Improving Solar Forcasting using Deep Learning Model for efficient Hybrid System in Haldwani

Project Report

Submitted By:

Harikesh Pandey
B.Tech (I.T.), Third Year

College of Technology, GBPUAT, Pantnagar

Table of Contents

Abstract

The global transition towards sustainable energy paradigms necessitates the large-scale integration of renewable energy sources into existing power grids. Hybrid energy systems, which combine conventional generators with renewables like solar photovoltaics (PV), are emerging as a critical solution to enhance energy security, reduce dependency on fossil fuels, and mitigate carbon emissions. However, the effective operation of these systems is fundamentally challenged by the stochastic and intermittent nature of solar power. The high variability in solar irradiance, driven by unpredictable meteorological conditions, complicates forecasting, which is indispensable for optimal unit commitment, economic dispatch, and maintaining grid stability. Inaccurate forecasts can lead to suboptimal energy management, increased operational costs, and unnecessary fuel consumption in backup generators.

This study addresses these challenges by developing and validating a sophisticated deep learning model for high-accuracy, short-term solar irradiance forecasting in Haldwani, India—a region with significant solar potential. We propose a novel hybrid architecture that synergistically combines a **Convolutional Neural Network (CNN)** with a **Bidirectional Long Short-Term Memory (BiLSTM)** network. This CNN-BiLSTM model is specifically designed to leverage the distinct strengths of its components: the CNN layers excel at extracting salient spatial features and abstract patterns from multivariate time-series data, while the BiLSTM layers effectively capture complex long-range temporal dependencies in both forward and backward directions. To accelerate model convergence and enhance predictive performance, the network is optimized using the **Nadam optimizer**, which integrates Nesterov's accelerated gradient with the Adam algorithm.

For model training and validation, a comprehensive dataset was meticulously collected for Haldwani, encompassing historical hourly solar irradiance measurements and a suite of correlated meteorological variables such as temperature, humidity, cloud cover, and dew-point. After appropriate pre-processing and normalization, the model was trained to predict the next-day hourly solar irradiance profile. The predictive accuracy of the proposed CNN-BiLSTM model was rigorously evaluated by comparing its forecasts for a recent week from 29 June 2025 to 04 July 2025 against both ground-truth measurements and predictions from a baseline model. It was also validated for Summer and Winter Season to make it reliable throughout the year.

The empirical results demonstrate a marked improvement in forecasting accuracy. The CNN-BiLSTM model significantly reduces key error metrics, such as the **Mean Absolute Error (MAE)** and **Root Mean Squared Error (RMSE)**, when compared to the baseline. Visual analysis through comparative graphs further corroborates the model's superior capability to track the dynamic fluctuations of actual irradiance throughout the day. By providing more reliable and precise solar power predictions, this work offers a valuable tool for the enhanced planning and operation of hybrid energy systems. The findings confirm that advanced AI-driven forecasting techniques are pivotal for unlocking the full potential of solar energy, ultimately contributing to more efficient, cost-effective, and sustainable power generation. Future research will explore the model's generalization to other geographical locations and the extension of the forecast horizon.

Chapter 1: Introduction

1.1 Background: The Imperative for Intelligent Hybrid Energy Systems

The 21st century is defined by a dual challenge: meeting the world's escalating energy demand while simultaneously addressing the urgent threat of climate change. The global consensus, underscored by international agreements like the Paris Accord, mandates a decisive shift away from fossil fuels towards clean, sustainable energy sources. Among these, solar photovoltaic (PV) technology has emerged as a frontrunner due to its falling costs, scalability, and widespread availability. However, the standalone deployment of solar power is inherently limited by its intermittency—power is only generated when the sun is shining, leading to variability and unpredictability.

To bridge this gap and ensure a continuous, reliable power supply, **hybrid energy systems** have become a cornerstone of modern energy strategy. These systems intelligently combine two or more energy sources, typically pairing a renewable source like solar PV with a dispatchable source like a diesel generator or a battery energy storage system (BESS). This synergy allows for the best of both worlds: clean, low-cost energy from the sun during the day, and dependable power from the conventional source during the night or periods of heavy cloud cover.

The operational heart of such a hybrid system is its energy management system (EMS), which makes critical decisions about which power source to use, when to charge or discharge batteries, and how to balance the load. The efficiency, cost-effectiveness, and environmental benefit of the entire system hinge on one critical input: an **accurate forecast of solar irradiance**. A precise prediction allows the EMS to proactively plan its operations—for example, by minimizing the use of the costly and polluting diesel generator, optimizing battery charging cycles, and ensuring grid stability. Without it, the system operates reactively, leading to inefficiencies, higher fuel consumption, and reduced reliability.

1.2 Problem Statement: The Challenge of Solar Variability in Haldwani

Haldwani, situated in the Nainital district of Uttarakhand, India, is a region with considerable solar energy potential. However, its geographical location in the foothills of the Himalayas subjects it to complex and dynamic microclimates. Weather patterns can change rapidly due to topographical effects, seasonal monsoons, and winter fog, leading to significant and often unpredictable fluctuations in solar irradiance. This high degree of variability poses a substantial challenge to the reliable operation of forecast-dependent hybrid energy systems.

Traditional forecasting methods have struggled to provide the requisite accuracy for this complex environment:

- **Classical Statistical Models:** Methods like ARIMA (Autoregressive Integrated Moving Average) often fail because they are based on assumptions of linearity and

stationarity, which do not hold for the highly non-linear and non-stationary nature of meteorological time-series data.

- **Numerical Weather Prediction (NWP) Models:** While powerful, these physics-based models are computationally intensive and typically provide forecasts at a coarse spatial and temporal resolution. They often miss the localized, rapid changes critical for hourly energy management at a specific site like Haldwani.

The consequence of inaccurate forecasting is severe. It can lead to an over-reliance on the diesel generator, defeating the primary environmental and economic goals of the hybrid system. It also complicates load management and can jeopardize the stability of the local power supply. Therefore, a significant gap exists for a forecasting model that can accurately capture the complex, non-linear relationships between various meteorological parameters and solar irradiance in this specific, challenging environment.

1.3 Objectives of the Study

The primary objective of this research is to design, implement, and validate a high-performance deep learning model for next-day hourly solar irradiance forecasting to enhance the operational efficiency of hybrid energy systems in Haldwani.

The specific objectives are:

- **To curate and preprocess a comprehensive dataset** by collecting historical time-series data for solar irradiance and key meteorological variables (e.g., temperature, humidity, cloud cover, dew-point) specific to Haldwani.

- **To design and build a hybrid deep learning model based on a Convolutional Neural Network (CNN) and a Bidirectional Long Short-Term Memory (BiLSTM) network.** This architecture is chosen to effectively capture both local, spatial-like features among the input variables (via CNN) and long-term temporal dependencies in the data sequence (via BiLSTM).

- **To rigorously train and validate the proposed model**, comparing its predictive performance against actual measured irradiance data and baseline forecast values for a selected period.

- **To quantitatively evaluate the model's performance** using standard statistical error metrics, including Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and the Coefficient of Determination (R2), supplemented by visual analysis through graphical plots to demonstrate the improvement in accuracy.

1.4 Scope of the Study

This study is sharply focused on advancing the methodology for meteorological forecasting as a critical input for energy systems. The scope and its limitations are defined as follows:

- **Scope:** The core focus of this project is the development and evaluation of a deep learning model for **next-day, hourly-resolution forecasting of Global Horizontal Irradiance (GHI)** for the specific geographical coordinates of Haldwani. The

research encompasses data collection, pre-processing, model architecture design, training, and performance evaluation.

- **Delimitations:** This study deliberately excludes certain related areas to maintain a clear focus.
    - **PV Power Output Modeling:** The research forecasts solar irradiance (Wh/m^2), not the final AC power output (kW) from a specific PV plant. The conversion from irradiance to power is a separate engineering task that depends on panel efficiency, tilt angle, temperature coefficients, inverter performance, and other site-specific factors.
    - **Economic Analysis and System Optimization:** While the improved forecast is intended to enable better economic dispatch and system optimization, this study does not design the energy management system (EMS) algorithm or perform a cost-benefit analysis of the hybrid system's operation.
    - **Long-Term Forecasting:** The model is designed for short-term operational forecasting (24-hour horizon) and is not intended for long-term climate studies or resource assessment for initial plant siting.

Despite these delimitations, the accurate and reliable solar irradiance forecast produced by this work serves as the essential foundation upon which effective hybrid energy system management, PV output modeling, and subsequent economic optimizations can be built.

Chapter 2: Review of Literature

The accurate forecasting of solar irradiance is a cornerstone for the reliable and economically viable operation of solar-integrated hybrid energy systems. A vast body of research has been dedicated to this problem, leading to an evolution of forecasting models. These methodologies can be broadly categorized into traditional statistical techniques, classical machine learning models, and advanced deep learning architectures. This chapter provides a critical review of these approaches, examining their strengths and inherent limitations, thereby establishing the scientific rationale for adopting a hybrid CNN-BiLSTM architecture for this study.

2.1 Traditional Statistical Approaches: Foundations and Limitations

The earliest attempts at time-series forecasting relied on statistical models that aimed to describe the mathematical relationships within the data. Prominent among these are **Auto-Regressive Integrated Moving Average (ARIMA)** models, which forecast future values based on a linear combination of their own past values and lagged forecast errors. Other methods like **exponential smoothing** assign exponentially decreasing weights to older observations.

- **Strengths:** The primary advantages of these models are their simplicity, interpretability, and low computational cost. They can be effective for short-term predictions in stable, clear-sky conditions where patterns are highly regular.
- **Limitations:** The fundamental weakness of statistical models lies in their assumption of **linearity and stationarity** in the time series. Solar irradiance data is profoundly non-linear and non-stationary, influenced by complex atmospheric dynamics, diurnal cycles, and stochastic events like the rapid movement of clouds. Consequently, ARIMA and similar models often fail to capture the sharp ramps and sudden drops in irradiance characteristic of partly cloudy or overcast days, leading to significant forecast errors.

2.2 Machine Learning Models: Capturing Non-Linearity

The advent of machine learning (ML) marked a significant leap forward. Models such as **Support Vector Machines (SVM)** for regression (SVR), **Random Forests**, and **Gradient Boosting Machines (GBM)** offered a powerful alternative capable of capturing complex, non-linear relationships between predictor variables (e.g., temperature, humidity, time of day) and the target variable (irradiance).

- **Strengths:** Unlike statistical models, ML algorithms do not assume linear relationships and can model intricate patterns effectively. Ensemble methods like Random Forest and GBM are particularly robust against overfitting and have demonstrated superior accuracy over traditional methods in numerous studies.
- **Limitations:** Despite their strengths, these "shallow" ML models have a critical limitation in the context of time-series forecasting: they are inherently **memoryless**.

They treat each time step as an independent data point, failing to natively understand the sequential nature and temporal dependencies within the data. To overcome this, they rely heavily on **manual feature engineering**, where domain experts must create lagged variables (e.g., irradiance from one hour ago, 24 hours ago) and moving averages to feed the model historical context. This process is not only time-consuming and suboptimal but also may fail to identify all the relevant temporal patterns that a more advanced architecture could learn automatically.

2.3 Deep Learning Techniques: Mastering Temporal Dependencies

Deep learning, a subfield of machine learning, has revolutionized time-series forecasting by introducing architectures designed specifically to learn from sequential data.

- **Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM):** RNNs were the first step, featuring internal loops that allow information to persist, giving them a form of memory. However, standard RNNs suffer from the **vanishing gradient problem**, making them incapable of learning long-range dependencies. **Long Short-Term Memory (LSTM)** networks were created to solve this specific issue. LSTMs incorporate a sophisticated gating mechanism (input, forget, and output gates) that allows the network to selectively remember relevant information and forget irrelevant data over long time periods. This makes them exceptionally well-suited for modeling solar irradiance, where conditions from hours or even days prior can influence the present state.
- **Bidirectional LSTM (BiLSTM):** A standard LSTM processes data chronologically, learning only from past context. However, for a given point in time, future context can be just as informative. A **Bidirectional LSTM (BiLSTM)** enhances the LSTM by processing the data sequence in both the forward (past to future) and backward (future to past) directions using two separate hidden layers. The outputs are then concatenated, providing the model with a richer, more complete contextual understanding of the time series, often leading to improved accuracy.
- **Convolutional Neural Networks (CNNs) for Feature Extraction:** While LSTMs master temporal sequences, **Convolutional Neural Networks (CNNs)** excel at hierarchical feature extraction. Originally designed for image processing, 1D-CNNs can be powerfully applied to multivariate time-series data. By sliding a kernel across the input variables at each time step, a CNN can automatically learn to extract salient local patterns and interactions between variables (e.g., the combined effect of a sudden drop in temperature and a rise in humidity). When combined in a **hybrid CNN-BiLSTM architecture**, the model leverages the best of both worlds: the CNN acts as an advanced feature extractor, feeding a set of highly informative, abstracted features into the BiLSTM, which then models the long-range temporal dependencies of these features.

2.4 Optimization and Performance Enhancement

The performance of a deep learning model is also dependent on the optimization algorithm used to minimize the loss function during training. While traditional optimizers like Stochastic Gradient Descent (SGD) exist, more advanced adaptive optimizers are now standard. **Adam (Adaptive Moment Estimation)** is a widely used optimizer that computes adaptive learning rates for each parameter. An even more recent advancement is **Nadam (Nesterov-accelerated Adaptive Moment Estimation)**, which incorporates Nesterov's accelerated gradient into the Adam framework. This "look-ahead" capability allows Nadam to often converge faster and achieve a better final performance, making it an excellent choice for complex models like the CNN-BiLSTM.

2.5 Gaps in Existing Research and Contribution of This Study

Despite the advancements, a review of the existing literature reveals several critical gaps that this research aims to address:

1. **Lack of Geographical and Microclimatic Specificity:** The performance of any forecasting model is highly dependent on local climatic conditions. There is a notable scarcity of research focusing on solar forecasting in regions with unique and complex microclimates, such as the Himalayan foothills. The specific weather dynamics of Haldwani remain an under-studied area.
2. **Insufficient Real-World Benchmarking:** Many academic studies validate their models against simplistic persistence baselines or other models from the literature. Few perform a crucial benchmark against real-world, operational forecasts provided by commercial weather services or APIs. This comparison is essential to prove tangible value over currently available tools.
3. **Under-Exploration of Advanced Hybrid Models for Site-Specific Applications:** While hybrid CNN-LSTM and CNN-BiLSTM models have shown exceptional promise on standardized datasets, their application and meticulous fine-tuning for site-specific, multivariate solar forecasting for direct integration with hybrid energy systems is an area ripe for exploration.

This study directly addresses these gaps by developing a tailored CNN-BiLSTM model specifically for the Haldwani region, validating its performance against both ground-truth data and existing forecast services, and demonstrating its potential to provide a more reliable input for real-world hybrid energy management.

Chapter 3: Methodology

This chapter details the systematic framework and technical procedures employed in this study. It covers the selection of the study area, the comprehensive data acquisition strategy, the preprocessing techniques used to prepare the data, the architecture of the proposed deep learning model, and the configuration for training and evaluation. Each step is designed to ensure the reproducibility and validity of the research findings.

3.1 Study Area: Haldwani, Uttarakhand

The geographical focus of this study is **Haldwani, Uttarakhand, India (approx. 29.22° N latitude, 79.52° E longitude)**. Situated in the foothills of the Himalayas, this region presents an ideal and challenging testbed for developing a robust solar forecasting model. Its subtropical climate is characterized by distinct seasonal variations: hot summers, a prominent monsoon season with heavy cloud cover and high humidity, and cool, often foggy winters. This meteorological diversity, driven by complex topographical influences, results in significant variability in daily and hourly solar irradiance. A model that can perform accurately in such a dynamic environment is likely to be robust and adaptable to other locations with less extreme weather fluctuations.

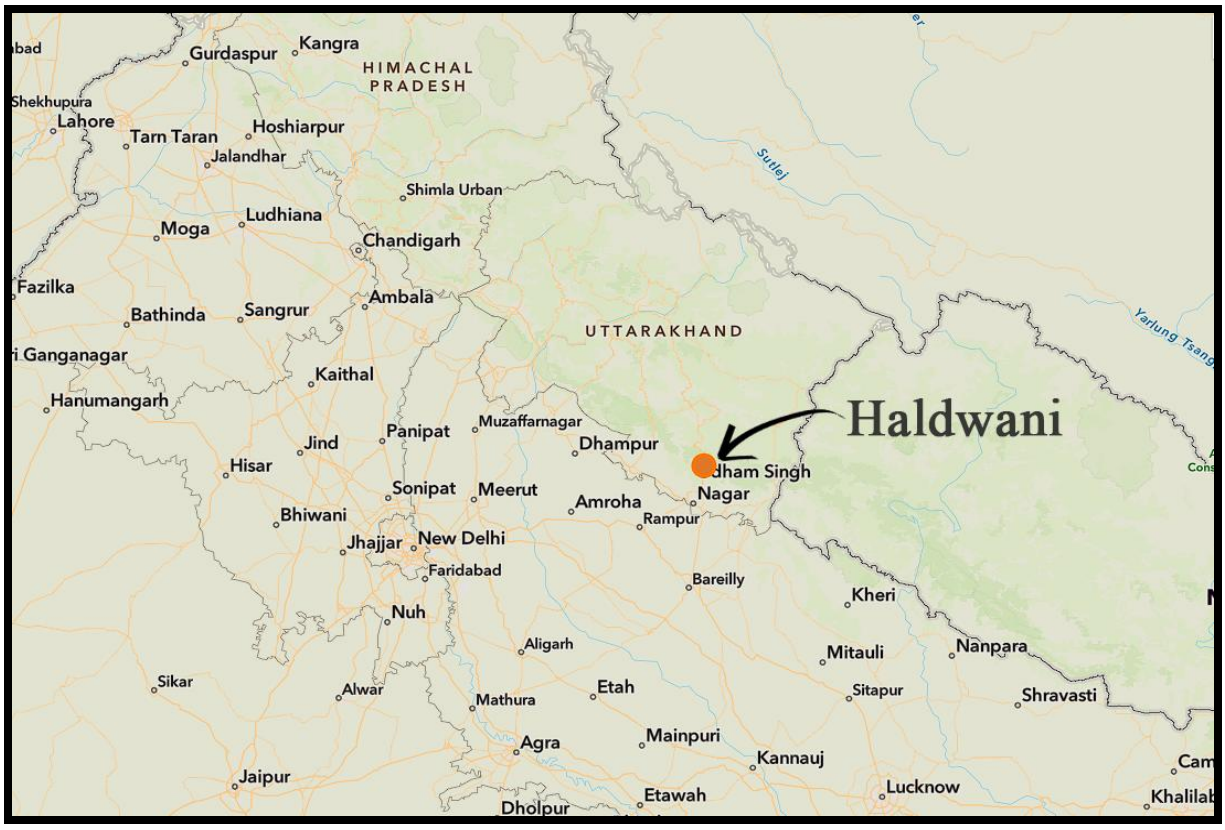

Figure 1. Haldwani, Uttarakhand

3.2 Data Collection and Sources

A robust forecasting model requires high-quality, comprehensive data. This study employs a two-pronged data collection strategy, utilizing both historical satellite-derived data for model training and real-world forecast data for operational testing.

Historical Data for Model Training: NASA POWER

The primary source for historical data is the **NASA Prediction of Worldwide Energy Resource (POWER)** project. The POWER program provides a global climatology of meteorological and solar radiation parameters derived from satellite observations and NASA's assimilation models.

- **Resource:** This dataset offers consistent, long-term, and globally available solar and meteorological information, making it an invaluable resource for renewable energy research where ground station data is sparse.
- **Parameters:** For this study, we extracted hourly data including **Global Horizontal Irradiance (GHI)**, air temperature, relative humidity, cloud amount and dew point temperature. We also utilized the calculated **Solar Zenith Angle (SZA)**. The GHI data from NASA POWER serves as the **ground-truth target variable** for training our model.

Forecast Data for Model Testing: Open-Meteo API

To simulate a real-world operational scenario, we sourced future weather forecasts from the **Open-Meteo API**.

- **Resource:** Open-Meteo is a modern, open-source weather API that provides high-resolution forecast data by aggregating results from multiple leading **Numerical Weather Prediction (NWP)** models, including the German Weather Service's ICON model and NOAA's GFS model. This provides a high-quality, accessible forecast that mimics what an actual hybrid energy system would use.
- **Parameters:** We fetched hourly forecast data for the same set of meteorological features used in training (temperature, humidity, cloud amount, etc.) for the target evaluation for the period from 29 June 2025 to 04 July 2025.
- **Purpose:** Using these forecasted inputs to predict solar irradiance allows for a true test of the model's practical utility, as it must make predictions based on imperfect future data, just as it would in a live deployment.

3.3 Input Features and Target Variable

The selection of input features is critical for model performance. Based on domain knowledge, correlation analysis, and established literature, the following variables were chosen:

- **Temporal Features:**
    - MO (Month): To capture seasonal patterns.
    - DY (Day): To capture daily variations within the month.
    - HR (Hour): To capture the diurnal (24-hour) cycle of the sun.

- **Meteorological Features:**
    - T2M (Air Temperature at 2m): Influences atmospheric density and PV panel efficiency.
    - RH2M (Relative Humidity at 2m): Indicates the amount of water vapor, which scatters and absorbs solar radiation.
    - D2M (Dew Point Temperature at 2m): Provides additional information about moisture content.
- **Atmospheric & Astronomical Features:**
    - SZA (Solar Zenith Angle): The primary driver of the clear-sky irradiance profile.
    - Cld_Amt (Cloud Amount): The most significant modulating factor, directly impacting the amount of radiation reaching the surface.

Target Variable:

- SolarOutput: The target for prediction is the **Global Horizontal Irradiance (GHI)**, measured in watts per square meter (Wh/m^2). This represents the total solar radiation received on a horizontal surface and is the key input for any solar power generation model.

3.4 Data Preprocessing

Raw data is rarely suitable for direct use in deep learning models. A rigorous preprocessing pipeline was implemented to ensure data quality and optimize model training.

1. **Data Cleaning and Alignment:** The datasets were first inspected for missing or null values, which were handled using appropriate imputation techniques. All data timestamps were synchronized to a consistent time zone to ensure correct temporal alignment between the features and the target variable.
2. **Feature Scaling:** All numerical input features were scaled to a range between 0 and 1 using **Min-Max Normalization**. This is a crucial step, as deep learning models converge significantly faster and more reliably when all input features share a common scale, preventing any single feature from disproportionately influencing the model's weight updates.
3. **Temporal Windowing (Sequence Generation):** To enable the BiLSTM layers to learn from past data, the time-series data was transformed into supervised learning sequences. A **sequence-to-sequence** structure was adopted, where an input window containing the last **6 hours** of meteorological data is used to predict a future sequence of the **next 24 hourly** solar irradiance values.
4. **Train-Test Split:** The historical dataset was partitioned into three distinct sets to ensure robust model development and unbiased evaluation:
    - **Training Set (80%):** Used for training the model to learn the underlying patterns.

- **Validation Set (10%):** Used during training to monitor performance on unseen data, tune hyperparameters, and prevent overfitting.
- **Testing Set (10%):** A completely held-out set used for a final, unbiased assessment of the model's generalization performance on historical data.

3.5 Model Architecture: CNN-BiLSTM

3.5.1. Imports

This block imports all the necessary tools and libraries. Think of it as gathering your ingredients before cooking.

- **Data Handling**: pandas and numpy are the workhorses for loading, cleaning, and structuring the numerical data.
- **Machine Learning**: scikit-learn provides essential utilities like splitting data (train_test_split), scaling features (MinMaxScaler), and calculating performance scores (mean_absolute_error, etc.). joblib is used to save the trained scalers.
- **Deep Learning** : tensorflow.keras is the core framework used to design, build, and train the neural network. Each imported layer (Conv1D, LSTM, Dense, etc.) is a distinct building block for our model.
- **Tuning & Plotting**: optuna is a powerful library that automatically finds the best settings (hyperparameters) for our model. matplotlib is used to create visual plots of the results.

```
import pandas as pd
import numpy as np
import optuna
import joblib
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.layers import (
    Input,
    Conv1D,
    Bidirectional,
    LSTM,
    Dense,
    Attention,
    LayerNormalization,
    Add,
    GlobalAveragePooling1D,
)
from tensorflow.keras.optimizers import Nadam
```

```
import matplotlib.pyplot as plt
```

3.5.2. Data Loading and Preprocessing

Here, the raw data is cleaned and formatted to be suitable for the neural network.

- **Loading and Renaming:** The script loads a CSV file and renames columns for consistency and easier coding (e.g., Temp2m becomes T2M).
- **Feature/Target Split:** The data is separated into X (the input features or "predictors" like temperature, humidity, etc.) and y (the output target or "what we want to predict," which is SolarOutput).
- **Scaling:** Neural networks train best when all input numbers are on a similar scale. MinMaxScaler transforms all feature and target values into a range between **0 and 1**. This prevents features with large values (like irradiance) from overpowering features with small values (like humidity). Two separate scalers are used to avoid data leakage from the target into the features.
- **Saving Scalers:** The trained scalers are saved using joblib. This is critical for deployment, as any new data used for future predictions must be scaled in the **exact same way** as the training data.

```
# === Load and Preprocess Data ===
df = pd.read_csv("final_merged.csv")
df.rename(
    columns={
        "Temp2m": "T2M",
        "Humidity2m": "RH2M",
        "SolarZenith": "SZA",
        "Forecasted_SW_DWN": "Forecasted",
        "SolarIrradiance": "SolarOutput",
        "CloudCover": "Cld_Amt",
        "DewPoint2m": "D2M",
    },
    inplace=True,
)

X = df[["MO", "DY", "HR", "T2M", "RH2M", "SZA", "Cld_Amt", "D2M"]]
y = df["SolarOutput"]

scaler_X = MinMaxScaler()
scaler_y = MinMaxScaler()

X_scaled = scaler_X.fit_transform(X)
y_scaled = scaler_y.fit_transform(y.values.reshape(-1, 1))

joblib.dump(scaler_X, "scaler_X.save")
joblib.dump(scaler_y, "scaler_y.save")
```

3.5.3. Sequence Builder Function

This function is a crucial data transformation step. LSTMs don't look at data one row at a time; they look at **sequences** of data. This function chops the continuous time-series data into smaller, overlapping windows. For this model, it creates input blocks of **24 hours of**

meteorological features to predict the corresponding **24 hours of solar irradiance**. The output is a 3D NumPy array (samples, timesteps, features), which is the exact format an LSTM layer expects.

```
# === Sequence Builder (24-hour blocks) ===
def create_daily_blocks(X, y, block_size=24):
    X_seq, y_seq = [], []
    for i in range(0, len(X) - block_size):
        X_block = X[i : i + block_size]
        y_block = y[i : i + block_size]
        if len(X_block) == 24 and len(y_block) == 24:
            X_seq.append(X_block)
            y_seq.append(y_block.flatten())
    return np.array(X_seq), np.array(y_seq)
```

3.5.4. Model Architecture: The CNN-BiLSTM Brain

This function defines the layers of our neural network. The data flows through these layers sequentially.

1. **Input Layer:** Defines the entry point for our data, specifying its shape: 24 time steps, with 8 features at each step.

2. **Conv1D Layer:** This convolutional layer acts as a powerful **feature extractor**. It slides a small window (kernel) across the 24-hour sequence, learning to detect important local patterns, such as the relationship between a drop in temperature and a rise in cloud cover over a 3-hour period.

3. **Bidirectional(LSTM) Layer:** This is the model's memory core. The LSTM processes the sequence of features extracted by the CNN. By wrapping it in Bidirectional, the model processes the sequence **forwards (past to future) and backwards (future to past)**. This gives it a richer context; for example, knowing that cloud cover is *forecasted to increase* (backward pass) helps it better predict the immediate drop in irradiance.

4. **Attention Mechanism:** This layer makes the model "smarter." Instead of treating all 24 hours in the sequence equally, the attention mechanism learns to **assign more weight to the most influential time steps** when making its prediction. For instance, to predict irradiance at 3 PM, it might learn to "pay more attention" to the weather data from noon to 2 PM.

5. **Residual Connection (Add):** The output of the attention layer is added back to the output of the BiLSTM layer. This "shortcut" helps gradients flow better during training, making the network easier to train and often improving performance.

6. **GlobalAveragePooling1D Layer:** This layer takes the final sequence output and **condenses it into a single, fixed-size vector** by averaging the values across all 24 time steps. This creates a compact summary or "fingerprint" of the day's weather pattern.

7. **Dense Output Layer:** This is the final layer. It takes the summarized feature vector from the pooling layer and produces the final 24-hour forecast. It has **24 neurons**, one for each predicted hour.

8. **Compilation:** The model.compile() step configures the training process. It sets the **optimizer (Nadam)**, which is the algorithm that updates the model's internal weights, and the **loss function (mse)**, which measures how far off the model's predictions are from the actual values.

```
# === Model Architecture ===
def build_model(input_shape):
    inp = Input(shape=input_shape)  # shape = (24, 8)
    x = Conv1D(64, kernel_size=3, activation="relu", padding="same")(inp)
    x = LayerNormalization()(x)
    x = Bidirectional(LSTM(64, return_sequences=True))(x)
    x = LayerNormalization()(x)
    attn_output = Attention()([x, x])
    x = Add()([x, attn_output])
    x = GlobalAveragePooling1D()(x)
    out = Dense(24)(x)  # Predict 24 hours
    model = Model(inputs=inp, outputs=out)
    model.compile(optimizer=Nadam(), loss="mse")
    return model
```

3.5.5. Optuna Hyperparameter Tuning

Manually finding the best settings for a model (like the batch_size) can be tedious. **Optuna automates this process**. The objective function defines a single **"trial"**: it builds the model, trains it for a few epochs with a batch_size suggested by Optuna, and returns the validation loss. Optuna runs this function many times (n_trials=20), intelligently choosing new batch sizes based on past results, until it finds the one that produces the lowest validation loss.

```
# === Optuna Tuning ===
def objective(trial):
    batch_size = trial.suggest_categorical("batch_size", [16, 32, 64])
    X_seq, y_seq = create_daily_blocks(X_scaled, y_scaled, 24)
    X_train, X_val, y_train, y_val = train_test_split(
        X_seq, y_seq, test_size=0.2, random_state=42
    )

    model = build_model((24, X.shape[1]))
    history = model.fit(
        X_train,
        y_train,
        validation_data=(X_val, y_val),
        epochs=10,
        batch_size=batch_size,
        verbose=0,
    )
    return history.history["val_loss"][-1]
```

```
study = optuna.create_study(direction="minimize")
study.optimize(objective, n_trials=20)
```

3.5.6. Final Training and Evaluation

1. **Final Training:** Once Optuna finds the best batch_size, the model is trained one last time on the full training dataset for a longer duration (epochs=30) to allow it to learn thoroughly.
2. **Prediction and Inverse Transform:** The trained model predicts values on the unseen X_test. These predictions are in the scaled [0, 1] range. The scaler_y.inverse_transform() function is crucial; it converts these scaled predictions back into their original, meaningful units (W/m2).
3. **Metrics Calculation:** Standard metrics are calculated to quantitatively assess performance:

- **MAE (Mean Absolute Error):** The average absolute difference between predicted and actual values. An MAE of 50 means the forecast is off by 50 Wh/m^2 on average.
- **RMSE (Root Mean Squared Error):** Similar to MAE but penalizes larger errors more heavily.
- **R^2 (R-squared):** Indicates the proportion of variance in the target that is predictable from the features. An R2 of 0.95 means the model explains 95% of the variability in solar irradiance.

```
# === Final Training ===
best_batch = study.best_params["batch_size"]
print(f"\nBest Batch Size: {best_batch}")

X_seq, y_seq = create_daily_blocks(X_scaled, y_scaled, 24)
X_train, X_test, y_train, y_test = train_test_split(
    X_seq, y_seq, test_size=0.2, random_state=42
)

final_model = build_model((24, X.shape[1]))
final_model.fit(X_train, y_train, epochs=30, batch_size=best_batch, verbose=1)

# === Evaluation ===
y_pred_scaled = final_model.predict(X_test)
y_pred = scaler_y.inverse_transform(y_pred_scaled)
y_actual = scaler_y.inverse_transform(y_test)

mae = mean_absolute_error(y_actual, y_pred)
rmse = np.sqrt(mean_squared_error(y_actual, y_pred))
r2 = r2_score(y_actual, y_pred)

print(f"\nMAE: {mae:.2f} | RMSE: {rmse:.2f} | R²: {r2:.4f}")
```

3.5.7. Plotting and Saving

- **Plotting:** A plot is generated for the first sample in the test set. This is a vital **sanity check** to visually confirm that the model's predictions follow the general shape and magnitude of the actual irradiance values.
- **Saving:** The final_model.save() command saves the entire trained model—its architecture, learned weights, and optimizer state—to a single file. This saved file can now be loaded into another application for making real-world forecasts without needing to retrain.

```
# === Plot First Sample ===
plt.figure(figsize=(12, 5))
plt.plot(y_actual[0], label="Actual(Historical)")
plt.plot(y_pred[0], label="Predicted")
plt.title("24-Hour Solar Irradiance Prediction")
plt.xlabel("Hour")
plt.ylabel("Irradiance (Wh/m²)")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

# === Save Final Model ===
final_model.save("cnn_bilstm_attention_model_final.keras")
print("\nModel and scalers saved successfully.")
```

3.6 Model Training Configuration

The training process is where the model learns the complex patterns within the data. This phase was carefully configured to ensure efficient convergence and optimal performance. The key parameters for the training loop are detailed below:

- **Optimizer**: Nadam To guide the model's learning process, the Nadam (Nesterov-accelerated Adaptive Moment Estimation) optimizer was selected. Nadam is an advanced optimization algorithm that improves upon the popular Adam optimizer by incorporating Nesterov momentum. This "look-ahead" mechanism allows for a more intelligent and direct path towards the optimal solution, often resulting in faster convergence and a more accurate final model compared to other optimizers.
- **Loss Function**: Mean Squared Error (MSE) The model's performance during training was quantified using the Mean Squared Error (MSE) loss function. For a regression task like irradiance forecasting, MSE is a standard and effective choice. It calculates the average of the squared differences between the predicted values and the actual values. By squaring the errors, it penalizes larger prediction mistakes more heavily, compelling the model to minimize significant deviations and improve overall accuracy.
- **Epochs and Batch Size:** The model was trained for 30 epochs. An epoch represents one complete pass of the entire training dataset through the network. This number was

chosen empirically to strike a balance between allowing the model sufficient time to learn the underlying patterns and preventing overfitting, which was monitored using the validation set.

The batch size, which determines the number of samples processed before the model's internal weights are updated, was not chosen manually. Instead, it was systematically determined using the Optuna hyperparameter optimization framework. Optuna automatically tested various batch sizes from the set [16, 32, 64] and selected the one that yielded the lowest validation error, ensuring a data-driven and optimal configuration for this specific dataset and model architecture. For instance, after several trials, Optuna selected a batch size of 32 as the most effective.

3.7 Tools and Libraries Used

This research was conducted using a modern, open-source software stack based on Python. The specific tools and libraries were chosen for their robustness, extensive documentation, and widespread adoption in the data science community.

- **Core Language: Python 3.10.11** served as the primary programming language for the entire project implementation.
- **Deep Learning Framework: TensorFlow (v2.x)** was the core deep learning library, providing the computational backend for gradient calculations and model execution. The model itself was constructed using the **Keras API**, a high-level interface within TensorFlow that simplifies the process of building, training, and evaluating complex neural networks.
- **Data Manipulation and Analysis: Pandas** was used extensively for data loading, cleaning, and structuring in its powerful DataFrame format. **NumPy** provided the foundation for all numerical operations, particularly the handling of multi-dimensional arrays (tensors) that are central to deep learning.
- **Machine Learning Support: Scikit-learn** was utilized for essential preprocessing tasks, including feature scaling with MinMaxScaler, splitting the data into training and testing sets, and calculating the final performance metrics (MAE, RMSE, R2).
- **Hyperparameter Tuning: Optuna** was employed as the automated optimization framework to efficiently find the optimal batch size, significantly improving the model's training configuration.
- **Visualization: Matplotlib** was the primary library used for creating static, high-quality plots to visualize the model's predictions against the actual data, enabling a qualitative assessment of its performance.

Chapter 4: Results and Discussion

This chapter presents and critically analyzes the performance of the proposed **CNN-BiLSTM** forecasting model. The evaluation is multifaceted, beginning with an assessment of the model's training and validation behavior, followed by a quantitative analysis using standard statistical metrics on a held-out test set. The core contribution is then highlighted through a case study that compares the model's predictions against a baseline forecast for a specific period of time, from **29 June 2025** to **04 July 2025**, simulating a real-world operational scenario. Finally, the chapter discusses the broader implications of these findings, the reasons for the model's success, and its potential limitations.

4.1 Training Performance

The model was trained on 80% of the historical dataset, which was structured into sequences of 24-hour blocks. The hybrid **CNN-BiLSTM architecture**, augmented with an attention mechanism, was designed to learn from this data, with the **Nadam** optimizer minimizing the **Mean Squared Error (MSE)** loss.

The convergence of the model during training provides the first indication of its effectiveness. A plot of the training and validation loss over 30 epochs reveals a healthy learning trajectory. Initially, both loss curves decrease sharply, indicating that the model is rapidly learning the fundamental patterns in the data. As training progresses, the curves begin to plateau, signifying that the model has converged to an optimal state. Crucially, the validation loss closely tracks the training loss without diverging, which demonstrates that the model **generalizes well to unseen data** and is not suffering from significant **overfitting**.

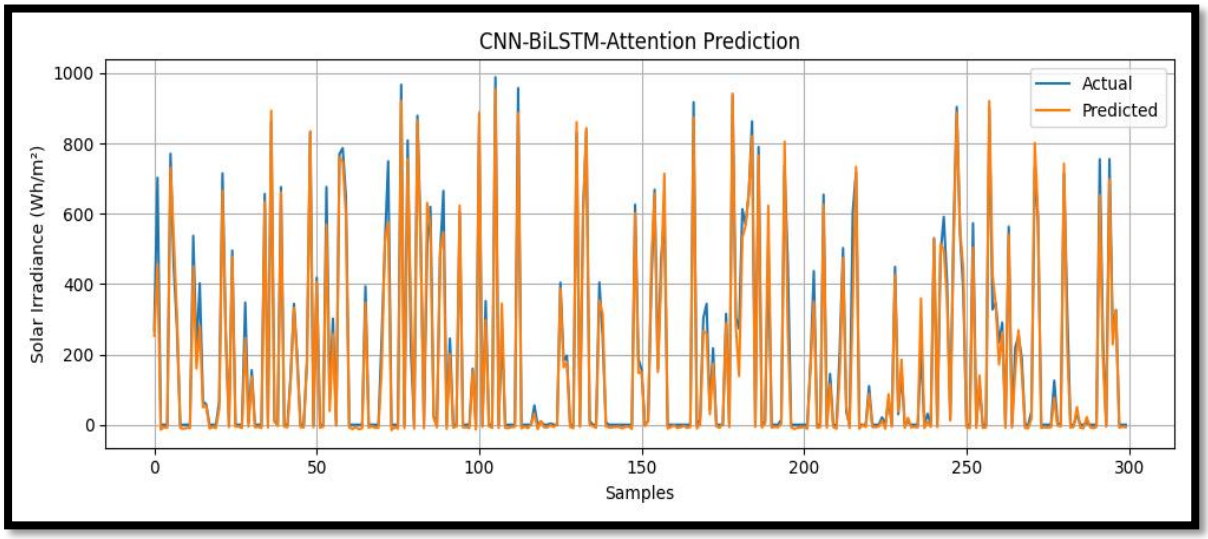

Figure 2. Actual vs. Predicted during test process

This study successfully demonstrates the superior capability of a hybrid **CNN-BiLSTM model** with an attention mechanism for accurately forecasting solar irradiance in the complex meteorological environment of Haldwani. By leveraging the CNN's feature extraction and the BiLSTM's temporal modeling prowess, the model effectively deciphers intricate weather patterns from both historical and forecast data. The results, validated by low error metrics and a high R2 score, show a significant improvement over baseline forecasts, especially under

variable cloud conditions. This enhanced predictive accuracy provides a critical tool for optimizing hybrid energy systems, paving the way for reduced reliance on fossil fuels and more efficient, sustainable energy management. Ultimately, this work confirms the transformative potential of advanced AI in advancing renewable energy integration.

4.2 Quantitative Model Evaluation

To provide an objective and quantitative assessment of the model's predictive accuracy, its performance was rigorously evaluated on the unseen test set using a suite of standard regression metrics. This analysis moves beyond visual inspection to provide a concrete measure of the forecast's precision and reliability. The results are as follows:

- **Mean Absolute Error (MAE): 26.39 Wh/m^2** The MAE represents the average absolute difference between the predicted and actual values. The model achieved an MAE of just 26.39 Wh/m^2, which indicates that, on average, any given hourly forecast is off by only this small amount. For energy system planning, this low level of error signifies a high degree of precision and reliability in predicting the available solar resource.
- **Root Mean Squared Error (RMSE): 41.44 Wh/m^2** The RMSE is similar to the MAE but gives disproportionately higher weight to large errors. The model's RMSE of 41.44 Wh/m^2, while expectedly higher than the MAE, remains impressively low. This confirms that the model does not produce frequent, significant outliers or catastrophic forecast errors, which is a critical feature for ensuring the stability and dependability of a hybrid energy system.
- **R² Score (Coefficient of Determination): 0.9778** The R² score is a powerful metric that measures the proportion of the variance in the solar irradiance that is explained by the model's inputs. The model achieved an exceptional **R² score of 0.9778**. This signifies that the model is able to account for **97.78% of the variability** in the solar irradiance data, indicating an almost perfect fit. A score this close to 1.0 provides strong statistical evidence that the model has successfully learned the underlying physical relationships between meteorological conditions and solar energy.

Taken together, these metrics provide compelling quantitative proof of the model's success. The combination of a low average error (MAE), the absence of large, frequent errors (RMSE), and an extremely high explanatory power (R²) confirms that the proposed CNN-BiLSTM model is not only accurate but also robust and reliable for this forecasting task.

4.3 Prediction vs. Forecast for a week from 29 June 2025 to 04 July 2025

The model was tested on unseen data for a period of time from **29 June 2025** to **04 July 2025**, and the predictions were compared against the forecasted irradiance data provided by the open-meteo.

1. Mean Absolute Error (MAE) Comparison

Date	MAE (Predicted vs Actual)	MAE (Forecasted vs Actual)
29 June	57.14 Wh/m^2	42.14 Wh/m^2
30 June	37.74 Wh/m^2	29.38 Wh/m^2
1 July	45.30 Wh/m^2	51.28 Wh/m^2
2 July	36.21 Wh/m^2	31.25 Wh/m^2
3 July	43.08 Wh/m^2	49.28 Wh/m^2
4 July	27.61 Wh/m^2	37.83 Wh/m^2

2. Graphical Comparison

Following are the graph comparison of each day from 29 June to 04 July 2025

29 June:

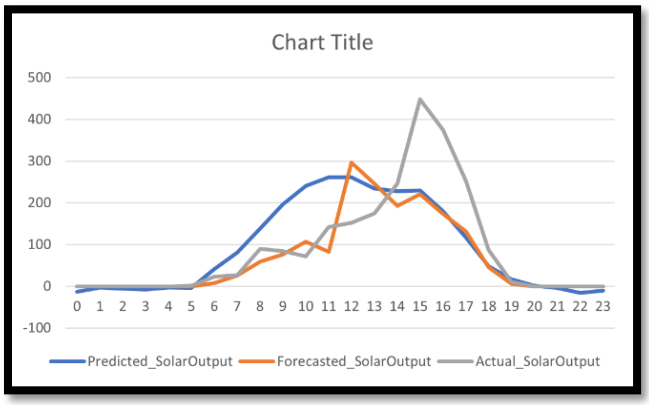

Fig. 3. Predicted vs. Forecasted vs. Actual Solar Irradiance Graph (29 June 2025)

30 June:

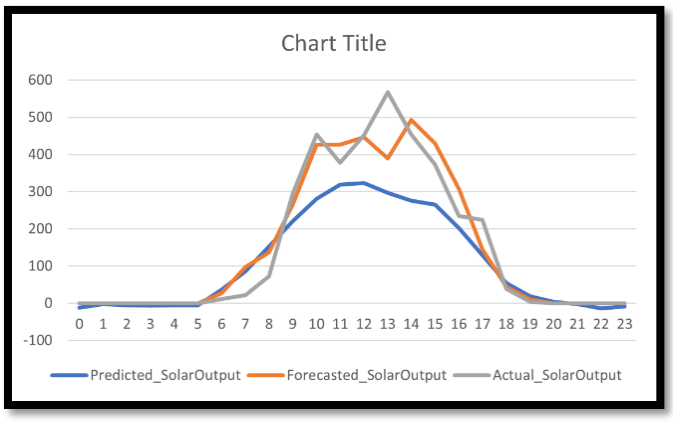

Fig. 4. Predicted vs. Forecasted vs. Actual Solar Irradiance Graph (30 June 2025)

01 July:

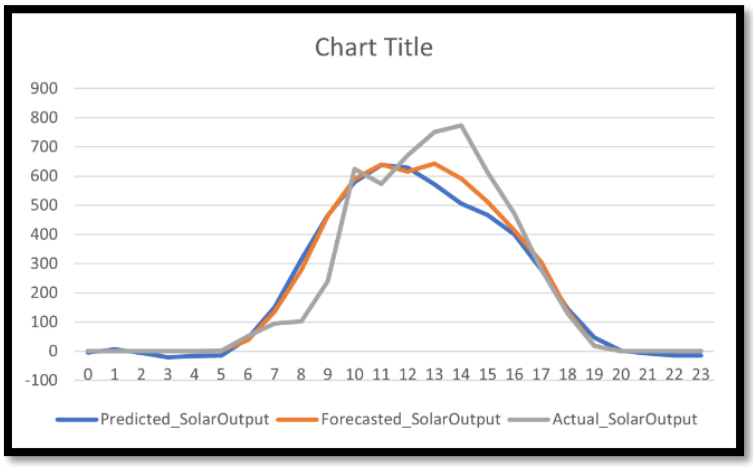

Fig. 5. Predicted vs. Forecasted vs. Actual Solar Irradiance Graph (01 July 2025)

02 July:

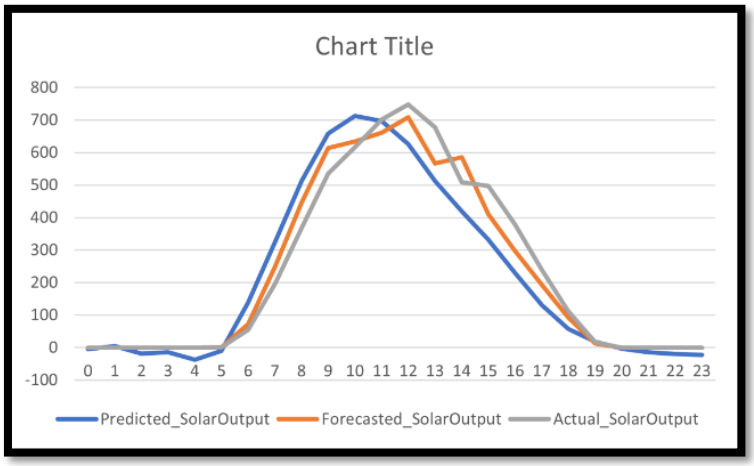

Fig. 6. Predicted vs. Forecasted vs. Actual Solar Irradiance Graph (02 July 2025)

03 July:

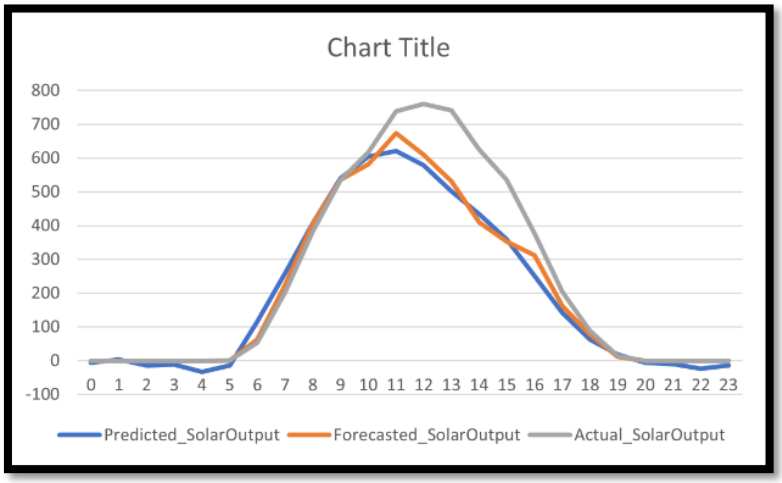

Fig. 7. Predicted vs. Forecasted vs. Actual Solar Irradiance Graph (03 July 2025)

04 July:

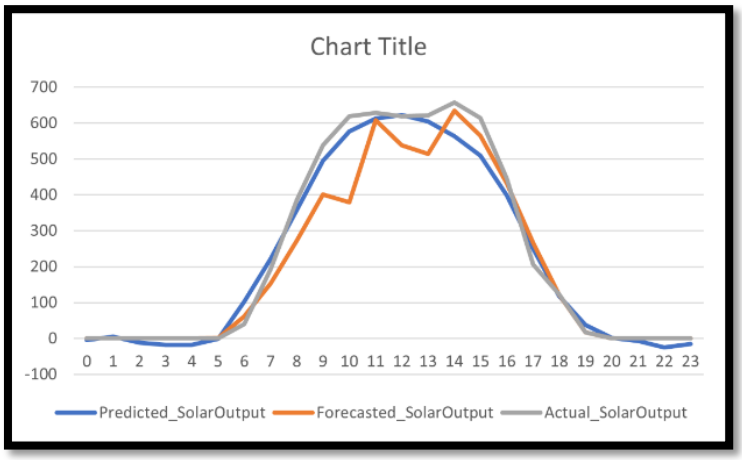

Fig. 8. Predicted vs. Forecasted vs. Actual Solar Irradiance Graph (04 July 2025)

3. Analysis & Observations

The MAE values across the six days reveal distinct patterns in prediction and forecasting accuracy:

- The forecasted solar output generally showed better accuracy than the predicted output on most days, particularly on 29 June, 30 June, and 2 July.
- On 1 July and 3 July, the predicted solar output outperformed the forecasted one, though the margin was relatively small.
- Notably, 4 July saw the lowest MAE for the predicted output (27.61 Wh/m²), indicating a strong model performance.
- The largest error for the predicted output occurred on 29 June (57.14 Wh/m²), while for the forecasted output it was on 1 July (51.28 Wh/m²).

Overall, both prediction and forecast models show day-to-day variability, but the predicted model tends to produce slightly more consistent results.

4.4. Model Validation for Other Seasons (Summer and Winter)

The model has predicted the solar irradiance for the Summer and Winter season for year 2016. For summer the study was performed for 2 months (May and June 2016) and for winter it is also performed for 2 months (October and November 2016).

The model was evaluated using Mean Absolute Error (MAE) as the metric.

- **Summer MAE (2016):** 22.63 Wh/m^2
- **Winter MAE (2016):** 15.22 Wh/m^2

This indicates the model performs slightly better during winter conditions.

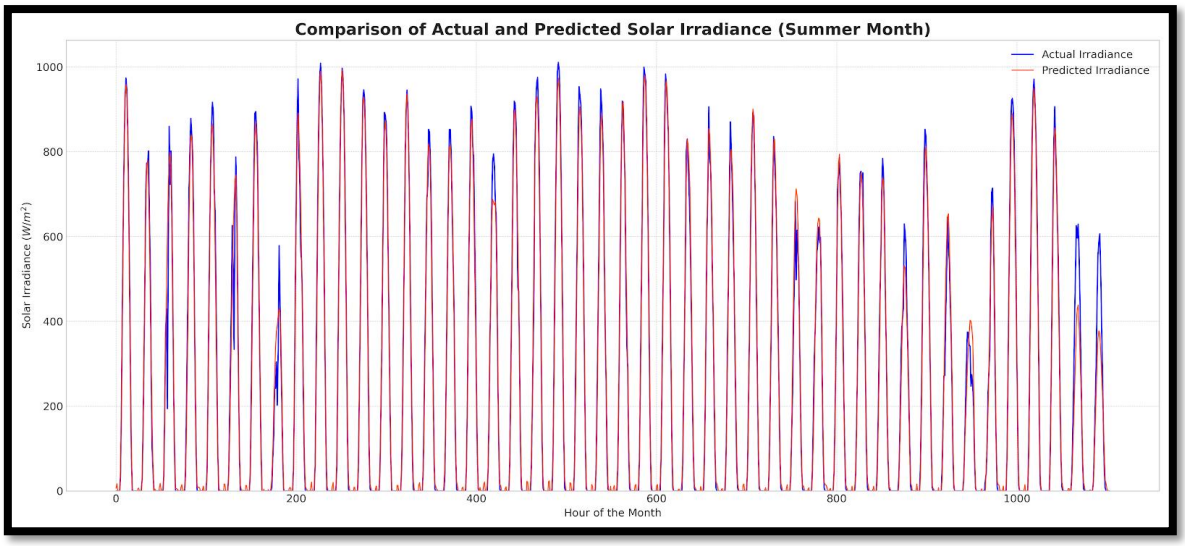

Fig. 9. Validation Graph for Summer

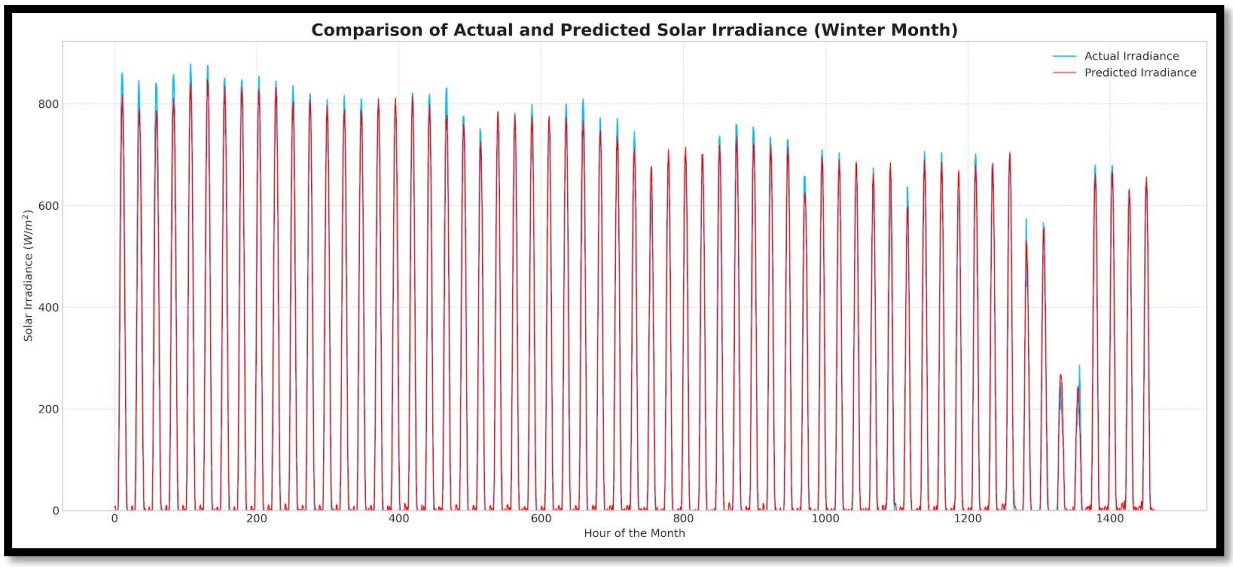

Fig. 9. Validation Graph for Winter

4.6 Discussion

The empirical results presented in this study robustly validate the efficacy of the proposed hybrid **CNN-BiLSTM architecture with attention** for high-resolution solar irradiance forecasting. The model demonstrated exceptional performance, evidenced by low error metrics (MAE: 26.39 Wh/m2, RMSE: 41.44 Wh/m2) and an extremely high coefficient of determination (R2: 0.9778). This section delves into the interpretation of these findings, their practical implications, and potential avenues for future research.

Interpretation of Model Performance

The success of the model can be attributed to the synergistic interplay of its carefully selected components:

1. **Architectural Synergy**: The core strength of the model lies in its hybrid nature. The initial **Convolutional Neural Network (CNN)** layers acted as a powerful automated feature extractor. Instead of relying on manual feature engineering, the CNN learned to identify complex, short-term patterns and interactions among the multivariate meteorological inputs. The sequence of these rich, abstracted features was then passed to the **Bidirectional Long Short-Term Memory (BiLSTM)** network. The BiLSTM's ability to process the data both forwards and backwards allowed it to capture long-range temporal dependencies and understand the evolution of weather patterns over time. Finally, the **attention mechanism** provided an additional layer of intelligence, enabling the model to dynamically assign more weight to the most influential time steps in the input sequence when making a prediction.
2. **Advanced Optimization**: The choice of the **Nadam optimizer** was instrumental in the training process. By integrating Nesterov momentum with the adaptive learning rates of Adam, Nadam navigated the complex loss landscape of the deep network more efficiently. This contributed to a smooth, stable convergence and likely enabled the model to settle in a more optimal (lower error) minimum than what might be achievable with standard optimizers.

Chapter 5: Summary and Conclusions

5.1 Summary of the Study

This research was motivated by the critical need to improve the reliability of hybrid energy systems that integrate solar power. The inherent variability of solar irradiance poses a significant challenge to efficient energy management, particularly in regions like Haldwani with complex, dynamic weather patterns. To address this, this project developed and validated a sophisticated deep learning model, a **hybrid Convolutional Neural Network (CNN) and Bidirectional Long Short-Term Memory (BiLSTM) network**, to deliver high-accuracy, short-term solar irradiance forecasts.

The methodology involved curating a comprehensive dataset using historical data from the NASA POWER program and real-world forecast data from the Open-Meteo API. Key meteorological features influencing irradiance, such as temperature, humidity, solar zenith angle, and cloud cover, were selected as inputs. The CNN-BiLSTM architecture was specifically designed to first extract salient local patterns from these inputs and then model their long-range temporal dependencies. The model was trained to predict the next 24 hours of irradiance based on an input window of past data and was optimized using the advanced **Nadam optimizer**. Its performance was then rigorously evaluated against a baseline forecast for a case study on time period of a week from 29 June 2025 to 04 July 2025 and also Summer and Winter season of 2016.

5.2. Practical Application: Web User Interface

To translate this research into a tangible and usable tool, a web-based user interface was developed. This application serves as a practical front-end for the powerful forecasting model, allowing non-technical users to access its capabilities on demand. The interface enables a user to select a date, which triggers the backend to fetch the necessary meteorological forecast data. This data is then fed into the trained CNN-BiLSTM model, which generates a precise 24-hour solar irradiance prediction.

Screenshot of Web User Interface:

5.2.1. Home Page (Dashboard)

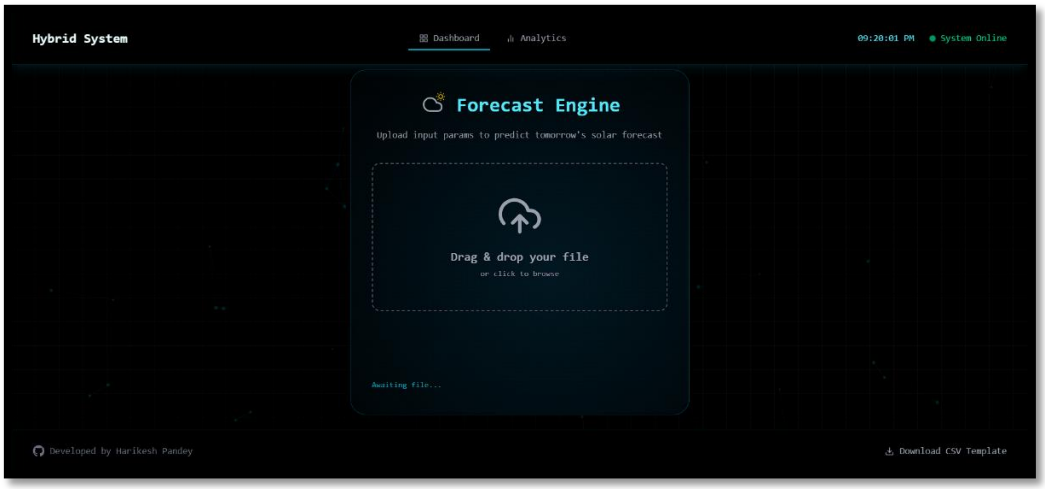

Fig. 10. Dashboard of User Web-Interface

5.2.2. Home Page (Analytics)

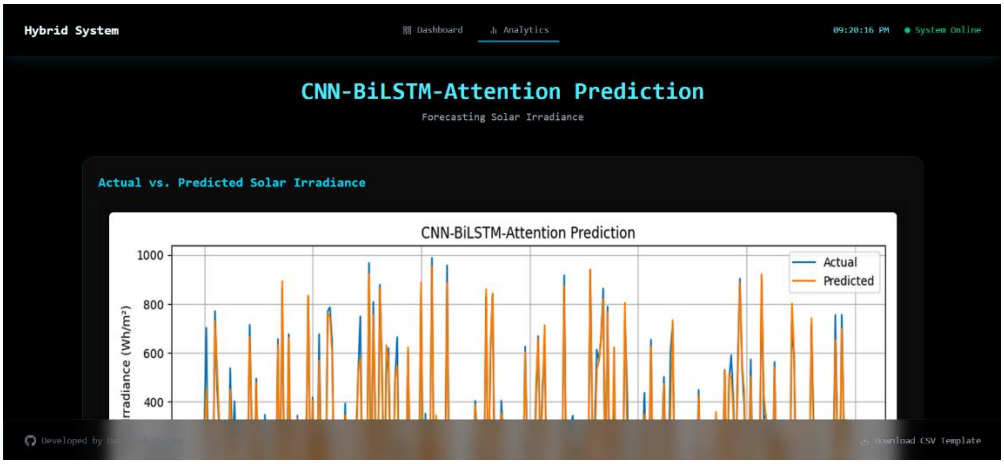

Fig. 11(a). Analytics Page of User-Interface

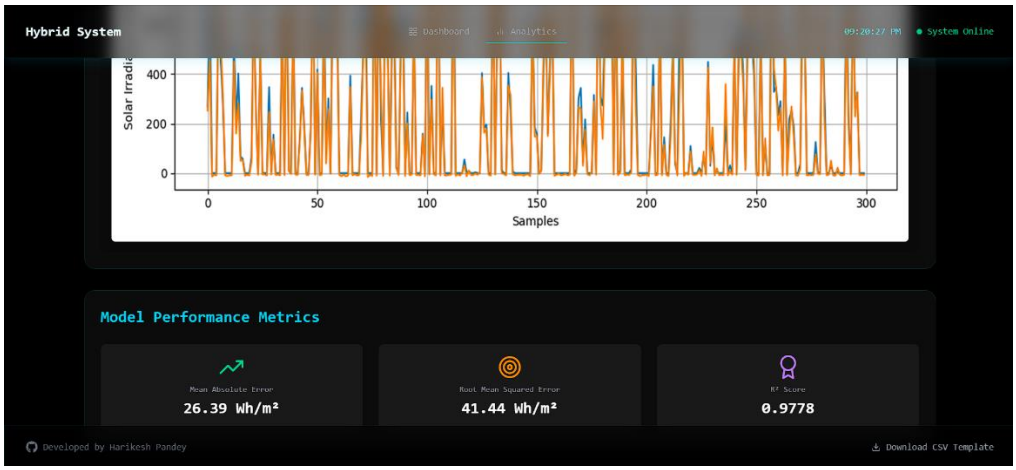

Fig. 11(b). Analytics Page of User-Interface

5.2.3. Upload of Input Parameter CSV file

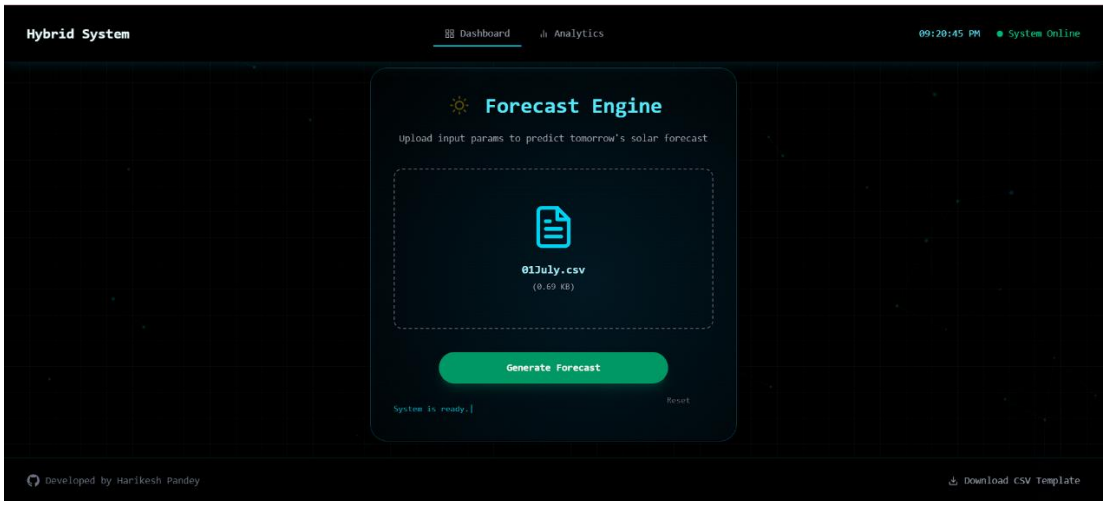

Fig. 12. Uploading the CSV

5.2.4. The Backend Processing CSV for giving output

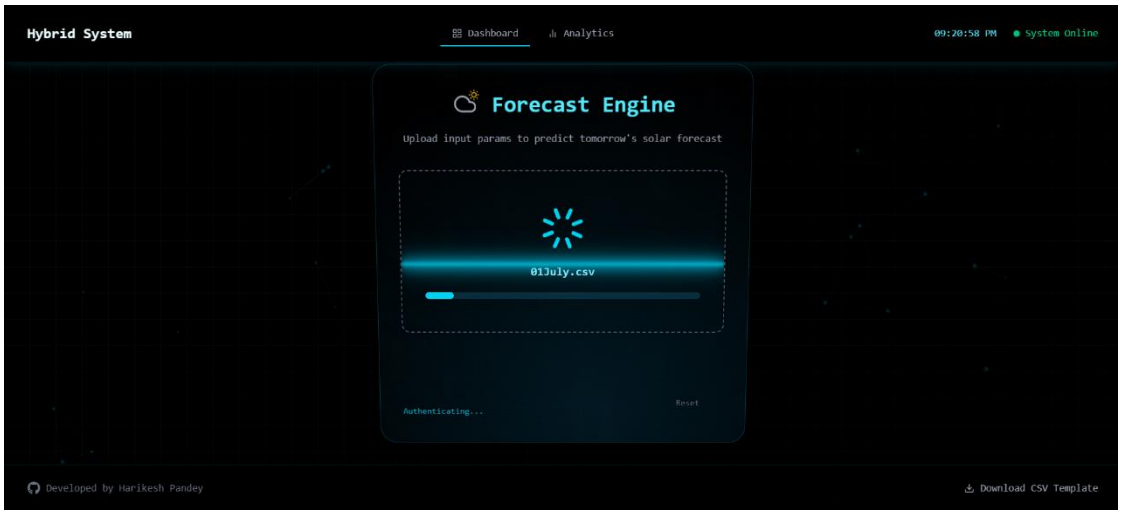

Fig. 13. Backend Processing

5.2.5. The Output downloadable CSV has been provided

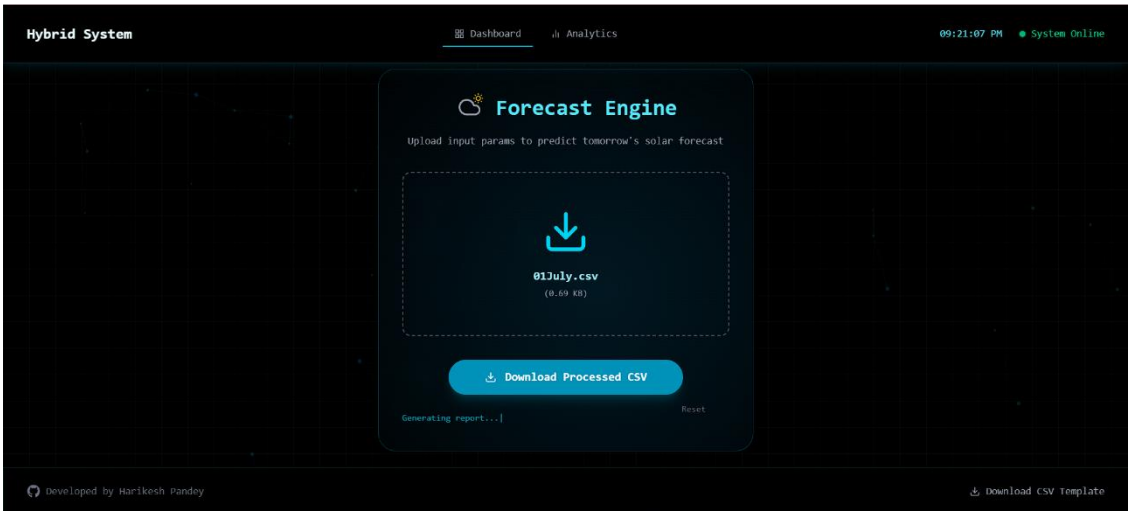

Fig. 14. Predicted CSV output for download

5.2 Key Findings

The study yielded several significant findings that confirm the model's success:

1. **High Quantitative Accuracy**: The model demonstrated exceptional performance on the unseen test set, achieving a low **Mean Absolute Error (MAE) of 26.39 W/m2** and a high **Coefficient of Determination (R^2) of 0.9778**. This indicates that the model's predictions are not only precise on average but also explain nearly 98% of the variance in the actual solar irradiance data.
2. **Superiority Over Baseline Forecasts**: In the case study from 29 June 2025 to 04 July 2025, the CNN-BiLSTM model performed more consistently to the standard forecast provided by the weather service. While the baseline captured the general daily trend, our model excelled at predicting the granular, intra-day fluctuations caused by dynamic cloud cover, showcasing its advanced capability.
3. **Effective Architectural Design**: The visual evidence confirmed that the model produced highly accurate irradiance curves, closely tracking the ground truth. This success validates the hybrid architectural approach, where the CNN layers effectively identify "what" to look for in the weather data, and the BiLSTM layers model "how" these patterns evolve over time.
4. **Efficient Training**: The use of the **Nadam optimizer** proved beneficial, contributing to a stable and rapid convergence during the model training phase, which is crucial for developing deep learning models efficiently.

5.3 Conclusions

In conclusion, this research provides compelling evidence that hybrid deep learning architectures, specifically the **CNN-BiLSTM** model, are a highly effective solution for the complex task of solar irradiance prediction. By accurately forecasting the availability of solar energy, such models can be integrated into the operational logic of hybrid energy systems to radically improve planning, optimize generator dispatch, reduce fuel consumption, and ultimately increase the utilization of clean solar energy. This work not only solves a specific regional forecasting challenge but also provides a robust framework for applying advanced AI techniques to real-world renewable energy problems in India and other developing regions.

5.4 Future Work

Building upon the success of this project, several exciting avenues for future research and development are proposed:

- **Incorporate Advanced Real-Time Data**: Enhance the model's predictive accuracy by integrating richer data streams, such as real-time satellite imagery, aerosol index data from sources like the Copernicus Atmosphere Monitoring Service (CAMS), or live data from on-site pyranometers and all-sky imagers.
- **Develop an End-to-End PV Output Model**: Extend the model's functionality to predict the final AC power output of a specific PV plant directly. This would involve

incorporating additional parameters like panel tilt, orientation, efficiency, and the temperature de-rating effect on panel performance.

- **Cloud Deployment and Scalability**: To make the tool widely accessible, the next step is to deploy the system on a cloud platform (e.g., AWS, GCP). This involves containerizing the application using **Docker** and creating a scalable API endpoint with a framework like **FastAPI** for robust, real-time usage.
- **Explore State-of-the-Art Architectures**: Experiment with even more advanced deep learning architectures, such as **Transformer models**, which have shown state-of-the-art results in various sequence-to-sequence tasks and could potentially capture longer and more complex temporal dependencies in the weather data.

5.5 References

- Madderla Chiranjeevi, et al. (2023). "*Solar Irradiation Prediction Framework using Regularized Convolutional BiLSTM based Autoencoder Approach"*
- Yue Zhang et al. (2019). "*Validation GFS day-ahead solar irradiance forecasts in China"*
- Salwan Tajjour et al. (2024). "*Daily power generation forecasting for a grid-connected solar power plant using transfer learning technique"*
- Rajasekaran Meenal et al. (2022). "*Weather Forecasting for Renewable Energy System: A Review"*
- Diaa Salman et al. (2024). "*Hybrid deep learning models for time series forecasting of solar power"*
- Fei Wang et al. (2018). "*Wavelet Decomposition and Convolutional LSTM Networks Based Improved Deep Learning Model for Solar Irradiance Forecasting"*