DESIGN OF A MATLAB GUI FOR SHORT-TERM SOLAR FORECASTING BASED ON DEEP LEARNING

Received Sept. 13, 2023; accepted Sept. 20, 2023
Available online Sept. 30, 2023

D. Matushkin¹, A. Bosak²

Author for correspondence: Dmytro Matushkin,
e-mail: dm.mtkn@ill.kpi.ua

In energy systems, it is crucial to forecast solar energy generation for optimization of operations and to mitigate the impact of uncertainty. Forecasting solar energy involves predicting solar irradiance, for which historical solar irradiance and weather parameter data are typically required. However, such data are often unavailable for residential and commercial solar microgrids. This study proposes an hourly forecasting model for next-day Solar Power Production (SPP) that doesn’t rely on historical solar irradiance data. The forecasting is performed using deep learning techniques like Long Short-Term Memory (LSTM), and the results are integrated into the MATLAB & Simulink simulation platform. A graphical user interface (GUI) within the MATLAB & Simulink software complex is presented as a simulation platform for hourly SPP forecasting. This platform serves as a useful tool for researchers studying energy management and forecasting as well as planning renewable energy-based energy system operations. The study involves predicting the amount of energy generated by Solar Power Production (SPP). The developed GUI was employed to forecast the SPP output power over test days. The forecasted data were then evaluated using Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE). Experimental results demonstrated reasonable accuracy, with the proposed model achieving an RMSE of 0.835 W and an MAE of 0.353 W in certain datasets.

Keywords: short-term forecasting, solar power plant, deep learning, Long Short-Term Memory, graphical user interface, MATLAB & Simulink simulation.

¹PhD student. https://orcid.org/0000-0003-4431-7862
²Cand. of tech. Sciences., Assoc. Prof. https://orcid.org/0000-0003-0545-9980

¹, ²National Technical University of Ukraine «Igor Sikorsky Kyiv Polytechnic Institute», Kyiv, Ukraine.
List of used designations and abbreviations

SES – solar energy systems
ES – Energy System
VRE – variable renewable energy
GUI – graphical user interface
GSR – global solar radiation

SPP – Solar Photovoltaic Power
DL – Deep Learning
ML – Machine Learning
LSTM-RNN – Long Short-Term Memory Recurrent Neural Networks

Introduction. Photovoltaic generation exhibits a rather unstable nature due to its dependence on solar energy, which is influenced by solar radiation and meteorological parameters such as air temperature, humidity, wind speed, etc [1, 2]. These destabilizing fluctuations pose significant challenges for the integration of Solar Energy Systems (SES) into the Energy System (ES). However, the use of Variable Renewable Energy (VRE) contributes to reducing the energy need for balancing and regulating capacity. SES generation is highly flexible, allowing it to be adjusted to adapt to changes in energy demand.

The VRE forecasting significantly impacts various operations for management of the ES, including planning, dispatch, real-time balancing, and reserve requirements for the ES. By incorporating forecasts from local generators, ES operators can predict rapid VRE changes, enabling them to economically balance consumption and scheduled generation on a daily and intraday basis.

In particular, demand-side management [3-4] and generation planning [5-6] become more critical due to research into Smart Microgrids [6-8]. Effective demand-side management and grid operation planning are directly related to the assessment of solar energy. Therefore, the evaluation of solar energy [8-10] becomes a crucial requirement for both grid operation planning [9-10] and research into energy generation planning a day-ahead [11-12].

The volumes of input data and forecast models for different components can vary. This necessitates the development of an adaptive forecasting and planning model system.

It is worth noting that this problem is not reliably and precisely solved. Numerous algorithms and software complexes are proposed, new software products are being developed, but there are no widely accepted “industry standards” for VRE forecasting.

There are two main approaches to solar energy forecasting: direct and indirect. Direct methods obtain solar energy directly by forecasting [13]. In indirect methods, solar irradiation is first forecasted and then converted into solar energy through mathematical relationships [14-15]. This method is essential for planners and researchers who lack solar energy data. In this case, solar irradiation data can be used for indirect solar energy forecasting.

Graphical user interface (GUI) tools allow for solving solar energy forecasting and planning problems using straightforward methods. So far, several GUI tools have been developed, such as the monthly global solar radiation forecasting model [16], a comprehensive photovoltaic simulation model [17], a model for forecasting photovoltaic power for the day ahead [18], and a model for studying the efficiency of a solar tower power plant [19].

In this study, a MATLAB GUI was developed based on existing user interface designs presented in the literature for forecasting photovoltaic power.

This research can be summarized as follows:

- Solar energy data for the day ahead were forecasted using 10-minute global solar radiation (GSR) data from July 1, 2020, to December 31, 2020;
- The developed MATLAB GUI model allows the user to allow the user to forecast the generated Solar Photovoltaic Power (SPP) in graphical form.

Various methodologies for generating probabilistic solar energy forecasts are extensively discussed in the literature. For instance, the work discusses nearest neighbor methods [20], vector autoregressive models [21], volatility estimation methods [22], and ensemble models [23]. Additionally, there are examples of probabilistic solar energy forecasts using Deep Learning (DL) methods [24]. One of the main advantages of the latter is their ability to extract simple features from high-dimensional, complex data [25], making them suitable for forecasting tasks.

Furthermore, DL models are the most suitable approach for forecasting solar irradiation, especially when dealing with complex and large datasets. DL models have been successfully applied in various domains, including image processing, classification, and forecasting, due to their
ability to effectively handle complex data without the need for expert evaluation. For instance, a DL model presented in [26] is used for short-term wave energy forecasting. Similarly, DL models are widely employed for various forecasting tasks such as wind speed [27], PV power [28], solar irradiation [29, 43], and more. The known issues associated with conventional neural networks (CNN), such as gradient vanishing and training complexity, can be easily addressed using DL networks. For time series forecasting, a complex neural network is developed in [30], while deep learning is used for forecasting solar irradiation at 30 locations in Turkey in [31]. Hence, DL models are more accurate compared to Machine Learning (ML) models and empirical models in terms of accuracy.

Formulation of the problem. The primary objective of this article is to address the necessity of reliable SESs by ensuring the availability of meteorological weather data for the region where these systems are deployed. Many countries have developed their own forecasting models, which serve as valuable tools for energy planning (EP). However, in Ukraine, there is still a shortage of EP models based on solar irradiation. This work represents the initial attempt to create a GSR model for Solar Photovoltaic Power (SPP) located in the village of Velyka Dymerka in the Kyiv region of Ukraine. Furthermore, this research endeavors to provide a user-friendly interface for the utilization of the developed models.

The central focus of this study is the development of a user interface within MATLAB & Simulink for the purpose of forecasting SPP generation. This interface is designed to be easily accessible and usable by individuals working in the fields of SPP development and performance evaluation.

Within this article, we propose the utilization of deep Long Short-Term Memory Recurrent Neural Networks (LSTM-RNN) to forecast solar irradiation for the upcoming day.

This research represents a preliminary experiment aimed at establishing a starting point for the creation of intelligent solar applications.

Deep Learning. Deep learning (DL) is a subset of the broader family of Machine Learning (ML) methods that employ multiple layers of processing to learn data with several levels of abstraction [32]. It uncovers complex structures within large datasets using the backpropagation algorithm to indicate how the machine should adjust its internal parameters used for computation at each layer from the representation at the previous level [33]. Architectures within DL encompass Feedforward Neural Networks (FFNN), Recurrent Neural Networks (RNN), Deep Belief Networks (DBN), and Restricted Boltzmann Machines (RBM). Among these architectures, FFNN and RNN are the most widely used. Convolutional neural networks (CNN), which are a subset of FFNN, excel at processing images, videos, and audio. Deep Long Short-Term Memory networks (LSTM), which belong to the RNN family, are well-suited for sequential data like text, language, and time series. DL has achieved significant success across various domains. It has outperformed other ML methods in image recognition [34, 35], speech recognition [36, 37], natural language understanding [38], language translation [39, 40], particle accelerator data analysis [41], brain circuit reconstruction [42], etc. DL has achieved significant success across various domains. It has outperformed other ML methods in image recognition [34, 35], speech recognition [36, 37], natural language understanding [38], language translation [39, 40], particle accelerator data analysis [41], brain circuit reconstruction [42], and more.

Long Short Term Memory Recurrent Neural Network (LSTM-RNN). In this research, an LSTM-RNN network was employed for forecasting solar power generation in a SPP. The LSTM-RNN method is a subset of DL, developed to address the issues of vanishing and exploding gradients inherent in simple RNNs [44]. This is achieved through the incorporation of memory cells within its gating mechanism system. Consequently, it generates an additive gradient as opposed to a vanishing gradient, offering a larger gradient magnitude to facilitate the training of LSTM-RNN cells [45]. The cells of LSTM-RNN have three key functions: write to memory, read from memory, and reset memory. LSTM-RNN is closely tied to recurrent gating units, known as “forget gates” [46]. These gates mitigate the challenges of vanishing and exploding gradients during backpropagation by allowing errors to propagate across multiple layers. In essence, LSTM-RNN possesses the capability to learn tasks that require memory of previously learned information, occurring over a certain time span.

The functioning of LSTM-RNN occurs in four steps. In the first step (Fig. 1), the “forget gate” algorithm is executed which determines how much previous information it should retain. Here, new and previous data presented in vector form are passed through a sigmoid function. The sigmoid function confines values between 0 and 1. If the output is 0, the memory cell forgets the previous data. If the output is 1, the data is retained.

Fig. 1. Forget Gate Layer

This step is described by the expression (1) [47, 48].

\[
    f_t = \sigma(W_{xf} \cdot x_t + W_{hf} \cdot h_{t-1} + b_f),
\]

where \( \sigma \) denotes the sigmoid activation function, \( W_{xf} \) is the weight matrix for input \( x_t \), \( x_t \) is the memory cell matrix of the input vector over time \( t \), \( W_{hf} \) is the weight matrix for the inputs \( h_{t-1}, h_{t-1} \) is matrix of previous states, and \( b_f \) is matrix of shift vector.

At the second step (Fig. 2), the “input gate” layer determines how much of this unit should be added to the current state and determines which value will be updated.
The input gate $i_t$ defines which data is stored in the new candidate state of the cell $\tilde{C}_t$ [47, 48]:

$$\tilde{C}_t = f_t \cdot C_{t-1} + i_t \cdot \tanh(W_{xc} \cdot x_t + W_{hc} \cdot h_{t-1} + b_c)$$

(2)

$$i_t = \sigma(W_{xi} \cdot x_t + W_{hi} \cdot h_{t-1} + b_i)$$

(3)

It has two operational functions: sigmoid and hyperbolic tangent ($\tanh$). $\tanh$ is also an activation function. It is used to control values processed by the network, transforming them into a close interval to ensure they always fall within the range of $[-1;1]$. Both the vectors of previous and new data are provided to both functions. The output results are multiplied together (Fig. 3), and the output is passed to the cell state, updating the value of the cell state.

The new candidate state of the cell $\tilde{C}_t$ and the previous candidate state of the cell $C_{t-1}$ are used to update the last state of the cell $C_t$. This step is described by the expression (4) [47, 48].

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t$$

(4)

The final step is the output layer algorithm (Fig. 4).

When the forward pass is completed at time step $t=T$, which is when the last weight coefficient of the sequence is reached, the reverse pass known as Back-Propagation Through Time (BPTT) is initiated. The error gradient is computed iteratively until $t=1$, at this stage, the weights of the LSTM network are updated iteratively using an optimization method known as "gradient descent optimization technique." This training procedure entails adjusting the weights in a way that minimizes the error function. In this study, the root mean square error (RMSE) was utilized, where $(y(k))$ represents the target generation and, accordingly, the output of LSTM-RNN for each training sample $(k)$. The cost of the error function is evaluated based on the network's performance on the data to be predicted. It is computed after each training iteration. In the LSTM-RNN model, there are various gradient descent methods that can be used to work with parameter spaces for learning tasks. Here, the Adam method is used, which is a stochastic optimizer that calculates different individual learning rates for different parameters based on estimated first and second gradients for each epoch ($\theta$).

Design of the MATLAB-based GUI. A GUI was created to facilitate the usage of the developed DL model. The development of a solar irradiance forecasting tool requires us to first develop the DL model and then create a graphical interface based on MATLAB, as shown in Fig. 5.
Development of LSTM-RNN model. The development of an LSTM model involves several steps. In general, there are five main stages (as shown in Fig. 5): 1) Data collection, 2) Preprocessing of data, 3) Network creation, 4) Training, and 5) Model performance evaluation.

The first step in model development involves collecting and preparing the dataset. Solar irradiance and SPP generation are the main collected data, which serve as input parameters for training and forecasting for the day ahead. The data from the dataset were split into two subsets in a 90:10 ratio, where 90% of the data were used for training the LSTM model, and the forecasted data from the trained model were validated against the actual 10% of data that were held out for assessing the prediction performance.

The second step involves data preprocessing, where three data preprocessing procedures are conducted for more effective model training. These procedures include: 1) handling missing data, 2) data normalization, and 3) random shuffling of data. Missing data are replaced by the mean value of neighboring values within the same measurement period. The normalization procedure before inputting the data is generally a good practice as mixing variables with large and small amplitudes can confuse the learning algorithm regarding the importance of each variable and may lead it to discard variables with smaller amplitudes [49].

Before applying data to the ML algorithm, it is necessary to preprocess the data to correct or remove outliers and fill in missing values.

Other steps in data preparation for ML include feature scaling and encoding. Machine learning algorithms often perform poorly when input features have vastly different scales, so we transformed the data to have a zero mean and unit variance.

During the creation of the LSTM-RNN model, the following parameters are specified:
- Number of layers;
- Hyperparameters (training parameters);
- Performance evaluation metrics.

We utilized the Matlab Deep Learning Toolbox [50] to develop the forecasting framework employed in this study. The framework, shown in Fig. 6, consists of an input layer, LSTM hidden layers, a fully connected layer, and an output layer.

The input layer introduces the time series data into the network. The hidden layers learn dependencies between time steps in the temporal data. To perform forecasting, the network concludes with a fully connected layer and an output layer.

The first LSTM block takes the initial state and the first time step of the training example \( x_t \) and computes the first output \( h_1 \) and updated cell state \( C_1 \). At time step \( t \) the block takes the current network state \( (C_{t-1}, h_{t-1}) \) and the next time step of the training example \( x_t \) and computes the output \( h_t \) and updated cell state \( C_t \). The final output data is the solar irradiance forecast \( g_1 \ldots g_T \).

During the training process, the weights are adjusted to make the actual outputs of the network closely match the target (measured) outputs. For each combination of input variables, various network architectures are explored to determine the optimal LSTM architecture (i.e., the lowest mean squared deviation) for each combination of input variables. Subsequently, different learning algorithms are employed, involving variations in the number of hidden layers and activation functions of the hidden/output layers.

Hyperparameter Optimization. Achieving high efficiency with LSTM-RNN networks involves optimizing numerous hyperparameters. These parameters affect both performance and the time/memory costs of algorithm execution. The selection of hyperparameters often makes the difference between average and state-of-the-art performance in ML algorithms [51]. While there may be some general recommendations for suitable values of these hyperparameters [52], optimization remains necessary since optimal values depend on the data type being used, the specific datasets being compared, evaluation criteria, and other factors. The hyperparameters we optimized are listed in Table 1. Learning rate is arguably the most crucial hyperparameter [53]. The number of hidden layers in the network determines its depth. We experimented with optimizers such as adam [54], rmsprop [55], and sgd [56], finding that adam performed better than the others. We employed a standard scaler and employed full gradient descent.

Table 1. Hyperparameter optimization for LSTM model

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Value Search Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning rate</td>
<td>0,0005</td>
</tr>
<tr>
<td>Optimization solver</td>
<td>adam</td>
</tr>
<tr>
<td>LearnRateSchedule</td>
<td>piecewise</td>
</tr>
<tr>
<td>MiniBatchSize</td>
<td>100</td>
</tr>
<tr>
<td>L2Regularization</td>
<td>0,0005</td>
</tr>
<tr>
<td>Feature scaling</td>
<td>Standard</td>
</tr>
<tr>
<td>Number of layers</td>
<td>3</td>
</tr>
<tr>
<td>Hidden units/layer</td>
<td>10</td>
</tr>
<tr>
<td>Number of epochs</td>
<td>1250</td>
</tr>
</tbody>
</table>

Forecasting Results and performance metrics. The final step involves evaluating the effectiveness of the developed LSTM-RNN model. To quantitatively assess the model’s performance and identify any trends in its effectiveness, a
statistical analysis is conducted. This analysis includes metrics such as Root Mean Square Error (RMSE) (7) and Mean Absolute Error (MAE) (8).

Equations of indicators are formulated as follows:

\[ RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2} \]  
\[ MAE = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i| \]  

where \( N \) is the sample size; \( y_i \) is the actual value; \( \hat{y}_i \) is the predicted value.

These metrics can be used to characterize the deviation of forecasted values around measured data, stemming from spatial aggregation. Lower values of these indicators indicate higher forecast quality. For predicting the day ahead, the values of these errors are crucial as they indicate in absolute terms how much real data can deviate from the forecasted values [57].

After training and testing the data, the forecasted values yielded an RMSE of 0.835 W and MAE of 0.353 W, confirming that the forecast testing provided an effective outcome (Table 2).

**Table 2. Statistical errors of the best performing LSTM model**

<table>
<thead>
<tr>
<th>Epoch</th>
<th>RMSE, [kW]</th>
<th>MAE, [kW]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1250</td>
<td>0.83486</td>
<td>0.35298</td>
</tr>
</tbody>
</table>

The prediction of SPP generation for the day ahead was carried out using the trained model with maximum achievable efficiency. The forecasted data is presented in Fig. 7.

![Fig. 7. Measured and predicted SPP generation](image)

**Building the MATLAB-based GUI.** The tool for forecasting SPP generation was developed using the MATLAB Graphical User Interface Development Environment (GUIDE). GUIDE automatically generates a code file containing MATLAB functions that control the GUI’s operation. The code helps initialize the GUI and contains a structure for event handlers of the GUI components and executable elements when interacting with the user. The MATLAB editor can be used to add code to event handlers to perform necessary actions [58]. The GUI was built based on the best-performing developed LSTM-RNN model’s efficiency.

The main window is designed in such a way that the user can perform forecasting (“Forecast”), obtain more information about the tool (“About”), learn about the description of the studied SPP, and exit the tool (“Exit”), as shown in Fig. 8 and 9.

![Fig. 8. “Main” window](image)

![Fig. 9. “Solar Power Plant” window](image)

The “Forecast” window is divided into three sections: input data, output data, and a set of buttons (Fig. 10).

![Fig. 10. “Forecast” window](image)
The “Input Data” section consists of a text input field, a dropdown menu, and four buttons. The user can choose a file with input data by clicking the “Load Data” button. This will open a window allowing the user to select a file. Then the user can select a model from the dropdown menu. By clicking the “Model Param” button, the user can input the hyperparameters needed for training the LSTM model, and the “Metrics” button will open a window with calculated evaluation metrics for each training and testing iteration. The user can click the “Forecast” button to predict the output solar power generation, as shown in Fig. 11 and 12, respectively.

The prediction results of the SPP generation will be displayed in a graphical format, as shown in Fig. 13.

At the bottom of the “Forecast” window, there are three buttons: “Clear”, “Main”, and “Exit”. The functionality (transition to a specific window) changes according to the current window. The “Clear” button clears the input data, graph, and output data table. The “Main” button opens the main window and closes the current window. Finally, the “Exit” button closes and terminates the tool. At the bottom of the window, there are three buttons: “Clear”, “Main” and “Exit”.

Conclusions. This study developed an LSTM-RNN model for forecasting solar power generation in the village of Velyka Dymerka, Kyiv region, Ukraine. In this research, a deep recurrent neural network called LSTM was utilized, using only exogenous features to address the task at hand. The model uses the current solar power plant generation as an input feature. This approach eliminates the need for historical solar irradiance data, which is costly to measure. In comparison to other analogous models available in the literature, the LSTM-RNN model presented in this study showcases competitive performance in solar power generation forecasting. While the specific models in the literature may vary in architecture and approach, our LSTM-based model stands out for its ability to forecast solar energy generation without relying on historical solar irradiance data. Several studies have explored the use of traditional regression models, such as linear regression, for solar energy forecasting. However, these models often struggle to capture the nonlinear relationships between environmental factors and solar power generation. In contrast, our LSTM-RNN model excels in handling such nonlinearities, providing superior forecasting accuracy. Additionally, some research has explored the application of decision tree-based models, such as Random Forests or Gradient Boosting, for solar energy prediction. While these models can handle complex interactions, they may require a substantial amount of historical data and often fall short when faced with time-series forecasting tasks. The LSTM-
RNN model, as demonstrated in our study, effectively addresses this challenge. The achieved RMSE of 0.835 W and MAE of 0.353 W for most effective model not only compare favourably with similar deep learning approaches but also demonstrate competitive performance when measured against conventional regression-based models. This highlights the potential of LSTM-RNNs as a reliable choice for solar energy forecasting. Furthermore, our development of a user-friendly GUI tool in MATLAB adds practical value to the model's applicability. While some research may focus solely on model development, the inclusion of a user interface facilitates its adoption and utilization by solar energy professionals in Ukraine. Regarding legislative compliance, the proposed model aligns with the 5% deviation allowance for hourly power forecasts specified in Section XVII of the Ukrainian Law on the Electricity Market. This ensures that our forecasting results meet the regulatory standards, further emphasizing the practical suitability of our approach in real-world energy planning and management. In summary, the LSTM-RNN model presented in this study stands out as a robust and accurate tool for solar power generation forecasting, particularly when compared to traditional regression models and decision tree-based approaches. Its competitive performance, user-friendly interface, and compliance with regulatory requirements position it as a valuable asset in the field of solar energy in Ukraine, with potential applications in other regions as well. Further research and validation across diverse geographic locations and climatic conditions can enhance its versatility and reliability.

REFERENCES


58. MATLAB Creating Graphical User Interfaces, revision of March 2015 for MATLAB 8.5 (Release 2015a), The MathWorks Inc., Natick, MA, USA.