IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received March 26, 2021, accepted April 5, 2021, date of publication April 9, 2021, date of current version April 20, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3072280

PM2.5 Prediction Using Genetic Algorithm-Based Feature Selection and Encoder-Decoder Model

MINH HIEU NGUYEN[1], PHI LE NGUYEN[1], KIEN NGUYEN[2], (Senior Member, IEEE), VAN AN LE[3], (Graduate Student Member, IEEE), THANH-HUNG NGUYEN[1], AND YUSHENG JI[3,4], (Senior Member, IEEE)

[1]School of Information and Communication Technology, Hanoi University of Science and Technology, Hanoi 11615, Vietnam
[2]Graduate School of Engineering, Chiba University, Chiba 263-8522, Japan
[3]Department of Informatics, The Graduate University for Advanced Studies, SOKENDAI, Tokyo 240-0193, Japan
[4]National Institute of Informatics, Tokyo 101-8430, Japan

Corresponding author: Phi Le Nguyen (lenp@soict.hust.edu.vn)

This work was supported by the Vingroup Joint Stock Company (Vingroup JSC), Vingroup, and in part by the Vingroup Innovation Foundation (VINIF) under Project VINIF.2020.DA09. The work of Minh Hieu Nguyen was supported in part by the Vingroup Joint Stock Company, and in part by the Domestic Master/ Ph.D. Scholarship Program of Vingroup Innovation Foundation (VINIF), Vingroup Big Data Institute (VINBIGDATA), under Grant VINIF.2020.ThS.BK.08.

ABSTRACT The concentration of fine particulate matter (PM2.5), which represents inhalable particles with diameters of 2.5 micrometers and smaller, is a vital air quality index. Such particles can penetrate deep into the human lungs and severely affect human health. This paper studies accurate PM2.5 prediction, which can potentially contribute to reducing or avoiding the negative consequences. Our approach's novelty is to utilize the genetic algorithm (GA) and an encoder-decoder (E-D) model for PM2.5 prediction. The GA benefits feature selection and remove outliers to enhance the prediction accuracy. The encoder-decoder model with long short-term memory (LSTM), which relaxes the restrictions between the input and output of the model, can be used to effectively predict the PM2.5 concentration. We evaluate the proposed model on air quality datasets from Hanoi and Taiwan. The evaluation results show that our model achieves excellent performance. By merely using the E-D model, we can obtain more accurate (up to 53.7%) predictions than those of previous works. Moreover, the GA in our model has the advantage of obtaining the optimal feature combination for predicting the PM2.5 concentration. By combining the GA-based feature selection algorithm and the E-D model, our proposed approach further improves the accuracy by at least 13.7%.

INDEX TERMS PM 2.5, genetic algorithm, feature selection, long short-term memory, encoder-decoder model.

I. INTRODUCTION

Industrialization and urbanization have brought considerable convenience to human lives. However, they are generally associated with severe air pollution. Accordingly, people have raised concerns about air quality, especially near living areas. Particulate matter 2.5 (PM2.5) is one of the most important indexes to evaluate the severity of air quality, which is directly related to human health. PM2.5 particles in the air can bypass the nose and throat and penetrate deep into the lungs, causing many diseases, such as cardiovascular disease and respiratory disease. In [1], the authors reveal that long-term exposure to PM2.5 may lead to heart attack and stroke. Therefore, accurate PM2.5 forecasting is crucial and may help governments and citizens find suitable solutions to control or prevent negative conditions.

PM2.5 forecasting is a time series prediction problem that is commonly solved using recurrent neural networks (RNNs), including LSTM [2]. The LSTM-based model has advantages in air quality prediction [3]. In [4], the authors also use LSTM but combine gas and PM2.5 concentrations to predict air quality in Taiwan. The work in [5] exploits deep learning to build a hybrid neural network model that can forecast PM2.5 multiple steps ahead. In [6], Yanlin *et al.* present a hybrid model that integrates graph convolutional networks and LSTM to predict PM2.5. In [7], the authors utilize the k-nearest neighbor algorithm to mine spatial-temporal information. The historical information of related locations is then used as the input of the LSTM, adaptive temporal extractor (ASE), and artificial neural network (ANN) models. Several

The associate editor coordinating the review of this manuscript and approving it for publication was Wai-keung Fung.

other deep learning models for predicting air quality can be found in [8]–[11]. Despite considerable effort, air quality prediction models still suffer from two issues: restrictions of the input and output lengths and unoptimized feature selection.

The first issue indicates that the number of time steps in a model's output cannot exceed that of the input; i.e., the model cannot predict the future with upcoming steps that exceed the input data's length. Therefore, it is essential to remove this limitation in PM2.5 prediction. The second issue arises from the fact that air quality data include dozens of factors other than PM2.5, such as various concentrations, temperature and humidity. These factors may or may not be related to PM2.5. However, appropriate use of some of these factors may improve the prediction accuracy. Meanwhile, misuse may not only degrade the accuracy but also add extra computational time. Therefore, choosing the optimal feature combination is essential.

This paper aims to address the two issues described above. As a solution, we propose a novel PM2.5 prediction model that combines a genetic algorithm (GA) and an encoder-decoder (E-D) model. The GA is exploited to perform feature selection in a near-optimal manner, thereby enriching the prediction model. Additionally, we leverage the encoder-decoder model to build a PM2.5 prediction model with high accuracy. As a result, the proposed model can efficiently handle different sizes (in terms of the number of time steps) of input and output. To demonstrate the effectiveness of our proposed approach, we evaluate the GA-based feature selection on the Hanoi [12] and Taiwan datasets [11]. The evaluations show that the GA-based feature selection outperforms other methods. We then compare our model to the state-of-the-art method ST-DNN in [11] using the Taiwan dataset. Compared to ST-DNN, our model improves the accuracy from 14.82% to 41.71%. By combining the GA-based feature section algorithm and the E-D model, our proposed approach further increases the accuracy by at least 3%.

The remainder of this paper is organized as follows. We describe the motivations in Section II. Section III presents our proposal. The performance of evaluation is introduced in Section IV. Section VI introduces related works. Finally, Section VII concludes the paper.

II. MOTIVATION

Our research is motivated by the two issues in air quality prediction mentioned in the previous section. In the following, we describe our ideas to address these problems.

A. RESTRICTION BETWEEN THE INPUT AND OUTPUT LENGTH

In previous prediction models [3], [11], the number of time-steps of the output cannot exceed that of the input. However, we sometimes expect the output to have more time steps than the input. For example, we may want to use historical data of the past 30 days to estimate the air quality of the next 60 days. A simple method to overcome this restriction is to use iterative estimation [13], in which a forecasting model predicts one time step ahead and then iteratively concatenates the predicted result to the current input. The concatenation is fed to the forecaster to obtain the multistep-ahead prediction. The greatest benefit of this strategy is the ease of training because it needs to consider only the one-step-ahead forecasting error. Moreover, we can generate multistep samples by recursively using the prediction outcome. However, the forecaster is fed the predicted values instead of the ground truth; therefore, the prediction error accumulates throughout the estimation process, eventually becoming significant.

We aim to leverage the encoder-decoder model to address this issue. The model includes an encoder and a decoder module that break the prediction process into two independent steps: encoding and decoding. The former extracts the features of the input, and the output of the encoding process is fed into the decoder module. Then, we obtain the predicted values as a result of the decoding step. As a result, the encoder and decoder modules are independent. Therefore, the number of time steps of the decoder's output is not restricted by that of the encoder's input. Moreover, to capture the temporal relation of the data, we utilize LSTM units [2], which can learn long-term dependencies, to construct the encoder and the decoder.

B. DIFFICULTY IN SELECTING FEATURES

In addition to PM2.5, the data used for air quality prediction usually comprise many other features, which raises the concern of choosing additional features for PM2.5 prediction. *If we use a random number or all the features, does the prediction accuracy improve?* To answer this question, we conduct prediction on a dataset of air quality monitored in the center of Hanoi, Vietnam [12]. The dataset comprises 21 features, including temperature, CO, NO, NO2, NOx, O3, PM10, PM2.5, RH, and SO2. We use the random search algorithm, which chooses ten combinations of features randomly. We then use each combination, only PM2.5, and all features as inputs to train an XGBoost model [14]. As in [3], [15], we consider the mean absolute error (MAE) metric, which is calculated as follows.

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

where y_i is the ground truth; $\hat{y}_i$ is the predicted values; and n is the number of data points. The mean absolute error (MAE) achieved by the twelve combinations are shown in Fig. 1. Additional features do not guarantee better MAE compared to using only PM2.5. In the case of all features being used, the MAE value is not better than that of several random combinations because the use of a large number of features interferes with the prediction model, degrading the accuracy. Moreover, using all the features may slow the training process. Therefore, it is essential to select features for the prediction model.

A common method of feature selection is to use correlation evaluation algorithms, such as Pearson's [16].

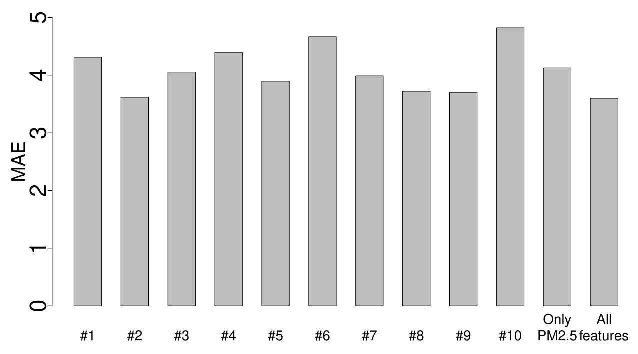


FIGURE 1. MAE achieved by XGboost with different combinations.

The algorithms first determine the correlation between features and then select the most correlated features to include in the prediction model. Another approach is to use an embedded model such as Lasso [17] or random forest [18], which have associated feature selection methods. However, these methods require considerable time and human brainpower. In this work, we exploit the genetic algorithm (GA), which is a meta-heuristic inspired by the process of natural selection based on Darwin's theory of evolution. GA efficiently searches among all possible feature combinations to find the optimal combination that achieves the maximum prediction accuracy. The search process begins with a set of individuals (i.e., the initial population), where each individual is a solution to the problem, and the goodness of each individual is evaluated by a fitness function. By performing the GA operations on the individuals, we improve the fitness over the generations. Consequently, after several generations, we achieve a solution whose fitness value is optimal. Although GA does not ensure the best solution, it tends to provide a good solution that is near the global optimal within a reasonable search time.

III. PROPOSED PREDICTION MODEL

A. OVERVIEW

Figure 2 presents an overview of our proposed method, which includes GA-based feature selection and a prediction module that are detailed in Section III-B and Section III-C, respectively. In the left module, each individual in the initial population represents a feature combination. The GA-based method includes numerous iterations (i.e., generations), in each of which genetic operations of crossover and mutation generate new individuals. Each individual is evaluated with respect to fitness. We define fitness as the MAE, which is the prediction model's output with a specified feature combination acting as the model's input. In each generation, the individuals with better fitness remain; the others are removed from the population. By repeating these processes, the fitness values of the population are improved in each generation. Finally, we identify the individual with the optimal fitness value. The right module is the encoder-decoder based prediction for PM2.5. The module includes a prediction model that uses a

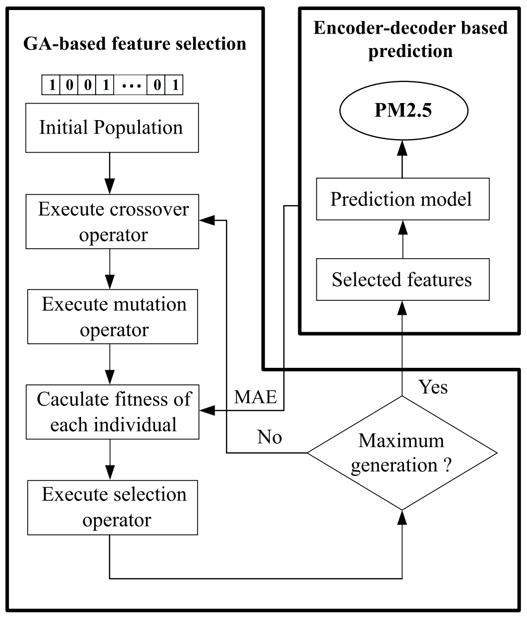

GA-based feature selection
1 0 0 1 ··· 0 1
Initial Population
Execute crossover operator
Execute mutation operator
Caculate fitness of each individual
Execute selection operator
MAE
No
Maximum generation ?
Yes
Encoder-decoder based prediction
PM2.5
Prediction model
Selected features

FIGURE 2. Overview of the proposed model.

data set that consists of features selected by the GA-based module as the input.

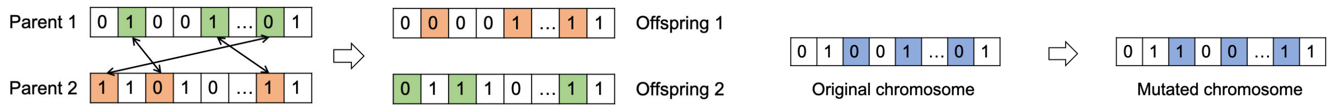
B. GA-BASED FEATURE SELECTION

Let n be the number of all features in the data set. Each individual is a binary string of length n that encodes a feature combination. Specifically, the k-th gene obtains the value of either 1 or 0, which indicates whether the k-th feature is selected (as shown in Fig. 3). A genetic algorithm adopts the fitness value to represent how "good" an individual is. In our algorithm, we define the fitness of an individual as the mean absolute error (MAE), which is obtained by the prediction model. For each generation, we first determine all the newly created individuals, each of which is used to train the prediction model. We then evaluate the model to obtain the MAEs.

Wind	Temperature	Humidity	Barometer	....	PM10
0	1	0	0		1

FIGURE 3. Encoding a feature combination (the white and gray cells represent the selected feature encoded by 1 and 0, respectively).

The initial population is generated randomly. The crossover and mutation algorithms work as follows. Let $A = \{a_1, a_2, \ldots, a_n\}$ and $B = \{b_1, b_2, \ldots, b_n\}$ be two parents who will be crossed. We randomly select m genes of A and B and swap them to produce two offspring, where m is a random number that varies in the range from 1 to $\frac{n}{2}$. Moreover, to retain good features, we propose a heuristic algorithm for selecting parents when performing the crossover as follows. Let p_c be the crossover probability and N be the population size; then, we choose among the N individuals $N \times p_c$



(a) Crossover: the green and orange cells represent genes that are swapped to create two offspring.

(b) Mutation: the blue cells represent genes that are mutated.

FIGURE 4. Illustration of the GA's crossover and mutation operations.

individuals to crossover. We choose the $\frac{n \times p_c}{2}$ individuals who have the best fitness values. The other $\frac{n \times p_c}{2}$ parents are randomly chosen among the remaining $N - \frac{n \times p_c}{2}$ individuals. For mutation, we randomly select a gene segment of a parent and then invert the values of the genes (i.e., change bit 0 to 1, and vice versa). To increase the diversity of the individuals, we use two selection operators: the best individual selection operator and the random selection operator. Specifically, 50% of the best individuals are selected for the next evolution, and the remaining 50% are chosen randomly. Figure 4 illustrates the crossover and mutation operations of our algorithm.

C. ENCODER-DECODER MODEL-BASED PREDICTION

Our model obtains information from the previous l time steps and predicts the PM2.5 values h time steps in the future. Let e_x and d_x be the inputs of the encoder and decoder, and let d_y be the output of the decoder. These values are defined as follows.

Input:

$$e_x = \left(x_i^k, x_{i+1}^k, \ldots, x_{i+l-1}^k\right),$$
$$d_x = \left(x_{i+l-1}^k, x_{i+l}^k, \ldots, x_{i+l+h-2}^k\right),$$

Output:

$$d_y = (y_{i+l}, y_{i+l+1}, \ldots, y_{i+l+h-1}),$$

where:

- $k = \{0, 1\}$ represents the {removed, selected} feature.
- i is the iteration counter, which ranges from 0 to the length of the training set.
- l is the number of time steps in the input sequence.
- h is the number of time steps in the output sequence.
- x is the input data of the selected feature.
- y is the predicted PM2.5 value.

We leverage the encoder-decoder architecture, wherein the encoder and decoder comprise l and h LSTM units, respectively. We also use the Adam optimizer [19] to optimize the learning rate automatically. In the following discussion, we first present the LTSM unit's details and then the encoder-decoder model structure.

1) LONG SHORT-TERM MEMORY (LSTM)

LSTM is tremendously helpful in multistep time series forecasting. It performs the same task for all elements with the output of the current step, which can be used as the input in the upcoming step. Unlike a traditional neural network, LSTM can store previously calculated information, thereby gaining benefits when the processing data are in the form of a sequence. The original NN suffers from a critical weakness: the vanishing gradient problem, where earlier steps' contribution becomes insignificant in future steps, so the model fails to capture long-term dependencies. LSTM addresses the vanishing gradient problem by introducing three special units: forget gate, update gate, and output gate. The gates can be used to decide which information and how much of each type of information should be remembered. The main advantage of LSTM compared to other forms of RNNs is the ability to learn long-term dependencies.

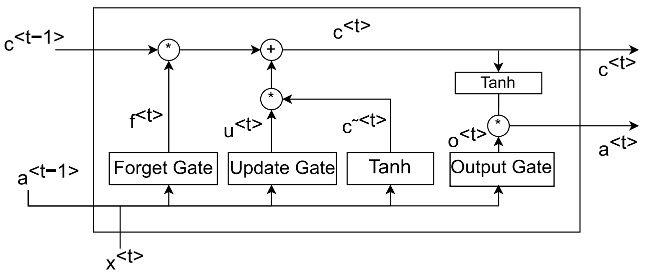


FIGURE 5. Structure of the LSTM unit.

Figure 5 shows the typical structure of an LSTM unit, consisting of three main gate structures: update gate $u^{<t>}$, forget gate $f^{<t>}$, and output gate $o^{<t>}$. The input of an LSTM unit comprises three elements: a cell state $c^{<t-1>}$ and a hidden state $a^{<t-1>}$ outputted by the previous LSTM unit and the input data of the current unit $x^{<t>}$. $\tilde{c}^{<t>}$ is a candidate for replacing the memory cell, which is calculated from the current input and the previous hidden state.

The forget gate and the update gate layer are both computed with a sigmoid function, whose input comprises the previous hidden state and the current input. Thus, the output values of the forget gate and the update gate are within the range (0, 1). The value of the forget gate is applied to the previous cell state $c^{<t-1>}$, while the value of the update gate is multiplied by the new memory $\tilde{c}^{<t>}$ to form the new cell state. Intuitively, the forget gate controls to what extent previous information is forgotten, while the update gate decides the extent of the current input and the previous hidden state to be written onto the new cell state. The closer the value is to 0, the more information is forgotten and ignored, and the closer the value is to 1, the more information is maintained and stored. $o^{<t>}$ is the output gate, which determines what the next hidden state should be. The equations for each state are as follows:

$$\begin{aligned}
\tilde{c}^{<t>} &= tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c) \\
u^{<t>} &= \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u) \\
f^{<t>} &= \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f) \\
o^{<t>} &= \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_o) \\
c^{<t>} &= u^{<t>} * \tilde{c}^{<t>} + f^{<t>} * c^{<t-1>} \\
a^{<t>} &= o^{<t>} * tanh(c^{<t>}), \qquad (1)
\end{aligned}$$

where W corresponds to the weight matrix; b is the bias coefficient; and σ and *tanh* are activation functions.

The new state $c^{<t>}$ is calculated from the previous state $c^{<t-1>}$ and the new memory candidate $\tilde{c}^{<t>}$. First, the previous cell state $c^{<t-1>}$ is pointwise multiplied by the forget vector $f^{<t>}$, which has the ability to drop values in the cell state if $f^{<t>}$ goes to 0. Then, we add $u^{<t>} * \tilde{c}^{<t>}$, which updates the cell state to new values that the neural network finds relevant. This process yields our new cell state calculated by $f^{<t>} * c^{<t-1>} + u^{<t>} * \tilde{c}^{<t>}$, which decides the information to forget and update. Finally, the new hidden state $a^{<t>}$ is obtained by multiplying the output gate with the *tanh* of the new cell state. The new cell state and the new hidden state are carried over to the next time step.

In summary, the greatest advantages of LSTM and using gates are that the forget gate decides what is relevant to keep from previous steps, the update gate decides what information is relevant to add from the current step, and the output gate determines what the next hidden state should be.

2) ENCODER-DECODER MODEL

The E-D model was first proposed to solve a natural language processing problem [20]. The initial input is a sequence of words of length m (i.e., $x = \{x_1, x_2, \ldots, x_m\}$). After passing through the model, a newly created sequence at the output $y = \{y_1, y_2, \ldots, y_n\}$ has length n, which can be the as same as n or different. The model can adopt either RNN, GRU, or LSTM, depending on the application. The E-D model includes an encoder and a decoder. The encoder has a stack of several recurrent units, each of which accepts an element of the input sequence with an arbitrary length. It then collects information for that element and propagates it forward. The output is a hidden state and a cell state called the encoder state that aims to encapsulate all input elements' information to support the accurate prediction by the decoder, which is the second component. The state is then fed into the decoder as the initial state. While the process of producing the encoder state is the same as equation 1, the hidden state of the decoder at the first time step is computed by:

$$\begin{aligned}
\tilde{c}^1 &= tanh(W_c[a^{ec}, < GO >] + b_c) \\
u^1 &= \sigma(W_u[a^{ec}, < GO >] + b_u) \\
f^1 &= \sigma(W_f[a^{ec}, < GO >] + b_f) \\
o^1 &= \sigma(W_u[a^{ec}, < GO >] + b_o) \\
c^1 &= u^1 * \tilde{c}^1 + f^1 * c^{ec} \\
a^1 &= o^1 * tanh(c^1), \qquad (2)
\end{aligned}$$

where a^{ec} is the encoder state and <GO> is the decoder's seeding value, which is zero in our model.

Similar to the encoder, the decoder has a stack of several recurrent units. However, each recurrent unit accepts a hidden state from the previous unit as the input and predicts an output of y_i at time step i. The hidden state of the decoder at the n-th time step is computed as follows:

$$\begin{aligned}
\tilde{c}^{<t>} &= tanh(W_c[a^{<t-1>}, p^{<t-1>}] + b_c) \\
u^{<t>} &= \sigma(W_u[a^{<t-1>}, p^{<t-1>}] + b_u) \\
f^{<t>} &= \sigma(W_f[a^{<t-1>}, p^{<t-1>}] + b_f) \\
o^{<t>} &= \sigma(W_u[a^{<t-1>}, p^{<t-1>}] + b_o) \\
c^{<t>} &= u^{<t>} * \tilde{c}^{<t>} + f^{<t>} * c^{<t-1>} \\
a^{<t>} &= o^{<t>} * tanh(c^{<t>}), \qquad (3)
\end{aligned}$$

where $p^{<t-1>}$ is the prediction result of the previous time step.

After obtaining the hidden state $a^{<t-1>}$, we pass it through a regular neural network layer called the dense layer to obtain the final prediction, which is calculated by the following equation:

$$p^{<t-1>} = W_p * a^{<t-1>} + b_p, \qquad (4)$$

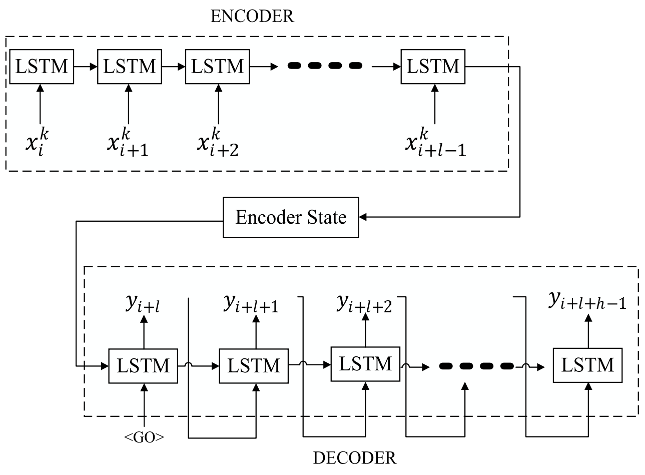


FIGURE 6. Structure of the encoder-decoder model.

Figure 6 illustrates the proposed E-D model. The encoder has an input length of l, while the output length of the decoder is h. $x_i^k, x_{i+1}^k, \ldots, x_{i+l-1}^k$ is the input data, which consists of the historical selected features from i to $(i+l-1)$ corresponding to l time steps. $y_{i+l}, y_{i+l+1}, \ldots, y_{i+l+h-1}$ are the predicted PM2.5 values from $(i + l)$ to $(i + l + h - 1)$ corresponding to h time steps. The encoder receives each element in the input and processes them through l LSTM units to create a final hidden state. This hidden state is then fed into the decoder, which consists of h LSTM units, to predict PM2.5 in h steps.

TABLE 1. Details of missing data in the datasets.

Dataset	Indicator	Total data	Missing values	Missing rate
Hanoi	PM2.5	17123	272	1.59%
	PM10	17123	272	1.59%
	PM1.0	17123	272	1.59%
	Others	17123	0	0%
Taiwan	Temp	32832	203	0.62%
	CO	32832	484	1.47%
	NO	32832	1200	3.65%
	NO2	32832	1200	3.65%
	NOx	32832	1200	3.65%
	O3	32832	462	1.41%
	RH	32832	202	0.62%
	SO2	32832	666	2.03%
	WD_HR	32832	250	0.76%
	Wind direction	32832	192	0.58%
	Wind speed	32832	202	0.62%
	WS_HR	32832	253	0.77%
	PM10	32832	623	1.90%
	PM2.5	32832	891	2.71%

IV. PERFORMANCE EVALUATION

A. EVALUATION SETTINGS

To ease the presentation, we name our proposed model ED-LSTM. This section aims to answer two questions related to prediction accuracy.

1) How much does the feature selection algorithm improve the results compared to other selection methods?
2) How much does our proposed prediction model gain compared to state-of-the-art models?

Regarding the first question, we initially combine ED-LSTM with various feature selection algorithms. Then, we compare their performance with that of our proposed GA-based algorithm using two datasets. The first dataset, named Hanoi dataset [12], is the air quality collected in Hanoi, Vietnam. The Hanoi dataset contains hourly data from January 2016 to January 2018 for the features mentioned in Section II. The second dataset, named the Taiwan dataset, contains hourly data collected from January 2014 to September 2017 [11]. The dataset includes PM2.5 and other indicators, such as time, ambient temperature, CO, NO, NO2, NOx, O3, PM10, RH, and SO2. In both datasets, there are some missing data points. The missing number of each indicator and its percentage are shown in 1. It is necessary to do data imputation to compensate for the missed data. We see that the missing rates are relatively small and do not bias any specific parameters in ED-LSTM. Hence, we leverage the median value to fill in the lost places. To further confirm the method's effectiveness, we compare it to the one in [11] (the original one with Taiwan data set). With such missing rates, the experiment results, with such missing rates, the imputation methods do not impact the models' performance. Therefore, throughout this paper, we present the results using the median values for data imputation.

Concerning the second question, we compare ED-LSTM with three existing works. The first one (i.e., AE-BiLSTM) is a combination of auto-encoder and bi-LSTM neural network [21]. The second one, AC-LSTM proposed in [22], leverages CNN and LSTM network. The last one (i.e., ST-DNN) exploits both the spatial and temporal relationship to predict the PM 2.5 [11].

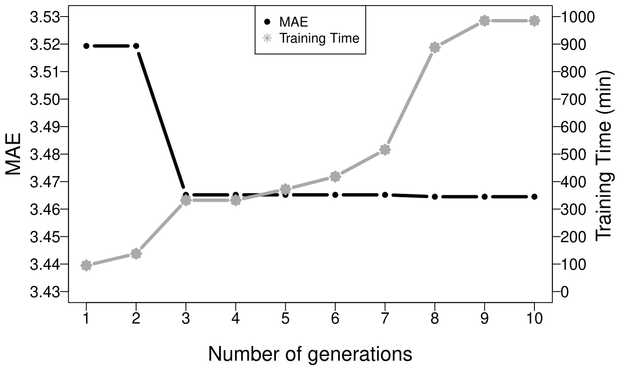


FIGURE 7. Impact of the number of generations.

TABLE 2. Parameter settings.

Batch Size	200	Neuron Units	100
Test Size	0.2	Epochs	300
l	48	Optimizer	Adam
h	$1 \sim 31$	Early Stopping	15

Table 2 summarizes the hyperparameters in ED-LSTM. *Batchsize* is the number of training samples propagated through the network. *Testsize* defines the percentage of the data set used as the test set. l and h denote the numbers of time steps of the input and output, respectively. An *epoch* refers to one cycle through the full training dataset. The *earlystopping* value is the threshold for terminating the training process to prevent overfitting. We follow the common method to adjust the hyperparameters of deep learning models, that is, to try various sets of parameters and choose the best one.

B. IMPACT OF THE GA's NUMBER OF GENERATIONS

In this experiment, we study the impact of the number of generations in our GA-based feature selection algorithm. We consider two metrics: prediction accuracy and training time. Figure 7 shows the MAE and the training time when we vary the number of generations from 1 to 10. In general, increasing the number of generations reduces the MAE but increases the training time. Specifically, when the number of generations increases from 1 to 3, the MAE drops severely and remains almost stable beyond that point. Concerning the training time, it increases significantly as the number of generations increases from 1 to 3. The rate of increase slows as the number of generations increases from 4 to 7 and becomes substantial beyond that point. According to the experimental results, a moderate number of generations from 5 to 7 should be considered to achieve high prediction accuracy while maintaining an acceptable training time.

C. COMPARING FEATURE SELECTION ALGORITHMS

We compare the proposed GA-based feature selection to the following methods: using only PM2.5 data, all features, XGBoost, and Pearson's correlation. The XGBoost package provides a built-in function called *feature_importances* that receives the input as a list of features and returns the importance coefficient of each feature [14]. We first sort the features in decreasing order of the coefficient. Then, we gradually remove features with low importance from the original feature list (i.e., the list containing all features). For the remaining features, we train the prediction model and evaluate the MAE. Pearson's correlation measures the strength of the linear relationship between two features. The correlation ranges from -1 to 1. We choose features with an absolute correlation value greater than 0.3, which means they are moderately or highly correlated to PM2.5. Moreover, we remove features whose absolute correlation with PM2.5 is greater than 0.9 to prevent multicollinearity (i.e., the strong correlation reduces the model's capability to identify significant independent variables).

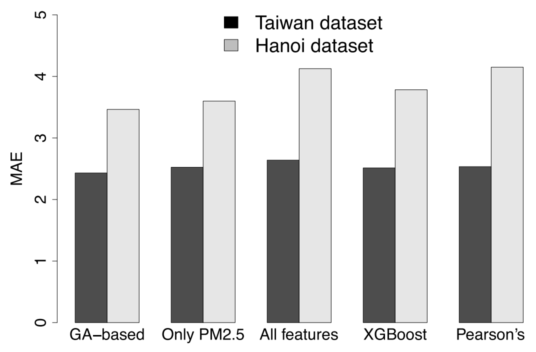


FIGURE 8. Comparison of feature selection algorithms.

Figure 8 shows the MAE values for one-step ahead prediction attained by the feature selection methods. Our proposed GA-based method has the smallest MAE on the Taiwan dataset and Hanoi dataset, thereby achieving the best performance. In the GA-based method on the Hanoi dataset, the optimal feature combination includes wind speed, temperature, radiation, PM10, and PM2.5. The figure also presents an interesting observation: using all the features results in worse performance (i.e., higher MAE) than using only PM 2.5. Therefore, prediction based on more input features does not guarantee a better outcome. However, the GA-based method reduces the MAE by 6% and 16% compared to using PM2.5 and all the features, respectively. Additionally, the MAE attained by our method is 90% and 83% smaller than that of XGBoost and Pearson's correlation. For the Taiwan dataset, the GA-based method reduces the MAE by 4%, 8%, 3%, and 4% compared to using PM2.5, all features, XGBoost and Pearson's correlation, respectively.

We compare the PM2.5 prediction (i.e., from one to eight time-steps ahead) of the GA-based feature selection and all features as input with the Hanoi dataset. The results are shown in Fig. 9. In all cases, the results with the features selected by GA have more accurate peaks. It again proves the effectiveness of the GA-based feature selection method.

D. COMPARING PREDICTION MODELS

This section compares our ED-LSTM to AE-BiLSTM [21], AC-LSTM [22], and ST-DNN [11]. Since all three models' source codes are not publicly available, we have initially reimplemented them. We have successfully recreated AE-LSTM and AC-LSTM based on the papers' instructions. However, we cannot get ST-DNN in the same way due to the lack of details. Instead, we have copied the numerical results directly from the ST-DNN original paper. We perform two experiments for the comparison. In the first one, we use the Hanoi dataset to evaluate our model, AE-LSTM, and AC-LSTM. In the second one, to guarantee a fair comparison between ED-LSTM and ST-DNN, we use the Taiwan dataset and the features suggested by ST-DNN's paper [11]. The data is prepared as follows. We partition the data into a training set (from Jan. 2014 to Sep. 2016) and a testing set (from Oct. 2016 to Sep. 2017) with a ratio of 2 : 1. The experiment is conducted by receiving the historical data of the past 48 hours and predicting the value of PM2.5 in the next $(1, 2, \ldots, 6)$ hours.

TABLE 3. ED-SLTM, AE-BiLSTM, and AC-LSTM use all features (Hanoi dataset) .

h	ED-LSTM		AE-BiLSTM		AC-LSTM	
	MAE	Computation time (s)	MAE	Computation time (s)	MAE	Computation time (s)
1	**4.395**	0.164	10.817	0.078	5.290	**0.051**
2	**5.575**	0.279	11.091	0.157	5.850	**0.102**
3	**6.076**	0.332	11.305	0.241	6.365	**0.156**
4	**6.662**	0.462	11.573	0.326	6.770	**0.208**
5	**6.986**	0.535	11.677	0.402	7.278	**0.261**
6	**7.246**	0.637	11.680	0.481	7.765	**0.314**

TABLE 4. ED-SLTM, AE-BiLSTM, and AC-LSTM use the selected features by GA (Hanoi dataset) .

h	ED-LSTM (proposed method)		AE-BiLSTM		AC-LSTM	
	MAE	Computation time (s)	MAE	Computation time (s)	MAE	Computation time (s)
1	**3.592**	0.167	8.242	0.083	3.919	**0.053**
2	**4.603**	0.126	8.567	0.166	4.762	**0.104**
3	**5.173**	0.375	8.830	0.264	5.531	**0.154**
4	**5.491**	0.457	9.083	0.330	6.168	**0.207**
5	**6.135**	0.559	9.323	0.411	6.748	**0.256**
6	**6.684**	0.629	9.634	0.493	7.710	**0.314**

1) COMPARING ED-SLTM, AE-BiLSTM, AND AC-LSTM

We evaluate ED-LSTM, AE-BiLSTM, and AC-LSTM with the Hanoi dataset and show the MAE and computation time in Tables 3 and 4. The average computation time of a model, defined as the model's average time to perform one prediction, is measured as follows. We track the moment when feeding data to a model and the moment of receiving the prediction result. The duration between two moments is the measured time. Table 3 shows the results when using

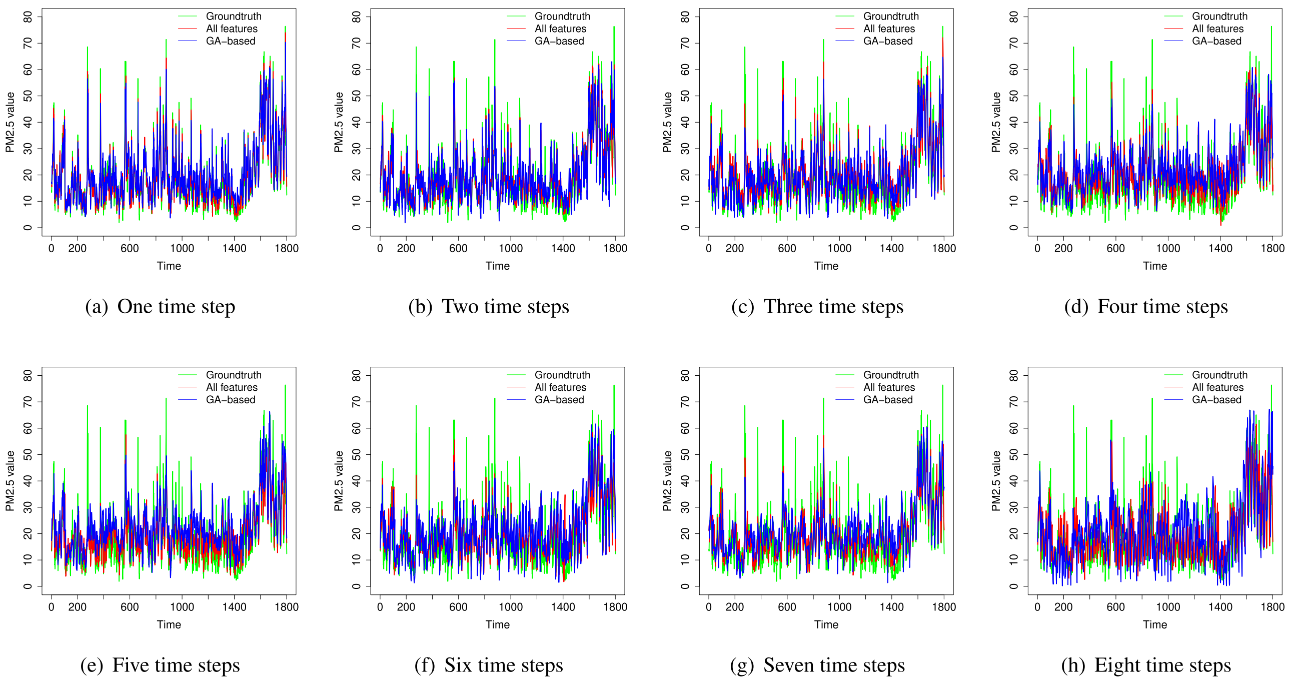


FIGURE 9. Comparison of GA-based feature selection and using all the features for the Hanoi dataset.

all the features as the input, while Table 4 depicts the results when using the features selected by our GA-based algorithm.

First, we can see that ED-LSTM achieves the highest prediction accuracy regarding both two feature sets. When using all features, ED-LSTM reduces the MAE by 46%, and 6.4% compared to AE-BiLSTM and AC-LSTM, respectively. With the features selected by the GA-based algorithm, the ED-LSTM model further reduces the MAE by smaller than 53.7%, and 20.1% on average compared to AE-BiLSTM and AC-LSTM. As shown in Tables 3 and 4, with each feature selection, AE-BiLSTM attains the lowest computation time, while ED-LSTM and AC-LSTM achieve similar performance. We can see that although the computation time of ED-LSTM is higher than AE-BiLSTM, their gap is relatively small. Within the scope of PM 2.5 prediction (e.g., data sampling, PM variation), the ED-LSTM's computation time is acceptable for real-time prediction.

Second, we cross-compare the performance between two types of feature selections in the three models. From the results in Tables 3 and 4, each model's MAE achieved using the features selected by the GA-based algorithm is smaller than that attained by using all features as the input. To be more specified, by using the GA-based, our ED-LSTM model reduces the MAE by 14.7% on average and 18.3% in the best case. The AE-BiLSTM and AC-LSTM models improve the MAE by 21.3% and 12.4% on average by leveraging the features selected by the GA-based algorithm. These results prove that the features selected by our proposed GA-based algorithm are not only effective for ED-LSTM but also the other prediction models.

We visualize the prediction results on the Hanoi dataset with different time-step values ahead in Fig. 11 and 12. In all cases, our proposed ED-LSTM model outperforms the AE-BiLSTM model. This is reflected by the fact that the ED-LSTM model's prediction results are much closer to the ground truth than the AE-BiLSTM model. In comparison between the ED-LSTM model and AC-LSTM model, it can be seen that the ED-LSTM model captures more peaks than the AC-LSTM model. By comparing between Fig. 11 and 12, we can see that the prediction results with the selected features from GA are closer to the ground-truth than the results when using all features. This result again proves the data shown in Tables 3 and 4, in which the MAE value of GA is always better than that obtained when using all the attributes.

To further demonstrate the effectiveness of the proposed model, we input the features selected by our GA-based algorithm into in the encoder-decoder model to predict PM2.5 one-month ahead. Because we have hourly data, we calculate the mean value of each day from 24-hour data. We fix the input length to 70 and vary the length of the output from 1 to 31 to see how the output length affects the model's accuracy. Figure 10 shows that the model produces low and stable MAEs as the number of time steps in the output varies. Furthermore, the last three test cases in the chart achieve the lowest MAE.

2) COMPARING ED-LSTM AND ST-DNN

In this section, we compare the performance of our proposed model with the ST-DNN model. Note that due to the impossibility of reproducing the ST-DNN results, we have copied the results from Fig. 23 in the ST-DNN paper [11]

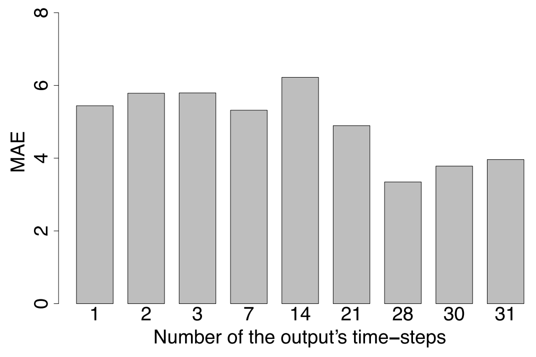


FIGURE 10. MAE of the proposed model with different output lengths.

TABLE 5. Comparing the MAE of the proposed ED-LSTM model and the ST-DNN model (using the features proposed by [11]).

Case	ED-LSTM (proposed method)	ST-DNN (A+L+C)	Improvement
l = 48, h = 1	2.454	2.881	14.82%
l = 48, h = 2	3.930	5.362	26.71%
l = 48, h = 3	3.907	6.524	40.11%
l = 48, h = 4	4.844	7.364	34.22%
l = 48, h = 5	4.604	7.923	41.89%
l = 48, h = 6	5.142	8.821	41.71%

to the third column. The results are collected and shown in Table 5. ST-DNN (A+L+C) indicates the best combination of adaptive artificial neural networks (A), long short-term memory (L), and convolutional neural network (C) in ST-DNN. Our model outperforms ST-DNN under all experimental settings. The fourth column shows the performance gap between our proposed approach and ST-DNN, ranging from 14.82% to 41.89%. Interestingly, the performance gap tends to increase as we increase the time steps of the output.

To illustrate the accuracy of our prediction model, we visualize the details of the prediction results in Table 5 and Fig. 13. The figure includes six subfigures, each of which presents the prediction over time with a different number of time steps ahead. Moreover, we plot the ground truth in the subfigures for comparison. When the number of the output timesteps is small (i.e., Figures 13(a) and 13(b)), the predicted data accurately match the ground truth data. Even the peak points are successfully predicted. In the other figures, as the number of time steps ahead increases, the prediction becomes slightly less accurate than that in Fig. 13(a).

In summary, ED-LSTM outperforms the existing models in terms of prediction accuracy. The improvement comes from two reasons. The first one is introducing the GA-based feature selection algorithm, which helps determine the optimal feature combination. The second one is that the combination of encoder-decoder and LSTM network helps extract meaningful information from the input. On the other hand, ED-LSTM has a slightly higher computation time than the model with the lowest value. However, ED-LSTM's computation time is still sufficient for real-time prediction of PM2.5.

V. DISCUSSION

According to our GA-based feature selection methods, the feature combination that produces the best performance for predicting PM2.5 is *{wind speed, temperature, radiation, PM10}*. Therefore, these features have the greatest effect on PM2.5. Indeed, we have measured the correlation between PM2.5 and all the features using three methods: Spearman's correlation (SC), Pearson correlation (PC), and mutual information score (MIC). Table 6 represents the absolute values of SC, PC and MIC. The wind speed, temperature, radiation, and PM10 show high correlations with PM2.5, which is reflected by the high absolute values of SC, PC, and MIC.

TABLE 6. Correlation of features.

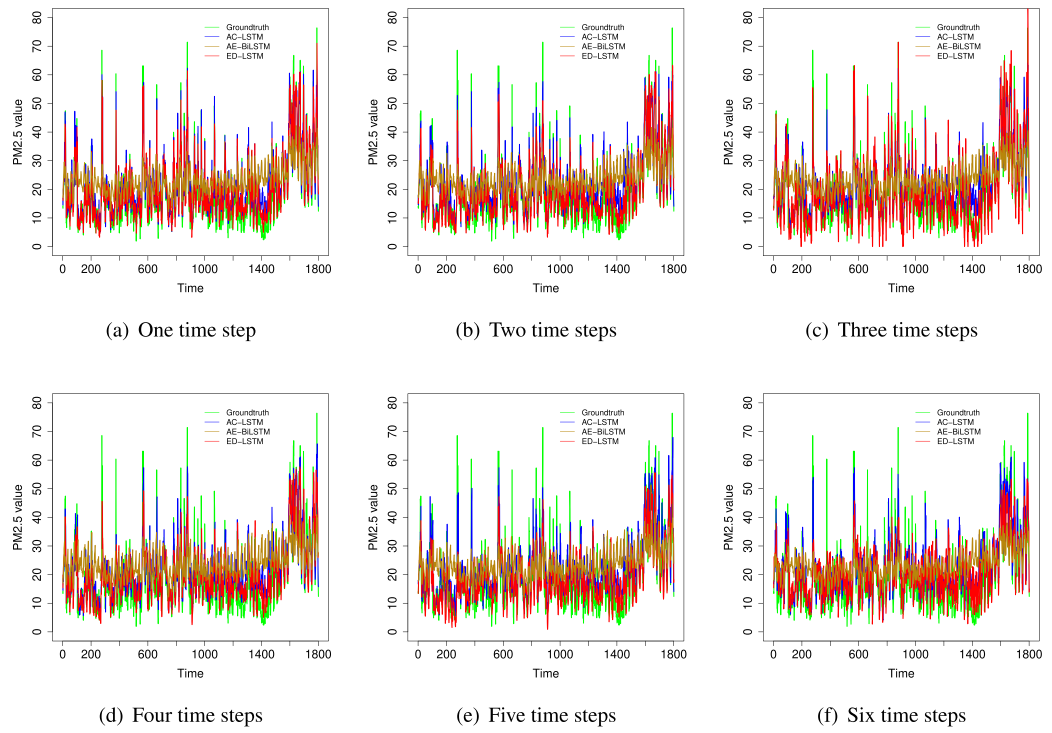
	|SC|	|PC|	|MIC|
Time	0.345790	0.292338	0.992296
Month	0.184097	0.138431	0.401139
Day	0.003414	0.004860	0.511531
Year	0.295231	0.258422	0.130763
Hour	0.062853	0.072817	0.480540
Wind speed	**0.263086**	**0.259349**	**0.989226**
Wind direction	0.009217	0.040491	0.989369
Temperature	**0.386038**	**0.391475**	**0.989201**
Relative humidity	0.047072	0.084301	0.734780
Barometer	0.354467	0.366005	0.882035
Radiation	**0.187909**	**0.214549**	**0.896700**
Inner temperature	0.402831	0.361091	0.985135
PM10	**0.924673**	**0.934090**	**0.994290**
PM1	0.979046	0.985997	0.994193

Our findings are consistent with the results of previous studies. In [23], the authors showed that temperature is positively correlated with PM2.5, and a threshold decides the correlation between wind speed and PM2.5. Specifically, when the wind speed is less than 3 m/s, it is negatively correlated with PM2.5, and beyond that point, it is positively correlated with PM2.5. The work in [24] also found that PM2.5 is affected the most by wind speed and temperature. The relationship between radiation and temperature with PM2.5 is studied in [25]. The authors in [25] showed that increasing temperature and decreasing radiation lead to increases in PM2.5. [26]–[28] have shown the that PM10 is strongly related to PM2.5.

In summary, our GA-based feature selection method has selected the optimal feature combination for predicting PM2.5. The optimal combination includes features that have a considerable effect on PM2.5.

VI. RELATED WORKS

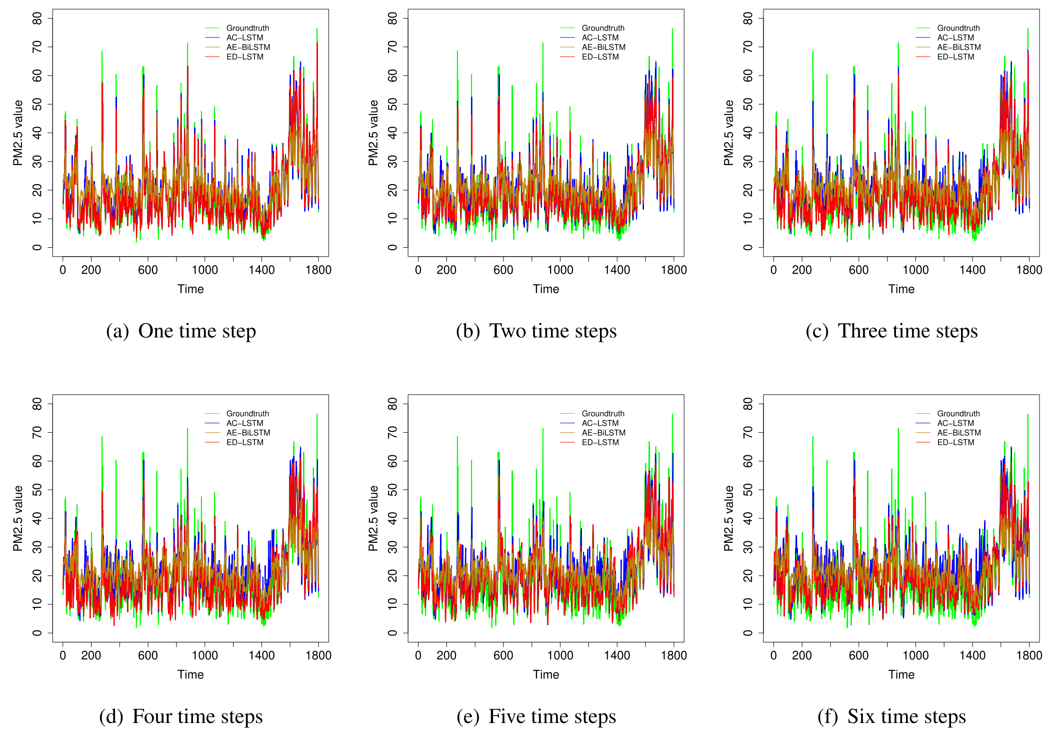
This section briefly reviews work related to PM2.5/air quality prediction models and GA-related methods. Kök *et al.* [3] predicted air quality using a deep learning model that includes three parts. In the first part, the training data are fed into an LSTM layer with an input sequence length of 8 and output length of 1. Second, the predicted data are labeled according to the daily air quality index (AQI) values. Finally, a decision unit is developed to map the observed data and predicted alarm situations. The model succeeds in employing LSTM with high accuracy, but the input and output are not



(a) One time step (b) Two time steps (c) Three time steps

(d) Four time steps (e) Five time steps (f) Six time steps

FIGURE 11. Comparison between models using Hanoi dataset with all features.

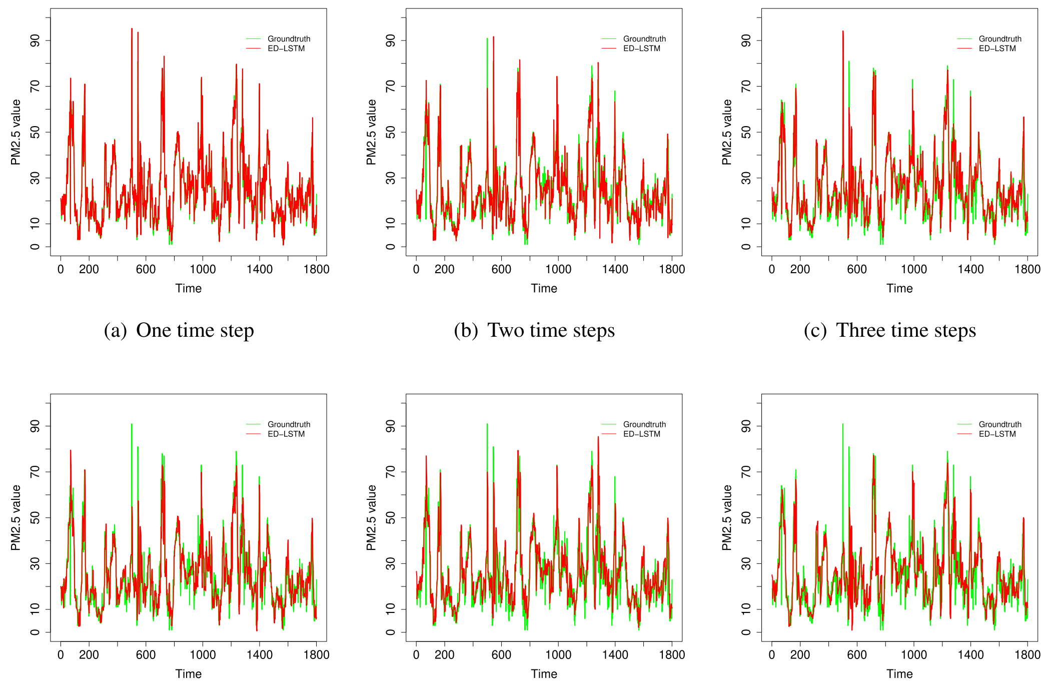


(a) One time step (b) Two time steps (c) Three time steps

(d) Four time steps (e) Five time steps (f) Six time steps

FIGURE 12. Comparison between models using Hanoi dataset with features selected by GA.

flexible. Several other models, such as ST-DNN [11], deep air learning (DAL) [10], and GC-DCRNN [15], exploit spatial data to formulate the relationships between spatial-temporal data. However, ST-DNN and GC-DCRNN do not identify the factors that affect air quality. Additionally, the models have a high time cost because of the preprocessing. The DAL



(a) One time step (b) Two time steps (c) Three time steps

(d) Four time steps (e) Five time steps (f) Six time steps

FIGURE 13. **Comparison between models using Taiwan dataset with features selected by [11].**

model performs feature selection during the training process by inserting a layer between the input layer and the second layer of the neural network. DAL, however, aims to discover the importance of different input features to the predictions, not to increase the prediction accuracy. The authors reveal the main relevant factors to the variation in air quality and provide proof to support air pollution prevention and control.

In [29], the authors use a sequence-to-sequence model to predict PM2.5. They feed all air pollutants into the model without concern for their appropriation. The main problem is that predicting all air pollutants will results in an "accumulation of errors." For example, when each feature's prediction results are inaccurate, it will negatively affect PM2.5 forecasting. Even if a feature does not affect PM2.5, it will cause the outcomes to be more inaccurate. In [21], the dataset consists of five features other than PM2.5; therefore, there are $5! = 120$ feature combinations. However, the authors do not describe how to select the optimal combination. In the experiments, the authors present results for seven combinations without explaining why these combinations are selected. L. Yan *et al.* use the E-D model to predict PM2.5 in [30]. The authors use all other features, including the monthly average PM2.5 concentration, daily average PM2.5 concentration, PM10 concentration, AQI, SO2, CO, NO2, O3, average temperature, humidity, pressure, and wind speed per hour per day. As there is no consideration of the optimal feature selection, the use of all features may complicate the model and degrade the prediction accuracy. In [31], the authors divide the features into groups and feed each group into a separate encoder. This approach has two serious problems. First, similar to other existing works, the use of all features, including features not related to PM2.5, degrades the prediction accuracy. Second, as they use a separate encoder for each feature group, the number of encoders is equal to the number of feature groups. Consequently, the model becomes complex when the number of features is extensive. Bo Zhang *et al.* in [21] conduct two training phases to predict PM2.5. In the former phase, the authors use an auto-encoder model to extract the relationship between multiple climate variables and the PM2.5 concentration. This phase is also responsible for compressing the input data to reduce the complexity. The output of the first phase is then fed into the second phase, which uses a Bi-LSTM network to predict the future PM2.5 based on the historical data. In [22], the authors take advantage of three types of data, including recent air pollutants, meteorology, and adjacent station's PM2.5. The authors first utilize a one-dimensional convolutional neural network to extract the air quality data's spatiotemporal correlation. The extracted feature vector is then fed into an LSTM layer with an attention mechanism to predict future air quality. In summary, the existing works use various variants of the ED model to perform PM2.5 prediction, but none consider feature selection.

The choice of hyperparameters generally has a significant impact on the performance of a deep-learning model. A nonexhaustive list of parameters includes the number of hidden layers, the number of neurons in each layer, the weights, and the learning rate. In previous works, these hyperparameters are typically chosen based on trial and evaluation, which is time consuming and does not guarantee optimal performance. A feasible approach to address this problem is to exploit search techniques to find the optimal settings. In such a context, the genetic algorithm (GA), which is a meta-heuristic search and optimization method, shows potential. In [32], the authors use GA to determine the optimal structure of a deep belief network for detecting different types of attacks in IoT networks. They then propose a GA-based approach to adaptively adjust the number of hidden layers and the number of neurons in each layer according to the attack type. Bouktif *et al.* in [33] exploit a GA to determine the optimal time lag and number of layers of an LSTM model for predicting the future electric load. In [34], the authors propose a deep long short-term memory (DLSTM) model to predict petroleum production. A GA is used to infer the optimal selection of the model hyperparameters, such as the number of epochs, the number of hidden neurons, and the lag. W. Liu *et al.* leverage support vector machine (SVM) for PM 2.5 prediction in [35]. They use the genetic algorithm (GA) and particle swarm optimization (PSO) to optimize the model's parameters, thereby improving the prediction accuracy. In [36], the authors propose a PM2.5 prediction model that utilizes multiple resolution data as input. An ensemble approach is leveraged to combine information retrieved from the low- and high-resolution data. The authors use the nondominated sorting genetic algorithm (NSGA-II) to optimize the ensemble weights. These works differ from our approach in that they leverage a GA to tune the parameters, not to optimize the input feature combination. Indeed, the models proposed in [35] and [36] take all the features as the input.

In this research, we apply a GA for air quality prediction model feature selection. The integration substantially improves the accuracy of PM2.5 prediction. Another distinct characteristic of our model is the ability to predict multiple steps ahead, as desired.

VII. CONCLUSION

In this paper, we have presented a novel prediction model for PM2.5 that combines PM2.5 with other air quality-related features. Our model utilizes a GA-based feature selection algorithm and an ED-LSTM prediction model. While the GA-based algorithm efficiently determines the near-optimal feature combination, the ED-LSTM model leverages LSTM units to loosen the restriction on the length of the input and output data. The experimental results indicate that our ED-LSTM model can improve the MAE up to 53.7% compared to that of state-of-the-art PM2.5 prediction models. Moreover, our proposed approach that includes the GA-based feature selection algorithm further improves the prediction accuracy by at least 13.7%.

REFERENCES

[1] C. A. Pope, R. T. Burnett, M. J. Thun, E. E. Calle, D. Krewski, K. Ito, and G. D. Thurston, "Lung cancer, cardiopulmonary mortality, and long-term exposure to fine particulate air pollution," *J. Amer. Med. Assoc.*, vol. 287, no. 9, pp. 1132–1141, 2002.

[2] S. Hochreiter and J. J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[3] I. Kok, M. U. Simsek, and S. Ozdemir, "A deep learning model for air quality prediction in smart cities," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2017, pp. 1983–1990.

[4] Y.-T. Tsai, Y.-R. Zeng, and Y.-S. Chang, "Air pollution forecasting using RNN with LSTM," in *Proc. IEEE 16th Int. Conf Dependable, Autonomic Secure Comput., 16th Int. Conf. Pervasive Intell. Comput., 4th Int. Conf Big Data Intell. Comput. Cyber Sci. Technol. Congr. (DASC/PiCom/DataCom/CyberSciTech)*, Aug. 2018, pp. 1074–1079.

[5] Y. Zhou, F.-J. Chang, L.-C. Chang, I.-F. Kao, and Y.-S. Wang, "Explore a deep learning multi-output neural network for regional multi-step-ahead air quality forecasts," *J. Cleaner Prod.*, vol. 209, pp. 134–145, Feb. 2019.

[6] Y. Qi, Q. Li, H. Karimian, and D. Liu, "A hybrid model for spatiotemporal forecasting of $PM_{2.5}$ based on graph convolutional neural network and long short-term memory," *Sci. Total Environ.*, vol. 664, pp. 1–10, May 2019.

[7] C. Wen, S. Liu, X. Yao, L. Peng, X. Li, Y. Hu, and T. Chi, "A novel spatiotemporal convolutional long short-term neural network for air pollution prediction," *Sci. Total Environ.*, vol. 654, pp. 1091–1099, Mar. 2019.

[8] J. Ma, Z. Li, J. C. P. Cheng, Y. Ding, C. Lin, and Z. Xu, "Air quality prediction at new stations using spatially transferred bi-directional long short-term memory network," *Sci. Total Environ.*, vol. 705, Feb. 2020, Art. no. 135771.

[9] J. Wang, P. Du, Y. Hao, X. Ma, T. Niu, and W. Yang, "An innovative hybrid model based on outlier detection and correction algorithm and heuristic intelligent optimization algorithm for daily air quality index forecasting," *J. Environ. Manage.*, vol. 255, Feb. 2020, Art. no. 109855.

[10] Z. Qi, T. Wang, G. Song, W. Hu, X. Li, and Z. Zhang, "Deep air learning: Interpolation, prediction, and feature analysis of fine-grained air quality," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 12, pp. 2285–2297, Dec. 2018.

[11] P.-W. Soh, J.-W. Chang, and J.-W. Huang, "Adaptive deep learning-based air quality prediction model using the most relevant spatial-temporal relations," *IEEE Access*, vol. 6, pp. 38186–38199, 2018.

[12] *Hanoi Dataset*. Accessed: Nov. 2020. [Online]. Available: https://shorturl.at/ilZ26

[13] X. Shi and D.-Y. Yeung, "Machine learning for spatiotemporal sequence forecasting: A survey," 2018, *arXiv:1808.06865*. [Online]. Available: https://arxiv.org/abs/1808.06865

[14] M. Z. Joharestani, C. Cao, X. Ni, B. Bashir, and S. Talebiesfandarani, "$PM_{2.5}$ prediction based on random forest, XGBoost, and deep learning using multisource remote sensing data," *Atmosphere*, vol. 10, no. 7, p. 373, Jul. 2019.

[15] Y. Lin, N. Mago, Y. Gao, Y. Li, Y.-Y. Chiang, C. Shahabi, and J. L. Ambite, "Exploiting spatiotemporal patterns for accurate air quality forecasting using deep learning," in *Proc. 26th ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, Nov. 2018, pp. 359–368.

[16] J. Biesiada and W. Duch, "Feature selection for high-dimensional data—A Pearson redundancy based filter," in *Computer Recognition Systems 2*, vol. 45. Berlin, Germany: Springer, 2007, pp. 242–249.

[17] D. Panda, R. Ray, A. A. Abdullah, and S. R. Dash, "Predictive systems: Role of feature selection in prediction of heart disease," *J. Phys., Conf. Ser.*, vol. 1372, Nov. 2019, Art. no. 012074.

[18] R. Genuer, J.-M. Poggi, and C. Tuleau-Malot, "Variable selection using random forests," *Pattern Recognit. Lett.*, vol. 31, no. 14, pp. 2225–2236, Oct. 2010.

[19] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: https://arxiv.org/abs/1412.6980

[20] I. Sutskever, O. Vinyals, and V. Q. Le, "Sequence to sequence learning with neural networks," 2014, *arXiv:1409.3215*. [Online]. Available: https://arxiv.org/abs/1409.3215

[21] B. Zhang, H. Zhang, G. Zhao, and J. Lian, "Constructing a $PM_{2.5}$ concentration prediction model by combining auto-encoder with bi-LSTM neural networks," *Environ. Model. Softw.*, vol. 124, Feb. 2020, Art. no. 104600.

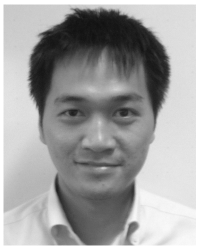
[22] S. Li, G. Xie, J. Ren, L. Guo, Y. Yang, and X. Xu, "Urban $PM_{2.5}$ concentration prediction via attention-based CNN–LSTM," *Appl. Sci.*, vol. 10, no. 6, p. 1953, Mar. 2020.

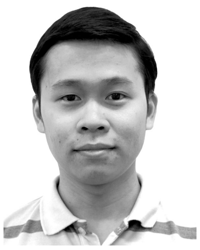
[23] J. Wang and S. Ogawa, "Effects of meteorological conditions on $PM_{2.5}$ concentrations in Nagasaki, Japan," *Int. J. Environ. Res. Public Health*, vol. 12, no. 8, pp. 9089–9101, 2015.

[24] P. D. Hien, V. T. Bac, H. C. Tham, D. D. Nhan, and L. D. Vinh, "Influence of meteorological conditions on $PM_{2.5}$ and $PM_{2.5-10}$ concentrations during the monsoon season in Hanoi, Vietnam," *Atmos. Environ.*, vol. 36, no. 21, pp. 3473–3484, Jul. 2002.
[25] M. Préndez, M. Egido, C. Tomás, J. Seco, A. Calvo, and H. Romero, "Correlation between solar radiation and total syspended particulate matter in Santiago, Chile—Preliminary results," *Atmos. Environ.*, vol. 29, no. 13, pp. 1543–1551, Jul. 1995.
[26] X. Zhou, Z. Cao, Y. Ma, L. Wang, R. Wu, and W. Wang, "Concentrations, correlations and chemical species of $PM_{2.5}/PM_{10}$ based on published data in China: Potential implications for the revised particulate standard," *Chemosphere*, vol. 144, pp. 518–526, Feb. 2016.
[27] E. Maraziotis and N. Marazioti, "Statistical analysis of inhalable (PM_{10}) and fine particles ($PM_{2.5}$) concentrations in urban region of Patras, Greece," *Global Nest J.*, vol. 10, no. 2, pp. 123–131, 2008.
[28] D. Zhao, H. Chen, E. Yu, and T. Luo, "$PM_{2.5}/PM_{10}$ ratios in eight economic regions and their relationship with meteorology in China," *Adv. Meteorol.*, vol. 2019, pp. 1–15, Feb. 2019.
[29] B. Liu, S. Yan, J. Li, G. Qu, Y. Li, J. Lang, and R. Gu, "A sequence-to-sequence air quality predictor based on the n-step recurrent prediction," *IEEE Access*, vol. 7, pp. 43331–43345, 2019.
[30] L. Yan, Y. Wu, L. Yan, and M. Zhou, "Encoder-decoder model for forecast of $PM_{2.5}$ concentration per hour," in *Proc. 1st Int. Cognit. Cities Conf. (IC3)*, Aug. 2018, pp. 45–50.
[31] Y. Zhang, Q. Lv, D. Gao, S. Shen, R. Dick, M. Hannigan, and Q. Liu, "Multi-group encoder-decoder networks to fuse heterogeneous data for next-day air quality prediction," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 4341–4347.
[32] Y. Zhang, P. Li, and X. Wang, "Intrusion detection for IoT based on improved genetic algorithm and deep belief network," *IEEE Access*, vol. 7, pp. 31711–31722, 2019.
[33] S. Bouktif, A. Fiaz, A. Ouni, and M. Serhani, "Optimal deep learning LSTM model for electric load forecasting using feature selection and genetic algorithm: Comparison with machine learning approaches," *Energies*, vol. 11, no. 7, p. 1636, Jun. 2018.
[34] A. Sagheer and M. Kotb, "Time series forecasting of petroleum production using deep LSTM recurrent networks," *Neurocomputing*, vol. 323, pp. 203–213, Jan. 2019.
[35] W. Liu, G. Guo, F. Chen, and Y. Chen, "Meteorological pattern analysis assisted daily $PM_{2.5}$ grades prediction using SVM optimized by PSO algorithm," *Atmos. Pollut. Res.*, vol. 10, no. 5, pp. 1482–1491, Sep. 2019.
[36] H. Liu, Z. Duan, and C. Chen, "A hybrid multi-resolution multi-objective ensemble model and its application for forecasting of daily $PM_{2.5}$ concentrations," *Inf. Sci.*, vol. 516, pp. 266–292, Apr. 2020.

MINH HIEU NGUYEN received the B.E. degree from the Hanoi University of Science and Technology, Hanoi, Vietnam, in 2019, where he is currently pursuing the master's degree with the School of Information and Communication Technology. His research interests include optimization, time-series prediction, machine learning, and deep learning.

PHI LE NGUYEN received the B.E. and M.S. degrees from The University of Tokyo, in 2007 and 2010, respectively, and the Ph.D. degree in informatics from The Graduate University for Advanced Studies, SOKENDAI, Tokyo, Japan, in 2019. She is currently an Assistant Professor with the School of Information and Communication, Hanoi University of Science and Technology, Vietnam. Her research interests include networks architectures, performance analysis and optimization in the Internet of Thing networks, mobile edge computing networks, and data mining.


KIEN NGUYEN (Senior Member, IEEE) received the B.E. degree in electronics and telecommunication from the Hanoi University of Science and Technology (HUST), Vietnam, in 2004, the Ph.D. degree in informatics from The Graduate University for Advanced Studies, SOKENDAI, Tokyo, Japan, in 2012. He is currently working as an Assistant Professor with the Graduate School of Science and Engineering, Chiba University. He is also involved in IETF activities. He has published more than 100 publications in peer-reviewed journals and conferences, three patents, and several Internet drafts. His research interests include the Internet, the Internet of Things technologies, and wired and wireless communication. He is a member of IEICE.


VAN AN LE (Graduate Student Member, IEEE) received the B.S. degree in computer science and engineering from the Ho Chi Minh City University of Technology, Vietnam, in 2016. He is currently pursuing the Ph.D. degree with the Department of Informatics, The Graduate University for Advanced Studies, SOKENDAI, Tokyo, Japan. His research interests include networks management and control, networks traffic monitoring, and big data.

THANH-HUNG NGUYEN received the Ph.D. degree in computer science from the University of Grenoble. He is currently an Assistant Professor with the School of Information and Communication, Hanoi University of Science and Technology, Vietnam. His research interests include the modeling and verification of component-based systems.

YUSHENG JI (Senior Member, IEEE) received the B.E., M.E., and D.E. degrees in electrical engineering from The University of Tokyo. She joined the National Center for Science Information Systems, Japan (NACSIS), in 1990. She is currently a Professor with the National Institute of Informatics, Japan (NII), and The Graduate University for Advanced Studies, SOKENDAI, Tokyo, Japan. Her research interests include networks architectures, resource management, and performance analysis for wired and wireless communication networks.