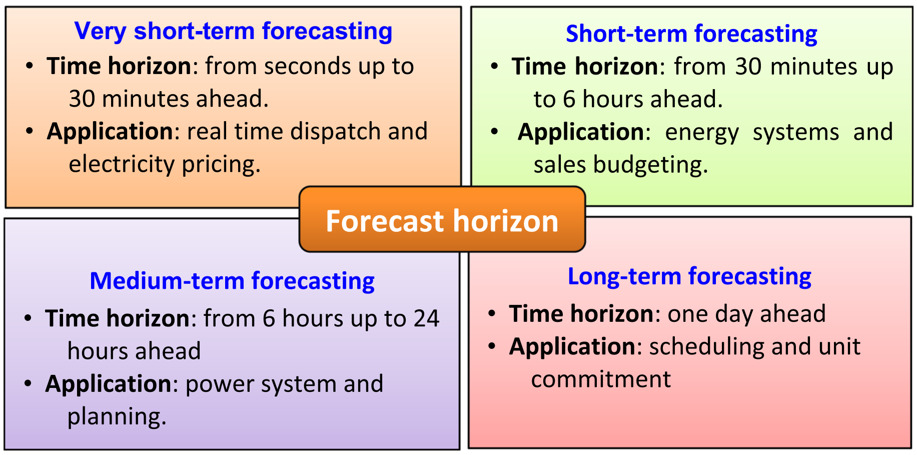

Very short-term forecasting
• Time horizon: from seconds up to 30 minutes ahead.
• Application: real time dispatch and electricity pricing.
Short-term forecasting
• Time horizon: from 30 minutes up to 6 hours ahead.
• Application: energy systems and sales budgeting.
Forecast horizon
Medium-term forecasting
• Time horizon: from 6 hours up to 24 hours ahead
• Application: power system and planning.
Long-term forecasting
• Time horizon: one day ahead
• Application: scheduling and unit commitment

Fig. 1 Categorization of forecast horizons

Short-term solar irradiance forecasting [32] is a challenging task that needs to be urgently solved to cope with the electricity production of many companies and factories. There is also the necessity of having a clean, renewable source of energy that can preserve our environment from pollution and global warming. All the aforementioned reasons motivate us to perform a comparative study to investigate the accuracy of many existing forecasting methods. Our work's primary contribution can be mentioned as follows:

- This study investigates the performance of various deep learning algorithm employed for predicting short-term solar irradiance, such as artificial neural network, Convolutional Neural Network (CNN), Long Short Term Memory (LSTM), recurrent neural network, temporal convolutional network, gated recurrent unit, echo state networks, residual neural network and the hybrid model CNN-LSTM.
- Also, another comparison among many ML algorithms will be conducted for tackling the solar irradiance forecasting, including linear regression, stochastic gradient descent regression, random forest, least absolute shrinkage and selection operator, gradient boosting regression, decision tree regression and K-nearest neighbor regression.
- The GSCV with fivefold cross validation is suggested for hyperparameter tuning for all ML and DL models.
- The dataset was collected from Islamabad over five years, from 2015 to 2019, at hourly intervals using precise meteorological instruments.
- Several statistical analyses are performed, such as *Adjusted* R^2 *score*, *NRMSE*, *MAD*, *MSE* and *MAE* to examine the performance of different algorithm.
- SHAP and LIME are examples of XAI that is responsible for interpreting and understanding the reasons behind getting specific results.

The main organization of the paper is shown as follows. Sect. "Literature Review" introduces a literature review of the recent algorithms developed for predicting the solar irradiance. Sect. "Time series analysis" illustrates the time series analysis and the process flow that is adopted by all ML and DL models in this paper. Sect. "Materials and methods" presents the structure of nine DL algorithms in addition to seven ML algorithms. In Sect. "Results analysis and discussion", the results are covered and discussed. Lastly, the conclusions and recommendations for further research are offered in Sect. "Conclusions and future works".

Literature review

Solar irradiance prediction is a challenging task that has garnered the attention of many researchers. Various advanced methods have proven their outperformance in predicting solar irradiance using ML and DL algorithms, aiming at maximizing solar energy production. Artificial Neural Network (ANN) is one of the most common DL model employed for SIF [33–36]. Integrating fuzzy logic with the ANN [37] improves the forecasting accuracy by 3.2% compared to the standalone ANN. Typically, fuzzy logic determines the correlation between features. Another study by Kumar and Kalavathi [38] proved that the ANN model outperforms the Adaptive Neuro-Fuzzy Inference System (ANFIS) model to predict PV power output.

Moreover, Qing and Niu [39] suggested an hour-ahead SIF in desert locations, considering the frequent dust storms. The MLP model outperforms other models, such as support vector regression, K-nearest neighbor, and decision tree regression, in terms of accuracy. The results of backpropagation ANN [40] show that predicting PV irradiance is better and more accurate compared to the Autoregressive Moving Average (ARMA). A comparative study [41] using six ML algorithms inferred that ANN is the most effective and predictive model out of the six algorithms. Gairaa et al. [42] aimed to forecast hourly global solar irradiation for time horizons from $h+1$ to $h+6$, using two approaches: multiple linear regression and ANN models.

Long short-term memory (LSTM) networks, which belong to the type of Recurrent Neural Networks (RNN), can learn about long-term dependencies while escaping the issue of gradient vanishing. LSTM [43] is another DL model proposed for the SIF problem and achieves better outcomes than support vector regression. Additionally, the LSTM model in [44] considers the interdependence between hours of the same day. The LSTM model [45] performs hyper-parameter tuning to find the optimal values of the parameters. After that, further data is added to see how it affects the improved model. In [46], LSTM is enhanced to predict the short-term and long-term solar irradiance for real-world data. Michael et al. [47] proposed a novel forecast model incorporating LSTM with Bayesian optimization and LSTM with a drop-out structure. To apply the LSTM model's predictions for solar radiation estimation, K-means and the red/blue ratio classify image pixels into clouds or the sky [48].

Researchers have adopted various models for SIF, including support vector machine [49], random forest [50], K-Nearest Neighbor [51], gated recurrent unit [52], echo state network [53], and temporal convolution network [54]. In this study [55], deep learning based on MLP, GRU, LSTM, CNN and CNN-LSTM models uses monthly data to estimate solar radiation. CNN outperforms other models with a MSE for global horizontal irradiance of 12.68 W/m^2. The hybridization of the ML models offers a potent technique that exploits the strengths of various models. ALHM [56] is a hybrid forecast method that combines the ANN with a support vector machine, which has proven to be more accurate for five minutes ahead and daily forecasts.

Also, CNN is incorporated with LSTM [57], such that CNN is used to extract spatial features and LSTM is used to extract temporal features from historical solar irradiance. The work of Kumari and Toshniwal [58] proposed the LSTM-CNN by using LSTM to get the temporal features from time-series data of solar irradiance and then CNN to get the spatial features. Gao et al. [59] suggested integrating CNN-LSTM with CEEMDAN, which extracts features from constitutive series in historical data, in order to reliably predict solar irradiance. In [60], the hybrid model CNN-LSTM-MLP is mixed with the error correction and the VMD method to achieve the one-hour ahead solar irradiance forecasting. Also, Malakouti et al. [61] proposed the CNN-LSTM model for predicting the wind power, which demonstrated more accurate results. Guermoui et al. [62] developed an enhanced hybrid model for multi-hour ahead DNI forecasting. In [63], the authors applied the adjusted Boland–Ridley–Lauret (BRL) model was applied on two time scales, daily and hourly components.

Time series analysis

Time series analysis is a technique for analyzing data gathered over periods of time. In time series analysis, data comprises a set of observations or samples recorded at regular intervals throughout a predetermined time frame, as opposed to being recorded at random intervals. Time series analysis usually needs a lot of data points to guarantee consistency and reliability. Time series data is commonly used in forecasting, which predicts future data based on previous data. Figure 2 demonstrates the process flow and steps of analyzing the time series model for Global Horizontal Irradiance (GHI) data.

After gathering the solar irradiance data from photovoltaic grids, data preprocessing and feature extraction are performed to eliminate any noise, zero, and null values from this data. The next step conducts data normalization to eliminate biased findings and generate features that are on a comparable scale. There are two separate sets of data: the training set and the testing set. The model is trained with the chosen training set, and its performance is checked with a testing set. The final step discusses various statistical analyses done to investigate the performance of the trained models.

Materials and methods

Artificial Intelligence (AI) [64] encompasses methods that enable machines, especially computer systems, to imitate human intelligence in order to perform various tasks. ML is a part of AI that contains advanced techniques that allow machines to learn from data. Furthermore, DL is a branch of ML that uses neural networks for complex data processing. This section will introduce various ML algorithms employed for solving the SIF problem. Also, we will present the most famous algorithms used for tackling the problem in the field of DL. Figure 3 depicts a general view of AI, ML, and DL. Figure 4 exhibits the nine DL algorithms and seven ML algorithms that will be examined in our study.

Deep learning algorithms

The subsection delves into the most prominent deep learning models that are highly suggested for SIF, including Artificial Neural Network (ANN) [65], Multilayer Perceptron

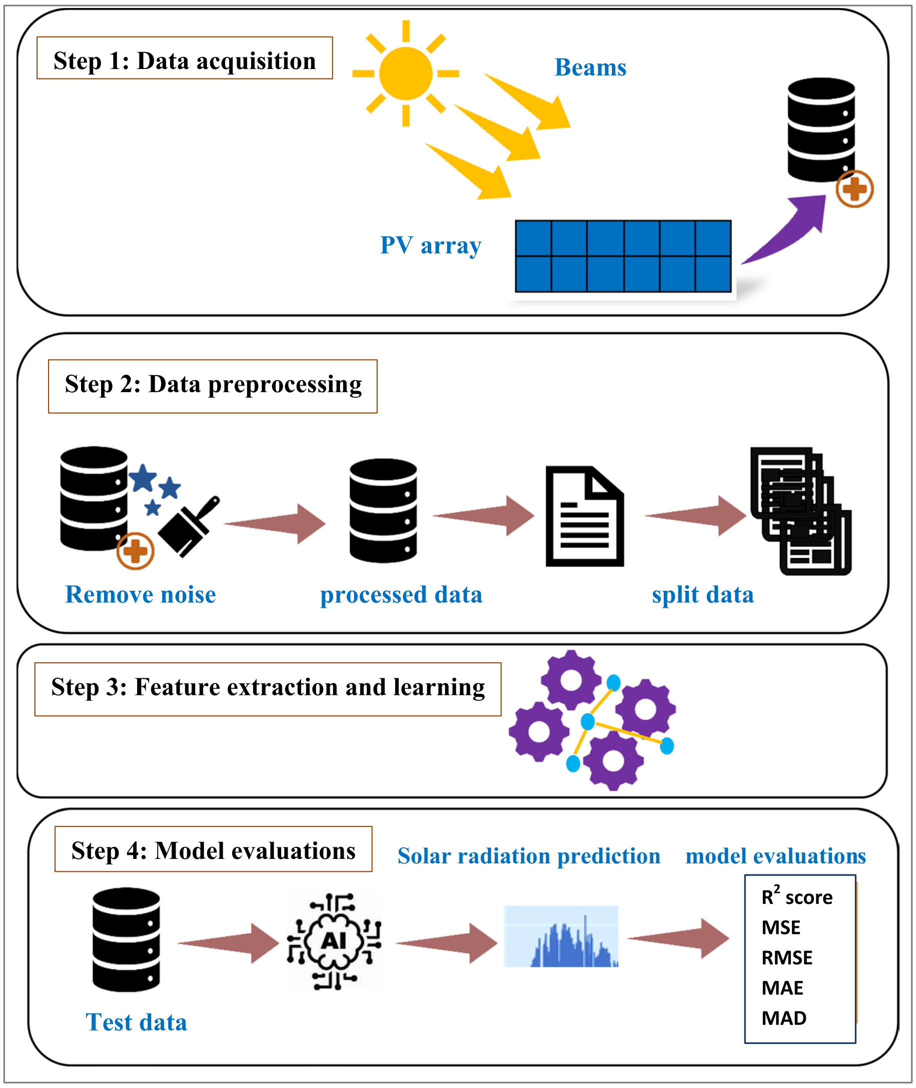


Fig. 2 The process flow of the time series model

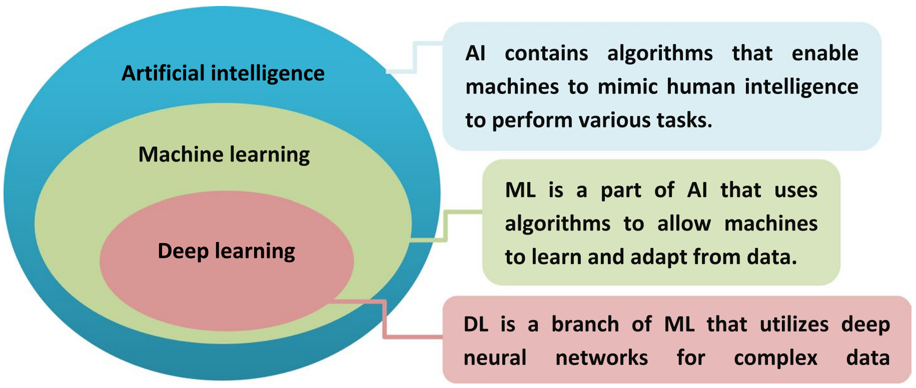

Artificial intelligence
Machine learning
Deep learning
AI contains algorithms that enable machines to mimic human intelligence to perform various tasks.
ML is a part of AI that uses algorithms to allow machines to learn and adapt from data.
DL is a branch of ML that utilizes deep neural networks for complex data

Fig. 3 General view of artificial intelligence, machine learning, and deep learning

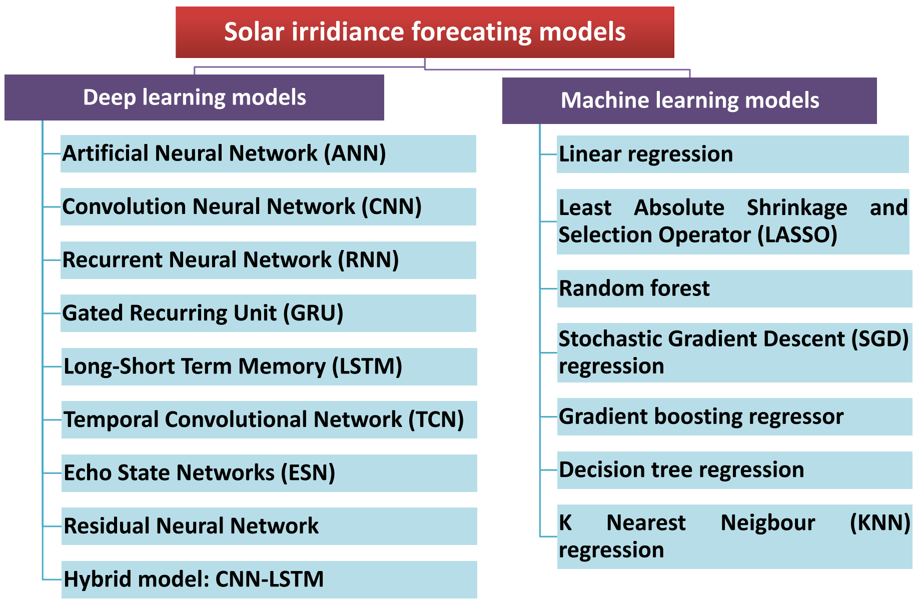

Solar irridiance forecating models
Deep learning models
Artificial Neural Network (ANN)
Convolution Neural Network (CNN)
Recurrent Neural Network (RNN)
Gated Recurring Unit (GRU)
Long-Short Term Memory (LSTM)
Temporal Convolutional Network (TCN)
Echo State Networks (ESN)
Residual Neural Network
Hybrid model: CNN-LSTM
Machine learning models
Linear regression
Least Absolute Shrinkage and Selection Operator (LASSO)
Random forest
Stochastic Gradient Descent (SGD) regression
Gradient boosting regressor
Decision tree regression
K Nearest Neigbour (KNN) regression

Fig. 4 Classification of solar irradiance forecasting models

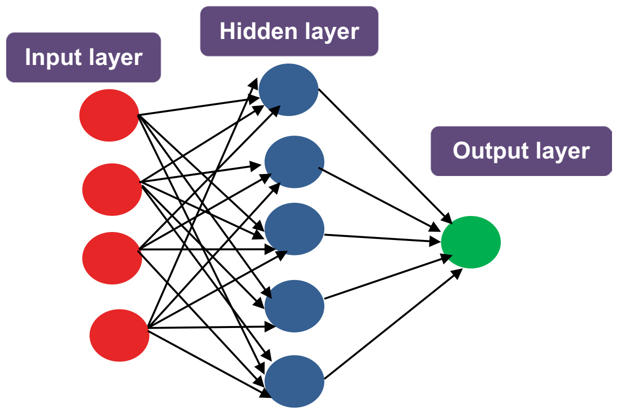


Fig. 5 Architecture of ANN

(MLP) neural network [66], Convolutional Neural Network (CNN) [67], Recurrent Neural Network (RNN) [68], Long Short Term Memory (LSTM) [69], Temporal Convolutional Network (TCN) [70], Gated Recurrent Unit (GRU) [71], Echo State Networks (ESN) [72], Residual Neural network [73], and finally the hybrid model CNN-LSTM [74]. The main structure and details about these DL algorithms will be explained below.

Artificial neural network

The ANN [75, 76] is a computational model consisting of linked neurons arranged into input, hidden, and output layers, designed to simulate the operations of the human brain, as seen in Fig. 5. It is highly recommended for recognizing patterns and forecasting tasks [77]. The input layer is mainly the first layer that receives initial data from the real world. ANN can contain one or more hidden layers. Data is passed from the input layer to the hidden layer, which processes it using learning methods and an activation function. Furthermore, the output layer is the last layer of ANN, consisting of a specific number of nodes representing the output values.

Convolutional neural network

Similar to other forms of ANNs, CNNs [67, 78] are composed of interconnected artificial neurons through weights and arranged in layers. It is typically designed to learn and extract features from visual data or images. Through the use of convolutional layers, pooling layers, CNNs can learn from raw pixel data. By using CNN on time series data for forecasting, it becomes possible to effectively capture temporal relationships and extract important features. Additionally, CNN can interpret the incoming data as a sequence of one-dimensional signals. The CNN utilizes one-dimensional convolutions and pooling procedures to learn local patterns and detect temporal correlations within the data.

CNN comprises many layers like convolution layer, flatten layer, pooling layer, and output layer. For clarity, Fig. 6 displays the architecture of CNN. The convolution layer is usually called the feature extractor layer since it gets the best features of the image and time series data by sliding the filter over the next receptive field of the same data. The CNN employs the Rectified Linear Unit (ReLU) activation function for setting all negative values to be zero using the following equation (see Fig. 7):

$$f(x) = \max(0, x) \tag{1}$$

The pooling layer helps to lessen the spatial volume of data after the convolution layer. There are many pooling methods, such as average pooling, max pooling, and L2-norm pooling. The max pooling approach chooses the highest value within the window, as can be seen in Fig. 8. It can be found after a convolution layer or between two of them but not after the fully connected layer to lessen the computational cost. The flatten layer is located between the CNN and the ANN, and its function is to convert the output of the CNN into an input that the ANN can process to learn complex patterns and make predictions. Finally, the output layer obtains the final output.

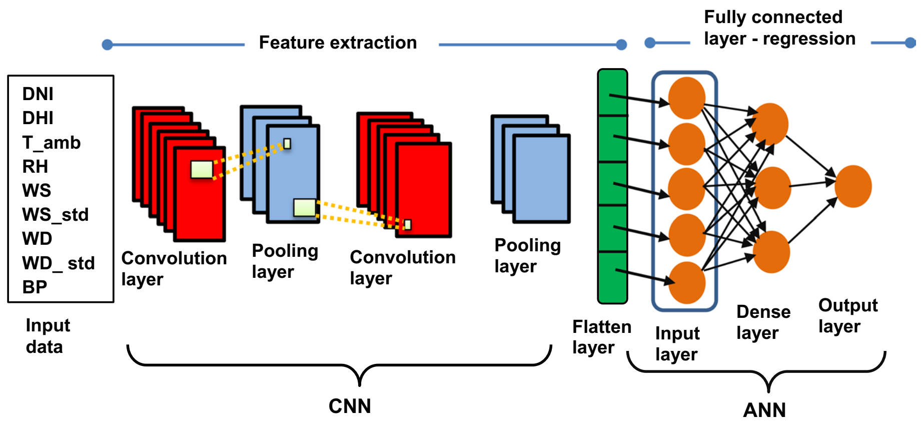


Fig. 6 Architecture of CNN

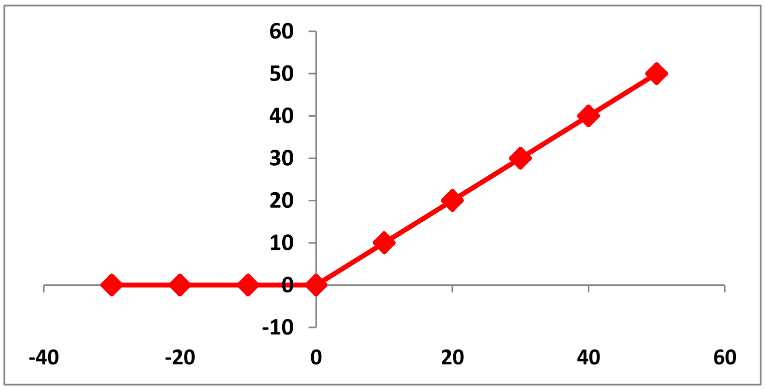


Fig. 7 Graphical representation of the ReLU activation function

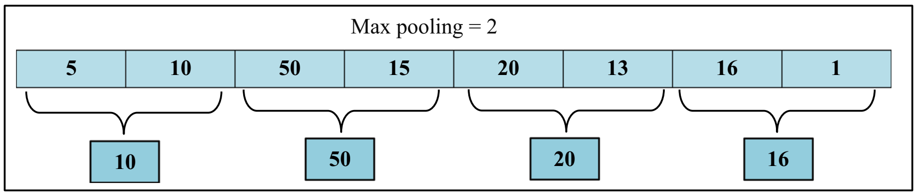


Fig. 8 Max pooling technique

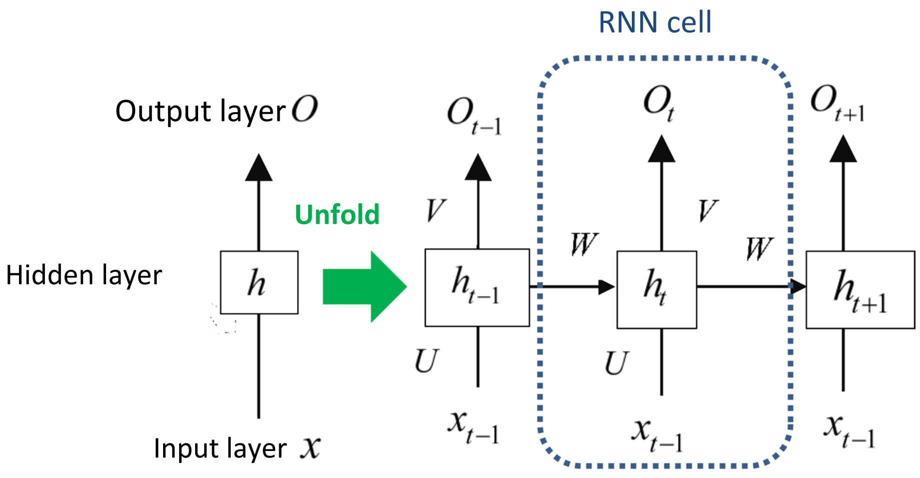


Fig. 9 Architecture of RNN cell

Recurrent neural network

Unlike other neural networks architectures, RNN [79] enables bidirectional information flow through both forward and backward propagation. RNNs are suggested for handling sequence and time series data [80], such as audio, sensor data [81], and text [82]. The RNN consists of three layers: an input layer (x), a recurrent hidden layer (h), and an output layer (O), as shown in Fig. 9. It can be modeled mathematically as follows [83]:

$$h_t = f_{RNN}(U \times x_t + W \times h_{t-1} + b) \tag{2}$$

Table 1 Types of RNN

Type of RNN	Description	Applications
One to one	Single input and single output	Predicting the next word
One to many	Single input and multiple outputs	Image caption
Many to one	multiple inputs and single output	Sentimental analysis and forecasting
Many to many	multiple inputs and multiple outputs	Machine translation

Table 2 Specifications of the used weather data

Variables	Description	Unit	Instrument used for collection
Unnamed: 0	Contain index of samples	–	–
datetime	Date and time for 55 months from year 2015–2020	Hour	–
GHI	Global horizontal irradiance	w/m^2	K&Z CMP21 pyranometer
DHI	Diffuse horizontal irradiance	w/m^2	K&Z CMP21 pyranometer
DNI	Direct normal irradiance	w/m^2	K&Z CHP1 pyrheliometer
T_amb	Ambient temperature	$^\circ c$	Campbell CS215
RH	Relative humidity	%	Campbell CS215
WS	Wind speed	m/s	NRG 40H anemometer
WS_gust	WS maximum within the time interval	m/s	NRG 40H anemometer
WD	Wind direction from North to East	$^\circ w$	NRG 200 wind direction sensor
WD_std	WD standard deviation	$^\circ w$	-
BP	Barometric Pressure	Atmosphere (atm)	Campbell CS100
dni_dev	DNI deviation	-	-
cleaning	cleaning	-	-

$$\hat{y}_t = \sigma(Vh_t) \tag{3}$$

where x_t and h_t are is the input and the hidden states at the current time step t. Furthermore, h_{t-1} refers to the hidden state at the previous time step. The variable $\widehat{y_t}$ refers to predicted output at a time step t. The parameters U, W, and V are the weight vectors for input, hidden, and output layers, respectively. The symbol f_{RNN} indicates the used activation function and b indicates the bias term. The σ is the sigmoid activation function. The feedback loop in its current hidden layer enables the RNN to keep the memory of past information. This short-term memory makes the network to analyze previous inputs when producing output and revealing the relationships between data points that are far from each other (Table 1).

Table 2 encounters various types of RNN according to the number of input/output layers. Also, the table mentions the best suited application for each type of RNN. RNNs work by applying back propagation through time. In this manner, weights for the current and previous inputs are updated by propagation of the error from the last time step to the first one. An advantage of RNN is that the model size does not increase with the size of the input. Long time steps cause weight gradients to vanish, becoming small numbers close to zero. Thus, the network does not learn and isn't suitable for long-term dependencies.

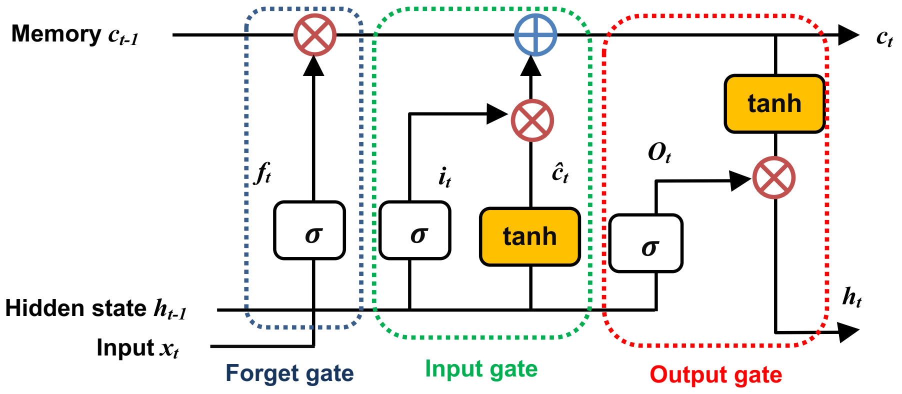


Fig. 10 Architecture of LSTM cell

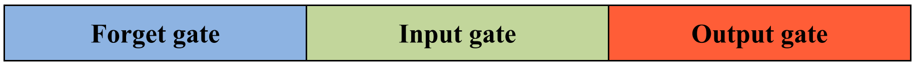


Fig. 11 The function of main LSTM gates

Long short term memory

LSTM can tackle the vanishing gradient problem faced by RNN, since it depends on gates mechanism by introducing input, output, and forget gates [83]. Figure 10 shows the LSTM cell structure. These gates control which information is retained throughout the network. The forget gate (f_t) seeks to forget and delete any irrelevant information from the LSTM cell in a specific time step, as can be seen in Fig. 11. The input gate (i_t) adds and updates new information while the output gate (O_t) returns the updated information. The mathematical model of LSTM cell is defined as follows:

$$f_t = \sigma(w_f x_t + U_f h_{t-1} + b_f) \tag{4}$$

$$\hat{c_t} = \tanh(w_c x_t + U_c h_{t-1} + b_c) \tag{5}$$

$$i_t = \sigma(w_i x_t + U_i h_{t-1} + b_i) \tag{6}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot c_t \tag{7}$$

$$O_t = \sigma(w_O x_t + U_O h_{t-1} + b_O) \tag{8}$$

$$h_t = O_t \odot \tanh(c_t) \tag{9}$$

where the parameters w_i, w_c, w_f, w_o, U_i, U_f, U_c and U_O represent weight vectors. x_t, h_t and c_t indicate the input, hidden state and cell state at a time step t. h_{t-1} and c_{t-1} indicate the previous hidden state and the previous cell state at a time step $t-1$. Moreover, $\hat{c_t}$ determines the candidate memory whereas the symbol $\odot$ determines the element wise dot product. b_f, b_c, b_O and b_i are considered bias terms. σ and tanh refers to the sigmoid and the tanh activation functions.

Gated recurrent unit

Rather than LSTM, GRU is another network that handles the vanishing gradient problem by introducing the reset and update gate. Despite being similar to LSTM, GRU has few advantages. Since GRU has fewer variables than LSTM, its architecture is simpler and more compact. The structure of the GRU cell is given in Fig. 12. These gates determine which information is retained throughout the network. The mathematical model of the GRU cell can be formulated as:

$$r_t = \sigma(w_r x_t + U_r h_{t-1} + b_r) \tag{10}$$

$$z_t = \sigma(w_z x_t + U_z h_{t-1} + b_z) \tag{11}$$

$$\widehat{h}_t = \tanh(w_h x_t + U_h h_{t-1} + b_h) \tag{12}$$

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \widehat{h}_t \tag{13}$$

where r_t and z_t indicate both the reset gate and the update gate. w_r, w_z,w_h, U_r, U_z and U_h refer to the weight vectors for input and hidden states. b_r, b_z and b_h represent bias terms. h_t and $\widehat{h}_t$ are the hidden state and the candidate one at a time step t where as h_{t-1} is the previous hidden state. σ and tanh are the sigmoid and the tanh activation functions.

Temporal convolutional network

TCNs [84, 85] are a powerful tool for handling sequence data, and it offers several benefits over traditional sequence models, such as RNN and LSTM. Figure 13 exhibits the architecture of TCN. TCNs are parallelizable networks and they can avoid gradient issues (vanishing gradients and exploding gradients). Also, they can handle sequences of varying lengths even long-term and short-term which makes it a versatile choice for many time series and sequence-based tasks. Unlike other convolutional networks, TCN employs the causal and dilated convolutions to handle the increased network depth for longer inputs [86]. The dilation factor (d = 1, 2, 4) is generally doubled at each layer. TCNs use 1D convolutional layer, where each layer in the network

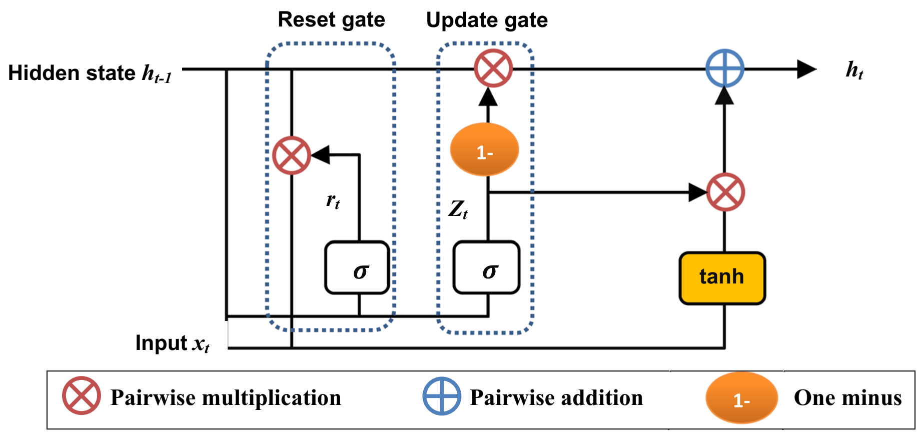


Fig. 12 Architecture of GRU cell

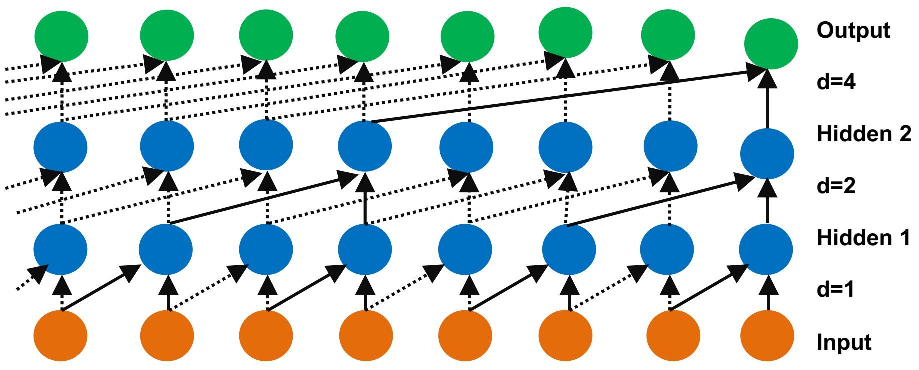


Fig. 13 Architecture of TCN

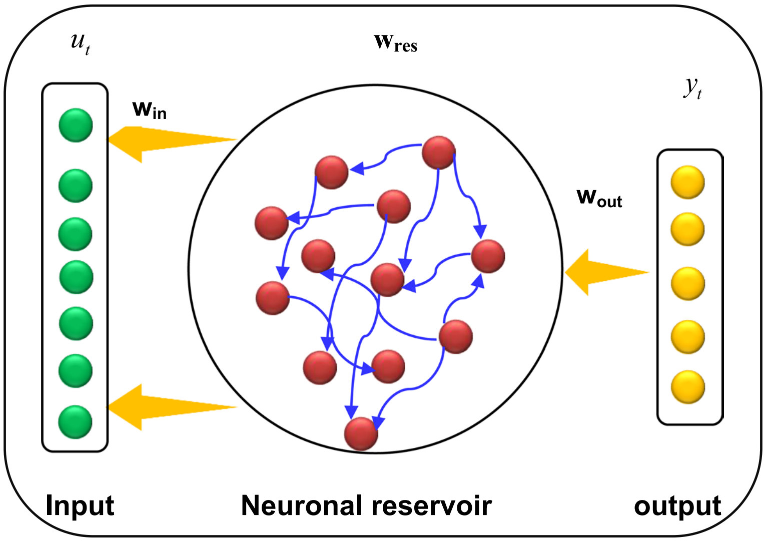


Fig. 14 Architecture of ESN

sees all previous layers' outputs. The TCN model also contains a residual block structure, similar to ResNet [87, 88], for making a skip connection and to facilitate training of deep networks and prevent vanishing gradient descent.

Echo state networks

ESN [89, 90] is a type of RNN. It consists of an input layer (u_t), a reservoir layer (w_{res}) and an output layer (y_t), as depicted by Fig. 14. The reservoir layer comprises a large number of recurrently connected nodes. In the figure, w_{in} refers to the connection weights between input and hidden layers. Additionally, w_{out} indicates the connection between the hidden layer and the output layer. The connection weights between the input layer and the reservoir layer remain unchanged once they have been initialized [91]. Only the weights of the output layer can be trained that simplifies the training process reducing computational complexity. ESN is beneficial for time series forecasting because it can successfully extract temporal dynamics and nonlinear patterns from data.

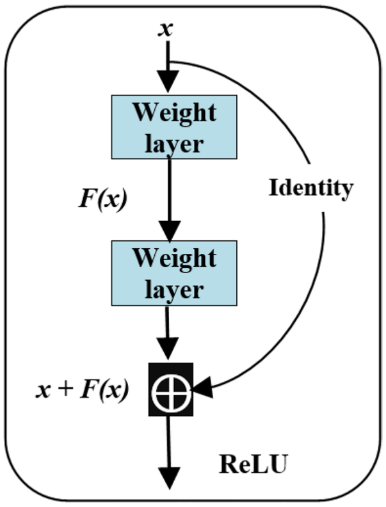


Fig. 15 Residual connection

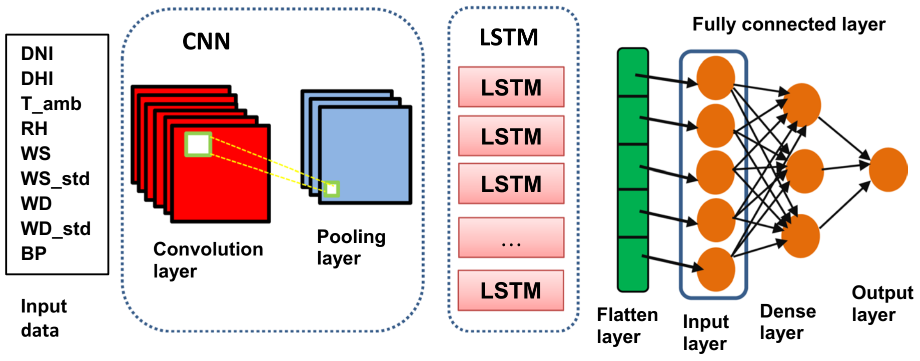


Fig. 16 Architecture of the hybrid CNN-LSTM model

Residual neural network

Residual networks [92] are characterized by skipping connections or jumping over some layers (see Fig. 15). The residual connection performs an identity mapping to x, then the element-wise addition will be done $x + F(x)$. After that, the ReLU activation function will be applied to $x + F(x)$. If x and $F(x)$ has the same dimension, the element-wise addition will be used. Otherwise, the identity mapping with a linear transformation will be done as $w.x + F(x)$, where w is a weight matrix. They allow gradients to directly flow through a network, without passing through non-linear activation functions that make neural networks to fall into the gradient vanishing problem.

CNN-LSTM

A CNN-LSTM [93] is a hybrid model that incorporates CNN layers at the beginning of CNN-LSTM model to extract relevant features from input data. These features are then passed to the subsequent LSTM layers for interpreting the features across time steps [94]. The architecture of the CNN-LSTM model is depicted in Fig. 16, containing a fully connected layer of large number of neurons, organized in three layers: input, dense and output.

Machine learning algorithms

This subsection presents some of the most popular ML algorithms like linear regression [95], Stochastic Gradient Descent (SGD) regression [96], Least Absolute Shrinkage and Selection Operator (LASSO) [97], random forest [98], gradient boosting regression [99], decision tree regression [100] and K-Nearest Neighbor (KNN) regression [101]. These regression models are regarded as popular tool for statistical analyses and they will be explained below.

Linear regression

Linear regression [102, 103] is a particular type of supervised ML algorithm employed for tackling regression issues and estimating the correlation between two both variables. It postulates a linear relationship between the independent variable (feature) and the dependent one (target) with the objective of finding the best fit line that expresses the relationship. The line is found by lessening the sum of squared differences between the real values and predicted ones. The generalized formulation for linear regression is:

$$\hat{y} = X\beta + b \tag{14}$$

$$\beta = (X^T X)^{-1} X^T y \tag{15}$$

$$b = mean(y) - mean(X\beta) \tag{16}$$

$$\hat{y}_{new} = X_{new}\beta + b \tag{17}$$

where $\hat{y}$ indicates the predicted value.X denotes the matrix of input features. β defines a vector of weights or coefficients.b is the intercept term (bias).X_{new} refers to the new data. $\hat{y}_{new}$ is the new predicted Value. The used weights vector for linear regression is β= [134.77742582, 135.5658808, 32.3798911, -13.60180839, 14.06928621, -18.5702262, -1.53548565, 46.40629438, 11.85316545], while the intercept value (b) is equal to 315.6134357765278.

Stochastic gradient descent regression

One of the supervised ML algorithms that may be used to tackle regression issues is called SGD regression [104]. The iteratively updating of the model weights using a small random portion of the training data instead of the entire draining data makes SGD a common choice for large-scale regression tasks.

Least absolute shrinkage and selection operator

LASSO regression [105] is regarded as an extension of Ordinary Least Square (OLS). With OLS, the estimates may suffer from high variance and lack of interpretation due to the presence of large number of predictors [106]. Hence, LASSO comes to handle the aforementioned drawbacks of OLS. It is used for regression problems with more accurate prediction. The main objective of LASSO regression is to discover the values of the coefficients that reduce the summation of the squared differences between the actual values and predicted values [107]. LASSO function adds a penalty term to the loss function of OLS equal to the absolute value of the weights associated with each feature variable. The loss function for LASSO regression can be expressed as below:

$$LASSO_{loss} = OLS_{loss} + Penalty\_term = \sum_{i=1}^{m} (y_i - \sum_{j=1}^{p} \hat{y}_{ij} w_j)^2 + \alpha \sum_{j=1}^{p} |w_j| \tag{18}$$

where m and p refers to the number of observations and dimensionality of features. α is a hyperparameter that indicates the strength of the regularization.w_j are the assigned weights for LASSO. L1 regularization moves any weight values to zero to allow other coefficients to take non-zero values. y_i denotes the i^{th} actual value and $\hat{y}_{ij}$ represents the i^{th} predicted value of feature j. The used LASSO weights vector in our work is [134.78748341, 135.60866578, 32.32577104, −13.64606087, 12.66907409, −17.07588968, −1.32131259, 46.09668833, 11.72830064] and LASSO Intercept is equal to 315.6143943704034.

Random forest

Random forest [108] is an ensemble technique that has the ability for solving both regression and classification problems with the use of multiple decision trees. It supports the bagging technique where each decision tree is trained on a random subset of the training data. It has gained a significant popularity in recent years because of its impressive performance, simplicity in implementation and little processing requirements [109]. It is used to deal with bias-variance trade-offs for reducing the variance of a prediction model.

Random forests [110, 111] involve creating multiple decision trees during training and then outputting the average prediction (in the case of regression) of the individual trees. Each tree is trained on a random subset of the data and features, which helps to reduce overfitting and improve generalization. The pooling of multiple trees helps stabilize predictions and increase accuracy. Additionally, this approach provides better interpretability and faster training times in many practical applications. The random forest approach contains three parameters to adjust:

- *Number of estimators* represents the number of trees. Increasing the number of estimators can lead to higher accuracy, but it is computationally expensive.
- *Maximum features* indicate the number of features for making a split decision.
- *Maximum depth of a tree* indicates how deep the tree can grow. The deeper the tree, the more complex it is.

Gradient boosting regression

Gradient boosting regression [112] is a supervised ML problem based on the idea of an ensemble method derived from a decision tree. It seeks to minimize the loss function of the gradient boosting regression model through the addition of many weak learners using gradient descent. Theses decision trees are built in a greedy approach with split points that minimize the function loss.

Decision tree regression

A decision tree regression is a ML algorithm that aims to predict a continuous target that partitions the features/variables into regions and passing predictions to each region based on feature values. It works by constructing a tree-based structure by recursively splitting the feature space based on input values. The leaves reflect the values of target variables values, while the branch lines represent the combinations of input variables that result in these values [113]. It predicts continuous target variables by traversing the tree and assigning values to unseen data points.

KNN regression

KNN is another ML algorithm that is highly recommended for regression and classification tasks. The model constructs its prediction based on finding the K nearest data points to a particular input by averaging the observations in the same neighborhood or choosing the majority class [26]. The distance between the neighbors and the given data can be calculated using Euclidean distance. The Euclidean distance (*ED*) between two points $x = (x_1, x_2, ..., x_d)$ and $y = (y_1, y_2, ..., y_d)$ in a d-dimensional space is calculated as:

$$ED_{x,y} = \sqrt{\sum_{j=1}^{d} (x_j - y_j)^2} \tag{19}$$

The KNN is built by K neighbors and each of the neighbors is assigned a weight based on its distance to the query point. The closer a neighbor, the higher its weight is. The weight of the i^{th} neighbor (w_i) is given by:

$$w_i = \frac{1}{ED_{x,y_i}} \tag{20}$$

ED_{x,y_i} is the distance between the query point x and its i^{th} neighbor y_i. The predicted value ($\hat{y}$) for the query point is the weighted average of the target values of its nearest neighbors that can be computed by:

$$\hat{y} = \frac{\sum_{k=1}^{K} w_k \cdot y_k}{\sum_{k=1}^{K} w_k} \tag{21}$$

y_k is the target value of the k^{th} neighbor, where $k = 1, 2, ..., K$.

Interpretable results

Explainable Artificial Intelligence (XAI) allows anyone to understand the logic or reasons produced by the ML model. It aims to develop AI systems that demonstrate more transparency in their decision-making processes. The level of interpretability of a model directly correlates with the ease of understanding the underlying justifications for certain judgments or projections. The ML model is more interpretable than another if its explanations are more understandable to a person than the choices made by the other model. Understanding the underlying reasoning behind a model's prediction is crucial for the widespread adoption of prediction in many applications.

Shapley additive explanations

SHapley Additive exPlanations (SHAP) [114] utilizes the principles of game theory to elucidate the predictions of any ML model by quantifying the impact of each feature on the prediction output. The SHAP method identifies the primary features that influence the model's forecast. SHAP approximates the original model to a specific input and reduces the impact of missing features as follows:

$$g(z\prime) = \varphi_o + \sum_{j=1}^{m} \varphi_j z_j\prime \tag{22}$$

where g represents the explanation model. $z\prime \in \{0,1\}^m$ is a binary variable where 1 indicates that the simplified features are same as the original and 0 indicates that the features are not the same as the original. Moreover, φ_j refers to the attribute effect for the j^{th} feature such that $j = 1, ..., m$ and m determines the number of simplified features.

Local interpretable model-agnostic explanation

The Local Interpretable Model-Agnostic Explanation (LIME) [115] serves as a "explainer" to elucidate the prediction outcome for individual data samples. The LIME output consists of interpretations that depict the contribution of each feature to the prediction for a specific sample, serving as a means of providing local interpretability.

Results analysis and discussion

This section will explore the solar irradiance dataset and the results of different models from ML and DL on this data. SubSect. "Environment setup" will present the configuration of the experimental environment in which our tests are conducted. The characteristics of the solar irradiance dataset are outlined in the following subSect. "Dataset description". SubSect. "Dataset distribution" explains the distribution of data. The preprocessing steps and feature selection are discussed in subSect. "Data preprocessing and feature selection". An illustration of the data that has been divided into training, testing, and validation may be seen in subSect. "Data split". SubSect. "Hyperparameters tuning" introduces the GSCV to optimize the hyperparameters of various ML and DL models. SubSect. "Evaluation metrics of algorithms for solar irradiance forecasting" presents the models evaluation for checking their performance. Finally, the results of DL and ML models are discussed and statistically analyzed in subSect. "Results for deep learning models" and subSect. "Results for machine learning algorithms", respectively.

Environment setup

For fair comparison among all algorithms, they are implemented and run on the same data taken from Kaggle website. All algorithms are trained with GPU NVidia Tesla P100 and RAM of 16 GB. They are developed using Python environment of Version 3.10.12.

Dataset description

This study utilizes global horizontal solar irradiance data acquired in Islamabad, located at 33.64°N, 72.98°E and 500 m above sea level with meteorological parameters. The data was gathered over five years from 2015 to 2019 taken at hourly intervals by employing accurate meteorological devices. The weather data contains 14 columns. The dataset comprises 41256 samples. The characteristics and specifications of the data are listed in Table 2.

Dataset distribution

To understand the distribution of data, we employ several statistical measures, including mean (*Mean*) and standard deviation (*Std*) that are calculated as in Eq. (23) and Eq. (24), respectively. Furthermore, box plots [116] are utilized to summarize data distributions, detect any skewness, identify outliers, and make comparisons between distributions. They offer five measures, including the minimum (*Min*), first quartile, median, third quartile, and maximum (*Max*) of the data values, that help to identify data spread and identify potential anomalies.

$$Mean = \frac{\sum_{i=1}^{n} x_i}{n} \tag{23}$$

$$Std = \sqrt{\frac{\sum_{i=1}^{n} x_i - Mean}{n}} \tag{24}$$

x_i is the i^{th} observation in data, where $1 \leq i \leq n$. n indicates the number of observations. Table 3 introduces the statistical analysis of data distribution. From the table, the dataset has different distributions, where some variables have a wide range of values with significant variations, such as GHI and DNI, while other variables have smaller ranges and

Table 3 The statistical analysis on the solar irradiance dataset distribution

Dataset	Measure							
	Count	Mean	Std	Min	First quartile	Second quartile	Third quartile	Max
GHI	41256	197.59	278.98	0.00	0.00	7.00	375.00	1040.00
DNI	41256	177.70	301.89	0.00	0.00	0.00	313.00	2777.00
DHI	41256	88.43	127.92	0.00	0.00	5.00	152.00	843.00
T_amb	41256	21.90	8.49	1.00	14.90	22.80	28.40	45.30
RH	41256	59.02	20.83	10.00	42.00	59.00	76.00	100.00
WS	41256	1.78	1.38	0.00	0.70	1.50	2.60	13.20
WS_gust	41256	4.19	2.47	0.00	2.40	3.60	5.40	31.00
WD	41256	175.18	108.24	0.00	70.00	186.00	275.00	360.00
WD_std	41256	13.04	10.08	0.00	4.60	12.00	19.50	82.20
BP	41256	944.77	9.57	881.00	940.00	946.00	951.00	1002.00
dni_dev	41256	−9.59	20.74	−100.00	−16.20	0.00	0.00	100.00
cleaning	41256	0.01	0.12	0.00	0.00	0.00	0.00	1.00

lower variations, such as T_amb, RH, WS, WS_gust, WD, WD_std, and BP. The variables dni_dev and cleaning have a substantial number of zero values.

Principal Component Analysis (PCA) is a statistical technique used to reduce the dimensionality of a dataset while retaining the variation in the data. This is particularly useful when dealing with high-dimensional data to make it easier to visualize and analyze, without losing significant information. From Fig. 17, it is evident that the cumulative explained variance plateaus after 9 components. This means that while PCA can reduce the dimensions, the first 9 components or features already capture most of the variability in the data. Since our data originally contains 9 features, which is the total number of features in the original dataset, would not provide any dimensionality reduction benefits.

Furthermore, Fig. 18 displays the importance of each feature. A higher importance value means that the specific feature will have a larger effect on the model that is being used to predict the target value GHI. The input DHI values have a significant effect on the target GHI with an importance value of 1.441757. Moreover, Fig. 19 shows the relationship between each feature and the other features. Figure 20 depicts the *entropy* [117] for the data samples over time, which takes a value from 0 to 1. A higher *entropy* value indicates a more uncertain or random distribution, whereas a lower *entropy* value corresponds to a more predictable or deterministic distribution. By inspecting the figure, we can see that most entropy values lies in the interval from 0.4 to 0.8. When the entropy is moving toward 1, it means that the dataset has a random distribution, which puts more challenges in predicting the solar irradiance.

Data preprocessing and feature selection

Data preprocessing and feature selection ensure data quality, data normalization, and noise reduction. Data distribution shows that dni_dev and cleaning features

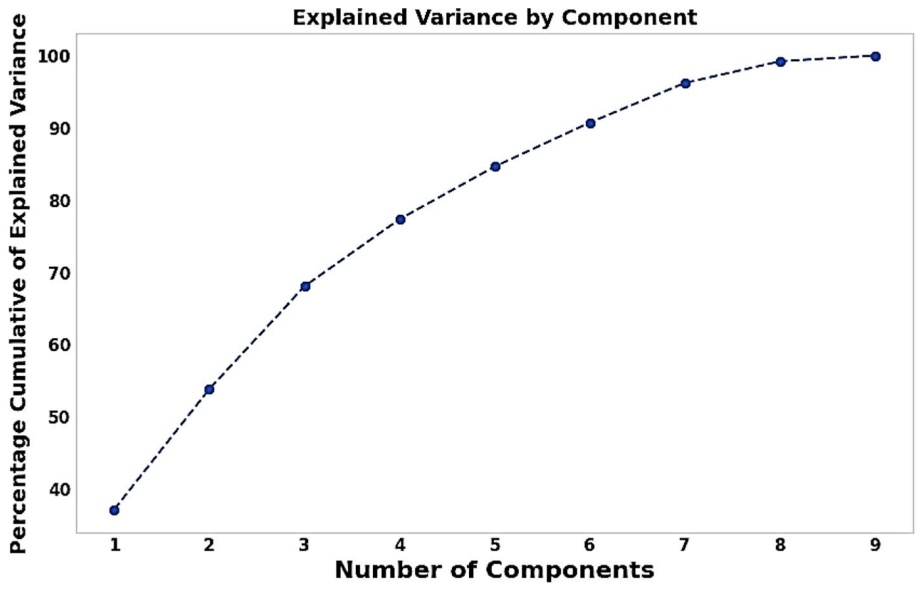


Fig. 17 PCA reduction technique for the weather data

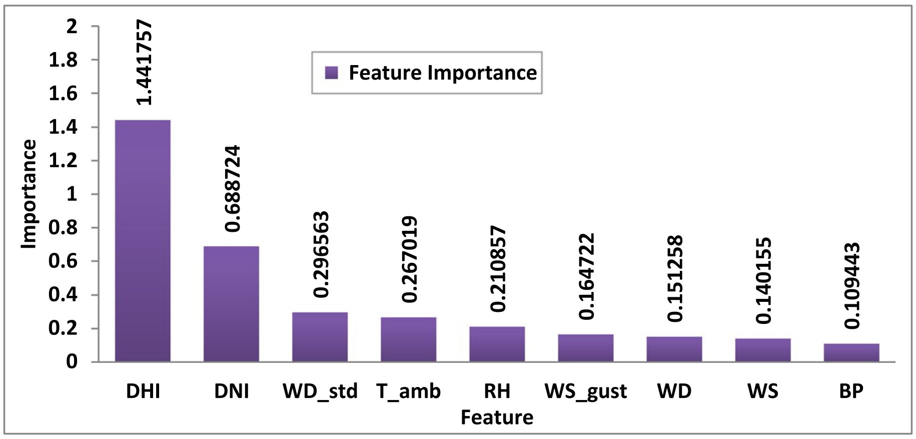


Fig. 18 The importance of each feature of the weather data

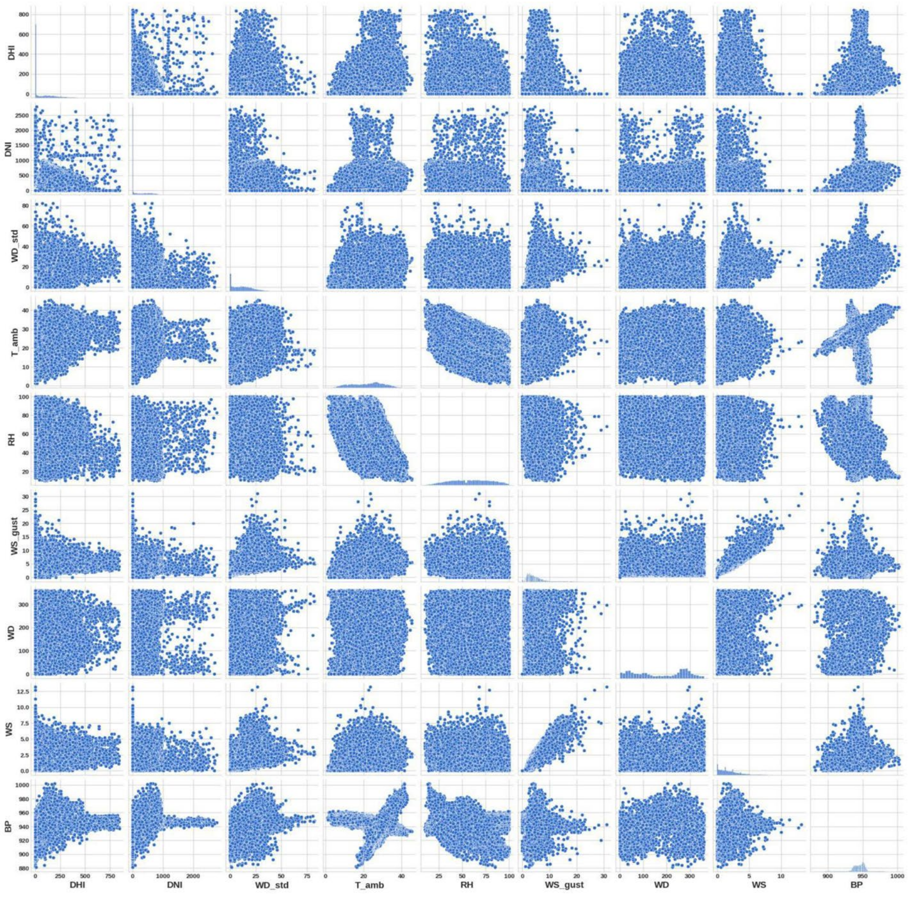


Fig. 19 The relationship between each feature and the other features

have many zero values. Hence, dni_dev and cleaning is dropped from data, as well as unnamed: 0 is dropped as it contains the index of samples. Also, the contribution and correlation of these features to the output is weak, thus deleting them will be advantageous. From the Person correlation plot in Fig. 21, the darker degree of red and

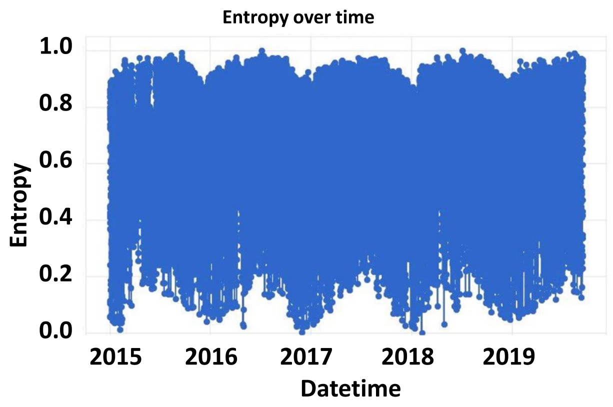


Fig. 20 Entropy for the data samples over time

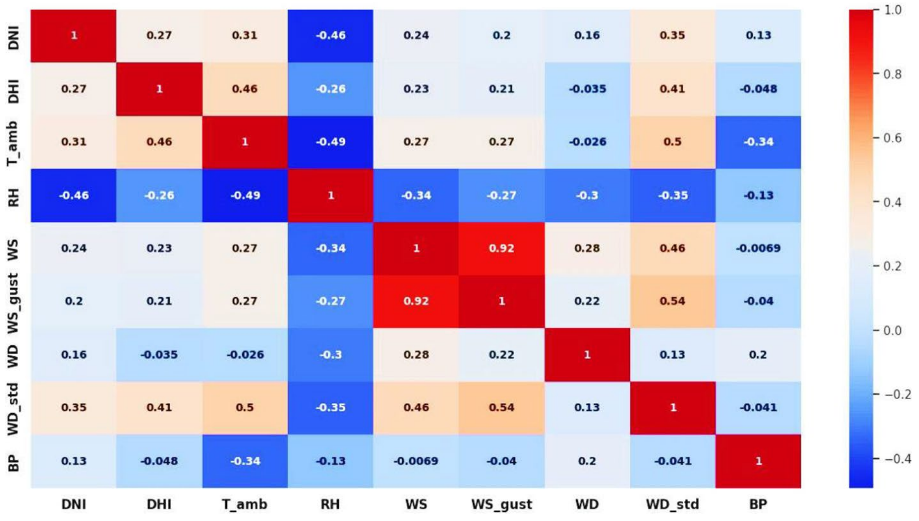


Fig. 21 The heatmap correlation of the features

blue colors indicates a strong correlation, while the brighter one indicates weak correlation. Finally, the remaining features that will be taken into consideration are GHI, DNI, DHI, T_amb, RH, WS, WS_gust, WD, WD_std and BP.

The analysis reveals that a large number of values, approximately 19,403 from GHI are equal to zero. Indeed, the zero values represent night values, and it is clear that they are zero owing to the absence of solar radiation. Thus, it is advisable to clean the data at night from 19:00 to 5:00 in the morning and only utilize data from the daylight period when solar radiation is present. Zero values will be dropped to 4236 after this process.

Since there are several data distributions—wide range and small range—data normalization is crucial for eliminating bias results, creating features of a comparable scale, enhancing ML algorithm stability and convergence, improving model accuracy and generalizability by minimizing outliers, and improving efficiency by minimizing complexity and storage needs. The standard scalar normalization (z) of data can be calculated by:

$$z = \sqrt{\frac{x_i - Mean}{Std}} \tag{25}$$

where x_i refers to the i^{th} value from data to be normalized. *Mean* and *Std* are computed as in Eq. (24) and Eq. (25), respectively.

From the statistical analysis of the data nature, input data contains DNI, DHI, T_amb, RH, WS, WS_gust, WD, WD_std and BP that has a high correlation with the target data (GHI) with a window size of 4 for neural networks. The window size in a neural network is crucial for effectively capturing local information, compressing information, extracting attributes at different scales, and achieving a balance between local and global information.

Data split

After the normalization step, the data split phase starts, as depicted in Fig. 22. For ML algorithms, the data is divided into training and testing sets for the standard training and evaluation process. However, for DL algorithms, it is common to split the data into training, testing, and validation sets. We use the training set to train the DL model, the testing set to assess its performance, and the validation set for hyperparameter tuning and model selection. This helps in choosing the best hyperparameters for the model and comparing the performance of different models due to the complexity of DL algorithms compared to ML ones. The validation set is unnecessary for ML algorithms because they have simple model architectures in contrast to DL models that have more complicated structures.

For the DL algorithms the data is divided into three sets: training, validation and testing, with 70%, 15%, and 15%, respectively as adopted by many previous studies [43, 118–120]. For ML approaches, the data is divided into two sets: training and testing, with 20,000 records of the data for the training process and 5,785 records for the testing process [36].

Hyperparameters tuning

GSCV [121, 122] is a powerful technique for optimizing the hyperparameters of DL models. Cross-validation is a crucial method used to build better-fitting models by training and testing on all parts of the training dataset. The grid search with fivefold cross-validation [123, 124] is utilized on the training and validation sets and the testing set is kept away for final evaluation. The number of folds can be increased or decreased. A higher number of folds value may lead to a less biased model, and more complex model.

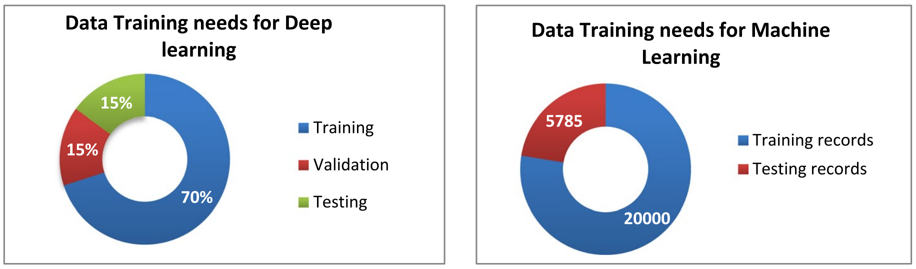


Fig. 22 Data split for DL and ML algorithms

A lower number of folds value may like the simple train-test split method. fivefold cross validation [125, 126] is adopted in many previous works. In GSCV, the training set is partitioned into five equal-sized folds or parts. During the five iterations, one fold is used for testing while the remaining folds are used for training. This process is repeated until all folds have been used for testing.

Figure 23 depicts the used grid search with fivefold cross-validation for DL algorithms. For fair comparison among all DL algorithms, it has been taken into account that the models are trained and validated on the same parts of the data, as can be seen by the figure. Also, the grid search cross-validation is performed on the training set for better hyperparameters tuning, as can be illustrated in Fig. 24. Moreover, each algorithm is trained and tested on the same parts of the data, as in the figure.

Evaluation metrics of algorithms for solar irradiance forecasting

Various performance measures are employed for evaluating all the models, including Mean Squared Error (*MSE*), R^2 *score*, *Adjusted* R^2 *score*, Median Absolute Deviation (*MAD*), Root Mean Squared Error (*RMSE*), Normalized Root Mean Squared Error (*NRMSE*) and Mean Absolute Error (*MAE*). *MSE* is a primary performance metric that measures the average of the squares of errors and computed by:

$$MSE = \frac{\sum_{i=1}^{n} (y_i - \widehat{y}_i)^2}{n} \tag{26}$$

n indicates the number of the samples and y_i indicates the i^{th} actual value, where $i = 1, \ldots, n$. $\widehat{y}_i$ represents the i^{th} predicted value. R^2 score is a statistical indicator that indicates how closely is the regression predictions from the actual data points. The 1.0 value means that the regression predictions completely fit the actual data. When the R^2

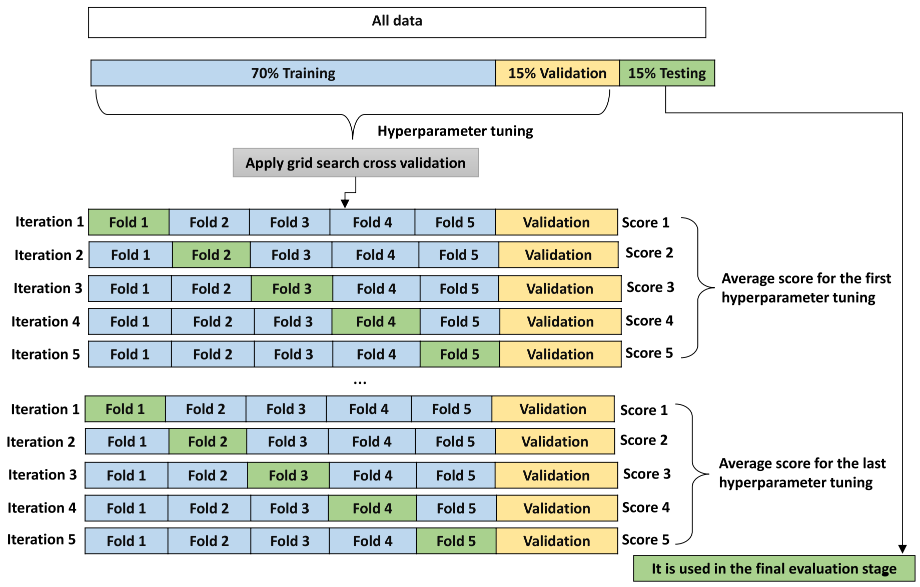


Fig. 23 Grid search cross-validation for DL algorithms

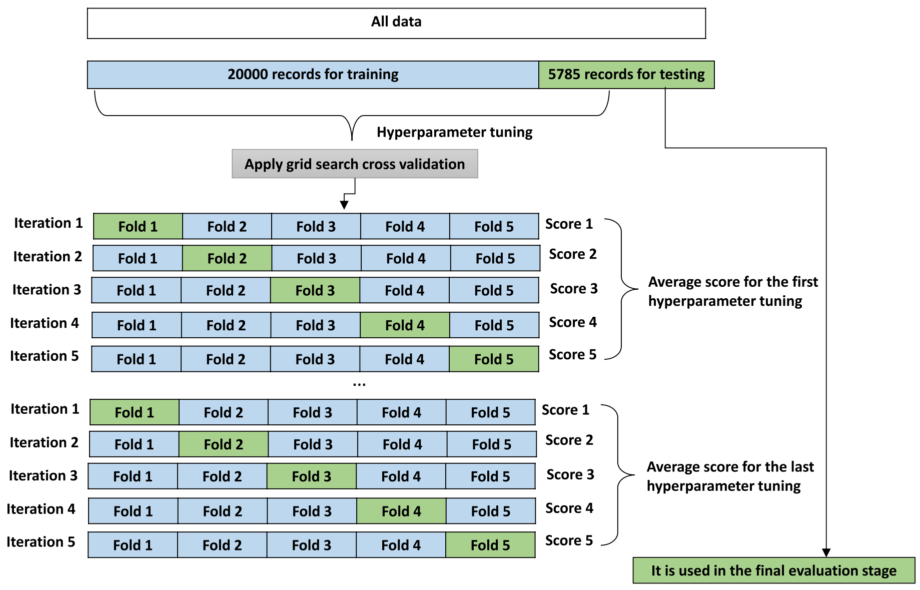

All data
20000 records for training
5785 records for testing
Hyperparameter tuning
Apply grid search cross validation
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Score 1
Score 2
Score 3
Score 4
Score 5
Average score for the first hyperparameter tuning
...
Average score for the last hyperparameter tuning
It is used in the final evaluation stage

Fig. 24 Grid search cross-validation for ML algorithms

score value moves from one to zero, the predictions move away from the actual data. R^2 *score* can be calculated by:

$$R^2(\%) = 1 - \frac{\sum_{i=1}^{n} (\widehat{y}_i - \overline{y}_i)^2}{\sum_{i=1}^{n} (y_i - \overline{y}_i)^2} \times 100 \tag{27}$$

$\overline{y}$ represents the mean of all the values. R^2 *score* measures the overall fit of the model to the data and has a tendency to increase when additional independent variables are added to the model, even if they have little or no explanatory power. This can lead to overfitting, while *adjusted* R^2 *score* considers the goodness-of-fit while taking into account the number of predictors and avoids overfitting. It can be computed as:

$$Adjusted\, R^2 = 1 - \left[\frac{(1 - R^2)(n-1)}{(n-k-1)}\right] \tag{28}$$

where k represents the number of independent variables in the model. *RMSE* is a common performance metric measures the average difference between predicted values and real values. A lower *RMSE* indicates a superior model and more accurate predictions. The lower *RMSE* determines that there is a small difference between the real and the predicted values. *RMSE* can be formulated mathematically as follows [127]:

$$RMSE(w/m^2) = \sqrt{\frac{\sum_{i=1}^{n} (y_i - \widehat{y}_i)^2}{n}} \tag{29}$$

The Normalized *RMSE* (*NRMSE*) is an extension for *RMSE* which is more suitable for different scales of data and can be calculated as follows:

$$NRMSE = \frac{RMSE}{y_{\max} - y_{\min}} \tag{30}$$

$y_{\max}$ and $y_{\min}$ determine the maximum and minimum actual values. Another metric to evaluate the models is *Mean Absolute Deviation (MAD). MAD* is mainly defined as the median difference between the observations (actual values) and model output (predictions). It can be expressed as:

$$MAD = Median(|y_i - \widehat{y}|) \tag{31}$$

Median is the midpoint of a data collection; half of the data points have values lower than or equal to it, and half have values higher or equal to it. *MAE* is defined as the average variance between the real and predicted values and computed using Eq. (32) [127].

$$MAE(w/m^2) = \frac{1}{n} \times \sum_{i=1}^{n} |y_i - \widehat{y}_i| \tag{32}$$

Results for deep learning models

In this subsection, we will investigate the performance of many DL algorithms for tackling SIF problem. To ensure a fair comparison among all DL algorithms, they were trained under identical environmental circumstances. We used Keras API of version 2.12.0 which is a Python framework for deep learning to implement all our selected DL models. Keras is built on top of TensorFlow and offers a comprehensive set of tools for constructing and optimizing neural networks. The loss function for each model is also determined by squaring the mean error, as shown in Eq. (26). The Adam optimizer is utilized with all DL except for the DNN model, we use rmse optimizer. Table 4 records the best learning rate values obtained by the grid search cross validation for each DL model. The batch size is defined as 32, indicating the quantity of samples processed prior to updating the model. Moreover, the window size is equal to 4. The architecture of various DL models is stored in Table 5.

The number of epochs is set to 100, where it represents the number of complete iterations through the training dataset. Early stopping is triggered if a fault arises during

Table 4 Learning rate for each DL model

DL model	Learning rate
Residual NN	0.01
ESN	0.1
TCN	0.001
LSTM	0.001
GRU	0.001
Simple RNN	0.001
CNN-LSTM	0.001
ANN	0.01
CNN	0.001
MLP	0.1

Table 5 The architecture of different DL models

Model	Architectures	Model	Architectures
Residual NN	Input layer (4,9) Residual Block consist of (Dense layer (unit = 64, activation = relu), Dense layer (unit = 9, activation = relu), Adding Second Dense with input layer, Dense layer (unit = 64, activation = relu)) Flatten layer Output layer (unit = 1, activation = linear)	CNN-LSTM	Input layer (4,9) Conv1D (filters = 128, kernel_size = 3, padding = same, activation = relu) Conv1D (filters = 128, kernel_size = 3, padding = same, activation = relu) Dropout (0.3) MaxPooling1D (pool_size = 1) LSTM (64, return_sequences = True) LSTM (64) Output layer (unit = 1, activation = linear)
MLP	Input layer (4,9) Dense layer (unit = 50, activation = relu) Flatten layer Output layer (unit = 1, activation = linear)	ESN	Input layer (4,9) ESN layer (256, return_sequences = True, use_norm2 = True) Flatten layer Output layer (unit = 1, activation = linear)
ANN	Input layer (4,9) Dense layer (unit = 32, activation = relu) Dropout (0.1) Dense layer (unit = 32, activation = relu) Dense layer (unit = 32, activation = relu) Dropout (0.1) Flatten layer Output layer (unit = 1, activation = linear)	CNN	Input layer (4,9) Conv1D (filters = 64, kernel_size = 3, padding = same, activation = relu) Conv1D (filters = 64, kernel_size = 3, padding = same, activation = relu) Dropout (0.5) MaxPooling1D (pool_size = 1) Flatten layer Dense layer (unit = 100, activation = relu) Output layer (unit = 1, activation = linear)
GRU	Input layer (4,9) GRU layer (32, return_sequences = True) Dropout (0.1) GRU layer (32, return_sequences = True) Dropout (0.1) GRU layer (32, return_sequences = True) Dropout (0.1) Output layer (unit = 1, activation = linear)	Simple RNN	Input layer (4,9) Simple RNN layer (32, return_sequences = True) Dropout (0.1) Simple RNN layer (32, return_sequences = True) Dropout (0.1) Simple RNN layer (32, return_sequences = True) Dropout (0.1) Output layer (unit = 1, activation = linear)
LSTM	Input layer (4,9) LSTM layer (32,return_sequences = True) LSTM layer (16, activation = relu) Output layer (unit = 1, activation = linear)	TCN	Input layer (4,9) TCN layer (nb_filters = 256, return_sequences = True, dilations = [1, 2, 4, 8, 16, 32]) Flatten layer Output layer (unit = 1, activation = linear)

training or if there is no improvement in the model's validation performance for a specific period of epochs ($patience = 5$). The parameter *patience* represents the number of epochs with no improvement. The kernel size for CNN and CNN-LSTM is equal to 3. Additionally, the ReLU activation function is utilized by the hidden layer, whilst the linear activation function is utilized by the output layer. Furthermore, a lower dropout rate is set to 0.2 in order to prevent overfitting.

Table 6 records the results of various DL models, including ANN, CNN, RNN, LSTM, GRU, TCN, ESN, Residual NN, MLP and CNN-LSTM. We employ many performance metrics to assess the model's quality, such as the number of parameters, the R^2 *score*,*Adjusted*R^2*score*, *MSE*, *RMSE*, *NRMSE*, *MAE* and *MAD*. The bold font in the table indicates the best results obtained by the existing models. From the results, we can see that CNN-LSTM comes in the first rank by achieving the maximum *Adjusted* R^2 *score* value of 0.984, which means the regression predictions are

Table 6 The results of the DL models for solar irradiance forecasting

Model	# Parameters	Adjusted R^2 score	MSE	RMSE	NRMSE	MAE	MAD
Residual NN	2,122	0.976	2107.720	45.910	0.044	24.486	14.410
ESN	69,121	0.843	13,768.036	117.337	0.113	73.917	52.860
TCN	2,176,257	0.981	1676.784	40.949	0.039	20.869	10.208
LSTM	8,529	0.976	2095.702	45.779	0.044	23.252	12.001
GRU	16,833	0.975	2211.138	47.023	0.045	19.759	8.931
RNN	5,537	0.978	1947.132	44.126	0.042	21.679	10.323
CNN-LSTM	135,361	**0.984**	**1382.722**	**37.185**	**0.036**	**15.090**	**6.932**
ANN	2,561	0.970	2663.559	51.610	0.050	29.851	16.256
CNN	39,945	0.982	1542.367	39.273	0.038	22.112	15.339
MLP	**701**	0.946	4777.664	69.121	0.066	38.351	24.398

Bold indicates the best results

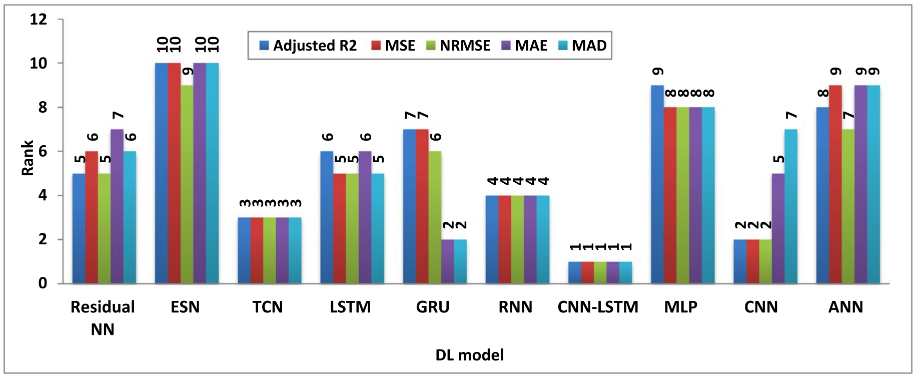


Fig. 25 The rank of each DL model using adjusted R^2 score, MSE, NRMSE, MAE and MAD metrics

more closely aligned with the actual data than other comparing models. Also, CNN-LSTM obtains the minimum values for $NRMSE = 0.036$ and $MSE = 1265.721$, and the second minimum values for $MAE = 16.461$ and $MAD = 8.498$. The CNN achieved the second ranking compared to its counterparts with *Adjusted* R^2 *score* of 0.982. On the other side, TCN comes in third rank with *Adjusted* R^2 *score* = 0.981 and has the largest number of parameters with value of 2,176,257. Moreover, RNN comes in the fourth rank with *Adjusted* R^2 *score* = 0.978, followed by LSTM, Residual NN, GRU, ANN, MLP and ESN. The number of parameters affects the model's complexity. MLP has the least number of parameters equal to 701.

Figure 25 displays the rank of each model using five performance metrics, including the *Adjusted* R^2 *score*, *MSE*, *NRMSE*, *MAE*, and *MAD*. It can be seen that the CNN-LSTM model achieves the best results compared to their rivals. Furthermore, ESN has the worst performance. Figure 26 shows the sum of ranks over the previous five metrics to give a general view on the model ranking for solar radiance forecasting. The rank of DL models from the best to the worst comes as follows: CNN-LSTM, TCN, CNN, RNN, GRU, LSTM, Residual NN, MLP, ANN and ESN.

Figure 27 describes the hybrid CNN-LSTM model for predicting solar irradiance. The figure depicts that the model contains two convolution layers, one dropout layer, one pooling layer, two LSTM layers and one dense layer. Presentation of

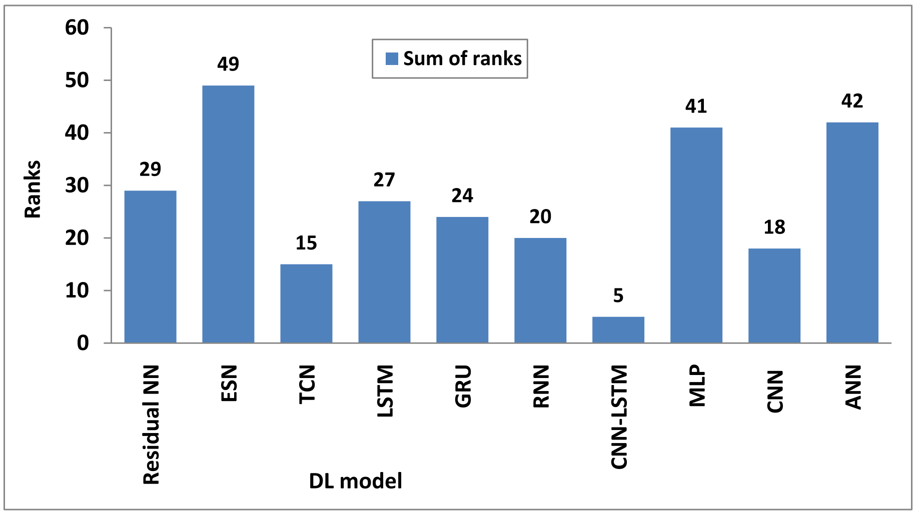
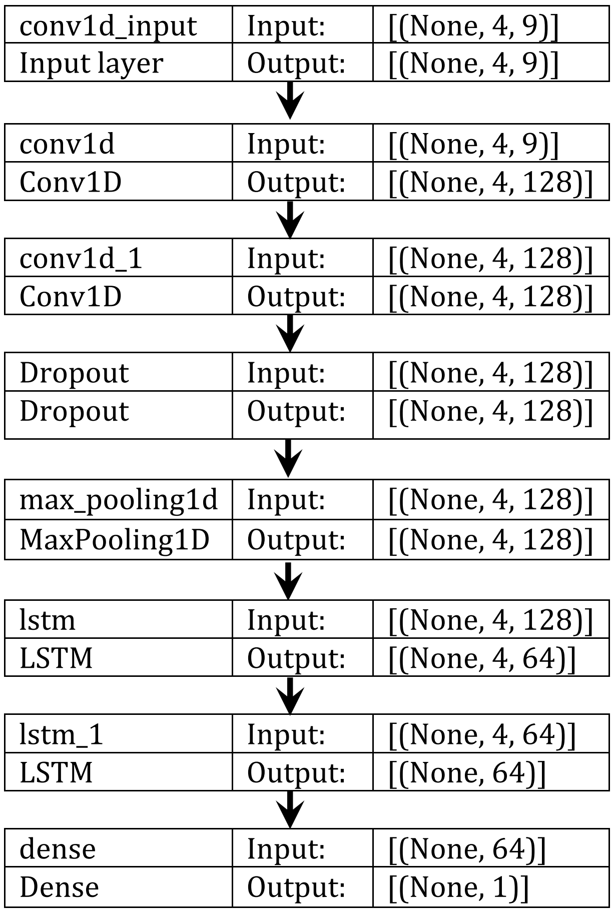


Fig. 26 The sum of ranks of each DL model



conv1d_input	Input:	[(None, 4, 9)]
Input layer	Output:	[(None, 4, 9)]
conv1d	Input:	[(None, 4, 9)]
Conv1D	Output:	[(None, 4, 128)]
conv1d_1	Input:	[(None, 4, 128)]
Conv1D	Output:	[(None, 4, 128)]
Dropout	Input:	[(None, 4, 128)]
Dropout	Output:	[(None, 4, 128)]
max_pooling1d	Input:	[(None, 4, 128)]
MaxPooling1D	Output:	[(None, 4, 128)]
lstm	Input:	[(None, 4, 128)]
LSTM	Output:	[(None, 4, 64)]
lstm_1	Input:	[(None, 4, 64)]
LSTM	Output:	[(None, 64)]
dense	Input:	[(None, 64)]
Dense	Output:	[(None, 1)]

Fig. 27 CNN-LSTM model summary

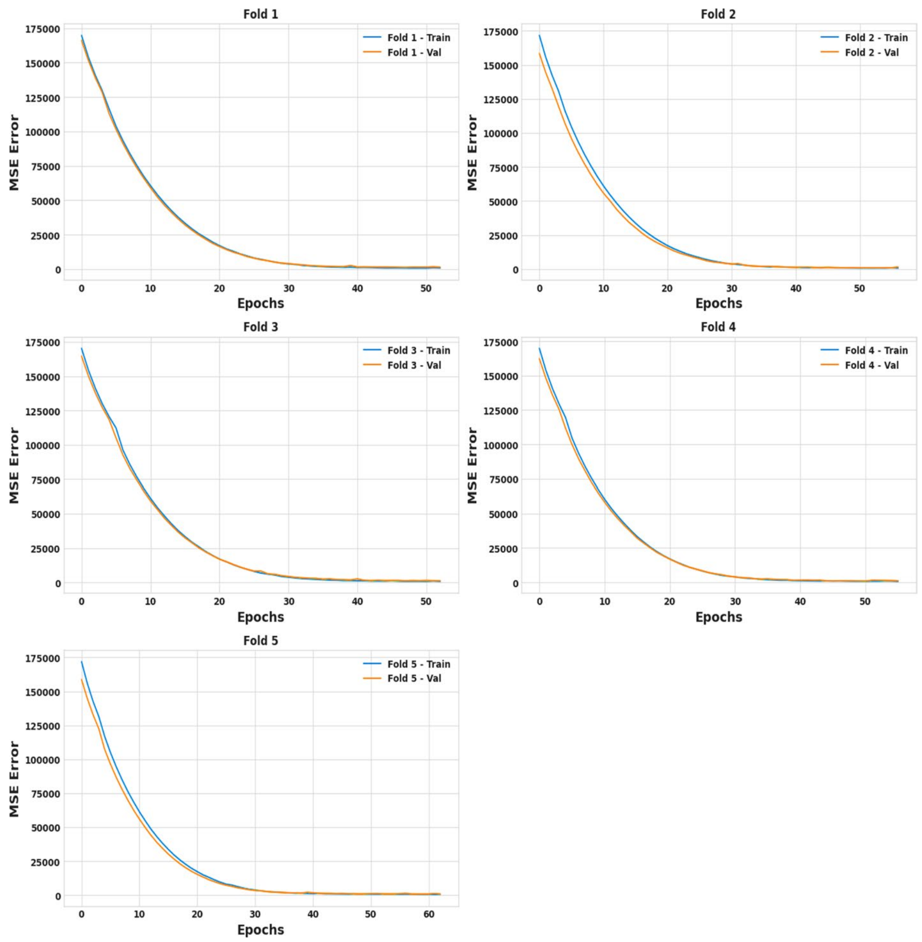


Fig. 28 The loss curve for training and validation set for all five folds of CNN-LSTM model

the CNN-LSTM model's training and validation loss curves can be found in Fig. 28. According to the figure, the CNN-LSTM has varying numbers of epochs for the different folds due to the implementation of the early stopping parameter. This model terminated the training process if the loss in both training and validation did not change during model fitting. It is clear that the loss values of the training set and the validation set for all folds are in good agreement with one another. This suggests that the predictions made by the proposed model are in line with the actual data, and that there is neither overfitting nor underfitting present. Figure 29 depicts the predictions of CNN-LSTM model compared to the actual ones for the first 100 time steps.

Results for machine learning algorithms

This section examines the performance of several ML models, such as linear regression, SGD regression, LASSO, gradient boosting regression, random forest, decision tree regression, and KNN regression. Table 7 contains the best parameters values of different parameters associated with each ML model that are found using the grid search cross validation technique. By inspecting the table, we display the various values of each parameter for all ML models. Also, the best hyperparameters values are recorded. We

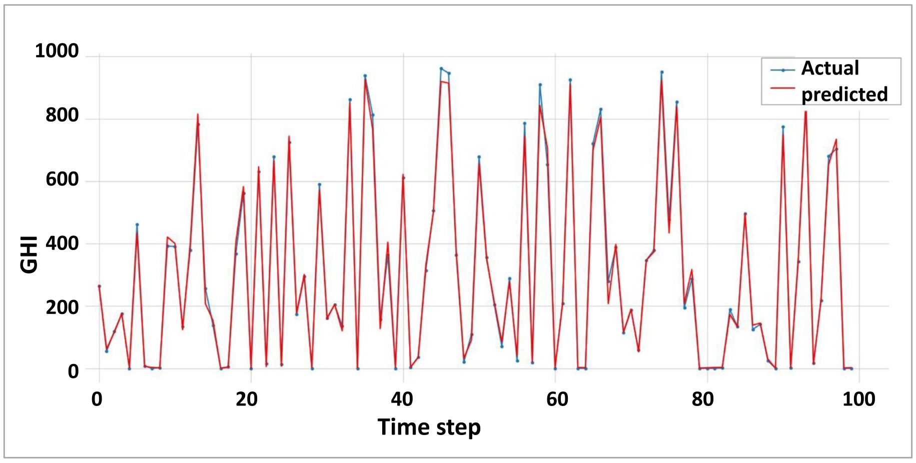


Fig. 29 CN-LSTM predictions

used Sci-kit API of version 1.2.2 which is a Python framework for ML to implement all our selected ML models.

Table 8 presents a comparison among numerous ML algorithms, including linear regression, SGD regression, LASSO, random forest, gradient boosting regression, decision tree regression and KNN regression. The table shows that the gradient boosting regression gets the best results for most of the performance metrics. Gradient boosting regression has the highest *Adjusted R^2 score* with a value of 0.962, which puts it at the top of the algorithms. It also obtains the minimum values for MSE = 3415.02, NRMSE = 0.058 and MAE = 29.33. KKN regression attains the second highest value of the *Adjusted R^2 score*, which is 0.945. Moreover, linear regression comes in the third rank with an *Adjusted R^2 score* value of 0.893.

Figure 30 displays the ranking of the existing seven ML approaches according to five distinct evaluations: Adjusted R^2 score, MSE, NRMSE, MAE, and MAD. We can observe that gradient boosting regressions attains the first ranking in five metrics, while SGD shows the worst performance. Also, Fig. 31 displays the ranking of several algorithms based on the total sum of rankings for all performance metrics. A lower sum of ranks indicates that the ML model performs better. The gradient boosting regression has a minimum value of the sum of ranks with a value of 5. Then, KNN comes in at the next rank after gradient boosting regression. SGD reserves the last rank with a value of 26, which demonstrates poor performance. Moreover, Fig. 32 depicts the predictions obtained by gradient boosting regression for the first 100 time steps, which has proven that the predicted values of gradient boosting regression are closely aligned with the actual values.

To explain the previous results obtained by ML models and the outperformance of gradient boosting regression, XAI with SHAP and LIME are introduced. Mean absolute SHAP values are often shown as bar plots that order features according to their importance, as seen in Fig. 33. The ordering of features and the respective magnitudes of the mean absolute SHAP values are the most important factors to consider. Here, we can see that DHI is the most influential feature contributing to the GHI output, whereas the least informative feature contributing to the GHI output is WS_gust.

Table 7 The best hyperparameters values found by grid search cross validation of each ML model

Algorithm name	Parameter name	Parameter values	Best parameter value
Linear regression	Intercept	[True, False]	True
	Copy_X	[True, False]	True
	N_jobs	[−1, 1]	−1
	Weights (β)	[134.77742582, 135.5658808, 32.3798911, −13.60180839, 14.06928621, −18.5702262, −1.5354856, 46.40629438, 11.85316545]	
SGD regression	Alpha Random state Max iter Tol	[0.1, 0.001, 1] [33] [1000, 2000] [1e-3, 1e-4]	0.1 33 1000 1e-3
LASSO	Alpha Random state Max iter	[0.1, 1.0, 10.0] 33 [1000, 2000]	0.1 33 1000
	Weights	[134.78748341, 135.60866578, 32.32577104, −13.64606087, 12.66907409, −17.07588968, −1.32131259, 46.09668833, 11.72830064]	
Random forest regression	Max depth of tree Number of estimators Random state Max Features Bootstrap	[2, 3] [50, 100] 33 [1.0, 'sqrt', 'log2'] [True, False]	3 100 33 1.0 True
Gradient boosting regression	Max depth of tree Number of estimators Random state Learning rate	[2, 3] [50, 100] 33 [0.1, 0.5, 1.5]	3 100 33 0.5
Decision tree regression	Max depth of tree Random state Criterion	[2, 3, 5] 33 ['squared_error', 'friedman_mse', 'absolute_error']	5 33 squared_error
KNN regression	K-neighbors Weights Algorithm	[3, 5, 7] ['uniform', 'distance'] ['auto', 'ball_tree', 'kd_tree', 'brute']	7 distance auto
MLP Regressor	Activation Solver Alpha Hidden layers	['relu', 'tanh'] ['adam', 'sgd'] [0.0001, 0.001, 0.01, 0.1] [(50), (100), (200), (200, 100), (200, 100, 50)]	relu adam 0.0001 (200, 100, 50)

Table 8 The results of ML algorithms

Model	Adjusted R^2 Score	MSE	RMSE	NRMSE	MAE	MAD
Linear regression	0.810	16,891.74	129.968	0.129	78.57	51.972
SGD regression	0.810	16,926.44	130.102	0.129	79.41	52.277
LASSO	0.810	16,894.36	129.978	0.129	78.53	51.934
Random forest	0.810	16,894.36	129.978	0.129	78.53	51.934
Gradient boosting regression	**0.962**	**3415.02**	**58.438**	**0.058**	**29.33**	**15.626**
Decision tree regression	0.893	9543.61	97.691	0.097	56.25	28.930
KNN regression	0.945	4907.74	70.055	0.069	39.04	20.329

Bold font indicates the best results

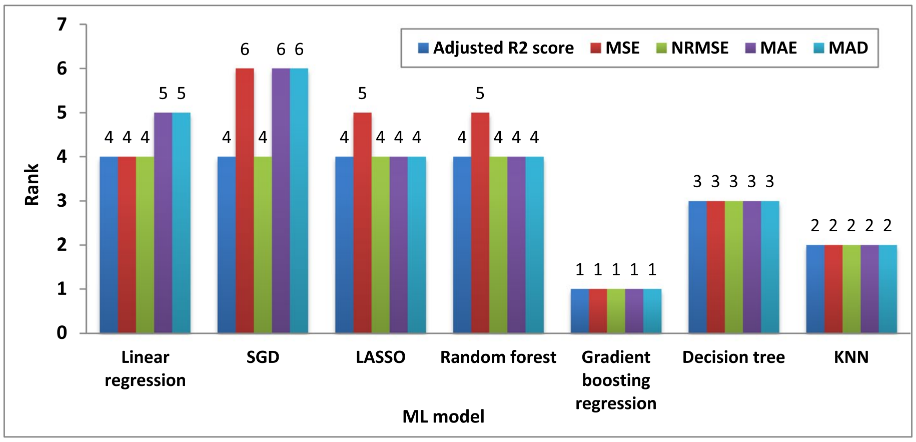


Fig. 30 The rank of each ML model using *Adjusted R*^{2} *score*, *MSE*, *NRMSE*, *MAE* and *MAD* metrics

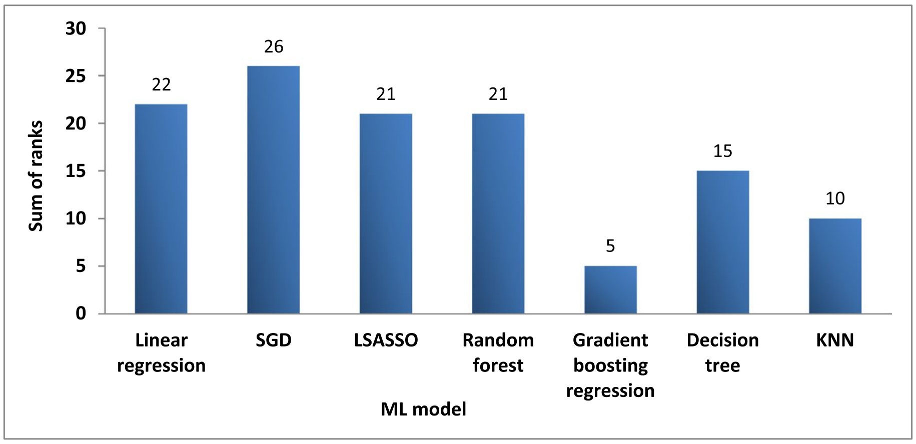


Fig. 31 The sum of ranks of each ML model

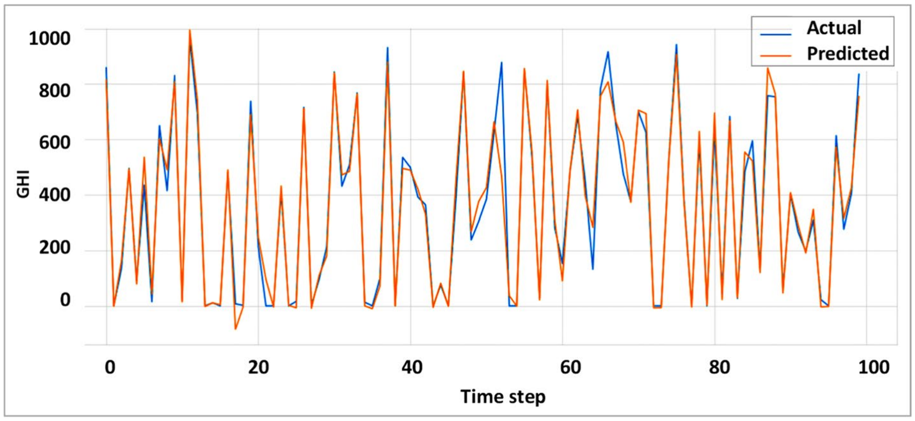


Fig. 32 Gradient boosting regression predictions

The bar plot in Fig. 33 did not tell us how the underlying values of each feature relate to the model's predictions. Figure 34 illustrates how the features contribute to the model's predictions. Each feature of the data is represented by a row in the plot.

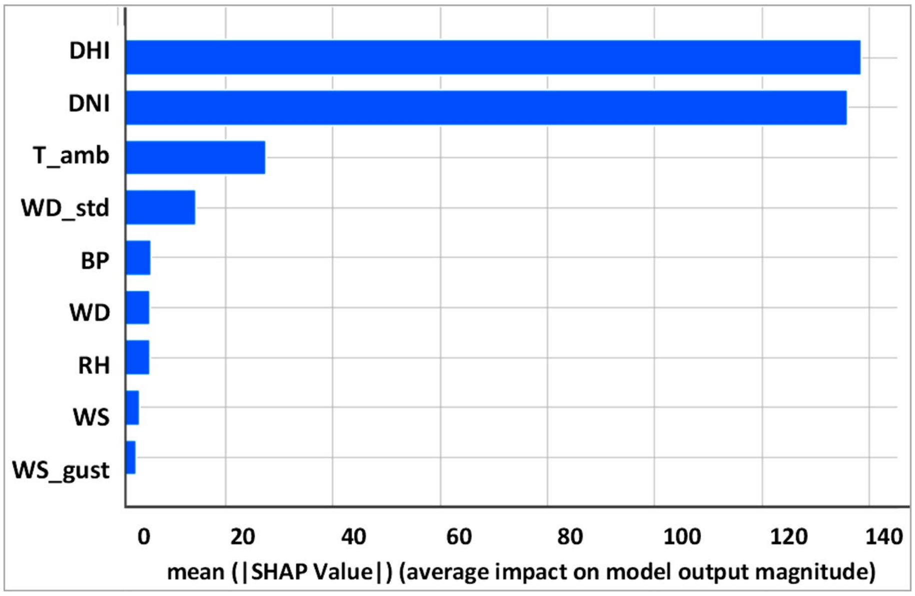


Fig. 33 Mean absolute values of the SHAP for all features

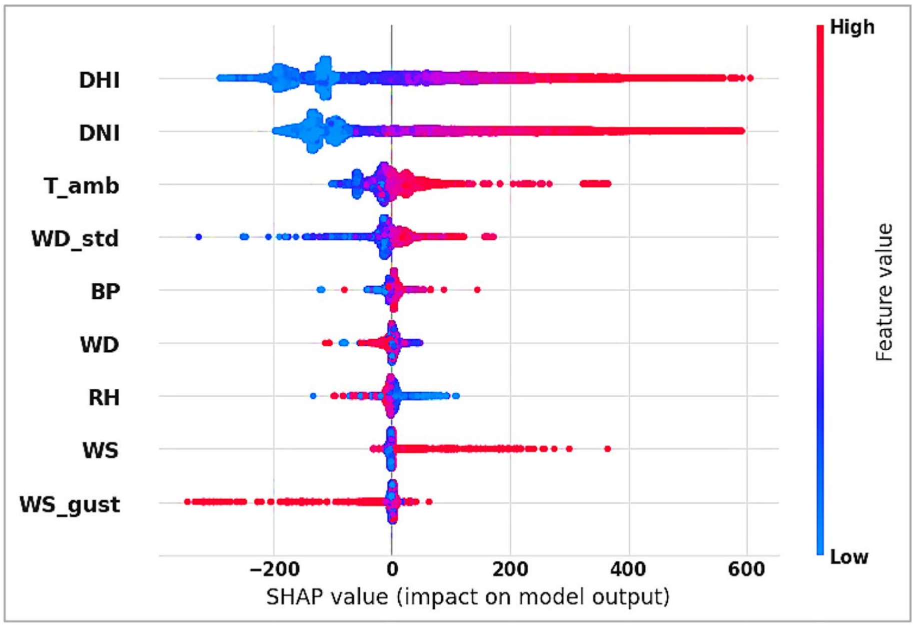


Fig. 34 Summary of beeswarm plot ranked by mean absolute SHAP value

The horizontal x-axis indicates the SHAP values. The blue color refers to lower values of the feature, whereas the red color refers to higher values of this feature. The feature that has a wide spread of SHAP values, has a greater contribution to the model's predictions. The feature values that are assembled around zero have a minimal impact on the model's predictions. From the plot, we can see that the lower values of DHI have negative SHAP values, whereas the higher values of DHI have positive SHAP values. Also, DHI has a wide spread of SHAP values, which reflects the significant effect on

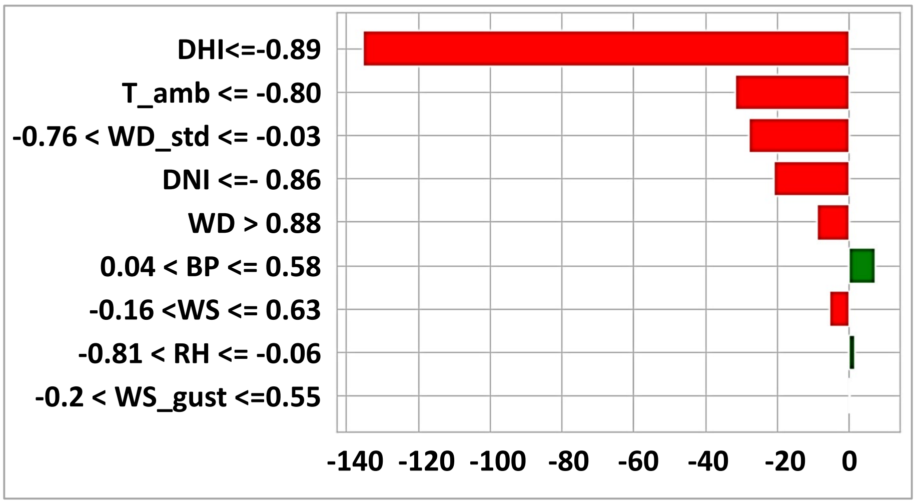


Fig. 35 LIME plot

the model's predictions. LIME plot in Fig. 35 shows that DHI and DNI have the most impact for output.

Conclusions and future works

Various DL and ML algorithms were employed for SIF. This paper presents a comparison for the most common nine DL algorithms, such as ANN, CNN, LSTM, GRU, RNN, TCN, ESN, CNN-LSTM MLP and residual NN. Another experiment is conducted to compare among seven of the existing algorithms, containing linear regression, SGD regression, LASSO, random forest, decision tree regression and KNN regression. The GSCV is employed to find the best hyperparameter values for all DL and ML models. The dataset was gathered from Islamabad over five years at hourly intervals using accurate meteorological instruments. Moreover, we measure the effectiveness of each model using various metrics like MSE, *Adjusted* R^2 *score*, RMSE, MAE and MAD. Additionally, SHAP and LIME are used in our study to provide an explanation and comprehension for the acquired result of the best model. For deep learning algorithms, the statistical analysis shows that CNN-LSTM outperforms its counterparts, whereas gradient boosting regression achieves a superior results compared to other ML models.

Hybridization can be a powerful tool, as hybrid models can achieve better results than individual ones. Hence, in the future, we hope to develop a hybrid model for solar irradiance forecasting. Hyperparameter tuning strategies seek to find the optimal values of parameters that lead to improved performance and the learning process of ML algorithms. Other hyperparameter tuning can be a future direction through the integration of metaheuristic techniques to optimize the parameter values [128, 129]. Possible future directions to enhance the work would revolve mainly around different time horizons: short-term, medium-term, and long-term, to study the effectiveness of the proposed model. This study is interested in solar irradiance forecasting, and we aim at investigating other challenging problems, such as wind forecasting [130], PV power forecasting [131] and price forecasting [132].

Author contributions

Doaa El-Shahat and Ahmed Tolba: Investigation, Methodology, Resources, Visualization, Software, Writing—original draft, Writing—review & editing. Mohamed Abouhawwash: Conceptualization, Methodology, Visualization, Software, Writing—review & editing. Mohamed Abdel-Basset: Conceptualization, Methodology, Resources, Visualization, Validation, Supervision, Writing—review & editing.

Funding

Open access funding provided by The Science, Technology & Innovation Funding Authority (STDF) in cooperation with The Egyptian Knowledge Bank (EKB).

Data availability

https://www.kaggle.com/datasets/ahmdtolba/solardata-islamabad/data. No datasets were generated or analysed during the current study.

Declarations

Ethics approval and consent to participate

The author confirms the sole responsibility for this manuscript. The author read and approved the final manuscript.

Competing interests

The authors declare no competing interests.

Received: 27 February 2024 Accepted: 26 August 2024
Published online: 18 September 2024

References

1. Steg L. Psychology of climate change. Annu Rev Psychol. 2023;74:391–421.
2. Eckardt NA, et al. Climate change challenges, plant science solutions. Plant Cell. 2023;35(1):24–66.
3. Mirón IJ, Linares C, Díaz J. The influence of climate change on food production and food safety. Environ Res. 2023;216: 114674.
4. Matthews T. Humid heat and climate change. Prog Phys Geog Earth Environ. 2018;42(3):391–405.
5. Lee, H., et al., Climate change 2023: synthesis report. Contribution of working groups I, II and III to the sixth assessment report of the intergovernmental panel on climate change. 2023.
6. Sain K. Climate change and fossil fuels: impacts, challenges and plausible mitigation. J Geol Soc India. 2023;99(4):454–8.
7. Ozdemir I, et al., COP28. 2023.
8. Nijsse FJ, et al. The momentum of the solar energy transition. Nat Commun. 2023;14(1):6542.
9. Nassar YF, et al. Carbon footprint and energy life cycle assessment of wind energy industry in Libya. Energy Convers Manage. 2024;300: 117846.
10. Kamran M. Hydro energy renewable energy conversion systems. Amsterdam: Elsevier; 2021.
11. Soltani M, et al. Environmental, economic, and social impacts of geothermal energy systems. Renew Sustain Energy Rev. 2021;140: 110750.
12. Obaideen K, et al. Solar energy: Applications, trends analysis, bibliometric analysis and research contribution to sustainable development goals (SDGs). Sustainability. 2023;15(2):1418.
13. Chinnasamy S, Arunachalam A. Experimental investigation on direct expansion solar-air source heat pump for water heating application. Renew Energy. 2023;202:222–33.
14. Madhankumar S, et al. A review on the latest developments in solar dryer technologies for food drying process. Sustain Energy Technol Assess. 2023;58: 103298.
15. Kurbonov K. Use of renewable energy sources for heating buildings. Open Access Repos. 2023;4(03):349–54.
16. Kumar SS, et al. Solar powered water pumping systems for irrigation: a comprehensive review on developments and prospects towards a green energy approach. Mater Today Proc. 2020;33:303–7.
17. Tuly S, et al. Effects of design and operational parameters on the performance of a solar distillation system: a comprehensive review. Groundw Sustain Dev. 2021;14: 100599.
18. Niyommaneerat W, Suwanteep K, Chavalparit O. Sustainability indicators to achieve a circular economy: a case study of renewable energy and plastic waste recycling corporate social responsibility (CSR) projects in Thailand. J Clean Prod. 2023;391: 136203.
19. Morey M, et al. A comprehensive review of grid-connected solar photovoltaic system: architecture, control, and ancillary services. Renew Energy Focus. 2023;45:307–30.
20. Guermoui M, et al. A comprehensive review of hybrid models for solar radiation forecasting. J Clean Prod. 2020;258: 120357.
21. Akhter MN, et al. Review on forecasting of photovoltaic power generation based on machine learning and metaheuristic techniques. IET Renew Power Gener. 2019;13(7):1009–23.
22. Bassous GF, Calili RF, Barbosa CH. Development of a low-cost data acquisition system for very short-term photovoltaic power forecasting. Energies. 2021;14(19):6075.
23. Nie Y, et al. Open-source sky image datasets for solar forecasting with deep learning: a comprehensive survey. Renew Sustain Energy Rev. 2024;189: 113977.
24. Morf H. A validation frame for deterministic solar irradiance forecasts. Renew Energy. 2021;180:1210–21.

25. Voyant C, et al. Machine learning methods for solar radiation forecasting: a review. Renew Energy. 2017;105:569–82.
26. Demir V, Citakoglu H. Forecasting of solar radiation using different machine learning approaches. Neural Comput Appl. 2023;35(1):887–906.
27. Ajith M, Martínez-Ramón M. Deep learning algorithms for very short term solar irradiance forecasting: a survey. Renew Sustain Energy Rev. 2023;182: 113362.
28. Kumari P, Toshniwal D. Deep learning models for solar irradiance forecasting: a comprehensive review. J Clean Prod. 2021;318: 128566.
29. Liu J, et al. Hourly stepwise forecasting for solar irradiance using integrated hybrid models CNN-LSTM-MLP combined with error correction and VMD. Energy Convers Manage. 2023;280: 116804.
30. Lu Y, et al. Predicting surface solar radiation using a hybrid radiative transfer-machine learning model. Renew Sustain Energy Rev. 2023;173: 113105.
31. Lara-Benítez P, et al. Short-term solar irradiance forecasting in streaming with deep learning. Neurocomputing. 2023;546: 126312.
32. Ferkous K, et al. A novel learning approach for short-term photovoltaic power forecasting-a review and case studies. Eng Appl Artif Intell. 2024;133: 108502.
33. Feng J, Wang W, Li J. An LM-BP neural network approach to estimate monthly-mean daily global solar radiation using MODIS atmospheric products. Energies. 2018;11(12):3510.
34. Praynlin, E. and J.I. Jensona. Solar radiation forecasting using artificial neural network. In 2017 Innovations in Power and Advanced Computing Technologies (i-PACT). 2017. IEEE.
35. Pazikadin AR, et al. Solar irradiance measurement instrumentation and power solar generation forecasting based on Artificial Neural Networks (ANN): a review of five years research trend. Sci Total Environ. 2020;715: 136848.
36. Haider SA, et al. Deep learning and statistical methods for short-and long-term solar irradiance forecasting for Islamabad. Renew Energy. 2022;198:51–60.
37. Sivaneasan B, Yu C, Goh K. Solar forecasting using ANN with fuzzy logic pre-processing. Energy Proc. 2017;143:727–32.
38. Kumar KR, Kalavathi MS. Artificial intelligence based forecast models for predicting solar power generation. Mater Today Proc. 2018;5(1):796–802.
39. Lima M, et al. MLP back propagation artificial neural network for solar resource forecasting in equatorial areas. Renew Energy Power Qual J (RE&PQJ). 2018;1:175–80.
40. Laopaiboon, T., et al. Hour-ahead solar forecasting program using back propagation artificial neural network. in 2018 international conference and utility exhibition on green energy for sustainable development (ICUE). 2018. IEEE.
41. Ledmaoui Y, et al. Forecasting solar energy production: a comparative study of machine learning algorithms. Energy Rep. 2023;10:1004–12.
42. Gairaa K, et al. Contribution of ordinal variables to short-term global solar irradiation forecasting for sites with low variabilities. Renew Energy. 2022;183:890–902.
43. Alzahrani A, et al. Solar irradiance forecasting using deep neural networks. Proc Comput Sci. 2017;114:304–13.
44. Qing X, Niu Y. Hourly day-ahead solar irradiance prediction using weather forecasts by LSTM. Energy. 2018;148:461–8.
45. Ashfaq Q, et al. Hour-ahead global horizontal irradiance forecasting using long short term memory network. In 2020 IEEE 23rd International Multitopic Conference (INMIC). 2020. IEEE.
46. Lee D, Kim K. Recurrent neural network-based hourly prediction of photovoltaic power output using meteorological information. Energies. 2019;12(2):215.
47. Michael NE, et al. Short-term solar irradiance forecasting based on a novel Bayesian optimized deep Long Short-Term Memory neural network. Appl Energy. 2022;324: 119727.
48. Eşlik AH, Akarslan E, Hocaoğlu FO. Short-term solar radiation forecasting with a novel image processing-based deep learning approach. Renew Energy. 2022;200:1490–505.
49. Bae KY, Jang HS, Sung DK. Hourly solar irradiance prediction based on support vector machine and its error analysis. IEEE Trans Power Syst. 2016;32(2):935–45.
50. Huertas Tato J, Centeno Brito M. Using smart persistence and random forests to predict photovoltaic energy production. Energies. 2018;12(1):100.
51. Chen C-R, Three Kartini U. K-nearest neighbor neural network models for very short-term global solar irradiance forecasting based on meteorological data. Energies. 2017;10(2):186.
52. Kong X, et al. Multi-step short-term solar radiation prediction based on empirical mode decomposition and gated recurrent unit optimized via an attention mechanism. Energy. 2023;282: 128825.
53. Seepanomwan, K. Better Multi-step Time Series Prediction Using Sparse and Deep Echo State Network. In 2023 8th International Conference on Control and Robotics Engineering (ICCRE). 2023. IEEE.
54. Song, Z. and L.E. Brown. Multi-dimensional evaluation of temporal neural networks on solar irradiance forecasting. In 2019 IEEE Innovative Smart Grid Technologies-Asia (ISGT Asia). 2019. IEEE.
55. Azizi N, et al. Deep learning based long-term global solar irradiance and temperature forecasting using time series with multi-step multivariate output. Renew Energy. 2023;206:135–47.
56. Wang Y, et al. Adaptive learning hybrid model for solar intensity forecasting. IEEE Trans Industr Inf. 2018;14(4):1635–45.
57. Zang H, et al. Short-term global horizontal irradiance forecasting based on a hybrid CNN-LSTM model with spatiotemporal correlations. Renew Energy. 2020;160:26–41.
58. Kumari P, Toshniwal D. Long short term memory–convolutional neural network based deep hybrid approach for solar irradiance forecasting. Appl Energy. 2021;295: 117061.
59. Gao B, et al. Hourly forecasting of solar irradiance based on CEEMDAN and multi-strategy CNN-LSTM neural networks. Renew Energy. 2020;162:1665–83.
60. Huang X, et al. Hybrid deep neural model for hourly solar irradiance forecasting. Renew Energy. 2021;171:1041–60.

61. Malakouti SM, et al. Predicting wind power generation using machine learning and CNN-LSTM approaches. Wind Eng. 2022;46(6):1853–69.
62. Guermoui M, et al. Enhancing direct normal solar irradiation forecasting for heliostat field applications through a novel hybrid model. Energy Convers Manage. 2024;304: 118189.
63. Guermoui M, Boland J, Rabehi A. On the use of BRL model for daily and hourly solar radiation components assessment in a semiarid climate. Euro Phys J Plus. 2020;135(2):1–16.
64. Zhuhadar LP, Lytras MD. The application of AutoML techniques in diabetes diagnosis: current approaches, performance, and future directions. Sustainability. 2023;15(18):13484.
65. Abiodun OI, et al. State-of-the-art in artificial neural network applications: a survey. Heliyon. 2018. https://doi.org/10.1016/j.heliyon.2018.e00938.
66. Mfetoum IM, et al. A multilayer perceptron neural network approach for optimizing solar irradiance forecasting in Central Africa with meteorological insights. Sci Rep. 2024;14(1):3572.
67. Albawi, S., T.A. Mohammed, and S. Al-Zawi. Understanding of a convolutional neural network. In 2017 international conference on engineering and technology (ICET). 2017. IEEE.
68. Kasongo SM. A deep learning technique for intrusion detection system using a recurrent neural networks based framework. Comput Commun. 2023;199:113–25.
69. Staudemeyer RC, ER Morris, 2019 Understanding LSTM--a tutorial into long short-term memory recurrent neural networks. arXiv preprint arXiv:1909.09586.
70. Tandale SB, Stoffel M. Recurrent and convolutional neural networks in structural dynamics: a modified attention steered encoder–decoder architecture versus LSTM versus GRU versus TCN topologies to predict the response of shock wave-loaded plates. Comput Mech. 2023. https://doi.org/10.1007/s00466-023-02317-8.
71. Zarzycki K, Ławryńczuk M. Advanced predictive control for GRU and LSTM networks. Inf Sci. 2022;616:229–54.
72. Yao X, et al. Echo state network with multiple delayed outputs for multiple delayed time series prediction. J Franklin Inst. 2022;359(18):11089–107.
73. Li H. Short-term wind power prediction via spatial temporal analysis and deep residual networks. Front Energy Res. 2022;10: 920407.
74. Ghimire S, et al. Deep learning CNN-LSTM-MLP hybrid fusion model for feature optimizations and daily solar radiation prediction. Measurement. 2022;202: 111759.
75. Abraham A. Artificial neural networks handbook of measuring system design. Hoboken: Wiley; 2005.
76. Yegnanarayana B. Artificial neural networks. PHI Learning Pvt. Ltd.: Delhi; 2009.
77. Vaka M, et al. A review on Malaysia's solar energy pathway towards carbon-neutral Malaysia beyond Covid'19 pandemic. J Clean Prod. 2020;273: 122834.
78. Gu J, et al. Recent advances in convolutional neural networks. Pattern Recogn. 2018;77:354–77.
79. Medsker LR, Jain L. Recurrent neural networks. Des Appl. 2001;5(64–67):2.
80. Wang J, et al. NGCU: a new RNN model for time-series data prediction. Big Data Res. 2022;27: 100296.
81. Uddin MZ, et al. A body sensor data fusion and deep recurrent neural network-based behavior recognition approach for robust healthcare. Inform Fusion. 2020;55:105–15.
82. Bani-Almarjeh M, Kurdy M-B. Arabic abstractive text summarization using RNN-based and transformer-based architectures. Inf Process Manage. 2023;60(2): 103227.
83. Zhang W, et al. Prediction high frequency parameters based on neural network. IOP Conf Ser Mater Sci Eng. 2019. https://doi.org/10.1088/1757-899X/631/5/052035.
84. Lee K, Ray J, Safta C. The predictive skill of convolutional neural networks models for disease forecasting. PLoS ONE. 2021;16(7): e0254319.
85. He Y, Zhao J. Temporal convolutional networks for anomaly detection in time series. J Phys Conf Ser. 2019. https://doi.org/10.1088/1742-6596/1213/4/042050.
86. Ji Q, et al. Short-term prediction of the significant wave height and average wave period based on VMD-TCN-LSTM algorithm. EGUsphere. 2023;2023:1–27.
87. Sarwinda D, et al. Deep learning in image classification using residual network (ResNet) variants for detection of colorectal cancer. Proc Comput Sci. 2021;179:423–31.
88. Ronald M, Poulose A, Han DS. iSPLInception: an inception-ResNet deep learning architecture for human activity recognition. IEEE Access. 2021;9:68985–9001.
89. Jaeger H. Echo state network. Scholarpedia. 2007;2(9):2330.
90. Gallicchio C, Micheli A, Pedrelli L. Design of deep echo state networks. Neural Netw. 2018;108:33–47.
91. Zheng K, et al. Long-short term echo state network for time series prediction. IEEE Access. 2020;8:91961–74.
92. Alaeddine H, Jihene M. Deep residual network in network. Comput Intell Neurosci. 2021;2021:1–9.
93. Livieris IE, Pintelas E, Pintelas P. A CNN–LSTM model for gold price time-series forecasting. Neural Comput Appl. 2020;32:17351–60.
94. Rajagukguk RA, Ramadhan RA, Lee H-J. A review on deep learning models for forecasting time series data of solar irradiance and photovoltaic power. Energies. 2020;13(24):6623.
95. James G, et al. An introduction to statistical learning: with applications in python. Cham: Springer; 2023.
96. Alrababeh NM, BaniMustafa AM. Regression for predicting effort in object-oriented software projects. SSRN Elect J. 2022. https://doi.org/10.2139/ssrn.4141236.
97. Li Y, Lu F, Yin Y. Applying logistic LASSO regression for the diagnosis of atypical Crohn's disease. Sci Rep. 2022;12(1):11340.
98. Borup D, et al. Targeting predictors in random forest regression. Int J Forecast. 2023;39(2):841–68.
99. Velthoen J, et al. Gradient boosting for extreme quantile regression. Extremes. 2023;26(4):639–67.
100. Rathore SS, Kumar S. A decision tree regression based approach for the number of software faults prediction. ACM SIGSOFT Soft Eng Notes. 2016;41(1):1–6.
101. Goyal R, Chandra P, Singh Y. Suitability of KNN regression in the development of interaction based software fault prediction models. Ieri Procedia. 2014;6:15–21.

102. James G, et al. Linear regression. In: James G, Witten D, Hastie T, Tibshirani R, Taylor J, editors., et al., An introduction to statistical learning With applications in python. Cham: Springer; 2023. p. 69–134.
103. Groß J. Linear regression. Berlin: Springer, Berlin Heidelberg; 2003.
104. Jastrzębski, S., et al., Three factors influencing minima in sgd. arXiv preprint, 2017.
105. Ranstam J, Cook J. LASSO regression. J Br Surgery. 2018;105(10):1348–1348.
106. Chatterjee T, Chowdhury R. Improved sparse approximation models for stochastic computations in Handbook of neural computation. Amsterdam: Elsevier; 2017.
107. Tang N, et al. Solar power generation forecasting with a LASSO-based approach. IEEE Internet Things J. 2018;5(2):1090–9.
108. Rigatti SJ. Random forest. J Insur Med. 2017;47(1):31–9.
109. Babar B, et al. Random forest regression for improved mapping of solar irradiance at high latitudes. Sol Energy. 2020;198:81–92.
110. Munshi, A. and R. Moharil. Solar radiation forecasting using random forest. In AIP Conference Proceedings. 2022. AIP Publishing.
111. Villegas-Mier CG, et al. Optimized random forest for solar radiation prediction using sunshine hours. Micromachines. 2022;13(9):1406.
112. Zemel R, Pitassi T. A gradient-based boosting algorithm for regression problems. Adv Neu Inform Proc Syst. 2000. https://doi.org/10.3389/fnbot.2013.00021.
113. Voyant C, et al. Prediction intervals for global solar irradiation forecasting using regression trees methods. Rene Energy. 2018;126:332–40.
114. Lundberg SM, Lee S-I. A unified approach to interpreting model predictions. Adv Neu Inform Proc Syst. 2017. https://doi.org/10.1093/bioadv/vbad016.
115. Zafar MR, Khan N. Deterministic local interpretable model-agnostic explanations for stable explainability. Machi Learn Knowled Ext. 2021;3(3):525–41.
116. Benjamini Y. Opening the box of a boxplot. Am Stat. 1988;42(4):257–62.
117. Ding, R., et al., Evaluation of landslide susceptibility in mountainous areas of Changji city at the northern foot of Tianshan Mountain based on coupled model of weight of evidence and Shanon's entropy. 2022.
118. Mellit A, Pavan AM, Lughi V. Deep learning neural networks for short-term photovoltaic power forecasting. Renew Energy. 2021;172:276–88.
119. Phan Q-T, et al. A novel forecasting model for solar power generation by a deep learning framework with data preprocessing and postprocessing. IEEE Trans Ind Appl. 2022;59(1):220–31.
120. Boubaker S, et al. Deep neural networks for predicting solar radiation at Hail Region. Saudi Arabia Ieee Access. 2021;9:36719–29.
121. Chadha A, Kaushik B. A hybrid deep learning model using grid search and cross-validation for effective classification and prediction of suicidal ideation from social network data. N Gener Comput. 2022;40(4):889–914.
122. Malakouti SM, Menhaj MB, Suratgar AA. The usage of 10-fold cross-validation and grid search to enhance ML methods performance in solar farm power generation prediction. Cleaner Eng Technol. 2023;15: 100664.
123. Bates S, Hastie T, Tibshirani R. Cross-validation: what does it estimate and how well does it do it? J Am Stat Assoc. 2024;119(546):1434–45.
124. Berrar D. Cross-validation. Amsterdam: Elsevier; 2019.
125. Satria, A., O.S. Sitompul, and H. Mawengkang. 5-Fold cross validation on supporting k-nearest neighbour accuration of making consimilar symptoms disease classification. In 2021 International Conference on Computer Science and Engineering (IC2SE). 2021. IEEE.
126. Yadav, S. and S. Shukla. Analysis of k-fold cross-validation over hold-out validation on colossal datasets for quality classification. In 2016 IEEE 6th International conference on advanced computing (IACC). 2016. IEEE.
127. Hodson TO. Root-mean-square error (RMSE) or mean absolute error (MAE): when to use them or not. Geosci Model Dev. 2022;15(14):5481–7.
128. Dong Y, et al. Predicting dissolved oxygen level using Young's double-slit experiment optimizer-based weighting model. J Environ Manage. 2024;351: 119807.
129. Vaisakh T, Jayabarathi R. Analysis on intelligent machine learning enabled with meta-heuristic algorithms for solar irradiance prediction. Evol Intel. 2022;15(1):235–54.
130. Wang H-Z, et al. Deep learning based ensemble approach for probabilistic wind power forecasting. Appl Energy. 2017;188:56–70.
131. Khelifi R, et al. Short-Term pv power forecasting using a hybrid TVF-EMD-ELM strategy. Int Trans Elect Energy Syst. 2023;2023(1):6413716.
132. Zhao Y, Li J, Yu L. A deep learning ensemble approach for crude oil price forecasting. Energy Econ. 2017;66:9–16.