https://jaslda.journals.pnu.ac.ir

Payame Noor University

Statistical Learning & Data Analysis
Volume 1, Issue 1, Summer- Autumn 2026, Pages 1–22
doi:10.30473/jaslda.2026.13501

OPEN ACCESS

JASLDA
Journal of Advances in Statistical Learning and Data Analysis

Research Article

Benchmarking Recurrent Neural Network Architectures for Sequential Prediction with Application to Air Pollution Forecasting

Abbas Pak[1,✉], Mohammad Javad Nematollahi[2]

[1,✉]Department of Computer Sciences, Shahrekord University, Shahrekord, Iran
[2]Department of Geology, Faculty of Sciences, Urmia University, Urmia, Iran

Article History

Received: August 05, 2026;
Accepted: September 12, 2026;
Available online: September 19, 2026.

Abstract. This study presents a comprehensive benchmarking analysis of state-of-the-art recurrent neural network (RNN) architectures for sequential prediction tasks. A wide range of recurrent models, including Simple RNN, Independently Recurrent Neural Network (IndRNN), Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), Bidirectional LSTM (BiLSTM), Residual BiLSTM (ResBiLSTM) are systematically investigated under a unified experimental framework. The proposed benchmarking framework evaluates all models on a real-world air pollution prediction task using multivariate environmental and meteorological data. Performance is assessed using multiple standard metrics, including root mean square error (RMSE), mean absolute error (MAE), and the coefficient of determination (R^2), enabling a comprehensive evaluation of predictive accuracy and generalization capability. Experimental results demonstrate that IndRNN achieves the best overall performance, providing superior accuracy while maintaining a relatively simple and computationally efficient architecture compared to other deep recurrent variants. The actual-versus-predicted analysis further supports this conclusion and IndRNN exhibits the closest alignment between predicted and observed values, achieving the highest coefficient of determination ($R^2 = 0.6375$) and the lowest prediction error (RMSE = 17.0203) among the evaluated recurrent architectures. These findings highlight the effectiveness of IndRNN as a strong candidate for practical sequential forecasting applications.

Keywords. Sequential prediction; Deep learning; Recurrent neural network; Air pollution forecasting.

MSC. 92B20, 62M20 .

How to Cite
Abbas Pak, Mohammad Javad Nematollahi. (2026). "Benchmarking Recurrent Neural Network Architectures for Sequential Prediction with Application to Air Pollution Forecasting". *Journal of Advances in Statistical Learning and Data Analysis*, 1(1), 1–22. https://doi.org/10.30473/jaslda.2026.13501

✉Corresponding author. **E-mail:** abbas.pak1982@gmail.com
©2026 by the authors. Licensee PNU, Tehran, Iran. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution 4.0 International (CC BY4.0) (http:/creativecommons.org/licenses/by/4.0)

1 Introduction

Sequential prediction is a fundamental problem in machine learning and data analytics, aiming to model temporal dependencies in ordered observations and forecast future states of dynamic systems. Accurate sequential forecasting plays a critical role in numerous domains, including finance, healthcare, transportation, energy systems, climate modeling, and environmental monitoring [1]. Unlike conventional prediction tasks, sequential prediction requires models to capture temporal dependencies, nonlinear interactions, seasonality, and long-range relationships embedded within time-series data [2, 3]. Consequently, developing effective learning architectures for sequential data has become a central topic in machine learning and artificial intelligence research, motivating extensive advances in recurrent, convolutional, attention-based, and transformer-oriented forecasting models [4, 5, 6].

Traditional forecasting methods, including autoregressive integrated moving average (ARIMA), seasonal ARIMA (SARIMA), exponential smoothing, and other statistical approaches, have been extensively used for temporal modeling and forecasting applications across economics, energy, transportation, environmental science, and public health domains [7]. Although these methods provide interpretable solutions and perform adequately for relatively simple linear processes, their effectiveness often deteriorates when confronted with highly nonlinear, multivariate, and nonstationary real-world data. More recently, machine learning techniques, including Decision Trees, Random Forests, Extra Trees, Gradient Boosting Machines (GBM), and Light Gradient Boosting Machine (LightGBM), have demonstrated strong predictive capabilities by modeling complex nonlinear relationships, high-order feature interactions, and heterogeneous data structures without requiring strict assumptions of linearity or stationarity [8, 9, 10, 11, 12]. However, these methods generally operate on fixed-length feature representations and do not explicitly capture temporal dependencies, making them less suitable for sequential prediction problems characterized by long-term memory and dynamic temporal structures.

The emergence of deep learning has significantly advanced the field of sequential modeling by enabling hierarchical representation learning from complex temporal data [1, 13]. Transformer-based architectures attracted considerable attention in time-series forecasting because of their ability to model long-range temporal dependencies through self-attention mechanisms and to capture relationships across different positions in a sequence without relying exclusively on recurrent connections. However, their effectiveness can depend on factors such as data volume, sequence length, computational requirements, and model complexity. Among deep learning approaches, recurrent neural networks (RNNs) have emerged as one of the most influential architectures for sequential prediction because of their ability to process temporal information through recurrent connections and maintain internal memory states across time steps [14, 15]. Unlike feedforward neural networks, which process observations independently, RNNs propagate hidden-state information from previous time steps to subsequent ones, allowing the network to capture temporal dependencies and contextual relationships within sequential data [14, 16]. Consequently, RNN-based architectures have become fundamental tools for sequence modeling tasks such as time-series forecasting, speech recognition, natural language processing, and spatio-temporal prediction [16, 17]. However, conventional RNNs often suffer from vanishing and exploding gradient problems, limiting their capability to learn long-term dependencies in complex temporal sequences. To address these limitations, several advanced recurrent architectures have been proposed, including Long Short-Term Memory (LSTM) networks, Gated Recurrent Units (GRU), Bidirectional LSTMs (BiLSTM), Independently Recurrent Neural Network (IndRNN), and Residual Bidirectional LSTM (ResBiLSTM) [14, 18]. These architectures introduce specialized gating mechanisms, residual connections, attention modules, and spatiotemporal feature

extraction capabilities that significantly improve learning efficiency and predictive performance in complex sequential environments.

Despite substantial progress in recurrent deep learning models, a major limitation in the existing literature is the absence of a unified benchmarking framework that systematically compares a broad spectrum of recurrent architectures under identical experimental conditions. Many studies focus on only a small subset of recurrent architectures, often comparing one or two deep learning models against limited baselines. Moreover, differences in preprocessing procedures, feature selection methods, forecasting horizons, and evaluation metrics make it difficult to draw reliable conclusions regarding the relative strengths and weaknesses of competing architectures. Recent review studies have emphasized the need for unified benchmarking frameworks capable of evaluating state-of-the-art recurrent models under consistent experimental conditions [19].

Motivated by these challenges, this study presents a comprehensive benchmarking analysis of state-of-the-art recurrent neural network architectures for sequential prediction. The proposed framework systematically evaluates multiple recurrent models, including Simple RNN, IndRNN, LSTM, GRU, BiLSTM, and ResBiLSTM. To ensure a fair and reproducible comparison, all models are trained and evaluated using identical preprocessing, feature engineering, and validation protocols on a real-world air pollution forecasting dataset. Air quality prediction constitutes a challenging sequential prediction task due to the presence of complex nonlinear relationships, temporal dependencies, seasonal patterns, and external environmental influences. These characteristics make air pollution forecasting an appropriate testbed for assessing the effectiveness of modern recurrent neural network architectures.

The primary contributions of this work are fourfold. First, A unified benchmarking framework is developed for the comparative evaluation of state-of-the-art RNN architectures for sequential prediction. Second, multiple advanced recurrent architectures are systematically investigated under identical experimental settings. Third, the effectiveness of advanced recurrent architectures in modeling complex temporal dependencies is assessed using real-world environmental data. Finally, comprehensive performance evaluation is conducted using multiple statistical metrics to assess predictive accuracy, robustness, and generalization capability. The remainder of this paper is organized as follows. Section 2 describes the dataset, preprocessing procedures, the investigated models and benchmarking framework. Section 3 details the experimental setup and evaluation metrics. Section 4 discusses the comparative results and analysis. Section 5 provides a discussion of the key results and their broader significance.

2 Materials and Methods

2.1 Dataset Description and Preprocessing

The experiments are conducted using a real-world multivariate air quality dataset collected from an environmental monitoring station reported by [12, 20]. The dataset covers a ten-year period from January 2013 to December 2023, with observations recorded at a daily temporal resolution. This relatively long and consistent time span provides sufficient temporal coverage for training and testing the predictive models while allowing the models to capture seasonal and longer-term temporal patterns in air pollutant concentrations. The dataset contains measurements of atmospheric pollutants together with meteorological variables that influence pollutant dispersion

and accumulation. The predictive model includes carbon monoxide (CO), ozone (O3), nitrogen dioxide (NO2), sulfur dioxide (SO2), particulate matter with an aerodynamic diameter of 10 or less (PM10), temperature, dew point, relative humidity , wind speed, and atmospheric pressure, to estimate the concentration of fine particulate matter (PM2.5). To ensure data quality and improve model reliability, several preprocessing steps are applied before model development. At first, rows containing missing observations are removed to eliminate potential bias introduced by data imputation and to ensure that all models are trained using complete observations. Since sequential forecasting requires preservation of temporal dependencies, all observations are sorted according to their timestamps before feature engineering. Further, predictor variables are normalized using Min-Max normalization

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

where x denotes the original feature value. Normalization accelerates neural network convergence and prevents variables with large numerical ranges from dominating the optimization process.

2.2 Feature Engineering

Feature engineering is an essential step in enhancing the performance of sequential prediction models because it enriches the information available for forecasting. Historical pollutant observations are incorporated into the dataset through lagged variables, where measurements from previous time steps are used as additional explanatory features. This enables the models to capture temporal dependencies and persistence patterns commonly observed in air pollution data. To represent short-term temporal dynamics more effectively, rolling statistical measures are computed using window sizes of 3, 7, and 14 time steps. These statistics include the rolling mean, rolling standard deviation, rolling minimum, and rolling maximum, which collectively provide information about local trends, variability, and extreme values in the recent history of the series. Temporal variables often exhibit periodic behavior; therefore, cyclical encoding is applied to variables such as hour of the day, day of the week, and month of the year. Sine and cosine transformations are used to preserve the cyclical nature of these variables and to avoid artificial discontinuities between the end and the beginning of each cycle. Since recurrent neural networks operate on sequential data rather than independent observations, the multivariate time series is transformed into supervised learning samples using a sliding-window approach. For a sequence length L, each input sample consists of a sequence of consecutive observations from x_{t-L+1} to x_t, while the corresponding target is the next observation y_{t+1}. This transformation produces overlapping input–output pairs that are appropriate for training recurrent learning models and allows the networks to learn temporal relationships across successive time steps.

2.3 Benchmark Models

To provide a comprehensive evaluation of recurrent neural network architectures for sequential prediction, six representative recurrent models with different structural characteristics and memory mechanisms were selected for benchmarking. The chosen architectures include the conventional Simple Recurrent Neural Network (SimpleRNN), Independently Recurrent Neural Network (IndRNN), Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), Bidirectional

Long Short-Term Memory (BiLSTM), and Residual Bidirectional Long Short-Term Memory (ResBiLSTM). These models were selected because they represent the major evolutionary stages of recurrent learning, ranging from basic recurrent structures to advanced gated, bidirectional, and residual architectures. All models were implemented using the same deep learning framework and trained under identical preprocessing, optimization, and evaluation conditions to ensure a fair comparison. The objective of this benchmark is to investigate the ability of each architecture to capture nonlinear temporal dependencies, retain historical information, and provide accurate forecasts for multivariate air pollution time-series data. The following subsections briefly describe the theoretical foundations and structural characteristics of each recurrent model used in this study.

2.3.1 Recurrent Neural Network

The Simple Recurrent Neural Network (SimpleRNN), also referred to as the vanilla RNN, is the fundamental recurrent architecture designed for processing sequential data. Unlike feedforward neural networks, SimpleRNN introduces recurrent connections that allow information from previous time steps to influence the current hidden state. This recurrent structure enables the network to model temporal dependencies in time-series sequences and sequential observations. In a sequential prediction problem, let the input sequence be represented as

$$X = \{x_1, x_2, \ldots, x_T\}$$

where $x_t \in \mathbb{R}^d$ denotes the input vector at time step t, d is the number of input features, and T is the sequence length. The SimpleRNN maintains a hidden state h_t that is updated recursively according to the current input and the previous hidden state. The hidden-state update is defined as

$$h_t = \phi(W_{xh}x_t + W_{hh}h_{t-1} + b_h)$$

where $W_{xh} \in \mathbb{R}^{m\times d}$ is the input-to-hidden weight matrix, $W_{hh} \in \mathbb{R}^{m\times m}$ is the recurrent hidden-to-hidden weight matrix, $b_h \in \mathbb{R}^m$ is the hidden bias vector, m is the number of hidden units, $\phi(\cdot)$ is the activation function, typically the hyperbolic tangent function (tanh). The hyperbolic tangent activation is expressed as

$$\phi(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

which constrains the hidden-state values to the interval $(-1, 1)$, helping to stabilize the learning process. The output of the network at time step t is computed from the hidden state as $y_t = W_{hy}h_t + b_y$ where $W_{hy} \in \mathbb{R}^{k\times m}$ is the hidden-to-output weight matrix, $b_y \in \mathbb{R}^k$ is the output bias vector, k is the output dimension. For regression-based forecasting tasks such as air pollution prediction, the output layer usually employs a linear activation function, and the predicted value for the next time step is given by $\hat{y}_{t+1} = W_{hy}h_t + b_y$. The recursive nature of SimpleRNN can be expanded to show its dependence on all previous inputs. By repeatedly substituting the hidden-state equation, the hidden state at time step t becomes

$$h_t = \phi(W_{xh}x_t + W_{hh}\phi(W_{xh}x_{t-1} + W_{hh}h_{t-2} + b_h) + b_h).$$

This expression illustrates that the current hidden representation theoretically contains information from the entire history of the sequence. Therefore, the network can model temporal

relationships between current observations and past inputs. For a sequence of length T, the overall recurrent computation proceeds as follows:

$$h_1 = \phi(W_{xh}x_1 + W_{hh}h_0 + b_h)$$

$$h_2 = \phi(W_{xh}x_2 + W_{hh}h_1 + b_h)$$

$$\vdots$$

$$h_T = \phi(W_{xh}x_T + W_{hh}h_{T-1} + b_h).$$

The initial hidden state h_0 is commonly initialized to a zero vector, $h_0 = 0$. During training, the network parameters $\Theta = \{W_{xh}, W_{hh}, W_{hy}, b_h, b_y\}$ are optimized by minimizing a loss function over all training sequences. For regression problems, the MSE loss is commonly used:

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

where N is the number of training samples, y_i is the observed target value, and $\hat{y}_i$ is the predicted value produced by the network. The gradients of the loss with respect to the recurrent parameters are computed using Backpropagation Through Time (BPTT). Because the same recurrent weight matrix W_{hh} is reused at every time step, the gradient involves repeated multiplication by this matrix:

$$\frac{\partial \mathcal{L}}{\partial h_t} = \frac{\partial \mathcal{L}}{\partial h_{t+1}} W_{hh}^T \operatorname{diag}(\phi'(a_t))$$

where

$$a_t = W_{xh}x_t + W_{hh}h_{t-1} + b_h.$$

When the eigenvalues of W_{hh} are smaller than one, repeated multiplication causes the gradients to decay exponentially, leading to the vanishing gradient problem. Conversely, eigenvalues larger than one may produce exploding gradients. These issues make it difficult for SimpleRNN to learn long-term dependencies effectively.

Despite these limitations, SimpleRNN remains an important baseline architecture for sequential forecasting research because of its conceptual simplicity and relatively low computational cost. In this study, the SimpleRNN model serves as the fundamental recurrent benchmark against which more advanced architectures such as LSTM, GRU, BiLSTM, ResBiLSTM, and IndRNN are compared. The experimental results provide insight into the extent to which modern recurrent mechanisms improve the modeling of complex temporal dependencies in multivariate air pollution time-series data.

2.3.2 Independently Recurrent Neural Network

The Independently Recurrent Neural Network (IndRNN) is an advanced recurrent architecture proposed to overcome the limitations of conventional RNNs, particularly the vanishing and exploding gradient problems that arise when learning long-term temporal dependencies. Unlike SimpleRNN, where neurons within the same hidden layer are recurrently connected through a full recurrent weight matrix, IndRNN assigns an independent recurrent connection to each neuron. This structural modification significantly simplifies gradient propagation and enables

deeper and more stable recurrent networks. For a hidden layer containing m neurons, the hidden state of the j-th neuron at time step t is computed as

$$h_{t,j} = \phi(w_j^T x_t + u_j h_{t-1,j} + b_j)$$

where $w_j \in \mathbb{R}^d$ is the input weight vector for neuron j, u_j is the independent recurrent weight associated with neuron j, $h_{t-1,j}$ is the previous hidden state of the same neuron, b_j is the bias term, $\phi(\cdot)$ is the activation function, typically ReLU or tanh. In vector form, the hidden-state update for the entire layer can be written as

$$h_t = \phi(W x_t + u \odot h_{t-1} + b)$$

where $W \in \mathbb{R}^{m \times d}$ is the input weight matrix, $u \in \mathbb{R}^m$ is the vector of independent recurrent weights, $\odot$ denotes element-wise multiplication, $b \in \mathbb{R}^m$ is the bias vector. This formulation highlights the key difference between SimpleRNN and IndRNN. In SimpleRNN, the recurrent term is $W_{hh} h_{t-1}$ which mixes information across all hidden neurons through a dense recurrent matrix. In contrast, IndRNN replaces this operation with $u \odot h_{t-1}$ so that each neuron depends only on its own previous state. Consequently, neurons within a layer are independent in the recurrent dimension, while interactions between neurons are learned through the feedforward weight matrix W. For sequential forecasting, the output layer is computed as $\hat{y}_t = W_{hy} h_t + b_y$ where W_{hy} is the hidden-to-output weight matrix and b_y is the output bias vector. Expanding the recurrence over multiple time steps gives

$$h_{t,j} = \phi(w_j^T x_t + u_j \phi(w_j^T x_{t-1} + u_j h_{t-2,j} + b_j) + b_j).$$

Assuming a linear activation for analytical purposes, the hidden state can be approximated as

$$h_{t,j} \approx \sum_{k=0}^{t-1} u_j^k (w_j^T x_{t-k}) + u_j^t h_{0,j}.$$

This expression reveals that the influence of previous inputs is controlled solely by the scalar recurrent weight u_j. The gradient of the loss with respect to a hidden state becomes

$$\frac{\partial \mathcal{L}}{\partial h_{t,j}} = \frac{\partial \mathcal{L}}{\partial h_{T,j}} \prod_{k=t+1}^{T} u_j \phi'(a_{k,j})$$

where $a_{k,j} = w_j^T x_k + u_j h_{k-1,j} + b_j$. Because the recurrent coefficient is a single scalar rather than a full matrix, the gradient behavior is much easier to analyze and control. By constraining the magnitude of u_j, IndRNN can effectively avoid both gradient explosion and gradient vanishing over long sequences. This property allows IndRNN to process substantially longer temporal sequences than conventional RNNs while maintaining stable training dynamics. Another important advantage of IndRNN is its compatibility with deep recurrent architectures. Multiple IndRNN layers can be stacked as

$$h_t^{(l)} = \phi(W^{(l)} h_t^{(l-1)} + u^{(l)} \odot h_{t-1}^{(l)} + b^{(l)})$$

where l denotes the layer index. Since recurrent dependencies are independent within each layer, stacking does not introduce the severe optimization difficulties commonly encountered in deep traditional RNNs.

The computational complexity of an IndRNN layer is also lower than that of a standard RNN. A conventional RNN requires matrix multiplication involving the recurrent weight matrix $W_{hh} \in$

$\mathbb{R}^{m \times m}$, resulting in O(m²) recurrent computations per time step. IndRNN requires only element-wise multiplication with the recurrent vector u, reducing the recurrent computation to O(m). This reduction improves computational efficiency and scalability for large hidden dimensions. In this study, the IndRNN architecture is employed as an advanced recurrent benchmark for sequential air pollution forecasting. Its independently recurrent structure enables efficient learning of nonlinear temporal dependencies while maintaining stable gradient propagation across different sequence lengths.

2.3.3 Long Short-Term Memory

The Long Short-Term Memory (LSTM) network is an enhanced recurrent neural network architecture designed to overcome the vanishing gradient problem of conventional RNNs and to capture long-term temporal dependencies more effectively. LSTM introduces a memory cell and three gating mechanisms that regulate the flow of information through the network. For an input vector x_t, previous hidden state h_{t-1}, and previous cell state c_{t-1}, the LSTM computations are defined as follows:

Forget gate:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$$

Input gate:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$

Candidate cell state:

$$\tilde{c}_t = \tanh(W_c x_t + U_c ht - 1 + b_c)$$

Cell state update:

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

Output gate:

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$$

Hidden state:

$$h_t = o_t \odot \tanh(c_t)$$

where $\sigma(\cdot)$ denotes the sigmoid activation function, $\tanh(\cdot)$ is the hyperbolic tangent function, and $\odot$ represents element-wise multiplication. The forget gate determines which information from the previous cell state should be retained, the input gate controls the incorporation of new information, and the output gate determines which part of the updated memory is exposed as the hidden representation. This gating mechanism allows LSTM to preserve relevant information over long sequences while reducing the effect of vanishing gradients.

2.3.4 Gated Recurrent Unit

The Gated Recurrent Unit (GRU) is a simplified variant of the LSTM architecture that was developed to improve computational efficiency while preserving the ability to learn long-term temporal dependencies. Unlike LSTM, GRU combines the cell state and hidden state into a single hidden representation and employs only two gating mechanisms: the update gate and the reset gate. This simpler structure reduces the number of trainable parameters and often leads

to faster training and inference. For an input vector x_t and the previous hidden state h_{t-1}, the GRU computations are given by:

Update gate

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z)$$

Reset gate

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r)$$

Candidate hidden state

$$\tilde{h}_t = \tanh(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h)$$

Final hidden state

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$$

The reset gate controls how much information from the previous hidden state should be forgotten when computing the candidate hidden state, while the update gate determines the proportion of previous information that is retained versus the amount of new information that is incorporated into the current hidden state. This adaptive mechanism enables GRU to capture both short-term and long-term dependencies efficiently. Compared with LSTM, GRU has fewer parameters because it does not maintain a separate memory cell or output gate. Consequently, it generally requires less memory and shorter processing time, making it particularly suitable for real-time sequential forecasting applications.

2.3.5 Bidirectional Long Short-Term Memory

The Bidirectional Long Short-Term Memory (BiLSTM) network extends the conventional LSTM architecture by processing the input sequence in both forward and backward directions. While a standard LSTM only utilizes past observations to predict the current output, BiLSTM can exploit information from both previous and subsequent time steps, enabling the network to learn richer temporal representations. This characteristic is particularly useful for complex sequential patterns where contextual information from the entire sequence contributes to the prediction task. Given an input sequence $X = \{x_1, x_2, \ldots, x_T\}$, two independent LSTM layers are employed. The forward LSTM processes the sequence from $t = 1$ to T:

$$\overrightarrow{h_t} = \text{LSTM}f(x_t, \overrightarrow{h_{t-1}}, \overrightarrow{c_{t-1}})$$

The backward LSTM processes the sequence in reverse order:

$$\overleftarrow{h_t} = \text{LSTM}b(x_t, \overleftarrow{h_{t+1}}, \overleftarrow{c_{t+1}})$$

The final hidden representation is obtained by combining the forward and backward hidden states, typically through concatenation:

$$h_t = [\overrightarrow{h_t}; \overleftarrow{h_t}]$$

where $[\cdot;\cdot]$ denotes vector concatenation. The output layer then computes the prediction as

$$\hat{y}_t = W_y h_t + b_y$$

where W_y is the output weight matrix and b_y is the bias vector. By incorporating information from both temporal directions, BiLSTM can capture more comprehensive dependencies than unidirectional LSTM, often leading to improved prediction accuracy. However, this advantage comes at the cost of increased computational complexity and longer processing time, since two LSTM networks must be trained simultaneously.

2.3.6 Residual Bidirectional Long Short-Term Memory

The Residual Bidirectional Long Short-Term Memory (ResBiLSTM) network combines the advantages of Bidirectional LSTM (BiLSTM) and residual learning to improve the training of deeper recurrent architectures. Although BiLSTM can capture contextual information from both forward and backward temporal directions, stacking multiple bidirectional recurrent layers often leads to optimization difficulties, slower convergence, and degradation of performance. Residual connections alleviate these issues by providing a direct pathway for information and gradient propagation across layers. Let $h_t^{(l-1)}$ denote the input representation to the l-th BiLSTM layer. The bidirectional recurrent transformation is first computed as

$$\tilde{h}_t^{(l)} = \text{BiLSTM}^{(l)}(h_t^{(l-1)})$$

where $\tilde{h}_t^{(l)}$ is the output produced by the forward and backward LSTM units at layer l. A residual connection is then introduced by adding the input of the layer directly to its output:

$$h_t^{(l)} = \tilde{h}_t^{(l)} + h_t^{(l-1)}$$

This skip connection allows the network to learn only the residual mapping

$$\mathcal{F}(h_t^{(l-1)}) = \tilde{h}_t^{(l)} - h_t^{(l-1)}$$

instead of learning the entire transformation from scratch. Consequently, the updated representation can be expressed as

$$h_t^{(l)} = h_t^{(l-1)} + \mathcal{F}(h_t^{(l-1)}).$$

The residual formulation offers two important advantages. First, it facilitates gradient flow during backpropagation, reducing the risk of vanishing gradients when multiple recurrent layers are stacked. Second, it preserves useful low-level temporal features from earlier layers while allowing deeper layers to focus on learning complementary high-level representations. For the final prediction layer, the output sequence representation is transformed into the forecast value as

$$\hat{y}_t = W_y h_t^{(L)} + b_y$$

where $h_t^{(L)}$ is the output of the final residual bidirectional layer, W_y is the output weight matrix, and b_y is the bias vector. Compared with standard BiLSTM, ResBiLSTM is capable of modeling more complex temporal relationships by exploiting both bidirectional context and deep residual learning. However, the additional bidirectional recurrent computations and residual transformations increase the number of trainable parameters and computational cost. Therefore, ResBiLSTM is expected to provide improved representational capacity at the expense of longer training and inference times.

3 Experimental Setup

To ensure a fair and reproducible comparison, all forecasting models were trained and evaluated under identical experimental conditions. The same dataset, preprocessing pipeline, feature engineering procedure, training–testing partition, optimization strategy, and evaluation metrics were applied to every model. This unified experimental framework minimizes the influence of external

factors and enables an objective assessment of the relative performance of different machine learning and recurrent neural network architectures. The dataset was divided chronologically into training and testing subsets to preserve the temporal dependencies inherent in time-series data. The first 80% of the observations were used for model training and the remaining 20% for independent testing. Random shuffling was deliberately avoided to prevent information leakage from future observations into the training process.

Recurrent neural networks require sequential inputs rather than independent observations. A sliding-window approach was employed to generate input sequences from the multivariate time series. Each input sample consisted of the observations from the previous L time steps, while the target corresponded to the pollutant concentration at the next time step. To do this, the sequence lengths of 7, 14, and 28 time steps were selected to examine the ability of the RNN architectures to capture temporal dependencies over short-, medium-, and longer-term historical windows. Because the observations are recorded daily, these sequence lengths correspond to 7, 14, and 28 days of historical information, respectively. Thus, the selected windows provide a progressively broader temporal context while allowing the effect of increasing historical information on predictive performance to be systematically assessed. These values were chosen as representative temporal windows rather than as an exhaustive hyperparameter search over all possible sequence lengths.

To ensure reproducibility, all recurrent neural network architectures were trained using a consistent optimization and training protocol. The models were optimized using the Adam optimizer with its default learning rate of (0.001), and mean squared error (MSE) was used as the loss function, with mean absolute error (MAE) additionally monitored as an evaluation metric. For the recurrent layers, the hyperbolic tangent (tanh) activation function was employed, while a fully connected layer with 32 neurons and ReLU activation was used before the final output layer. The models were trained for a maximum of 200 epochs with a batch size of 32. To reduce overfitting and avoid unnecessary training after convergence, early stopping was applied based on the validation loss. Specifically, training was terminated when the validation loss did not improve for 10 consecutive epochs (patience = 10). Model performance is evaluated using three widely adopted regression metrics, mean absolute error (MAE), root mean square error (RMSE), and coefficient of determination (R^2). These metrics jointly assess prediction accuracy, error magnitude, and goodness of fit. All experiments are implemented in Python using Scikit-learn, TensorFlow, Keras, NumPy, Pandas, and LightGBM libraries. Training is performed on the same computing platform to ensure fair comparisons of computational efficiency among all investigated models.

4 Results

4.1 Comparative Performance for Different Sequence Lengths

The predictive performance of the investigated RNN architectures for sequence lengths of 7, 14, and 28 time steps are summarized in Tables 1–3. The results indicate that advanced recurrent architectures substantially outperform the conventional SimpleRNN model across all sequence lengths. Among the evaluated models, IndRNN consistently achieves the best overall predictive performance, demonstrating superior accuracy and robust generalization capability while maintaining moderate computational cost. For the shortest sequence length (Seq = 7),

IndRNN achieved the best performance with an MAE of 12.0001, an RMSE of 17.0203, and an R^2 value of 0.6375. This represents the highest explanatory power and the lowest prediction error among all evaluated architectures. BiLSTM produced the second-best R^2 value (0.6336) with an RMSE of 17.1359, followed by GRU ($R^2 = 0.6247$, RMSE = 17.3427). In contrast, SimpleRNN showed the weakest performance, yielding an RMSE of 18.3115 and an R^2 of 0.5816, indicating limited ability to capture the complex temporal dependencies present in the air pollution time series. Although ResBiLSTM achieved competitive accuracy (MAE = 12.0178, RMSE = 17.1445), its processing time (115.63 s) was substantially higher than that of IndRNN (35.15 s), suggesting a less favorable balance between accuracy and computational efficiency.

Increasing the sequence length to 14 time steps did not lead to significant performance improvements for most models. IndRNN again obtained the best overall results, achieving an MAE of 12.0081, an RMSE of 17.0455, and an R^2 of 0.6357. These values are very close to those obtained for Seq = 7, demonstrating the stability of the IndRNN architecture across different temporal contexts. GRU and ResBiLSTM achieved similar performance levels, with R^2 values of 0.6109 and 0.6091, respectively. However, ResBiLSTM required 257.13 seconds of processing time, which was more than seven times higher than GRU (35.72 s) and considerably higher than IndRNN (49.12 s). BiLSTM showed a slight degradation in performance compared with Seq = 7, while SimpleRNN remained the least accurate model with an RMSE of 18.4663 and an R^2 of 0.5724.

For the longest sequence length (Seq = 28), IndRNN continued to provide the best predictive performance, achieving an MAE of 12.1637, an RMSE of 17.0463, and an R^2 of 0.6272. GRU showed highly competitive results, with an RMSE of 17.0636 and an R^2 of 0.6204, indicating that GRU remains a strong alternative for longer input sequences. LSTM and BiLSTM exhibited slightly lower predictive accuracy, with RMSE values of 17.9616 and 17.8434, respectively. ResBiLSTM achieved moderate performance ($R^2 = 0.6142$), but its processing time increased dramatically to 432.38 seconds, making it the most computationally expensive architecture in the benchmark. The substantial increase in computational cost was not accompanied by a corresponding improvement in prediction accuracy.

Table 1: Model Performance Comparison for Sequence Length = 7

Model	**MAE**	**RMSE**	**R^2**	**Processing Time (s)**
SimpleRNN	13.0628	18.3115	0.5816	17.07
IndRNN	**12.0001**	**17.0203**	**0.6375**	35.15
LSTM	12.2221	17.4613	0.6133	49.05
GRU	12.2133	17.3427	0.6247	25.11
BiLSTM	12.2261	17.1359	0.6336	33.10
ResBiLSTM	12.0178	17.1445	0.6189	115.63

Table 2: Model Performance Comparison for Sequence Length = 14

Model	**MAE**	**RMSE**	**R^2**	**Processing Time (s)**
SimpleRNN	13.3757	18.4663	0.5724	27.60
IndRNN	**12.0081**	**17.0455**	**0.6357**	49.12
LSTM	12.4911	17.8321	0.6077	79.56
GRU	12.5966	17.6169	0.6109	35.72
BiLSTM	12.3924	17.8375	0.6011	86.50
ResBiLSTM	12.2111	17.4766	0.6091	257.13

Table 3: Model Performance Comparison for Sequence Length = 28

Model	**MAE**	**RMSE**	**R^2**	**Processing Time (s)**
SimpleRNN	13.8777	18.9215	0.5752	27.82
IndRNN	**12.1637**	**17.0463**	**0.6272**	37.59
LSTM	12.4214	17.9616	0.5934	74.65
GRU	12.2553	17.0636	0.6204	45.48
BiLSTM	12.2646	17.8434	0.5987	110.89
ResBiLSTM	12.1841	17.4950	0.6142	432.38

A comparison across the three sequence lengths reveals that increasing the historical input window beyond seven time steps does not necessarily improve forecasting accuracy. For most models, the best or near-best performance was obtained with Seq = 7. In particular, IndRNN achieved its highest R^2 (0.6375) and lowest RMSE (17.0203) using the shortest sequence length. This suggests that the most informative temporal dependencies for the considered air pollution forecasting task are concentrated within relatively recent observations, and that longer sequences may introduce redundant or noisy information that does not contribute substantially to predictive performance. The results also show that processing time generally increases with sequence length for all recurrent architectures. This increase is especially pronounced for BiLSTM and ResBiLSTM, whose bidirectional and residual structures require substantially greater computational resources as the sequence length grows.

Considering all evaluation metrics and sequence lengths together, IndRNN demonstrates the most consistent and reliable performance across the entire benchmark. It achieves the highest R^2 values, while also maintaining the lowest RMSE values in all three experimental settings. Although GRU and BiLSTM produce competitive results, their predictive accuracy remains slightly inferior to that of IndRNN. From a practical perspective, IndRNN offers an effective trade-off between forecasting accuracy and computational efficiency. While its processing time is moderately higher than that of SimpleRNN and GRU, it remains substantially lower than the computational cost required by the more complex ResBiLSTM architecture. These findings indicate that IndRNN is particularly well suited for real-world sequential air pollution forecasting applications where both prediction accuracy and computational efficiency are important considerations.

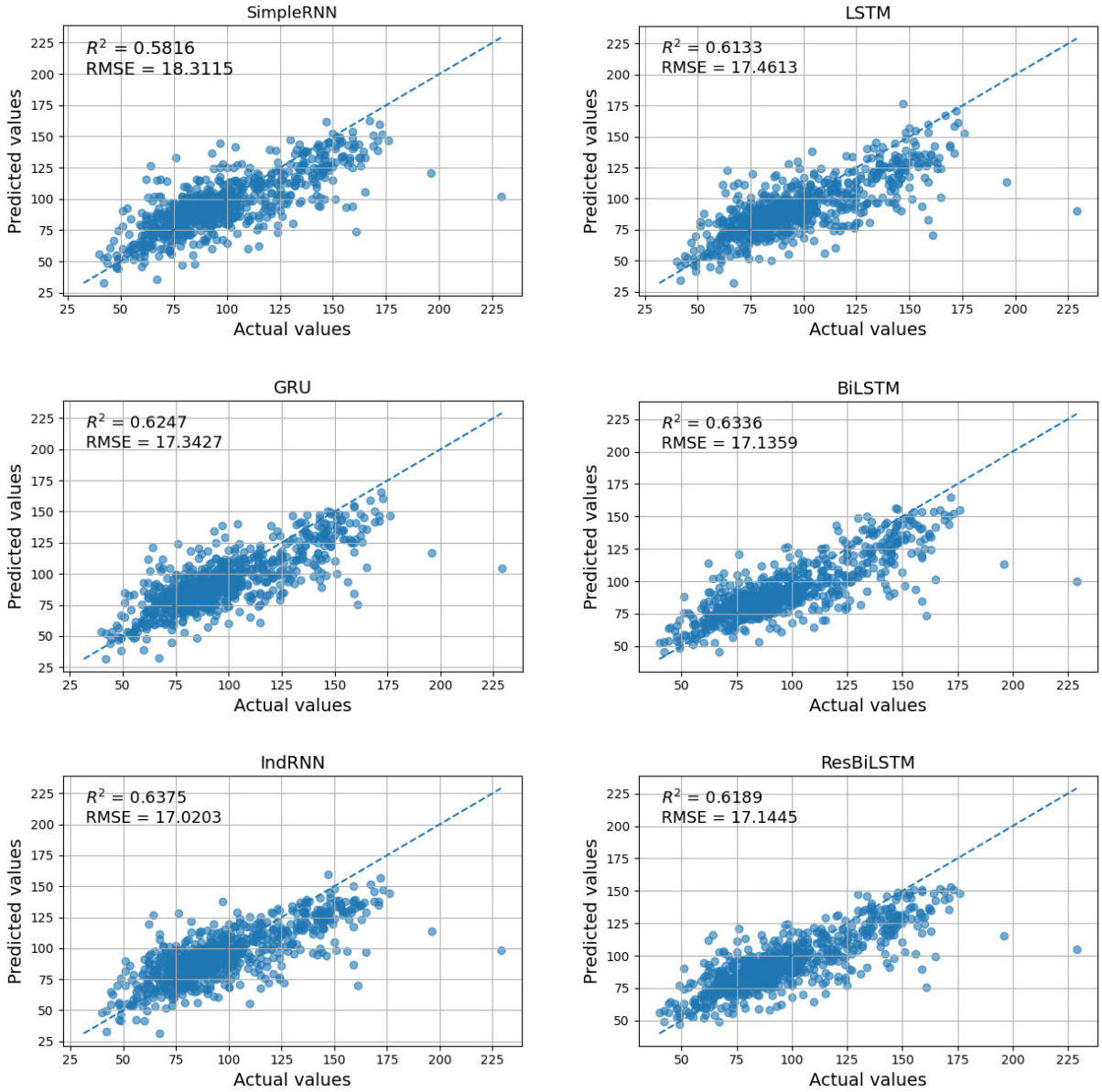


Figure 1: Scatter plots of the actual and predicted pollutant concentrations

4.2 Actual versus Predicted Values

The scatter plots of the actual and predicted pollutant concentrations for the RNN architectures evaluated in this study are presented in Figure 1. The dashed diagonal line represents the ideal one-to-one relationship between observed and predicted values. Therefore, points located closer to this line indicate higher predictive accuracy and better agreement between model outputs and the ground-truth observations.

The SimpleRNN model exhibited the weakest predictive performance among the investigated architectures, with an R^2 value of 0.5816 and an RMSE of 18.3115. The corresponding scatter plot shows a relatively large dispersion of points around the diagonal line, particularly for higher pollutant concentrations, indicating limited capability in capturing complex temporal dependencies and extreme pollution events. The IndRNN model improved the predictive

performance substantially, achieving an R^2 of 0.6375 and an RMSE of 17.0203. The predicted values are more tightly clustered around the ideal line compared with SimpleRNN, suggesting that the independent recurrent structure enhances the learning of long-term temporal patterns in the air pollution time series.

The LSTM architecture achieved an R^2 of 0.6133 with an RMSE of 17.4613. Although the model demonstrated a reasonable ability to follow the overall trend of the observed data, noticeable prediction errors remained for both low and high concentration ranges. The gating mechanism of LSTM improved temporal modeling compared with SimpleRNN, but its performance was inferior to several of the more advanced recurrent architectures. The GRU model produced one of the best overall results, with an R^2 of 0.6247 and an RMSE of 17.3427. As illustrated in Figure 1, the majority of the predicted values are concentrated near the diagonal reference line, indicating strong agreement with the observed measurements. The GRU architecture achieved a favorable balance between prediction accuracy and model complexity, demonstrating its effectiveness for sequential air pollution forecasting. The BiLSTM model achieved competitive performance, yielding an R^2 of 0.6336 and an RMSE of 17.1359. The bidirectional processing of temporal information allowed the model to capture richer contextual dependencies, resulting in a tighter distribution of points around the ideal prediction line and improved robustness across different concentration levels. The Residual BiLSTM architecture obtained an R^2 of 0.6189 and an RMSE of 17.1445. The residual connections contributed to stable learning and reduced prediction variance; however, the performance gain over the standard BiLSTM was relatively modest.

Overall, the actual-versus-predicted analysis confirms that advanced recurrent architectures provide substantially better predictive capability than the conventional SimpleRNN model. Among the evaluated networks, IndRNN achieved the highest R^2 value and demonstrated highly competitive accuracy with a comparatively simple architecture. The visual proximity of the IndRNN predictions to the ideal one-to-one line, together with its strong quantitative performance, supports the conclusion that IndRNN represents the most effective and practically efficient recurrent architecture for the considered air pollution forecasting task.

5 Discussion

This study presented a comprehensive benchmark of state-of-the-art RNN architectures for sequential air pollution prediction using real-world environmental and meteorological data. The experimental results demonstrate IndRNN architecture achieved superior performance across all experimental settings. The best overall performance was obtained for Seq = 7, where IndRNN achieved $R^2 = 0.6375$ and RMSE = 17.0203. The consistency of these results across different sequence lengths suggests that the independent recurrent structure is highly effective in learning both short-term and moderately long-term temporal patterns in air pollution data. The superior performance of IndRNN can be attributed to its independently recurrent hidden units, which mitigate the gradient vanishing and exploding problems commonly encountered in conventional RNN architectures. By decoupling recurrent connections across neurons, IndRNN facilitates more stable gradient propagation through time, enabling the network to capture nonlinear temporal dependencies more effectively than standard RNN, LSTM, and even some bidirectional recurrent architectures. This finding is consistent with recent studies reporting that simplified recurrent structures with improved gradient flow can achieve competitive or superior forecasting performance while maintaining lower architectural complexity.

LSTM and GRU are generally regarded as the dominant recurrent architectures for time-series forecasting because of their gating mechanisms for controlling information flow. In the present study, both architectures achieved substantially better performance than SimpleRNN, confirming their ability to model temporal dependencies in air pollution sequences. However, neither architecture consistently outperformed IndRNN. GRU produced competitive results, particularly for the longer sequence length (Seq = 28), where its RMSE (17.0636) was very close to that of IndRNN (17.0463). The relatively small performance gap suggests that GRU remains a strong alternative for sequential forecasting tasks, especially when computational efficiency is a primary concern. GRU also required less processing time than LSTM for all sequence lengths, which agrees with the theoretical advantage of its simpler gating structure. Nevertheless, the consistently higher R^2 values achieved by IndRNN indicate that the independent recurrent mechanism provides additional representational benefits beyond the conventional update and reset gates employed by GRU. LSTM exhibited stable but comparatively weaker performance, with RMSE values ranging from 17.46 to 17.96 and R^2 values between 0.5934 and 0.6133. Although LSTM successfully captured the overall temporal trends, its larger number of parameters and more complex memory structure may have reduced its ability to generalize effectively on the available dataset.

BiLSTM achieved the second-highest R^2 value for Seq = 7 (0.6336), demonstrating that bidirectional processing can improve the extraction of contextual temporal information. By incorporating both forward and backward dependencies during training, BiLSTM is able to learn richer sequence representations than unidirectional recurrent networks. However, this improvement came at the cost of increased computational complexity, and its performance deteriorated slightly as the sequence length increased. The Residual BiLSTM architecture did not provide substantial accuracy improvements despite its significantly higher computational cost. For Seq = 28, ResBiLSTM required more than 432 seconds of processing time, which was nearly twelve times higher than GRU and more than eleven times higher than IndRNN. The modest gain in predictive accuracy did not justify this dramatic increase in computational expense, suggesting that residual bidirectional structures may be unnecessarily complex for the considered air pollution forecasting task.

An important observation from the benchmark is that increasing the sequence length beyond seven time steps did not improve predictive performance. For most architectures, the best results were obtained with Seq = 7, while longer sequences either produced negligible improvements or slightly degraded the forecasting accuracy. This finding indicates that the dominant predictive information in the studied air pollution series is concentrated within relatively recent observations. Air pollutant concentrations are strongly influenced by short-term meteorological conditions, local emission sources, and recent atmospheric interactions, which may reduce the contribution of distant historical observations. Longer sequences can introduce redundant or noisy information, increasing the difficulty of optimization and potentially reducing generalization performance. The effect of sequence length was also evident in the computational results. Processing time increased for all recurrent architectures as the input window became longer, with the increase being particularly severe for BiLSTM and ResBiLSTM. These results highlight the importance of selecting an appropriate sequence length rather than assuming that longer historical windows necessarily lead to better forecasting performance.

The scatter plots of actual versus predicted values further support the quantitative results. IndRNN exhibited the tightest clustering of points around the ideal reference line, indicating the highest agreement between predicted and observed pollutant concentrations. The reduced dispersion across both low and high concentration ranges suggests that IndRNN is capable of modeling not only the average temporal behavior but also more extreme pollution episodes.

In contrast, SimpleRNN showed the largest spread of prediction errors, particularly for higher pollutant concentrations, confirming its limited capacity for learning complex temporal dynamics. GRU and BiLSTM produced reasonably concentrated prediction distributions, but their scatter plots exhibited slightly greater deviations from the ideal line than those observed for IndRNN.

From an operational perspective, forecasting models for environmental monitoring systems must satisfy two often conflicting requirements: high predictive accuracy and reasonable computational efficiency. The results of this study suggest that IndRNN provides an effective compromise between these objectives. Although its processing time is moderately higher than that of GRU, it remains substantially lower than the computational cost associated with more complex bidirectional and residual architectures while consistently delivering the best predictive performance. These characteristics make IndRNN particularly attractive for real-time or near-real-time air quality forecasting systems, where models must be updated frequently as new monitoring data become available. Accurate short-term forecasts can support early warning systems, traffic and emission management strategies, and public health interventions aimed at reducing exposure to hazardous pollution events. For practitioners, model selection should nevertheless be based on task-specific validation rather than on architecture type alone. A suitable operational workflow should first establish a chronological evaluation scheme that reflects the intended forecasting setting, followed by comparison of candidate architectures using both predictive metrics and computational requirements. Where rapid deployment, limited computational resources, or model simplicity are priorities, computationally less demanding architectures such as SimpleRNN or GRU may remain practical alternatives if their predictive performance is adequate for the application. Conversely, where predictive accuracy is prioritized and sufficient computational resources are available, IndRNN may be considered a strong candidate based on the results obtained in this study. Importantly, the selected architecture should be re-evaluated using local and temporally independent data before operational deployment, because model performance and architecture ranking may change across locations, pollutants, and forecasting periods.

Although the present study focuses on short-term PM2.5 forecasting, the benchmarking framework is not inherently restricted to this pollutant. The same recurrent architectures can be applied to other air pollutants, such as CO, O3, NO2, SO2, and PM10, provided that appropriate target-specific preprocessing, temporal features, and forecasting horizons are defined. However, the relative performance of the investigated RNN architectures should not be assumed to remain identical across pollutants, because different pollutants exhibit distinct temporal dynamics, emission sources, atmospheric lifetimes, and relationships with meteorological conditions. Therefore, the superior performance of IndRNN observed in this study should be interpreted as evidence for its effectiveness for the present PM2.5 forecasting task rather than as a universal ranking across all pollutants.

Similarly, the applicability of these findings to other climatic regions requires empirical validation. Atmospheric stability, temperature and humidity regimes, wind characteristics, emission patterns, and seasonal variability can differ substantially among climatic and geographic settings and may alter the temporal dependencies that recurrent models need to learn. Consequently, the proposed benchmarking framework is potentially transferable to other monitoring locations and climatic regions, but the reported performance levels and the relative ranking of the architectures may vary. Multi-site and multi-climate evaluations involving diverse pollutant targets would therefore be an important direction for future research to assess the robustness and geographical generalizability of the observed findings.

Several limitations should be considered when interpreting the findings of this study. The analysis was conducted using a single air-quality dataset from one monitoring setting and therefore

does not establish that the observed performance differences among the recurrent architectures are universally applicable. The geographic and climatic characteristics of the study area, as well as its local emission patterns and meteorological conditions, may influence the temporal structure of PM2.5 concentrations and consequently the relative performance of the evaluated models. In addition, the analysis covers a specific observation period; therefore, changes in emission sources, meteorological regimes, or long-term pollution patterns may affect model performance when applied to other time periods.

Although this study focuses on air quality forecasting, the benchmarking results may also provide broader methodological guidance for other sequential prediction problems in which temporal dependencies must be learned from ordered observations. The comparison of recurrent architectures highlights the importance of considering the characteristics of the temporal dynamics, the length of the relevant historical window, predictive performance, and computational complexity when selecting a sequential model. In particular, the finding that increasing the sequence length did not consistently improve forecasting performance suggests that longer historical windows should not be assumed to be beneficial without empirical validation. Similarly, the strong performance of IndRNN in the present application indicates that architectures designed to facilitate stable learning of temporal dependencies may be useful candidates in other sequential prediction settings. However, these implications should be interpreted as general methodological guidance rather than evidence that IndRNN will consistently outperform alternative architectures in other domains. Sequential datasets from fields such as energy forecasting, traffic prediction, environmental monitoring, industrial process monitoring, and biomedical time-series analysis may exhibit substantially different temporal structures and noise characteristics. Therefore, architecture selection in these domains should be based on systematic, task-specific benchmarking using temporally appropriate validation and consideration of both predictive accuracy and computational requirements.

6 Conclusion

This study presented a comprehensive benchmarking analysis of state-of-the-art recurrent neural network architectures for sequential prediction using a real-world air pollution forecasting task. A unified experimental framework was developed to ensure a fair comparison among conventional recurrent models and advanced recurrent architectures. The experimental results demonstrated that advanced recurrent architectures substantially outperform the conventional SimpleRNN model, confirming the importance of effective memory and temporal representation mechanisms for modeling complex multivariate environmental time-series data. Among all evaluated architectures, IndRNN consistently achieved the best overall predictive performance across all sequence lengths. The comparative analysis also revealed that increasing the sequence length beyond seven time steps did not improve forecasting performance for the considered dataset. In many cases, longer input windows slightly degraded prediction accuracy while substantially increasing computational cost. This finding suggests that recent historical observations contain the most informative temporal patterns for short-term air pollution forecasting. In practical terms, the findings of this study suggest that IndRNN is a strong candidate for real-time and operational air quality forecasting systems, where both accuracy and computational efficiency are critical requirements. The proposed benchmarking framework also provides a reproducible basis for future comparative studies on sequential prediction models.

Declarations

Author Contributions

Conceptualization: Pak, Nematollahi; Methodology: Pak; Software and Validation: Pak; Formal Analysis: Pak; Investigation: Nematollahi; Resources: Nematollahi; Data Curation: Nematollahi; Writing – Original Draft: Pak; Writing – Review & Editing: All authors; Visualization: Pak.

Acknowledgments

The authors received no specific funding for this research. The authors would like to thank all colleagues and reviewers whose constructive comments significantly improved the quality of this manuscript.

Conflict of Interest

The authors declare that they have no known competing financial interests or personal relationships that could have influenced the work reported in this paper.

Artificial Intelligence Statement

Artificial intelligence (AI) tools, including large language models, were used solely for language editing and improving readability. AI tools were not used for generating ideas, performing analyses, interpreting results, or writing the scientific content. All scientific conclusions and intellectual contributions were made exclusively by the authors.

Publisher's Note

The publisher remains neutral regarding jurisdictional claims in published maps and institutional affiliations.

References

[1] Benidis, K., Rangapuram, S. S., Flunkert, V., et al. (2022). Deep learning for time series forecasting: tutorial and literature survey. *ACM Computing Surveys*, 55(6), 1–36, https://doi.org/10.1145/3533382.

[2] Masini, R. P., Medeiros, M. C., and Mendes, E. F. (2021). Machine learning advances for time series forecasting. *Journal of Economic Surveys*, 35(5), 1511–1548. https://doi.org/10.1111/joes.12429.

[3] Li, Y., Lu, X., Xiong, H., et al. (2023). Towards long-term time-series forecasting: feature, pattern, and distribution. Proceedings of AAAI / arXiv preprint arXiv:2301.02068.

[4] Lim, B., and Zohren, S. (2021). Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A*, 379(2194), 20200209. https://doi.org/10.1098/rsta.2020.0209.

[5] Kong, X., Chen, Z., Liu, W., Ning, K., Zhang, L., Marier, SM, Liu, Y., Chen, Y., Xia, F., (2025). Deep learning for time series forecasting: a survey. *International Journal of Machine Learning and Cybernetics*. https://doi.org/10.1007/s13042-025-02560-w.

[6] Salinas, D., Flunkert, V., Gasthaus, J., and Januschowski, T. (2020). DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3), 1181–1191. https://doi.org/10.1016/j.ijforecast.2019.07.001.

[7] Makridakis, S., Spiliotis, E., and Assimakopoulos, V. (2020). The M4 Competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1), 54–74. https://doi.org/10.1016/j.ijforecast.2019.04.014.

[8] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu. T., (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. *Advances in Neural Information Processing Systems*, 30, 3146–3154. https://proceedings.neurips.cc/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html

[9] Essamlali, I., Nhaila, H., El Khaili, M. (2024). Supervised machine learning approaches for predicting key pollutants and for the sustainable enhancement of urban air quality: A systematic review. *Sustainability*, 16(3), 976. https://doi.org/10.3390/su16030976.

[10] Karmoude, M., Munhungewarwa, B., Chiraira, I., et al. (2025). Machine learning for air quality prediction and data analysis: Review on recent advancements, challenges, and outlooks. *Science of the Total Environment*, 1002, 180593. https://doi.org/10.1016/j.scitotenv.2025.180593.

[11] Ravindiran, G., Karthick, K. , Rajamanickam, S., Datta, D., Das, B., Shyamala, G., Hayder, G. and Maria, A., (2025). Ensemble stacking of machine learning models for air quality prediction for Hyderabad city in India. *iScience*, 28(2), 111894. https://doi.org/10.1016/j.isci.2025.111894.

[12] Rad, A.K., Nematollahi, M.J., Pak, A., Mahmoudi, M., (2025). Predictive modeling of air quality in the Tehran megacity via deep learning techniques. *Scientific Reports*, 15, 1367. https://doi.org/10.1038/s41598-025-85675-0.

[13] Pak, A., Rad, A.K., Nematollahi, M.J. Mahmoudi, M., (2025). Application of the Lasso regularisation technique in mitigating overfitting in air quality prediction models. *Scientific Reports*, 15, 547. https://doi.org/10.1038/s41598-024-84200-z.

[14] Mienye, I. D., Sun, Y., Wang, Z. (2024). Recurrent Neural Networks: A Comprehensive Review of Architectures, Variants, and Applications. *Information*, 15(9), 517. https://doi.org/10.3390/info15090517.

[15] Ghojogh, B., Ghodsi, A. (2023). Recurrent Neural Networks and Long Short-Term Memory Networks: Tutorial and Survey. arXiv preprint arXiv:2304.11461.

[16] Lindemann, B., Müller, T., Vietz, H., et al. (2021). A survey on long short-term memory networks for time series prediction. *Procedia CIRP*, 99, 650–655. https://doi.org/10.1016/j.procir.2021.03.088.

[17] Fang, W., Chen, Y., Xue, Q. (2021). Survey on Research of RNN-Based Spatio-Temporal Sequence Prediction Algorithms. *Journal on Big Data*, 3(3), 97–110. https://doi.org/10.32604/jbd.2021.016709.

[18] Waqas M, Humphries UW., (2024). A critical review of RNN and LSTM variants in hydrological time series predictions. *MethodsX*, 13:102946, https://doi.org/10.1016/j.mex.2024.102946.

[19] Al Selwi, S. M., Hassan, S. U., Sumiea, E. H., Abdulkadir, S. J., Talpur, N., Fayyaz, A. M. (2024). RNN LSTM: From applications to modeling techniques and beyond—Systematic review. *Journal of King Saud University – Computer and Information Sciences*, 36(8), 102068. https://doi.org/10.1016/j.jksuci.2024.102068.

[20] Rad, A. K., Razmi, S. O., Nematollahi, M. J., Naghipour, A., Golkar, F., Mahmoudi, M. (2024). Machine learning models for predicting interactions between air pollutants in Tehran Megacity, Iran. Alexandria Engineering Journal, 104, 464-479. https://doi.org/10.1016/j.aej.2024.08.023.

Author(s) Bio-sketches

Abbas Pak received a Ph.D. in Statistics from Shahid Chamran University of Ahvaz. His research interests include applied statistics, and computational methods. He is currently affiliated with Shahrekord University as a faculty member. Corresponding author: abbas.pak1982@gmail.com

Mohammad Javad Nematollahi received a Ph.D. in Environmental Geology from Shiraz University. His main research interests include Environmental Pollution and Geostatistics. He is currently affiliated with Urmia University as a faculty member. mj.nematollahi@urmia.ac.ir