International Journal of Environmental Science and Technology (2023) 20:2959–2982
https://doi.org/10.1007/s13762-023-04763-6

ORIGINAL PAPER

Genetic algorithm-based hyperparameter optimization of deep learning models for $PM_{2.5}$ time-series prediction

C. Erden[1,2]

Received: 22 June 2022 / Revised: 19 October 2022 / Accepted: 6 January 2023 / Published online: 25 January 2023

Abstract

Since air pollution negatively affects human health and causes serious diseases, accurate air pollution prediction is essential regarding environmental sustainability. Although conventional statistical and machine learning methods have been widely used for air quality forecasting, they have limitations in finding nonlinear relations and modeling sequential data. In recent years, deep learning methods such as long short-term memory, recurrent neural networks, and gated recurrent units have been successfully applied in several research areas, including time-series forecasting. In this study, deep learning algorithm is employed to predict the $PM_{2.5}$ dataset, including air pollutants (NO, NO_2, NO_X, O_3, $PM_{2.5}$, SO, and SO_2) and meteorological features (wind speed, wind direction, and air temperature) in Istanbul metropolitan. Deep learning algorithms have many hyperparameters such as learning and dropout rate, the number of hidden layers and units in each hidden layer, activation function, loss function, and optimizer that need to be optimized in order to achieve optimal training performance. Therefore, a genetic algorithm-based hyperparameter optimization approach is proposed to find the best parameter combination. The prediction results of deep learning algorithms are compared with default hyperparameters and random search algorithms to confirm the efficacy of the genetic algorithm approach. The proposed method outperforms the other configurations, with the MSE error reduced by 13.38% and 55.30% for testing performance, respectively. The experimental results revealed that genetic algorithms are promising and applicable in hyperparameter optimization of deep neural network models, especially in air quality forecasting.

Keywords Gated recurrent units · Genetic algorithms · Hyperparameter optimization · Long-short term memory · Particulate matter · Recurrent neural networks

Introduction

Air pollution is one of today's major environmental problems because of the increasing use of carbon dioxide (Chen et al. 2017). Air pollution seriously affects many human organs, and about 4.2 million people die annually from air pollution (WHO 2021). The main drivers of air pollution are particulate matter ($PM_{2.5}$ or PM_{10}), ozone (O_3), nitrogen (NO, NO_2, NO_X), carbon monoxide (CO), and sulfate elements (SO, SO_2). $PM_{2.5}$ refers to the proportion of fine particles in the air less than 2.5 μm (Geng et al. 2015). Since $PM_{2.5}$ is a microscopic particle compared with other pollutants, exposure to $PM_{2.5}$ increases the risk of heart attack, damages the lungs and bloodstream systems, and causes several diseases like heart disease, lung cancer, and pneumonia (Hooyberghs et al. 2005). Therefore, accurate modeling and predicting air pollutants, especially $PM_{2.5}$, is essential for protecting human health (Zheng et al. 2013). The difficulties in $PM_{2.5}$ predictions are given as follows: (i) the data have nonlinear relationships, (ii) it requires working with large data sets, and (iii) researchers must deal with noisy, missing, or incomplete data. Linear relationships can be found using early time-series statistical techniques like autoregressive integrated moving averages (ARIMA). However, they are inefficient for air quality forecasting (Meng et al. 2016; Santhosh et al. 2019). In recent years, air quality forecasting improved with memory characteristics of deep learning(DL)

Editorial responsibility: Samareh Mirkia.

✉ C. Erden
cerden@subu.edu.tr

[1] Faculty of Applied Sciences, Sakarya University of Applied Sciences, Sakarya, Turkey

[2] AI Research and Application Center, Sakarya University of Applied Sciences, Sakarya, Turkey

methods such as long short-term memory neural network (LSTM), gated recurrent unit (GRU), and recurrent neural network (RNN) by discovering nonlinear and uncertain relations in data (Bai et al. 2016; Jiang et al. 2015).

DL, a type of artificial neural network (ANN), has successful applications in several fields of research: image processing for disease detection (Bhattacharya et al. 2021; Cao et al. 2019), algorithmic trading in stock markets (Pricope 2021; Théate and Ernst 2021), natural language processing (Otter et al. 2020; Young et al. 2018), autonomous cars (Fayjie et al. 2018; Possatti et al. 2019). Besides, DL has also been applied for time-series forecasting problems in which future data is predicted by discovering relationships between historical data. On the other hand, to obtain more accurate training performance related to selecting the DL method and its parameters accurately. Those selections are made considering the data's size and characteristics and the studied problem's machine learning class. In addition to the selected algorithm, having an acceptable learning performance (more accurate, more reliable, more realistic) also highly depends on the selected parameters of the model (Cooney et al. 2020).

In machine learning models, there are two types of parameters. One is model parameters that reflect data characteristics. The second is hyperparameters, which affect learning quality and algorithm performance and do not change during training. In comparison, the model parameters in artificial neural network (ANN) are weights; parameters such as the number of hidden layers, the number of neurons in the layers, activation function, and learning rate are known as hyperparameters. Hyperparameters are configured by the user, while the algorithm adapts the model parameters during learning (Bergstra et al. 2011). The suitable configuration of the hyperparameters impacts the performance of a learning algorithm. Therefore, the machine learning engineer configures the hyperparameters using different techniques that require careful study.

A skilled researcher can typically use the trial-and-error tuning method, but inexperienced researchers train with default hyperparameters (Thornton et al. 2013). In one tuning method, "exhaustive search," the researcher tries every combination in the predefined hyperparameter search space. However, an exhaustive search is not applicable and efficient in the case of a high number of combinations. To address these challenges, many researchers have been studying hyperparameter optimization (HPO) for three machine learning types: supervised, unsupervised, and reinforced learning (Hruschka et al. 2009). HPO is crucial in DL since it deals with large datasets and complex data relations (Yang and Shami 2020). Intelligent optimization techniques like meta-heuristics can achieve optimal hyperparameters by scanning the search space effectively. Meta-heuristics have been widely utilized in three stages of machine learning: (i) feature selection in the data preprocessing (Hancer et al. 2015), (ii) HPO in the learning stage, and (iii) in the back-propagation process (Telikani et al. 2022).

Hybrid deep learning models are presented to improve the effectiveness of individual deep learning models. Those studies can be given under three groups: deep learning with machine learning, optimization algorithms, and multiple or hybrid deep learning methods. In a recent study, Zaini et al. (2022) mentioned the limited hybrid studies in deep learning for air quality forecasting. It was emphasized that metaheuristics and deep learning methods should be combined. Therefore, this study utilized a genetic algorithm-based method to optimize the hyperparameters of LSTM, GRU, and RNN deep-learning neural models. The study aims to find the optimal combination of a set of hyperparameters in a defined search space without the researcher's experience requirement. Hourly $PM_{2.5}$ data in Istanbul/Turkey and meteorological data from 2015 to 2019 were used for the experimental study. After the data preprocessing phase on the dataset, the effectiveness of the deep learning algorithms determined in the modeling phase was improved with different hyperparameters. The problem target is continuous data, so regression metrics such as mean squared error (MSE), mean absolute error (MAE), and root-mean-square error (RMSE) were employed. The main contributions of this study can be summarized as follows: (1) An applicable genetic algorithm-based HPO technique has been presented for deep neural network models to achieve high prediction performance on the $PM_{2.5}$ data. (2) The selected data is an actual data set taken from Istanbul metropolitan, consisting of many incomplete, incorrect, or missing data and outliers. The preprocessing to prepare data for the deep learning model with data mining techniques has been employed. (3) The dataset includes air pollutants as well as meteorological data. The combined dataset was trained to predict the $PM_{2.5}$ concentration of the next hour by time windowing with 24-h historical data.

Related works

This section presents studies in two subsections: HPO methods for deep learning models and $PM_{2.5}$ time-series forecasting.

HPO for deep learning algorithms

Deep learning has recently been at the top of artificial intelligence studies. Compared to conventional machine learning techniques, deep neural networks can learn nonlinear relations in big data sets more accurately (Huang et al. 2019;

Mendoza et al. 2019; Yan and Han 2018). HPO studies have gained researchers' attention, especially in the 2020s (Guo et al. 2020; Zela et al. 2018).

In general, HPO algorithms are classified into different categories, such as:

- *Manual search:* It can only be used in tiny hyperparameter spaces. Trial-and-error is used to search for the best hyperparameters. However, it depends highly on the user's experience and domain knowledge and cannot be shown as an effective method because the number of attempts will be limited (Abreu 2019; Olof 2018).
- *Grid search:* The method of searching on the grid with a specified step size was developed by Lerman (1980). In the grid search, a result is reached by trying all possible combinations, which means a complete search. It has a design in which all factors are considered (Bergstra and Bengio 2012). It can operate in small search spaces and requires high processing power. Therefore, they can be used with few hyperparameters or other algorithms' initial phases.
- *Random search:* It works similarly to grid search, and there is no guarantee of an optimal solution. A random selection of all combinations is carried out. Bergstra and Bengio (2012), in their experimental benchmarking study, revealed that the best random search achieved better results in a shorter time than trial-and-error and grid search methods. Despite all these things, expanding the search space has led to searching for more intelligent methods in HPO.
- *Bayesian optimization and its variants* (Močkus 1975): It aims to reach the global optimum point with low experimentation. It generally gives better results than grid and random search and has a more efficient algorithm structure.
- *Tree parzen estimators:* give better results than Bayesian, thanks to conditional variables (Strobl et al. 2008).
- *Early-stopping policy***:** Since the HPO consumes much time and computational power, it is necessary to use resources effectively. In some cases, the limits of resources may be exceeded. In this case, to optimize the hyperparameters according to these limited resources like median stopping (Golovin et al. 2017; Liaw et al. 2018), curve fitting (Swersky et al. 2014) can be employed.
- *Meta-heuristics and population-based:* HPO is a computational optimization problem. Optimization generally means maximizing or minimizing the value of a given objective function under certain conditions. Optimization, in general, is quite challenging. Therefore, it can be classified in many respects (Burke et al. 2010). In one of them, optimization algorithms can be classified as algorithms that give the best and most approximate results. The most approximate algorithms are meta-heuristic algorithms such as genetic algorithms, simulated annealing, and Tabu search. According to another taxonomy, optimization problems are divided into two based on a single solution or a population-based basis. (Boussaïd et al. 2013). The general flowchart of population-based algorithms is given in Fig. 1 (Stork et al. 2020):

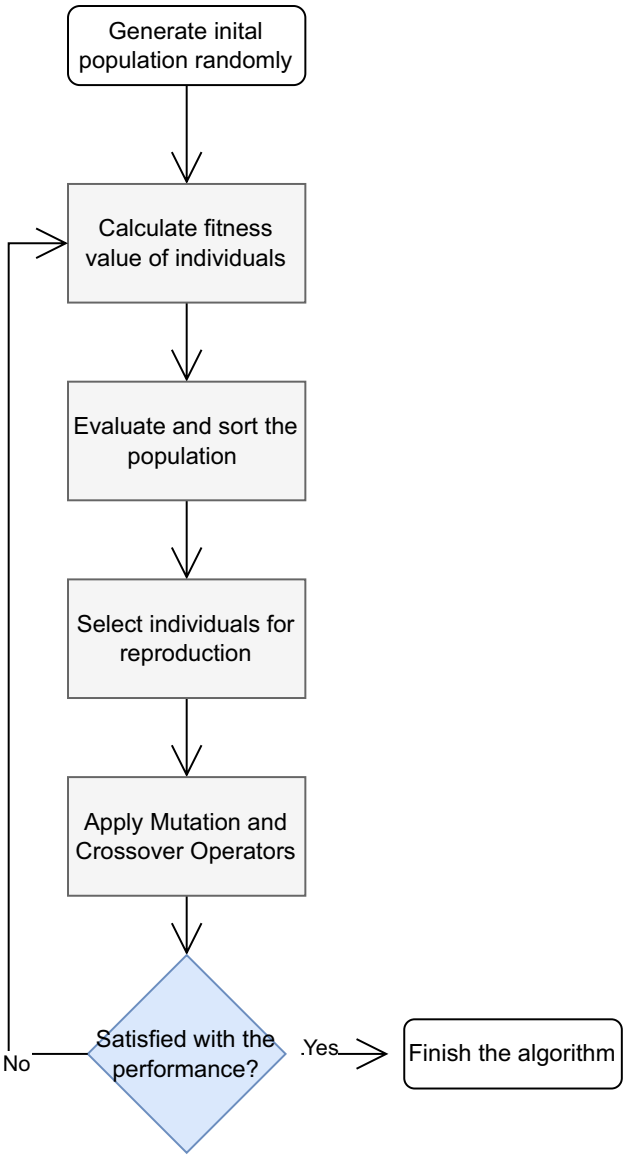

Fig. 1 A typical workflow of a population-based algorithm

As mentioned before, HPO is a promising challenge field for deep learning researchers. Researchers have proposed various algorithms for HPO. Kunang et al. (2021) studied

intrusion detection systems using the advantages of random and grid search algorithms. After the data preprocessing stage, one of the hyperparameter combinations obtained the best detection performance. Data sets, NSL-KDD (Tavallaee et al. 2009) and CSE-CIC-IDS2018 ("IDS 2018, " 2022), are used to demonstrate the validity of performance. Accuracy, detection rate, false alarm rate, precision, and F1-score values were used as performance metrics. The hyperparameters optimized in this study are the number of neurons and layers, the learning rate, initialization weights, the activation function, and the loss function. Firstly, the random search was used to scan search space randomly. Secondly, the size of the sample was reduced with a grid search. The number of combinations was calculated as 1962 for NSL-KDD and 1067 for CSE-CIC-IDS2018. Some hyperparameters were held constant such as epoch number is 30 for validation and 50 for training, and the Adam optimization algorithm is the optimization function. The learning rate value is the most influential variable on training performance, followed by the number of hidden layers and units in the hidden layer, respectively. Batch size does not influence performance; it instead affects learning speed. In conclusion, feature extraction with different autoencoder models (AE, SAE, and DAE) combined with the DNN model. Kaur et al. (2020) conducted an HPO study with a three-stage grid search technique in another study. For stage 1, hyperparameters are taken as the number of layers, the number of neurons in hidden layers, activation function, epochs = 100, batch size = 10, optimizers = SGD, and learning rate = 0.001. In Stage 2, SGD, RMSprop, Adagrad, Adam, and Nadam have been added to the optimizers list, and 1.0, 0.1, 0.01, 0.001, 0.0001, and 1e − 05 have been added to the learning rates list. In the third stage, 25, 50, 75, 100, 150, 175, and 200 were added to epochs, 20, 50, 100, 150, and 200 values to batch sizes.

Yaseen et al. (2019) conducted an HPO study to recognize different faces and a video analysis study. The study achieved better performance with the mathematical model developed for hyperparameters. The precision, recall, and F1 score are performance metrics. The hyperparameters are batch size, loss function, learning rate, the ratio of weights, and activation functions. In the developed study, the object recognition system with 97% accuracy was introduced, and the benefits of HPO were mentioned.

Alamri et al. (2022) presented a method combining the bees algorithm and Bayesian optimization to increase CNN performance for the Cifar10DataDir dataset. Taguchi method and ANOVA analysis have been employed to improve factor levels by hyperparameters. Although the training rate in the CIFAR10 dataset decreased from 92.68 to 90.53%, validation and test accuracy increased to 82.22% from 80.34%. In the digits and crack dataset, it has been shown that good results are obtained, although there are no significant advantages.

In case evolutionary algorithms are employed with deep learning, learning efficiency improves. Telikani et al. (2022) reviewed nearly 200 studies in the literature related to evolutionary computing. This review is an essential reference to the extent of the work on combining machine learning and evolutionary algorithms, including the genetic algorithm, ant colony optimization, and particle swarm optimization. Guo et al. (2019) developed a model using Taboo search and genetic algorithms for HPO on the well-known MNIST and flowers dataset. The model was compared with random search, Bayesian optimization, simulated annealing, TPE, and CMA_ES. It was concluded that the best result was produced with the developed model. The hyperparameters discussed are determined as learning rate, batch size, number of F1 units, dropout rate, and L2 weight decay. Table 1 summarizes the studies on different HPO and deep neural network approaches.

$PM_{2.5}$ prediction

$PM_{2.5}$ refers to the amount of particulate matter up to 2.5 μm in the air. $PM_{2.5}$, black carbon, and NO_x are the variables that significantly impact emission inventory in industry and city transportation (Gidhagen et al. 2021). Geng et al. (2021) reported that deaths caused by $PM_{2.5}$ increased by 23% between 2002 and 2017; therefore, it is crucial to predict it accurately. The high and negative impacts on human health have increased the importance of air quality forecasting. In previous studies, researchers forecasted real-time air quality using statistical methods (Salvador et al. 2004; Slini et al. 2006; Zhang et al. 2012), classic machine learning (Bozdağ et al. 2020; Choubin et al. 2020; Debry and Mallet 2014; Zickus et al. 2002) based on past air quality and meteorological data. Among the various machine learning and statistical techniques, deep learning has been promising for $PM_{2.5}$ forecasting in recent years. The first air quality forecasting study developed using deep learning was published in 2015 (Biancofiore et al. 2015). Following this study, numerous air quality studies have been developed using different deep neural networks such as CNN (Chae et al. 2021; Raimondi et al. 2005), LSTM (Kim and Oh 2018; Park et al. 2017), GRU (Becerra-Rico et al. 2020), and RNN (Feng et al. 2021; Nurcahyanto et al. 2022). Yang et al. (2021) studied 1-h to 24-h forecasting of spatiotemporal $PM_{2.5}$ concentrations

Table 1 Deep learning approaches with HPO studies

Reference	Optimization Algorithm	Deep Learning Methods	Performance Metrics	Application Area or Dataset	Hyperparameters
Kothandapani and Kuppuswamy (2020)	Differential evolution	LSTM	Accuracy (ACC)	MNIST, IMDB, and UCI HAR	Activation function (sigmoid, hyperbolic tangent (tanh) and rectified linear unit (ReLU))
Bibaeva (2018)	Evolutionary computation and local search	CNN	Error rate	Image recognition datasets	Activation function, pooling type, filter size, and step size
Kunang et al. (2021)	Random and grid search	DNN	ACC, detection rate, false alarm rate, precision, and F1-score	NSL-KDD and CSE-CIC-IDS2018	Number of hidden layers and nodes in each hidden layer, activation function, kernel initialization, and learning rate value
Yaseen et al. (2019)	Mathematical model	CNN	Precision, recall, and F1 score	Video data captured from different cameras	Batch size, loss function, learning rate, the ratio of weights, and activation functions
Alamri et al. (2022)	Bees algorithm and bayesian optimization	CNN	ACC	Cifar10DataDir	Section depth, initial learning rate, momentum, and regularization
Badan (2019)	Evolutionary algorithms	CNN	ACC	MNIST	kernelSize, outChannels, stride, and padding
Chung and Shin (2020)	GA	CNN	ACC	Stock market prediction KOSPI dataset	Number of kernels, kernel size, and window size
Popko and Weinstein (2016)	Fuzzy logic	CNN	ACC	Handwritten digits recognition	
Huang et al. (2011)	GA	Support Vector Machine	ACC	Fault diagnosis	Kernel parameter γ and penalty parameter C
Kaur et al. (2020)	Grid search optimization	DL	ACC and loss function	Parkinson's disease	Layers(Number), Neuron in a different layer (Number)
Yadav et al. (2020)	Trial–Error	LSTM	RMSE	Stock market	Number of Neurons
Guo et al. (2019)	Genetic algorithm and Tabu algorithm	CNN	ACC	MNIST and n Flower-5 dataset	Learning rate, batch size, number of F1 units, dropout rate, and L2 weight decay

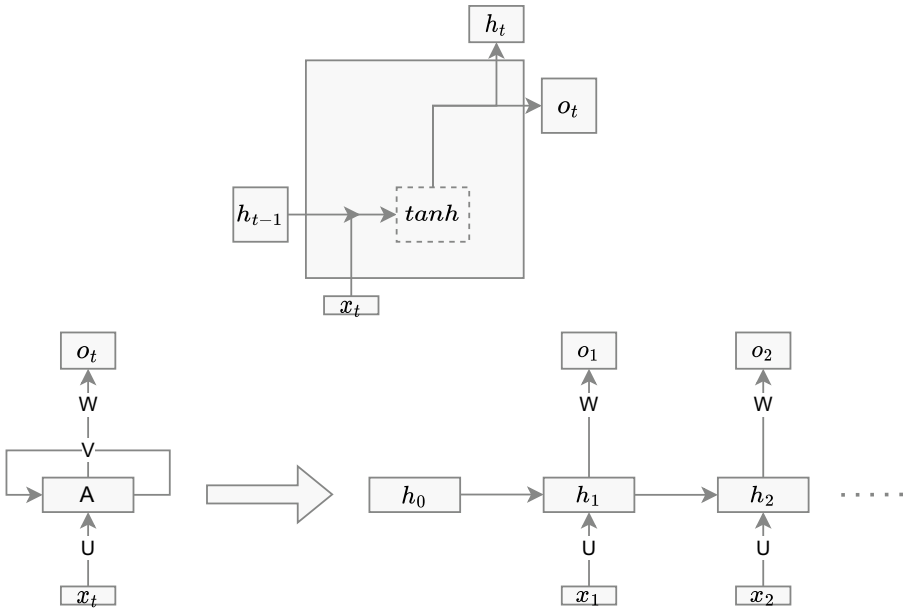

Fig. 2 Structure of a sample RNN

in the Beijing/China region using LSTM, CNN, and CNN-LSTM. It was shown that the best algorithm was CNN-LSTM, and the period was 12 h. Heydari et al. (2022) studied improving the optimization of the parameters of the deep learning model with particle swarm optimization and multiverse optimization algorithm. The data set consists of actual data in the Iran region, and as a result, it was found that the study model gave better results than the compared models. Yeo et al. (2021) proposed CNN and GRU models to predict $PM_{2.5}$ substance concentration for a dataset from 25 Seoul, South Korea stations. Akbal and Ünlü (2021) used deep learning methods to estimate the amount of particulate matter per day for Ankara/Turkey data and reached 94% accuracy. Z. Zhang et al. (2021) used bidirectional LSTM and variational mode decomposition to estimate the Chinese $PM_{2.5}$ data. According to this study, it was concluded that the developed model was stable.

Furthermore, the increase in the methods developed for air quality forecasting required review studies. Studies with statistical techniques like ARIMA (auto-regressive integrated moving average), SARIMA (seasonal ARIMA), artificial neural networks, decision trees, and k-nearest neighbor, as well as deep learning models like LSTM, were reviewed by Das et al. (2022). In another literature review, Zaini et al. (2022) stated that machine learning models are often preferred in air quality forecasting. They mentioned the application areas and limitations of different deep learning methods by addressing 110 recent articles. According to this study, it was determined that hybrid and ensemble models would be more effective for air quality prediction. Hybrid deep learning models include three types, deep learning with machine learning, deep learning with optimization algorithms, or combining different deep learning methods. As a result of their detailed literature study, the authors mentioned that the hybrid deep learning studies using optimization algorithms like Bayesian optimization or metaheuristic for HPO were limited in the literature. This study conducted air quality forecasting using deep learning methods and a metaheuristic algorithm.

Materials and methods

The main methods in this study are deep learning algorithms and genetic algorithms. This section introduces deep learning algorithms (RNN, LSTM, and GRU), which are useful for time-series forecasting and genetic algorithms, then gives details on the method applied and the dataset.

Deep learning algorithms

Recurrent neural networks (RNN)

RNN is a deep neural network and a variant of ANN that mimics human brain neural systems and can store previous information in the network. Unlike ANNs, RNNs have input, hidden, and output layers (Assaad et al. 2008). RNNs are generally applied to sequential data such as voice recognition (Graves 2012), natural language processing (Kim and Lee 2016; Park et al. 2018; Zoph et al. 2016), and time-series forecasting (Canizo et al. 2019; Tokgöz and Ünal 2018; Walid and Alamsyah 2017). Because the observation sequence substantially impacts the time-series analysis results, RNNs can efficiently discover data patterns while training. Figure 2 illustrates a typical RNN cell.

Nodes are the neural network cell at a single timestamp. X is the features set and X_t is the value of X at time t. O_t is the output and h_t is the hidden state. Input layers are connected to the hidden layer, and hidden layers are connected to the output layers. These connections are represented by the weight matrices U, W, and V:

$$h_t = \tanh\left(Wh_{t-1} + UX_t\right) \tag{1}$$

$$y_t = \tanh\left(Vh_t\right) \tag{2}$$

Since popular deep learning libraries Keras (Chollet 2015) and Tensorflow (Abadi et al. 2015) include LSTM, GRU, and RNN structures, the comparison studies are increased in the literature. RNNs can memorize up to 10-time steps and pay more attention to the more recent inputs (Graves 2012). Therefore, inputs earlier than 10-step can never affect the model, which causes a problem called vanishing gradient problem. LSTM and GRU methods have been developed as a variant of the RNN to tackle the vanishing gradient problem (Pascanu et al. 2013). In this study, multivariate time-series forecasting will be carried out on hourly $PM_{2.5}$ datasets. $PM_{2.5}$, the target variable, is affected by the current and previous states. For this reason, RNN and its variants, LSTM, and GRU deep learning models were preferred.

LSTM

LSTM is a variant of RNNs, consisting of three gates with memory characteristics in long and short periods thanks to RNN. It is specially designed to solve the vanishing gradient problem in modeling the sequential data by Hochreiter and Schmidhuber (1997). Therefore, LSTM can overcome long dependence on a neural network system and is suitable for time-series forecasting applications (Abdullah et al. 2019; Greff et al. 2016). A classic LSTM consists of three types of gates: input, forget, and output gates. The input gate controls the information from the memory cell; similarly, the output gate controls the output flows. Those gates are mathematically formulated as follows (Freeman et al. 2018):

$$f_t = \sigma\left(W_f\left(h_{t-1}, X_t\right) + B_f\right) \tag{3}$$

$$i_t = \sigma\left(W_i\left(h_{t-1}, X_t\right) + B_i\right) \tag{4}$$

$$o_t = \sigma\left(W_o\left(h_{t-1}, X_t\right) + B_o\right) \tag{5}$$

where f_t, i_t, o_t Denote the forget, input, and output gates, respectively. X_t Denotes the values that the feature receives at t time. H_t is the output cell of the previous. W denotes the weight matrix, and B denotes the bias term. σ denotes the sigmoid function.

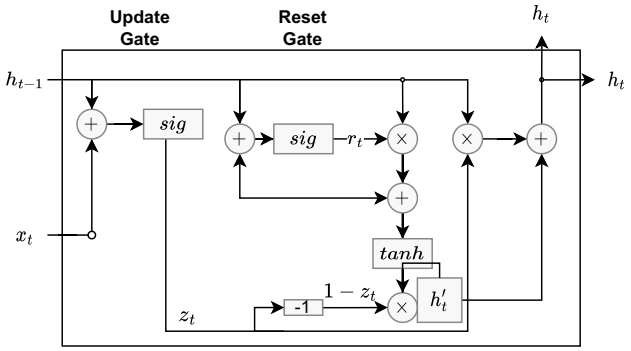

Fig. 3 Structure of a sample GRU

GRU

GRU unit has been designed as a simple variant of LSTMs. GRU has only updated (z) (similar to the input and forget gates in LSTM) and reset (r) gates. Information to remember is kept at the update gate, as shown in Fig. 3. Its lower complexity means that GRU operates with fewer parameters than LSTMs and requires less computational power during the training. The following equations are given for the gates of GRU:

$$r = \sigma\left(W_r\left(h_{t-1}, X_t\right) + U_r X_t\right) \tag{6}$$

$$z = \sigma\left(W_z\left(h_{t-1}, X_t\right) + U_z X_t\right) \tag{7}$$

$$c = \tanh\left(W_c\left(h_{t-1} * r\right) + U_c X_t\right) \tag{8}$$

$$h_t = (z*c) + \left((1 - z)*h_{t-1}\right) \tag{9}$$

Genetic algorithms (GA)

GA is a popular metaheuristic algorithm inspired by the idea of transferring biological development to algorithms (Holland 1992). GAs are generally used to achieve the best result from large solution spaces. GA has successful applications in many areas, including hyperparameter optimization for machine learning models (De Jong 1988; Grefenstette 1993). Chromosomes (individuals) represent a randomly generated

solution in the genetic algorithm and have a fitness value. Populations consist of more than one chromosome. Chromosomes ordered according to fitness value in the population are passed on to the next population by the selection, crossover, and mutation operators. Chromosomes with high fitness values transmit high information to the next population. Thus, the genetic properties of good chromosomes are preserved and the development becomes continuous.

Chromosome representation and determination of fitness function have a high impact on the algorithm's performance. In this study, the prediction performance of the deep learning algorithm was determined as the fitness value of the chromosomes. In the study steps, the data set consisting of meteorological and air quality data was first obtained. Then,

Table 2 List of GA parameters

Parameter	Value
Number of generation	50
Number of chromosomes in a population	10
Number of crossover chromosomes	6
Number of mutation chromosomes	4
Number of elitist chromosomes	1
Number of random chromosomes	1
Crossover rate	0.5
Mutation rate	0.2
Number of crossover points	1
Number of mutation points	3

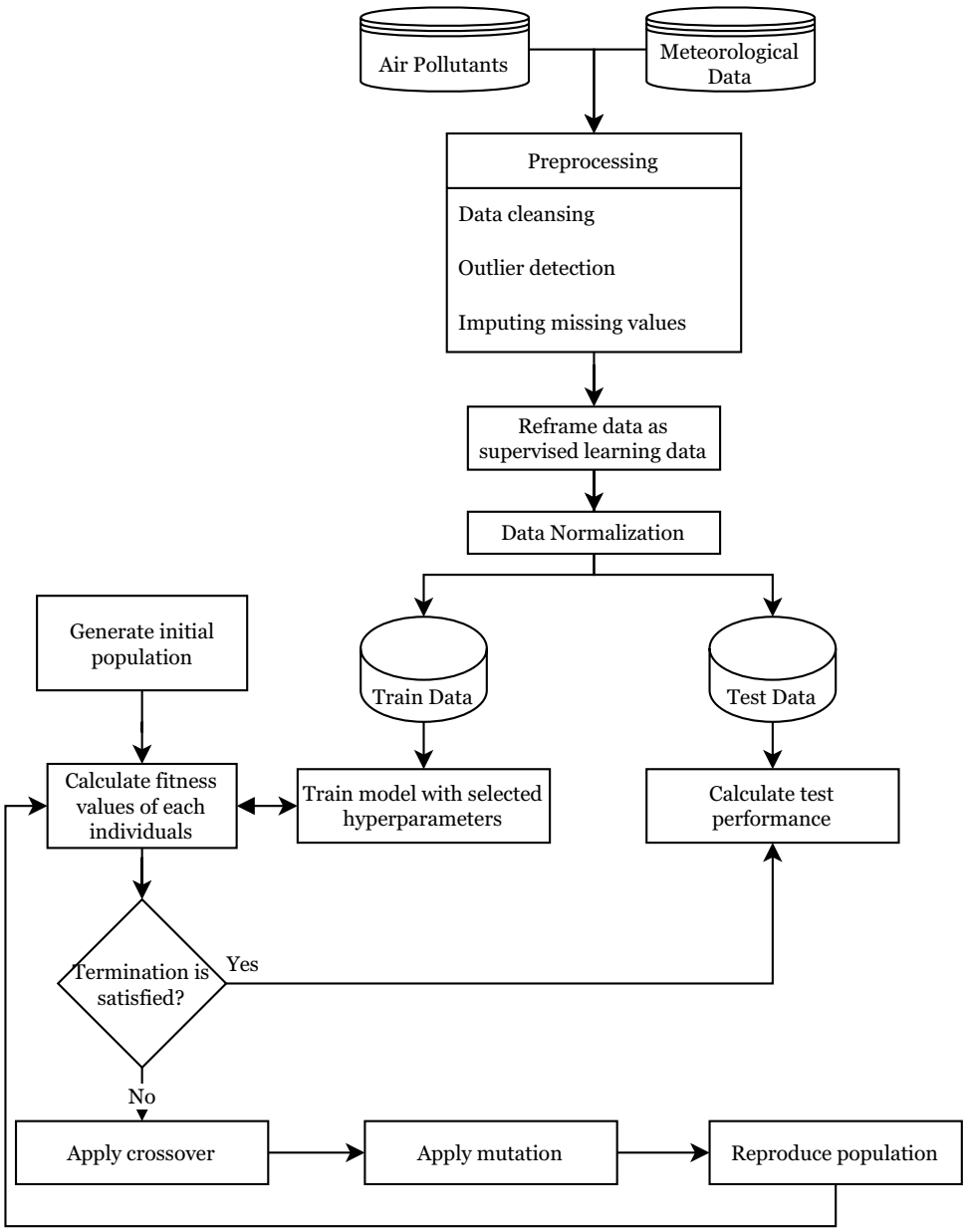

Fig. 4 Workflow of the proposed GA-deep learning method

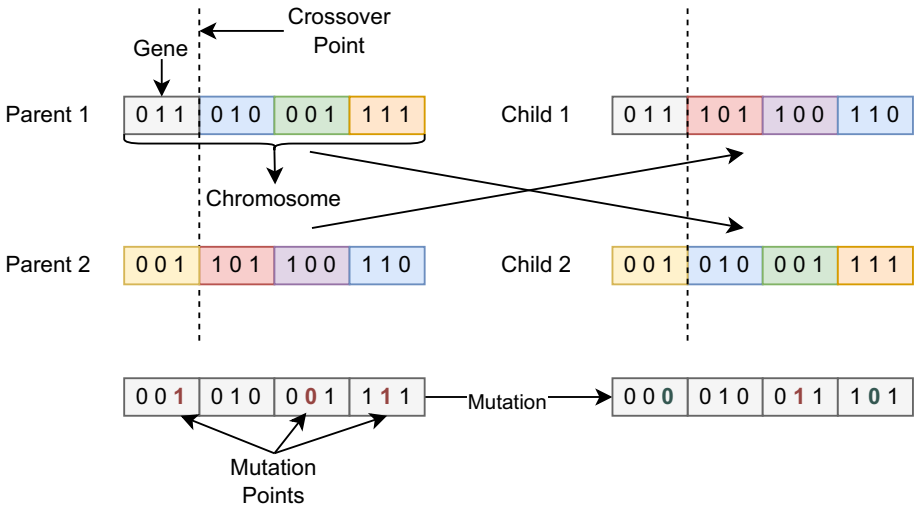

Fig. 5 Crossover and mutation operators

data preprocessing steps were performed. The performances of the training using the hyperparameter configuration of the chromosome generated by the genetic algorithm were analyzed. After assigning the performances as the fitness of the chromosome, the processes of the genetic algorithm were applied. The overall framework of the proposed GA to optimize the hyperparameters of deep learning algorithms is shown in Fig. 4.

Basic four steps were applied to employ a suitable GA design for predicting the $PM_{2.5}$ concentrations. The most appropriate configurations in each step after many trials have been reached. The processes of the GA algorithm can be given as follows:

1—Initialization: The random initial population is generated first in GA; the population size depends on the type of algorithm. Therefore, after a careful experimental study, the population size, generation size, crossover size and rate, mutation size and rate, and the number of crossover and mutation points were determined as given in Table 2. After the data preparation stages, ten chromosomes were randomly generated for the initial population. The fitness of each chromosome was calculated using the MSE performance of the test set. Also, it is necessary to test the model on the training data set and compare the test and training performances to calculate fitness. After the chromosomes were ranked according to their fitness value, the population's best, average, and worst chromosomes were generated, which will be used in the following steps.

2—Selection and Reproduction: The selection operator selects the higher-fitness chromosome to reproduce the next generation's new chromosomes. According to the roulette wheel selection technique, high-fitness chromosomes have more chance of being selected than lower-fitness chromosomes. In this study, a random individual is created in each generation for greater genetic diversity besides the roulette circle. Also, for elitist selection, the best individual is passed on to the next generation to avoid losing the best-performing combination in the previous population and ensure that the genetic algorithm is constantly at a minimum better point than the previous population. Since it is possible to eliminate the best individual during the algorithm, elitism can retain the best individuals. Finally, suppose there is more than one identical chromosome (having the same hyperparameters) in the population. In that case, these individuals should be eliminated, and instead, a new random chromosome is generated for the population. In this study, we generated 50 generations using the selection and reproduction approaches mentioned above.

3—Crossover: Crossover operator is the most important operator in GA. Crossover is performed at a certain rate called crossover rate after selection. A single cut point allows the mating pool's chromosomes to be crossed from a specific point. Gene change occurs between two individuals selected to ensure diversity in the population. In this study, for the crossover population, six chromosomes were selected according to the following possibilities: [0.3, 0.2, 0.15, 0.12, 0.10, 0.07, 0.03, 0.02, 0.006, 0.004]. The selection possibilities of each individual for crossover in each generation were held constant. Because in the roulette selection, the fitness difference between chromosomes will decrease toward the last generation.

4—Mutation: One chromosome gene is simply replaced by a possible value in the mutation step. It performs according to a predetermined mutation rate. A mutation operator has been developed to be trapped in local optima. The mutation population is formed using the multi-point mutation technique as in crossover. In this study, a chromosome is selected according to the previously mentioned possibilities for mutation population. The genes that need to be considered and controlled in the mutation operator are hidden layer size and deep learning algorithms. If the mutated gene is one of them, gene

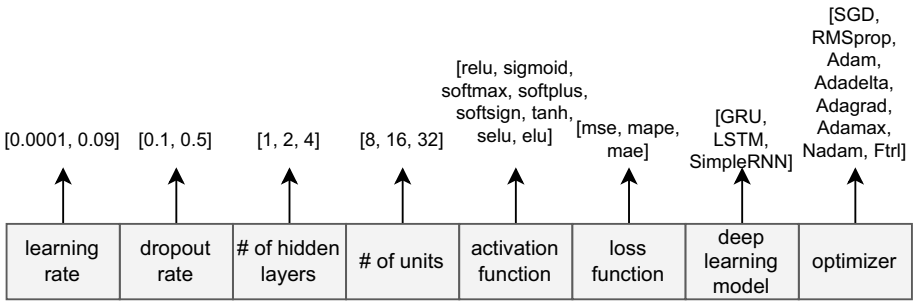

Fig. 6 Chromosome representation

control is carried out to protect the chromosome's feasibility. We employed single-point crossover and multi-point mutation for this study. Figure 5 illustrates both crossover and mutation operators.

The chromosome representation, which shows a potential solution for the problem, is essential in designing a GA. A chromosome must be able to find a possible solution for the problem. Figure 6 represents a chromosome structure and the search space of each gene. After crossover and mutation operators, a control mechanism is needed as the hidden layer size gene affects the dimension of the deep learning model gene. A correction is made if the hidden layer differs from the previous chromosome due to the crossover. Crossover is performed with a crossover rate of 0.6, and the crossover point is selected randomly. There are three random points in the mutation stage, which will be done with a probability of 0.5. GA parameters and their values are provided in Table 2. In order to reach the most appropriate values of crossover and mutation operators, we carefully used trial and error techniques.

GA is an algorithm that tries to find the best or near-best result. However, many difficulties can be experienced with the chromosomes. Some chromosomes may have "not a number (nan)" predictions in case of incompatible learning and dropout rates. It is not possible to calculate performance with nan values. Therefore, an inferior performance to the non-adaptive chromosomes was assigned. So, chromosomes with low fitness have a low chance of selection for the next generations.

Air quality dataset

The experimental studies' dataset was collected from the Istanbul/Kağıthane observation center between 2015-01-01 and 2019-11-30. Meteorological data is also included in the dataset. The raw dataset consists of 43,058 samples with $PM_{2.5}$, SO_2, NO, NO_2, NO_X, O_3, temperature, wind speed, relative humidity, and air pressure features. The first 70% of the data set is used as a train and 30% as a test set. The descriptive statistics are presented in Table 3. Figure 7 shows the time series of weekly observed $PM_{2.5}$ concentrations.

Air quality forecasting studies require working with data sets with complex and multidimensional characteristics. The variables used in forecasting PM2.5 are divided into air pollutants such as NO_2, NO_X, and SO_2 and meteorological factors such as temperature, wind speed, and humidity. When previous studies are examined, different input parameters are preferred in various studies (Chen et al. 2018; Li et al. 2017; Martínez et al. 2018). Variables affecting air quality forecasting for data sets from different sites may vary. However, air pollutants like SO_2, NO, NO_2, NO_X, and O_3 (Aksangür et al. 2022) and meteorological features like wind speed, humidity, and temperature are commonly used (Arain et al. 2007). Kristiani et al. (2021) proposed five deep-learning models

Table 3 Descriptive statistics of the dataset

	Unit	Count	Mean	Std	Min	25%	50%	75%	Max
$PM_{2.5}$	$\mu g/m^3$	43,057	26.2	17.0	0.0	14.5	20.8	32.3	86.7
SO_2	$\mu g/m^3$	43,057	5.9	5.1	0.0	2.3	3.9	7.8	24.3
NO	$\mu g/m^3$	43,057	33.9	33.3	0.0	11.2	20.5	44.8	142.3
NO_2	$\mu g/m^3$	43,057	36.5	26.4	0.0	17.2	29.4	49.2	144.9
NO_X	$\mu g/m^3$	43,057	93.2	80.5	0.0	37.8	64.3	121.1	370.7
O_3	$\mu g/m^3$	43,057	44.8	29.0	0.0	19.1	43.1	68.2	189.8
Temperature	℃	43,057	15.7	7.6	0.0	9.4	15.9	22.1	37.7
Wind Speed	m/s	43,057	1.9	1.2	0.0	0.9	1.7	2.7	7.3
Relative Humidity	%	43,057	77.5	16.8	12.5	66.7	80.1	91.1	100.0
Air Pressure	mbar	43,057	1006.1	6.5	979.8	1001.7	1005.4	1010.2	1033.3

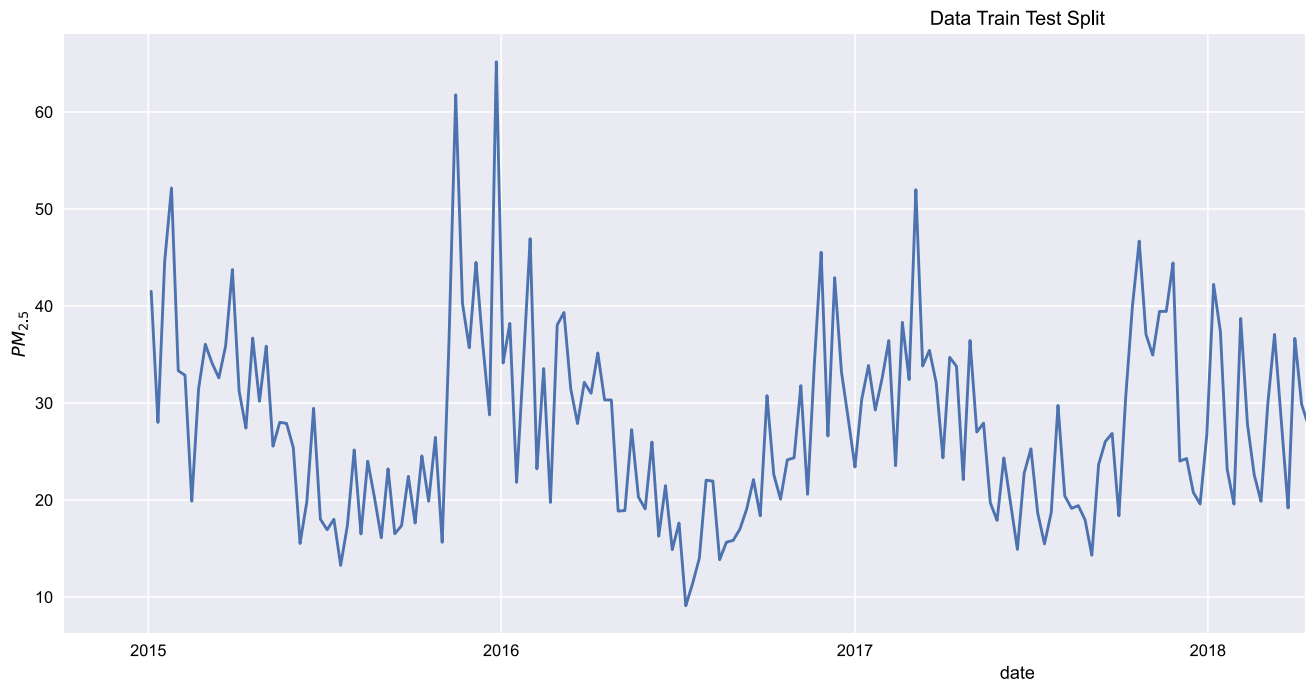

Fig. 7 Weekly PM2.5 concentrations in the period 2015–2019

to predict PM2.5 in Taichung City. The model's most successful result was achieved with 17 parameters such as $PM_{2.5}$, NO, NO_x, CO, NO_2, SO_2, wind direction, and wind speed. In another similar study, Choi et al. (2018) predicted PM2.5 using air pollutants and meteorological elements such as fine particulates (PM_{10}), ozone (O_3), carbon monoxide (CO), sulfur dioxide (SO_2), nitrogen dioxide (NO_2). Overall, nine variables selected from the literature and the heat map for the linear correlations between features are shown in Fig. 8. Although the high correlation in input parameters is not desirable in machine learning studies, no feature extraction was performed in this study due to the frequent use of the selected features in the literature. In addition, correlation matrix values explain pairwise comparisons, while in deep learning, multiple relationships will be analyzed.

Since the data set is actual, it contains many missing, incorrect, or noisy data. This study detected incorrect data, such as the amount of ozone (O_3) having a negative value, and recorded it as missing data. Then, the extreme data points were determined using the equations below and recorded as missing data. Some data are not recorded in the raw data, that is, missing data. Those missing data were imputed by the interpolation method.

$$\mathrm{IQR} = \mathrm{Q_3} - \mathrm{Q_1} \tag{10}$$

$$\text{lower} = Q_1 - 1.5 \times \text{IQR} \tag{11}$$

$$\text{upper} = Q_3 + 1.5 \times \text{IQR} \tag{12}$$

where interquartile range (IQR) is a measure of the spread, Q_1 denotes the first 25%, and Q_3 is the third 25% of the dataset (Dekking et al. 2005). Lower and upper are the boundaries where extreme points start. The min–max scaling method was used as the data scaling technique; data was scaled in the range 0–1 using the formula as follows:

$$X_{\text{std}} = \frac{\left(X - X_{\text{min}}\right)}{\left(X_{\text{max}} - X_{\text{min}}\right)} \tag{13}$$

$$X_{\text{scaled}} = X_{\text{std}} \times \left(X_{\text{max}} - X_{\text{min}}\right) + X_{\text{min}} \tag{14}$$

The dataset must be reframed into supervised learning to develop a multivariate time-series prediction model. In this study, the first 24-h data are used to predict the 25th $PM_{2.5}$ data, as shown in Fig. 9. $X_t^{(i)}$ is the value of i^{th} feature at time t y_{t+1} is the value of the output variable at time $t+1$, and k shows the number of features.

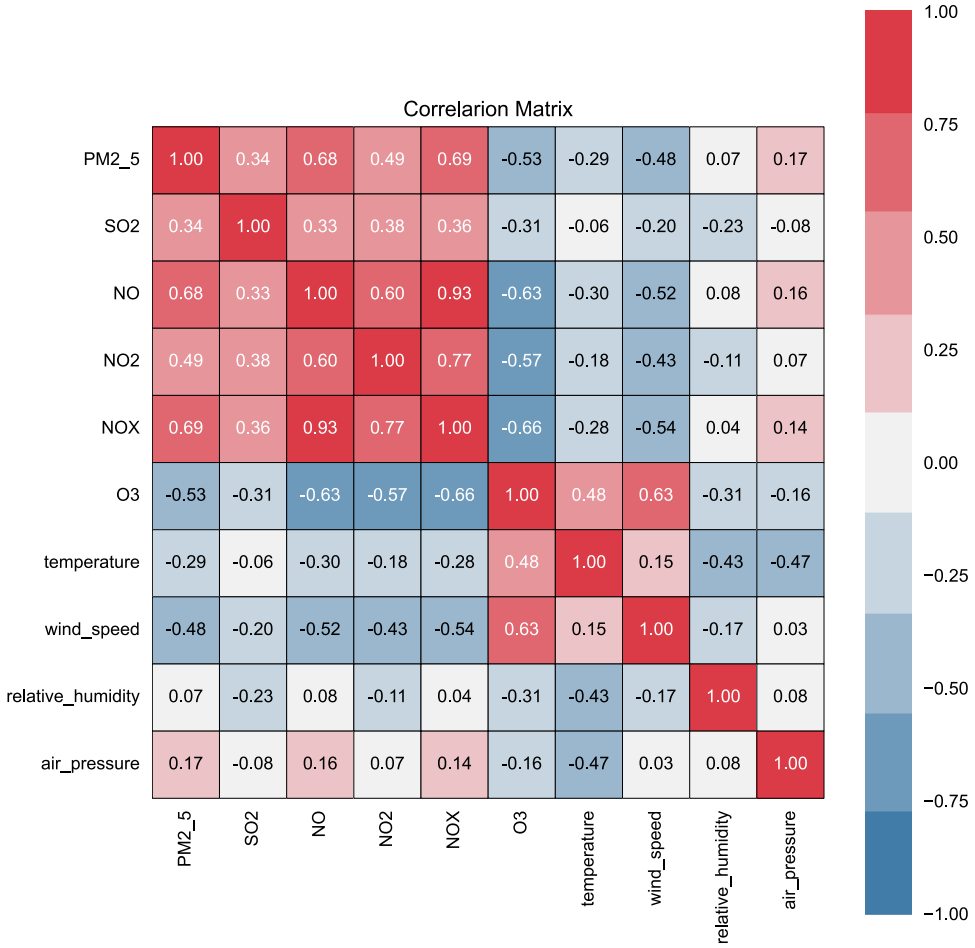

Fig. 8 Correlation matrix of the air quality and meteorological variables

Performance metrics

Mean square error (MSE), mean absolute error (MAE), root-mean-square error (RMSE), coefficient of determination (R^2), mean absolute percentage error (MAPE), maximum residual error (ME), and explained variance regression score (EVS) are the seven regression performance metrics that are used for the comparison and evaluation of the models. In order to show the performance of the genetic algorithm, one appropriate function is needed. Therefore, the MSE values of the test set were determined as a fitness function. So, the individual with a low MSE value has a better performance. All metrics used in the study are defined as follows:

$$\begin{array}{cccccc} & & \textbf{Inputs} & & & \textbf{Output} \\ X_{t-24}^{(1)} & X_{t-24}^{(2)} & \cdots & X_{t-24}^{(k-1)} & X_{t-24}^{(k)} & y_{t+1} \\ X_{t-23}^{(1)} & X_{t-23}^{(2)} & \cdots & X_{t-23}^{(k-1)} & X_{t-23}^{(k)} & \\ \vdots & \vdots & & \vdots & \vdots & \\ X_{t}^{(1)} & X_{t}^{(2)} & \cdots & X_{t}^{(k-1)} & X_{t}^{(k)} & \end{array}$$

Fig. 9 Reframe input and outputs

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2 \tag{15}$$

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \hat{y}_i\right| \tag{16}$$

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2}{n}} \tag{17}$$

$$R^2 = 1 - \frac{\sum\left|y_i - \hat{y}_i\right|}{\sum\left|y_i - \bar{y}\right|} \tag{18}$$

$$\mathrm{MAPE} = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{19}$$

where y_i $\hat{y}_i$ n and $\bar{y}$ are the actual and predicted observations, number of observations, and mean value.

Results and discussion

This study was coded in Python programming language with version 3.9. Data visualization, manipulation, and GA-based HPO deep learning models are implemented with Keras, pandas (Pandas 2021), NumPy (van der Walt et al. 2011), and Scikit-Learn (Pedregosa et al. 2011) libraries. The Papermill package is used for experimental studies of the genetic algorithm.

In the initial phase, the dataset consisting of hourly air pollutants and meteorological data was preprocessed. During the data preprocessing, incorrect outliers and erroneous data were recorded as missing or incomplete. Then, the missing data were filled with the interpolation method, and it was ensured that there was no missing data in the data set. The filled dataset is reframed for the 24-h supervised learning problem. After that, the genetic algorithm for selecting hyperparameters of deep artificial neural networks is adapted.

Selection of hyperparameters

The hyperparameters chosen for deep neural networks include categorical variables like activation and loss functions and optimizers, continuous variables like learning rate and dropout rate, and integers like the number of layers and units in each layer. Some of the basic definitions for the hyperparameters can be given as follows:

Table 4 Hyperparameters and ranges

Hyperparameter	Value or range	Data type
Epoch	60	Constant
Batch Size	120	Constant
Dropout rate	From 0.1 to 0.5	Continuous
Learning rate	From 0.0001 to 0.09	Continuous
Number of layers	[1, 2, 4]	Integer
Number of units	[8, 16, 32]	Integer
Activation function	[ReLu, Sigmoid, Softmax, Softplus, Softsign, Tanh, Selu, Elu]	Categorical
Loss function	[MSE, MAPE, MAE]	Categorical
Deep neural network	[GRU, LSTM, RNN]	Categorical
Optimizer	[SGD, RMSprop, Adam, Adadelta, Adagrad, Adamax, Nadam, Ftrl]	Categorical

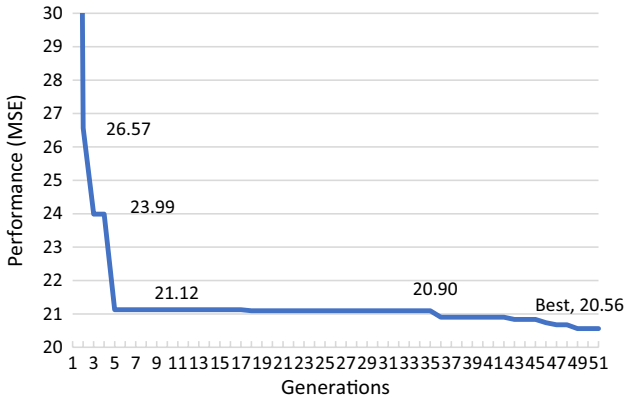

Fig. 10 Performance progression of the generations

- The learning rate is a crucial hyperparameter for adjusting weights in deep neural networks. A low learning rate can result in slow converge performance, while a high learning rate can cause the optimum point to be missed (Goodfellow et al. 2016). The learning rate typically takes values between 0.0001 and 0.1.
- The dropout rate means that some networks that have little effect on deep learning are removed from the system. It is a regularization technique that is used mainly to prevent overfitting problems.
- The number of layers specifies how many hidden layers will be in the network topology.
- The number of units specifies the number of deep neural networks in hidden layers.
- The activation function is a hyperparameter used to activate the deep neural network.
- The loss function shows the deep learning algorithm located in the hidden layer.
- Optimizer is the optimization algorithm to be used to determine the weights.

In this study, options of the Keras library were used while selecting hyperparameters (Chollet 2015). Accordingly, epoch and batch size hyperparameters were kept constant, and others varied within predefined ranges. Constant hyperparameters were tuned as a result of careful experimental studies. Other hyperparameters can be given as follows and summarized in Table 4 with their ranges and data types.

Genetic algorithm application

A genetic algorithm consisting of 50 generations developed on a randomly generated initial population according to the predetermined hyperparameter ranges was run, and the

Table 5 Hyperparameter settings of top five individuals

Individuals	Learning rate	Dropout rate	Hidden layer size	Unit size	Activation function	Loss function	DL model	Optimizer
#1 Individual	0.0804	0.1610	1	16	selu	MAE	LSTM	Adadelta
#2 Individual	0.0672	0.1380	1	16	selu	MAPE	LSTM	Adadelta
#3 Individual	0.0804	0.1610	1	16	selu	MAPE	LSTM	Adadelta
#4 Individual	0.0804	0.1610	1	16	selu	MAE	GRU	Adadelta
#5 Individual	0.0672	0.1380	1	16	selu	MAE	GRU	Adadelta

Table 6 Performance comparisons of the top five individuals

	Test							Train						
	EVS	MAE	MAPE	ME	MSE	R^2	RMSE	EVS	MAE	MAPE	ME	MSE	R^2	RMSE
#1	0.9050	2.9691	0.1482	50.5699	20.5611	0.9050	4.5344	0.9034	3.6275	0.1655	70.5674	30.9469	0.9021	5.5630
#2	0.9059	2.9359	0.1402	52.0015	20.6758	0.9044	4.5471	0.9057	3.6259	0.1631	70.7953	31.1210	0.9015	5.5786
#3	0.9066	2.9254	0.1370	52.9518	20.7392	0.9041	4.5540	0.9071	3.6119	0.1588	70.8928	30.9840	0.9019	5.5663
#4	0.9045	2.9118	0.1399	53.3592	20.7644	0.9040	4.5568	0.9051	3.5475	0.1571	69.6635	30.1735	0.9045	5.4930
#5	0.9043	2.9207	0.1395	52.2016	20.8362	0.9037	4.5647	0.9042	3.5749	0.1558	69.8281	30.5099	0.9034	5.5236

progress of the minimum MSE through the generations is shown in Fig. 10. The number of new individuals produced in 50 generations with a population of 10 is 261. The CPU time required is measured as 1236 min in total. The MSE metric for the test set was used as a fitness function. Since MSE values are spreading more widely, it offers a better perspective than RMSE. A lower MSE means better fitness performance. Figure 10 shows that reductions over 50 iterations were close to the best.

The comparative results of the top five individuals among the populations produced by the genetic algorithm are given in Tables 5 and 6. In addition, the fitness values of individuals with given hyperparameters and different performance criteria are presented. Accordingly, the best individuals are increased toward the last generations. The hidden layer number of the best individuals is 1, the activation function is 'selu' and the optimizer is Adadelta.

The fitness value of the best individual generated by the genetic algorithm was 20.5611. For the performance comparison of the genetic algorithm, training was carried out with random search and default parameters. The random search algorithm achieves the result with random trials among the hyperparameters. The default hyperparameters represent the default hyperparameter values that the Keras library provides for training. When the $PM_{2.5}$ time-series prediction model is trained with the hyperparameters of the best individual, test and training with the predicted values are shown in Figs. 11 and 12, respectively. The graphics are illustrated for approximately one month of test and training prediction data due to reading difficulties. According to the prediction results, the genetic algorithm has proven superior to others.

In addition, the relationship between observed and predicted values is shown using scatter plots. It is seen that a good match is achieved between the forecast values and the actual values. Since the $PM_{2.5}$ time-series forecasting problem is a regression problem, it is essential to check residuals after training a regression model. Residual plots are commonly used to evaluate regression performance. A residual value measures how far a regression line misses a data point vertically. The best fit of a set of data is a regression line. The lines can be considered averages; some data points will fit the line, while others will not (Klein and Dabney 2013). The residuals are displayed on the vertical axis, while the fitted values are displayed on the horizontal axis. As shown in Fig. 13f, residuals are gathered around the center regression line.

Learning curve plots are used to monitor the performances over epochs of the deep learning training. Underfitting and overfitting problems can be detected using learning curve plots. If the learning curves do not seem appropriate, learning parameters are reviewed. This study employs learning curve plots to show the performances obtained during deep learning training. Figure 14c shows the decrease in training and validation losses. Since a genetic algorithm optimizes the hyperparameters in the

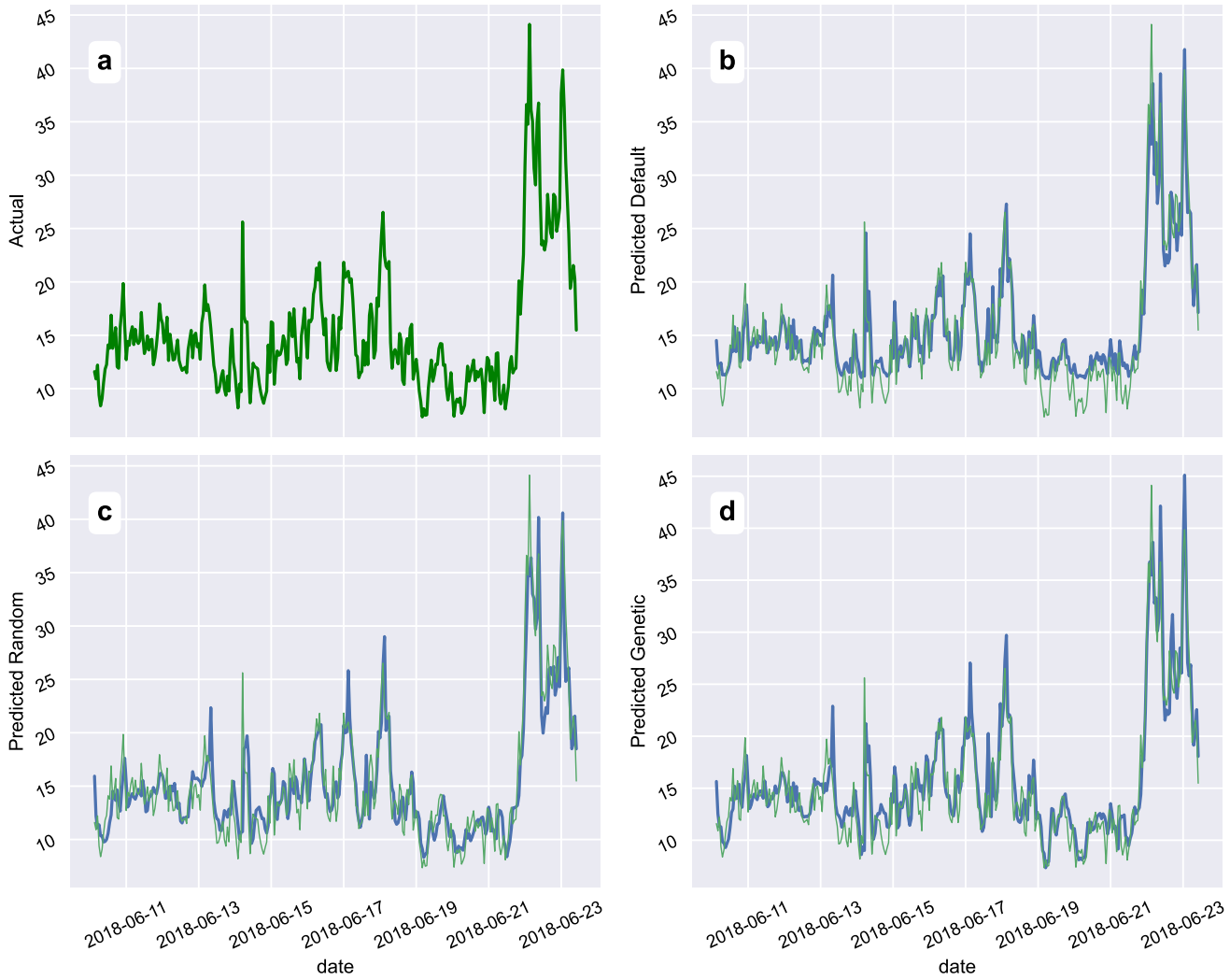

Fig. 11 Test data time-series plot slice of PM2.5 concentrations predicted by (**b**) default parameter setting, (**c**) random search setting, (**d**) genetic algorithm-based settings

developed model, the decreases in losses are continuous during training. Thus, underfitting and overfitting problems, which are the most critical problems of machine learning, are eliminated. In this study, the epoch size is 30, and losses converged after 30 epochs. Increasing the epoch size unnecessarily may result in overfitting or underfitting problems.

In order to test the efficiency of the genetic algorithm, a random search algorithm and default parameter model have been employed. Accordingly, it was determined whether the results of the obtained performance differed from those of 50 runs with hyperparameter combinations for consistency. The boxplots for the test and train sets obtained from 50 runs are visualized in Fig. 15. The genetic algorithm has a lower MSE average than other algorithms. The model's results working with default parameters have worse performance than the others.

Multivariate ANOVA (MANOVA) was employed to examine the statistically significant difference between the test and train performances of the algorithms. Wilks' Lambda, Lawley—Hotelling Trace, Pillai's Trace, and Roy's Largest Root were frequently used as test statistics. One criterion was not selected because there was no consensus in the literature on which one to use (Huberty and Olejnik 2006). The results of the MANOVA test are shared in Table 7.

As shown in Table 7, the p-values for the algorithm performances are significant at the 0.05 significance level, which means that the algorithm impacts the training performances. The Pillai's trace test statistics are statistically

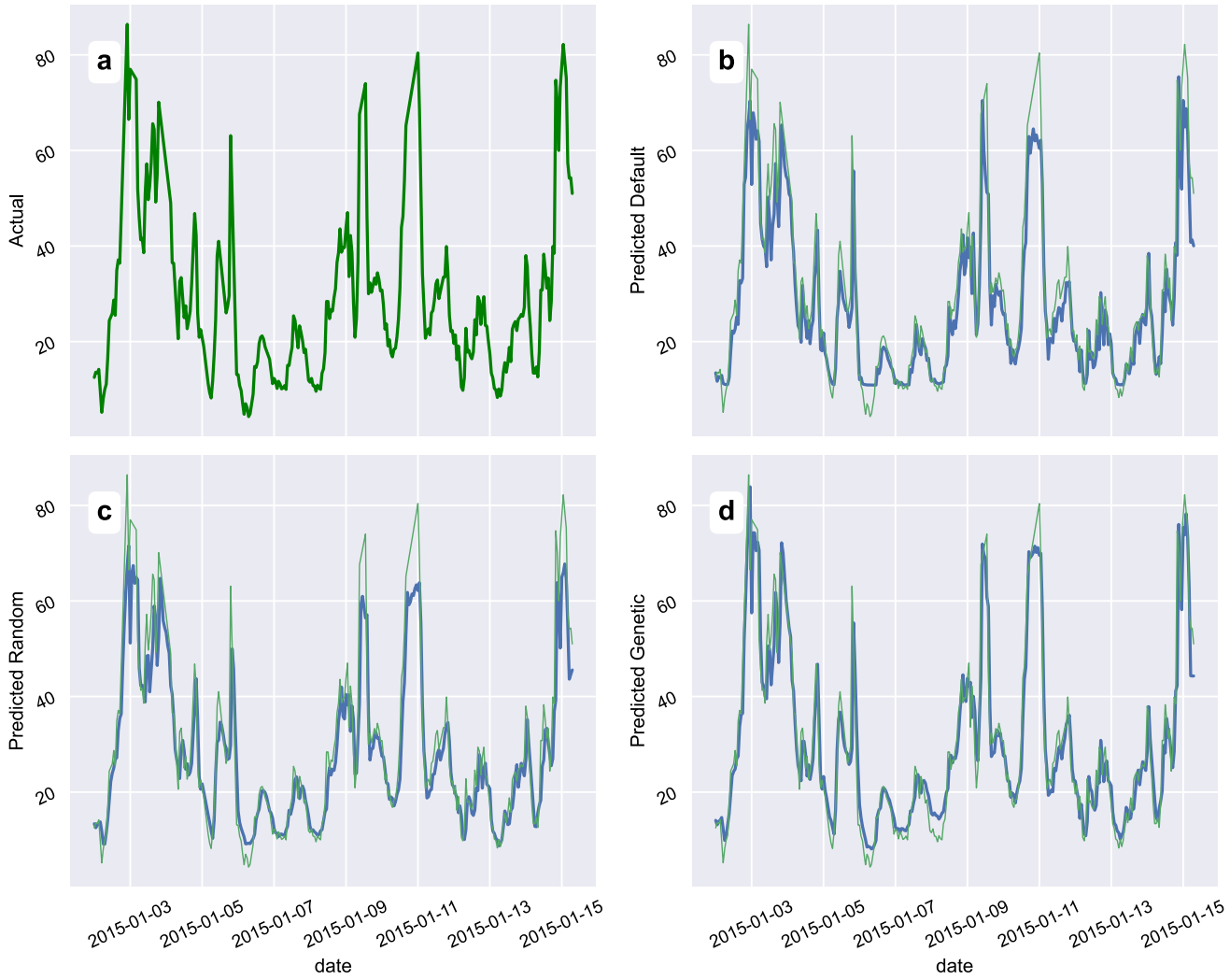

Fig. 12 Train data time-series plot slice of PM2.5 concentrations predicted by (**b**) default parameter setting, (**c**) random search setting, (**d**) genetic algorithm-based settings

significant [Pillai's Trace = 1.30, F (4, 294) = 137, $p < 0.001$] and indicate that the algorithm has a statistically significant association with both test and train performances. Following this, a post hoc test was conducted to determine which algorithm performed better, and the results are shared in Table 8.

Averages can reinforce the significant differences shown in the post hoc test. Table 9 shows the averages for which algorithm performance is better than the others. The averages showed a significant difference at the level of 0.05 (Sig. < = 0.05). According to these tests, the excellent performance value given by the genetic algorithm is shown by statistical tests for both test and train sets.

Conclusion

This study proposes the GA-based HPO method to improve the deep learning performance for $PM_{2.5}$ predictions. The proposed method is tested with a real air quality dataset observed in Istanbul/Turkey, between 2015 and 2019. Since Istanbul is a city with a high population and developed industry, there is a need for studies on air quality. The air quality indicators PM_{10} and $PM_{2.5}$ data are a multidimensional, complex time series and may contain incomplete and erroneous data due to sensor errors. For this reason, it is seen that deep learning algorithms are often used instead of traditional methods (statistical and machine learning).

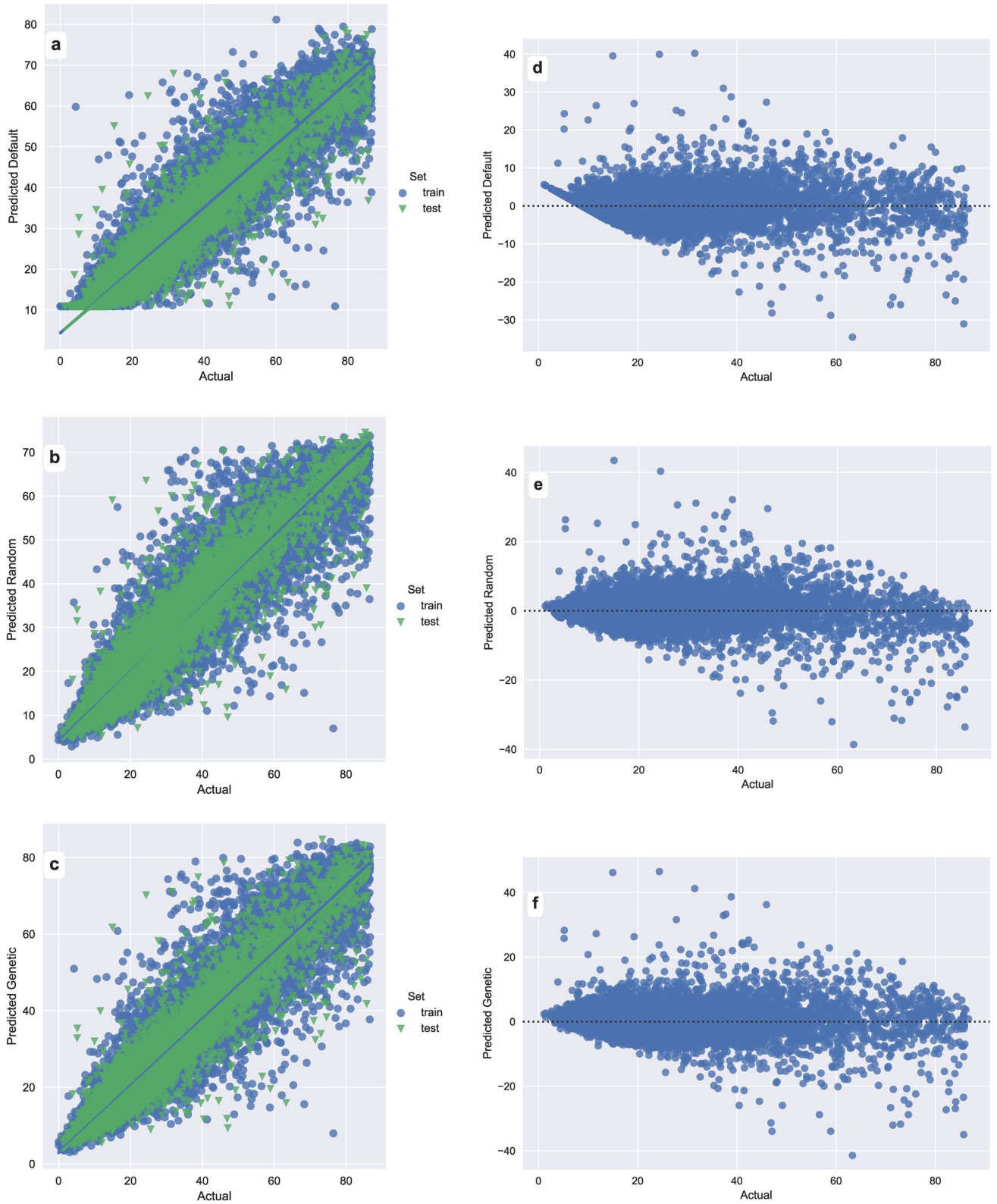

Fig. 13 Predicted versus actual values by (**a**) default setting, (**b**) random search setting, (**c**) genetic algorithm setting, residuals of (**d**) default setting, (**e**) random search setting, (**f**) genetic algorithm

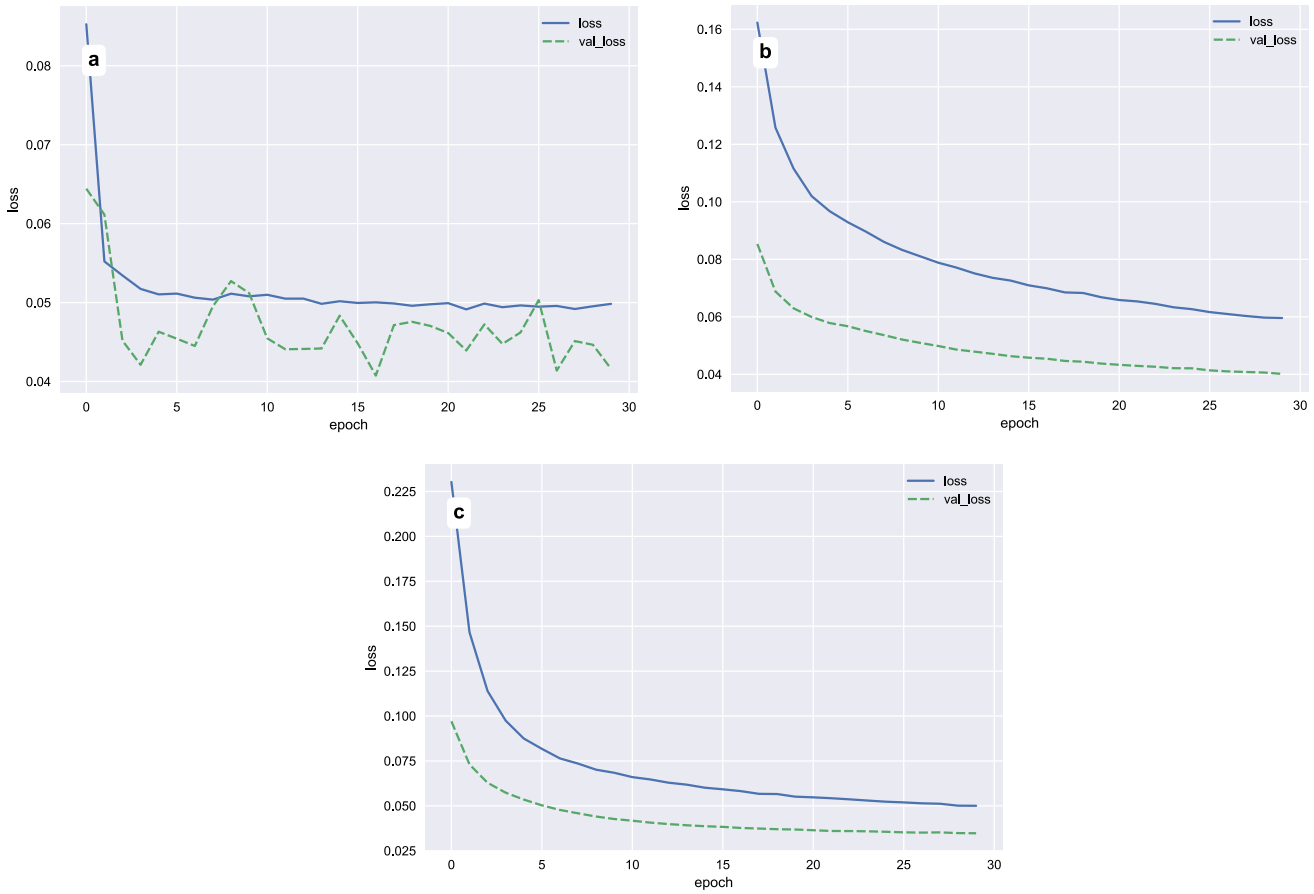

Fig. 14 Loss and validation loss (**a**) default setting, (**b**) random search setting, (**c**) genetic algorithm setting

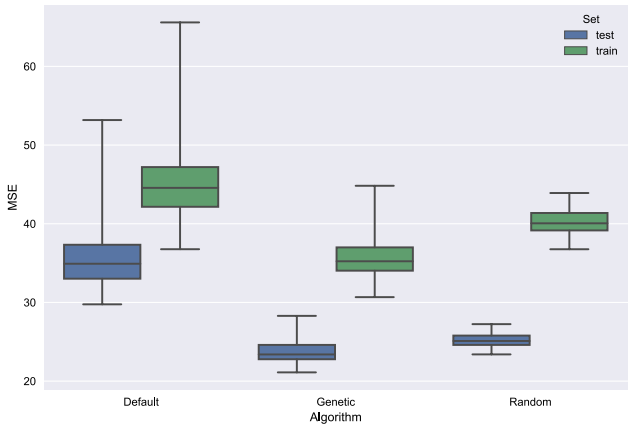

Fig. 15 Performance comparisons of the algorithms

Instead of working with the default hyperparameters, this study searched for the optimal hyperparameters of LSTM, GRU, and RNN deep neural network genetic algorithms. Each hyperparameter impacts learning performance, but the most important is related to network topology. An increase in the number of hidden layers makes discovering complex and non-linear relations possible while the computation time requirement increases. Since the air quality dataset consists of big data and multivariate relationships, it takes a long time to train a model. Metaheuristics like genetic algorithms can be applied to get enough good performance in a reasonable computational time. Nevertheless, genetic algorithm parameters like population size and the number of

Table 7 MANOVA tests for the algorithms

	Test statistic		DF		
Criterion		*F*	Num	Denom	*P*
Wilks'	0,05,005	253.307	4	292	0.000
Lawley-Hotelling	11.92065	432.123	4	290	0.000
Pillai's	1.30329	137.492	4	294	0.000
Roy's	11.29564				

$s=2\ m=-0.5\ n=72$

Table 8 Post hoc tests for test and train groups

Dependent variable				Std. error	Sig	95% confidence interval	
						Lower bound	Upper bound
Test	Tukey HSD	Genetic	Random	0.625	0.035	−3.053	−0.091
			Default	0.625	0.000	−14.048	−11.087
		Random	Genetic	0.625	0.035	0.091	3.053
			Default	0.625	0.000	−12.476	−9.515
		Default	Genetic	0.625	0.000	11.087	14.048
			Random	0.625	0.000	9.515	12.476
	LSD	Genetic	Random	0.625	0.013	−2.808	−0.336
			Default	0.625	0.000	−13.804	−11.332
		Random	Genetic	0.625	0.013	0.336	2.808
			Default	0.625	0.000	−12.231	−9.759
		Default	Genetic	0.625	0.000	11.332	13.804
			Random	0.625	0.000	9.759	12.231
Train	Tukey HSD	Genetic	Random	0.777	0.000	−6.648	−2.970
			Default	0.777	0.000	−12.492	−8.814
		Random	Genetic	0.777	0.000	2.970	6.648
			Default	0.777	0.000	−7.683	−4.005
		Default	Genetic	0.777	0.000	8.814	12.492
			Random	0.777	0.000	4.005	7.683
	LSD	Genetic	Random	0.777	0.000	−6.344	−3.274
			Default	0.777	0.000	−12.188	−9.118
		Random	Genetic	0.777	0.000	3.274	6.344
			Default	0.777	0.000	−7.379	−4.310
		Default	Genetic	0.777	0.000	9.118	12.188
			Random	0.777	0.000	4.310	7.379

Table 9 Homogeneous subsets for (a) train and (b) test set'

Group		*N*	Subset (a)		
			1	2	3
Tukey HSD[a,b]	Genetic	50	23.594		
	Random	50		25.166	
	Default	50			36.161
	Sig		1.000	1.000	1.000
The error term is Mean Square (Error) = 9.778					
Group		*N*	Subset (b)		
			1	2	3
Tukey HSD[a,b]	Genetic	50	35.447		
	Random	50		40.256	
	Default	50			46.100
	Sig		1.000	1.000	1.000
The error term is Mean Square(Error) = 15.081					

[a]Uses Harmonic Mean Sample Size = 50,000. [b]Alpha = .05

generations can affect computational time negatively unless they are well-tuned. When genetic algorithms are set up correctly, they can obtain better prediction performance without trapping local optima.

Table 10 presents the comparison findings between the studies published in the literature and the technique used in this study. While some of the studies estimated $PM_{2.5}$, some of them worked on PM_{10} prediction. $PM_{2.5}$ values have lower

Table 10 Performance comparisons of PM predictions in the literature

References	Region	Pollutant	Method	RMSE	R^2
Kristiani et al. (2021)	Taichung City	$PM_{2.5}$	LSTM	14.95	
Huang and Kuo (2018)	Beijing and Shanghai	$PM_{2.5}$	CNN-LSTM	24.23	
Li et al. (2020)	Beijing	$PM_{2.5}$	CNN-LSTM	17.93	
Ma et al. (2019)	Guangdong	$PM_{2.5}$	LSTM and Bi-LSTM	9.16	
Menares (2021)	Santiago de Chile	$PM_{2.5}$	LSTM		0.79
Samal (2021)	Beijing	$PM_{2.5}$	MTCAN	8.00	
Xu (2020)	Beijing-Tianjin-Hebei	$PM_{2.5}$	LSTM-based spatial predictor	28.77	0.77
Cekim (2020)	cities in Turkey	PM_{10}	ARIMA and SSA	20 avg	
Şahin et al. (2011)	Istanbul	PM_{10}	CNN and SPM	29.00	
Ceylan and Bulkan (2018)	Sakarya	PM_{10}	ANN	16.99	0.83
Kurnaz and Demir (2022)	Sakarya	PM_{10}	LSTM	14.09	
Aceves-Fernández et al. (2020)	Mexico City	PM_{10}	CNNs	18.77	0.62
Proposed model	Istanbul	$PM_{2.5}$	LSTM, GRU, RNN	4.53	0.91

Extreme learning machine group teaching optimization algorithm (ELM), multi-directional temporal convolutional artificial neural network (MTCAN), bi-directional LSTM (BiLSTM), singular spectrum analysis (SSA), and statistical persistence method (SPM)

averages than PM_{10} values, so a lower error is expected in $PM_{2.5}$ estimations compared to PM_{10}. When the literature studies' results are examined, the proposed model is a promising method for predicting air quality data with **4.53** RMSE and 0.91 R^2 performance.

There are many study areas worth studying in the future, such as:

- Other proven metaheuristic algorithms or new population-based algorithms, such as particle swarm optimization and ant colony optimization, can be applied.
- The developed method can be tried on different time-series data sets. It can be studied in larger hyperparameter spaces. The values of the hyperparameter range taken here are limited.
- The developed method can be tried on different time-series data sets.
- Meta-heuristic algorithms can also provide an alternative to the SGD algorithm for estimating weights in deep neural networks.
- The method developed for the dynamic prediction of $PM_{2.5}$ models can be advanced.

Code availability The datasets and Python code of the study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M (2015) TensorFlow: large-scale machine learning on heterogeneous systems

Abdullah S, Ismail M, Ahmed AN, Abdullah AM (2019) Forecasting particulate matter concentration using linear and non-linear approaches for air quality decision support. Atmosphere 10:667

Abreu S (2019) Automated architecture design for deep neural networks. arXiv:1908.10714 [cs, stat]

Aceves-Fernández MA, Domínguez-Guevara R, Pedraza-Ortega JC, Vargas-Soto JE (2020) Evaluation of key parameters using deep convolutional neural networks for airborne pollution (PM10) prediction. Discret Dyn Nature Soc. https://doi.org/10.1155/2020/2792481

Akbal Y, Ünlü KD (2021) A deep learning approach to model daily particular matter of Ankara: key features and forecasting. Int J Environ Sci Technol. https://doi.org/10.1007/s13762-021-03730-3

Aksangür İ, Eren B, Erden C (2022) Evaluation of data preprocessing and feature selection process for prediction of hourly PM10 concentration using long short-term memory models. Environ Pollut 311:119973. https://doi.org/10.1016/j.envpol.2022.119973

Alamri NMH, Packianather M, Bigot S (2022) Deep learning: parameter optimization using proposed novel hybrid bees Bayesian Convolutional Neural Network. Appl Artif Intel. https://doi.org/10.1080/08839514.2022.2031815

Arain MA, Blair R, Finkelstein N, Brook JR, Sahsuvaroglu T, Beckerman B, Zhang L, Jerrett M (2007) The use of wind fields in a land use regression model to predict air pollution concentrations for health exposure studies. Atmos Environ 41:3453–3464

Assaad M, Boné R, Cardot H (2008) A new boosting algorithm for improved time-series forecasting with recurrent neural networks. Inf Fusion 9:41–55

Badan F (2019) Evolutionary algorithms in convolutional neural network design 6

Bai Y, Li Y, Wang X, Xie J, Li C (2016) Air pollutants concentrations forecasting using back propagation neural network based on wavelet decomposition with meteorological conditions. Atmos Pollut Res 7:557–566. https://doi.org/10.1016/j.apr.2016.01.004

Becerra-Rico J, Aceves-Fernández MA, Esquivel-Escalante K, Pedraza-Ortega JC (2020) Airborne particle pollution predictive model using gated recurrent unit (GRU) deep neural networks. Earth Sci Inf 13:821–834

Bergstra J, Bardenet R, Bengio Y, Kégl B (2011) Algorithms for hyper-parameter optimization. In: Shawe-Taylor J, Zemel R, Bartlett P, Pereira F, Weinberger KQ (eds) Advances in neural ınformation processing systems. Curran Associates, Inc.

Bergstra J, Bengio Y (2012) Random search for hyper-parameter optimization. J Mach Learn Res 13:281

Bhattacharya S, Maddikunta PKR, Pham Q-V, Gadekallu TR, Chowdhary CL, Alazab M, Piran MJ (2021) Deep learning and medical image processing for coronavirus (COVID-19) pandemic: a survey. Sustain Cities Soc 65:102589

Biancofiore F, Verdecchia M, Di Carlo P, Tomassetti B, Aruffo E, Busilacchio M, Bianco S, Di Tommaso S, Colangeli C (2015) Analysis of surface ozone using a recurrent neural network. Sci Total Environ 514:379–387. https://doi.org/10.1016/j.scitotenv.2015.01.106

Bibaeva V (2018) Using metaheuristics for hyper-parameter optimization of convolutional neural networks, İn: 2018 IEEE 28th ınternational workshop on machine learning for signal processing (MLSP). Presented at the 2018 IEEE 28th ınternational workshop on machine learning for signal processing (MLSP), IEEE, Aalborg, pp 1–6. https://doi.org/10.1109/MLSP.2018.8516989

Boussaïd I, Lepagnot J, Siarry P (2013) A survey on optimization metaheuristics. Inf Sci 237:82–117

Bozdağ A, Dokuz Y, Gökçek ÖB (2020) Spatial prediction of PM10 concentration using machine learning algorithms in Ankara, Turkey. Environ Poll 263:114635

Burke EK, Hyde M, Kendall G, Ochoa G (2010) A classification of hyper-heuristic approaches. In: Gendreau M, Potvin J-Y (eds) Handbook of metaheuristics. Springer, pp 449–468

Canizo M, Triguero I, Conde A, Onieva E (2019) Multi-head CNN–RNN for multi-time series anomaly detection: an industrial case study. Neurocomputing 363:246–260

Cao Y, Liu Z, Zhang P, Zheng Y, Song Y, Cui L (2019) Deep learning methods for cardiovascular image. J Artif Intel Syst 1:96–109

Cekim HO (2020) Forecasting PM10 concentrations using time series models: a case of the most polluted cities in Turkey. Environ Sci Pollut Res 27:25612–25624. https://doi.org/10.1007/s11356-020-08164-x

Ceylan Z, Bulkan S (2018) Forecasting PM10 levels using ANN and MLR: a case study for Sakarya City. Global Nest J 20:281–290. https://doi.org/10.30955/gnj.002522

Chae S, Shin J, Kwon S, Lee S, Kang S, Lee D (2021) PM10 and PM2.5 real-time prediction models using an interpolated convolutional neural network. Sci Rep 11:1–9

Chen L-J, Ho Y-H, Lee H-C, Wu H-C, Liu H-M, Hsieh H-H, Huang Y-T, Lung S-CC (2017) An open framework for participatory PM2.5 monitoring in smart cities. IEEE Access 5:14441–14454. https://doi.org/10.1109/ACCESS.2017.2723919

Chen G, Wang Y, Li S, Cao W, Ren H, Knibbs LD, Abramson MJ, Guo Y (2018) Spatiotemporal patterns of PM10 concentrations over China during 2005–2016: a satellite-based estimation using the random forests approach. Environ Pollut 242:605–613

Choi J-E, Lee H, Song J (2018) Forecasting daily PM 10 concentrations in Seoul using various data mining techniques. Commun Stat Appl Methods 25:199–215

Chollet, F., 2015. Keras [WWW Document]. URL https://keras.io Accessed 4 April 21

Choubin B, Abdolshahnejad M, Moradi E, Querol X, Mosavi A, Shamshirband S, Ghamisi P (2020) Spatial hazard assessment of the PM10 using machine learning models in Barcelona, Spain. Sci Total Environ 701:134474

Chung H, Shin K (2020) Genetic algorithm-optimized multi-channel convolutional neural network for stock market prediction. Neural Comput Appl 32:7897–7914. https://doi.org/10.1007/s00521-019-04236-3

Cooney C, Korik A, Folli R, Coyle D (2020) Evaluation of hyperparameter optimization in machine and deep learning methods for decoding imagined speech EEG. Sensors 20:4629. https://doi.org/10.3390/s20164629

Das R, Middya AI, Roy S (2022) High granular and short term time series forecasting of $$\hbox {PM}_{2.5}$$ air pollutant—a comparative review. Artif Intell Rev 55:1253–1287. https://doi.org/10.1007/s10462-021-09991-1

Debry E, Mallet V (2014) Ensemble forecasting with machine learning algorithms for ozone, nitrogen dioxide and PM10 on the Prev'Air platform. Atmos Environ 91:71–84. https://doi.org/10.1016/j.atmosenv.2014.03.049

Dekking FM, Kraaikamp C, Lopuhaä HP, Meester LE (2005) A modern ıntroduction to probability and statistics: understanding why and how. Springer

Fayjie AR, Hossain S, Oualid D, Lee D-J (2018) Driverless car: Autonomous driving using deep reinforcement learning in urban environment, İn: 2018 15th International conference on ubiquitous robots (UR). IEEE, pp 896–901

Feng R, Zhou R, Shi W, Shi N, Fang X (2021) Exploring the spatial heterogeneity and temporal homogeneity of ambient PM10 in nine core cities of China. Sci Rep 11:1–13

Freeman BS, Taylor G, Gharabaghi B, Thé J (2018) Forecasting air quality time series using deep learning. J Air Waste Manag Assoc 68:866–886. https://doi.org/10.1080/10962247.2018.1459956

Geng G, Zhang Q, Martin RV, van Donkelaar A, Huo H, Che H, Lin J, He K (2015) Estimating long-term PM2.5 concentrations in China using satellite-based aerosol optical depth and a chemical transport model. Remote Sens Environ 166:262–270. https://doi.org/10.1016/j.rse.2015.05.016

Geng G, Zheng Y, Zhang Q, Xue T, Zhao H, Tong D, Zheng B, Li M, Liu F, Hong C, He K, Davis SJ (2021) Drivers of PM2.5 air pollution deaths in China 2002–2017. Nat Geosci 14:645–650. https://doi.org/10.1038/s41561-021-00792-3

Gidhagen L, Krecl P, Targino AC, Polezer G, Godoi RHM, Felix E, Cipoli YA, Charres I, Malucelli F, Wolf A, Alonso M, Segersson D, Castelhano FJ, Amorim JH, Mendonça F (2021) An integrated assessment of the impacts of PM2.5 and black carbon particles on the air quality of a large Brazilian city. Air Qual Atmos Health 14:1455–1473. https://doi.org/10.1007/s11869-021-01033-7

Golovin D, Solnik B, Moitra S, Kochanski G, Karro J, Sculley D (2017) Google vizier: a service for black-box optimization, İn: Proceedings of the 23rd ACM SIGKDD International conference on knowledge discovery and data mining. pp 1487–1495

Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT press

Graves A (2012) Sequence Transduction with Recurrent Neural Networks. https://doi.org/10.48550/ARXIV.1211.3711

Grefenstette JJ (1993) Genetic algorithms and machine learning, İn: Proceedings of the sixth annual conference on computational learning theory. pp 3–4

Greff K, Srivastava RK, Koutník J, Steunebrink BR, Schmidhuber J (2016) LSTM: a search space odyssey. IEEE Trans Neural Netw Learn Syst 28:2222–2232

Guo B, Hu J, Wu W, Peng Q, Wu F (2019) The Tabu_Genetic algorithm: a novel method for hyper-parameter optimization of learning algorithms. Electronics 8:579. https://doi.org/10.3390/electronics8050579

Guo Y, Li J-Y, Zhan Z-H (2020) Efficient hyperparameter optimization for convolution neural networks in deep learning: a distributed particle swarm optimization approach. Cybern Syst 52:36–57

Hancer E, Xue B, Karaboga D, Zhang M (2015) A binary ABC algorithm based on advanced similarity scheme for feature selection. Appl Soft Comput 36:334–348. https://doi.org/10.1016/j.asoc.2015.07.023

Heydari A, Majidi Nezhad M, Astiaso Garcia D, Keynia F, De Santoli L (2022) Air pollution forecasting application based on deep learning model and optimization algorithm. Clean Techn Environ Policy 24:607–621. https://doi.org/10.1007/s10098-021-02080-5

Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9:1735–1780

Holland JH (1992) Genetic algorithms. Sci Am 267:66–73

Hooyberghs J, Mensink C, Dumont G, Fierens F, Brasseur O (2005) A neural network forecast for daily average PM10 concentrations in Belgium. Atmos Environ 39:3279–3289. https://doi.org/10.1016/j.atmosenv.2005.01.050

Hruschka ER, Campello RJ, Freitas AA (2009) A survey of evolutionary algorithms for clustering. IEEE Trans Syst Man Cybern Part C Appl Rev 39:133–155

Huang Y, Cheng Y, Bapna A, Firat O, Chen D, Chen M, Lee H, Ngiam J, Le QV, Wu Y (2019) Gpipe: efficient training of giant neural networks using pipeline parallelism. Adv Neural Inf Process Syst 32:103–112

Huang J, Hu X, Yang F (2011) Support vector machine with genetic algorithm for machinery fault diagnosis of high voltage circuit breaker. Measurement 44:1018–1027. https://doi.org/10.1016/j.measurement.2011.02.017

Huang C-J, Kuo P-H (2018) A deep CNN-LSTM model for particulate matter (PM2.5) forecasting in smart cities. Sensors 18:2220. https://doi.org/10.3390/s18072220

Huberty CJ, Olejnik S (2006) Applied MANOVA and discriminant analysis. Wiley

IDS 2018 [WWW Document], 2022. . Canadian Institute for Cybersecurity. URL https://www.unb.ca/cic/datasets/ids-2018.html. Accessed 26 May 22

Jiang Q, Sun YL, Wang Z, Yin Y (2015) Aerosol composition and sources during the Chinese Spring Festival: fireworks, secondary aerosol, and holiday effects. Atmos Chem Phys 15:6023–6034

De Jong K (1988) Learning with genetic algorithms: an overview. Mach Learn 3:121–138

Kaur S, Aggarwal H, Rani R (2020) Hyper-parameter optimization of deep learning model for prediction of Parkinson's disease. Mach vis Appl 31:32. https://doi.org/10.1007/s00138-020-01078-1

Kim S-K, Oh T-I (2018) Real-time PM10 concentration prediction LSTM model based on IoT streaming sensor data. J Korea Acad Ind Coop Soc 19:310–318

Kim H, Lee J-H (2016) A recurrent neural networks approach for estimating the quality of machine translation output, İn: Proceedings of the 2016 conference of the north american chapter of the association for computational linguistics: human language technologies. pp 494–498

Klein G, Dabney A (2013) The cartoon introduction to statistics. Hill and Wang, New York

Kothandapani V, Kuppuswamy S (2020) Towards activation function search for long short-term model network: a differential evolution based approach. J King Saud Univ Comput Inf Sci. https://doi.org/10.1016/j.jksuci.2020.04.015

Kristiani E, Kuo T-Y, Yang C-T, Pai K-C, Huang C-Y, Nguyen KLP (2021) PM2.5 forecasting model using a combination of deep learning and statistical feature selection. IEEE Access 9:68573–68582. https://doi.org/10.1109/ACCESS.2021.3077574

Kunang YN, Nurmaini S, Stiawan D, Suprapto BY (2021) Attack classification of an intrusion detection system using deep learning and hyperparameter optimization. J Inf Secur Appl 58:102804. https://doi.org/10.1016/j.jisa.2021.102804

Kurnaz G, Demir AS (2022) Prediction of SO2 and PM10 air pollutants using a deep learning-based recurrent neural network: case of industrial city Sakarya. Urban Clim 41:101051. https://doi.org/10.1016/j.uclim.2021.101051

Lerman PM (1980) Fitting segmented regression models by grid search. J Roy Stat Soc: Ser C (appl Stat) 29:77–84

Li T, Hua M, Wu X (2020) A hybrid CNN-LSTM model for forecasting particulate matter (PM2.5). IEEE Access 8:26933–26940. https://doi.org/10.1109/ACCESS.2020.2971348

Li T, Shen H, Yuan Q, Zhang X, Zhang L (2017) Estimating ground-level PM2.5 by fusing satellite and station observations: a geo-intelligent deep learning approach. Geophys Res Lett 44:11–985

Liaw R, Liang E, Nishihara R, Moritz P, Gonzalez JE, Stoica I (2018) Tune: a research platform for distributed model selection and training. https://arxiv.org/abs/1807.05118

Ma J, Cheng JCP, Lin C, Tan Y, Zhang J (2019) Improving air quality prediction accuracy at larger temporal resolutions using deep learning and transfer learning techniques. Atmos Environ 214:116885. https://doi.org/10.1016/j.atmosenv.2019.116885

Martínez NM, Montes LM, Mura I, Franco JF (2018) Machine learning techniques for PM 10 levels forecast in Bogotá, İn: 2018 ICAI workshops (ICAIW). IEEE, pp 1–7

Menares C (2021) Forecasting PM2.5 levels in Santiago de Chile using deep learning neural networks. Urban Clim. https://doi.org/10.1016/j.uclim.2021.100906

Mendoza H, Klein A, Feurer M, Springenberg JT, Urban M, Burkart M, Dippel M, Lindauer M, Hutter F (2019) Towards automatically-tuned deep neural networks. Automated machine learning. Springer, Cham, pp 135–149

Meng X, Fu Q, Ma Z, Chen L, Zou B, Zhang Y, Xue W, Wang J, Wang D, Kan H, Liu Y (2016) Estimating ground-level PM10 in a Chinese city by combining satellite data, meteorological information and a land use regression model. Environ Poll. https://doi.org/10.1016/j.envpol.2015.09.042

Močkus J (1975) On Bayesian methods for seeking the extremum. In: Marchuk GI (ed) Optimization techniques IFIP technical conference. Springer, pp 400–404

Nurcahyanto H, Prihatno AT, Alam MM, Rahman MH, Jahan I, Shahjalal M, Jang YM (2022) Multilevel RNN-based PM10 air quality prediction for industrial internet of things applications in cleanroom environment. Wirel Commun Mobile Comput. https://doi.org/10.1155/2022/1874237

Olof SS (2018) A comparative study of black-box optimization algorithms for tuning of hyper-parameters in deep neural networks

Otter DW, Medina JR, Kalita JK (2020) A survey of the usages of deep learning for natural language processing. IEEE Trans Neural Netw Learn Syst 32:604–624

Pandas, development team, (2020) pandas-dev/pandas: Pandas. https://doi.org/10.5281/zenodo.3509134

Park H, Cho S, Park J (2018) Word RNN as a baseline for sentence completion, İn: 2018 IEEE 5th ınternational congress on ınformation science and technology (CiSt). IEEE, pp 183–187

Park J-H, Yoo S-J, Kim K-J, Gu Y-H, Lee K-H, Son U-H (2017) PM10 density forecast model using long short term memory, in: 2017 Ninth ınternational conference on ubiquitous and future networks (ICUFN). IEEE, pp 576–581

Pascanu R, Mikolov T, Bengio Y (2013) On the difficulty of training recurrent neural networks, İn: International conference on machine learning. PMLR, pp 1310–1318

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: machine learning in python. J Mach Learn Res 12:2825–2830

Popko EA, Weinstein IA (2016) Fuzzy logic module of convolutional neural network for handwritten digits recognition. J Phys Conf Series. https://doi.org/10.1088/1742-6596/738/1/012123

Possatti LC, Guidolini R, Cardoso VB, Berriel RF, Paixão TM, Badue C, De Souza AF, Oliveira-Santos T (2019) Traffic light recognition using deep learning and prior maps for autonomous cars, İn: 2019 International joint conference on neural networks (IJCNN). IEEE, pp 1–8

Pricope T-V (2021) Deep reinforcement learning in quantitative algorithmic trading: a review. https://arxiv.org/abs/2106.00123

Raimondi PM, Bue AL, Vitale MC (2005) A CNN adaptive model to estimate PM10 monitoring, İn: 2005 IEEE conference on emerging technologies and factory automation. IEEE, pp 6-pp

Salvador P, Artınano B, Alonso DG, Querol X, Alastuey A (2004) Identification and characterisation of sources of PM10 in Madrid (Spain) by statistical methods. Atmos Environ 38:435–447

Samal KKR (2021) Multi-directional temporal convolutional artificial neural network for PM2.5 forecasting with missing values: a deep learning approach. Urban Clim. https://doi.org/10.1016/j.uclim.2021.100800

Santhosh M, Venkaiah C, Kumar DMV (2019) Short-term wind speed forecasting approach using ensemble empirical mode decomposition and deep Boltzmann Machine. Sustain Energy Grids Netw 19:100242. https://doi.org/10.1016/j.segan.2019.100242

Slini T, Kaprara A, Karatzas K, Moussiopoulos N (2006) PM10 forecasting for Thessaloniki, Greece. Environ Model Softw 21:559–565

Stork J, Eiben AE, Bartz-Beielstein T (2020) A new taxonomy of global optimization algorithms. Nat Comput. https://doi.org/10.1007/s11047-020-09820-4

Strobl C, Boulesteix A-L, Kneib T, Augustin T, Zeileis A (2008) Conditional variable importance for random forests. BMC Bioinformat 9:1–11

Swersky K, Snoek J, Adams RP (2014) Freeze-thaw Bayesian optimization. https://arxiv.org/abs/1406.3896

Tavallaee M, Bagheri E, Lu W, Ghorbani AA (2009) A detailed analysis of the KDD CUP 99 data set, İn: Proceedings of the second IEEE ınternational conference on computational ıntelligence for security and defense applications, CISDA'09. IEEE Press, Ottawa, Ontario, Canada, pp 53–58

Telikani A, Tahmassebi A, Banzhaf W, Gandomi AH (2022) Evolutionary machine learning: a survey. ACM Comput Surv 54:1–35. https://doi.org/10.1145/3467477

Thornton C, Hutter F, Hoos HH, Leyton-Brown K (2013) Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms, İn: Proceedings of the 19th ACM sıgkdd ınternational conference on knowledge discovery and data mining. pp 847–855

Théate T, Ernst D (2021) An application of deep reinforcement learning to algorithmic trading. Expert Syst Appl 173:114632

Tokgöz A, Ünal G (2018) A RNN based time series approach for forecasting Turkish electricity load, İn: 2018 26th signal processing and communications applications conference (SIU). IEEE, pp 1–4

WHO, 2021. Health impacts [WWW Document]. URL https://www.who.int/teams/environment-climate-change-and-health/air-quality-and-health/health-impacts Accessed 28 May 22

Walid A, Alamsyah IU (2017) Recurrent neural network for forecasting time series with long memory pattern. J Phys Conf Ser. https://doi.org/10.1088/1742-6596/824/1/012038

van der Walt S, Colbert SC, Varoquaux G (2011) The NumPy array: a structure for efficient numerical computation. Comput Sci Eng 13:22–30. https://doi.org/10.1109/MCSE.2011.37

Xu X (2020) Forecasting air pollution PM2. 5 in Beijing using weather data and multiple kernel learning. J Forecast 39:117–125

Yadav A, Jha CK, Sharan A (2020) Optimizing LSTM for time series prediction in Indian stock market. Procedia Comput Sci 167:2091–2100

Yan B, Han G (2018) Effective feature extraction via stacked sparse autoencoder to improve intrusion detection system. IEEE Access 6:41238–41248

Yang L, Shami A (2020) On hyperparameter optimization of machine learning algorithms: theory and practice. Neurocomputing 415:295–316. https://doi.org/10.1016/j.neucom.2020.07.061

Yang J, Yan R, Nong M, Liao J, Li F, Sun W (2021) PM2.5 concentrations forecasting in Beijing through deep learning with different inputs, model structures and forecast time. Atmos Pollut Res 12:101168. https://doi.org/10.1016/j.apr.2021.101168

Yaseen MU, Anjum A, Rana O, Antonopoulos N (2019) Deep learning hyper-parameter optimization for video analytics in clouds. IEEE Trans Syst Man Cybern Syst 49:253–264. https://doi.org/10.1109/TSMC.2018.2840341

Yeo I, Choi Y, Lops Y, Sayeed A (2021) Efficient PM2.5 forecasting using geographical correlation based on integrated deep learning algorithms. Neural Comput Appl 33:15073–15089. https://doi.org/10.1007/s00521-021-06082-8

Young T, Hazarika D, Poria S, Cambria E (2018) Recent trends in deep learning based natural language processing. İEEE Comput İntell Mag 13:55–75

Zaini N, Ean LW, Ahmed AN, Malek MA (2022) A systematic literature review of deep learning neural network for time series air quality forecasting. Environ Sci Pollut Res 29:4958–4990. https://doi.org/10.1007/s11356-021-17442-1

Zela, A., Klein, A., Falkner, S., Hutter, F., 2018. Towards automated deep learning: Efficient joint neural architecture and hyperparameter search. https://arxiv.org/abs/1807.06906

Zhang Y, Bocquet M, Mallet V, Seigneur C, Baklanov A (2012) Real-time air quality forecasting, part I: history, techniques, and current status. Atmos Environ 60:632–655. https://doi.org/10.1016/j.atmosenv.2012.06.031

Zhang Z, Zeng Y, Yan K (2021) A hybrid deep learning technology for PM2.5 air quality forecasting. Environ Sci Pollut Res 28:39409–39422. https://doi.org/10.1007/s11356-021-12657-8

Zheng Y, Liu F, Hsieh H-P (2013) U-Air: when urban air quality inference meets big data. In: KDD. https://doi.org/10.1145/2487575.2488188

Zickus M, Greig AJ, Niranjan M (2002) Comparison of four machine learning methods for predicting PM10 concentrations in Helsinki, Finland. Water Air Soil Pollut Focus 2:717–729

Zoph B, Vaswani A, May J, Knight K (2016) Simple, fast noise-contrastive estimation for large rnn vocabularies, İn: Proceedings of the 2016 conference of the north american chapter of the association for computational linguistics: human language technologies. pp 1217–1222

Şahin ÜA, Ucan ON, Bayat C, Tolluoglu O (2011) A New approach to prediction of SO2 and PM10 concentrations in Istanbul, Turkey: cellular neural network (CNN). Environ Forensics 12:253–269. https://doi.org/10.1080/15275922.2011.595047