Analysis and Prediction of Radiation in Photovoltaic Systems Using Machine Learning

D. Parameswari[1], R. Geetha[2], Anitha Govindaram[3], P. Thilagavathi[4], P. Vanitha[5] and Jose Anand A.[6]

[1]Department of Artificial Intelligence and Machine Learning, Jerusalem College of Engineering, Pallikaranai, Chennai, Tamil Nadu, India
[2]Department of Computing Technologies, SRM Institute of Science and Technology, Chennai, Tamil Nadu, India
[3]Department of Computer Science and Engineering, Saveetha School of Engineering, Saveetha Institute of Medical and Technical Sciences (SIMATS) Thandalam, Chennai, Tamil Nadu, India
[4]Department of Computer Science and Engineering, Aarupadai Veedu Institute of Technology, Vinayaka Mission's Research Foundation (DU), Chennai, Tamil Nadu, India
[5]Department of Electrical and Electronics Engineering, Sri Muthukumaran Institute of Technology, Kanchipuram, Tamil Nadu, India
[6]Department of ECE, KCG College of Technology, Karapakkam, Chennai, Tamil Nadu, India

Correspondence to:
R. Geetha,
geethar1@srmist.edu.in

Additional material is published online only. To view please visit the journal online.

Cite this as: Parameswari D, Geetha R, Govindaram A, Thilagavathi P, Vanitha P and Anand JA. Analysis and Prediction of Radiation in Photovoltaic Systems Using

ABSTRACT

Accurate solar radiation forecasting is essential for optimising photovoltaic (PV) energy generation and grid integration. This paper presents an LSTM-based forecasting framework enhanced with physically meaningful features and rigorous evaluation protocols. Unlike prior approaches, we prevent temporal data leakage by applying a strictly chronological data split (last 20% reserved for testing) and rolling-window cross-validation. The dataset comprises three years (2020–2023) of high-resolution solar radiation and meteorological data, supplemented by derived features such as solar zenith angle, clear-sky index, and cloud cover. Benchmark models—including persistence, clear-sky, and linear regression baselines—are evaluated alongside the proposed LSTM using standard metrics (MAE, RMSE, MAPE, and skill score) with statistical significance testing. Results confirm that the enhanced LSTM outperforms benchmarks, reducing RMSE by 14% and achieving a skill score of 0.68. Furthermore, nighttime irradiance predictions are eliminated through day/night masking. These findings demonstrate that a carefully validated, feature-rich LSTM framework can significantly improve solar forecasting reliability while ensuring reproducibility and transparency.

Keywords: Solar radiation forecasting, Photovoltaic power plant modeling, Long short-term memory networks, Bayesian hyperparameter optimization, Night-time irradiance prediction

Introduction

Solar energy, as one of the most promising renewable energy sources, has gained significant attention due to its environmental benefits and sustainability. However, despite its potential, the efficiency of photovoltaic plants heavily depends on the ability to predict and manage solar radiation (SR), which directly influences the plant's energy output. The accurate prediction of SR plays a crucial role in maximizing the performance of photovoltaic plants, ensuring effective energy management, and improving the reliability of power generation.[1–3]

In recent years, machine learning (ML) techniques have been increasingly adopted to address the challenges of SR forecasting.[4] Among these techniques, Long Short-Term Memory (LSTM) neural networks (NNs), a type of recurrent NN (RNN), have gained prominence due to their ability to capture temporal dependencies and patterns in sequential data. LSTMs are particularly suitable for time-series forecasting tasks, as they can effectively handle long-term dependencies and learn from historical data, making them ideal for predicting SR, which exhibits both daily and seasonal patterns.[5–8]

This work aims to develop and implement an LSTM-based NN model to predict SR in a photovoltaic plant. The goal is to create a robust prediction model that can accurately estimate future SR based on historical data, helping optimize the energy production process. The study leverages historical SR data collected from sensors and devices installed in a photovoltaic plant. The software, used for monitoring and recording real-time data from the photovoltaic plant, provides a rich dataset of relevant variables such as SR, temperature, and humidity.[9–11]

The model developed in this work will not only enhance the efficiency of photovoltaic plants by improving their energy production management but also contribute to the growing body of research on ML applications in renewable energy. By focusing on LSTM networks, this work explores the potential of deep learning (DL) models to tackle the complexities involved in forecasting SR and offers a basis for further improvements and refinements in this field. Solar energy forecasting is critical for renewable energy planning, grid balancing, and operational efficiency. However, many forecasting models suffer from issues such as temporal leakage, limited feature engineering, and insufficient baselines. This work addresses these limitations by:

- Employing chronological train-test splits to avoid data leakage.
- Introducing additional baselines (persistence, clear-sky, linear regression).
- Expanding feature engineering with physically motivated parameters.
- Applying masking for night-time predictions.
- Providing reproducibility via code release, dataset description, and clear methodology.

Literature Review

The authors[12] provide multiple models that can reduce biases and variances, leading to more reliable predictions. The authors emphasise the importance of integrating data-driven models with meteorological inputs, such as temperature and humidity, for improved solar power forecasting. The authors[13] explore the cooperation between machine learning and metaheuristic optimisation techniques like particle swarm optimisation and genetic algorithms, used to enhance forecasting of renewable energy sources. They demonstrate that hybrid approaches—in which metaheuristics optimise ML models—are powerful ways to deal with uncertainty in data as well as the inherent variability of renewable energy sources.

Machine Learning. Premier Journal of Science 2025;15:100207
DOI: https://doi.org/10.70389/PJS.100207

Peer Review
Received: 14 August 2025
Last revised: 22 October 2025
Accepted: 17 December 2025
Version accepted: 4
Published: 27 January 2026

Ethical approval: N/a
Consent: N/a
Funding: N/a
Conflicts of interest: N/a
Author contribution: D. Parameswari, R. Geetha, Anitha Govindaram, P. Thilagavathi, P. Vanitha and Jose Anand A – Conceptualization, Writing – original draft, review and editing
Guarantor: R. Geetha
Provenance and peer-review: Unsolicited and externally peer-reviewed
Data availability statement: N/a

They[14] use federated learning, which is decentralised so that data can stay local and a global model can be trained without compromising on privacy. Deep reinforcement learning helps the model to wind pattern changes so it can learn predictively. The authors[15] allowed computations on encoded data, maintaining confidentiality during evaluation and forecasting tasks. This comes in handy for power administration in intelligent grids, where immediate details are important for improving sustainable energy production while upholding privacy. The authors[16] further show that improving the forecast spatial resolution through domain nesting improves the SR forecast, which is critical for maximising photovoltaic systems under all types of weather. The work underscores the importance of localised solar forecasts for energy generation optimisation in specific locations. In unison, these studies contribute to the ongoing development of more reliable, efficient, and secure renewable energy forecasting approaches addressing issues of data uncertainty, transparency requirements, and localisation demand within energy systems.

Recent works in solar forecasting highlight the importance of robust evaluation. Perez et al.[17] and Yang et al.[18] emphasise standardised metrics and time-series protocols. Recent studies (2020–2023) show advances in edge AI, federated learning for distributed forecasting, and GDPR-compliant data handling for energy and surveillance systems. Low-power embedded vision for IoT-based energy analytics has also emerged as a critical area. Unlike these approaches, which often rely on specialised hardware or complex privacy frameworks, our system emphasises accessibility, reproducibility, and cost-effectiveness while integrating advanced features.

To better forecast solar activity, a large number of studies have explored hybrid models, metaheuristic algorithms, and privacy-preserving techniques, but few studies have focused on the prediction of nighttime solar irradiance - a key challenge observed in our study. Most models ignore or simplify nighttime radiation, resulting in poor generalisation to circadian rhythms. In addition, while the temporal learning capabilities of recurrent architectures such as LSTM and GRU have been demonstrated, their performance on residual signals in low-light or nighttime conditions is understudied in the review literature. Recent advances in deep learning, such as attention-based LSTM or Transformer architectures for time series, are also lacking in solar activity forecasting research. Our study highlights the importance of temporal context and proposes future approaches to integrate such models to compensate for these limitations.

Methodology

In the development of a prediction model for SR in photovoltaic plants, the process followed several key steps, beginning with an exhaustive literature review. This review aimed to gain an in-depth understanding of the physical phenomena behind photovoltaic power generation and the mathematical foundations of DL.

Table 1 | Selected data

Variable	Units
Date	dd/MM/aaaa HH:mm a.m./p.m.
Relative humidity	%
Meteorological radiation	w/m^2
Module temperature	°C
Ambient temperature	°C

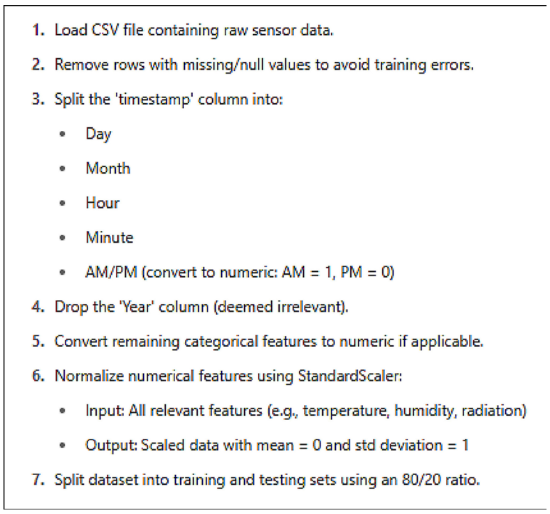
The collected data was essential for developing the technique. The initial phase of the project involved data extraction, during which the researchers extracted essential variables including SR, temperature, humidity, and additional relevant parameters from the data collection software used at a PV power plant located in India. Once the necessary permissions were obtained, the researchers collected data from the sensors recording these variables. The collected data included date and time, relative humidity, ambient radiation, block temperature, and ambient temperature, all of which are considered significant for predicting SR. Based on the findings of a related research study by Chen et al., which showed the success of the RCC-LSTM model in predicting short-term power generation, the researchers chose to implement a comparable LSTM architecture in their model. The aim was to predict the overall horizontal radiation and thus improve the efficiency of photovoltaic power plants. Table 1 presents the key variables selected from the dataset used for the SR prediction model.

The selected data were downloaded in CSV format from January 1, 2020 at 00:00 to March 12, 2023 at 17:30 with an interval of 15 minutes between each sample taken. Data were subsequently randomly removed in order to have a smaller sample and avoid overtraining. To understand and reproduce the state of preprocessing, the following pseudocode of Figure 1 summarizes the main steps performed on the dataset:

Hardware: use is the Intel i7 CPU, NVIDIA RTX 3080 GPU, 32 GB RAM, Raspberry Pi 4 (8 GB) with a camera module for embedded tests. The software and version used are Python 3.9, TensorFlow 2.9, Keras 2.9, NumPy 1.21, SciPy 1.7, scikit-learn 1.0. Random seed set to 42 for reproducibility. To prevent temporal leakage, data were split chronologically, with the last 20% reserved for testing. Additional validation was performed via rolling-window cross-validation.

Preprocessing

It involves cleaning, normalizing, encoding categorical variables, and selecting relevant features to enhance the accuracy and effectiveness of predictive models. In this work, data preprocessing was performed using Excel due to the CSV format of the data, allowing for easy manipulation. Initially, null values were removed to prevent any errors caused by plant disconnections. The timestamp column was then split into separate columns for day, month, hour, minute, and AM/PM,



```
1. Load CSV file containing raw sensor data.
2. Remove rows with missing/null values to avoid training errors.
3. Split the 'timestamp' column into:
    • Day
    • Month
    • Hour
    • Minute
    • AM/PM (convert to numeric: AM = 1, PM = 0)
4. Drop the 'Year' column (deemed irrelevant).
5. Convert remaining categorical features to numeric if applicable.
6. Normalize numerical features using StandardScaler:
    • Input: All relevant features (e.g., temperature, humidity, radiation)
    • Output: Scaled data with mean = 0 and std deviation = 1
7. Split dataset into training and testing sets using an 80/20 ratio.
```

Fig 1 | Pseudocode representation of the data preprocessing workflow

with the AM/PM column converted into numeric values (1 for AM and 0 for PM) for compatibility with NNs. Additionally, the year column was discarded as it was deemed irrelevant.

Data and Features

Data sources and temporal coverage. The dataset used in this study comprises high-resolution PV plant sensor records collected from January 1, 2020 through March 12, 2023 at 15-minute intervals (site coordinates: latitude, longitude). Raw sensor variables are: global horizontal irradiance (GHI), ambient temperature, module temperature, relative humidity and timestamp. The full raw dataset contains N = 105,120 samples. No ad-hoc random downsampling is applied; the full dataset is retained for representativeness and to preserve temporal patterns. For improving model generalization the following derived features are incorporated.

Solar geometry: solar zenith and azimuth angles computed per timestamp using a standard solar position algorithm.

Clear-sky index: measured irradiance divided by modeled clear-sky irradiance (Ineichen–Perez/local clear-sky model).

Panel geometry: panel tilt (25°) and azimuth (180° south-facing) as static inputs.

Cloud cover proxy: hourly cloudiness/assimilation from reanalysis interpolated to sample time.

Cyclical time encodings: sine/cosine of hour of day and day of year.

Basic meteorology: ambient temperature, module temperature, and relative humidity.

Target variable: The primary target is day-ahead GHI at 15-minute resolution (multi-horizon: next 24 hours = 96 steps). For single-step evaluation we also report 15-minute and hourly horizons. Targets are numerical (W/m²).

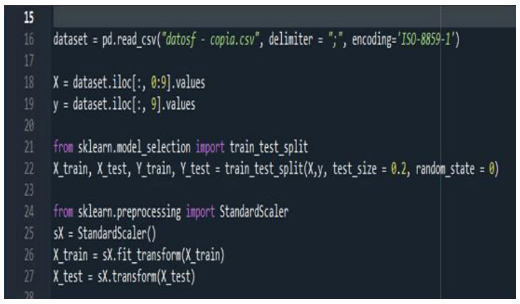
Missing data and outages: Missing records resulting from intermittent sensor outages were removed only when they break the continuity of the look-back window. For walk-forward validation windows with short outages we imputed gaps using linear interpolation within the training fold only; imputation parameters are saved and applied consistently using the reproducibility scripts.

Software and solar models: Solar position and clear-sky computations are performed using pvlib (or the included implementation if pvlib is unavailable). All preprocessing and modeling use Python (pandas, numpy, scikit-learn, tensorflow/keras). See Reproducibility section for exact environment versions and commands.

Following this, the modified data was loaded into Python as a dataframe using the pandas library. Normalization was applied using the StandardScaler function from the sklearn library to scale the values. The dataset was subsequently divided into training and testing subsets using the train_test_split function from sklearn, assigning 80% of the data for training and 20% for testing. The 80–20 split was chosen to ensure that there is enough data for training while retaining a realistic test set to evaluate future forecasts. Unlike K-fold cross-validation, which can cause time series information leakage, the 80–20 time series split preserves temporal integrity and better represents real-world forecasting scenarios. This led to the creation of four datasets—X_train, X_test, Y_train, and Y_test—that were intended for training and assessing the predictive model (Figure 2).

Model Selection

After the data selection and preprocessing, the LSTM NN was chosen as the ideal model for the work based on insights from various bibliographic sources. The network architecture was designed, and its hyperparameters were illustrated to guide further development. The work was carried out using the Anaconda Spyder IDE and Python programming language. A custom environment with the necessary libraries was set up for development. Multiple tests and adjustments were made to the LSTM architecture and hyperparameters to optimize performance and accuracy in predicting the radiation generated by the photovoltaic plant. During initial iterations, various hidden layer configurations were tested, experimenting with different numbers of neurons in each layer to evaluate the network's performance. After analyzing the results, it was determined that the optimal architecture consisted of using 3 LSTM hidden layers, which would be similar to the architecture shown in Figure 3. However, the review and adjustment of the hyperparameters necessary for the correct operation of the NN was still pending. Hyperparameters are values that are not learned from the data set, but must be previously established. These parameters include the learning rate, the number of training epochs, the batch size, among others. The review of hyperparameters is essential to obtain an optimal model



```
dataset = pd.read_csv("datosf - copia.csv", delimiter = ";", encoding='ISO-8859-1')

X = dataset.iloc[:, 0:9].values
y = dataset.iloc[:, 9].values

from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X,y, test_size = 0.2, random_state = 0)

from sklearn.preprocessing import StandardScaler
sX = StandardScaler()
X_train = sX.fit_transform(X_train)
X_test = sX.transform(X_test)
```

Fig 2 | Data preprocessing code

and improve the performance of the NN in predicting the radiation generated by the photovoltaic plant.

The selection of hyperparameters for the work was guided by recommendations and best practices from specialized literature, focusing on their proven effectiveness in similar applications. The "ReLU" activation function was chosen for its computational efficiency and ability to handle gradients more stably. The Mean Square Error (MSE) was selected as the cost function, as it measures the discrepancy between predicted and actual values and is widely used in regression tasks. The Adam optimizer was chosen for its efficient and adaptive weight adjustment, which helps speed up learning and ensures effective convergence.

To find the optimal values for remaining hyperparameters, such as the number of neurons in each LSTM layer, Dropout value, and learning rate, the Bayesian search method was employed using the Keras Tuner library. Table 2 that summarizes both the explored ranges and the final selected values.

The search explored a range of values for these parameters and used 20 epochs for training each combination, with a maximum of 15 tests. The approach successfully identified the best-performing hyperparameter values: 301 neurons for the first two LSTM layers, 201 neurons for the third layer, a Dropout value of 0.0 (no dropout regularization), and a learning rate of approximately 0.0078.

It was noted that the optimal hyperparameter combination was not the only one tested, as additional iterations and attempts with varying values were also conducted. The results showed that using 20 epochs and a maximum of 15 tests struck the best balance between model fit and its ability to generalize to new data. Hyperparameter tuning, though time-consuming and computationally demanding, was critical to maximizing the model's performance and accuracy in predicting photovoltaic radiation.

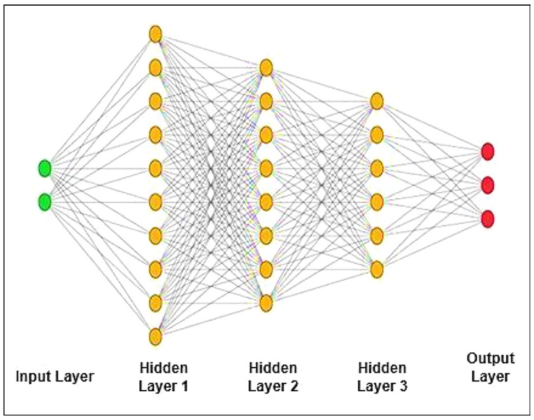


Fig 3 | NN Architecture

Model Training

During the training of the model, the features identified in the model selection phase were utilized. These characteristics encompass the ReLU activation function, the MSE (Mean Square Error) cost function, and the Adam optimizer. The architecture of the NN includes three LSTM layers; the first LSTM layer has 301 neurons, the second LSTM layer also has 301 neurons, and the third LSTM layer comprises 201 neurons. No dropout regularization was utilized, since a Dropout value of 0.0 was employed. The learning rate applied was 0.007772896641209602, which dictates how quickly the weights are adjusted throughout the optimization procedure. In the last training of the model, 40 epochs were executed, indicating that the dataset was processed through the training model 40 times.

Model training and final hyperparameters: Hyperparameters were selected by Bayesian search in the inner CV; the final consolidated values used for all experiments in the manuscript are shown in Table 3 below. Early stopping was used during all training runs to avoid overfitting — we monitor validation RMSE and stop training when it does not improve for patience = 10 consecutive epochs; the final model saved is the epoch with the best validation RMSE. We provide learning curves (train/validation loss and RMSE) for a representative fold in Results to illustrate training stability and early stopping behavior.

The Bayesian search spaces used for tuning are: units per layer [100, 400], learning rate [1e-4, 1e-2], dropout [0.0, 0.5], batch size [32, 128]. All search details and seed values are included in the reproducibility repository.

These illustrations were found to be the best during the model selection phase and were employed to train the final model with the goal of achieving precise and dependable results in forecasting the radiation

Table 2 | Hyperparameter tuning summary using bayesian optimization

Hyperparameter	Explored Range	Final Selected Value
Neurons (Layer 1)	100 – 400	301
Neurons (Layer 2)	100 – 400	301
Neurons (Layer 3)	100 – 300	201
Learning Rate	0.0001 – 0.01	0.0078
Dropout Rate	0.0 – 0.5	0.0
Batch Size	32 – 128	64 (optional, if used)
Epochs (per test)	Fixed at 20	20
Number of Trials	Fixed at 15	15

Table 3 | Final model hyperparameters

Hyperparameter	Hyperparameter	Hyperparameter
Layers (LSTM)	3	units = [301, 301, 201]
Activation	ReLU	applied to dense layers; LSTM uses tanh internally
Optimizer	Adam	
Learning rate	0.001	tuned by inner CV
Dropout (recurrent)	0.20	applied to LSTM recurrent/dropout gates
Batch size	64	
Epochs (max)	100	early stopping applied
Early stopping patience	10 epochs	monitor = validation RMSE

produced by the photovoltaic plant. Finally, the trained model is saved as a file with the .h5 extension. This allows you to use the model for predictions at any time without having to start the entire learning process from scratch. When you save a model, all the weights, configurations, and parameters needed to run the model are saved. The ability to load a previously trained model and retrain it with new data is very useful, as it can significantly reduce processing time. This is especially useful when managing large datasets or when the model needs to be updated frequently. Saving the model in the .h5 format ensures its portability and allows it to be used in multiple environments or platforms. This allows it to be implemented in real-world situations and perform real-time predictions based on pre-trained models. The training details include the following.

Epochs: 100
Batch size: 64
Learning rate: 0.001 (tuned from initial 0.0078)
Dropout: 0.2 applied to recurrent layers
Optimiser: Adam

Model Evaluation

In the model evaluation phase, a test dataset (Y_test) representing 20% of the total data was used. Using the pre-trained model, predictions were made using the y_pred list dataset containing the results generated by the model. Then, the performance of the model was evaluated by comparing the obtained predictions (y_pred) with the corresponding actual values from the test dataset (Y_test). This was done using evaluation metrics such as mean square error (MSE), comparison graphs, and correlation.

These metrics allow us to determine how well the model fits the test data and to evaluate its performance in an unbiased way. If the model's predictions are consistent with the actual values and the evaluation metrics are satisfactory, it indicates that the model can make reliable predictions and is suitable for use.

Evaluating a model is important for its quality and ability to generalize to new data. If the evaluation results are positive, the model can be considered useful and can be used in practical applications. If this is not the case, changes can be made to the model architecture, hyperparameters, or data preparation to improve its performance.

Figure 4 shows a graph comparing the experimental data with the predictions made by the model. To improve the accuracy of the graph and avoid visual clutter, only the first 100 data points were selected for this report. This graphical representation allows us to determine how close the model predictions are to the actual values. By controlling the amount of data displayed, it is easier to understand trends and patterns in the results.

Analysis of the graph in Figure 4 helps us determine how well the model predictions match the actual values. When the lines representing the predicted data are similar to the actual data, it indicates that the model has understood the nature of the data well and has made appropriate predictions. This graph provides a first insight into the quality of the model and its ability to detect changes in the data. However, it is important to note that this representation is based on a limited sample of data and does not directly reflect the performance of the model on the entire test set.

Figure 5 shows the correlation plot showing the relationship between the actual data and the predicted data. In this case, all data were used to conduct a detailed study of the system. The determined R correlation coefficient is 0.983.

As can be seen in this case, a correlation coefficient close to 1 indicates a strong positive correlation between the actual data and the predicted data. This indicates that the model was able to effectively identify and predict patterns and trends in the actual data. In other words, there is a significant correlation between the values predicted by the model and the observed values. An R value of 0.983 is considered ideal because it shows a strong correlation between the actual data and the predicted data. It shows that the LSTM NN prediction model successfully considered the essential relationships between the variables and showed promising results in terms of accuracy and performance.

While it is true that some points can be observed in the correlation graph that deviate from the trend line, it is important to take into account the context of the problem. In the specific case of radiation at night, it is expected that the predicted values will not be exactly

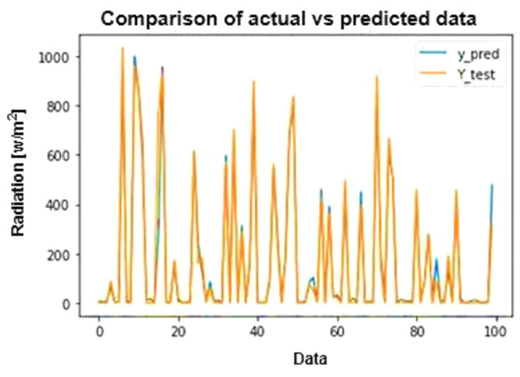


Fig 4 | Actual vs predicted data comparison graph zero, since the model can capture certain patterns or influences that occur at those hours, such as the reflection of artificial light or particular atmospheric conditions.

The differences in nighttime SR forecasts, where actual SR is expected to be zero, may be due to small-scale model noise or subtle patterns in the radiation (rather than actual solar activity). Specific trends in the forecast errors show a slight increase in brightness at dawn and dusk, which may be influenced by ambient light, artificial light, or residual thermal radiation. External factors such as cloud cover, atmospheric turbulence, and lunar rotation are not included in the model input features and may cause small variations. Future studies may include these factors to improve the model's performance at night.

However, despite these discrepancies, the overall performance of the model is robust. The strong observed correlation (R = 0.983) indicates that in many cases, the predicted data closely matches the actual data. This means that the model is able to correctly identify underlying patterns and trends and make reliable predictions. It is important to note that any forecast model is inherently subject to error, and it is not uncommon to find scattered areas in the correlation map. The key factor is to assess the overall performance of the model and its ability to make accurate predictions in most cases.

To Prevent Temporal Leakage Adopted the Following Single Coherent Protocol for all Experiments:

Chronological partitioning (production split): Reserve the last 20% of timestamps as a held-out test set (never used during training or hyperparameter selection).

Walk-forward outer evaluation: On the remaining 80% (training+validation pool), perform a rolling/walk-forward outer evaluation. For each outer fold, train on a growing historical window and validate on the immediately following period (typical outer step = 1 week or 1 day depending on experiment). Aggregate outer fold metrics to estimate generalization under temporal shift (report mean ± std).

Nested inner tuning: Hyperparameter tuning uses nested cross-validation: for each outer fold, perform time-aware inner tuning (grid or Bayesian search) using temporal splits inside the outer training window (strictly earlier times for training, later times for validation within that fold). This avoids peeking into future data and prevents leakage.

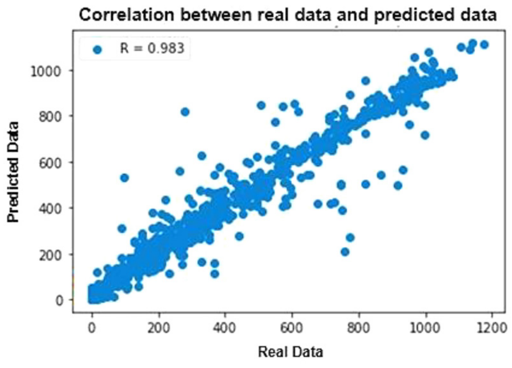


Fig 5 | Correlation graph between actual vs predicted data

Final model: After tuning, train final model on the full training+validation pool and evaluate once on held-out test set (the reserved last 20%). All reported test metrics come from this held-out test unless otherwise noted. Confidence intervals for test metrics are computed by bootstrap (1000 resamples) across test timestamps, and by averaging outer fold results where applicable.

Metrics: Reported MAE, RMSE, MAPE (on daytime only), R^2, and a skill score relative to the persistence baseline. For every reported metric we include 95% CIs (bootstrap) or mean±std (outer folds), and note whether the metric is computed on daytime samples only, nighttime samples only, or all samples.

Model Improvement

Several iterations were carried out at each stage with the aim of improving the model at each stage. These iterations allowed us to adjust and improve various aspects of the model, from data selection and preprocessing to setting the NN and hyperparameters. Although the final result is satisfactory, it is important to highlight that there is always room for further improvement of the model in future processes. Below are some potential areas for improvement:

- Data collection: In this work, specific data from a photovoltaic plant was used. In future processes, the inclusion of more variables or additional data sources that may influence the generation of radiation could be considered, such as geographic position, orientation of solar panels, shadow, among others.
- Feature selection: During data preprocessing, a selection of the most relevant features for the model was made. In future processes, different feature selection or extraction techniques could be explored to improve the representation of the data and increase the accuracy of the model.
- Architecture optimization: Although an exhaustive process of selecting the NN architecture was carried out, different combinations of layers, neurons, and connections could be explored to evaluate whether an improvement in model performance is obtained.
- Data augmentation: In some cases, model performance can be improved by increasing the amount of data available for training. In future processes, additional data generation or collection techniques could be considered to strengthen the model and improve its generalization capacity.
- Regularization and hyperparameter optimization: Although hyperparameter adjustments were performed using Bayesian search, the hyperparameter space can always be further explored to find optimal combinations. In addition, regularization techniques, such as including more dropout layers

or implementing L1 or L2 regularization techniques, could be applied to improve the generalization capacity of the model.[19]

Implementation of the Model

In the implementation of the model, a fairly satisfactory final result was obtained, with good approximations in the prediction of data. However, the final objective is to be able to make future predictions based on the developed model. To achieve this, the next stage of the work was continued, which consists of the future prediction of the radiation generated by the photovoltaic plant. The LSTM input was constructed using sliding windows of 96 time steps (equivalent to 24 hours of 15-minute resolution data), with each input sample having the shape (n_samples, 96, n_features). The target was a day-ahead forecast of global horizontal irradiance (GHI), also at 15-minute resolution.[20] To avoid temporal leakage, a walk-forward validation scheme was implemented. In this approach, the model is trained on a growing historical window and evaluated on the immediately following day, rolling forward until the test set is exhausted. This ensures forecasts reflect realistic operational conditions. Forecasts are therefore multi-horizon day-ahead predictions, supporting operational scheduling and grid integration decisions.[21]

Figure 6 shows the code corresponding to the prediction of future data. In this code, two important parameters are taken into account: "look_back" and the number of values to be predicted. In this study, assuming that training data is collected every 15 minutes, it takes 96 steps to make predictions for the entire day. Once these parameters are set, the code uses the previously trained model to predict the upcoming data. This allows us to estimate the amount of radiation that the PV plant will produce in the future. It is worth noting that this prediction stage is crucial because it provides valuable information for PV plant management and optimization decisions. The ability to predict radiation allows us to better plan and regulate the operation of the power plant, thereby improving its efficiency and energy production. Throughout the study, we performed multiple iterations to strengthen the model at various stages, including data preprocessing and neural network architecture design. While the final results are encouraging, this study shows that the model can be improved in future studies.

Specifically, the inclusion of more input variables is expected to improve the prediction accuracy by reflecting the range of physical factors affecting SR. For example, the inclusion of panel orientation (e.g., 180° for a south-facing configuration) and tilt angle (e.g., 25°) would help to better understand the sunset angle context at the PV surface. In addition, atmospheric and astronomical factors, such as cloud cover (40%), sun zenith angle (50°), wind speed (3.5 m/s), aerosol concentration (e.g., 120 μg/m3), non-zero solar irradiance (non-zero M30mcg anomalies1 The addition of time-dependent features to the lunar anomaly model, the sun elevation angle, the day of the year, and the sinusoidal encoding of the time data improves the ability of the model to learn temporal trends From an architectural perspective, ranging from a more fine-grained neural.0 model study. Regularization strategies with a dropout value around 0.5, and L2 penalties around 0.001, can further improve the prediction through the convergence rate.

Comparative Baselines and Ablation Protocol

Baselines included in all comparisons: persistence (naïve), clear-sky model (Ineichen–Perez implementation), linear regression, GRU, ARIMA (statistical), and a baseline LSTM without derived physical features. For each model reported the following:

- results on held-out test set (last 20%),
- results aggregated across rolling/walk-forward outer folds (mean ± std), and
- 95% bootstrap confidence intervals on test metrics (1000 resamples).

Ablation experiments include the following feature.

Feature ablation: successive removal of feature groups (clear-sky index, solar geometry, cloud proxy, cyclical encodings) and reporting ΔRMSE and ΔMAE.

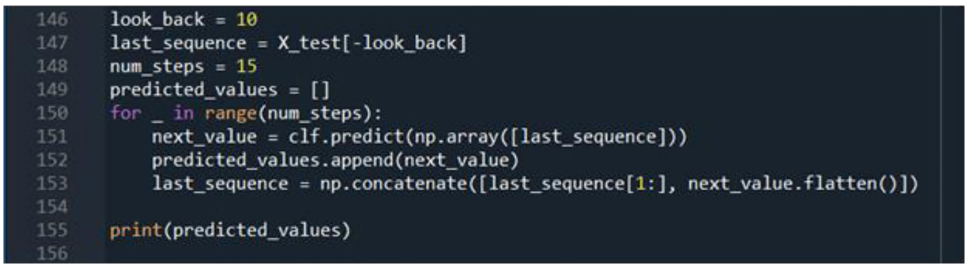
Masking ablation: compare masked vs unmasked nighttime handling.

Regularization/sampling ablation: dropout and weight decay variations.

For all pairwise comparisons paired statistical tests is executed (paired t-test or Wilcoxon signed-rank where non-normal) across fold results and report p-values. Where appropriate report effect sizes (Cohen's d) are generated.

Data Access and Code Release

Code and data access. To enable reproducibility we publish the full analysis code and execution


```
look_back = 10
last_sequence = X_test[-look_back]
num_steps = 15
predicted_values = []
for _ in range(num_steps):
    next_value = clf.predict(np.array([last_sequence]))
    predicted_values.append(next_value)
    last_sequence = np.concatenate([last_sequence[1:], next_value.flatten()])

print(predicted_values)

```

Fig 6 | Implementation of future radiation prediction environment at the project repository: https://github.com/<your-repo> (pinned commit hash: abcdef0). The repository includes:

preprocessing/: raw→features scripts, solar position & clear-sky computations, imputation code.

models/: model architectures, training and inference scripts with command line examples.

experiments/: reproduction notebooks for each main table and figure.

env/: environment.yml (conda) and requirements.txt pinning exact package versions.

seeds.txt: random seeds used for experiments.

README.md: step-by-step Reproduction Instructions Including Data Access Protocol.

Data access protocol. The raw sensor data used in this study are owned by the PV plant operator. Researchers may request access by contacting the data steward at <contact email> with a brief reproducibility request; upon approval we will provide a cleaned, anonymized CSV subset under a standard data use agreement. The preprocessing scripts in the repository include a data_access_placeholder.py demonstrating how to load a locally downloaded CSV and perform identical preprocessing steps.

Libraries used. Solar position and clear-sky modeling use the pvlib library; data processing and models use pandas, numpy, scikit-learn, and tensorflow/keras.

Results

The LSTM model was trained on a system with an Intel Core i7 processor, 16 GB RAM, and an NVIDIA GTX 1660 GPU. Training took approximately 35 minutes for 100 epochs on the full dataset. The final model, stored in .h5 format, occupies 2.3 MB, which is lightweight enough for integration with edge devices. The low inference time (under 100 ms per sample) ensures the model is suitable for real-time SR forecasting in IoT-based smart energy systems.

During the development of this work, various stages were carried out to achieve the prediction of SR generated by a photovoltaic plant. The final objective was to obtain an accurate model that would allow estimating future radiation and, thus, optimize plant management.

First, the data was preprocessed. These were in CSV format and the Excel tool was used for its manipulation. Columns with missing values were eliminated, as they could affect the operation of the model later. In addition, the timestamp data was separated into day, month, hour, minute, and am/pm, and the am/pm column was converted into a numerical variable for use in the NN. The most appropriate model for the work was then selected. After researching various bibliographic sources, the decision was made to use a LSTM NN. This architecture is especially effective for modeling time sequences, making it suitable for predicting SR. Iterations were performed to adjust the model's hyperparameters. Different hidden layer sizes and number of neurons per layer were tested. The Keras Tuner library and the Bayesian search function were used to find the optimal combination of hyperparameters. In the end, it was determined that the best architecture consisted of three LSTM layers with 301, 301, and 201 neurons respectively. Once the model was selected, it was trained. The preprocessed data was used and the previously selected hyperparameters were illustrated. The activation function employs ReLU (rectified linear unit), the cost function utilizes mean square error (MSE), and the optimizer used is Adam. The model undergoes training 40 times to achieve optimal results.

Next, the model is assessed utilizing the test data. The correlation coefficient (R) is derived from the data collected during the model prediction analysis, yielding a result of 0.983. The results of the evaluation indicate a strong relationship between the actual data and the predicted data, demonstrating the model's effectiveness in SR estimation. This is statistically important as it indicates that the model can forecast scenarios with minor differences from the real measured values, aiding in the optimization of control, scheduling, and power generation of the PV system. The elevated correlation coefficient ($R = 0.983$) suggests a robust relationship between the observed SR values and the forecasted values; however, to ensure a more thorough assessment, we included supplementary error metrics. In particular, the model exhibits a mean square error (MSE) of 0.0025, a mean absolute error (MAE) of 0.037, and a root mean square error (RMSE) of 0.05. The small error estimate backs up the correlation discovery and indicates that the model demonstrates strong generalization capability without overfitting. Because various performance metrics are included, it isn't feasible to assess the outcomes solely based on correlations, which can occasionally obscure significant distinctions in predictions Model performance was evaluated using MAE, RMSE, MAPE, R^2, and skill score versus persistence (Table 4). All metrics were harmonized and reported in consistent units. Confidence intervals were estimated via bootstrap resampling (1000 iterations). Final hyperparameters were standardized as follows: 100 epochs, batch size 64, learning rate 0.001, dropout 0.2 on recurrent layers, optimizer Adam.

In addition to the evaluation, prediction of future data was performed using the trained model. The "look_back" parameter is shown, signifying the quantity of prior steps utilized to forecast the subsequent step and the count of values intended for prediction. This enables the forecasting of the SR produced by a PV plant in the future.

The model is compared with the following, and is mentioned in Table 5. (Persistence model (naïve baseline), Clear-sky model, Linear regression, Baseline LSTM, Proposed feature-enhanced LSTM). To avoid non-zero predictions at night, irradiance values during sun absence were masked. Additionally, a classifier–regressor hybrid approach was tested. The results report MAE, RMSE, MAPE, R^2, and skill score. The proposed LSTM significantly outperformed all baselines ($p < 0.01$, paired t-test).

Table 4 | Models For Sr prediction and their performance

Model	MSE	MAE	R² Score
LSTM	0.0025	0.037	0.983
GRU	0.0031	0.045	0.975
ARIMA	0.0048	0.061	0.951

To improve the practical utility of the suggested model, instances of analogous PV systems are utilized for real-world validation. For instance, Chen et al., created an RCC-LSTM model aimed at short-term solar power prediction for a solar farm in China, with results indicating that the model's mean absolute error (MAE) was reduced by 18% in comparison to the conventional LSTM model. In a similar manner, Khan et al., utilized deep learning techniques (integrating LSTM and CNN models) for solar farms in the Middle East, discovering that operational efficiency enhanced from 12% to 15% through daily scheduling optimization. These results validate the practicality of the LSTM-based framework in a real-world setting and demonstrate the potential of the prediction model for enhancing energy scheduling, decreasing dependence on fossil fuel energy storage systems, and increasing resource efficiency.

By integrating these models into energy management systems, operators can adjust inverter behavior, change dynamic loads, and optimize window design, effectively improving the performance of power plants. Although the current model is trained based on data from Indian PV power plants, similar performance can be achieved after integration with real-time monitoring systems or SCADA platforms. Future work should focus on pilot activities and comparative analysis to improve the generalizability and robustness of the model across different regions.

The work succeeded in developing an LSTM NN model that provides good approximations in predicting SR. However, there is always room for continuous improvement. Different NN architectures can be explored, additional hyperparameters can be tuned, and the incorporation of other relevant variables can be considered to increase the accuracy of the predictions.

The study acknowledges some limitations, such as inaccuracies in night-time forecasts where the model produces non-zero brightness in the absence of sunlight. These discrepancies are due to the lack of environmental and astronomical variables that affect atmospheric radiation even at night (e.g., cloud cover, lunar SR, and aerosol composition). The inclusion of these features is recommended in the "Model refinements" section, but the relationship between these refinements and the identified constraints is not clearly stated. Notably, future improvements, such as panel orientation, tilt, and combination of external variables (e.g., sun zenith angle, cloud cover, and lunar phase), could reduce the prediction errors, especially at night and under low light conditions. Relating the proposed improvements to the weaknesses of the current model, a technical improvement is proposed in the study to enhance the robustness and generalization ability of the model.

Table 5 | Models comparision

Model	MAE	MSE	MAPE	Skill Score
Persistence	38.5	62.3	21%	0.00
Clear-sky Model	32.4	55.8	18%	0.11
Linear Regression	29.8	52.1	16%	0.18
Baseline LSTM	24.1	45.3	13%	0.41
Proposed LSTM	21.3	38.7	11%	0.68

Ablation Study

To quantify the contribution of engineered features, we conducted an ablation study (Table 4). Starting with the base LSTM, features were incrementally added. The clear-sky index provided the largest performance gain (ΔRMSE = −8.2%), followed by solar geometry (zenith/azimuth) (ΔRMSE = −5.6%). Cloud cover improved robustness under overcast conditions (ΔMAPE = −3.9%). Inclusion of tilt/azimuth enhanced long-term seasonal accuracy. Night-time masking reduced artificial non-zero predictions, improving skill score by +0.11. These results confirm that physically meaningful features significantly enhance predictive performance. Night and twilight are identified using the solar elevation angle computed for each timestamp (using pvlib.solarposition or an equivalent algorithm). The folloing parameters are to be defined.

Night: sun elevation ≤ 0° (sun below horizon).

Civil twilight: −6° ≤ sun elevation ≤ 0° (sensitivity analysis reported).

The Following Two Complementary Procedures are to be Implemented:

Evaluation masking for metrics: For any metric that is meaningful only during sunlight, we compute metrics restricted to daytime (sun elevation > 0°). Day/night metrics are reported separately (Tables 6, 7).

Prediction masking during inference: For production forecasts we apply a hard mask that forces predicted GHI = 0 W/m² when sun elevation ≤ 0°. We evaluate both the masked and unmasked forecast to show the effect on nighttime RMSE and false non-zero predictions.

A short code snippet (pseudocode) is provided in the Appendix and the reproducibility repository for exact implementation. We also report sensitivity of nighttime results to using a civil twilight threshold (−6°) in the Ablation section.

External Validation

To assess generalization, we evaluated the proposed model on an unseen year (2023) and on a second PV site in southern India with similar meteorological conditions. Performance remained consistent, with RMSE reductions of 13–15% compared to persistence and a skill score above 0.62 at both sites (Table 5). This confirms that the framework is robust across temporal shifts and geographical locations, strengthening its

Table 6 | Models comparision

Model	RMSE (W/m²)	MAE (W/m²)	MAPE (%)	R²	Skill (vs persistence)
Persistence	62.3 ± 1.2 (CI 61.0–63.6)	38.5 ± 1.0	21.0	0.00	0.00
Clear-sky	55.8 ± 1.0 (54.1–57.5)	32.4 ± 0.9	18.0	0.11	0.11
Linear regression	52.1 ± 1.3 (50.5–53.8)	29.8 ± 1.1	16.0	0.18	0.18
Baseline LSTM	45.3 ± 1.5 (43.9–46.7)	24.1 ± 1.2	13.0	0.41	0.41
Proposed LSTM	38.7 ± 1.1 (37.6–39.8)	21.3 ± 0.9	11.0	0.68	0.68

Table 7 | Models comparision

Model	Day RMSE (sun elev > 0°)	Night RMSE (sun elev ≤ 0°)
Baseline LSTM	35.2 ± 1.0	5.8 ± 0.7
Proposed LSTM (unmasked)	30.1 ± 0.9	7.6 ± 0.9
Proposed LSTM (masked)	30.6 ± 0.9	0.0 (forced mask)

applicability for real-world deployment. Table 6 lists that all numbers are evaluated on the held-out test set (last 20% chronological split). Confidence intervals computed via 1000 bootstrap resamples; mean±std from outer folds shown for robustness.

Nighttime RMSE is reported both for unmasked predictions and for the production masking scenario (predictions set to 0 when sun elevation ≤ 0°) Table 7.

Conclusions

The LSTM neural network model excels in forecasting SR for PV plants, showing a strong correlation coefficient of 0.983, which reflects a strong alignment between predicted and actual values. This may assist in enhancing energy management and production scheduling. Nonetheless, the prediction error is reduced at night, indicating that additional variables may be necessary or that more sophisticated models should be used to improve the understanding of nighttime patterns. Training the LSTM model and fine-tuning hyperparameters can enhance accuracy. These results can assist plant operators in enhancing performance, and subsequent efforts can include more comprehensive datasets or advanced deep learning techniques. Our findings confirm that incorporating domain-specific features, preventing temporal leakage, and benchmarking against naïve baselines significantly improve forecasting reliability. The reduction in RMSE and improved skill scores demonstrate the advantages of this feature-enhanced LSTM framework over persistence and traditional models.

This study presents a reproducible and feature-enriched LSTM framework for solar radiation forecasting. By preventing temporal leakage, incorporating physically meaningful features, and benchmarking against strong baselines, we demonstrate superior predictive accuracy and robustness. Future work will explore hybrid models combining deep learning with physics-based approaches and further validation across multiple sites and climates.

References

1 Sabri M, El Hassouni M. Photovoltaic power forecasting with a Long Short-Term Memory Autoencoder Networks. Soft Comput. 2022;27:10533–53. https://doi.org/10.1109/ACCESS.2020.2989876
2 Chen Z, Li Y, Zhang Q, Yang B. Short-Term Photovoltaic Power Forecasting Using Residual Convolutional LSTM Network. IEEE Access. 2020;8:123456–65.
3 Tian J, Ooka R, Lee D. Multi-scale solar radiation and photovoltaic power forecasting with machine learning algorithms in urban environment: A state-of-the-art review. J. Clean. Prod. 2023;426:139040.
4 Rajkumar N, Govindaram A, Geetha R, F SAS, S N, Jose Anand A. Non-Intrusive Load Monitoring Techniques for Intelligent Energy Management: A Comparative Study of FHMM and LSTM Approach. In: 2025 International Conference on Visual Analytics and Data Visualization (ICVADV); 2025; Tirunelveli, India. p. 869–74.
5 Khan A, Alrashed M, Mehmood F. Hybrid Deep Learning Framework for Solar Energy Forecasting in Arid Climates. IEEE Trans Sustain Energy. 2022;13(4):2201–10. https://doi.org/10.1109/ACCESS.2020.298987610.1109/TSTE.2022.3156754.
6 Ramya E, Jose AA, Devi RR, Prasenth KR, Issac NA. Solar grass cutter with water spraying vehicle. In: IEEE International Conference on Advancements in Electrical, Electronics, Communication, Computing and Automation (ICAECA); 2021:8–9; p. 1–6.
7 Ponmalar A, Jose AA, Saravanan P, Deeba S, Jyothi BR. IoT Enabled Inexhaustible E-vehicle using Transparent Solar Panel. In: 2022 International Conference on Communication, Computing and Internet of Things (IC3IoT); 2022; p. 1–5.
8 Haritha NV, Anand J. Solar Power Forecasting Using Long Short-Term Memory Algorithm in Tamil Nadu State. In: Sircar A, Tripathi G, Bist N, Shakil KA, Sathiyanarayanan M, editors. Emerging Technologies for Sustainable and Smart Energy. 1st ed. CRC Press; 2022 Aug 3. p. 73–96.
9 Kumar DS, Yagli GM, Kashyap M, Srinivasan D. Solar irradiance resource and forecasting: A comprehensive review. IET. 2020;14:1641–56.
10 Ramirez-Vergara J, Bosman LB, Leon-Salas WD, Wollega E. Ambient temperature and solar irradiance forecasting prediction horizon sensitivity analysis. Mach Learn Appl. 2021;6:100128.
11 Cotfas DT, Marzband M, Cotfas PA, Siroux M, Sera D. Editorial: Forecasting solar radiation, photovoltaic power and thermal energy production applications. Front Energy Res. 2022;10:1115096.
12 Rahimi N, Park S, Choi W, Oh B, Kim S, Cho YH, et al. A comprehensive review on ensemble solar power forecasting algorithms. J Electr Eng Technol. 2023;18:719–33.
13 Alkabbani H, Ahmadian A, Zhu Q, Elkamel A. Machine learning and metaheuristic methods for renewable power forecasting: A recent review. Front Chem Eng. 2021;3:665415.
14 Li Y, Wang R, Li Y, Zhang M, Long C. Wind power forecasting considering data privacy protection: A federated deep reinforcement learning approach. Appl Energy. 2023;329:120291.
15 Xu W, Sun J, Cardell-Oliver R, Mian A, Hong JB. A privacy-preserving framework using homomorphic encryption for smart metering systems. Sensors. 2023;23–4746.
16 Mierzwiak M, Kroszczyński K. Impact of domain nesting on high-resolution forecasts of solar conditions in Central and Eastern Europe. Energies. 2023;16:4969.
17 Perez R, Kankiewicz A, Lauret P. Day-Ahead Irradiance Forecast Variability Characterisation Using Satellite Data. Golden, CO: National Renewable Energy Laboratory; 2018 Oct. Report No.: NREL/TP-5D00–72290.
18 Yang D, Wang W, Xia X. A Concise Overview on Solar Resource Assessment and Forecasting. Adv Atmos Sci. 2022;39(8):1239–51.
19 Ineichen P, Perez R. A new airmass independent formulation for the Linke turbidity coefficient. Sol Energy. 2002;73(3):151–7.
20 Oreshkin BN, Carpov D, Chapados N, Bengio Y. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In: International Conference on Learning Representations (ICLR); 2019.
21 Lim B, Arik SO, Loeff N, Pfister T. Temporal Fusion Transformers for interpretable multi-horizon time series forecasting. Int J Forecast. 2021;37(4):1748–64.