```
% ======================
% Solar Home Energy Management System Forecasting using LSTM (Optimized for Speed)
% ======================
clc; clear; close all;

%% 1️⃣ Load and Preprocess Data

% Load Fixed Load Data File
opts = detectImportOptions('Final_Fixed_Load_Data.xlsx', 'VariableNamingRule',
'preserve');
opts = setvaropts(opts, 'Date', 'Type', 'char'); % Read Date as text
loadData = readtable('Final_Fixed_Load_Data.xlsx', opts);

% Load Other Data Files
solarData = readtable('Final_Corrected_Solar_Data.csv');
batteryData = readtable('Final_Battery_Data.csv');

%% 2️⃣ Convert Date Columns to Proper Format

% Debug: Display First Few Date Entries
disp("First 5 Dates in Load Data:"); disp(loadData.Date(1:5));

% Convert Load Data Date Column
try
    loadData.Date = datetime(loadData.Date, 'InputFormat', 'dd-MMM-yyyy
HH:mm:ss.SSS', 'Format', 'yyyy-MM-dd');
catch
    disp("⚠️Warning: Default format failed! Trying alternative formats...");
    formats = {'yyyy-MM-dd', 'dd-MMM-yyyy HH:mm:ss.SSS', 'MM/dd/yyyy HH:mm:ss',
'dd/MM/yyyy HH:mm:ss'};
    for i = 1:length(formats)
        try
            loadData.Date = datetime(loadData.Date, 'InputFormat', formats{i});
            disp(['✅Date conversion successful with format: ', formats{i}]);
            break;
        catch
            continue;
        end
    end
end

% Convert Dates in Solar and Battery Data
solarData.Date = datetime(solarData.Date, 'InputFormat', 'yyyy-MM-dd', 'Format',
'yyyy-MM-dd');
batteryData.Date = datetime(batteryData.Date, 'InputFormat', 'yyyy-MM-dd',
'Format', 'yyyy-MM-dd');

%% 3️⃣ Ensure Consistent Data Types for Merging

% Convert any cell array to datetime
if iscell(loadData.Date)
    loadData.Date = datetime(loadData.Date, 'InputFormat', 'yyyy-MM-dd');
end

% Convert to table format for consistency
loadData = table(loadData.Date, loadData.("Load (kW)"), 'VariableNames', {'Date',
'Load_kW'});

%% 4️⃣ Merge Data Based on Date
```

```
data = outerjoin(solarData, batteryData, 'Keys', 'Date', 'MergeKeys', true);
data = outerjoin(data, loadData, 'Keys', 'Date', 'MergeKeys', true);

disp("✅ Data Successfully Merged!");

%% 5️⃣ Feature Engineering

% Compute Features
data.Solar_Power = data.GHI * 0.18 * 10; % Estimated solar power
data.Battery_SOC = (data.Watt_hr / (10 * 1000)) * 100; % Battery SoC (%)
data.Load_Demand = data.Load_kW; % Load demand in kW

% Normalize Data for ML Model
X = normalize([data.Solar_Power, data.Battery_SOC, data.Load_Demand]);

%% Debug Step: Check Data Before Removing NaNs
disp("Initial Size of X:"); disp(size(X));

% Fill NaN values instead of removing them
X = fillmissing(X, 'linear'); % Interpolate missing values

disp("Size of X after filling NaNs:"); disp(size(X));

%% 6️⃣ Optimize Dataset for Faster Training
sample_size = min(10000, size(X, 1)); % Use first 10,000 samples (or all if less)
X = X(1:sample_size, :);

%% Check if Data is Empty Before Training
if isempty(X)
    error("⚠️ Error: No valid data available for training. Please check the
dataset.");
end

% Define Training & Target Variables
X_train = X(1:end-10, :); % Training features (excluding last 10 days)
Y_train = X(2:end-9, :); % Target values (next day forecast)

% Ensure X_train and Y_train are not empty
if isempty(X_train) || isempty(Y_train)
    error("⚠️ Error: X_train or Y_train is empty. Data preprocessing might have
removed too many rows.");
end

%% 7️⃣ Define Optimized LSTM Model (Faster Training)

layers = [
    sequenceInputLayer(3) % 3 input features
    lstmLayer(25, 'OutputMode', 'sequence') % Reduce neurons for faster training
    fullyConnectedLayer(3) % Output layer
    regressionLayer];

%% 8️⃣ Faster Training Options (Use GPU If Available)

%% Check GPU availability (Fix for GPU Error)
try
    gpuInfo = gpuDevice(); % Try detecting GPU
    executionEnv = 'gpu';
    disp("✅ Using GPU for training.");
catch
```

```
    executionEnv = 'cpu'; % If error occurs, switch to CPU
    disp("⚠GPU not available. Using CPU (training may be slower).");
end


options = trainingOptions('adam', ...
    'MaxEpochs', 25, ... % Fewer epochs for speed
    'MiniBatchSize', 64, ... % Larger batch size
    'ExecutionEnvironment', executionEnv, ... % Auto-select CPU or GPU
    'Verbose', false);

%% 9 Train the LSTM Model
net = trainNetwork(X_train', Y_train', layers, options);

%%  Make Predictions for Future Values

X_test = X(end-9:end-1, :); % Last 10 days as input
Y_pred = predict(net, X_test')'; % Predict future values

% Convert Predictions Back to Original Scale
predictedSolar = Y_pred(:,1);
predictedSOC = Y_pred(:,2);
predictedLoad = Y_pred(:,3);

%% 11 Plot the Results

figure;
subplot(3,1,1);
plot(data.Date(end-9:end-1), predictedSolar, 'r', 'LineWidth', 2);
title('Predicted Solar Power');

subplot(3,1,2);
plot(data.Date(end-9:end-1), predictedSOC, 'b', 'LineWidth', 2);
title('Predicted Battery SOC');

subplot(3,1,3);
plot(data.Date(end-9:end-1), predictedLoad, 'g', 'LineWidth', 2);
title('Predicted Load Demand');

disp('✅Forecasting Completed Successfully!');
```