Problem Statement :   
Improving solar irradiance forecasting for Hybrid Solar-Generator System in Haldwani.

Abstract :

The model made in CNN-BiLSTM model able to predict the solar irradiance much accurately than forecast model by weather site(open-meteo).

Haldwani has sub-tropical climate which results in varied climate result in hard to predict the weather. The longer the prediction the higher uncertainty it will achieve.

Data Collection :

The data has been collected from NASA POWER program(power.larc.nasa.gov).The data has been collected for Haldwani (LAT : 29.1863 , LONG : 79.5186) for 7 years from 7 October 2017 to 30 July 2024. Got 59731 entries for that .

The attributes taken are :

1. Cloud Amount (Cloud Percentage)

2. Original Recorded Solar Irradiance (Wh/m2)

3. Temperature at 2 Meters (degree Celsius)

4. Dew/Frost Point at 2 Meters (degree Celsius)

5. Relative Humidity at 2 Meters (%)

6. Solar Zenith Angle (Degrees)

The data is taken from renewable energy committee for hourly temporal regions.

Input Features for Model :

1. Month

2. Day

3. Hour

4. Temp2m → Air temperature at 2m

5. Humidity2m → Humidity at 2m

6. Cld\_Amt → Cloud percentage in sky

7. Solar Zenith

Output of Model:

SolarIraadiance

Data CSV:

The CSV Looks Like This:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| YEAR | MO | DY | HR | CloudCover | SolarIrradiance | Temp2m | DewPoint2m | Humidity2m | SolarZenith | Forecasted\_SW\_DWN |
| 2017 | 10 | 7 | 0 | 33.98 | 0 | 22.68 | 18.77 | 78.74 | 90 | 0 |
| 2017 | 10 | 7 | 1 | 9.34 | 0 | 22.6 | 18.6 | 78.25 | 90 | 0 |
| 2017 | 10 | 7 | 2 | 31.49 | 0 | 22.42 | 18.45 | 78.42 | 90 | 0 |
| 2017 | 10 | 7 | 3 | 49 | 0 | 22.12 | 18.31 | 79.23 | 90 | 0 |
| 2017 | 10 | 7 | 4 | 51.58 | 0 | 21.83 | 18.17 | 79.96 | 90 | 0 |
| 2017 | 10 | 7 | 5 | 33.35 | 0 | 21.63 | 17.99 | 79.97 | 90 | 0 |
| 2017 | 10 | 7 | 6 | 5.22 | 138.3 | 22.68 | 19.35 | 81.8 | 79.26 | 140.07 |
| 2017 | 10 | 7 | 7 | 5.78 | 353.42 | 26.37 | 20.18 | 68.99 | 67.37 | 357.7 |
| 2017 | 10 | 7 | 8 | 11.38 | 563.88 | 30.62 | 17.58 | 45.89 | 55.7 | 570.15 |
| 2017 | 10 | 7 | 9 | 19.14 | 725.28 | 32.3 | 16.61 | 39.23 | 45.53 | 736.88 |
| 2017 | 10 | 7 | 10 | 7.22 | 828.72 | 33.15 | 16.96 | 38.15 | 38.13 | 841.4 |
| 2017 | 10 | 7 | 11 | 24.81 | 832.5 | 32.7 | 18.22 | 42.35 | 35.34 | 854.72 |
| 2017 | 10 | 7 | 12 | 34.44 | 795.67 | 32.46 | 19.05 | 45.18 | 38.19 | 823.67 |
| 2017 | 10 | 7 | 13 | 19.13 | 664.4 | 32.32 | 19.17 | 45.88 | 45.63 | 693.47 |
| 2017 | 10 | 7 | 14 | 27.16 | 507.25 | 32.04 | 19.5 | 47.57 | 55.82 | 532.17 |
| 2017 | 10 | 7 | 15 | 27.05 | 306.05 | 32.28 | 18.4 | 43.84 | 67.51 | 327.95 |
| 2017 | 10 | 7 | 16 | 6.9 | 121.7 | 30.38 | 18.67 | 49.69 | 79.46 | 122.47 |
| 2017 | 10 | 7 | 17 | 8.29 | 0 | 27.14 | 21.33 | 70.75 | 90 | 0 |
| 2017 | 10 | 7 | 18 | 8.23 | 0 | 26.21 | 19.75 | 67.72 | 90 | 0 |
| 2017 | 10 | 7 | 19 | 5.7 | 0 | 25.89 | 18.91 | 65.48 | 90 | 0 |
| 2017 | 10 | 7 | 20 | 8.21 | 0 | 25.58 | 18.19 | 63.78 | 90 | 0 |
| 2017 | 10 | 7 | 21 | 7.5 | 0 | 25.08 | 17.7 | 63.71 | 90 | 0 |
| 2017 | 10 | 7 | 22 | 11.35 | 0 | 23.89 | 17.87 | 69.15 | 90 | 0 |
| 2017 | 10 | 7 | 23 | 11.35 | 0 | 22.9 | 18 | 74.01 | 90 | 0 |
| 2017 | 10 | 8 | 0 | 13.24 | 0 | 22.34 | 17.93 | 76.15 | 90 | 0 |
| 2017 | 10 | 8 | 1 | 8.43 | 0 | 22.39 | 17.66 | 74.72 | 90 | 0 |
| 2017 | 10 | 8 | 2 | 9.45 | 0 | 22.23 | 17.57 | 75.03 | 90 | 0 |
| 2017 | 10 | 8 | 3 | 11.35 | 0 | 22.13 | 17.46 | 75.03 | 90 | 0 |
| 2017 | 10 | 8 | 4 | 13.22 | 0 | 22.01 | 17.36 | 75.07 | 90 | 0 |
| 2017 | 10 | 8 | 5 | 10.71 | 0 | 21.71 | 17.32 | 76.26 | 90 | 0 |
| 2017 | 10 | 8 | 6 | 15.73 | 113.43 | 22.61 | 18.78 | 79.23 | 79.43 | 118.03 |

And rest of the data .

Model :

Using this model has been created   
  
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

# === 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")

# === 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)

# === 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

# === Optuna Tuning (fast: 8 trials only) ===

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=30,

batch\_size=batch\_size,

verbose=0,

)

return history.history["val\_loss"][-1]

study = optuna.create\_study(direction="minimize")

study.optimize(objective, n\_trials=20)

# === 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=20, 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}")

# === Plot First Sample ===

plt.figure(figsize=(12, 5))

plt.plot(y\_actual[0], label="Actual")

plt.plot(y\_pred[0], label="Predicted")

plt.title("24-Hour Solar Irradiance Prediction")

plt.xlabel("Hour")

plt.ylabel("Irradiance (Wh/m²)")

plt.legend()

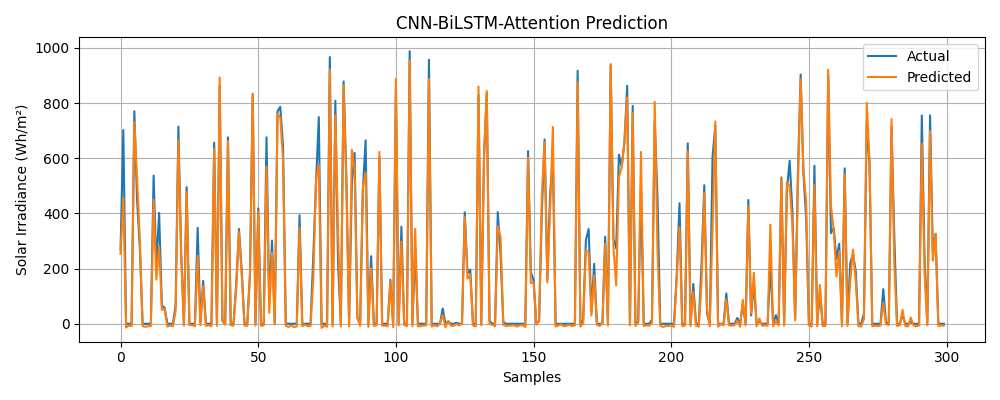
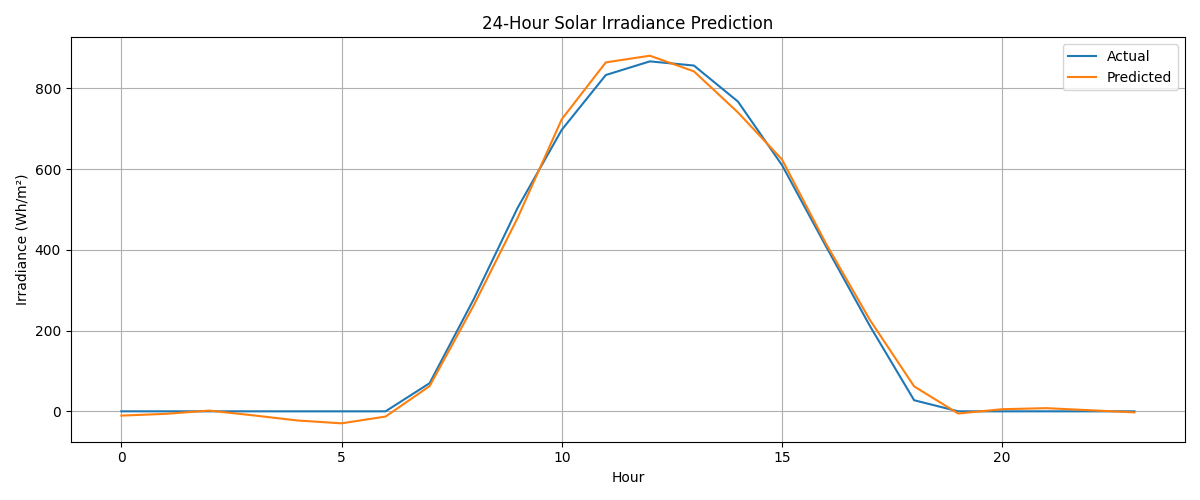
plt.grid(True)

plt.tight\_layout()

plt.show()

# === Save Final Model ===

final\_model.save("cnn\_bilstm\_attention\_24hr\_model.keras")

print("\nModel and scalers saved successfully.")  
  
The figures model provided :  
  
  
  
  
  
The following performance matrices achieved :  
MAE : 26.14 Wh/m2

RMSE : 41.44 Wh/m2

R2 : 0.9778