electronics

MDPI

Article

Temporal Convolutional Network with a Physics-Informed Decoder for 15-Minute-Ahead Solar Power Forecasting

Muhammad Imran Munawar, Jeff Kilby and Krishnamachar Prasad *

School of Engineering, Mathematics & Computer Science, Auckland University of Technology, Auckland 1010, New Zealand; muhammad.munawar@autuni.ac.nz (M.I.M.); jeffrey.kilby@aut.ac.nz (J.K.)
* Correspondence: krishnamachar.prasad@aut.ac.nz

Abstract

As renewable energy grows, photovoltaic power integration into power grids has increased, and high solar penetration creates challenges for power system operation. Accurate, near-real-time solar power forecasting over short time intervals is therefore important for supporting stable grid operation and efficient energy scheduling. This study presents a novel Enhanced Temporal Convolutional Network (TCN) model to predict solar power output 15 min in advance. The model uses dilated convolutions to capture temporal patterns, while multi-head attention and adaptive feature gating help identify important dependencies and relevant inputs. It also extracts long-term trends and short-term variations using seasonal decomposition. A physics-informed decoder builds the forecast by correcting a persistence-based reference forecast. The model was trained on a dataset comprising 5.7 years of real-world solar power generation records from Arizona. After bias correction, the model achieves a root mean squared error (RMSE) of 281.0 $\pm$ 4.6 Wh, a mean absolute error (MAE) of 108.8 $\pm$ 2.9 Wh, and a coefficient of determination (R^2) of 0.928 $\pm$ 0.002 across three random seeds. Compared with the baseline models, the proposed Enhanced TCN reduces the RMSE by 7.0% to 18.8% over five deep learning models, by 8.3% over persistence and by 4.9% over smart persistence. Without bias correction, the model performs no better than smart persistence. The ablation study shows that the physics-informed decoder gives the main gain.

Keywords: solar power forecasting; temporal convolutional networks; deep learning; short-term prediction; renewable energy

Check for updates

Academic Editors: Teodora Lazar and Dragos Pasculescu

Received: 2 September 2026
Revised: 14 October 2026
Accepted: 15 October 2026
Published: 19 October 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Climate change is a widespread concern. More than two-thirds of all greenhouse gas emissions are carbon dioxide (CO_2) [1]. This is a major driver of the push to shift toward cleaner energy sources. The International Renewable Energy Agency (IRENA) reported a record-breaking expansion of renewable energy in 2024. By the end of 2024, renewable sources constituted 46.4% of global installed capacity. In 2024 alone, the world installed approximately 585 GW of new renewable energy capacity. Photovoltaic power drove this growth, adding about 452 GW of new capacity, the largest single-technology addition ever recorded [2]. As more photovoltaic (PV) power connects to the grid, power system operators face major challenges in maintaining efficiency, security, and stability [3]. The power industry is adopting more renewable energy because of environmental concerns and ambitious goals. This expansion is largely due to solar photovoltaic panels. However,

solar energy's variable, inconsistent output affects grid stability. Accurate forecasting is therefore needed to reduce this uncertainty [4].

Many studies have examined PV power forecasting models [5]. Forecasting over a few minutes to one hour is generally known as very short-term forecasting. It helps grid operators respond to sudden changes caused by cloud movement and panel shading [6]. A study of very short-term PV forecasting with cloud modelling confirms that cloud behaviour is one of the biggest drivers of output uncertainty and must be accounted for in any short-term prediction approach [7]. Research on very short-term wind and solar power forecasting [8] shows that while many methods have been developed, computationally expensive approaches often suit low-cost or resource-limited systems poorly. It also shows that most earlier studies have focused on irradiance projections rather than fully integrating solar power systems into the grid. Methods based on high-resolution camera images or satellite data increase system costs and make it harder to respond quickly to sudden weather changes, reducing short-term forecast accuracy [9]. Recent work has examined 5- to 15 min forecasting using techniques such as Ensemble Batch Prediction Intervals (EnbPI), Jackknife+ after Bootstrap (J+ aB), and Inductive Conformal Prediction (ICP). However, these remain computationally demanding and are not well suited to rapidly changing weather conditions [10,11]. Comparative analyses of deep learning architectures and large-scale solar forecasting approaches have also been explored [12,13]. For battery management and short-term load coverage, a 15- to 30 min forecasting horizon is usually sufficient. Forecasts shorter than 5 min get too complicated, while those longer than 30 min become inaccurate because the weather changes so quickly. So a 15 min horizon best balances accuracy and usefulness for grid management. Distributed generation and microgrids have made control and load scheduling more important. Short-term forecasts help operators better manage battery systems and power operations. Intra-hour PV prediction over short intervals is useful for battery management and grid support [14]. Hybrid deep learning approaches for short-term PV forecasting have also been compared across multiple models to assess which work best in practice [15]. Most forecasting methods rely on statistical models. Auto-Regressive Integrated Moving Average (ARIMA) is one of the most widely used methods. This has also been reported in reviews comparing different machine learning methods for solar PV forecasting [16]. Other commonly used methods include Long Short-Term Memory (LSTM), Random Forest (RF), and Extreme Gradient Boosting (XGBoost). A comprehensive study of these models for solar and wind power forecasting found that LSTM achieves higher forecasting accuracy but requires more computational resources, while XGBoost and RF also provide good results with less computational complexity [17]. A more detailed study has assessed the robustness of XGBoost, RNN, KNN, RF, and LSTM models for solar power forecasting. This study considers different lagged input steps and examines model sensitivity and performance through cross-validation [18]. Researchers have also combined deep learning methods with data reconstruction techniques for ultra-short-term solar power forecasting. These approaches aim to improve forecast accuracy [19]. Most of these methods provide a single deterministic value for each time step. As a result, they do not effectively represent the uncertainty in solar power output. This can limit their practical use. Traditional ML methods also struggle to capture continuous temporal dependencies and the complex fluctuations in intra-hour solar power generation [20,21]. LSTM-based models can learn complex patterns in solar data and often perform well in research studies. However, they require more computational resources. This can make real-time deployment difficult on resource-constrained systems [22]. Researchers have also developed a CNN-based model for ultra-short-term PV forecasting. The model uses meteorological variables as inputs to improve PV power output prediction [23]. The Gated Recurrent Unit (GRU) network can achieve prediction results similar to LSTM. It uses fewer

parameters and requires less training time [24]. Researchers have also explored hybrid parallel architectures combining TCN, Transformer, and BiGRU to improve ultra-short-term PV prediction [25]. Temporal Convolutional Networks (TCNs) have attracted interest in their strong short-term predictive ability. TCNs are computationally efficient and capture long-range temporal dependencies through dilated convolutions [26]. A model combining a data-driven method with reduced physical modelling has been proposed for regional distributed PV forecasting, using grid division and a TCN-BiLSTM architecture alongside numerical weather prediction data [27].

Despite recent improvements, deep learning-based solar forecasting still has some limitations. Most studies focus on specific locations and fixed time frames, and they pay little attention to intra-hour forecasting [28]. A forecasting window of 15 to 30 min is important for energy storage decisions in building management and grid operations [9,14]. Previous very short-term forecasting research often used only three to five environmental features, prioritising computational efficiency over thorough atmospheric modelling. A lot of research also relies on datasets with limited temporal coverage and few features, which affects how well models generalise across different environmental conditions [29,30]. To better capture short-term atmospheric dynamics, feature engineering can expand a 17-feature dataset to 72. Using a larger feature space can improve forecasting reliability and prediction performance. It does not significantly increase the computational complexity [31]. Researchers have studied hour-ahead solar irradiance forecasting using multivariate GRU networks [32]. CNN-based models have also been used for PV output forecasting using a sliding-window algorithm [33]. Both approaches provide useful references for the design choices made in this work. Research on optimal generation scheduling and operating reserve management for PV-integrated microgrids shows that stochastic solar irradiance makes accurate PV generation forecasting very difficult, and that RNN-based models can help manage this uncertainty in stand-alone systems [34]. Probabilistic density prediction using deep learning has also been applied to small-scale solar generation [35], underscoring broader interest in uncertainty quantification for solar output estimation. Hybrid CNN-GRU architectures have been applied directly to multi-step-ahead photovoltaic power forecasting [36] and to short-term global horizontal irradiance forecasting [37], both showing that combining CNNs for spatial feature extraction with GRUs for temporal modelling can improve prediction accuracy in solar applications. PV power prediction models that combine deep CNNs with GRU units have been shown to yield accurate results [38]. PV generation forecasting using combined CNN and RNN networks has also been validated across different settings [39]. An empirical comparison of generic convolutional and recurrent networks for sequence modelling showed that TCN-style architectures can match or outperform recurrent models across a range of tasks [40]. Graph neural network approaches such as DEST-GNN have been proposed for multi-site intra-hour PV forecasting [41], pointing to a promising future direction that goes beyond single-site prediction by capturing spatial dependencies among multiple PV installations. A separate line of work deals with noisy labels in classification. For example, PSSCL selects clean training samples step by step and uses a contrastive loss [42]. These methods are designed for image labels that can be wrong. The present work is a regression task with the measured PV output as the target, so the two approaches are not directly comparable. However, measurement errors in PV data are related, and robust training is a possible direction for future work.

The existing literature does not adequately address three key points in intra-hour PV forecasting. Existing studies often do not use persistence or smart persistence as benchmarks, although they perform well at short horizons. The information available at forecast time is often not clearly stated, so the risk of data leakage cannot be assessed. Physical knowledge

known in advance, such as a clear-sky profile and the sunlight flag, is rarely used in the network's output stage.

To address these gaps, this work presents an Enhanced Temporal Convolutional Network (Enhanced TCN) architecture for intra-hour solar PV power forecasting in flexible energy systems. The primary purpose is to develop a model that can accurately forecast solar PV power generation across multiple temporal scales for real-time grid management and energy scheduling. The model is tested using data from a solar PV system deployed in Arizona, USA. The data is from 1 January 2017 to 31 August 2022. It consists of 196,776 samples collected every 15 min and is publicly available. This dataset is used because it is large and continuous. A large dataset helps the model learn diverse weather patterns and capture sudden short-term changes. Data from New Zealand is not used due to limited availability of long-term high-frequency public datasets [31].

Most existing models employ either convolution-based or attention-based methods. A unified model is developed by combining multiple components. The proposed approach uses dilated convolutions to capture patterns at different time scales. It uses multi-head attention to model long-range dependencies. Feature gating helps the model focus on important inputs. Seasonal decomposition separates the data's trend from abrupt changes. This integrated approach lets the model handle both short-term fluctuations and long-term trends more easily. As the number of input features increases, the model can represent real weather conditions more accurately. A physics-informed decoder then builds the forecast by correcting a persistence-based reference. It uses only information that is available at the forecast time. The key contributions of this research work are as follows:

- A physics-informed decoder is proposed. It builds the forecast as a learned correction to persistence and smart persistence using only values known at the forecast time.
- The decoder is combined with known building blocks (causal TCN, seasonal decomposition, attention pooling, feature gating), and an ablation study measures what each one adds.
- The model is evaluated without data leakage on 5.7 years of 15 min PV data with three random seeds, persistence references and the Diebold–Mariano test.
- After the bias correction, the model achieves R^2 = 0.928 and RMSE = 281.0 Wh, which is significantly lower than those of five deep learning models, persistence and smart persistence.

The rest of the paper is organised as follows. Section 2 presents the background and the proposed Enhanced TCN model. Section 3 describes the dataset and the methodology. Section 4 presents the results and discussion. Section 5 concludes the paper.

2. Background

Very short-term solar power forecasting at a 15 min horizon is challenging for energy trading and power grid management. The theoretical foundations of causal convolutions, dilated convolutions, and Temporal Convolutional Networks (TCNs) follow standard formulations in the TCN literature. The main contribution of this work is the physics-informed decoder. The encoder is built from known components: dilated causal convolutions, seasonal decomposition, attention pooling, and feature gating. For high-frequency time series, traditional recurrent models struggle with training stability and are computationally expensive. This paper proposes an Enhanced TCN architecture for 15-minute-ahead solar power forecasting to overcome these limitations, as shown in Figure 1.

For sequence tasks, Temporal Convolutional Networks [43,44] are a powerful substitute for recurrent models. TCNs use causal convolutions to maintain temporal order. While recurrent models handle sequences step by step, TCNs enable parallel processing over all time steps. As a result, TCNs use hardware more effectively and compute more quickly.

Causal convolution, which ensures that predictions at any time step depend only on the current and historical inputs, is the fundamental operation of TCNs. This operation can be expressed as follows for a discrete time series:

$$(F *_{\text{causal}} x)(t) = \sum_{k=0}^{K-1} f_k \cdot x_{t-k} \quad (1)$$

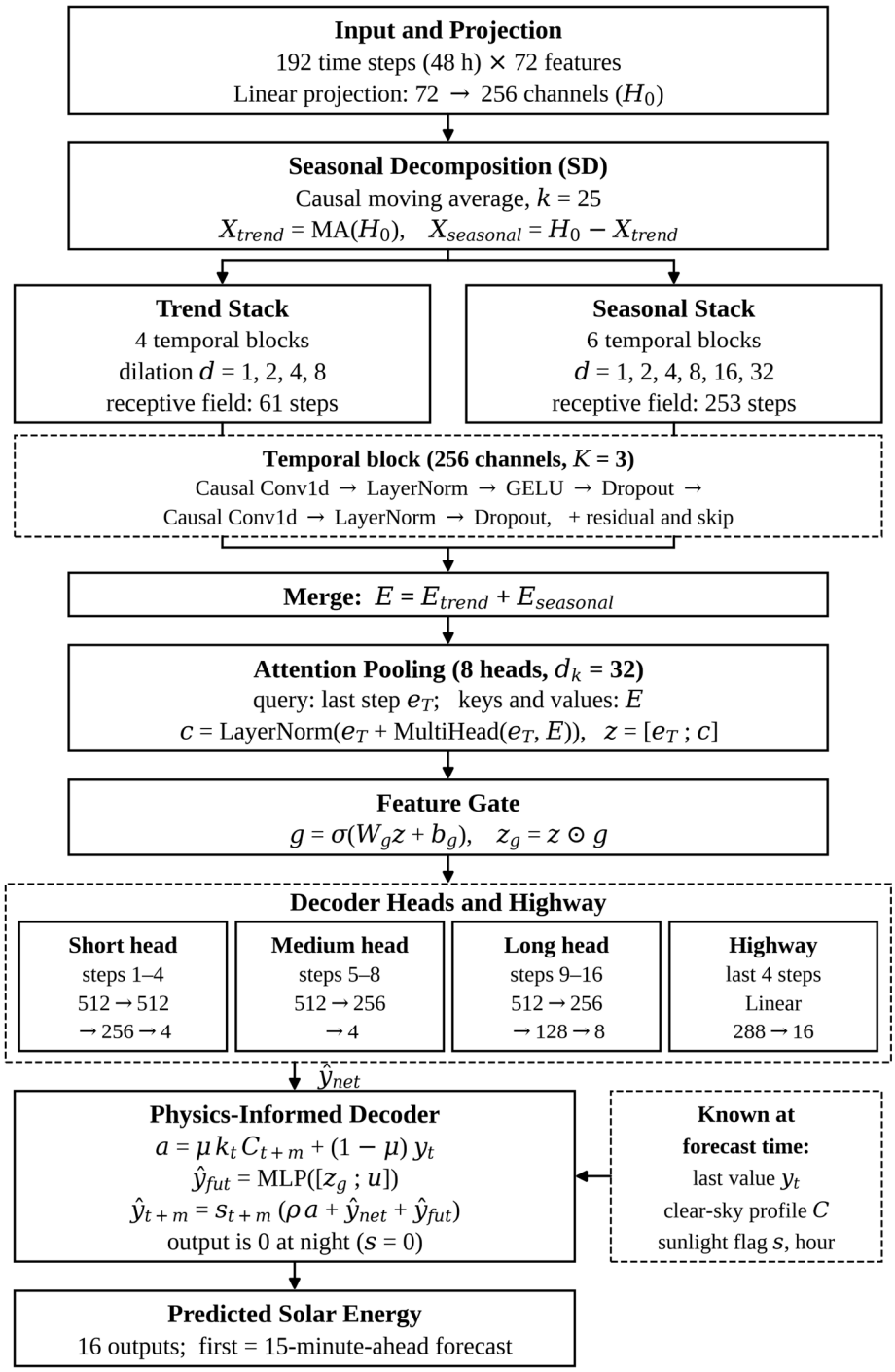


Figure 1. Architecture of the proposed Enhanced TCN model. Input: 192 timesteps × 72 features; hidden dimension: 256; dilations d ∈ {1, . . ., 8} (trend) and d ∈ {1, . . ., 32} (seasonal); attention heads H = 8; output: 15-minute-ahead forecast from the physics-informed decoder.

Therefore, the set of learnable filter coefficients is $F = \{f_0, f_1, \ldots, f_{k-1}\}$. x_t is the input at time step t, and K is the kernel size. The convolution uses only the current and previous

inputs, as shown by the term x_{t-k}. The model is prevented from viewing future data by this causality constraint. This is crucial for real-time forecasting. In practice, dilated causal convolutions increase the receptive field. As a result, the model can capture long-range temporal dependencies with minimal additional work. Temporal processing is based on standard causal convolutions. Capturing long-range dependencies requires deep networks. Dilated convolutions resolve this issue. With little additional processing, they greatly increase the receptive field. By adding a dilation factor, the dilated convolution expands Equation (1).

$$(F *_d x)(t) = \sum_{k=0}^{K-1} f(k) \cdot x_{t-d \cdot k} \tag{2}$$

Thus, the dilation factor is d. It regulates the distance between kernel components. The receptive field grows exponentially with network depth by stacking layers with exponentially expanding dilation rates $d_l = 2^l$ for layer l. This keeps computational cost linear.

Forecasting accuracy depends on the window length and the selected input features. An input window of 192 time steps for 15 min forward prediction was used. This is equivalent to 48 h of historical data at 15 min intervals. The window provides enough context to capture daily trends and recent weather variations; in practice, it works well. Three dimensions make up the input tensor $X \in \mathbb{R}^{\mathrm{BxTxD}}$. The batch size is B. The sequence length is T = 192. The number of features is D. Weather information, temporal features, and computed features are all included in the whole feature set:

$$X = \left[x_{weather}, x_{temporal}, x_{statistical}, x_{engineered}\right] \tag{3}$$

Temperature, relative humidity, atmospheric pressure and GHI are the most important meteorological variables in $x_{weather}$. These elements directly affect solar panels' power output and efficiency. Over time, patterns can be seen in the temporal component $\mathrm{x_{temporal}}$. It uses the time of day. Sine and cosine modifications are used to prevent sudden changes at midnight:

$$h_{sin} = \sin\left(\frac{2\pi h}{24}\right),\ cos = \cos\left(\frac{2\pi h}{24}\right) \tag{4}$$

Thus, h represents the hour of the day, ranging from 0 to 23. This illustration depicts gradual changes throughout time. Additionally, it provides the model with precise information about daily rhythms. The xstatistical component calculates rolling means, standard deviations, and maximum values for meteorological variables and historical power production over windows of 4-, 12-, 24-, and 48-time steps. The xengineered component includes derived features, including lagged power values at steps 1, 4, and 12; moving averages of meteorological variables; and a clear-sky power estimate computed from GHI and sunshine duration.

Solar power follows complex patterns across different time scales. The main focus here is 15 min prediction. Understanding trends and seasonal variations will improve results. A general framework is proposed for this purpose: the seasonal decomposition (SD) module; SD decomposes an input signal into trend and seasonal components. It helps the model capture slow and sudden changes in solar power. A linear layer first projects the input features to 256 channels. This gives the hidden sequence H_0. The trend is a moving average of H_0 that uses only the current and past time steps:

$$X_{\mathrm{trend}}(t) = \frac{1}{k}\sum_{j=0}^{k-1} H_0(t-j), \quad k = 25 \tag{5}$$

where $k = 25$ is the kernel size. For the first $k - 1$ time steps in the window, the first value is repeated on the left. The average at time step t therefore uses only values at or before t, so no future values are used. Since the data are sampled at 15 min intervals, the kernel covers approximately 6.25 h of historical observations. This smoothing operation removes rapid short-term oscillations while preserving the main daily irradiance trend. The seasonal component captures short-term fluctuations caused by rapidly changing weather conditions and is obtained by subtracting the smoothed trend component from the projected input:

$$X_{\text{seasonal}} = H_0 - X_{\text{trend}} \tag{6}$$

This decomposition allows the model to process smooth long-term behaviour and fast-changing short-term variations separately. The approach improves forecasting stability during rapidly changing weather conditions and helps the model better capture sudden irradiance fluctuations.

The kernel size of the moving average is fixed at $k = 25$, so the decomposition itself is not adaptive. For this reason, the module is called seasonal decomposition (SD) and not adaptive seasonal decomposition. The model adapts to the input through learnable TCN blocks, attention pooling, and feature gating that process the trend and seasonal components.

The trend and seasonal components are processed by two separate stacks of temporal blocks with dilated causal convolutions. The seasonal stack has six blocks, and the trend stack has four blocks. All blocks have 256 channels. As shown in Figure 1, the dilation rate of block l is $d_l = 2^l$. This gives dilation rates of 1, 2, 4, 8, 16 and 32 in the seasonal stack and 1, 2, 4 and 8 in the trend stack. In this paper, multi-scale refers to the time scales of the input patterns covered by different dilation rates. It does not refer to the forecast horizon. This paper considers a 15 min forecast horizon. Forecasting at longer horizons is left for future work. Each block extracts temporal features at a single time scale and uses a residual connection to stabilise gradients. Each block starts with a dilated causal convolution.

$$H_1^{(l)} = \text{Conv1D}\left(X^{(l)},\ W_1^{(l)},\ d_l,\ P_l\right) \tag{7}$$

where l is the block index, $X^{(l)} \in \mathbb{R}^{B \times T \times C}$ is the input feature map of block l with $C = 256$ channels, ${H_1}^{(l)}$ is the output of the first dilated causal convolution, ${W_1}^{(l)}$ are the learnable convolution weights with weight normalisation, and P_l is the padding needed to keep the length of the sequence. The padding is applied only on the left side, so that the convolution is causal:

$$P_l = (K - 1) \times d_l = 2 \times 2^l \tag{8}$$

The size of the kernel is $K = 3$. The left padding keeps the output the same length as the input, which is needed for the residual connections. After the first convolution, layer normalisation is applied over the channels:

$$H_2^{(l)} = \gamma \cdot \frac{H_1^{(l)} - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta \tag{9}$$

The parameters γ and β are trainable scaling and shifting parameters of the layer normalisation. They are learned during training by backpropagation and use the default PyTorch initialisation. The terms μ and σ^2 are the mean and variance over the channels at each time step, and $\varepsilon = 10^{-5}$ is a small constant for numerical stability. Layer normalisation does not depend on the batch statistics. The normalised output goes through the GELU activation function, and dropout is applied after it:

$$H_3^{(l)} = \text{Dropout}\left(\text{GELU}\left(H_2^{(l)}\right),\ p = 0.25\right) \tag{10}$$

To reduce overfitting, a dropout rate of $p = 0.25$ was used. The GELU activation introduces non-linearity, which enables the learning of complex temporal patterns. The second dilated convolution with the same parameters comes next.

$$H_4^{(l)} = \text{Conv1D}\left(H_3^{(l)}, W_2^{(l)}, d_l, P_l\right) \tag{11}$$

where $H_4{}^{(l)} \in \mathbb{R}^{B\times T\times C}$ is the output of the second dilated causal convolution and $W_2{}^{(l)}$ are its learnable weights. It has the same kernel size $K = 3$, dilation d_l and padding P_l as the first convolution. This output is then passed through layer normalisation and dropout:

$$H_5^{(l)} = \text{Dropout}\left(\text{LayerNorm}\left(H_4^{(l)}\right),\ p = 0.25\right) \tag{12}$$

The residual connection adds the processed features to the input of the block:

$$X^{(l+1)} = X^{(l)} + H_5^{(l)} \tag{13}$$

Because all blocks have the same number of channels, the residual path needs no projection. In addition, the outputs $H_5{}^{(l)}$ of all blocks in a stack are summed as skip connections and added to the output of the last block:

$$E_{\text{stack}} = \text{LayerNorm}\left(X^{(L)} + \sum_{l=0}^{L-1} H_5^{(l)}\right) \tag{14}$$

where L is the number of blocks in the stack ($L = 6$ for the seasonal stack and $L = 4$ for the trend stack). The skip connections pass features from all time scales directly to the output and help gradients flow through the deep network. The outputs of the two stacks are added to give the encoded sequence $E = E_{trend} + E_{seasonal}$, with $E \in \mathbb{R}^{B\times T\times C}$.

After the two stacks, an attention pooling layer summarises the encoded sequence. The dilated convolutions capture patterns at short, medium and long ranges. The attention layer then selects the time steps of the window that are most useful for the forecast. The encoded state of the last time step, e_T, serves as the query, and the full encoded sequence E provides the keys and values. Multi-head attention with $H = 8$ heads is used. For each head h, the query, key, and value projections are first calculated:

$$Q_h = e_T W_Q^h, \quad K_h = E W_K^h, \quad V_h = E W_V^h \tag{15}$$

The matrices $W_Q{}^h$, $W_K{}^h$ and $W_V{}^h$ are learned projections of size $d_{model} \times d_k$, where $d_{model} = 256$ and $d_k = d_{model}/H = 256/8 = 32$ is the dimension per head. Scaled dot-product attention is computed for every head.

$$Attention_h(Q_h, K_h, V_h) = \text{softmax}\left(\frac{Q_h K_h^T}{\sqrt{d_k}}\right) V_h \tag{16}$$

The scale factor $\sqrt{d_k}$ keeps the dot products from becoming too large. The softmax produces the attention weights. These weights show each time step's contribution to the forecast. The outputs of all attention heads are then concatenated and linearly transformed:

$$\text{MultiHead}(e_T, E) = \text{Concat}(\text{head}_1, \ldots, \text{head}_8) W_O \tag{17}$$

The output projection matrix W_O has the size $d_{model} \times d_{model}$. A residual connection with layer normalisation gives the attention context c:

$$c = \text{LayerNorm}(e_T + \text{MultiHead}(e_T, E)) \tag{18}$$

The encoded state of the last time step and the attention context are then joined into one vector:

$$z = [e_T; c], \quad z \in \mathbb{R}^{512} \tag{19}$$

For fifteen-minute predictions, not every element of this vector is equally significant. Depending on the weather, time of day, and recent power history, some elements are more useful than others. An adaptive feature-gating approach is used to handle this. A learned linear layer with sigmoid activation gives one gate value for each element:

$$g = \sigma(W_g z + b_g) \tag{20}$$

Here σ is the sigmoid activation, so the gate values are between 0 and 1. The learnable parameters are $W_g \in \mathbb{R}^{512\times512}$ and $b_g \in \mathbb{R}^{512}$. The gate is applied element by element:

$$z_g = z \odot g \tag{21}$$

Here $\odot$ is element-wise multiplication. The gating lets the model assign more weight to useful elements and less to others, and it changes with weather and time of day. The decoder converts the gated vector into the forecast. The network has 16 outputs, from 15 min to 4 h ahead, and the training loss uses all of them. This paper evaluates only the first output: the 15-minute-ahead forecast. The decoder has three heads. The short-term head predicts steps 1 to 4. Its first hidden layer has 512 units:

$$h_1 = \text{GELU}(W_1 z_g + b_1) \in \mathbb{R}^{512} \tag{22}$$

where $W_1 \in \mathbb{R}^{512\times512}$ and $b_1 \in \mathbb{R}^{512}$ are learnable parameters. Dropout follows to reduce overfitting. A second fully connected layer with 256 units further processes the representation:

$$h_2 = \text{GELU}(W_2\ \text{Dropout}(h_1) + b_2) \in \mathbb{R}^{256} \tag{23}$$

where $W_2 \in \mathbb{R}^{256\times512}$ and $b_2 \in \mathbb{R}^{256}$ are learnable parameters. After another dropout layer, a linear layer gives the first four steps:

$$\hat{y}_{1:4} = W_3\ \text{Dropout}(h_2) + b_3 \tag{24}$$

where $W_3 \in \mathbb{R}^{4\times256}$ and $b_3 \in \mathbb{R}^4$. The medium-term head has one hidden layer of 256 units and predicts steps 5 to 8. The long-term head has two hidden layers of 256 and 128 units and predicts steps 9 to 16. The three outputs are joined:

$$\hat{y}_{\text{dec}} = [\hat{y}_{1:4};\ \hat{y}_{5:8};\ \hat{y}_{9:16}] \in \mathbb{R}^{16} \tag{25}$$

A linear highway from the most recent inputs is then added to the decoder output:

$$\hat{y}_{\text{net}} = \hat{y}_{\text{dec}} + W_{\text{hw}} x_{\text{recent}} + b_{\text{hw}} \tag{26}$$

where $x_{recent} \in \mathbb{R}^{288}$ contains the 72 features of the last four time steps, $W_{hw} \in \mathbb{R}^{16\times288}$ and $b_{hw} \in \mathbb{R}^{16}$. The highway gives the model a direct linear path from the latest measurements to the forecast. The network output is $\hat{y}net$.

The last part of the model is a physics-informed (PI) decoder. It does not predict the solar energy directly. It starts with a simple reference forecast, the anchor, and the network learns a correction to it. The anchor is built from the last measured value and a clear-sky

profile. The clear-sky profile $C(d, s)$ is fitted on the training data only. For each day of the year d and each 15 min slot of the day s, it is the 95th percentile of the measured energy within 15 days around d:

$$C(d, s) = Q_{0.95}\{y_\tau : s(\tau) = s,\ |\, d(\tau) - d\,| \leq 15\} \tag{27}$$

where $Q_{0.95}$ is the 95th percentile over the training time steps τ that have the slot s and a day of the year within 15 days of d. The profile depends only on the calendar, so its value is known in advance for every future time step. The clear-sky index at the forecast time t is the ratio of the measured energy to the profile:

$$k_t = \min\left(\max\left(\frac{y_t}{C_t}, 0\right), 1.5\right) \tag{28}$$

The index is calculated only when C_t exceeds 5% of the profile maximum. At night and in the first and last minutes of the day, the last valid value is kept. Two reference forecasts are then calculated for each lead time m. Smart persistence keeps the clear-sky index constant, and plain persistence keeps the last measured value constant. The anchor is a blend of the two:

$$a_{t+m} = \mu_m\, k_t\, C_{t+m} + (1 - \mu_m) y_t \tag{29}$$

where μ_m is a weight between 0 and 1 that is learned for each lead time. In this way, the model can use the better of the two references at each lead time. The decoder also uses values of the target time steps that are known in advance. These are the clear-sky profile, the sunlight flag and the sine and cosine of the hour. Together with the two reference forecasts, they form the vector u. A small network with one hidden layer of 256 units gives a second correction from the gated vector and u:

$$\hat{y}_{\text{fut}} = W_5\, \text{GELU}\big(W_4\big[z_g;\ u\big] + b_4\big) + b_5 \tag{30}$$

The forecast is the weighted anchor plus the two corrections:

$$\hat{y}_{t+m} = s_{t+m}(\rho_m\, a_{t+m} + \hat{y}_{\text{net},m} + \hat{y}_{\text{fut},m}) \tag{31}$$

where ρ_m is a learned anchor weight for each lead time, and st+m is the sunlight flag for the target step. The sunlight flag is one when the sun is above the horizon and zero at night, so the forecast is zero at night. It is calculated from the sunrise and sunset times and is therefore known in advance. In the implementation, these operations are carried out on the standardised values. The correction output layers are initialised to zero, and ρ_m starts at 0.95. The untrained model is therefore almost equal to the anchor, and training only has to learn the correction. All inputs of the PI decoder are available at the forecast time. They are the last measured values and quantities that depend only on the calendar. No future measurement is used. The baseline models do not receive these decoder inputs. For this reason, the Enhanced TCN is also evaluated without the PI decoder (Section 4). The first forecast element is the predicted solar energy 15 min ahead.

In this paper, the term physics-informed refers to three pieces of physical knowledge built into the decoder. First, on a clear day, a PV system's output follows the sun's position, which depends on the day of the year and the time of day. Equation (27) represents this clear-sky profile. Second, cloud conditions change more slowly than the sun's position. For this reason, the clear-sky index of Equation (28) is kept constant for the next step, which is the assumption of smart persistence. Third, a PV system produces no energy when the sun is below the horizon, so the forecast is set to zero at night, and it cannot be negative. The third point is a hard constraint. The first two points enter the forecast through the anchor.

No physical equation is added to the loss function. The clear-sky profile is estimated from the training data, not calculated from a physical clear-sky model. The term is therefore used as prior physical knowledge at the network's output stage. It is not used in the sense of a physics-informed neural network that solves differential equations.

At this stage, training the model requires a loss function that penalises prediction errors and accounts for solar power characteristics. Here, we use an asymmetric loss that penalises under-predictions more strongly to favour safe forecasts for the grid:

$$\mathcal{L} = \frac{1}{NM}\sum_{i=1}^{N}\sum_{m=1}^{M} \lambda_m \, w_{i,m}|\hat{y}_{i,m} - y_{i,m}| \tag{32}$$

where N is the number of samples in a batch, $M = 16$ is the number of output steps, $\hat{y}_{i,m}$ is the predicted value and $y_{i,m}$ is the actual measurement, both in standardised units. The step weights λ_m are 5, 3, 2 and 1 for the first four steps, 1 for steps 5 to 8, 0.3 for steps 9 to 12 and 0.2 for steps 13 to 16, so the 15 min step has the largest weight. A small second term with a weight of 0.03 is added to this loss. It is the mean absolute value of the standardised output at the time steps without sunlight. The weights $w_{i,m}$ penalise under-predictions more strongly to favour safe grid operation.

The model has 16 outputs, but this paper evaluates only the first output (15 min). The other outputs are used in training as additional targets, and the step weights give the largest weight to the 15 min step. All baseline models are trained with the same 16 outputs and the same step weights, so the comparison between the models is fair. This design was chosen because the model is being extended to longer horizons. With this design, the same trained model can be used for the longer horizons. A comparison with a model that has a single 15 min output is left for future work.

$$w_{i,m} = \begin{cases} \alpha, & \hat{y}_{i,m} < y_{i,m} \text{ (under-prediction)} \\ 1, & \hat{y}_{i,m} \geq y_{i,m} \text{ (over-prediction)} \end{cases} \tag{33}$$

where $\alpha = 2.5$ is the asymmetric penalty factor. An under-prediction is therefore penalised 2.5 times more than an over-prediction of the same size. As a result, the trained models tend to over-predict. This bias is removed after training by a correction that is fitted on the validation set (Section 3). The same loss function is used for all models. The loss is minimised using the AdamW optimizer with a learning rate of 2×10^{-4} and weight decay of 10^{-3}. The learning rate linearly warms up over the first five epochs and then follows cosine decay. The gradients are clipped to a norm of 1.0. To check how much history the convolutions can use, the receptive field of the two stacks is calculated. With L blocks, two convolutions per block, kernel size $K = 3$, and dilation rates that double in each block, the receptive field is:

$$RF = 1 + 2\sum_{l=0}^{L-1}(K-1)\cdot 2^{l} = 1 + 2(K-1)\left(2^{L}-1\right) \tag{34}$$

For the seasonal stack with $L = 6$ blocks and the trend stack with $L = 4$ blocks, we obtain:

$$RF_{\text{seasonal}} = 1 + 4 \times 63 = 253, \quad RF_{\text{trend}} = 1 + 4 \times 15 = 61 \tag{35}$$

The seasonal stack's receptive field is 253 time steps, or about 63 h. It is larger than the 192-time-step input window, so the last time step can use the full window. The trend stack's receptive field is 61 time steps, or about 15 h. This is enough for the slowly changing trend component.

The 192-time-step input window corresponds to 48 h of historical data. It covers two full daily cycles, giving the model the daily pattern and recent weather changes.

3. Methodology

The Enhanced TCN model was applied to solar power data of a solar farm located in Arizona, USA. The exact location of the farm is not disclosed by the data provider for privacy reasons. The data were collected from 1 January 2017 to 31 August 2022. It consists of 196,776 samples taken every fifteen minutes. Each sample contains weather information such as GHI, temperature, pressure, humidity, wind speed, cloud cover and rain. It also includes power output in watt-hours. The data also provide sunshine hours, day length, and whether sunlight is present. Feature engineering was applied to convert the 17 raw features into 72 features, which are summarised in Table 1. The series contains no missing values or duplicated timestamps, and energy is never negative. Compared with a complete 15 min grid, 1824 timestamps (0.92%) are missing across 10 gaps, with the longest gap lasting about five days. These gaps were not filled, and no values were removed as outliers. A window that crosses a gap treats the time steps before and after the gap as neighbours. In the test set, 1920 of the 29,310 windows (6.55%) contain such a gap. Section 4 examines the effect on the results. The target variable, Energy delta (Wh), is the energy the PV system produces in each 15 min interval. The data include night-time samples. The test set covers 17 October 2021 to 31 August 2022.

Data preprocessing transforms the raw measurements into 72 input features. Time features use the sine and cosine of the hour, month and day of the year, together with the day of the week. For six weather variables, exponential moving averages and rolling means over 4, 12, and 24 steps are computed. For PV energy, rolling means, standard deviations, and maxima over 4, 12, 24, and 48 steps are computed, together with lagged values at 1, 4, and 12 steps. Changes in GHI over 1, 4, and 12 steps are also included. A GHI-sunshine index is calculated from the measured GHI and the sunlight fraction. It is an empirical index of current irradiance, not a physical clear-sky model. The ratio of measured energy to this index is used as a feature. The mean and maximum energy for each hour of the day are computed from the training set only. All rolling features use only current and past values. The model therefore uses only measurements that are available at the forecast time. No weather forecasts and no future measurements are used. All features are normalised using z-score normalisation. The scalers are fitted on the training data and then applied to the validation and test data. The clear-sky profile of the PI decoder (Equation (27)) is also fitted on the training set only. It is not one of the 72 input features.

The original 17 raw variables are expanded to 72 input features to capture short-term changes in solar power. The additional features include rolling statistics, cyclic time features, lagged power values, and GHI differences. Rolling statistics and exponential moving averages make up the largest part of the feature set, with 36 of the 72 features. They describe weather changes and PV energy across different time scales. This provides the model with information about recent changes, rather than requiring it to learn these patterns from raw measurements alone.

The dataset is temporally partitioned into training, validation, and test sets, with 137,743, 29,516, and 29,517 data points, respectively. This forms approximately 70%, 15%, and 15% of the data. This split ensures the model is tested on future data. These numbers are raw time steps. The series is split in time order before the sliding windows are generated, so no window contains time steps from two parts. With an input window of 192 steps and 16 output steps, this gives 137,536 training, 29,309 validation and 29,310 test windows. The target is normalised using the training set z-score. The model outputs are transformed back to Wh, and negative values are set to zero before calculating errors. The first windows

of the validation and test sets lie fully within their respective parts. Their rolling and lag features use a few earlier values from the preceding part.

Table 1. Input feature set of the Enhanced TCN model (D = 72).

Category	Features	Windows/Lags	Count
Raw meteorological	GHI, temperature, pressure, humidity, wind speed, rainfall, snowfall, cloud cover, weather type.	—	9
Raw solar	Sunlight flag, sunlight time, day length, sunlight fraction (two columns).	—	5
Weather type (one-hot)	Five weather classes.	—	5
Cyclic temporal	Sin/cos of hour-of-day, month, day-of-year.	—	6
Calendar	Day of week.	—	1
GHI differences	Change in GHI.	1, 4, 12 steps	3
Exponential moving average	GHI, temperature, pressure, humidity, wind speed, cloud cover.	Span of 4 steps	6
Rolling mean (weather)	GHI, temperature, pressure, humidity, wind speed, cloud cover.	4, 12, 24 steps	18
Rolling statistics (PV energy)	Mean, standard deviation and maximum of PV energy.	4, 12, 24, 48 steps	12
Lagged power	PV energy at lag-1 (15 min), lag-4 (1 h), lag-12 (3 h).	1, 4, 12 steps	3
GHI-sunshine index	GHI-sunshine index (measured GHI and sunlight fraction), ratio of PV energy to this index.	—	2
Hourly statistics	Mean and maximum PV energy for each hour of the day (training set only).	—	2
Total			72

Note: The total 9 + 5 + 5 + 6 + 1 + 3 + 6 + 18 + 12 + 3 + 2 + 2 = 72. All features are z-score normalised using scalers fitted on training data only.

The following parameters are used for training: batch size 32, AdamW optimiser with a learning rate of 2×10^{-4}, and weight decay of 10^{-3}. The learning rate linearly warms up over 5 epochs, then decays with a cosine schedule. The dropout rate is 0.25. Training runs for at most 100 epochs with early stopping on the validation loss (patience = 10), and the weights of the epoch with the lowest validation loss are used for testing. The asymmetric loss applies a penalty factor of $\alpha = 2.5$ to under-predictions to favour safe forecasts for the grid. All models are trained with three random seeds (42, 43 and 44). The results are reported as the mean and standard deviation over the seeds. The asymmetric loss makes the forecasts of all models too high on average. Therefore, a bias correction is applied after training. For each model, the mean error on the validation set is calculated separately for day and night and for each lead time. It is then subtracted from the test forecasts of that model. The correction uses only the validation set and applies the same procedure to all models. The results without the correction are also reported.

The remaining implementation details are as follows. The 72 features are mapped to 256 channels by a linear layer. All convolutions have a kernel size of 3 and use weight normalisation. The attention pooling has 8 heads. The first decoder head has hidden layers of 512 and 256 units, the second head has one hidden layer of 256 units, and the third head has hidden layers of 256 and 128 units. The network of Equation (30) has one hidden layer of 256 units. Its input includes the smart persistence value, the clear-sky profile, the sunlight flag, the sine and cosine of the hour, and the last measured value for each of the 16 output steps. The clear-sky index is calculated only when the clear-sky profile is above 5% of its maximum. Otherwise, the last valid value is used—the learning rate decays to 2% of its starting value. The training windows are shuffled in each epoch, and the validation and test windows are kept in time order. With these settings, the Enhanced TCN has 5,355,984 parameters.

All experiments were run on an NVIDIA GeForce RTX 4080 GPU (NVIDIA Corporation, Santa Clara, CA, USA) with 16 GB of memory on Linux using PyTorch 2.7.1 and CUDA 11.8.

The model's performance is evaluated using RMSE, MAE, and R^2. RMSE gives more weight to larger prediction errors. MAE represents the average prediction error. R^2 indicates how much of the variation in the target values the model explains. Mean bias is the mean of predicted minus measured values, so positive bias means over-prediction. The proposed Enhanced TCN is compared with five deep learning models: CNN-GRU, 1D CNN-GRU, LSTM, Plain TCN and Transformer. CNN-GRU has four convolutional layers with 1024, 512, 256, and 128 filters, followed by three GRU layers with 200, 100, and 50 units. The 1D CNN-GRU has the same layer sizes with one-dimensional convolutions. The LSTM has three layers with 256 hidden units. The Plain TCN has eight blocks with 512 channels and no decomposition, attention or gating. The Transformer has four encoder layers with a model dimension of 512 and 8 heads. All six models use the same dataset splits, preprocessing steps, loss function and training settings. No separate hyperparameter search was carried out for the individual models. Section 4.2 reports the number of parameters and the computational cost of each model. A persistence model is used as a reference. It takes the last measured value as the forecast for the next step. The Diebold–Mariano test [45] with squared errors is used to check if the difference between the Enhanced TCN and each other model is statistically significant. The test variance is estimated with the Newey–West estimator (Bartlett kernel, lag of 14 steps), which accounts for autocorrelation in the forecast errors. A smart persistence model [46] is used as a second reference. It keeps the clear-sky index in Equation (28) constant and multiplies it by the target step's clear-sky profile. All models are evaluated at the 15 min horizon. The contribution of the model components is examined in an ablation study, in which the PI decoder, the seasonal decomposition, the attention pooling or the feature gating is removed. The importance of the input features is measured by permutation. One group of features is shuffled between the test windows, and the increase in RMSE is recorded. The models are implemented in Python 3.13 using NumPy 2.3.3, Pandas 2.3.3, and scikit-learn 1.7.2.

4. Results and Discussion

This work compares the proposed Enhanced TCN with five deep learning models: CNN-GRU, 1D CNN-GRU, LSTM, Transformer, and Plain TCN. Persistence and smart persistence are included as reference models. The Enhanced TCN is also evaluated without the physics-informed decoder. Table 2 and Figure 2 show the test results at the 15 min horizon after bias correction, as the mean and standard deviation over three random seeds. The Enhanced TCN achieves an RMSE of 281.0 $\pm$ 4.6 Wh and an R^2 of 0.928 $\pm$ 0.002. These are the best values of all models. The RMSE is 7.0% lower than LSTM (302.1 Wh), 10.7%

lower than Transformer (314.6 Wh), 11.4% lower than CNN-GRU (317.3 Wh), 11.7% lower than Plain TCN (318.1 Wh), and 18.8% lower than 1D CNN-GRU (346.1 Wh). It is also 8.3% lower than persistence (306.3 Wh) and 4.9% lower than smart persistence (295.3 Wh). The Enhanced TCN has an MAE of 108.8 ± 2.9 Wh. This is lower than the MAE of all deep learning models and of persistence, but smart persistence has a lower MAE (102.5 Wh). The proposed model therefore mainly reduces the large errors. Without the physics-informed decoder, the Enhanced TCN's RMSE is 295.3 ± 10.9 Wh, the same as smart persistence. The decoder reduces the RMSE by 4.8% and improves stability across seeds.

Table 2. Test results at the 15 min horizon after the bias correction (mean ± standard deviation over three random seeds). The last column gives the RMSE without the bias correction.

Model	RMSE (Wh)	MAE (Wh)	R^2	Mean Bias (Wh)	RMSE Without Correction (Wh)
Enhanced TCN	**281.0 ± 4.6**	108.8 ± 2.9	**0.928 ± 0.002**	9.6	300.1 ± 10.4
Enhanced TCN (no PI)	295.3 ± 10.9	117.2 ± 2.9	0.921 ± 0.006	24.5	311.7 ± 10.5
Smart persistence	295.3	**102.5**	0.921	**−0.0**	**295.3**
LSTM	302.1 ± 1.9	124.9 ± 1.6	0.917 ± 0.001	22.3	336.6 ± 5.6
Persistence	306.3	118.0	0.915	−0.1	306.3
Transformer	314.6 ± 12.5	135.3 ± 7.8	0.910 ± 0.007	23.0	357.0 ± 17.7
CNN-GRU	317.3 ± 4.9	130.7 ± 2.5	0.909 ± 0.003	19.8	349.5 ± 2.0
Plain TCN	318.1 ± 10.3	135.8 ± 6.9	0.908 ± 0.006	10.3	346.5 ± 22.9
1D CNN-GRU	346.1 ± 10.7	144.4 ± 7.0	0.891 ± 0.007	22.2	370.7 ± 17.9

Bold values indicate the best performance in each column. Persistence and smart persistence do not depend on a random seed and are not corrected.

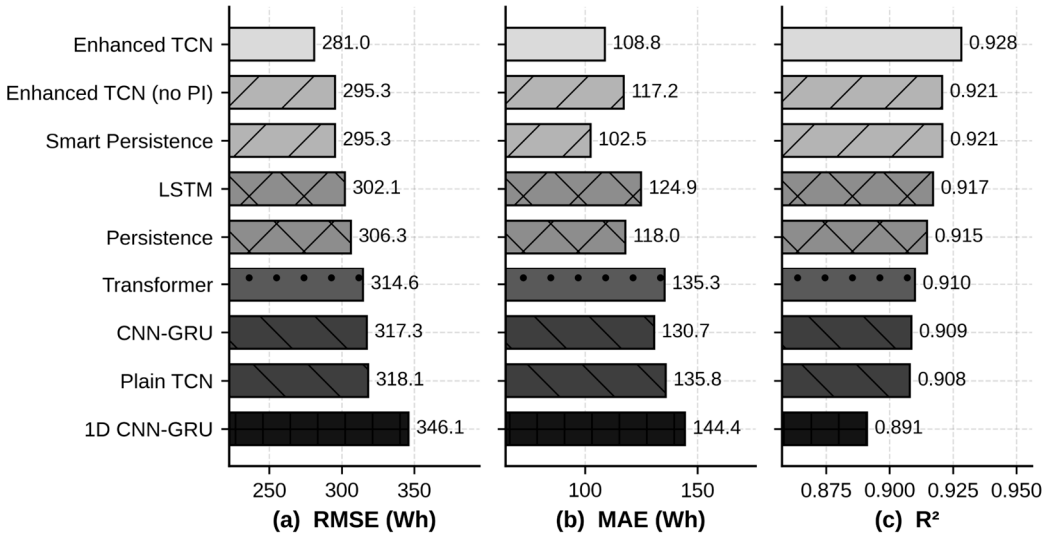


Figure 2. Comparison of all models at the 15 min horizon after the bias correction: RMSE in (**a**), MAE in (**b**), and R^2 in (**c**). The bars show the mean over three random seeds.

The last column of Table 2 gives the RMSE without the bias correction. All deep learning models have a positive mean bias between 70.6 and 113.9 Wh, because the asymmetric loss penalises under-prediction. This bias increases the RMSE. Without the correction, the Enhanced TCN has an RMSE of 300.1 ± 10.4 Wh. This is the lowest value among the deep learning models and is 2.0% lower than persistence. However, it is 1.6% higher than smart persistence (295.3 Wh). The advantage over smart persistence therefore appears only after

bias correction. The correction is fitted on the validation set. It reduces the mean bias of the Enhanced TCN on the test set from 70.6 to 9.6 Wh.

The bias correction is the same simple step for every trained model. After training, the mean error of the model on the validation set is calculated for the daytime and for the night. For the 15 min horizon, these are two constants per model, and they are subtracted from the test forecasts. No test data are used to find them. Each model uses its own constants. The two persistence models are not trained with the asymmetric loss and have a mean bias close to zero (see Table 2), so they are not corrected. The correction reduces the baselines' RMSE more than that of the Enhanced TCN, for example, by 34.5 Wh for the LSTM and 19.1 Wh for the Enhanced TCN. With and without the correction, the Enhanced TCN has the lowest RMSE of the deep learning models, and the LSTM is the best baseline. The bias comes from the asymmetric loss, which was chosen intentionally. The correction is therefore part of the forecasting procedure, not a tuning step on the test data.

Table 3 shows the Diebold–Mariano test between the Enhanced TCN and each other model after the bias correction. A negative value means that the Enhanced TCN has the smaller squared error. The statistics include the Newey–West correction for autocorrelation. The differences are significant at the 5% level for all models and all three seeds, including persistence and smart persistence. The same test was carried out without the bias correction. The Enhanced TCN is again significantly better than the five deep learning models in every seed. Compared with persistence, it is significantly better in two seeds and not different in one. Against smart persistence, there is no significant difference in two seeds, and smart persistence is significantly better in one seed. Therefore, Enhanced TCN outperforms all deep learning models in both cases, and it outperforms smart persistence only after bias correction.

Table 3. Diebold–Mariano test between the Enhanced TCN and each other model after the bias correction (15 min horizon, squared errors). The table gives the test statistic and the *p*-value in brackets for each seed. The variance is calculated with the Newey–West estimator with a lag of 14 steps. A negative value means that the Enhanced TCN has a smaller error.

Model Compared	Seed 42	Seed 43	Seed 44
LSTM	−4.50 (<0.001)	−3.59 (<0.001)	−5.24 (<0.001)
Plain TCN	−4.95 (<0.001)	−6.44 (<0.001)	−6.14 (<0.001)
CNN-GRU	−6.69 (<0.001)	−4.49 (<0.001)	−7.84 (<0.001)
Transformer	−5.87 (<0.001)	−8.95 (<0.001)	−6.80 (<0.001)
1D CNN-GRU	−5.13 (<0.001)	−6.48 (<0.001)	−5.67 (<0.001)
Enhanced TCN (no PI)	−6.13 (<0.001)	−2.28 (0.022)	−2.86 (0.004)
Persistence	−11.16 (<0.001)	−5.48 (<0.001)	−8.59 (<0.001)
Smart persistence	−7.06 (<0.001)	−2.54 (0.011)	−5.50 (<0.001)

The sensitivity of the test to the lag of the Newey–West estimator was also examined. Table A1 in Appendix A gives the Diebold–Mariano statistics after the bias correction for lags of 0 (no correction for autocorrelation), 14, 96 (one day) and 672 steps (one week). The statistics decrease as the lag increases because longer lags allow more autocorrelation in the loss differences. However, the conclusions remain the same. Against the five deep learning models and persistence, all differences are significant ($p \leq 0.001$) for all lags and all seeds. Against smart persistence, the largest *p*-value is 0.011 for a lag of 14 steps, 0.020 for a lag of one day and 0.022 for a lag of one week. Against the model without the physics-informed decoder, the differences are significant for all lags in two seeds. In the third seed (43), the *p*-value is 0.022 for a lag of 14 steps, 0.050 for a lag of one day and 0.036 for a lag of one

week, so this one comparison is at the limit of significance. Without the bias correction, the conclusions are also the same for all lags. The autocorrelation of the loss differences is small (between −0.09 and −0.02 at a one-step lag against smart persistence), so the correction lowers the test statistics but does not change the conclusions.

For each model, the scatter plots in Figure 3 show actual versus predicted values on the test set for one of the three runs (seed 44) after the bias correction. Each point is one 15 min forecast. In this run, the Enhanced TCN in Figure 3d has the least dispersion around the diagonal, with R^2 = 0.930 and RMSE = 277.7 Wh. The 1D CNN-GRU in Figure 3b has the widest scatter, with an R^2 of 0.889. The CNN-GRU in Figure 3a, the LSTM in Figure 3c, the Transformer in Figure 3e, and the Plain TCN in Figure 3f have R^2 values between 0.911 and 0.916.

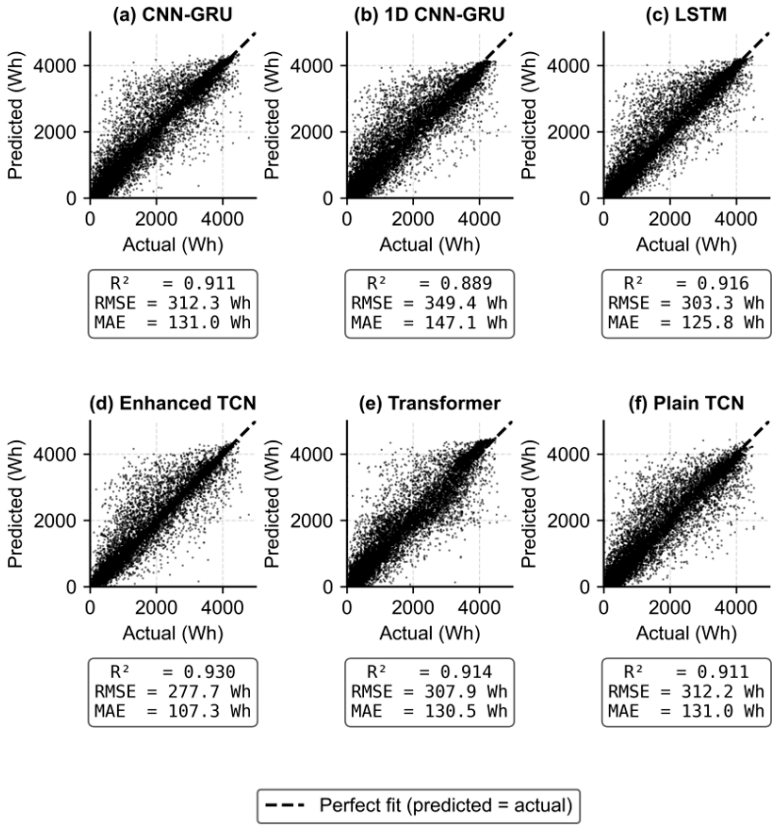


Figure 3. Actual versus predicted solar power at the 15-minute-ahead horizon for the six trained models (seed 44, after the bias correction). The dashed line denotes perfect prediction.

The forecasts are also shown as a continuous time series. Figure 4 shows the 15-minute-ahead forecasts of the six models from 10 to 16 April 2022 (seed 44, after the bias correction). At each time step, the models receive the latest measurements and predict the next 15 min interval. The values in the legend are calculated for these 7 days only. The period includes clear days (12, 13, and 16 April), days with rapid changes due to clouds (10 and 11 April), and two days with low generation (14 and 15 April). In this period, the Enhanced TCN has the lowest RMSE (346.3 Wh), followed by the Transformer (365.5 Wh) and the LSTM (386.9 Wh). The Plain TCN, CNN-GRU, and 1D CNN-GRU have RMSE values of 395.5, 400.6, and 413.0 Wh, respectively. All models perform worse than on the full test set because this period includes days with rapid changes.

Figure 5 shows the forecasts from 6 to 19 June 2022. In this period, the Enhanced TCN has the lowest RMSE (346.2 Wh) and the highest R^2 (0.906). The baselines have RMSE values of 365.7 Wh (CNN-GRU), 370.1 Wh (Transformer), 373.1 Wh (LSTM), 381.5 Wh

(Plain TCN) and 402.1 Wh (1D CNN-GRU). Most days in this period show rapid output changes, while the last two days are clear.

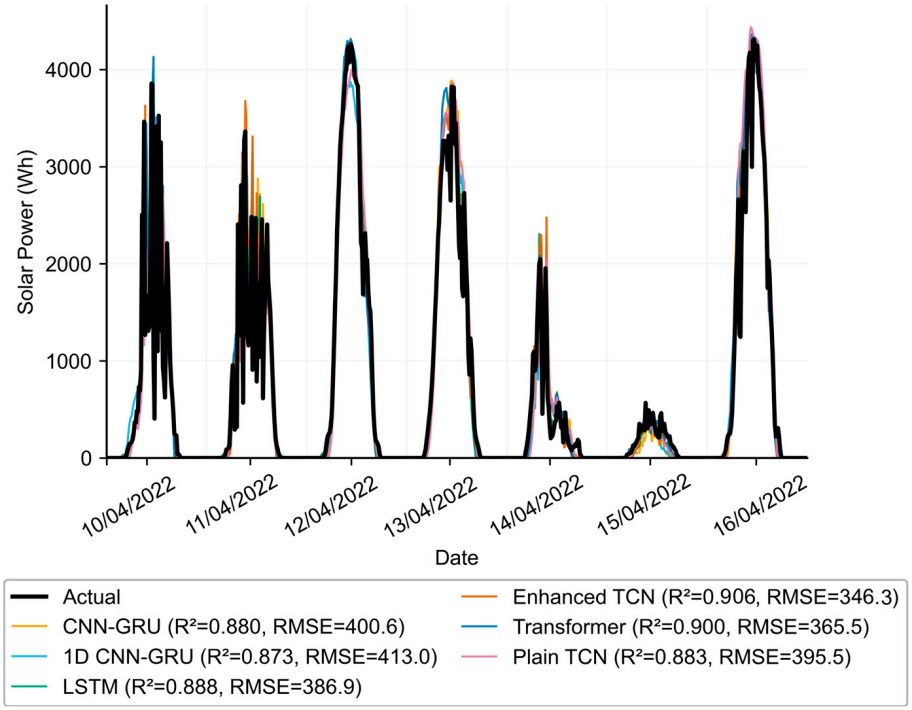


Figure 4. The 15-minute-ahead forecasts of the six trained models from 10 to 16 April 2022 (seed 44, after the bias correction). The legend gives R^2 and RMSE for this period.

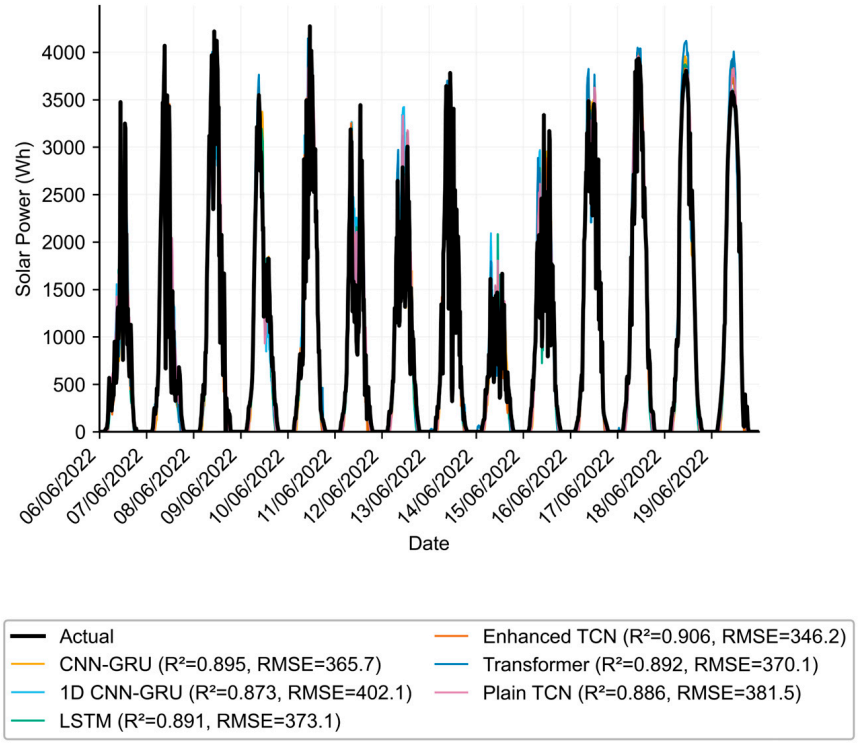


Figure 5. The 15-minute-ahead forecasts of the six trained models from 6 to 19 June 2022 (seed 44, after the bias correction). The legend gives R^2 and RMSE for this period.

Figure 6 shows the distribution of prediction errors for the six models on the test set without bias correction. The errors of the three seeds are combined. The error is the predicted minus the measured value. Errors below −1000 Wh and above 2000 Wh are collected in

the first and the last bar. All distributions peak at zero and have a longer positive tail. All models have a positive mean bias because the asymmetric loss penalises under-prediction. The Enhanced TCN has the smallest mean bias (70.6 Wh). The baselines have mean biases of 92.1 Wh for 1D CNN-GRU, 96.0 Wh for Plain TCN, 97.2 Wh for CNN-GRU, 104.8 Wh for LSTM, and 113.9 Wh for Transformer. After the bias correction, the mean bias of the Enhanced TCN is 9.6 Wh, and the mean biases of the baselines are between 10.3 and 23.0 Wh (see Table 2). The small positive bias suggests the test-set bias is slightly larger than the validation-set bias.

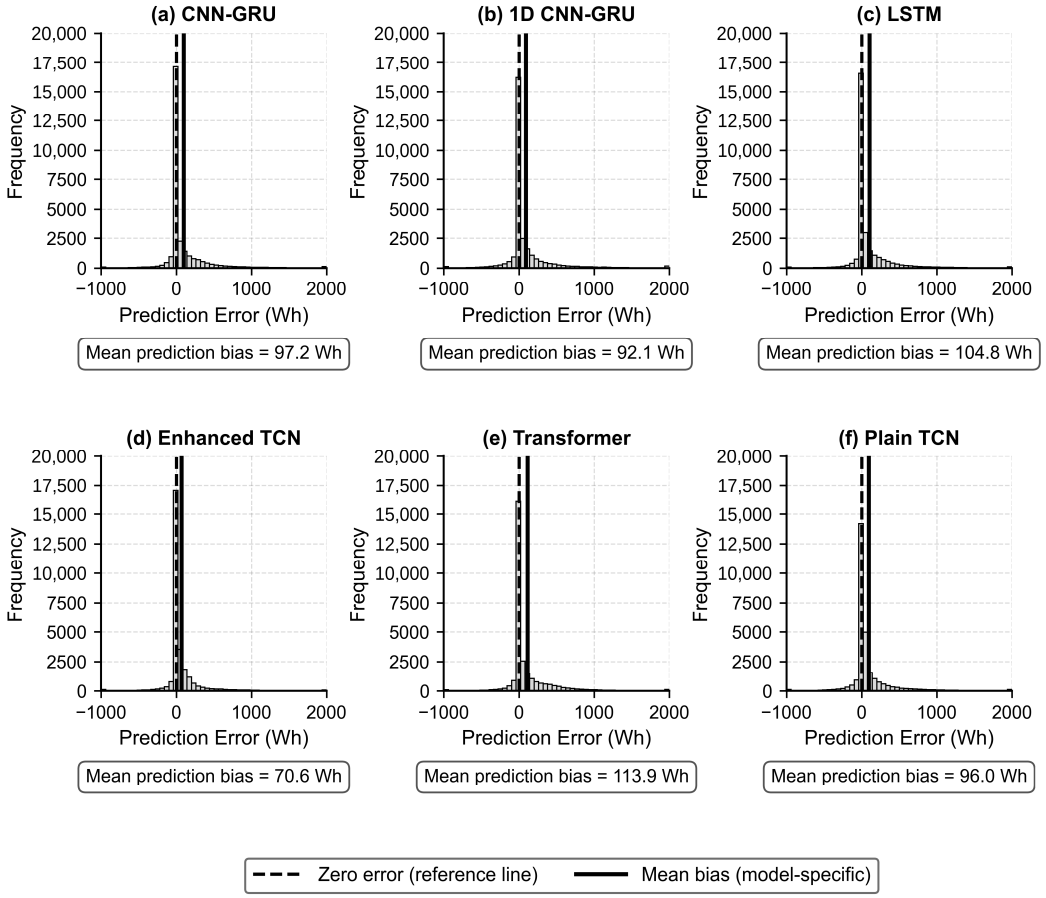


Figure 6. Prediction error distribution of the six trained models on the test set at the 15 min horizon without the bias correction (three seeds combined). Errors outside the axis range are collected in the first and last bars.

Table 4 shows the RMSE of the Enhanced TCN, the two persistence models, the LSTM, and the Plain TCN for different conditions and seasons after bias correction, averaged over three seeds. A day is clear when its measured energy is at least 0.8 of the clear-sky profile, partly cloudy between 0.5 and 0.8, and cloudy below 0.5. A rapid ramp is a change of more than 690 Wh between two time steps, or 15% of the clear-sky profile maximum. The classes with a ramp in the last step or in the last hour use the measured changes before the forecast time. The last ramp class contains the daytime samples of the days with at least three rapid ramps. The sunrise and sunset class contains the two hours after sunrise and the two hours before sunset. At night, all models have errors close to zero, so the differences come from daytime. The Enhanced TCN has the lowest RMSE on clear days (252.1 Wh), on partly cloudy days (485.7 Wh) and in all four seasons. On cloudy days, it is almost equal to smart persistence (356.8 and 354.3 Wh). In the daytime periods without ramps and around sunrise and sunset, smart persistence has the lowest error (185.5 and 94.9 Wh). During rapid ramps, all models have high errors. Here, the deep learning baselines have lower

errors (986.7 Wh for the Plain TCN and 1002.8 Wh for the LSTM) than the Enhanced TCN (1123.6 Wh), and the two persistence models have the highest errors (1298.4 and 1317.1 Wh). Rapid ramps are 4.2% of the test samples. The autumn values are based on six weeks only, because the test set starts on 17 October 2021. A rolling evaluation with retraining for each season was not performed.

Table 4. RMSE (Wh) by condition and season at the 15 min horizon after the bias correction (mean over three seeds).

Condition	Enhanced TCN	Smart Persistence	Persistence	LSTM	Plain TCN
All test samples (100%)	**281.0**	295.3	306.3	302.1	318.1
Daytime (52.1%)	**389.4**	409.3	424.5	418.7	440.9
Night-time (47.9%)	**0.0**	2.5	2.0	3.1	4.5
Clear days (11.1%)	**252.1**	284.6	330.5	283.5	297.3
Partly cloudy days (18.2%)	**485.7**	522.3	536.6	515.4	526.9
Cloudy days (22.7%)	356.8	**354.3**	358.8	386.5	423.4
Rapid ramps (4.2%)	1123.6	1298.4	1317.1	1002.8	**986.7**
No ramp, daytime (47.9%)	232.7	**185.5**	209.7	320.2	354.8
Ramp in the last step (4.2%)	816.4	903.8	908.5	845.5	844.9
Ramp in the last hour (10.2%)	713.9	772.1	782.5	746.9	759.1
Days with at least three ramps (27.8%)	495.8	532.5	545.2	515.7	525.1
Sunrise and sunset (16.7%)	114.4	94.9	129.3	140.1	162.9
Winter, Dec-Feb (25.9%)	146.6	150.2	161.8	165.1	190.1
Spring, Mar-May (30.1%)	341.4	358.3	372.2	354.0	371.2
Summer, Jun-Aug (30.0%)	340.2	359.9	370.4	374.1	388.1
Autumn, Oct-Nov (14.0%)	159.5	165.8	172.6	178.7	191.9

The share of the test samples in each class is given in brackets. Bold values indicate the lowest RMSE in each row.

The rapid-ramp class is selected based on the measured change in the target step. By this definition, the persistence error exceeds 690 Wh in every sample of the class, and the class is unknown at forecast time. It is the most difficult case for a model that starts from the last measured value. In this class, the Enhanced TCN forecasts, on average, 21% of the measured change, compared with 41% for the LSTM and the Plain TCN, explaining its higher error. It is still 13.5% better than smart persistence. Three further classes were therefore examined that use only information known at the forecast time. When a ramp occurred in the last step (4.2% of the samples), the Enhanced TCN has the lowest RMSE of all models (816.4 Wh, compared with 844.9 Wh for the Plain TCN, 845.5 Wh for the LSTM and 903.8 Wh for smart persistence). When a ramp occurred in the last hour (10.2% of the samples), it also had the lowest RMSE (713.9 Wh, compared with 746.9 Wh for the LSTM, 759.1 Wh for the Plain TCN and 772.1 Wh for smart persistence). On days with at least three ramps (141 of 306 test days), its daytime RMSE is 495.8 Wh, compared with 515.7 Wh for the LSTM, 525.1 Wh for the Plain TCN, and 532.5 Wh for smart persistence. The Enhanced TCN is therefore the most accurate model during ramp periods. Its limitation is the single step where a large change starts, where the LSTM and the Plain TCN react more strongly. A loss with a higher weight on ramp samples is left for future work.

Smart persistence is a strong baseline at the 15 min horizon, and the Enhanced TCN is the only trained model that outperforms it. Table 4 shows where the 4.9% gain comes

from. The Enhanced TCN performs better on clear days (11.4%), partly cloudy days (7.0%), and during rapid ramps (13.5%), where errors are large and most costly for grid operation. Smart persistence has a lower MAE (102.5 vs. 108.8 Wh) because it performs better during daytime periods without ramps and around sunrise and sunset, when both models' errors are small. The proposed model therefore reduces the large errors that RMSE measures. The proposed model costs 17.6 min to train and about 10 ms per forecast on a CPU (Table 5). This cost is small for a forecast that is needed every 15 min.

Table 5. Model size, computational cost, and training.

Model	Parameters (Million)	FLOPs (G)	Epochs Run	Time per Epoch (s)	CPU Inference (ms)
Enhanced TCN	5.36	1.57	12.0	87.9	9.9
LSTM	1.66	0.27	14.3	64.1	6.3
Transformer	13.71	4.85	15.0	180.3	16.8
CNN-GRU	10.13	35.05	20.3	833.8	99.9
Plain TCN	13.70	6.45	17.3	135.9	20.0
1D CNN-GRU	1.92	0.15	15.0	29.0	1.2

FLOPs are the multiply-add operations of one forecast. Epochs and time per epoch are means over three seeds. The inference time is for one forecast with batch size 1 on a CPU with four threads.

The effect of the gaps in the timestamps was examined in a sensitivity analysis. The test period contains eight gaps, each 1 to 4 days long. A window is counted as affected when its 192 input steps, the 48 earlier steps used by the rolling features or the target step are not continuous. This is the case for 1920 of the 29,310 test windows (6.55%). Without these windows, the Enhanced TCN's RMSE after bias correction is 287.1 $\pm$ 4.8 Wh, compared with 301.9 Wh for smart persistence, 313.0 Wh for persistence, 308.1 Wh for the LSTM, and 324.0 Wh for the Plain TCN. The reduction against smart persistence is again 4.9%. The model order does not change, and the Diebold–Mariano test remains significant against all models across all three seeds. The values are slightly higher for all models because the gaps fall between October and February, when errors are low. The gaps therefore do not change the conclusions.

4.1. Training Convergence Analysis

Figure 7 shows the training and validation loss of the six models for one run (seed 44). All models were trained with the same settings. The marker shows the epoch with the lowest validation loss, whose weights were used for testing. For the Enhanced TCN in Figure 7d, the validation loss is lowest at epoch 2. After that, the training loss continues to fall while the validation loss rises slowly, and training stops after 12 epochs. Across the three seeds, the best epoch for the Enhanced TCN was 1, 3, and 2. The model starts from the anchor forecast and needs only a few epochs to learn the correction. The learned anchor weights stayed close to their starting values. The anchor weight was 0.95, and the share of smart persistence in the anchor was between 0.50 and 0.52 across all lead times. The LSTM (Figure 7c), the 1D CNN-GRU (Figure 7b), and the Plain TCN (Figure 7f) show the same pattern, with best epochs between 4 and 7. All models start to overfit after a few epochs, so early stopping is needed. One reason is that the data are split in time order, so the validation set covers a different period than the training set. The Transformer (Figure 7e) has a strongly oscillating validation loss. The CNN-GRU (Figure 7a) trains for the most epochs (23), and both of its losses rise after epoch 13.

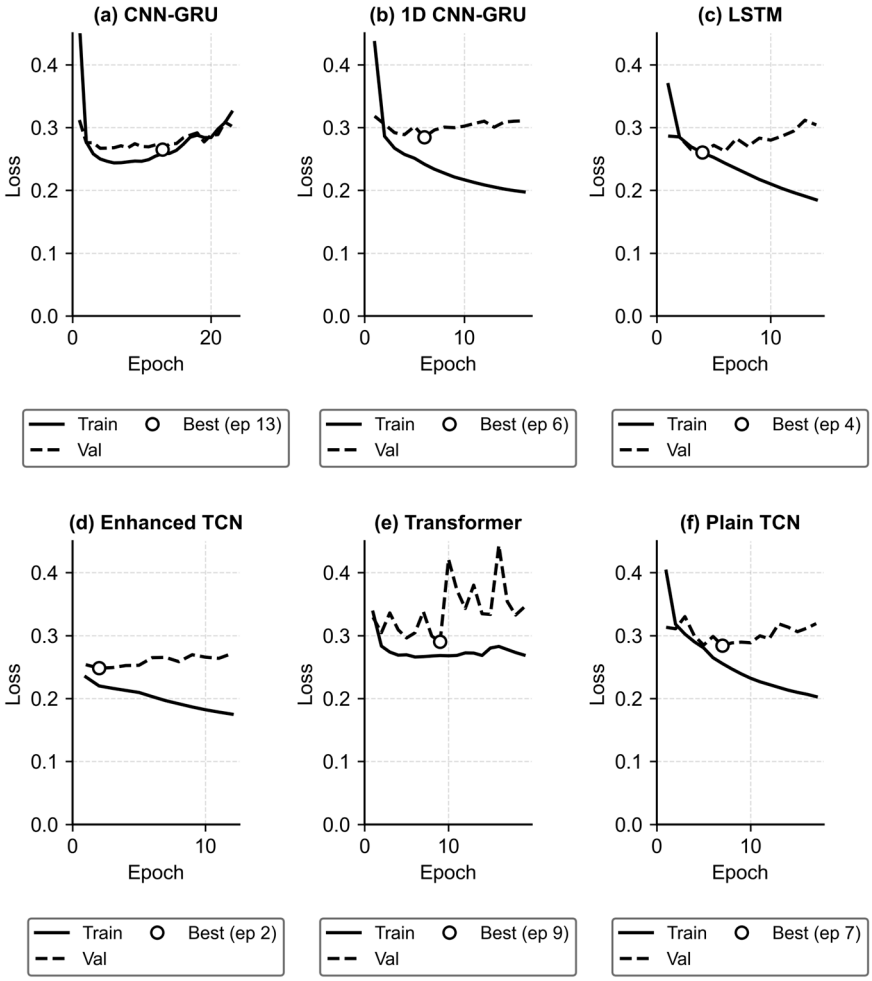


Figure 7. Training and validation loss curves of the six trained models (seed 44). The marker shows the epoch with the lowest validation loss.

4.2. Computational Cost and Feature Importance Analysis

The Enhanced TCN has 5.4 million parameters, compared with 13.7 million for the Plain TCN and the Transformer and 10.1 million for the CNN-GRU. Training the Enhanced TCN on an NVIDIA GeForce RTX 4080 GPU took 17.6 min on average over the three seeds (Figure 8). This was lower than the training time of CNN-GRU (276.3 min), Transformer (44.1 min), and Plain TCN (39.1 min). The LSTM (15.4 min) and the 1D CNN-GRU (7.2 min) were faster. Table 5 gives the size and the cost of all models. One forecast of the Enhanced TCN needs 1.57 G multiply-add operations, compared with 35.0 G for the CNN-GRU, 6.5 G for the Plain TCN and 4.8 G for the Transformer. The inference time for a single forecast is 9.9 ms on a CPU with four threads and 4.1 ms on the GPU. The feature preprocessing for one forecast takes about 32 ms on the CPU. Since a forecast is needed every 15 min, the model can also run without a GPU.

The importance of the input features was measured by permutation. One group of features was shuffled between the test windows, and the change in the RMSE was recorded. This was repeated three times for each group and for each of the three seeds. Figure 9 shows the mean change over the seeds. The changes are small. The largest increases occur for temperature (6.2 Wh), cloud cover (2.8 Wh), and wind speed (1.5 Wh). These three groups increase the RMSE in every seed. For humidity and past PV power, the RMSE decreases when the features are shuffled (-7.8 and -7.2 Wh on average), and this holds for every seed. For the other groups, the sign changes across seeds. These results show that no single group of the 72 window features is essential for the 15 min forecast. The most important

information is the last measured value and the clear-sky profile. It enters the model through the physics-informed decoder and is not part of the groups in Figure 9. The test was therefore repeated for the inputs of the decoder. The RMSE without the bias correction (300.1 Wh) increases by 555.7 Wh when the sunlight flag is shuffled, by 355.1 Wh for the last measured value and by 258.4 Wh for the clear-sky profile. Shuffling the time of day has no effect. When all 72 window features are shuffled together, the increase is 86.1 Wh. The forecast therefore depends mainly on the decoder inputs, which supports the decoder design. This agrees with the ablation study in Section 4.3.

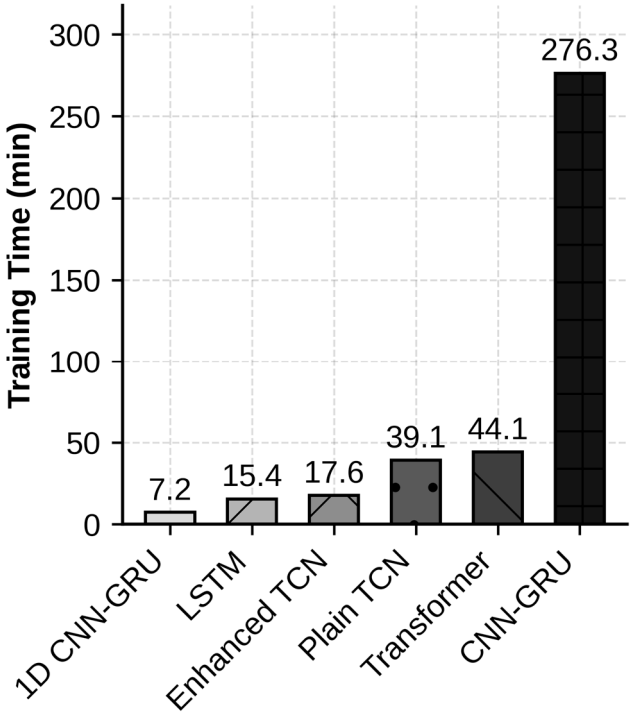


Figure 8. Training time of the six trained models on an NVIDIA GeForce RTX 4080 GPU (mean over three seeds).

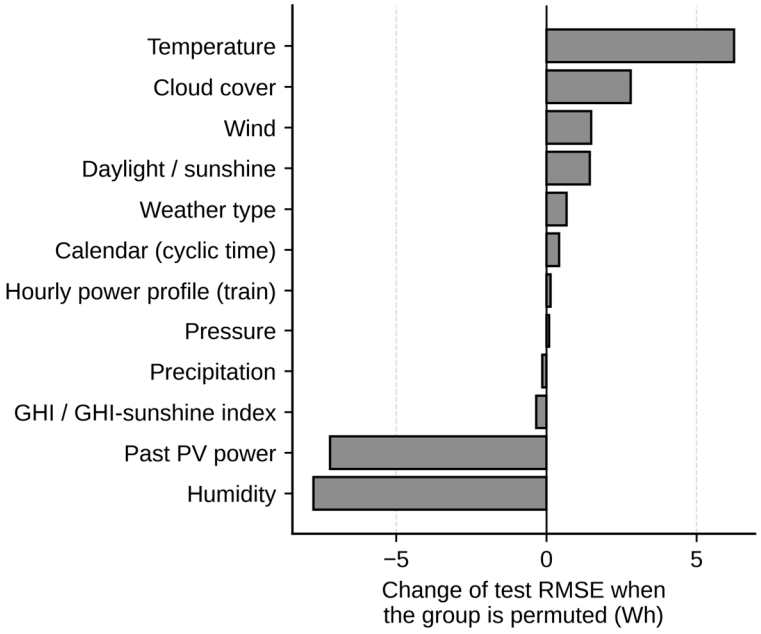


Figure 9. Permutation importance of the input feature groups for the Enhanced TCN (mean over three seeds). A positive value means that the RMSE increases when the group is shuffled.

4.3. Ablation Study

Table 6 and Figure 10 show the ablation study, averaged over three seeds. In each variant, one component of the Enhanced TCN is removed, and the model is trained again with the same settings. Removing the physics-informed decoder increases the RMSE after the bias correction from 281.0 to 295.3 Wh (5.1%). This is the only component with a clear effect, and the Diebold–Mariano test shows a significant difference across all three seeds (see Table 3). Without seasonal decomposition, the RMSE is 282.8 Wh; without feature gating, it is 281.3 Wh. These differences are smaller than the variation between the seeds. Without attention pooling, the mean RMSE is 274.4 Wh, lower than the full model. The attention pooling therefore does not improve the 15 min forecast on this dataset. The Plain TCN, which has none of these components, has an RMSE of 318.1 Wh. The variants have similar training times. All of them stop after 11 to 15 epochs and take about 16 to 20 min.

Table 6. Ablation study at the 15 min horizon (mean ± standard deviation over three random seeds). Each variant removes one component from the full Enhanced TCN. The last column gives the RMSE without the bias correction.

Variant	RMSE (Wh)	MAE (Wh)	R^2	RMSE Without Correction (Wh)
Full Enhanced TCN	281.0 ± 4.6	108.8 ± 2.9	0.928 ± 0.002	300.1 ± 10.4
w/o SD	282.8 ± 6.4	112.6 ± 5.2	0.927 ± 0.003	302.6 ± 12.1
w/o Attention	**274.4 ± 5.6**	106.1 ± 2.9	0.932 ± 0.003	288.6 ± 5.2
w/o Feature Gating	281.3 ± 11.2	110.0 ± 6.8	0.928 ± 0.006	301.6 ± 19.3
w/o PI Decoder	295.3 ± 10.9	117.2 ± 2.9	0.921 ± 0.006	311.7 ± 10.5
Plain TCN	318.1 ± 10.3	135.8 ± 6.9	0.908 ± 0.006	346.5 ± 22.9

Each variant removes one component from the full Enhanced TCN. The bold value indicates the lowest RMSE.

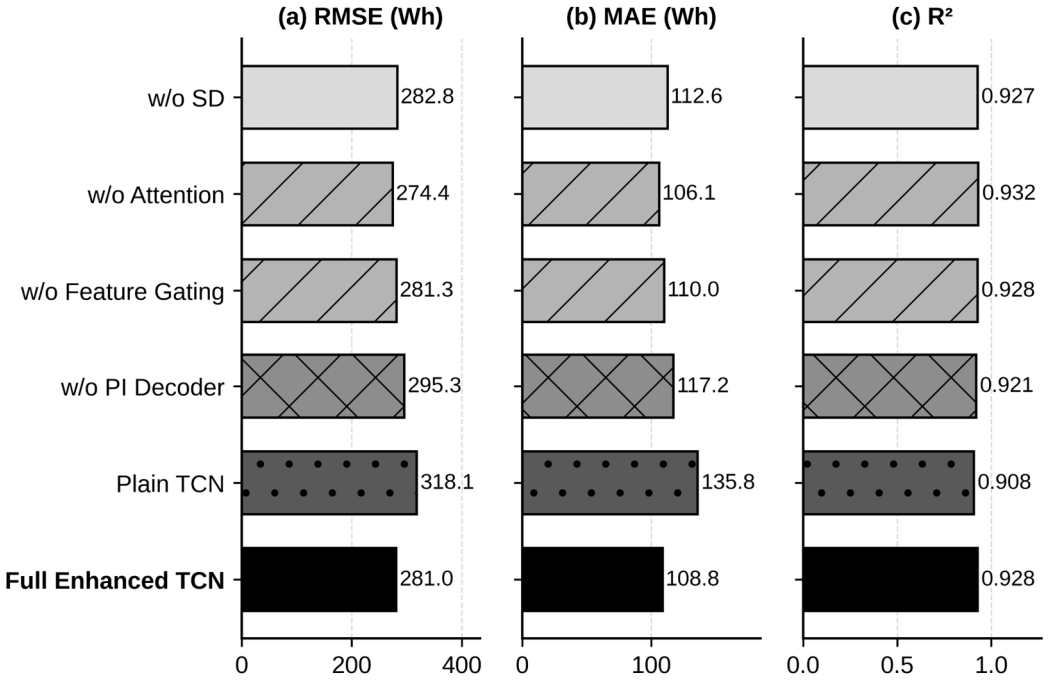


Figure 10. Ablation study results for the 15-minute-ahead forecasting horizon showing RMSE in (**a**), MAE in (**b**), and R^2 in (**c**) for each model variant.

The encoder without the physics-informed decoder has an RMSE of 295.3 Wh. This is 7.2% lower than the Plain TCN (318.1 Wh) and uses less than half the parameters. The decoder further reduces the RMSE by 4.8% and makes the results more stable across seeds. The effects of seasonal decomposition, attention pooling, and feature gating are within seed-to-seed variation at the 15 min horizon. They are kept in the model for longer horizons.

These results show that the proposed model's gain comes mainly from the physics-informed decoder. Seasonal decomposition, attention pooling, and feature gating provide no measurable gain at the 15 min horizon. The full model is reported as the proposed model because it was defined before the ablation study was carried out. Selecting the best variant on the test set would bias the results. A simpler encoder may suffice for this horizon and is left for future work.

5. Conclusions

This paper explains an Enhanced Temporal Convolutional Network for 15-minute-ahead solar power forecasting. The model combines seasonal decomposition, adaptive feature gating, dilated convolutions, and multi-head self-attention to capture patterns at different time scales. A physics-informed decoder builds the forecast by correcting a persistence-based reference forecast. The model was trained using real environmental and weather data collected in Arizona from January 2017 to August 2022. For each prediction, the previous 48 h of data were used as a sliding window of 192 time steps.

Over three random seeds and after the bias correction, the proposed model achieved an RMSE of 281.0 $\pm$ 4.6 Wh, an MAE of 108.8 $\pm$ 2.9 Wh, and an R^2 of 0.928 $\pm$ 0.002. Its RMSE was lower than that of all five deep learning baselines (7.0% to 18.8%), of persistence (8.3%) and of smart persistence (4.9%). The Diebold–Mariano test showed significant differences for all three seeds. Smart persistence had a lower MAE than the proposed model. In ramp periods, the proposed model had the lowest RMSE; at the single step where a large change starts, the LSTM and the Plain TCN had lower errors. Without bias correction, the proposed model still had the lowest RMSE among the deep learning models, but it was not better than smart persistence. The ablation study showed that the physics-informed decoder gives the main gain. Seasonal decomposition, attention pooling, and feature gating had no clear effect at the 15 min horizon. On an NVIDIA RTX 4080 GPU, the model completed training in about 18 min, which makes regular retraining practical. However, the model was evaluated using data from a single photovoltaic site with three random seeds and considered only deterministic point forecasting at the 15 min horizon. The bias correction and clear-sky profile are fitted to this site's data and must be refit for another site. At least one year of data from the new site is needed for the clear-sky profile. Sites with tracking systems, snow, or different panel orientations have different daily profiles, and this paper's results cannot be transferred to them without a new evaluation. The conclusions therefore apply only to this site and the 15 min horizon. Errors are higher on partly cloudy days than on clear days (see Table 4), so higher errors are expected in cloudier climates. Because of limited computing time, a separate hyperparameter search was not run for each model, and the seasonal results come from a single test period without retraining. Future work will use more random seeds, run a hyperparameter search for each model, and perform rolling evaluation with retraining for each season. Future work will examine a simpler encoder, a loss function that does not need a bias correction, probabilistic forecasting, multi-horizon prediction, and transfer learning approaches to improve uncertainty handling and model generalisation across different geographic locations.

Author Contributions: Conceptualisation: M.I.M. and J.K.; methodology: M.I.M.; software, M.I.M.; validation: M.I.M., J.K. and K.P.; formal analysis: M.I.M. and J.K.; investigation: M.I.M., J.K. and K.P.; resources: M.I.M., J.K. and K.P.; data curation: M.I.M. and J.K.; writing—original draft preparation:

M.I.M., J.K. and K.P.; writing—review and editing: M.I.M., J.K. and K.P.; visualisation: M.I.M.; supervision: J.K. and K.P.; project administration: J.K. and K.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data used for this research article is available online at https://www.kaggle.com/datasets/pythonafroz/renewable-power-generation-and-weather-conditions/data (accessed on 14 October 2026). The source code is part of an ongoing research project in which the model is extended to further forecast horizons and validated at other locations. It is therefore not publicly released at this stage. After the project is completed, the code will be made available at https://github.com/imranmunawarch-stack/Solar-Power-Forecasting-. The repository is currently private and will be made public after the project is completed. Until then, it is available from the corresponding author upon reasonable request. All model and training settings needed to reproduce the results are given in Sections 2 and 3.

Acknowledgments: During the preparation of this manuscript, the principal author used ChatGPT (free web version; OpenAI, San Francisco, CA, USA) to correct sentences and to improve the English. The other authors have reviewed and edited the output and take full responsibility for the content of this publication.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A

Table A1. Diebold–Mariano test sensitivity to Newey–West lag after bias correction (15 min horizon). A lag of 0 indicates no autocorrelation correction; 96 steps = 1 day, 672 steps = 1 week. Negative values mean the Enhanced TCN has a smaller error.

Model Compared	Lag	Seed 42	Seed 43	Seed 44
LSTM	0	−7.95 (<0.001)	−5.78 (<0.001)	−10.02 (<0.001)
	14	−4.50 (<0.001)	−3.59 (<0.001)	−5.24 (<0.001)
	96	−4.00 (<0.001)	−3.30 (0.001)	−4.55 (<0.001)
	672	−3.89 (<0.001)	−3.56 (<0.001)	−4.29 (<0.001)
Plain TCN	0	−10.11 (<0.001)	−15.08 (<0.001)	−11.91 (<0.001)
	14	−4.95 (<0.001)	−6.44 (<0.001)	−6.14 (<0.001)
	96	−4.25 (<0.001)	−5.49 (<0.001)	−5.41 (<0.001)
	672	−4.53 (<0.001)	−5.77 (<0.001)	−5.47 (<0.001)
CNN-GRU	0	−13.01 (<0.001)	−10.54 (<0.001)	−13.21 (<0.001)
	14	−6.69 (<0.001)	−4.49 (<0.001)	−7.84 (<0.001)
	96	−5.87 (<0.001)	−3.81 (<0.001)	−6.82 (<0.001)
	672	−5.09 (<0.001)	−3.51 (<0.001)	−6.00 (<0.001)
Transformer	0	−9.44 (<0.001)	−16.95 (<0.001)	−11.57 (<0.001)
	14	−5.87 (<0.001)	−8.95 (<0.001)	−6.80 (<0.001)
	96	−5.23 (<0.001)	−7.38 (<0.001)	−6.08 (<0.001)
	672	−5.55 (<0.001)	−6.37 (<0.001)	−5.62 (<0.001)
1D CNN-GRU	0	−12.85 (<0.001)	−17.05 (<0.001)	−16.33 (<0.001)
	14	−5.13 (<0.001)	−6.48 (<0.001)	−5.67 (<0.001)
	96	−4.10 (<0.001)	−5.23 (<0.001)	−4.32 (<0.001)
	672	−3.90 (<0.001)	−5.00 (<0.001)	−4.10 (<0.001)

Table A1. *Cont.*

Model Compared	Lag	Seed 42	Seed 43	Seed 44
Enhanced TCN (no PI)	0	−10.07 (<0.001)	−3.22 (0.001)	−3.88 (<0.001)
	14	−6.13 (<0.001)	−2.28 (0.022)	−2.86 (0.004)
	96	−5.47 (<0.001)	−1.96 (0.050)	−2.64 (0.008)
	672	−5.21 (<0.001)	−2.10 (0.036)	−2.64 (0.008)
Persistence	0	−12.87 (<0.001)	−7.06 (<0.001)	−10.72 (<0.001)
	14	−11.16 (<0.001)	−5.48 (<0.001)	−8.59 (<0.001)
	96	−10.16 (<0.001)	−4.96 (<0.001)	−7.91 (<0.001)
	672	−7.69 (<0.001)	−4.55 (<0.001)	−6.63 (<0.001)
Smart persistence	0	−7.67 (<0.001)	−3.18 (0.001)	−6.66 (<0.001)
	14	−7.06 (<0.001)	−2.54 (0.011)	−5.50 (<0.001)
	96	−6.58 (<0.001)	−2.32 (0.020)	−5.07 (<0.001)
	672	−5.46 (<0.001)	−2.28 (0.022)	−4.55 (<0.001)

References

1. Lee, C.T.; Hashim, H.; Ho, C.S.; Van Fan, Y.; Klemeš, J.J. Sustaining the low-carbon emission development in Asia and beyond: Sustainable energy, water, transportation and low-carbon emission technology. *J. Clean. Prod.* **2017**, *146*, 1–13. [CrossRef]
2. IRENA. Available online: https://www.irena.org/News/pressreleases/2025/Mar/Record-Breaking-Annual-Growth-in-Renewable-Power-Capacity (accessed on 16 August 2026).
3. Li, G.; Ding, C.; Zhao, N.; Wei, J.; Guo, Y.; Meng, C.; Huang, K.; Zhu, R. Research on a novel photovoltaic power forecasting model based on parallel long and short-term time series network. *Energy* **2024**, *293*, 130621. [CrossRef]
4. Zang, H.; Cheng, L.; Ding, T.; Cheung, K.W.; Liang, Z.; Wei, Z.; Sun, G. Hybrid method for short-term photovoltaic power forecasting based on deep convolutional neural network. *IET Gener. Transm. Distrib.* **2018**, *12*, 4557–4567. [CrossRef]
5. Wang, Z.; Wu, S.; Huang, Y.; Liu, R.; Liu, X. A comprehensive review of deep learning for solar nowcasting: Enhancing accuracy, reliability, and interpretability. *Appl. Energy* **2026**, *407*, 127378. [CrossRef]
6. Raza, M.Q.; Nadarajah, M.; Ekanayake, C. On recent advances in PV output power forecast. *Sol. Energy* **2016**, *136*, 125–144. [CrossRef]
7. Barbieri, F.; Rajakaruna, S.; Ghosh, A. Very short-term photovoltaic power forecasting with cloud modeling: A review. *Renew. Sustain. Energy Rev.* **2017**, *75*, 242–263. [CrossRef]
8. Tawn, R.; Browell, J. A review of very short-term wind and solar power forecasting. *Renew. Sustain. Energy Rev.* **2022**, *153*, 111758. [CrossRef]
9. Yang, D.; Yagli, G.M.; Srinivasan, D. Sub-minute probabilistic solar forecasting for real-time stochastic simulations. *Renew. Sustain. Energy Rev.* **2022**, *153*, 111736. [CrossRef]
10. Liu, M.; Ling, Q. Spatial–temporal multimodal fusion model for intra-hour solar power forecasting under variable weather conditions. *Renew. Energy* **2025**, *248*, 123043. [CrossRef]
11. Nam, N.B.; Ogliari, E.; Leva, S.; Pafumi, E.; Alberti, D.; Duong, M.Q. Comparative analysis of conformal prediction techniques and machine learning models for very short-term solar power forecasting. *Energy AI* **2025**, *21*, 100573. [CrossRef]
12. Abdelsattar, M.; Azim, M.A.; AbdelMoety, A.; Emad-Eldeen, A. Comparative analysis of deep learning architectures in solar power prediction. *Sci. Rep.* **2025**, *15*, 31729. [CrossRef] [PubMed]
13. Iqbal, S.; Shafiullah, M.; Aurangzeb, M.; Jamil, I.; Rehman, A.; Islam, A.; Ali, A.; Alharbi, S.S. Forecasting large-scale solar power plant energy production based on Monte Carlo simulations and long-short-term memory. *Results Eng.* **2025**, *27*, 106269. [CrossRef]
14. Su, Y.; Zhang, W.; Deng, G.; Wang, Z. An Intra-Hour photovoltaic power generation prediction method for flexible building energy systems and its application in operation scheduling strategy. *Sol. Energy* **2024**, *284*, 113031. [CrossRef]
15. Radzi, P.N.L.M.; Mekhilef, S.; Shah, N.M.; Akhter, M.N.; Seyedmahmoudian, M.; Almalaq, Y.; Wahyudie, A. Enhancing short-term solar power forecasting using a hybrid deep learning approach and a comparative analysis of machine learning models. *Energy Convers. Manag. X* **2025**, *29*, 101431. [CrossRef]
16. Gaboitaolelwe, J.; Zungeru, A.M.; Yahya, A.; Lebekwe, C.K.; Vinod, D.N.; Salau, A.O. Machine learning based solar photovoltaic power forecasting: A review and comparison. *IEEE Access* **2023**, *11*, 40820–40845. [CrossRef]

17. Mollasalehi, A.; Farhadi, A. Solar and wind power forecasting: A comparative review of LSTM, random forest, and XGBoost models. *arXiv* **2025**, arXiv:2509.24059. [CrossRef]
18. Kiasari, M.; Aly, H.H. Evaluating Solar Power Forecasting Robustness: A Comparative Analysis of XGBoost, RNN, KNN, RF, and LSTM with emphasis on Lagged Steps, Sensitivity, and Cross-Validation Techniques. In Proceedings of the 2024 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), Kingston, ON, Canada, 6–9 August 2024; pp. 686–692.
19. Wang, J.; Liu, J.; Zhang, S.; Ma, X. Ultra-short-term solar power forecasting by deep learning and data reconstruction. *arXiv* **2025**, arXiv:2509.17095.
20. Renold, A.P.; Sinha, N.; Gao, X.-Z. Hybrid machine learning approach for improved short-term PV power forecasting accuracy. *Results Eng.* **2025**, *28*, 107374. [CrossRef]
21. Luo, X.; Zhang, D. An adaptive deep learning framework for day-ahead forecasting of photovoltaic power generation. *Sustain. Energy Technol. Assess.* **2022**, *52*, 102326. [CrossRef]
22. Oprea, S.-V.; Bâra, A. Ultra-short-term forecasting for photovoltaic power plants and real-time key performance indicators analysis with big data solutions. Two case studies-PV Agigea and PV Giurgiu located in Romania. *Comput. Ind.* **2020**, *120*, 103230. [CrossRef]
23. Ren, X.; Zhang, F.; Yan, J.; Liu, Y. A novel convolutional neural net architecture based on incorporating meteorological variable inputs into ultra-short-term photovoltaic power forecasting. *Sustainability* **2024**, *16*, 2786. [CrossRef]
24. Elmousaid, R.; Drioui, N.; Elgouri, R.; Agueny, H.; Adnani, Y. Ultra-short-term global horizontal irradiance forecasting based on a novel and hybrid GRU-TCN model. *Results Eng.* **2024**, *23*, 102817. [CrossRef]
25. Wang, T.; Gong, Z.; Wang, Z.; Liu, Y.; Ma, Y.; Wang, F.; Li, J. Research and optimization of ultra-short-term photovoltaic power prediction model based on symmetric parallel TCN-TST-BiGRU architecture. *Symmetry* **2025**, *17*, 1855. [CrossRef]
26. Li, Y.; Song, L.; Zhang, S.; Kraus, L.; Adcox, T.; Willardson, R.; Komandur, A.; Lu, N. A TCN-based hybrid forecasting framework for hours-ahead utility-scale PV forecasting. *IEEE Trans. Smart Grid* **2023**, *14*, 4073–4085. [CrossRef]
27. Zhu, H.; Wang, Y.; Wu, J.; Zhang, X. A regional distributed photovoltaic power generation forecasting method based on grid division and TCN-Bilstm. *Renew. Energy* **2026**, *256*, 123935. [CrossRef]
28. Sharma, J.; Soni, S.; Paliwal, P.; Saboor, S.; Chaurasiya, P.K.; Sharifpur, M.; Khalilpoor, N.; Afzal, A. A novel long term solar photovoltaic power forecasting approach using LSTM with Nadam optimizer: A case study of India. *Energy Sci. Eng.* **2022**, *10*, 2909–2929. [CrossRef]
29. Alcañiz, A.; Grzebyk, D.; Ziar, H.; Isabella, O. Trends and gaps in photovoltaic power forecasting with machine learning. *Energy Rep.* **2023**, *9*, 447–471. [CrossRef]
30. Tsai, W.-C.; Tu, C.-S.; Hong, C.-M.; Lin, W.-M. A review of state-of-the-art and short-term forecasting models for solar PV power generation. *Energies* **2023**, *16*, 5436. [CrossRef]
31. Afroz. Available online: https://www.kaggle.com/datasets/pythonafroz/renewable-power-generation-and-weather-conditions/data (accessed on 14 October 2026).
32. Mazen, F.M.A.; Shaker, Y.; Abul Seoud, R.A. Forecasting of solar power using GRU–temporal fusion transformer model and DILATE loss function. *Energies* **2023**, *16*, 8105. [CrossRef]
33. Suresh, V.; Janik, P.; Rezmer, J.; Leonowicz, Z. Forecasting solar PV output using convolutional neural networks with a sliding window algorithm. *Energies* **2020**, *13*, 723. [CrossRef]
34. Vu, B.H.; Chung, I.-Y. Optimal generation scheduling and operating reserve management for PV generation using RNN-based forecasting models for stand-alone microgrids. *Renew. Energy* **2022**, *195*, 1137–1154. [CrossRef]
35. Afrasiabi, M.; Mohammadi, M.; Rastegar, M.; Afrasiabi, S. Deep learning architecture for direct probability density prediction of small-scale solar generation. *IET Gener. Transm. Distrib.* **2020**, *14*, 2017–2025. [CrossRef]
36. Li, Q.; Zhang, X.; Ma, T.; Liu, D.; Wang, H.; Hu, W. A Multi-step ahead photovoltaic power forecasting model based on TimeGAN, Soft DTW-based K-medoids clustering, and a CNN-GRU hybrid neural network. *Energy Rep.* **2022**, *8*, 10346–10362. [CrossRef]
37. Zhang, Y.; Ma, J.; Zeng, C.; Li, G. Short-term global horizontal irradiance forecasting using a hybrid convolutional neural network-gate recurrent unit method. *J. Phys. Conf. Ser.* **2021**, *2025*, 012001. [CrossRef]
38. Sabri, N.M.; El Hassouni, M. Accurate photovoltaic power prediction models based on deep convolutional neural networks and gated recurrent units. *Energy Sources Part A Recovery Util. Environ. Eff.* **2022**, *44*, 6303–6320. [CrossRef]
39. Babalhavaeji, A.; Radmanesh, M.; Jalili, M.; Gonzalez, S.A. Photovoltaic generation forecasting using convolutional and recurrent neural networks. *Energy Rep.* **2023**, *9*, 119–123. [CrossRef]
40. Bai, S.; Kolter, J.Z.; Koltun, V. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv* **2018**, arXiv:1803.01271. [CrossRef]
41. Yang, Y.; Liu, Y.; Zhang, Y.; Shu, S.; Zheng, J. DEST-GNN: A double-explored spatio-temporal graph neural network for multi-site intra-hour PV power forecasting. *Appl. Energy* **2025**, *378*, 124744. [CrossRef]
42. Zhang, Q.; Zhu, Y.; Cordeiro, F.R.; Chen, Q. PSSCL: A progressive sample selection framework with contrastive loss designed for noisy labels. *Pattern Recognit.* **2025**, *161*, 111284. [CrossRef]

43. Limouni, T.; Yaagoubi, R.; Bouziane, K.; Guissi, K.; Baali, E.H. Accurate one step and multistep forecasting of very short-term PV power using LSTM-TCN model. *Renew. Energy* **2023**, *205*, 1010–1024. [CrossRef]
44. Lea, C.; Vidal, R.; Reiter, A.; Hager, G.D. Temporal convolutional networks: A unified approach to action segmentation. In *Proceedings of the European Conference on Computer Vision*; Springer: Cham, Switzerland, 2016; pp. 47–54.
45. Diebold, F.X.; Mariano, R.S. Comparing predictive accuracy. *J. Bus. Econ. Stat.* **2002**, *20*, 134–144. [CrossRef]
46. Pedro, H.T.; Coimbra, C.F. Assessment of forecasting techniques for solar power production with no exogenous inputs. *Sol. Energy* **2012**, *86*, 2017–2028. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.