energies

MDPI

Article

Hardware-Centric Exploration of the Discrete Design Space in Transformer–LSTM Models for Wind Speed Prediction on Memory-Constrained Devices

Laeeq Aslam [1], Runmin Zou [1,*], Ebrahim Shahzad Awan [2], Sayyed Shahid Hussain [1], Kashish Ara Shakil [3,*], Mudasir Ahmad Wani [4] and Muhammad Asim [4,5]

[1] School of Automation, Central South University, Changsha 410083, China; 204608004@csu.edu.cn (L.A.); shahid@csu.edu.cn (S.S.H.)
[2] School of Engineering, Design and Built Environment, Western Sydney University, Penrith, NSW 2747, Australia; ebrahimawan@gmail.com
[3] Department of Computer Sciences, College of Computer and Information Sciences, Princess Nourah Bint Abdulrahman University, P.O. Box 84428, Riyadh 11671, Saudi Arabia
[4] EIAS Data Science Lab, College of Computer and Information Sciences, Prince Sultan University, Riyadh 11586, Saudi Arabia; mwani@psu.edu.sa (M.A.W.); masim@psu.edu.sa (M.A.)
[5] School of Electronic Information, Central South University, Changsha 410083, China
* Correspondence: rmzou@csu.edu.cn (R.Z.); kashakil@pnu.edu.sa (K.A.S.)

Abstract: Wind is one of the most important resources in the renewable energy basket. However, there are questions regarding wind as a sustainable solution, especially concerning its upfront costs, visual impact, noise pollution, and bird collisions. These challenges arise in commercial windmills, whereas for domestic small-scale windmills, these challenges are limited. On the other hand, accurate wind speed prediction (WSP) is crucial for optimizing power management in renewable energy systems. Existing research focuses on proposing model architectures and optimizing hyperparameters to improve model performance. This approach often results in larger models, which are hosted on cloud servers. Such models face challenges, including bandwidth utilization leading to data delays, increased costs, security risks, concerns about data privacy, and the necessity of continuous internet connectivity. Such resources are not available for domestic windmills. To overcome these obstacles, this work proposes a transformer model integrated with Long Short-Term Memory (LSTM) units, optimized for memory-constrained devices (MCDs). A contribution of this research is the development of a novel cost function that balances the reduction of mean squared error with the constraints of model size. This approach enables model deployment on low-power devices, avoiding the challenges of cloud-based deployment. The model, with its tuned hyperparameters, outperforms recent methodologies in terms of mean squared error, mean absolute error, model size, and R-squared scores across three different datasets. This advancement paves the way for more dynamic and secure on-device wind speed prediction (WSP) applications, representing a step forward in renewable energy management.

Keywords: wind speed prediction; power forecasting; hyperparameter tuning; model size optimization; renewable energy management; on-device deployment

Academic Editors: Jao-Hwa Kuang, Bing-Jean Lee, Ming-Hung Hsu, Thin-Lin Horng and Chia-Cheng Chao

Received: 25 February 2025
Revised: 4 April 2025
Accepted: 16 April 2025
Published: 23 April 2025

Citation: Aslam, L.; Zou, R.; Awan, E.S.; Hussain, S.S.; Shakil, K.A.; Wani, M.A.; Asim, M. Hardware-Centric Exploration of the Discrete Design Space in Transformer–LSTM Models for Wind Speed Prediction on Memory-Constrained Devices. *Energies* **2025**, *18*, 2153. https://doi.org/10.3390/en18092153

1. Introduction

Wind energy is an important alternative to fossil fuels and contributes significantly to the global renewable energy capacity. The Global Wind Report 2022 [1] noted a 94 GW

increase in wind energy in 2021, although growth slowed in major markets like China and the United States. Offshore wind saw significant expansion, particularly in China. However, one of the significant challenges in wind energy management is intermittency, which refers to the unpredictable nature of wind. Wind speeds, influenced by weather patterns, fluctuate throughout the day, resulting in inconsistent power output. However, this problem is usually managed by predicting wind speed in advance and then managing the integration of other available resources with the wind energy system for sustainable power generation. Various prediction techniques have been developed, which can be categorized into physical models, statistical models, and machine learning (ML) models.

Numerical Weather Prediction (NWP) models use physical principles for wind speed prediction (WSP), eliminating the need for historical data and model training. These models simulate atmospheric conditions to provide location-specific forecasts, as demonstrated by Brabec et al. [2]. However, their performance is sensitive to initial physical conditions, where minor errors can affect results. Another study by Moreno et al. [3] pointed out the complexity and high computational costs associated with NWP models. Additionally, accurate input data and substantial computational resources are required, since inaccuracies in initial atmospheric data can lead to forecast errors. Nevertheless, when integrated into hybrid models, NWP models become valuable tools for wind speed prediction, particularly in regions lacking historical wind data [2,4].

On the other hand, statistical models treat WSP as a stochastic process, using historical data to identify time-variable relationships. Common statistical models include autoregressive (AR), moving average (MA), autoregressive integrated moving average (ARIMA), and the Kalman filter [5]. These models have been applied extensively due to their simplicity and relatively low computational cost. However, they assume linear relationships, which may not accurately capture the nonlinear characteristics of wind speed (WS) time series. Torres et al. [6] applied an ARMA model to predict the hourly mean WS in Spain, demonstrating that although such models provide reasonable short-term forecasts, they struggle with the nonlinear and random nature of wind speeds. Li et al. [7] utilized ARMA-based approaches for WSP, showing that these models can be effective under certain conditions but have limitations in capturing complex wind patterns. Collectively, these statistical models contribute to understanding WS patterns by providing a foundational approach to time-series analysis. However, their limitations in capturing nonlinearity and randomness highlight the need for more advanced methods that can model the complex behavior of wind speeds.

Machine learning (ML) models, including artificial neural networks (ANNs), support vector machines (SVMs), and deep learning methods, have been applied in wind speed prediction (WSP) for their ability to model nonlinear relationships [8]. Unlike statistical models, ML models do not assume the normality of the residuals or the stationarity of the time series. ANNs model wind speed as a nonlinear system, as demonstrated by Bechrakis [9] and Mohandes et al. [10], who showed the potential of SVMs for WSP. These models capture patterns in wind speed data but require large datasets and significant computational power for training. Deep learning models, such as LSTM networks, capture long-term dependencies in wind speed data. Combining these approaches can improve prediction accuracy. Hybrid models (HM) integrate various techniques to maximize their benefits. Geng et al. [11] used LSTM networks for short-term WSP, achieving higher accuracy than traditional methods. Cai et al. [12] applied Extreme Gradient Boosting (XGBoost) for WSP, effectively handling large datasets and using regularization to prevent overfitting. Aslam et al. [13] proposed a physics-informed machine learning approach with a novel cost function to enhance WSP by collecting features from surrounding locations to predict future wind speeds. ANNs and SVMs model nonlinear relationships, while

LSTM and XGBoost handle sequential data and large datasets, respectively. Together, these techniques improve prediction accuracy and model complex wind speed behaviors.

Hybrid models combine multiple approaches to leverage their strengths and address individual limitations. HMs often combine statistical models with ML techniques, such as ARIMA with ANNs or SVMs, to improve prediction quality [14]. Recent studies have enhanced time-series data pre-processing using Singular Spectrum Analysis (SSA) and wavelet transforms to increase model accuracy [15]. Advances in HMs include transformer models, initially developed for natural language processing, applied to WSP. Wan et al. [16] and Pan et al. [17] integrated convolutional neural networks (CNNs) with LSTM networks to capture spatial and temporal dependencies in wind speed data. An attention mechanism was added to these models to focus on relevant parts of the input sequence, improving forecasting accuracy [18]. By combining different methodologies, HMs provide a balanced approach that enhances the accuracy and reliability of WSP, effectively managing the complexities and nonlinearity in wind speed data.

Since intermittency is just one challenge related to wind energy utilization, some of the other fundamental challenges in wind power production include visual impact [19–21], noise pollution [22,23], bird and bat collisions [24–26], upfront costs [27–29], and grid integration [30,31]. These issues primarily arise in commercial wind energy projects and raise questions about the sustainability of wind energy as a solution. Although these challenges are significant for commercial wind energy projects, increasing domestic small-scale windmill installations can reduce the impact of these problems.

In existing studies such as [32–34], the challenge of intermittency and prediction is mostly studied in the context of commercial windmills, which can afford large cloud servers to run sophisticated ML models for accurate predictions. In these projects, the focus is on improving prediction accuracy, and the size of the models is not a significant concern. For small windmills, however, accessing a server to run such models presents issues like latency, data privacy, security, bandwidth usage, cloud server costs, and energy consumption. Therefore, designing smaller models that predict WS with minimal error could avoid these challenges and efficiently manage energy locally. Deploying ML models on edge devices is a potential solution, but this area has not been well studied in the context of domestic wind energy production. Addressing these challenges for small windmills, this research investigates a discrete design space (DDS) comprising the hyperparameters of a transformer–LSTM hybrid model in such a way that not only is the prediction accuracy of such models improved but also the size of the models is significantly reduced.

The primary contributions of this research are summarized as follows:

- The proposal of a hybrid baseline model (HBM) that combines a transformer model with LSTM, designed for optimal tuning with variable hyperparameters.
- The introduction of a novel cost function and use of genetic algorithms for optimizing discrete design spaces (DDS) influenced by model hyperparameters.
- The incorporation of hardware-specific performance evaluations at each optimization step, ensuring effective deployment on targeted hardware.

This research enhances the understanding and application of transformer–LSTM models for wind speed prediction on low-power devices. It investigates the trade-offs between model size (MS) and prediction accuracy, providing solutions to challenges associated with memory constraints and performance optimization.

2. Proposed Methodology

This study utilizes a baseline model with two main components: a transformer encoder and LSTM layers. The transformer encoder extracts features by handling complex data patterns, while the LSTM layers, as the prediction head, capture temporal dependencies.

This section first details the transformer–LSTM baseline architecture and then discusses model performance metrics, which are integral to the cost and fitness functions. Finally, a hardware-centric DDS optimization scheme is presented.

2.1. Baseline Architecture

The baseline model's architecture is shown in Figure 1. First, the input features pass through a dense embedding layer. This layer changes the dimensions of the input data using Equation (1):

$$\text{Dimension} = N_H \times H_S \tag{1}$$

Here, N_H is the number of heads and H_S is the head size.

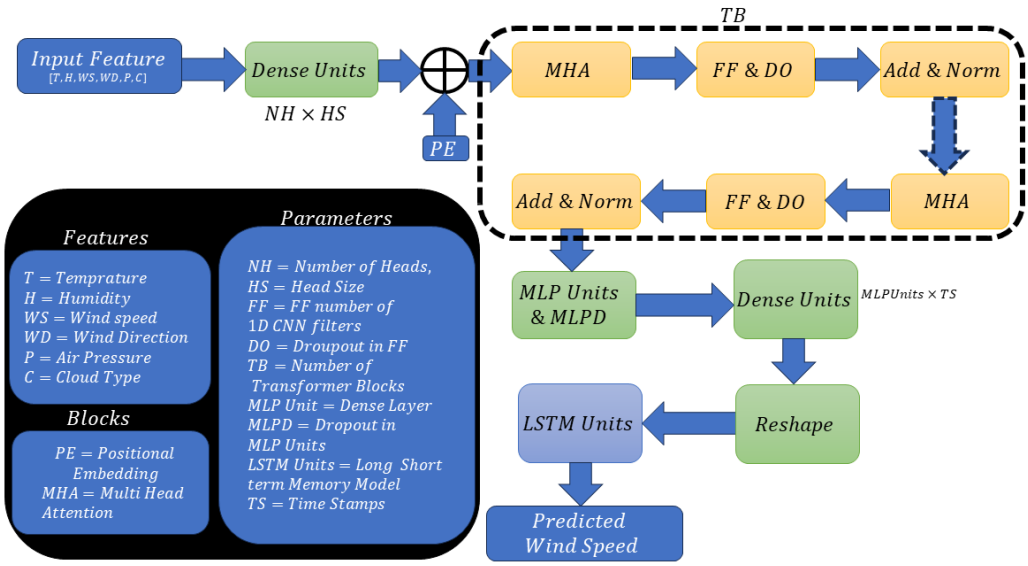

Figure 1. Block diagram of the proposed baseline model.

Next, positional embeddings are added using sinusoidal functions:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right) \tag{2}$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right) \tag{3}$$

where pos is the position in the sequence, i is the dimension index, and d_{model} represents the dimensions. Then, the features pass through the multi-head attention mechanism:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{4}$$

where Q, K, and V are the query, key, and value matrices, and d_k is the dimension of the keys. This computes attention scores, focusing on key parts of the input data. Next, the features pass through a 1D convolutional neural network (CNN) layer that applies a ReLU activation function, adding nonlinearity to help the model learn complex patterns. Following the CNN layer, an add-and-normalize layer is used:

$$\text{Layer Output} = \text{LayerNorm}(\text{Input} + \text{Output}) \tag{5}$$

This layer normalizes the data, stabilizing the learning process. The model then concatenates T_B transformer blocks, each with a multi-head attention mechanism, a CNN layer, a feed-forward network, and an add-and-normalize layer. Next, a layer of MLP dense

units is added. These dense layers enhance the representation. The features then pass through more dense layers, adjusting dimensions to match the number of time stamps. This is followed by a reshape layer, which organizes features by time stamps. The final dense layer allocates features to specific time stamps for better prediction accuracy. Finally, a layer of LSTM units acts as a regression head to predict future events, which is suitable for tasks like forecasting wind speeds. The LSTM uses several gates to manage information flow:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{6}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{7}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{8}$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{9}$$

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \tag{10}$$

$$h_t = o_t \cdot \tanh(C_t) \tag{11}$$

Equations (6)–(11) show how the LSTM manages information. The forget gate (f_t) discards irrelevant data, the input gate (i_t) updates the cell state with new information, and the output gate (o_t) outputs part of the cell state. These mechanisms help the LSTM make accurate predictions based on historical data.

This baseline model is better than a purely LSTM or transformer model because it leverages the strengths of both architectures. The transformer's attention mechanism excels at capturing contextual relationships in the data, while the LSTM is adept at handling long-term dependencies in sequential data. This combination allows the model to effectively address both short-term and long-term patterns, leading to more accurate predictions.

2.2. Model Performance Metrics

The performance of the model is quantitatively assessed using various metrics, such as *MSE*, *MAE*, R^2 score, the model's storage size on disk and its latency. The formulas for these metrics are as follows:

Coefficient of Determination (R^2 Score):

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{12}$$

Mean Squared Error (*MSE*):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{13}$$

Mean Absolute Error (*MAE*):

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{14}$$

In the above equations, y_i denotes the actual wind speed, $\hat{y}_i$ denotes the predicted WS, and $\bar{y}$ denotes the average of the actual wind speed. *MSE* is sensitive to larger errors due to its squaring of error terms. In contrast, *MAE* is sensitive to the median of errors as it considers the absolute value of errors.

2.3. Cost and Fitness Functions

The proposed baseline model has the ability to capture time-series data patterns. However, such models have a high MS if not properly optimized. Hence, there is a trade-off between prediction accuracy, quantified by the *MSE* and MS. Navigating this balance can be seen as the optimization of a DDS. Unlike in a continuous space, where we have infinite options for selecting parameter values, choices in a DDS are quantized, offering only a finite set of possibilities. The pivotal components of our model, all within this DDS, comprise the following:

- Head Size (HS): Influences the granularity of attention and overall model capacity.
- Number of Heads (NH): Enables concurrent attention mechanisms, facilitating the model's understanding of diverse data facets.
- Feed-Forward Dimension (FF): Signifies the inner processing capability of the feed-forward network within the transformer.
- Number of Transformer Blocks (TB): Multiple blocks augment the model's depth, enhancing its ability to discern complex patterns.
- MLP Units ($MLPU$): Governs the capacity of the dense layers within the model.
- Dropout (DO): Acts as a regularization knob in the transformer encoder section.
- MLP Dropout ($MLPD$): Acts as a regularization knob in the dense layer section, mitigating the risk of overfitting.
- LSTM Units ($LSTMU$): Determines the temporal processing power of the LSTM layers.

Considering these parameters, the optimization problem is given by Equation (15).

$$\min_{A} C \quad \text{such that} \quad AX - C = 0 \tag{15}$$

where

$$X = \begin{bmatrix} HS & NH & FF & TB & MLPU & DO & MLPD & LSTMU \end{bmatrix}^T \tag{16}$$

represents the known configurations of the model, consolidated into a column vector. Our objective is to identify the optimal coefficient matrix A to minimize the cost C. The cost metric C symbolizes the equilibrium between accuracy and MS, as defined in Equation (17):

$$C = \epsilon \times \left(\frac{MSE}{\alpha} \right) + (1 - \epsilon) \times \left(\frac{\text{Size of Model}}{\beta} \right) \tag{17}$$

where ϵ varies between 0 and 1, serving as a balancing agent between the model's predictive prowess (MSE scaled by α in m/s) and its computational footprint (size of model scaled by β in bytes).

Each column of A is subject to specific upper threshold constraints and has different data type requirements: $0 \leq A_1 \leq$ mhs, $0 \leq A_2 \leq$ mnh, $0 \leq A_3 \leq$ mffd, $0 \leq A_4 \leq$ mnb, $0 \leq A_5 \leq$ mmlpu, $0 \leq A_6 \leq$ mdo, $0 \leq A_7 \leq$ mmlpdo, $0 \leq A_8 \leq$ mlstmu. Here, the upper limits (mhs, mnh, mffd, etc.) represent the maximum permissible value of each parameter (Max_Head_Size, Max_Number_of_Heads, etc.), ensuring the model's feasibility and efficiency within the specified constraints.

In this work, a genetic algorithm (GA)-inspired technique is employed to identify models that minimize prediction error while also maintaining a minimal size. Consequently, we have devised a fitness function that incorporates an additional constraint to ensure model effectiveness. This constraint ensures that the model achieves a positive R^2 score, which is a standard metric for assessing the goodness of fit in regression models. The fitness function, therefore, integrates this additional criterion and is formulated as shown in Equation (18):

$$\text{Fitness} = C - \lambda \times R^2 \text{ score} \tag{18}$$

This method ensures that the chosen model balances accuracy and size while meeting a basic standard of predictive performance, as measured by the R^2 score.

2.4. Hardware-Centric DDS Optimization Scheme

In this section, the proposed hardware-centric DDS optimization algorithm is discussed. The proposed methodology is shown graphically in Figure 2. This scheme is based on the optimization principles of the genetic algorithm. The details of the proposed scheme are as follows:

1. A genetic algorithm initiates a random generation of models, where each gene represents a specific aspect of the model and one chromosome is represented as X in Equation (16). Once these models are generated, they are passed through a constraint check.
2. These models are then built and trained for a limited duration of 25 epochs. After training, they are deployed on a memory-constrained device to evaluate their performance along with the test data.
3. The fitness of these models is calculated based on the test results, using Equation (18), where λ is a Lagrange multiplier that penalizes models with a negative R^2 score after 25 epochs.
4. The GA process for generating new models involves selection, mutation, and crossover by considering the following criteria:
    - If there is an improvement in performance over the last three generations, the two best models (minimum fitness values) are selected for mutation and crossover.
    - If performance does not improve, the best parent and a second randomly selected gene are used for mutation and crossover.
    - During mutation or crossover, if any gene exceeds its predefined limits, a random value within the permissible range is reassigned to maintain the MS within the required memory space.
5. The process returns to step 2 unless a stopping criterion is met.
6. Finally, the top five models from the experiment are selected for further training over an extended number of epochs. They are then re-evaluated on the memory-constrained device to ascertain their final performance metrics.

In this work, models are trained on a server and tested on MCDs like the Jetson. This dual-environment and hardware-centric approach is more practical compared to the standard procedure of a neural architecture search, where models are trained and tested simultaneously on the same machine, typically a server. In our research, models are trained on servers to accelerate the training process and then tested on the actual hardware. Since results may vary when models are deployed on such devices, testing them before selecting the best parent ensures the selection of solutions that are optimal for the specific hardware. To transfer the trained models from the server to the Jetson device, Google's gRPC protocol is employed, ensuring efficient and reliable model transfer.

Training lasts for 25 epochs to quickly assess model behavior and adaptability. A model's performance during this period, as indicated by the *MSE* and R^2 scores, determines its potential for further development or the need for reevaluation. Predefined limits on hyperparameters prevent training failures in memory-constrained environments, ensuring that models remain within available memory space and are practical for real-world deployment.

Furthermore, an adaptive genetic algorithm strategy introduces new random genes when performance stagnates over three generations. This keeps the search space diverse and dynamic, promoting exploration and increasing the chance of finding effective models.

Hence, the top five models are selected based on initial performance metrics for extended training beyond 25 epochs. This approach ensures that promising candidates are refined further. The computational complexity of the proposed algorithm is analyzed in Appendix A. To facilitate a better understanding of the methodology, the pseudocode in Algorithm 1 succinctly outlines the scheme.

Algorithm 1 Hardware-centric DDS optimization algorithm

Require: N (population size), E (maximum epochs), S (stopping criteria)
Ensure: Optimized models M^*

$P \leftarrow \text{Initialize}(N)$ ▷ Generate initial population
$G \leftarrow 0$ ▷ Generation counter
while $\neg\text{StoppingCriteria}(G, S)$ **do**
  for all $m \in P$ **do**
    $\text{Train}(m, E)$
    $\text{Eval}(m)$
    $m.\varphi \leftarrow \epsilon \times MSE(m) + (1 - \epsilon) \times \text{Size}(m) - \lambda \times R^2(m)$
  end for
  $P' \leftarrow \emptyset$ ▷ New population
  if $\text{Improved}(P, 3)$ **then**
    $\{p_1, p_2\} \leftarrow \text{SelectBest}(P, 2)$
  else
    $\{p_1, p_2\} \leftarrow \{\text{SelectBest}(P, 1), \text{SelectRandom}(P)\}$
  end if
  for $i \leftarrow 1$ **to** $N/2$ **do**
    $\{c_1, c_2\} \leftarrow \text{Crossover}(p_1, p_2)$
    $\text{Mutate}(c_1)$
    $\text{Mutate}(c_2)$
    $\text{Constraint}(c_1)$
    $\text{Constraint}(c_2)$
    $P' \leftarrow P' \cup \{c_1, c_2\}$
  end for
  $P \leftarrow P'$
  $G \leftarrow G + 1$
end while
$M^* \leftarrow \text{SelectTop}(P, 5)$
$\text{ExtendedTraining}(M^*)$
$\text{ReEval}(M^*)$
return M^*

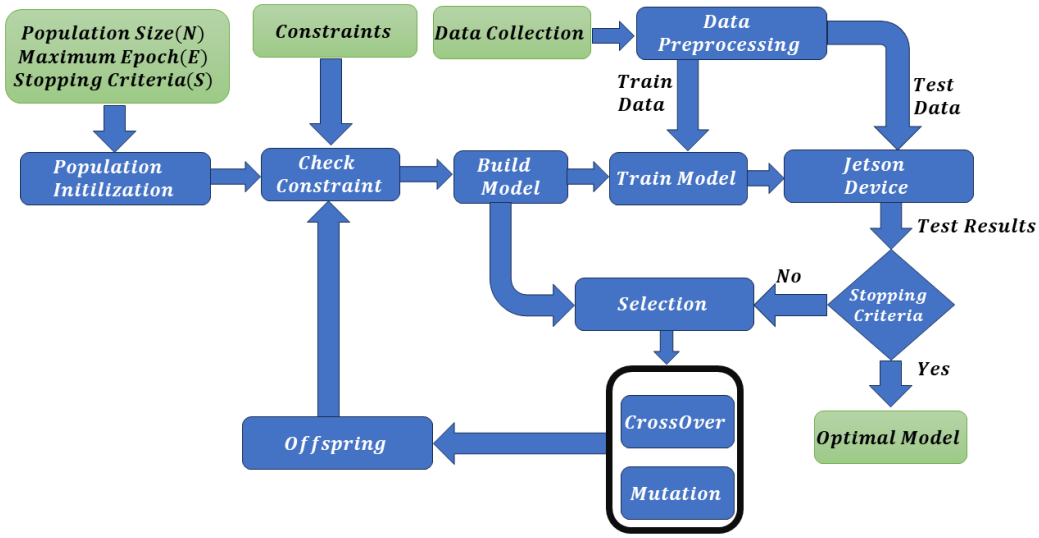

Figure 2. Block diagram of the proposed scheme.

3. Results and Comparative Analysis

This section discusses the results and conducts a comparative analysis of our proposed methodology, starting with a detailed overview of the datasets and parameter configurations.

3.1. Datasets and Parameter Configuration

The datasets used in this research were compiled by the National Renewable Energy Laboratory (NREL) and focus on three significant locations in Pakistan: Gwadar, Pasni, and Jhimpir. Despite being the fifth-most populous country, Pakistan has faced an energy crisis for decades, and the resources available to the government are insufficient to install commercial windmills. Instead, domestic small-scale windmills are installed by the local government with a lower budget, supporting the United Nations Sustainable Development Goals (SDGs), particularly SDG 7 (Affordable and Clean Energy).

The datasets contain meteorological variables at 60-min intervals, including temperature, wind speed (WS), wind direction, atmospheric pressure, and cloud type. This study focuses on DDS for MCD, targeting one-step-ahead forecasts. The dataset was preprocessed to include measurements from the past 24 h and then normalized. The model aimed to estimate the WS (in m/s) for the upcoming 25th hour, denoted as $\hat{y}_i$. After prediction, denormalization transformed $\hat{y}_i$ back to the actual scale to determine the real-world wind speed, y_i.

The maximum values for the model parameters are represented as a row vector $\mathbf{a}_{\max}$: mhs $= 512$, mnh $= 16$, mffd $= 8$, mnb $= 8$, mmlpu $= 256$, mdo $= 1$, mmlpdo $= 1$, and mlstmu $= 128$. Thus, $\mathbf{a}_{\max} = [512, 16, 8, 8, 256, 1, 1, 128]$. The models were trained on an Nvidia GeForce RTX 3080 Ti and, to ensure practical deployment on MCDs, the models were optimized for Nvidia Jetson Nano devices. The Nvidia Jetson Nano device is designed for edge computing and features a quad-core ARM Cortex-A57 CPU and an integrated GPU, making it energy-efficient and suitable for real-time applications in renewable energy management.

3.2. Results

In this study, the parameter λ in Equation (18) was set to 0.001. Additionally, α was set to 1 m/s, and β was set to 1,048,576 Bytes. These parameter values were intuitively determined by evaluating the performance of the baseline model, which has the maximum possible size for hyperparameters. Additionally, five distinct experiments were conducted, each varying the value of ϵ in Equation (18). The chosen values for ϵ in these experiments were 0.01, 0.25, 0.5, 0.75, and 0.99. In the experiment where the value of ϵ was 0.01, the primary focus was on minimizing the MS, with negligible concern for the *MSE*. As the value of ϵ increased to 0.25, there was a noticeable shift in priority, with greater emphasis on reducing *MSE* while lessening the importance of MS. This trend continued with higher values of ϵ; as ϵ increased, the significance given to minimizing *MSE* grew, consequently reducing the emphasis on the size of the model.

Initial training over 25 epochs yielded five distinct models for each case. Upon fine-tuning these models using the dataset from Gwadar city, results were obtained for varying values of epsilon. These results are depicted in Figures 3–7. Each figure comprises six subplots: the first subplot summarizes the results of the five models, while the remaining five subplots detail the DDS each model occupies. For instance, Figure 3 features a model in the first row and second column, highlighted in purple, with the optimal parameters $X = [310, 5, 2, 2, 100, 0.21, 0.31, 37]$. The notation 'NH 5/16' indicates that the optimal number of heads is 5, within the maximum allowed limit of 16.

3.2.1. Experiment with $\epsilon = 0.01$

Figure 3 showcases the top five models for $\epsilon = 0.01$, focusing on the trade-off between MS and performance. The most efficient model achieved an impressive balance with a total cost of 0.28, an *MSE* of 0.11 m/s, and an MS of only 0.28 MB. The second model, with a total cost of 0.33, an *MSE* of 0.22 m/s, and an MS of 0.33 MB, showed an increase in both size and *MSE*. This makes it a less favorable choice, as it does not efficiently balance the trade-off between size and performance.

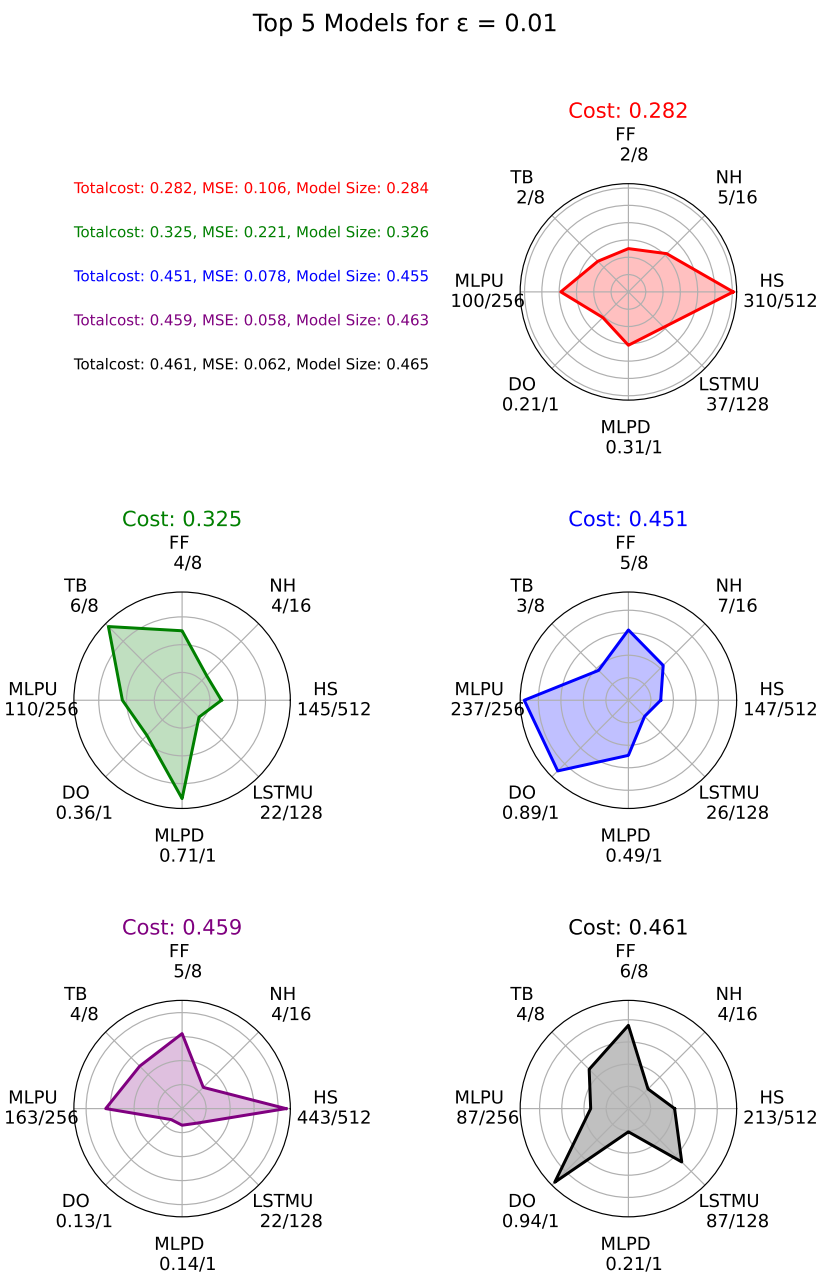

Figure 3. Top five models for $\epsilon = 0.01$.

In contrast, the third model demonstrated a significant improvement in performance at the expense of increased size. It registered a total cost of 0.45, an *MSE* of 0.08 m/s, and an MS of 0.45 MB, indicating a substantial enhancement in accuracy for a reasonable increment in MS. The fourth and fifth models, with slightly different sizes of 0.46 MB and 0.47 MB but identical total costs of 0.46 and MSEs of 0.06 m/s, showcased a consistent level of high performance for these larger models.

3.2.2. Experiment with $\epsilon = 0.25$

Figure 4 presents the top five models for $\epsilon = 0.25$, for which the balance between MS and performance was considered under different criteria compared to the earlier experiment. The best-performing model in this setup achieved a total cost of 0.29, an *MSE* of 0.06 m/s, and an MS of 0.36 MB, demonstrating an efficient balance between accuracy and compactness. The second model exhibited a total cost of 0.31, an *MSE* of 0.09 m/s, and an MS of 0.38 MB. This model indicates a preference for a slightly larger size while still maintaining high accuracy.

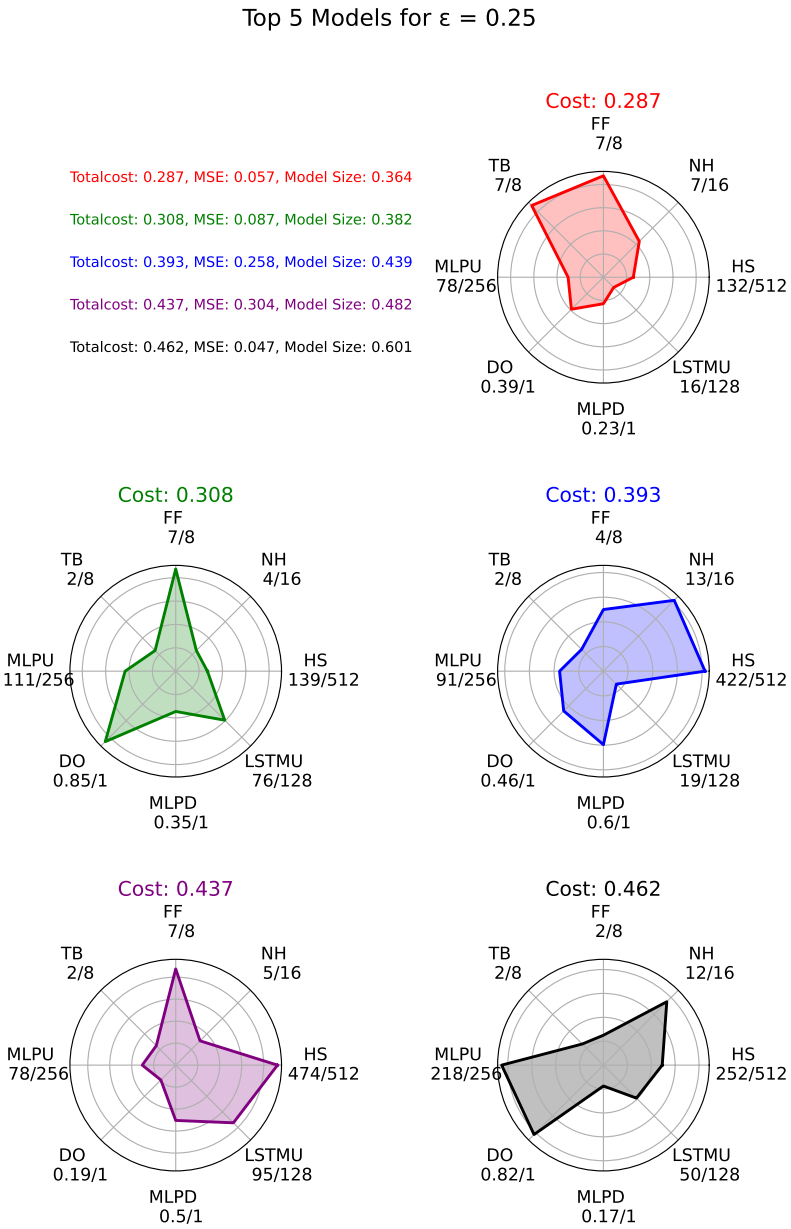

Figure 4. Top five models for $\epsilon = 0.25$.

Moving to the third model, it presented a total cost of 0.39, an increased *MSE* of 0.26 m/s, and an MS of 0.44 MB. This suggests a shift toward accommodating a larger MS in exchange for a moderate increase in error rates. The fourth model showed an increase in both total cost (0.44) and MS (0.48 MB) but with a slightly higher *MSE* of 0.3 m/s, indicating a trade-off for a larger MS against a marginal increase in error performance. Interestingly, the final model, despite having the largest size at 0.6 MB, achieved a low

MSE of 0.05 m/s and a total cost of 0.46. This outcome suggests that a larger MS can significantly improve accuracy, albeit at the cost of increased resource consumption. The detailed optimal parameters for these models are depicted in Figure 4.

3.2.3. Experiment with $\epsilon = 0.50$

In the experiment characterized by $\epsilon = 0.5$, as illustrated in Figure 5, a distinct set of outcomes was observed, indicating a more balanced trade-off between MS and performance. The first model achieved a total cost of 0.25, an *MSE* of 0.05 m/s, and an MS of 0.44 MB, demonstrating a favorable balance between accuracy and size.

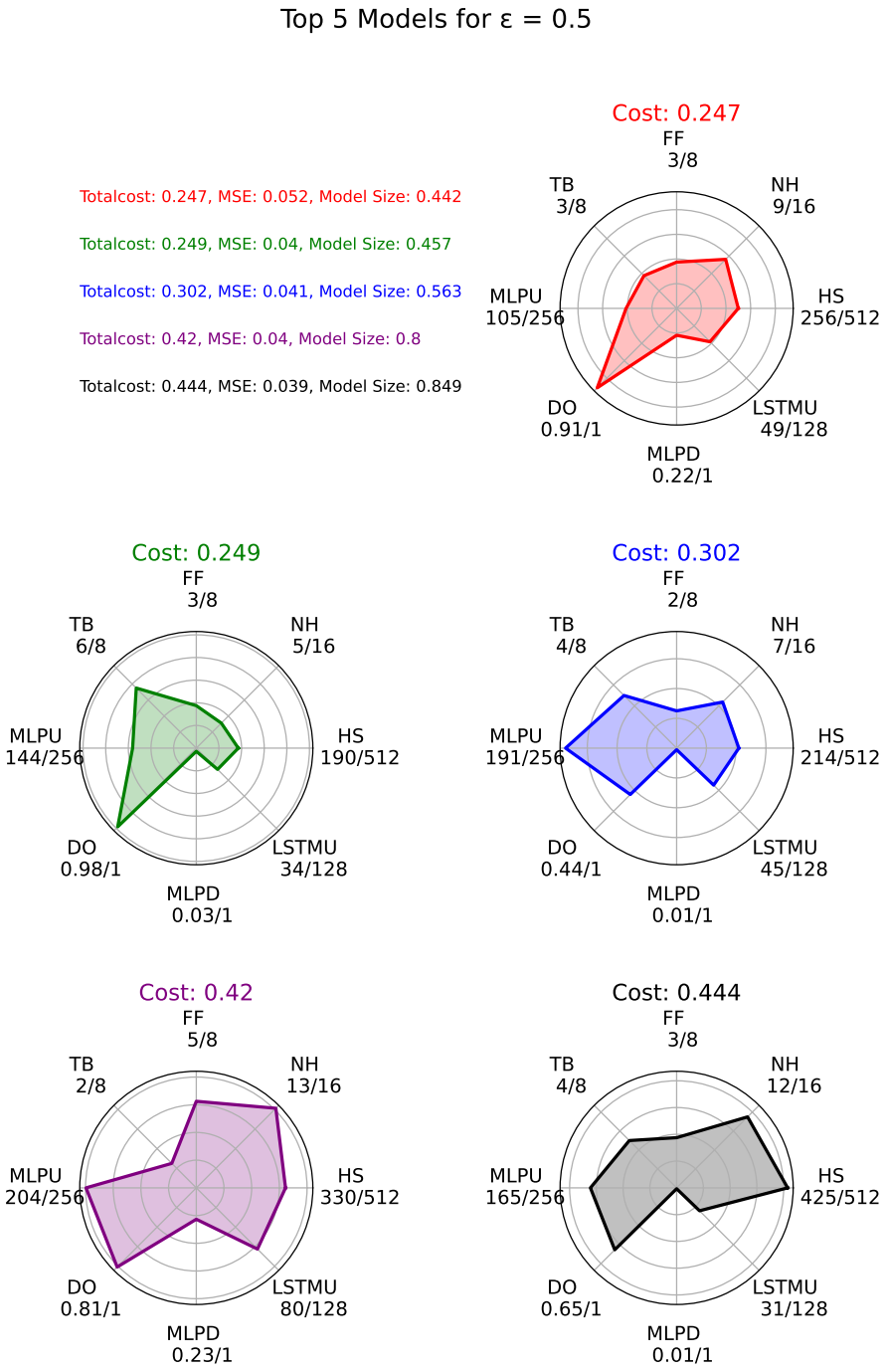

Figure 5. Top five models for $\epsilon = 0.5$.

The second model, with a slightly larger size of 0.46 MB, also recorded a total cost of 0.25 but achieved a slightly lower *MSE* of 0.04 m/s, indicating an incremental improvement in accuracy. Moving on to the third model, the total cost rose to 0.3, the *MSE* to 0.04 m/s, and the MS to 0.56 MB. This model indicates a preference for a larger size while maintaining a low error rate, balancing cost and performance.

The fourth model, significantly larger at 0.8 MB, demonstrated a total cost of 0.42 and an *MSE* of 0.04 m/s. This indicates a considerable enhancement in accuracy, which comes with a substantial increase in size. Finally, the fifth model, with a size of 0.85 MB, showed a total cost of 0.44 and an *MSE* of 0.04 m/s. Although similar in size to the fourth model, it presented a slightly higher total cost, making it slightly less efficient in comparison. The detailed parameters for these models, including their specific configurations, are further elucidated in Figure 5.

3.2.4. Experiment with $\epsilon = 0.75$

The experiment with $\epsilon = 0.75$, as shown in Figure 6, revealed a compelling set of results, emphasizing a greater focus on model accuracy while also considering MS. The first model in this series achieved a total cost of 0.218, an *MSE* of 0.05 m/s, and an MS of 0.721 MB, indicating a shift toward prioritizing accuracy while maintaining a moderately large size.

The second model, with an MS of 0.927 MB, had a total cost of 0.267 and an *MSE* of 0.047 m/s, indicating a further enhancement in accuracy but a noticeable increase in size. Advancing to the third model, the total cost rose to 0.324, the *MSE* to 0.076 m/s, and the MS to 1.065 MB, reflecting a continued emphasis on reducing the error rate, albeit at the expense of larger model dimensions.

The fourth model, at 1.191 MB, showed a cost of 0.343 and an *MSE* of 0.06 m/s. Despite its size, it balanced size and accuracy well. The fifth model, the largest at 1.311 MB, recorded a cost of 0.368 and an *MSE* of 0.054 m/s. Although it was the largest, it maintained good accuracy, illustrating a balance between size and performance. The details of these models are shown in Figure 6.

3.2.5. Experiment with $\epsilon = 0.99$

The final experiment, with $\epsilon = 0.99$, as shown in Figure 7, represents an approach with an emphasis on reducing *MSE* while placing minimal constraints on MS. Hence, the first model achieved a total cost of 0.059, an *MSE* of 0.047 m/s, and an MS of 1.217 MB, showing a significant emphasis on accuracy. The second model, with a total cost of 0.062, an *MSE* of 0.05 m/s, and a slightly smaller size of 1.163 MB, maintained a similar level of accuracy with a marginal reduction in size.

Moreover, the third model, with a total cost of 0.063, an *MSE* of 0.057 m/s, and a smaller MS of 0.655 MB, presented a better balance between size and accuracy compared to its predecessors. The fourth model, at 0.781 MB, showed a higher cost of 0.177 and an *MSE* of 0.171 m/s, indicating improvements in both size and accuracy. Finally, the fifth model, despite a slightly smaller size of 0.719 MB, recorded the highest cost of 0.207 and an *MSE* of 0.202 m/s. This highlights the difficulty of optimizing for high accuracy while maintaining a reasonable MS. Detailed insights into the model configurations and trade-offs are provided in Figure 7.

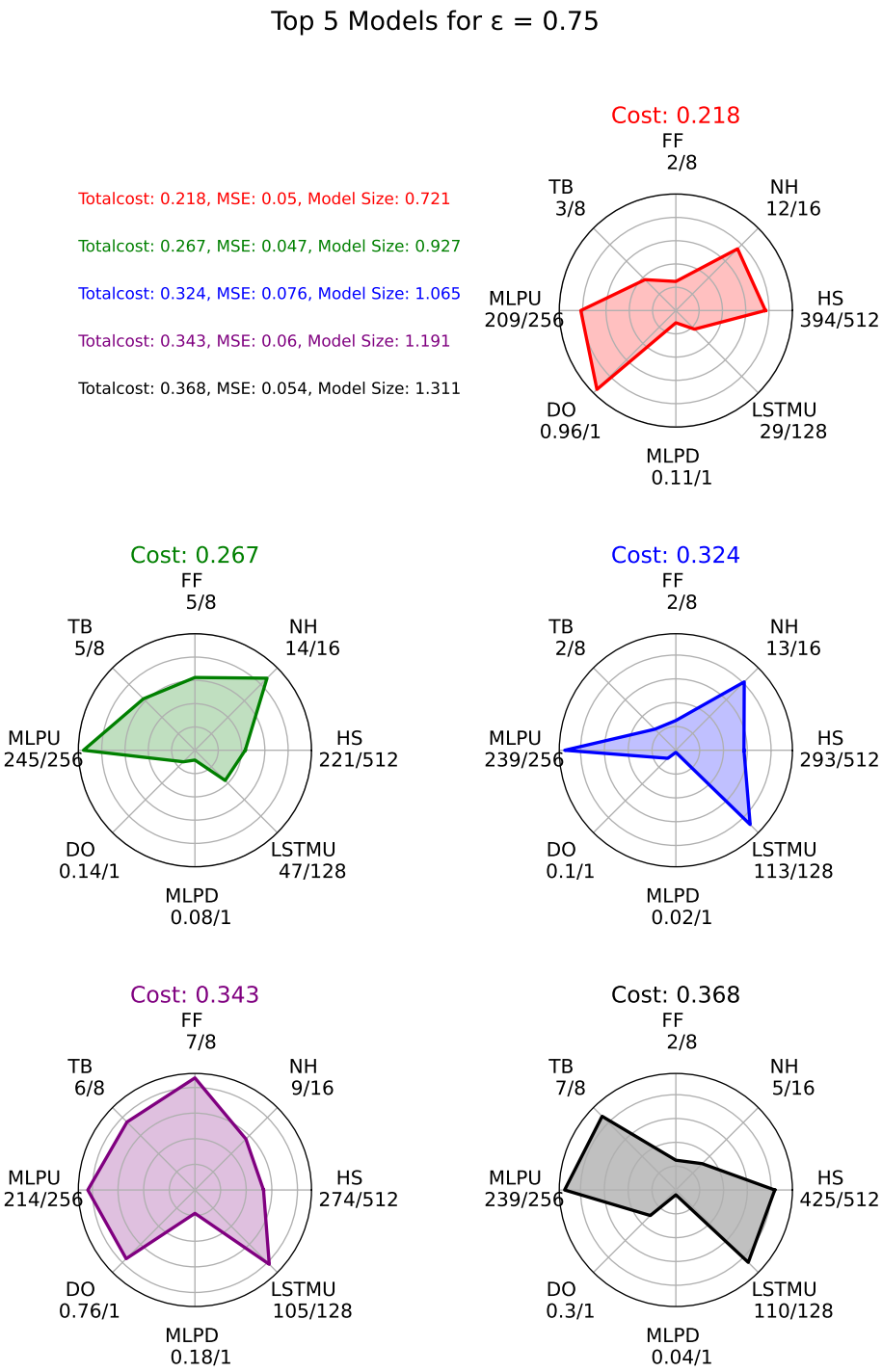

Figure 6. Top five models for $\epsilon = 0.75$.

This experiment revealed the top five models for each epsilon value, highlighting the trade-offs. For example, in Experiment 1 with $\epsilon = 0.01$, an *MSE* of 0.08 m/s or less is acceptable. We also limited MS to under 500 KB. The third model is suitable with a size of 0.455 MB and an *MSE* of 0.078 m/s. This approach emphasizes that model selection depends on specific application requirements and constraints.

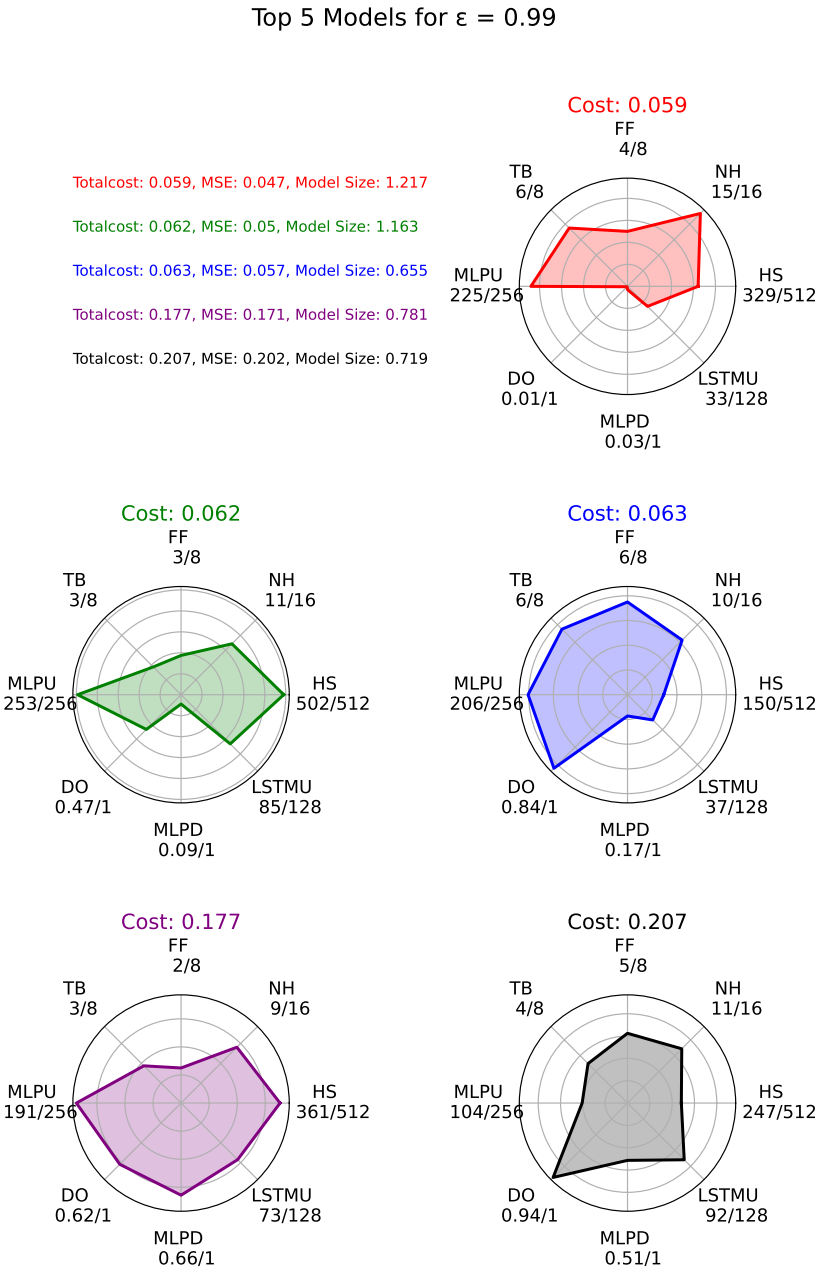

Figure 7. Top five models for $\epsilon = 0.99$.

3.3. Performance of Optimal Models Across Datasets

The models and results presented above were derived using the Gwadar dataset. However, when these models, configured with their optimal hyperparameters, were trained and tested on data from the other two datasets, Jhimpir and Pasni, they also exhibited acceptable performance, as shown in Table 1.

Table 1. Performance of optimal models across datasets.

Dataset	ϵ	Size (MB)	*MSE* (m/s)	*MAE* (m/s)	R^2	Latency (s)
Jhimpir	0.01	0.2862	0.0848	0.2180	0.9732	0.0848
	0.25	0.3812	0.0932	0.2284	0.9706	0.0341
	0.50	0.4587	0.0442	0.1573	0.9860	0.1557
	0.75	0.7232	0.0442	0.1573	0.9860	0.1620
	0.99	1.2104	0.0451	0.1583	0.9858	0.6929
Gwadar	0.01	0.2836	0.1065	0.2404	0.9600	0.0915
	0.25	0.3643	0.0568	0.1744	0.9786	0.1992
	0.50	0.4416	0.0522	0.1654	0.9804	0.1773
	0.75	0.7211	0.0498	0.1591	0.9850	0.3022
	0.99	1.2168	0.0470	0.1592	0.9823	0.6224
Pasni	0.01	0.2849	0.5513	0.5620	0.7900	0.0876
	0.25	0.3829	0.6610	0.6160	0.7514	0.0311
	0.50	0.4413	0.5412	0.5892	0.7964	0.1610
	0.75	0.7233	0.3263	0.4346	0.8772	0.2982
	0.99	1.2107	0.2856	0.4065	0.8925	0.6639

3.4. Analyzing the Influence of ϵ on Model Metrics

This section explores the relationship between the parameter ϵ and various model performance indicators, specifically *MSE*, *MAE*, R^2 score, and MS. Figure 8 graphically presents these relationships, allowing for a comprehensive comparison of how changes in ϵ affect each metric. As can be seen, a trend emerged showing that an increase in ϵ led to changes in both *MSE* and *MAE*, demonstrating the direct impact of ϵ on minimizing these error metrics. Furthermore, a positive correlation between ϵ and the R^2 score was also observed, suggesting improved predictive performance with larger ϵ values, and the MS increased with the value of ϵ.

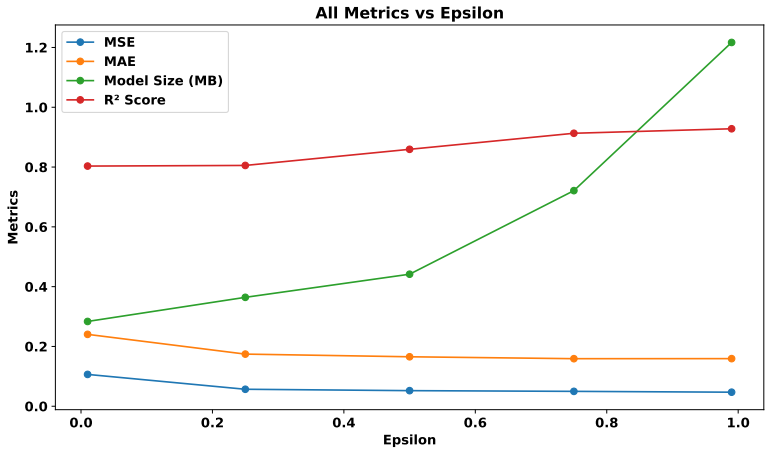

Figure 8. Impact of varying ϵ values on *MSE*, *MAE*, MS, and R^2 score.

3.5. Comparison

This section compares the proposed optimal model with three existing schemes. First, this work compares DeepAR [35], an autoregressive recurrent neural network model for probabilistic forecasting, with our proposed scheme. The second model for comparison is a CNN-based probabilistic forecasting framework [36]. Lastly, this work investigates the CNN-LSTM model [37], which integrates a CNN and an LSTM network for wind power prediction.

As shown in the results of the comprehensive comparative analysis presented in Table 2, the proposed scheme consistently outperformed existing models like DeepAR, DeepTCN, and CNN-LSTM across all datasets. For example, on the Jhimpir dataset, our proposed scheme achieved up to a 98.78% reduction in size, a 51.89% improvement in *MSE*, a 31.80% improvement in *MAE*, and a 9.80% increase in the R^2 score compared to DeepAR. Similar trends were observed on the Gwadar and Pasni datasets, with substantial improvements across all metrics. Specifically, compared to DeepTCN on the Gwadar dataset, the proposed scheme showed a 92.24% reduction in size, a 32.66% better *MSE*, an 18.15% better *MAE*, and a 3.52% higher R^2 score. Similarly, on the Pasni dataset, compared to CNN-LSTM, the proposed scheme demonstrated a 90.18% reduction in size, a 43.07% improvement in *MSE*, a 23.60% improvement in *MAE*, and an 18.29% increase in the R^2 score. The predicted versus actual data for each dataset are shown in Figures 9–11.

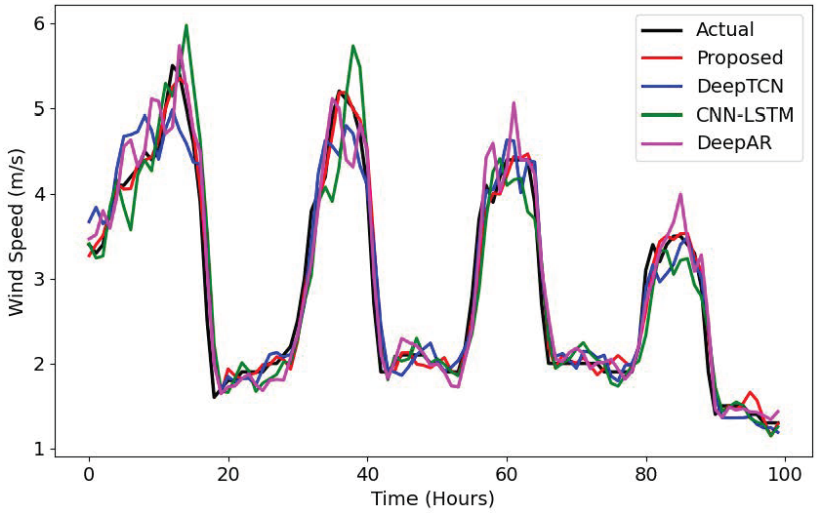

Figure 9. Performance comparison of DeepAR, DeepTCN, CNN-LSTM, and the proposed scheme on the Jhimpir dataset. The x-axis represents samples recorded at 1 h intervals, and the y-axis represents the wind speed (m/s).

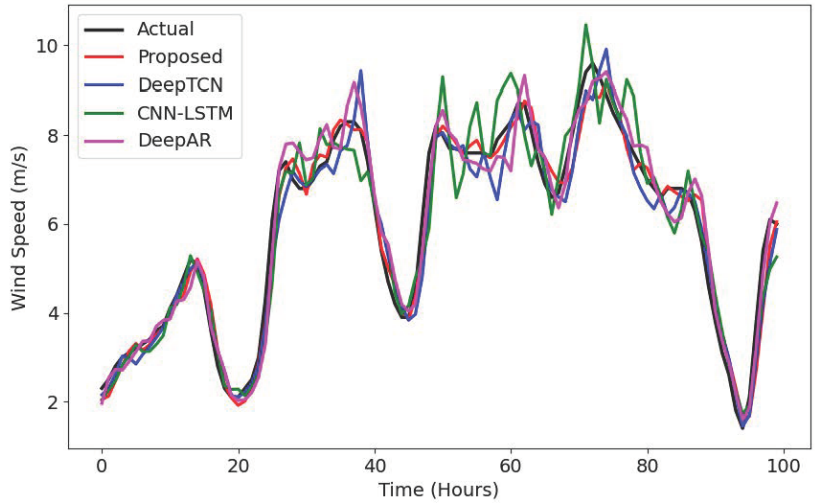

Figure 10. Performance comparison of DeepAR, DeepTCN, CNN-LSTM, and the proposed scheme on the Gwadar dataset. The x-axis represents samples recorded at 1 h intervals, and the y-axis represents the wind speed (m/s).

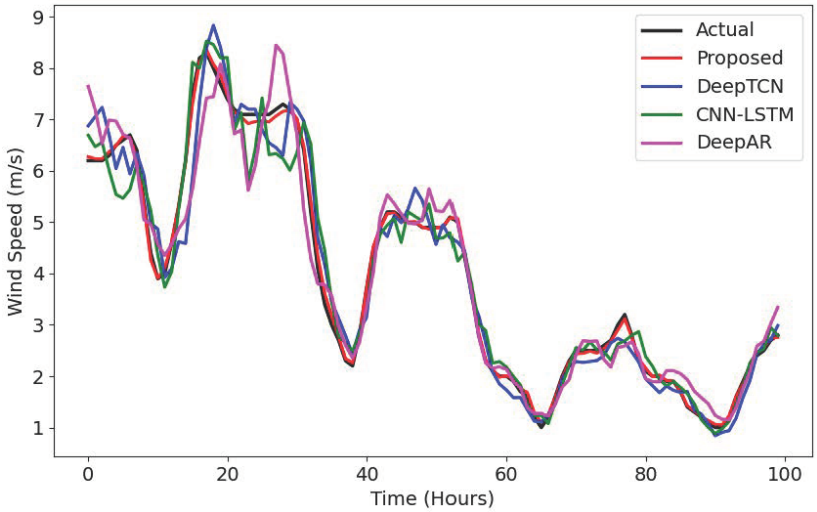

Figure 11. Performance comparison of DeepAR, DeepTCN, CNN-LSTM, and the proposed scheme on the Pasni dataset. The x-axis represents samples recorded at 1 h intervals, and the y-axis represents the wind speed (m/s).

Table 2. Comparison with existing schemes.

Dataset	Scheme	Size (MB)	*MSE* (m/s)	*MAE* (m/s)	R^2
Jhimpir	DeepAR [35]	100.3615	0.0921	0.2321	0.8974
	DeepTCN [36]	15.6832	0.0751	0.2075	0.9332
	CNN-LSTM [37]	12.6971	0.0793	0.2251	0.9162
	Proposed Scheme	1.2104	0.0451	0.1583	0.9858
Gwadar	DeepAR [35]	100.3628	0.0903	0.2561	0.9600
	DeepTCN [36]	15.6756	0.0698	0.1945	0.9489
	CNN-LSTM [37]	12.6348	0.0815	0.2678	0.8756
	Proposed Scheme	1.2168	0.0470	0.1592	0.9823
Pasni	DeepAR [35]	100.5483	0.4361	0.5620	0.7900
	DeepTCN [36]	15.6540	0.4275	0.4598	0.8219
	CNN-LSTM [37]	12.3256	0.5017	0.5321	0.7545
	Proposed Scheme	1.2107	0.2856	0.4065	0.8925

4. Conclusions

In this study, a Hybrid Baseline Model (HBM) was developed, integrating transformers for feature extraction with Long Short-Term Memory (LSTM) networks for time-series forecasting, specifically tailored for wind speed prediction (WSP) on low-power, memory-constrained devices (MCDs). The combination of transformers and LSTM networks enhanced the model's ability to extract features and process sequential data, as evidenced by significant reductions in mean squared error (*MSE*) and mean absolute error (*MAE*) compared to existing methods. A novel cost function was introduced, effectively managing the trade-offs between prediction accuracy and model size (MS), which is crucial for deployment on devices with limited resources. The results demonstrated up to a 92.24% reduction in model size and a 51.03% improvement in *MSE* over traditional models like DeepAR, highlighting the effectiveness of the proposed approach. The successful real-time deployment on Jetson Nano devices further confirmed the practical applicability of the model. Training on a server and testing on MCDs ensured both accuracy and feasibility for real-world scenarios. Additionally, the use of a genetic algorithm for iterative optimization based on actual test results further enhanced the model's performance and adaptability.

Author Contributions: Conceptualization, L.A. and R.Z.; methodology, E.S.A., S.S.H., L.A. and M.A.; software, L.A.; validation, L.A., E.S.A. and S.S.H.; formal analysis, L.A.; investigation, L.A. and M.A.; resources, L.A. and M.A.; writing—original draft preparation, L.A., M.A. and K.A.S.; writing—review and editing, L.A., R.Z., M.A.W. and K.A.S.; visualization, L.A.; supervision, R.Z.; project administration, R.Z.; funding acquisition, M.A. and K.A.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Princess Nourah bint Abdulrahman University Researchers Supporting Project, number PNURSP2025R757, Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia. The authors would also like to acknowledge the support of EIAS Data Science Lab, Prince Sultan University.

Data Availability Statement: The data used in this study are sourced from the National Renewable Energy Laboratory (NREL), available at https://www.nrel.gov/(accessed on 23 September 2024).

Acknowledgments: The authors would like to acknowledge the Princess Nourah bint Abdulrahman University Researchers Supporting Project, number PNURSP2025R757, Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia. The authors would like to thank Prince Sultan University for their valuable support.

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

Appendix A

The computational complexity of the hardware-centric discrete design space optimization scheme is derived through a sequential analysis of its fundamental operations. Let P represent the population size, G the number of generations, N the count of trainable parameters, D the training dataset size, α the testing data fraction ($0 < \alpha \leq 1$), and S the one-time hardware initialization time.

The total computational cost comprises three principal components. First, the hardware setup requires a fixed initialization duration $O(S)$. Second, per-generation costs accumulate through simulation training and hardware testing. Training dominates computational effort through forward/backward passes over the full dataset, scaling as $O(PND)$. Testing incurs inference costs proportional to the subset αD, yielding $O(\alpha PND)$. Genetic operations exhibit linear population scaling $O(P)$. Aggregating these components across G generations produces

$$C_{\text{total}} = O(S) + G \cdot [O(PND) + O(\alpha PND) + O(P)] \tag{A1}$$

For large-scale deployments, where $D \gg 1$, lower-order terms become negligible, simplifying to

$$C_{\text{total}} \approx O(S + GPND) \tag{A2}$$

Table A1. Computational complexity classification.

Component	Complexity Type	Scaling Behavior	Dominance
Hardware Setup	Constant	$O(S)$	Negligible
Training	Cubic	$O(PND)$	Most Dominant
Testing	Cubic (Reduced)	$O(\alpha PND)$	Significant
Genetic Operations	Linear	$O(P)$	Least Significant
Total Asymptotic Cost	Cubic × Linear	$O(S + GPND)$	Training-Driven

Practical implementation constraints emerge from this analysis. Memory limitations on edge devices cap model size at $N \leq 10^5$ parameters for 4GB RAM systems. Parallel evaluation across K devices reduces the effective computation time to $O(S + \frac{GPND}{K})$, while

testing subset reduction ($\alpha \to 0$) trades evaluation reliability for faster iterations. These constraints necessitate balancing the population size P, generations G, and model complexity N against available hardware resources.

References

1. Lee, J.; Zhao, F. *Global Wind Report 2022*; Global Wind Energy Council: Brussels, Belgium, 2022.
2. Brabec, M.; Craciun, A.; Dumitrescu, A. Hybrid numerical models for wind speed forecasting. *J. Atmos.-Sol.-Terr. Phys.* **2021**, *220*, 105669. [CrossRef]
3. Moreno, S.; Mariani, V.; dos Santos Coelho, L. Hybrid multi-stage decomposition with parametric model applied to wind speed forecasting in Brazilian northeast. *Renew. Energy* **2021**, *164*, 1508–1526. [CrossRef]
4. Nascimento, E.G.S.; de Melo, T.A.; Moreira, D.M. A transformer-based deep neural network with wavelet transform for forecasting wind speed and wind energy. *Energy* **2023**, *278*, 127678. [CrossRef]
5. Huang, H.; Chen, J.; Sun, R.; Wang, S. Short-term traffic prediction based on time series decomposition. *Phys. A* **2022**, *585*, 126441. [CrossRef]
6. Torres, J.L.; Garcia, A.; De Blas, M.; De Francisco, A. Forecast of hourly average wind speed with ARMA models in Navarre (Spain). *Sol. Energy* **2005**, *79*, 65–77. [CrossRef]
7. Li, L.; Xu, Q.; Wang, W.J.; Li, S.; Xue, S.; Lian, P.Y.; Wang, C.S.; He, F.L. ARMA model-based wind speed prediction for large radio telescope. *Acta Astron. Sin.* **2022**, *63*, 70.
8. Moreno, S.; dos Santos Coelho, L. Wind speed forecasting approach based on singular spectrum analysis and adaptive neuro fuzzy inference system. *Renew. Energy* **2018**, *126*, 736–754. [CrossRef]
9. Bechrakis, D.; Sparis, P. Wind speed prediction using artificial neural networks. *Wind Eng.* **1998**, *22*, 287–295.
10. Mohandes, M.A.; Halawani, T.O.; Rehman, S.; Hussain, A.A. Support vector machines for wind speed prediction. *Renew. Energy* **2004**, *29*, 939–947. [CrossRef]
11. Geng, D.; Zhang, H.; Wu, H. Short-term wind speed prediction based on principal component analysis and LSTM. *Appl. Sci.* **2020**, *10*, 4416. [CrossRef]
12. Cai, R.; Xie, S.; Wang, B.; Yang, R.; Xu, D.; He, Y. Wind speed forecasting based on extreme gradient boosting. *IEEE Access* **2020**, *8*, 175063–175069. [CrossRef]
13. Aslam, L.; Zou, R.; Awan, E.; Butt, S.A. Integrating Physics-Informed Vectors for Improved Wind Speed Forecasting with Neural Networks. In Proceedings of the 2024 14th Asian Control Conference (ASCC), Dalian, China, 5–8 July 2024.
14. Liu, H.; Tian, H.; Li, Y. Comparison of two new ARIMA-ANN and ARIMA-Kalman hybrid methods for wind speed prediction. *Appl. Energy* **2012**, *98*, 415–424. [CrossRef]
15. Zhang, Y.; Le, J.; Liao, X.; Zheng, F.; Li, Y. A novel combination forecasting model for wind power integrating least square support vector machine, deep belief network, singular spectrum analysis and locality-sensitive hashing. *Energy* **2019**, *168*, 558–572. [CrossRef]
16. Wan, A.; Chang, Q.; Khalil, A.B.; He, J. Short-term power load forecasting for combined heat and power using CNN-LSTM enhanced by attention mechanism. *Energy* **2023**, *282*, 128274. [CrossRef]
17. Pan, S.; Yang, B.; Wang, S.; Guo, Z.; Wang, L.; Liu, J.; Wu, S. Oil well production prediction based on CNN-LSTM model with self-attention mechanism. *Energy* **2023**, *284*, 128701. [CrossRef]
18. Li, W.; Li, Y.; Garg, A.; Gao, L. Enhancing real-time degradation prediction of lithium-ion battery: A digital twin framework with CNN-LSTM-attention model. *Energy* **2024**, *286*, 129681. [CrossRef]
19. Ata Teneler, A.; Hassoy, H. Health effects of wind turbines: A review of the literature between 2010–2020. *Int. J. Environ. Health Res.* **2023**, *33*, 143–157. [CrossRef]
20. Gkeka-Serpetsidaki, P.; Papadopoulos, S.; Tsoutsos, T. Assessment of the visual impact of offshore wind farms. *Renew. Energy* **2022**, *190*, 358–370. [CrossRef]
21. Bilgili, M.; Alphan, H. Visual impact and potential visibility assessment of wind turbines installed in Turkey. *Gazi Univ. J. Sci.* **2022**, *35*, 198–217. [CrossRef]
22. Lehnardt, Y.; Barber, J.R.; Berger-Tal, O. Effects of wind turbine noise on songbird behavior during nonbreeding season. *Conserv. Biol.* **2024**, *38*, e14188. [CrossRef]
23. Teff-Seker, Y.; Berger-Tal, O.; Lehnardt, Y.; Teschner, N. Noise pollution from wind turbines and its effects on wildlife: A cross-national analysis of current policies and planning regulations. *Renew. Sustain. Energy Rev.* **2022**, *168*, 112801. [CrossRef]
24. Zolotoff-Pallais, J.M.; Perez, A.M.; Donaire, R.M. General Comparative Analysis of Bird-Bat Collisions at a Wind Power Plant in the Department of Rivas, Nicaragua, between 2014 and 2022. *Eur. J. Biol. Biotechnol.* **2024**, *5*, 1–7. [CrossRef]
25. Richardson, S.M.; Lintott, P.R.; Hosken, D.J.; Economou, T.; Mathews, F. Peaks in bat activity at turbines and the implications for mitigating the impact of wind energy developments on bats. *Sci. Rep.* **2021**, *11*, 3636. [CrossRef] [PubMed]

26. Choi, D.Y.; Wittig, T.W.; Kluever, B.M. An evaluation of bird and bat mortality at wind turbines in the Northeastern United States. *PLoS ONE* **2020**, *15*, e0238034. [CrossRef]
27. Barter, G.E.; Sethuraman, L.; Bortolotti, P.; Keller, J.; Torrey, D.A. Beyond 15 MW: A cost of energy perspective on the next generation of drivetrain technologies for offshore wind turbines. *Appl. Energy* **2023**, *344*, 121272. [CrossRef]
28. Wiser, R.; Millstein, D.; Bolinger, M.; Jeong, S.; Mills, A. The hidden value of large-rotor, tall-tower wind turbines in the United States. *Wind Eng.* **2021**, *45*, 857–871. [CrossRef]
29. Turc Castellà, F.X. Operations and Maintenance Costs for Offshore Wind Farm. Analysis and Strategies to Reduce O&M Costs. Ph.D. Thesis, Universitat Politècnica de Catalunya, Escola Tècnica Superior d'Enginyeria Industrial de Barcelona, Barcelona, Spain, 2020.
30. Abdelateef Mostafa, M.; El-Hay, E.A.; ELkholy, M.M. Recent trends in wind energy conversion system with grid integration based on soft computing methods: Comprehensive review, comparisons and insights. *Arch. Comput. Methods Eng.* **2023**, *30*, 1439–1478. [CrossRef]
31. Ahmed, S.D.; Al-Ismail, F.S.; Shafiullah, M.; Al-Sulaiman, F.A.; El-Amin, I.M. Grid integration challenges of wind energy: A review. *IEEE Access* **2020**, *8*, 10857–10878. [CrossRef]
32. Suo, L.; Peng, T.; Song, S.; Zhang, C.; Wang, Y.; Fu, Y.; Nazir, M.S. Wind speed prediction by a swarm intelligence based deep learning model via signal decomposition and parameter optimization using improved chimp optimization algorithm. *Energy* **2023**, *276*, 127526. [CrossRef]
33. Saini, V.K.; Kumar, R.; Al-Sumaiti, A.S.; Sujil, A.; Heydarian-Forushani, E. Learning based short term wind speed forecasting models for smart grid applications: An extensive review and case study. *Electr. Power Syst. Res.* **2023**, *222*, 109502. [CrossRef]
34. Zhang, Y.; Pan, G.; Chen, B.; Han, J.; Zhao, Y.; Zhang, C. Short-term wind speed prediction model based on GA-ANN improved by VMD. *Renew. Energy* **2020**, *156*, 1373–1388. [CrossRef]
35. Salinas, D.; Flunkert, V.; Gasthaus, J.; Januschowski, T. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *Int. J. Forecast.* **2020**, *36*, 81–91. [CrossRef]
36. Chen, Y.; Kang, Y.; Chen, Y.; Wang, Z. Probabilistic forecasting with temporal convolutional neural network. *Neurocomputing* **2020**, *399*, 491–501. [CrossRef]
37. Wu, Q.; Guan, F.; Lv, C.; Huang, Y. Ultra-short-term multi-step wind power forecasting based on CNN-LSTM. *IET Renew. Power Gener.* **2021**, *15*, 1019–1029. [CrossRef]