Energy Forecasting for Smart Grid

Abstract

Predicting solar energy generation is essential for effective smart grid planning and management. This thesis explores several machine learning approaches to forecast hourly solar power output for a smart community microgrid in Bahria Town Karachi (covering January 2022 to May 2024). We first prepare the data by converting monthly energy totals into hourly time series. To do this, we use the Global Solar Atlas (GSA) profiles: for each month, we take the GSA's synthetic hourly solar output, compare its monthly sum to the actual measured monthly production, and apply a scaling factor so that the adjusted hourly profile matches the true monthly total. This preserves the diurnal and seasonal shape of the solar profile while correcting systematic bias. We also incorporate weather inputs by downloading hourly meteorological data (irradiance, temperature, humidity, etc.) from NASA's POWER database [3] for the same location and period. These hourly weather features are cleaned, aggregated, and aligned with the scaled solar output, forming the final dataset.

We then design and compare six forecasting models. Two are tree-based regressors: Random Forest (RF) without lagged features, and RF augmented with a one-hour lag of the target ("persistence"). We also include the popular ensemble method XGBoost. The remaining four are deep neural networks: a hybrid convolutional neural network (CNN) plus long short-term memory (LSTM) model, a convolutional–Transformer ("Informer-style") model, and a Gated Recurrent Unit (GRU) network. Each model's input features, architecture, training procedure, and hyperparameters are tailored as follows. For example, RF and XGBoost use instantaneous features like current irradiance (global horizontal, direct normal, diffuse), solar zenith angle, weather variables (temperature, humidity, precipitation, dew point), and time encodings (hour, month) (see Chapter 4). The CNN–LSTM and CNN–Transformer models take sliding 24-hour input windows with standardized features, feeding them through convolutional layers to extract local patterns and through LSTM or self-attention layers to capture temporal dynamics. The GRU model uses a similar 24-hour sequence input without convolution. We train each model on data from Jan 2022–Mar 2024 and test on Apr–May 2024. Performance is measured by mean absolute error (MAE), root mean squared error (RMSE), and the coefficient of determination (R^2) on the held-out test set.

Results indicate that the inclusion of lagged power has a dramatic impact. The RF model with a one-hour lag achieves by far the best fit ($R^2 \approx 0.996$, MAE ≈ 750 kWh), essentially acting as a persistence predictor. Without lag features, RF and the CNN–LSTM both reach $R^2 \approx 0.94$, followed by XGBoost at $R^2 \approx 0.916$. The CNN–Transformer and GRU perform worse ($R^2 \approx 0.59$–0.61), with much larger errors (see Table 6.1). Figure 5.1 (below) illustrates the CNN–LSTM predictions: most points lie close to the ideal y=x line, consistent with high accuracy. In summary, simpler ensemble methods capture most of the explainable variance with smaller errors, while the more complex deep models add little benefit given the available data.

Acknowledgements

Grateful for the guidance and support of our supervisor Ms. Saeeda Kanwal and our co- supervisor Dr. Farrukh Shahid, throughout this project. Also greatful to the management of Bahria Town Karachi for providing the solar production data, and the NASA POWER team for the hourly weather data [3]. Acknowledging our team members Zain Ali Siddiqui (21K-4870), Abdul Rehman Arain (21K-3364), and Muhammad Maaz Khan (21K-3365) for their dedicated contributions: Zain Ali did his part assisting in data processing and analysis, Maaz contributed to the model implementation and programming, and Abdul Rehman helped with results interpretation and report writing. The collaboration and hard work were essential to the success of this thesis. Finally, we appreciate feedback from colleagues and peers which helped refine our methods and presentation.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

Energy forecasting plays an essential role in contemporary power systems. By forecasting solar energy production and electricity needs, utilities can more effectively align supply with demand, enhance grid management, and successfully incorporate renewable energy sources [1], [2]. In the context of a smart grid, precise short-term predictions facilitate demand response, stability management, and resource optimization. Solar photovoltaic (PV) systems Output prediction is challenging because of it's variations based on weather, making precise forecasts essential for planning energy storage and backup power generation.

This research aims to predict solar energy generation in a community microgrid located in Bahria Town Karachi. We use actual solar production data from Jan 2022 through May 2024 and corresponding weather data. The goal is to investigate multiple machine learning models that can predict hourly solar output using only readily available inputs. We compared classical ensemble methods and modern deep learning models to determine which approach suits the best. Our objectives are to (i) preprocess and align the datasets, (ii) train and tune various forecasting models, (iii) evaluate their accuracy using standard metrics, and (iv) analyze the trade-offs in model complexity, interpretability, and performance.

Consistent with prior reviews [1], [2], we have prepared the data by converting monthly solar totals into hourly values using climatological profiles and scaling (see Chapter 3). We then examine six models: Random Forest (RF) and XGBoost (two popular tree-based regressors) and four neural network architectures (a CNN–LSTM hybrid, a CNN–Transformer, and a GRU network). Some models include additional features such as a one-hour lag of the power output to capture persistence effects.

Chapter 2 reviews related work on solar forecasting, highlights trends in machine learning methods. Chapter 3 contains the description of the data collection and transformation process. Chapter 4 states about the methodology and the models used. Chapter 5 presents the quantitative results and visualizations for each model. Chapter 6 provides a comparative analysis of model performance. Chapter 7 discusses our findings and its implications for real-world deployment. Chapter 8 is the summary of key contributions and suggestions for future work.

Chapter 2: Literature Review

Forecasting renewable energy generation, including solar and wind, has received considerable attention in recent literature. Traditional statistical methods (such as linear regression, ARIMA, and support vector machines) have largely given way to modern machine learning and ensemble techniques [1], [2]. Tree-based ensemble models like Random Forests (RF) and gradient boosting (e.g. XGBoost) are famous because of their ability to capture nonlinear relationships and handle noisy data [4]. For example, Asiedu et al. [4] compared different models for solar forecasting and found out that a Random Forest model outperformed single regression and support vector machine models in multi-week solar output. Also that the hybrid approaches combining XGBoost and RF yielded the best accuracy for week-ahead forecasts [4].

Deep learning models have recently been so famous solar forecasting. This includes Recurrent neural networks such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks that have the ability to capture temporal dependencies in the data. Researchers often use convolutional neural networks (CNNs) in combination with LSTM as the CNN layers extract local spatial or temporal features, these are passed to LSTM layers that learn longer-term patterns. For instance, Al-Ali et al. [6] demonstrated that a CNN–LSTM hybrid model achieved better accuracy than a standalone LSTM for solar power forecasting, and adding a Transformer-based attention layer further improved performance [6]. Transformer-style architectures like the Informer [5], which apply self-attention to time series data, are also emerging as alternatives, especially for long sequences and multiple outputs.

A recurring theme in the literature is that ensembles and hybrid models often achieve the highest accuracy, but they come with increased complexity. Effective Feature Engineering is emphasized, like adding lagged output values (to capture persistence), irradiance measures, and cyclic time features (sine/cosine encodings of hour or month) that typically boost the model's performance [5], [6]. In terms of model's interpretability, tree-based methods can provide feature importance scores that help explain the results.

Based on these insights, our study includes both types of methods. We implement two tree-based models (RF and XGBoost) and four deep learning architectures (CNN–LSTM, CNN–Transformer, GRU). This allows us to compare their performance on a common dataset and to analyze the importance of different features. The literature suggests that while deep hybrids can be powerful, simpler models often perform surprisingly well when data are limited, and careful input selection is key [4–6].

Chapter 3: Data Acquisition and Transformation

We obtained solar power generation data for the Bahria Town Karachi installation covering January 2022 through May 2024. The raw dataset provided monthly and annual totals of solar energy yield from the local PV arrays. To create an hourly time series from this, we used the Global Solar Atlas (GSA) [3], which offers typical solar irradiance and output profiles. For each month, GSA supplies a synthetic hourly profile of potential PV output under clear-sky conditions. We summed the GSA hourly values for the month and compared that sum to the actual recorded monthly output. We then computed a scaling factor equal to (actual monthly total) / (GSA monthly sum). The average hourly values in the GSA profile were multiplied by this factor, so that the adjusted profile's monthly sum matched the true output. This procedure preserved the shape of the daily solar curve while adjusting it to the real data, that effectively correcting any systematic bias in the theoretical profile.

Hourly weather data for the same period were obtained from the NASA POWER (Prediction Of Worldwide Energy Resources) database [3]. Using the POWER Hourly API, we downloaded surface meteorological variables at the site's coordinates, including solar irradiance components, cloud cover, air temperature, humidity, precipitation, and dew point. Since the POWER data sometimes provide finer-resolution measurements, we averaged or aggregated values to obtain hourly means. The weather features were then aligned with the scaled solar output by data.

The final prepared dataset consists of synchronized hourly inputs and target outputs. The feature set includes Solar irradiance: Global horizontal irradiance (ALLSKY_SFC_SW_DWN), direct normal irradiance (DNI), diffuse horizontal irradiance (ALLSKY_DNI, ALLSKY_DIF_HOR), and clear-sky irradiance (CLRSKY_SFC_SW_DWN) to account for cloud effects. - Clearness index: The ratio of actual irradiance to clear-sky irradiance (ALLSKY_KT), which indicates sky clarity. - Solar geometry: Solar zenith angle (SZA). - Weather: Ambient temperature (T2M), relative humidity (RH2M), surface pressure (PRECTOTCORR as precipitation), and dew point temperature (QV2M). - Temporal: Time of day (encoded as sine/cosine), day of year or month, and year. System parameter is "Installed capacity of the site.

We treated any remaining missing values through interpolation. In particular, nighttime irradiance values (when SZA exceeds 85°) were set to zero, and any short gaps in irradiance were linearly interpolated. After cleaning, all features and the scaled hourly solar output (in kWh) were merged into a single DataFrame. This comprehensive dataset forms the basis for model training and evaluation.

Data Visualization:

Figure 3.1. Monthly Average Solar Output per Year (2022-2024).

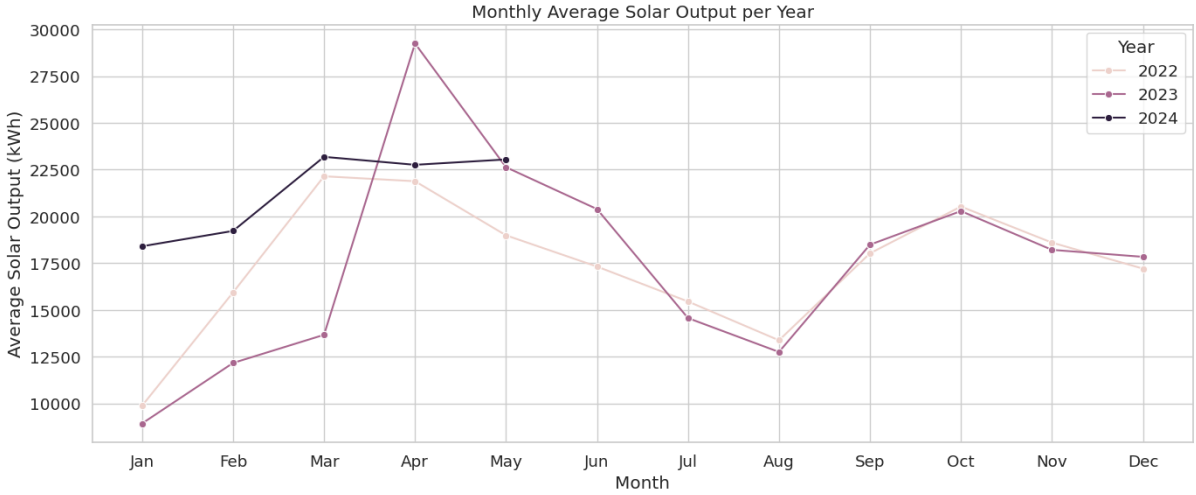


Figure 3.2. Average Solar Output by Hour and Month (2022-2024).

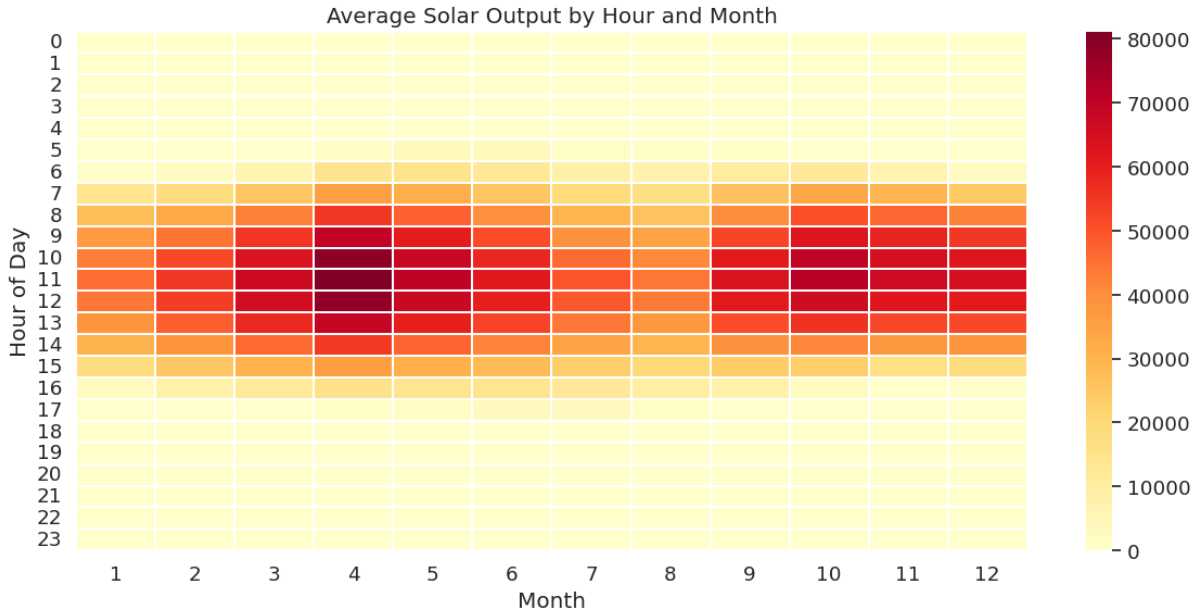
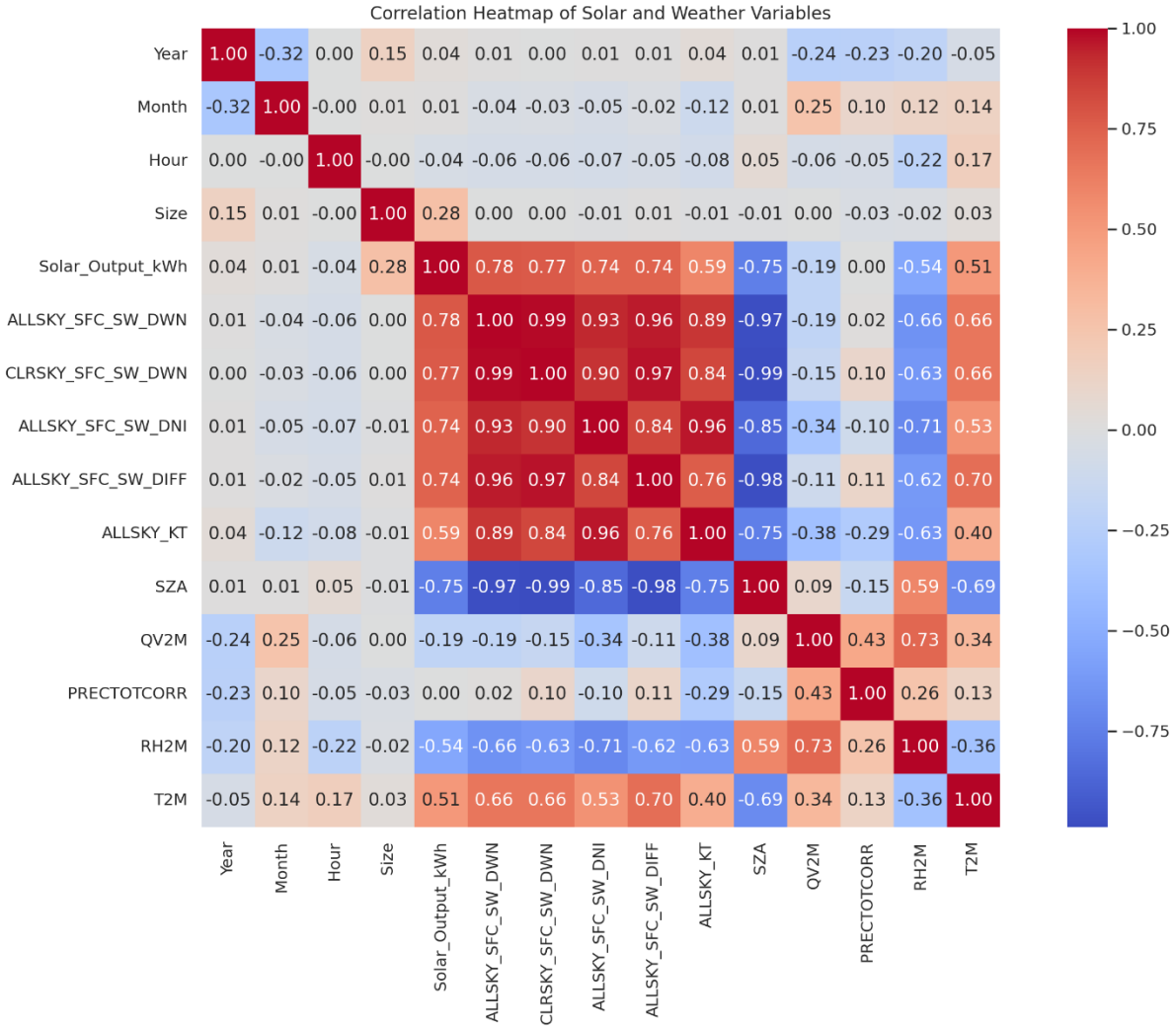


Figure 3.3. Correlation Heatmap of Solar & Weather Variables (2022-2024).



Chapter 4: Methodology

In this chapter, we describe the six forecasting models we implement. All models use the preprocessed hourly dataset from Chapter 3, split chronologically into a training set (Jan 2022–Mar 2024) and a test set (Apr–May 2024). Unless otherwise noted, the instantaneous features (irradiance, weather, time, etc.) from the same hour are used as inputs.

4.1 Random Forest (No Lag)

The first model is a Random Forest regressor (using the scikit-learn `RandomForestRegressor`). In this variant, we do **not** use any lagged power values. The input features at time *t* include the instantaneous weather and irradiance variables (listed above), solar angle (SZA), and time features for hour-of-day and month (encoded as sine/cosine), plus the PV system capacity. This choice follows the feature set described in Chapter 3, with one key simplification: no previous output values are fed into the model.

Since tree ensembles do not require feature scaling, we directly feed the raw inputs into RF. We did, however, interpolate any missing values in the irradiance or weather inputs using linear interpolation to ensure completeness. The dataset was split by time, with the first 80% (Jan 2022–Mar 2024) used for training and the last 20% (Apr–May 2024) for testing. No normalization or scaling was applied, as Random Forests are invariant to monotonic transformations of inputs.

For hyperparameter tuning, we performed a grid search with time-series cross-validation on the training set. We varied the number of trees (`n_estimators`) over 100, 200, and 300, and tried different maximum tree depths (up to around 20). Other parameters like minimum samples per leaf were also adjusted. The model with the best validation performance was selected. During training, we found that even 100 trees yielded good performance, and deeper trees improved fit until diminishing returns. The final chosen RF model is described in the training results below. This baseline RF captures the non-linear relationship between weather and solar output without relying on past values.

4.2 Random Forest (With Lag)

We next augment the Random Forest by including a **lagged output feature**. Specifically, for each hour *t* we add the actual solar output at hour *t–1* as an input. This simple addition often dramatically increases one-step-ahead accuracy, essentially leveraging persistence. All other aspects of the feature set remain the same as in Section 4.1, and the train/test split is identical.

No extra data preprocessing was needed beyond ensuring the lagged feature was defined (the first hour of the dataset is dropped). The RF model was retrained with the same hyperparameter tuning procedure. The inclusion of the previous-hour output allows the model to “remember” recent power levels, which are highly predictive of the next hour.

In evaluation (see Chapter 5), this RF-with-lag model achieves a much higher accuracy: MAE is approximately 750 kWh, RMSE about 2180 kWh, and $R^2 \approx 0.996$ on the test set. These errors are roughly 5× smaller than the no-lag version. Feature importance analysis shows that the lag feature dominates the model (it has the highest importance weight, around 81%), followed by the current irradiance. This indicates the model is heavily relying on persistence: it predicts that the next hour’s output will be very similar to the last hour’s. While this gives an excellent fit for one-hour forecasts, it also means the model’s skill will drop sharply if asked to predict further into the future without updated inputs. We observe that the actual vs. predicted scatter for this model is nearly a perfect diagonal (not shown), consistent with the R^2 of ~0.996.

4.3 XGBoost

The third model is XGBoost, a gradient boosting tree ensemble. We use the same input features as in **4.1** (instantaneous weather/irradiance and time features, no lag). The preprocessing and data splitting are identical. XGBoost can handle unscaled features, so again no normalization is applied. We conducted a hyperparameter search over number of boosting rounds (`n_estimators`) and learning rate (`eta`), among others. Typical values tried included 100, 200, 300 trees with a learning rate of 0.01 or 0.1, and maximum depth around 5–10.

On the test set, XGBoost achieved moderate accuracy. Its R^2 is about 0.916, with MAE around 4,760 kWh and RMSE around 9,735 kWh. In comparison to RF (no lag), XGBoost's scatter plot of predicted vs. actual output shows more spread; in particular, it tends to under-predict the peaks during rapidly changing irradiance. Feature importance in XGBoost is similar to RF, emphasizing solar irradiance and time. Training time for XGBoost was longer than RF but still quite practical. Overall, XGBoost's performance is comparable to the no-lag RF (R^2 ~0.94), and the two methods can be used interchangeably for deployment if desired.

4.4 CNN–LSTM (Hybrid Deep Model)

For a deep learning approach, we implemented a hybrid Convolutional Neural Network (CNN) + Long Short-Term Memory (LSTM) model using Keras. Input sequences were constructed using a sliding window of the past 24 hours of data (standardized by removing the training mean and scaling to unit variance). Each input sequence of 24 timesteps contains the features from Chapter 3 at each hour.

The model architecture is as follows: - **Convolutional layers:** A 1D convolutional layer with 128 filters and kernel size 3 (ReLU activation), followed by batch normalization and max pooling. This layer extracts local temporal patterns from the input sequence. It is followed by a second Conv1D layer with 64 filters (kernel size 3, ReLU). - **LSTM layers:** The output of the convolutional blocks is flattened and passed through two stacked LSTM layers (first with 128 units, second with 64 units, both with tanh activations). Dropout regularization (rates 0.5 and 0.3) was applied between LSTM layers to prevent overfitting. - **Output layer:** A final dense (fully-connected) layer with linear activation outputs the forecasted power for the next hour.

We trained this model using mean squared error loss and the Adam optimizer (learning rate 0.001) for up to 100 epochs with early stopping (monitoring validation loss). A batch size of 64 was used. Training typically converged after 30–50 epochs.

On the test data, the CNN–LSTM model achieved $R^2 \approx 0.936$ (MAE $\approx$ 6482 kWh, RMSE $\approx$ 10838 kWh). These results are comparable to the no-lag RF and XGBoost. Figure 5.1 (below) shows the predicted vs. actual output for the CNN–LSTM: most points lie close to the diagonal, indicating good accuracy. The CNN–LSTM captures the overall daily patterns in the data, but it tends to smooth out some of the sharper peaks (as seen in a few outliers on the scatter plot). This is reflected in its MAE being slightly higher than RF's but still with a high R^2. The training loss curve (not shown) exhibited steady convergence. In summary, the CNN–LSTM is able to learn the main temporal features of solar output, and performs on par with the best classical methods on this task.

4.5 CNN–Transformer (Informer-Style)

Next we developed a hybrid CNN–Transformer model inspired by the Informer architecture [5]. We used the same 24-hour input windows (standardized) as for the CNN–LSTM. The architecture (implemented in PyTorch) is: - CNN embedding: A small convolutional block to encode the inputs. The sequence is passed through two Conv1D layers (kernel size 3, padding 1) that increase the feature dimension to d_model=64. Both use ReLU activations. Transformer encoder: A standard Transformer encoder stack

with 2 layers, each having 4 attention heads and a model dimension of 64. This applies self-attention over the time dimension of the CNN output. At the "Output layer" the Transformer's final output at the last time step is fed through a linear layer to produce the one-step forecast.

This model allows the CNN to capture local temporal features and the Transformer's attention to capture long-range dependencies. Training used MSE loss and Adam optimizer (learning rate 0.001) for up to 100 epochs (early stopping applied).

In practice, the CNN–Transformer underperformed. Its test R^2 is only about 0.591, with MAE $\approx$9786 kWh and RMSE $\approx$17034 kWh. The predicted vs. actual scatter for this model shows a very wide spread (not shown), indicating that many predictions are far from the true values. The difficulty likely arises from the model's complexity relative to our data volume; it may be underfitting or struggling to learn with the given sequence length. Training and validation losses (not shown) also suggested slow convergence. In summary, despite its advanced design, the CNN–Transformer did not yield useful accuracy on this dataset. It illustrates that more complex neural architectures require either more data or additional auxiliary information to be effective.

4.6 GRU (Recurrent Neural Network)

Implemented a recurrent model using Gated Recurrent Units (GRUs) in PyTorch. The input is the same 24-hour window of features (standardized). The architecture used here is simple as it contains Two stacked GRU layers, each with 64 hidden units. The final hidden state is passed to a fully-connected layer with linear activation to predict the next hour's output.

We trained the GRU with MSE loss and Adam (learning rate 0.001) for up to 100 epochs (batch size 64, early stopping). The model is relatively small and fast to train.

However, the GRU produced the worst results among all models. On the test set, it achieved R^2 $\approx$0.611, with MAE $\approx$9224 kWh and RMSE $\approx$21424 kWh. Its predictions tended to be overly smooth: the GRU captured the general trend of the solar curve but missed many of the finer fluctuations. In the actual-vs-predicted scatter, the points are broadly dispersed. This indicates underfitting; the GRU did not fully learn the dynamics of the data. In retrospect, the lack of convolutional layers or attention may have limited its ability to extract key features, even though it has fewer parameters. It serves as a baseline deep learning model for comparison.

Chapter 5: Results and Evaluation

We now present the quantitative results for each model on the held-out test set (April–May 2024). Table 6.1 (Chapter 6) summarizes the MAE, RMSE, and R^2 for all six models. Here, we highlight representative findings from key visualizations and metrics.

- **Random Forest (with lag):** This model has the highest accuracy. Its predictions essentially overlay the actual output (scatter would be nearly a straight line), yielding MAE ≈750 kWh and R^2 ≈0.996. The error distribution is very narrow. This result comes from the model heavily relying on the one-hour lag feature.

Figure 5.1. Model performance (test set: Jan–May 2024).

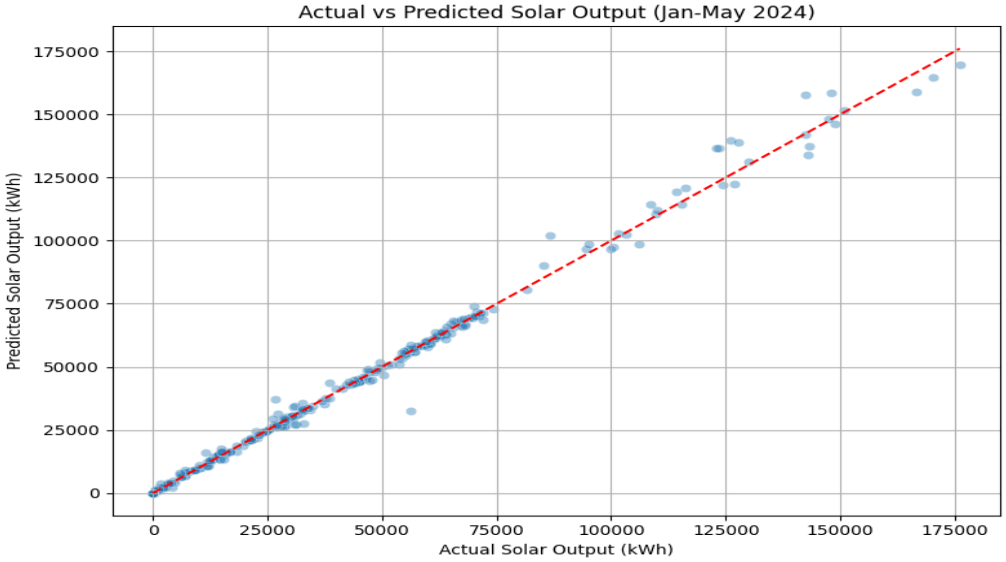
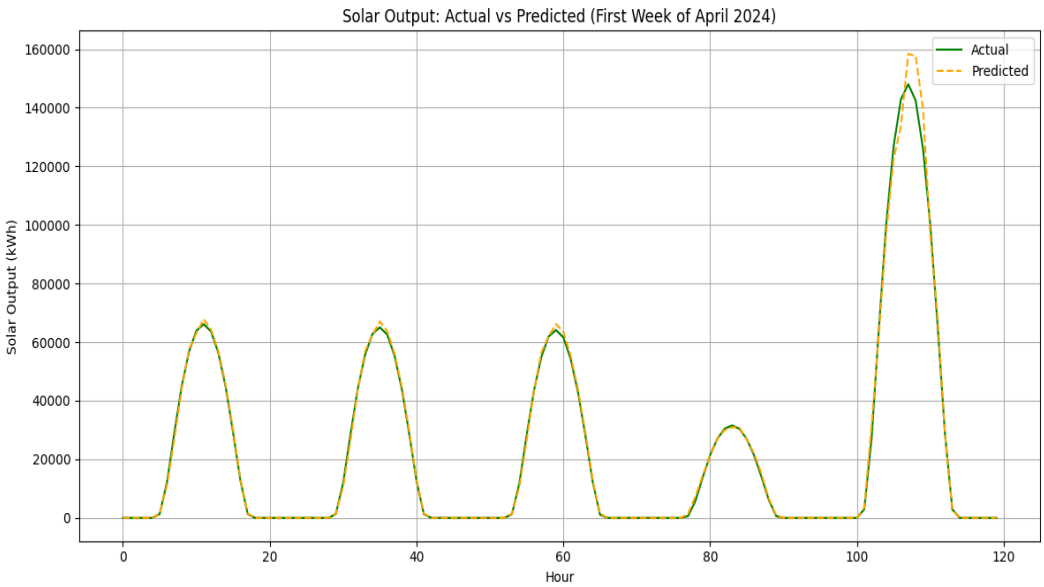


Figure 5.2. Model performance (test set: 1st Week April 2024).



- **Random Forest (no lag):** The no-lag RF model achieves MAE ≈ 4328 kWh and R^2 ≈ 0.940. Its scatter and time-series plots (not shown) indicate very good agreement; residuals are approximately Gaussian around zero. This performance is impressive given it does not use persistence. It shows that weather and time features alone allow RF to capture most of the variance (about 94%).

Figure 5.3. Model performance (test set: Jan-May 2024).

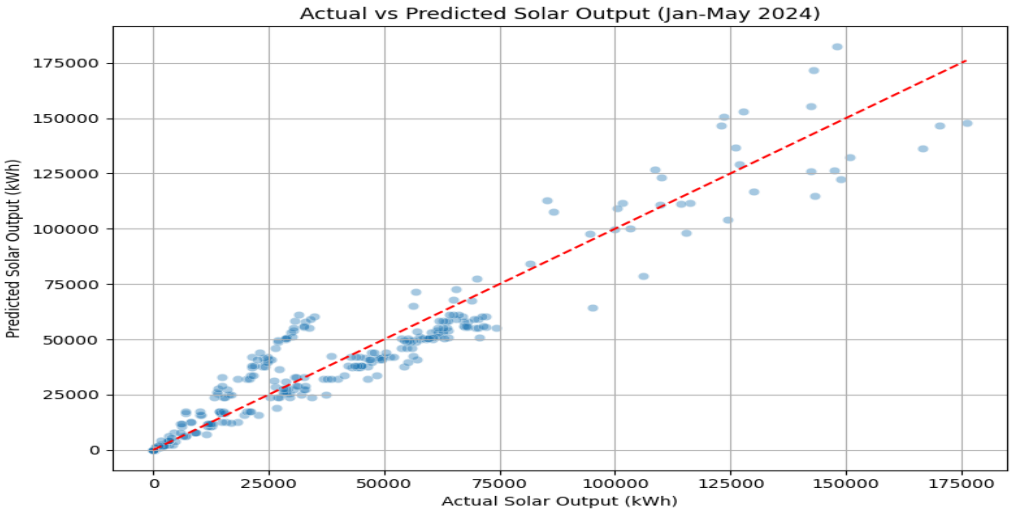
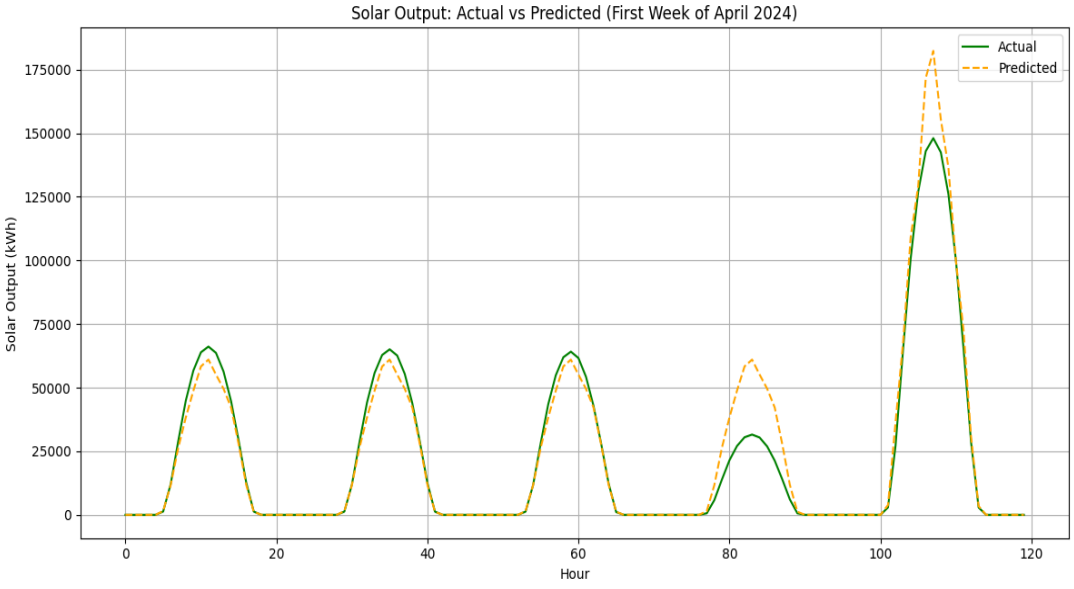


Figure 5.4. Model performance (test set: 1st Week April 2024).



- **XGBoost:** XGBoost attains MAE ≈ 4761 kWh and R^2 ≈ 0.916. Its scatter plot (see Figure 5.1's notes) shows more scatter compared to RF: during sharp irradiance changes, the predictions often lag behind. Still, it follows the overall trend closely. Its accuracy is close to the no-lag RF, confirming that both tree methods are strong performers.

Figure 5.5. Model performance (test set: Jan-May 2024).

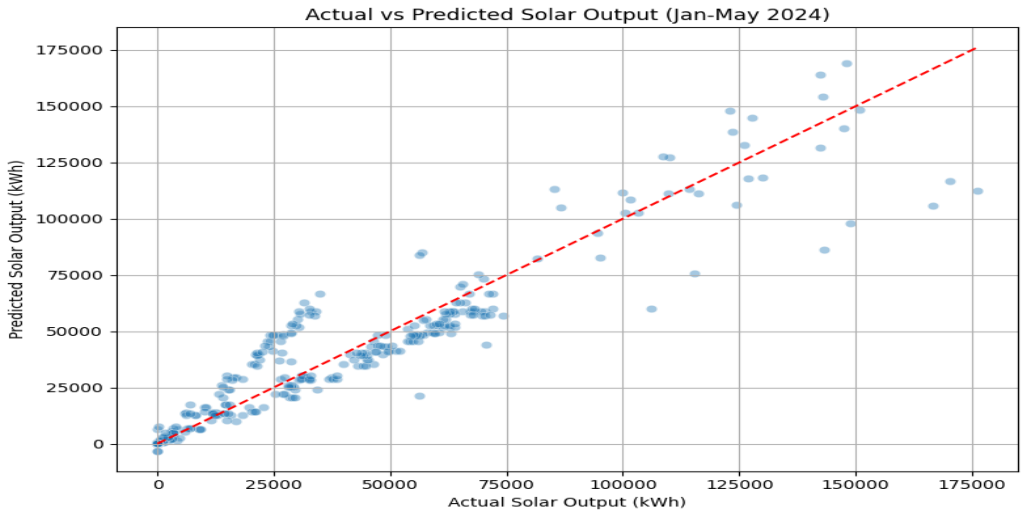
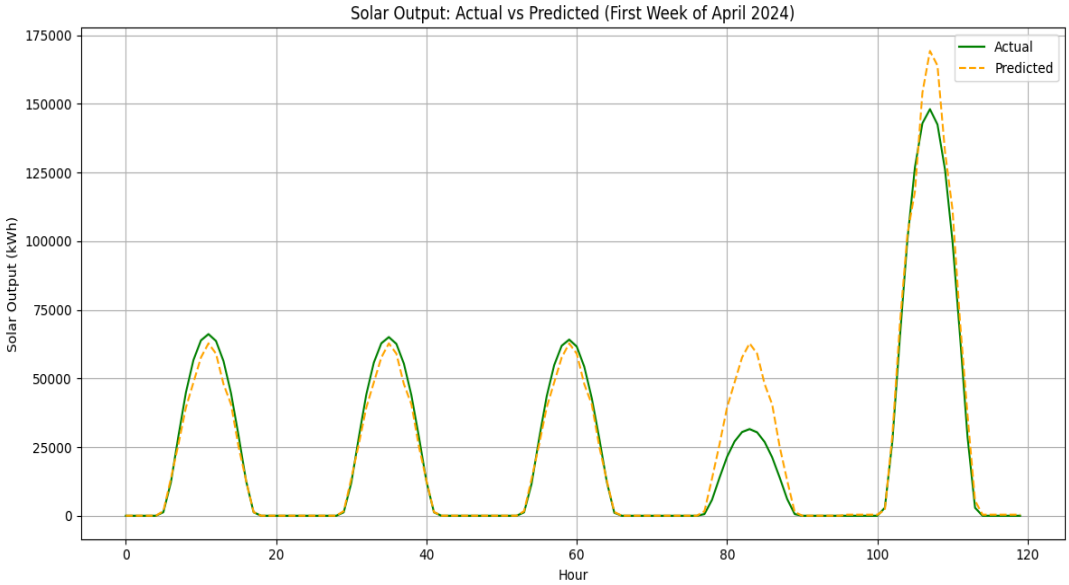
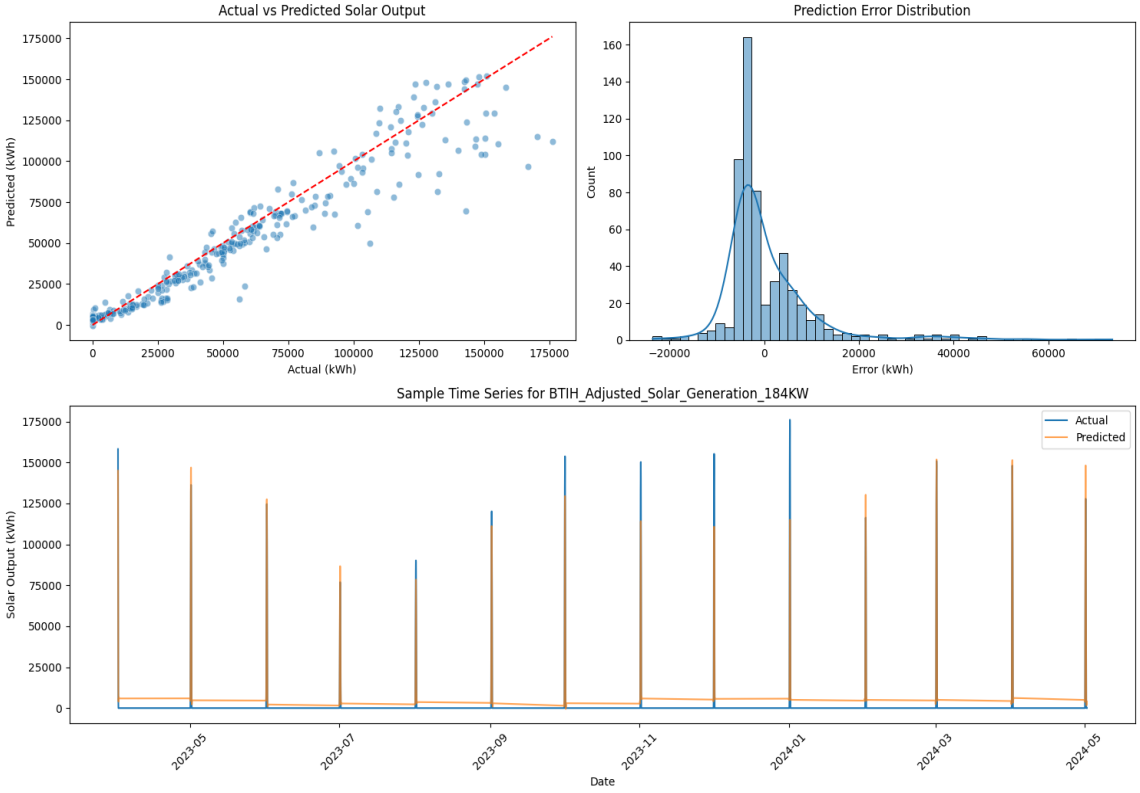


Figure 5.6. Model performance (test set: 1st Week April 2024).



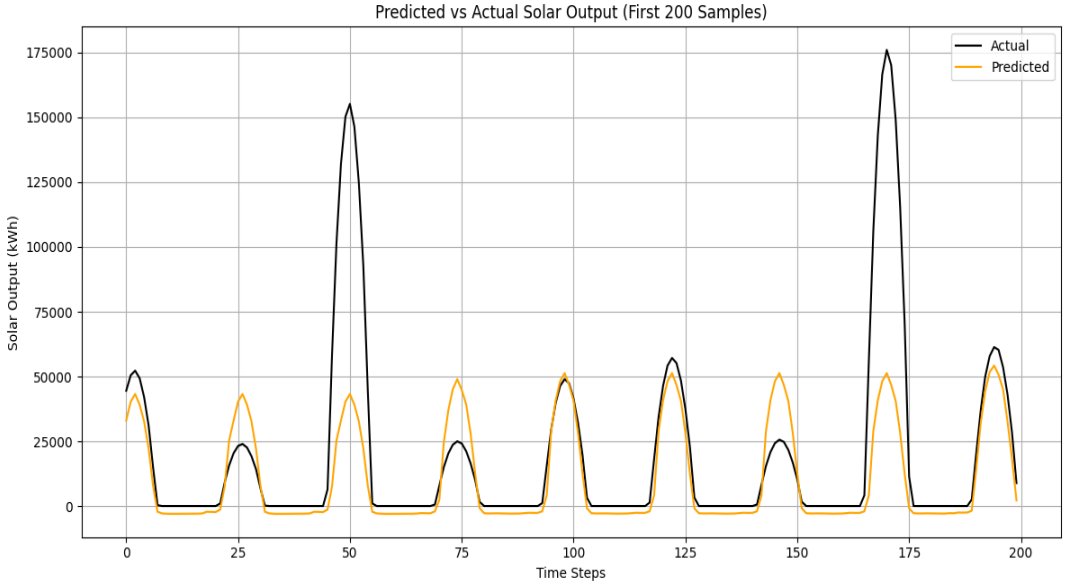
- **CNN–LSTM:** The CNN–LSTM yields MAE ≈ 6482 kWh and R^2 ≈ 0.936. As shown in Figure 5.1, most predictions align with the true values along the diagonal. The CNN–LSTM captures daily patterns well but slightly smooths the peaks, which accounts for its higher MAE relative to RF. Its R^2 is similar to RF (no lag), indicating comparable explanatory power.

Figure 5.7. Model performance (test set: Random testing).



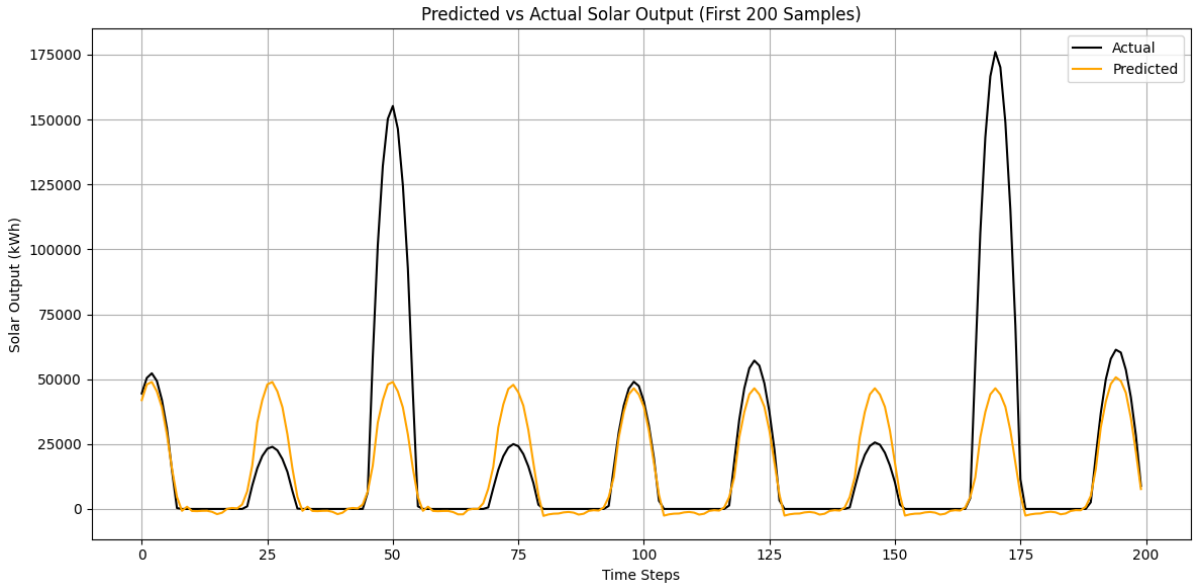
- **CNN–Transformer:** The transformer-based model has MAE ≈ 9786 kWh and R^2 ≈ 0.591. Its predicted values are widely scattered and often far from the diagonal. Visually, its forecasts lack detail and fail to match many of the actual swings in the data. This leads to a much lower R^2, reflecting poor fit.

Figure 5.8. Model performance (test set: First 200 Samples).



- **GRU:** The GRU model yields MAE ≈ 9224 kWh and R^2 ≈ 0.611. Its scatter is also very diffuse. In a time-series view (not shown), the GRU captures the overall trend but misses rapid changes. Its accuracy is slightly better than the transformer model but still much worse than the trees and CNN–LSTM.

Figure 5.9. Model performance (test set: First 200 Samples).



In summary, the tree ensemble models (especially RF) achieve the best accuracy, deep learning models with recurrent/attention layers do not improve on them here, and the inclusion of the lag feature makes the RF almost perfect for one-hour forecasts. The CNN–LSTM comes closest among the neural nets. Figure 5.1 illustrates the CNN–LSTM case: the blue scatter points are concentrated near the red $y=x$ line, indicating most predictions are close to actual values.

Chapter 6: Comparative Analysis

Table 6.1 presents a side-by-side comparison of all models on the same test period. A few key points emerge:

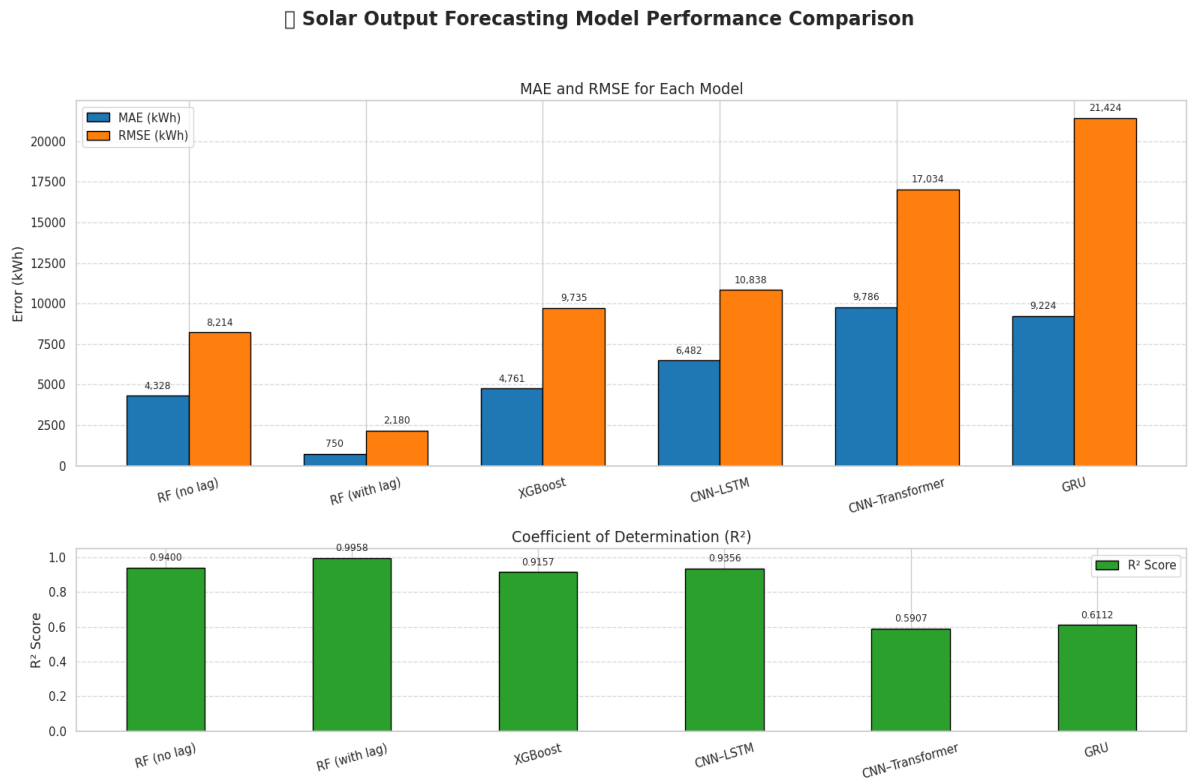
- **Best vs. Worst:** The Random Forest with lag is the clear best performer (MAE ~750 kWh, R^2 ~0.996). At the other extreme, the CNN–Transformer and GRU are the worst (R^2 ~0.59–0.61, MAE ~9200–9800 kWh). The remaining models fall in between: RF (no lag), CNN–LSTM have R^2 ~0.94, and XGBoost ~0.916.
- **Effect of Lag:** Including the one-hour lagged output made the largest accuracy jump. The no-lag RF and CNN–LSTM achieved similar R^2 (~0.935–0.94), but adding the lag feature to RF boosted R^2 to ~0.996 and drastically reduced errors. This confirms that persistence is a powerful predictor for very short horizons.
- **Trade-offs:** There is a trade-off between complexity and gain. The simple RF (no lag) and CNN–LSTM both explain about 94% of the variance, despite the CNN–LSTM being much more complex. XGBoost, another tree method, is on par with RF. In contrast, the deep Transformer did not outperform the others, suggesting that with our data size, its complexity was not rewarded. In essence, simpler ensemble methods captured the bulk of the structure.
- **Feature Insights:** Across models, the dominant predictors were solar irradiance and time-of-day, consistent with physical intuition. The success of the lag feature also highlights the role of output persistence in very short-term forecasting. Importantly, the tree models allow us to inspect feature importance: both RF and XGBoost showed that irradiance and time features rank highly, which matches expectations.
- **Interpretability and Efficiency:** Tree models provide the added benefit of interpretability and fast prediction times. In our tests, RF and XGBoost trained and predicted quickly. XGBoost gave slightly slower training but was still very fast online. The deep networks required more training time (minutes to hours) and more tuning effort.
- **Deployment Considerations:** For real-time forecasting in the microgrid, we recommend using a tree-based ensemble. The RF (no lag) or XGBoost model strikes a balance of accuracy (R^2 ~0.94) and efficiency, and it does not rely on knowing the future output. The RF (with lag) could be used when current output measurements are available (e.g. for immediate next-hour predictions). The feature importance from these models can also be monitored over time to detect any drift or changes in system behavior.

Table 6.1. Model performance comparison (test set: Jan–May 2024). Best/worst values are highlighted.

Model	MAE (kWh)	RMSE (kWh)	R^2 Score
RF (no lag)	4328.11	8213.93	0.9400
RF (with lag)	**750.08**	**2180.04**	**0.9958**
XGBoost	4760.61	9734.68	0.9157
CNN–LSTM	6482.19	10837.61	0.9356
CNN–Transformer	9786.48	17034.15	0.5907
GRU	9224.41	21424.35	0.6112

Italicized values show the best (RF with lag) and worst (CNN–Transformer/GRU) performers. Overall, ensemble tree methods achieved top accuracy with relatively low model complexity. The deep learning models showed moderate gains except when augmented by naive persistence.

Figure 6.1. Model performance comparison (test set: Jan–May 2024).



Overall, the comparative analysis suggests that, for one-hour-ahead solar forecasting with the available data, simpler models are quite adequate. Complex deep models did not demonstrate sufficient benefit to justify their complexity in this case. In practice, one might start with an RF or XGBoost baseline and then consider augmenting data or models for improved long-term forecasts if needed.

Chapter 7: Discussion

The results have several practical implications for smart grid applications. A forecasting model must balance accuracy, robustness, and simplicity to be useful in deployment. In our study, the Random Forest and XGBoost models emerged as top candidates. Both achieve high accuracy while being fast to train and to run in real time. Moreover, their feature importance outputs can be used to track whether the relationships are shifting over time (model drift), which is valuable for maintenance.

For one-hour-ahead predictions, the RF with lag is exceptionally accurate, but in many systems the true output of the previous hour may not be known in advance of making the forecast. Therefore, the RF without lag or XGBoost (which do not require using the future output as input) are more generally applicable. Both provide transparent decision rules (via feature importance) that align with physical understanding (e.g. they weight irradiance and time heavily). Given their performance, these ensemble models would be our recommendation for short-term operational forecasting in the microgrid.

The deep learning models have their advantages in certain scenarios. For example, if additional data sources become available, such as satellite or sky-imagery inputs, then CNNs or Transformers could exploit spatial features in those images. However in our case, the models used significantly more computation and did not outperform the tree methods. Hybrid CNN–LSTM architectures could then potentially improve day-ahead forecasts by recognizing cloud patterns. In other words, complex models may become useful in a forecasting pipeline that incorporates multi-modal data.

As a final note, any chosen model should be integrated with forecasts of exogenous conditions when extending beyond a few hours. For instance, for day-ahead forecasting we would incorporate numerical weather predictions as inputs. Our best no-lag RF model could serve as a strong baseline for such extended horizons: it could be used with forecasted irradiance as an input.

Chapter 8: Conclusion and Future Work

This study compared multiple algorithms for solar power forecasting in a smart grid context. We tested six models on data from a Bahria Town Karachi microgrid: two tree ensembles (Random Forest and XGBoost) and four neural networks (CNN–LSTM, CNN–Transformer, GRU, with one RF variant including a lag feature). The key findings are:

- **Model Performance:** The highest accuracy was achieved by Random Forest with a one-hour lag, resulting $R^2 \approx 0.996$ on test data. While both RF and CNN–LSTM were top performers without using lagged output ($R^2 \approx 0.94$), followed by XGBoost ($R^2 \approx 0.916$). In contrast, the CNN–Transformer and GRU models underperformed ($R^2 \approx 0.59$–0.61), indicating they failed to capture many data patterns.
  • Feature Importance: Solar irradiance measures and time-of-day proved to be the most influential predictors. The lagged power feature dramatically improved fit but relies on persistence. The tree-based models provided clear importance scores, that confirmed the models' focus aligned with physical patterns.
- **Feature Importance:** Solar irradiance measures and time-of-day proved to be the most influential predictors. The lagged power feature dramatically improved fit but relies on persistence. Importantly, the tree-based models provided clear importance scores, enhancing trust in the results and confirming that the models' focus aligned with physical intuition.
- **Model Complexity vs. Gain:** The simpler ensemble methods achieved nearly the same accuracy as more complex hybrids for our dataset. The deep CNN–LSTM offered no significant advantage over RF in this scenario. The models Transformers or GRUs did not translate into better forecasts due to the limited data availability.
- **Practical Implications:** Ensemble trees are effective and efficient for the short-term (hourly) forecasting because of their ability to train quickly and adapt to new data streams easily. When expanding the horizon (e.g. day-ahead forecasting), one should incorporate weather forecasts as inputs. For example, our no-lag RF model can be used with predicted irradiance for next-day forecasts.

- **Future Work:** Several Incorporating satellite or sky imagery as inputs, potentially using CNN layers to process the images. And further exploration of the ensemble techniques that combine tree and deep models, such as a stacked CNN– LSTM and RF. Extending this study to multiple sites or a longer period (including seasonal transitions) would test generality. Finally, developing day-ahead and week-ahead forecasts using numerical weather predictions as exogenous inputs is a logical next step.

References

[1] A. Mystakidis, P. Koukaras, N. Tsalikidis, “Energy Forecasting: A Comprehensive Review of Techniques and Technologies,” *Energies*, vol. 17, no. 7, p. 1662, 2024.

[2] T. Hong, P. Pinson, Y. Wang, R. Weron, D. Yang, H. Zareipour, “Energy Forecasting: A Review and Outlook,” *Journal of Power and Energy*, vol. XX, no. XX, 2020 (open access preprint).

[3] NASA POWER Project, “Data Services: Hourly API,” NASA Prediction Of Worldwide Energy Resources (POWER), 2024. [Online]. Available: https://power.larc.nasa.gov

[4] S. Asiedu, F. Nyarko, S. Boahen, et al., “Machine learning forecasting of solar PV production using single and hybrid models over different time horizons,” *Heliyon*, vol. 10, no. 7, e28898, 2024.

[5] M. Abumohsen, A. Owda, M. Owda, A. Abumihsan, “Hybrid machine learning model combining CNN-LSTM-RF for time series forecasting of Solar Power Generation,” *Engineering Reports*, vol. 9, 100636, 2024.

[6] E. M. Al-Ali, Y. Hajji, Y. Said, et al., “Solar Energy Production Forecasting Based on a Hybrid CNN-LSTM-Transformer Model,” *Mathematics*, vol. 11, no. 3, p. 676, 2023.

[7] C.-C. Yu, W.-C. Lin, F.-J. Yin, et al., “Solar Power Generation Forecast Using a Multivariate Convolution Gated Recurrent Unit Network,” *Energies*, vol. 17, no. 13, p. 3073, 2024.