forecasting

MDPI

Article

Recurrent Deep Learning Models for $PM_{2.5}$ Time Series Forecasting in Northern Thailand

Nawisa Jullapech [1], Mohamed Fazil Baksh [2] and Suttida Sangpoom [3,*]

[1] Department of Data Analysis Science and Digital Technology, Nakhon Si Thammarat Rajabhat University, Nakhon Si Thammarat 80280, Thailand; nawisa_jul@nstru.ac.th
[2] Department of Mathematics and Statistics, University of Reading, Reading RG6 6UR, UK; m.f.baksh@reading.ac.uk
[3] Center of Excellence in Data Science for Health Science, Division of Mathematics and Statistics, School of Science, Walailak University, Nakhon Si Thammarat 80160, Thailand
* Correspondence: suttida.sa@mail.wu.ac.th

S1. Supplementary Material

S1.1. Deep Learning Methods

S1.1.1. Recurrent Neural Networks.

Recurrent Neural Networks (RNNs) are specifically designed to effectively model and process sequential data by taking into consideration the sequential nature of the data [46]. The RNN takes as input a sequence of length L, $\{X_\ell\}_1^L = \{X_1, X_2, \ldots, X_L\}$, where each element can be represented as a vector of fixed length. In particular, here each $X_\ell \in \mathbb{R}^p$; $t = 1, \ldots, n$ is p-dimensional, where p represents the number of input features while L is the lag used in fitting the model. The output Y can also be a sequence or a scalar, depending on the application. The structure of a basic RNN with a hidden layer sequence $\{A_\ell\}_1^L = \{A_1, A_2, \ldots, A_\ell\}$, where each $A_\ell \in \mathbb{R}^K$, and a simple output Y is illustrated in Figure 11(a). As the network processes the sequence, one vector X_ℓ at a time, it updates the hidden state A_ℓ using both the current input vector X_ℓ and the previous hidden state $A_{\ell-1}$. Each hidden state A_ℓ is then passed to the output layer to generate a prediction O_ℓ for Y. The final output, O_L, corresponds to the last prediction in the sequence [42].

The collection of $K \times (p+1)$ shared input layer weights w_{kj} can be represented by a matrix $\mathbf{W}$. Similarly, the hidden-to-hidden weights u_{ks} form a $K \times K$ matrix $\mathbf{U}$, and the output layer weights β_k are represented by a vector $\boldsymbol{B}$ of length $K+1$. Then for the hidden layer sequence [42],

$$A_{\ell k} = g(w_{k0} + \sum_{j=1}^{p} w_{kj} X_{\ell j} + \sum_{s=1}^{K} u_{ks} A_{\ell-1,s});\ k = 1, \ldots, K,$$

while the output is defined by,

$$O_\ell = \beta_0 + \sum_{k=1}^{K} \beta_k A_{\ell k},$$

where g is the activation function and a linear output function is assumed.

S1.1.2. Long Short-Term Memory.

The LSTM network, proposed by Hochreiter and Schmidhuber [15], is a type of recurrent neural network designed to mitigate the vanishing gradient problem frequently

Check for updates

Academic Editor: Jun A. Zhang

Received: 22 June 2026
Revised: 23 August 2026
Accepted: 25 August 2026
Published: 8 September 2026

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

observed in traditional RNNs [16]. The LSTM produces two recurrent outputs at each time step (t): the hidden state (h_t) and the cell state (c_t). Both states are propagated from the cell at time t to the same cell at time $t+1$, enabling the network to retain temporal dependencies. In addition, an element of the input sequence $\mathbf{X}_t \in \mathbb{R}^p$ is provided to the cell at each time step.

Within the LSTM cell, the hidden state and the input sequence vector are fed into three distinct control gates at time t, that is, the forget gate (f_t), the input gate (i_t), and the output gate (o_t), each of which applies a sigmoid activation function (σ) [46],

$$f_t = \sigma(w_{f0} + \sum_{j=1}^{p} w_{fj} X_{tj} + \sum_{s=1}^{K} u_{fs} h_{t-1,s}),$$
$$i_t = \sigma(w_{i0} + \sum_{j=1}^{p} w_{ij} X_{tj} + \sum_{s=1}^{K} u_{is} h_{t-1,s}),$$
$$o_t = \sigma(w_{o0} + \sum_{j=1}^{p} w_{oj} X_{tj} + \sum_{s=1}^{K} u_{os} h_{t-1,s}),$$

where $w_{fj}, w_{ij}, w_{oj} \in \mathbb{R}^p$ and $u_{fs}, u_{is}, u_{os}, w_{f0}, w_{i0}, w_{o0} \in \mathbb{R}$ are weight parameters. These parameters are learned during the training process of the cell. The three gates can be interpreted as switches when their output values approach 1 or 0. However, the forget gate (f) regulates the retention of the current cell state, the input gate (i) modulates the incorporation of new information into the cell state, and the output gate (o) determines the extent to which the updated cell state influences the next hidden state.

Another scalar function, known as the cell update at time t, ($\tilde{c}_t$), is designed as a single neuron with a tanh activation function [46],

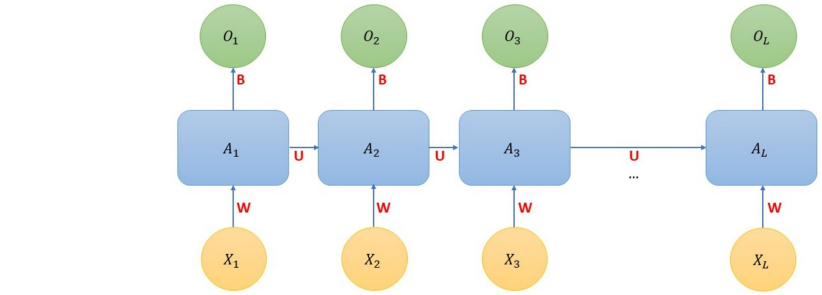
$$\tilde{c}_t = \tanh(w_{c0} + \sum_{j=1}^{p} w_{cj} X_{tj} + \sum_{s=1}^{K} u_{cs} h_{t-1,s}),$$

where $w_{cj} \in \mathbb{R}^p$ and $u_{cs}, w_{c0} \in \mathbb{R}$ are additional learnable parameters. Mathematically, the new cell state and hidden state (output vector) at time t, denoted by c_t and h_t respectively, are given by,

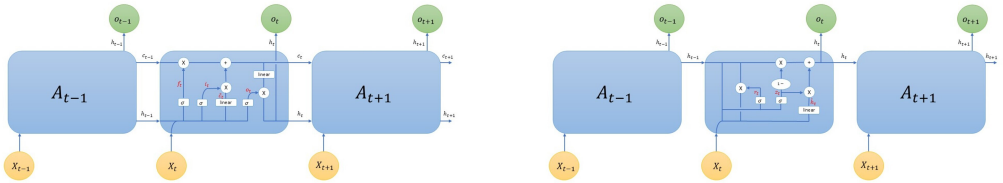
$$c_t = (f_t \times c_{t-1}) + (i_t \times \tilde{c}_t),$$
$$h_t = o_t \times \tanh(c_t),$$

where the arguments $(\mathbf{X}_t, h_{t-1})$ are omitted for clarity. A graphical illustration of the LSTM structure is shown in Figure 11(b). By propagating hidden states through time, recurrent neural networks are capable of handling input sequences of arbitrary length and producing output sequences of corresponding length.

https://doi.org/10.3390/forecast8050080



(a) the standard RNN



(b) the LSTM cell (c) the GRU cell

Figure S1. Recurrent architectures used in this study.

S1.1.3. Gated Recurrent Unit.

Compared to the LSTM, the GRU is a simplified version that uses only two gating signals instead of three. Both gates, the update gate (z_t) and the reset gate (r_t), use a sigmoid activation function (σ) [17], given at time t by,

$$z_t = \sigma(w_{z0} + \sum_{j=1}^{p} w_{zj} X_{tj} + \sum_{s=1}^{K} u_{zs} h_{t-1,s}),$$
$$r_t = \sigma(w_{r0} + \sum_{j=1}^{p} w_{rj} X_{tj} + \sum_{s=1}^{K} u_{rs} h_{t-1,s}),$$

where $w_{zj}, w_{rj} \in \mathbb{R}^p$ and $u_{zs}, u_{rs}, w_{z0}, w_{r0} \in \mathbb{R}$ are weight parameters learned during the training of the cell. The candidate hidden state $\tilde{h}_t$ and the hidden state (output vector) h_t, which together constitute the two core components of the GRU model, can be expressed mathematically at time t by [17]

$$\tilde{h}_t = \tanh(w_{h0} + \sum_{j=1}^{p} w_{hj} X_{tj} + \sum_{s=1}^{K} u_{hs}(r_{t,s} \times h_{t-1,s})),$$
$$h_t = [(1 - z_t) \times h_{t-1}] + [z_t \times \tilde{h}_t],$$

where $w_{hj} \in \mathbb{R}^p$ and $u_{hs}, w_{h0} \in \mathbb{R}$ are additional learnable parameters. A graphical illustration of the GRU function is given in Figure 11(c).

https://doi.org/10.3390/forecast8050080