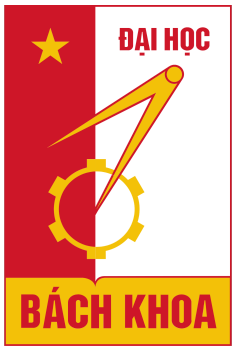
Hanoi University of Science and Technology
School of Information and Communication Technology



Time Series Forecasting: Air Quality Prediction

Lecturer: Than Quang Khoat

Group number: 4

Name	**Student ID**
Dinh Ngoc Lap Thanh	20226000
Pham Duc Phuoc	20235988
Le Hoang Kien	20235958
Nguyen Cong Huan	20225974

Contribution

Task	**Responsibilities**	**Team Member**
Data collection, preprocessing, and analysis	Data cleaning: check for missing data, outliers, etc.	Phuoc
	Data normalization, feature analysis with plots, trend exploration, etc.	Phuoc
	Transform data into format suitable for LSTM	Phuoc
Model design and training	Model coding: LSTM, GRU, 1D-CNN + LSTM	Thanh
	Write training and evaluation scripts	Thanh
	Train custom models and summarize results; document hyperparameters, loss, test results for report	Huan, Thanh, Kien, Phuoc
Demo development	Deploy demo using Python front-end libraries	Kien, Phuoc
Write report		Thanh
Slide preparation		Huan

Contents

List of Figures

List of Tables

Abstract

Admist the rapid urbanization of densely populated cities lies the growing issue of air pollution, which has become a critical environmental issue affecting thousands of its residents. Among the pollutants present, particularly fine particulate matter ($PM_{2.5}$), poses a serious threat to public health, yet direct and accurate measurement of $PM_{2.5}$ remains challenging due to the need for specialized, costly equipment. This mini project proposes a deep learning-based approach to predict $PM_{2.5}$ concentrations using more readily measurable environmental and meteorological features. These features are not only easier to collect but also scientifically correlated with $PM_{2.5}$ behavior due to shared sources and atmospheric interactions. We evaluate three neural architectures: Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), and a hybrid Convolutional Neural Network LSTM (CNN-LSTM). The models are trained on time series data and assessed using standard metrics such as Mean Squared Error (MSE), Mean Absolute Error (RMSE), and Coefficient of determination (R^2). Results show that the CNN-LSTM model outperforms others by effectively capturing both spatial and temporal dependencies in the data. This study demonstrates that accurate $PM_{2.5}$ prediction is feasible using accessible features, offering a practical alternative to direct measurement for air quality monitoring and early warning systems.

Keywords: $PM_{2.5}$ prediction, air quality, LSTM, GRU, CNN-LSTM,time series, environmental monitoring

1 Introduction

Air pollution has emerged as one of the most pressing environmental challenges in rapidly urbanizing and densely populated cities. Among the various air pollutants, fine particulate matter with a diameter of less than 2.5 micrometers ($PM_{2.5}$) is particularly harmful due to its ability to penetrate deep into the respiratory tract, posing significant health risks to humans. Prolonged exposure to $PM_{2.5}$ has been linked to respiratory diseases, cardiovascular problems, and even premature death.

Despite its critical importance, accurate and continuous measurement of $PM_{2.5}$ remains a challenge. Specialized instruments required for direct measurement are often expensive, require regular maintenance, and are limited in spatial coverage—especially in developing countries or regions with limited resources. This limitation highlights the need for alternative methods that can estimate$PM_{2.5}$ levels using more accessible environmental and meteorological data.

In this mini project, we explore the feasibility of predicting $PM_{2.5}$ concentrations using time series modeling and deep learning techniques. Specifically, we leverage features such as PM_{10} , gaseous pollutants (SO_2, NO_2, CO, O_3), meteorological conditions (temperature, dew point, wind speed), and cyclic time components (hour, month, and day encoded as sine and cosine functions). These features are not only easier to collect but are also scientifically correlated with $PM_{2.5}$ due to shared sources, atmospheric reactions, and weather-driven dispersion effects.

To model the temporal dependencies in the data, we evaluate and compare three neural network architectures: Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), and a hybrid Convolutional Neural Network with LSTM (CNN-LSTM). The models are trained and tested on historical air quality data, and their performance is assessed using standard regression metrics such as Mean Squared Error (MSE), Mean Absolute Error (MAE), and R-squared (R^2).

The results demonstrate that the CNN-LSTM model is the most effective among the three, capturing both local patterns and long-term temporal dependencies in the data. By relying on readily available features, this study presents a practical, low-cost solution for $PM_{2.5}$ estimation, which could be valuable for real-time air quality monitoring and early warning systems in resource-constrained environments.

2 Models

In this section, we present the three deep learning models used for air quality prediction in this study: GRU, LSTM, and CNN-LSTM. These models are based on the Recurrent Neural Network (RNN) architecture, which is designed to handle sequential data by maintaining a hidden state that captures information from previous time steps and is update every time step.

Air quality is not a static value; it depends on various factors that change over time, such as weather conditions, traffic density, and industrial activities. RNNs have a memory mechanism that allows them to retain information from previous time steps, enabling the model to recognize patterns and relationships within sequential data. This ability is essential for forecasting future pollution levels based on historical trends.

By modeling dependencies between different timestamps and environmental factors, RNNs provide a valuable tool for predicting pollution trends. Their capacity to analyze evolving conditions makes them instrumental in addressing environmental challenges.

2.1 Recurrent Neural Network (RNN)

As indicated in Figure 1, a Recurrent Neural Network (RNN) processes sequential inputs by maintaining a hidden state that is updated at each time step. The hidden state h_t is computed from the current input x_t and the previous hidden state h_{t-1}:

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

Although simple, traditional RNNs suffer from the vanishing gradient problem, making it difficult to learn long-term dependencies.

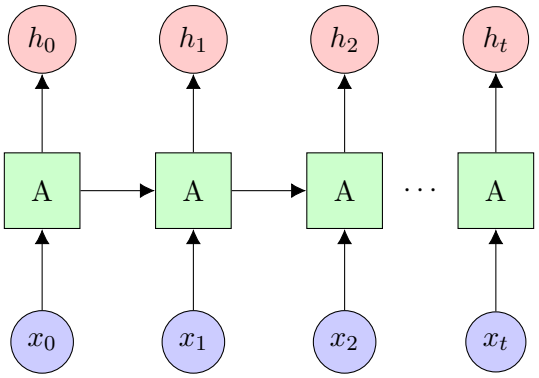


Figure 1: Recurrent Neural Network (RNN) unrolled over time, x_i is feature vector at time step i, h_i is the hidden state vector.

2.2 Gated Recurrent Unit (GRU)

GRU is a variant of RNN designed to capture long-term dependencies using gating mechanisms. Introduces two gates: the update gate and the reset gate. The GRU equations

are as follows:

$$
\begin{aligned}
r^{\langle t \rangle} &= \sigma\left(W_r \cdot \left[h^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_r\right) \\
z^{\langle t \rangle} &= \sigma\left(W_z \cdot \left[h^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_z\right) \\
\tilde{h}^{\langle t \rangle} &= \tanh\left(W_c \cdot \left[r^{\langle t \rangle} * h^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_c\right) \\
h^{\langle t \rangle} &= z^{\langle t \rangle} * h^{\langle t-1 \rangle} + \left(1 - z^{\langle t \rangle}\right) * \tilde{h}^{\langle t \rangle}
\end{aligned}
$$

Symbol explanation:

- $x^{\langle t \rangle}$: input at time step t
- $h^{\langle t \rangle}$: hidden state at time t
- $r^{\langle t \rangle}$: reset gate
- $z^{\langle t \rangle}$: update gate
- $\tilde{h}^{\langle t \rangle}$: candidate hidden state
- W_r, W_z, W_c: weight matrices
- b_r, b_z, b_c: bias vectors
- σ: sigmoid function, tanh: hyperbolic tangent
- $*$: element-wise multiplication

GRU is computationally more efficient than LSTM, while still able to learn long-term patterns.

2.3 Long Short-Term Memory (LSTM)

LSTM is a more powerful architecture than GRU, particularly effective in learning long-term dependencies thanks to its internal memory called cell state. As oppose to GRU which only has a hidden state that can fail to recall long-term memory, LSTM's cell state act as it's long-term memory bank, with a similar update mechanic as GRU, with the new exception of the output gate that can filter more relevant information to use as hidden state. In total, LSTM uses three gates: forget gate, input gate, and output gate. The LSTM updates are defined as follows:

$$
\begin{aligned}
f^{\langle t \rangle} &= \sigma\left(W_f \cdot \left[h^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_f\right) \\
i^{\langle t \rangle} &= \sigma\left(W_i \cdot \left[h^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_i\right) \\
\tilde{C}^{\langle t \rangle} &= \tanh\left(W_c \cdot \left[h^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_c\right) \\
C^{\langle t \rangle} &= f^{\langle t \rangle} * C^{\langle t-1 \rangle} + i^{\langle t \rangle} * \tilde{C}^{\langle t \rangle} \\
o^{\langle t \rangle} &= \sigma\left(W_o \cdot \left[h^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_o\right) \\
h^{\langle t \rangle} &= o^{\langle t \rangle} * \tanh\left(C^{\langle t \rangle}\right)
\end{aligned}
$$

Symbol explanation:

- $f^{\langle t \rangle}$: forget gate

- $i^{\langle t \rangle}$: input gate
- $o^{\langle t \rangle}$: output gate
- $C^{\langle t \rangle}$: cell state at time t
- $\tilde{C}^{\langle t \rangle}$: candidate cell value

LSTM is particularly effective in modeling long sequences because of its explicit memory structure.

2.4 CNN-LSTM

CNN-LSTM is a model that combines Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM). CNN is responsible for extracting localized features using filters to map short-temporal patterns in the data. Then, these features are passed on to the LSTM to learn the long-temporal relationships.

For example, a 24-hour air quality time series (including parameters such as PM2.5, CO, etc.) can be divided into smaller overlapped segments (windows). CNN processes each segment to identify meaningful features and then LSTM continues to analyze the trend of these features over time, helping to predict future pollution levels.

Due to its ability to combine feature extraction using both CNN intra-features learning and LSTM inter-time steps learning, this model is particularly effective in time series forecasting problems with non-correlated features, such as our use of air and weather information.

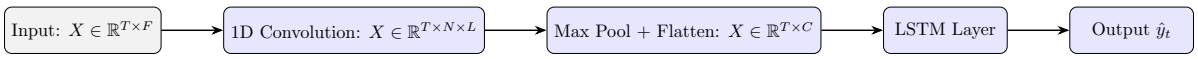


Figure 2: CNN-LSTM architecture use in our experiments. T denotes the number of time steps.N, L is the number of filters and the output size of CNN layer, respectively. $C = N \times L$ is the input size of LSTM layer.

3 Experiments

3.1 Data

Our experiments are conducted on the Aotizhongxin subset of the Beijing Multi-Site Air-Quality Dataset[1], which consists of a 35,064 hourly recorded air pollutants data sample at the Aotizhongxin station in Beijing.

We dropped the station column since it does not provide any significant variance. To capture the cyclical of time dimension data (hour, day, month), we compute sinusoidal transformations of time fields. The sin and cos representation ensures continuity of data on the unit circle. The year column is also dropped as it serves little value when predicting in short time spans.

Any row with missing $PM_{2.5}$ values is dropped entirely, since labels are essential for supervised learning and filling them with any value other than the ground truth would lead to false learning. We computed Pearson's correlation coefficients between all features and the target $PM_{2.5}$ using the following formula:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{1}$$

Here, x_i denotes a value of feature x at time step $i \in [1, n]$. $\bar{x}$, $\bar{y}$ are the average value of feature x and target y, respectively.

We discard the features other than sinusoidal time features with an absolute correlation smaller than 0.1 to reduce noise and irrelevant input. The missing data is filled using median imputation, and the dataset is normalized using Min-Max Scaling to stabilize the gradient during training and facilitate the model's convergence. To construct a training\testing sample, we find and group every 24-hour consecutive time step, resulting in 30,209 samples, the first 80% of which is used as training data. This setup will further test the performance of the model when trained with past data and validated on future data. The pipeline for data pre-processing is available in Google Colab. [2]

[1] Beijing Multi-Site Air-Quality Dataset

[2] `https://colab.research.google.com/drive/1m-mrHvXI6Y4Y1AefeQuCZA3waslOxgxs?usp=sharing`

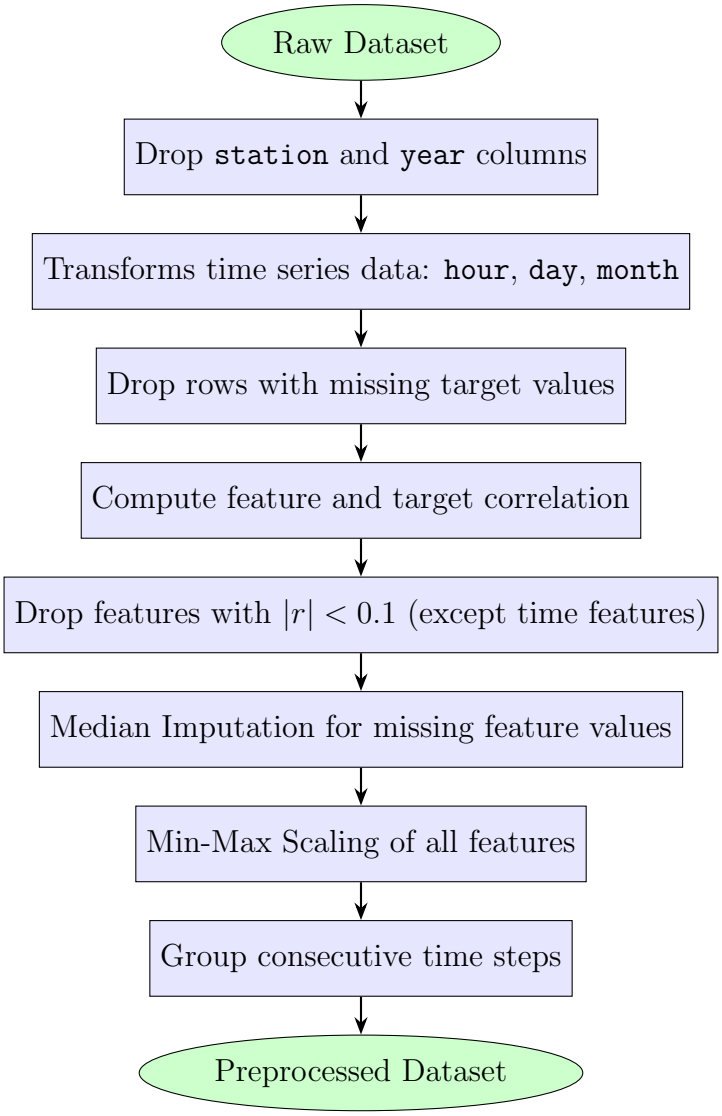


Figure 3: Data preprocessing pipeline

3.2 Results

Using the first 80% of the Aotizhongxin subset as training data for all of our model, we evaluate them on the rest of the subset and choose the best checkpoint. To scientifcally evaluate the prediction accuracy of the model, we uses MAE, MSE and R^2 as the overall evaluation index of the model. MAE describes how different the predicted value is from the true value. MSE measures the average modulus length of the predicted value error, regardless of direction. R^2 describes how similar the predicted value is to the true value. The calculation methods of MAE, MSE and R2 are shown in formula:

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

$$MAE = \frac{1}{m}\sum_{i=1}^{m}|y_i - \hat{y}_i| \tag{3}$$

$$R^2 = 1 - \frac{\sum_i (\hat{y}_i - y_i)^2}{\sum_i (\bar{y} - y_i)^2} \tag{4}$$

Here, m is the number of data samples in the test set, $\hat{y}_i$ is the predicted value; y_i is the true value; $\bar{y}$ is the average value of the true values.

Table 1 describes their architectures, training hyperparameters, and testing performance.

Model	GRU	LSTM	CNN-LSTM
Hidden Size	128	128	144
Batch Size	8	8	16
Learning Rate	$1e^{-3}$	$1e^{-3}$	$1e^{-3}$
Optimizer	Adam	Adam	Adam
Conv Kernel Size	-	-	3
Num Filters	-	-	64
Num Params (M)	55,041	73,345	563,153
Mean Squared Error (MSE)	295.89	<u>160.65</u>	**93.16**
Mean Absolute Error (MAE)	11.65	<u>8.34</u>	**6.10**
R-Squared (R^2)	0.952	<u>0.975</u>	**0.985**

Table 1: Model hyper-parameters for GRU, LSTM and CNN-LSTM models, found via sweeping different combinations and choosing the best performing models. The symbol - denotes the hyperparameter is not applicable for that model. **Bold** denotes the best recorded performance, <u>underline</u> denotes the second best.

Experimental results demonstrate that the CNN-LSTM architecture consistently outperforms the standalone GRU and LSTM models in all evaluation metrics. The performance gain is primarily attributed to the convolution layer, which acts as a feature extractor by capturing local temporal dependencies through learnable filters. These convolved representations, aggregated across short time windows, serve to enrich input to the LSTM, facilitating more effective modeling of long-term dependencies. This hybrid architecture leverages the complementary strengths of CNNs for local pattern recognition and LSTMs for sequence modeling.

We tested LSTM and GRU in different configurations, the results of which are summarized in Table 2. When trying to compensate performance by increasing the model's hidden size, we see a downward trend beyond a certain threshold, which can be due to overfitting. These results point to the need for a new and more optimal architecture, as demonstrated by the CNN-LSTM hybrid design that combines both a fine-grained local feature extractor and long-term information aggregation.

Model	#Params	MSE	MAE	R^2
GRU				
(h=64)	15,233	439.4503	12.8301	0.9350
(h=128)	55,041	**295.8861**	**11.6463**	**0.9525**
(h=160)	84,161	<u>326.9384</u>	<u>11.9175</u>	<u>0.9502</u>
(h=256)	208,385	699.2637	17.5498	0.8837
LSTM				
(h=64)	20,289	212.8400	9.8471	0.9680
(h=128)	73,345	**160.6500**	**8,3434**	**0,9745**
(h=160)	112,161	<u>182,6231</u>	<u>8,8725</u>	<u>0,9718</u>
(h=256)	277,761	227,7566	9,4975	0,9660

Table 2: Performance of GRU and LSTM on various configurations. h represents the size of hidden embedding vector in the RNN block.

4 Conclusion

In this project, we have examine the use of RNN-based model in predicting the $PM_{2.5}$ index on time series data collected from the Aotizhongxin station in Beijing. Our experiments show the large potential of CNN-LSTM in modeling time series data by combining local and long-term temporal patterns. These results suggest that hybrid architectures leveraging both convolution and recurrent layers offer a promising solution for accurate air quality forecasting.