Journal of
Information Systems Engineering
and Business Intelligence

Vol.9, No.2, October 2023
Available online at: http://e-journal.unair.ac.id/index.php/JISEBI

Enhancing Multi-Output Time Series Forecasting with Encoder-Decoder Networks

Kristoko Dwi Hartomo [1], **Joanito Agili Lopo** [2]*, **Hindriyanto Dwi Purnomo** [3]

[1)2)3)] *Faculty of Information Technology, Universitas Kristen Satya Wacana, Salatiga, Indonesia*

[1)] kristoko@uksw.edu, [2)] amalopo99@gmail.com [3)]hindriyanto.purnomo@uksw.edu

Abstract

Background: Multi-output Time series forecasting is a complex problem that requires handling interdependencies and interactions between variables. Traditional statistical approaches and machine learning techniques often struggle to predict such scenarios accurately. Advanced techniques and model reconstruction are necessary to improve forecasting accuracy in complex scenarios.
Objective: This study proposed an Encoder-Decoder network to address multi-output time series forecasting challenges by simultaneously predicting each output. This objective is to investigate the capabilities of the Encoder-Decoder architecture in handling multi-output time series forecasting tasks.
Methods: This proposed model utilizes a 1-Dimensional Convolution Neural Network with Bidirectional Long Short-Term Memory, specifically in the encoder part. The encoder extracts time series features, incorporating a residual connection to produce a context representation used by the decoder. The decoder employs multiple unidirectional LSTM modules and Linear transformation layers to generate the outputs each time step. Each module is responsible for specific output and shares information and context along the outputs and steps.
Results: The result demonstrates that the proposed model achieves lower error rates, as measured by MSE, RMSE, and MAE loss metrics, for all outputs and forecasting horizons. Notably, the 6-hour horizon achieves the highest accuracy across all outputs. Furthermore, the proposed model exhibits robustness in single-output forecast and transfer learning, showing adaptability to different tasks and datasets.
Conclusion: The experiment findings highlight the successful multi-output forecasting capabilities of the proposed model in time series data, with consistently low error rates (MSE, RMSE, MAE). Surprisingly, the model also performs well in single-output forecasts, demonstrating its versatility. Therefore, the proposed model effectively various time series forecasting tasks, showing promise for practical applications.

Keywords: Bidirectional Long Short-Term Memory, Convolutional Neural Network, Encoder-Decoder Networks, Multi-output forecasting, Multi-step forecasting, Time-series forecasting

Article history: Received 13 July 2023, first decision 14 September 2023, accepted 5 October 2023, available online 28 October 2023

I. INTRODUCTION

Time series regression poses challenges with increasingly complex decision-making, particularly in real-world scenarios. These challenges could be the lack of available data, the issue of missing data, and the required considering the relationships between variables [1], [2]. The multi-dimensional data and long-term forecasting also add to the problem's complexity. However, the statistical approaches and the classical machine learning techniques are struggling to meet the growing demands of modern complex problems. They lacked flexibility in handling varying numbers of series and time steps, managing missing values, and inability to capture non-linear temporal dependencies. Therefore, it requires advanced techniques and careful reconstruction of the model to ensure accurate prediction and the challenges in complex problems.

Deep Learning (DL) has revolutionized the field of pattern recognition, surpassing human-level accuracy in automatically recognizing patterns in spatial and temporal data [3]. DL has proven its capability to tackle complex problems that were beyond the reach of traditional machine-learning algorithms. This breakthrough captivated the interest of practitioners facing vast amounts of data across various domains. The success in computer vision, natural language processing, and speech recognition have demonstrated the versatility and potentiality to address various domain problems, including the time series analysis task.

* Corresponding author

Deep Learning (DL), a novel learning-based method, has recently been applied to time series analysis, with applications ranging from solar power forecasting, weather forecasting, electricity load forecasting, electricity price forecasting, and financial prediction [2]. This method utilizes fundamental or technical analysis to extract relevant textual information and analyze historical data, making it suitable for time series forecasting tasks. It offers the ability to capture complex patterns, support multivariate inputs, and adapt to varying levels of complexity. For example, Wang et al. [4] used a temporal convolutional network (TCN) consisting of a one-dimensional Convolutional Neural Network (CNN) for short-term load forecasting in the electric power market, Bandara et al.[5] employed time series clustering techniques combined with Recurrent Neural Networks (RNNs) for subgroup modeling, and Zerveas et al. utilized Transformer-based frameworks for unsupervised representation learning of multivariate time series [6]. These examples demonstrated the effectiveness of DL in tackling diverse time series analysis tasks. Furthermore, it opens opportunities and further explorations of another widely-used architecture in DL, such as Encoder-Decoder architecture. Encoder-Decoder architecture was initially designed for language translation tasks [7]. However, Encoder-Decoder networks have succeeded in various sequence-related tasks such as dense estimation, anomaly detection, gas distribution mapping, and disease classification [8]–[11].

The ability of the Encoder-Decoder enables its effective utilization in the field of time series analysis. It generally consists of an Encoder that encodes the input to a fixed-length vector representation and then uses a Decoder to decode the target output sequence from the fixed vector. It could be advantageous when facing long-term forecasting time series, which could learn the long temporal dependencies and solve the non-linear relationship between variables. Many researchers have applied Encoder-Decoder architecture in various domains and cases in the context of time series forecasting. Du et al.[12] used a temporal attention Encoder-Decoder model in five multivariate time series datasets and outperformed baseline methods, demonstrating their superior forecasting performance. Laubscher [13] developed an Encoder-Decoder Gated Recurrent Unit (GRU) to predict future reheater metal temperatures in coal power stations. Zhou et al.[14] utilized the Transformers algorithm to predict long sequence time series, particularly in electricity consumption planning. Jin et al. [15] proposed an Attention-based Encoder-Decoder network with Bayesian optimization to overcome the limitations of existing electrical load forecasting methods when dealing with time-series data. Lyu et al. [16] used the Encoder-Decoder model to present a multi-step prediction for gas concentration in coal mines.

As an increasingly complex problem in real-world cases, the need for multi-output has shown promising performance rather than single-output forecasting. Prior research in this domain has shown certain shortcomings that necessitate further exploration and research. Firstly, there exists a pressing need to address the limitations of past studies in handling real-world complexities, where multiple variables and factors can affect forecasted outcomes. This limitation compromises the practical applicability and accuracy of forecasting models [17], [18]. Secondly, there is a need for further investigation into integration and evaluation methods, specifically within the context of multi-output forecasting tasks. Predicting multiple variables allows leveraging potential correlation and dependencies among the variables, and the model can incorporate additional contextual information, improving forecasts' reliability and accuracy [19], [20]. In addition, there is a lack of exploration and validation of methods in the real-world scenario, where the forecasting task is important to overcome. Hence, this study addresses this problem by integrating Encoder-Decoder networks for multi-output time series forecasting. We propose a model that utilizes the Convolution Neural Network (CNN) with Bidirectional Long Short-Term Memory (LSTM) in the Encoder, while incorporating multiple LSTM modules and Linear transformation layers in the Decoder to generate target or output variables at each time step. Through the integration of these components, we aim to explore the capabilities of the Encoder-Decoder architecture in addressing multi-output time series forecasting and contribute and improve its practical applicability in the time series field.

II. Literature Review

Many researchers have proposed combining classic machine learning and modern deep learning techniques in the time series forecasting task. These approaches tackle the challenges of predicting multiple outputs in time series data.

Within the realm of classic machine learning, Chou et al. [21] utilized the least squares support vector regression (LSSVR) model with a multi-output scheme and accelerated particle swarm optimization (APSO) algorithm for financial time series forecasting. They used the exchange rates dataset's multivariate interval-valued time series (ITS) to forecast financial time trends. Talavera-Llames et al. [22] introduced a novel algorithm based on the general nearest neighbours procedure for time forecasting, specifically designed to handle multivariate data and make multi-output predictions. They evaluated their model using data from the Spanish electricity market, showcasing remarkable improvements in terms of accuracy and execution time. Jiang et al. [23] utilized backpropagation, bidirectional long short-term memory, and gated recurrent unit models for forecasting electricity price and load data from the Australian electricity market. They incorporated a strategy based on a multi-objective salp swarm algorithm and utilized a rolling forecast mechanism to handle multivariable and multi-input multi-output structures. Zhan et al. [24] proposed a new multivariate GBRT method, which considers the correlation among multi-outputs. They applied this approach to predict the longer trend of traffic speed using data extracted from three loop detectors in the US101-N freeway.

On the other hand, several studies focused on leveraging modern deep learning architectures for multi-output time series forecasting. Sadeque and Bui [25] introduced a deep-learning architecture for weather forecasting, specifically targeting wind speed, relative humidity, dew point, and temperature. They utilized a stacked Long Short-Term Memory (LSTM) layer, cascading the basic 1-hour-ahead model, which predicts the weather parameters for the 2 and 3 hours ahead. Azizi et al. [26] employed various models, including Convolutional Neural Networks (CNN) and CNN-LSTM, to simultaneously predict solar radiation and temperature. They conducted simulations with different input parameters and outputs to predict solar irradiation. Qu et al. [27] proposed a multi-output and spatiotemporal model that combined Graph Convolution Neural Networks (GCN) and Transformers for temperature forecasting of grain in storage. The GCN captured the sensor's spatial correlation and topological information, while Transformer captured long-term and short-term temporal features and dependencies. They used a real-granary dataset from Shaanxi, China, and evaluated it using Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). Zhang et al. [28] proposed a novel hybrid deep learning model called B-CM-C3D combining 3-D Convolutional Neural Network, Convolutional Minimum Gate Memory Neural Network, and Variational Bayesian Neural Network. This combination aimed to achieve multi-step ahead probabilistic forecasting of multiple hydrological variables for multiple stations, demonstrating improved prediction accuracy and reduced training time compared to another model. Lu et al. [29] developed a spatiotemporal wind power forecasting model based on a multi-output support vector machine and grey wolf optimizer strategy, known as the ST-GWO-MSVM forecasting model. They utilized wind power data from 15 wind farms and showed that the proposed model outperformed others.

Furthermore, additional studies addressed the limitations of existing methods and proposed innovative approaches. He et al. [30] address two limitations in multivariate time series forecasting methods based on attention-based encoder-decoder models. They proposed a novel prediction framework called Dynamic Co-Attention Networks (DCAN) to forecast traffic volume, air quality, and domestic house datasets. The model utilizes a two-stage variables embedding network to capture the potential semantics that integrates target and non-predictive variables. Wang et al. [31] introduced an attention-based recurrent neural network (RNN) model for multi-step prediction of target parameters using historical multivariate sensory time series. Lloret et al. [32] proposed two deep learning models: dilated causal convolutional (DCCN) and encoder-decoder recurrent neural network (EDRNN), for forecasting disaggregated freight flows. Shi and Wang [33] proposed a Convolutional-LSTM encoder-decoder (ConvLSTM-AE) hybrid model for multivariate output and multi-step prediction with short time intervals to predict agrometeorological variables such as air temperature, relative humidity, and wind speed.

While there have been significant advancements and research in multi-output time series forecasting, which incorporates classic machine learning and modern deep learning models, a need for further research and enhancement model integration and experimental cases in real-world conditions is essential. The encoder and decoder model has been used to solve the time series forecasting task in various domains, however, there is a gap in the literature above regarding the systematic integration and evaluation of encoder-decoder networks in this context. Furthermore, there is a lack of exploration and validation of methods in the real-world scenario. Constructing a method that can work in pre-modeling and post-modeling is crucial in time series forecasting. For instance, utilizing a decoder to produce multiple time steps or output variables simultaneously by using encoded context representation can be more effective than only training the model for specific time steps at different times. Using teacher-forcing and recursive forecasting can effectively maintain the training process and implementation in the real-world environment. In addition, the transfer learning methods to enhance pre-trained models or context representation in specific forecasting tasks will likely be more efficient and improve predictive accuracy rather than training the model from scratch. Considering these gaps, we covered all of the integration and experimental cases in this research to leverage the potential correlations and dependencies among variables, leading to a more comprehensive analysis and potentially improved forecast accuracy. Addressing this gap can contribute to advancing multi-output time series forecasting methods and enhance their applicability in various domains.

III. METHODS

The proposed model consists of an encoder-decoder module, and each module has some additional layers that become the core of the modules. The Encoder module has two fundamental layers: CNN and Bi-LSTM. At the top of the encoder part is a concatenation layer that receives forward-backwards hidden and cell state from Bi-LSTM layers. It merges them using a linear transformation and the tanh activation function, producing a vector representation of the feature. Meanwhile, the decoder module contains sharing Multi-LSTM and Linear Transformation layers that associate with numbers of the target variable. In other words, different LSTM and Linear Layers will handle each target feature, which shares their hidden and cell states or unique patterns to the different target feature layers. The overview of the proposed approach can be seen in Fig. 1.

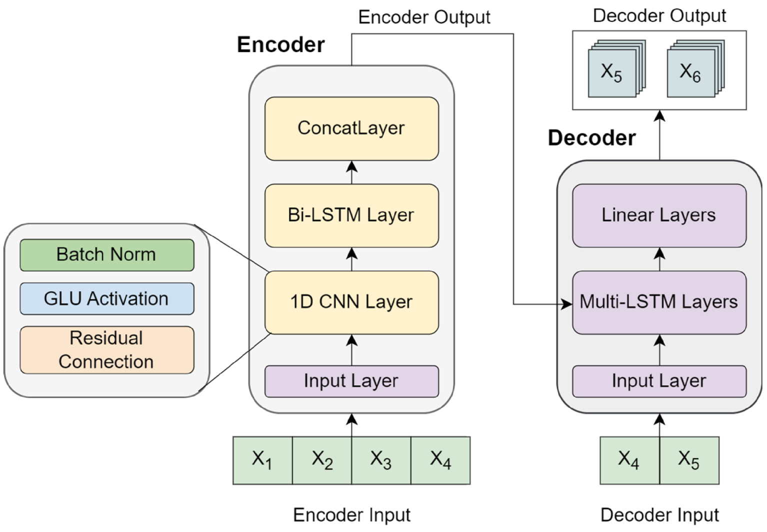


Fig. 1 Overview of the proposed approach

A. Encoder Module

The proposed encoder module consists of two layers: the 1-dimensional Convolutional Neural Network (1D CNN) followed by the Bidirectional Long Short-Term Memory (Bi-LSTM). The CNN layer will extract the implicit features in the data, detect spatial substructure and create meaningful spatial substructure as a result. The network can learn to identify significant features, trends, or patterns in the data that contribute to the forecasting task. Meanwhile, the CNN followed by LSTM has proven to be an effective model combination in the time series forecasting task[34], and it is a common approach to leveraging spatial and temporal information in the context vector representation. The CNN layers capture local patterns and extract meaningful features, meanwhile, the Bi-LSTM incorporates the temporal context and captures long-term dependencies by accessing past and future patterns. It enables a more comprehensive understanding of temporal dynamics and learning non-linear correlation features[12].

1) 1D Convolutional Neural Network (CNN)

The first layer in our proposed Encoder Module is 1-dimensional CNN. This dimension of the convolution operation has been used in the time series and natural language processing tasks. It allows for learning and capturing spatiotemporal patterns within feature vectors, enhancing the model's ability to comprehend and process information at each time step in the sequence dimension. The model can gain a deeper understanding of the temporal dynamics and relationships within the data, enabling them to generate meaningful representations. Using CNNs, the model employed parameter sharing to reduce the number of parameters and allows the network to learn shared patterns across different time series. The illustration of the convolution blocks can be seen in Fig. 2.

The convolution operation is performed by sliding the filters over the input sequence, followed by an element-wise activation function σ. Let X be the multivariate time series input of length T with N variables, K filters of size F, and a bias term b, the output (feature maps) are denoted as H with dimension $T' \times K$. The formula of the 1D CNN is defined in (1).

$$H_{t,k} = \sigma\left(\sum_{f=1}^{F}\sum_{n=1}^{N} w_{f,n,k} \cdot X_{t+f-1,n} + b_k\right) \quad (1)$$

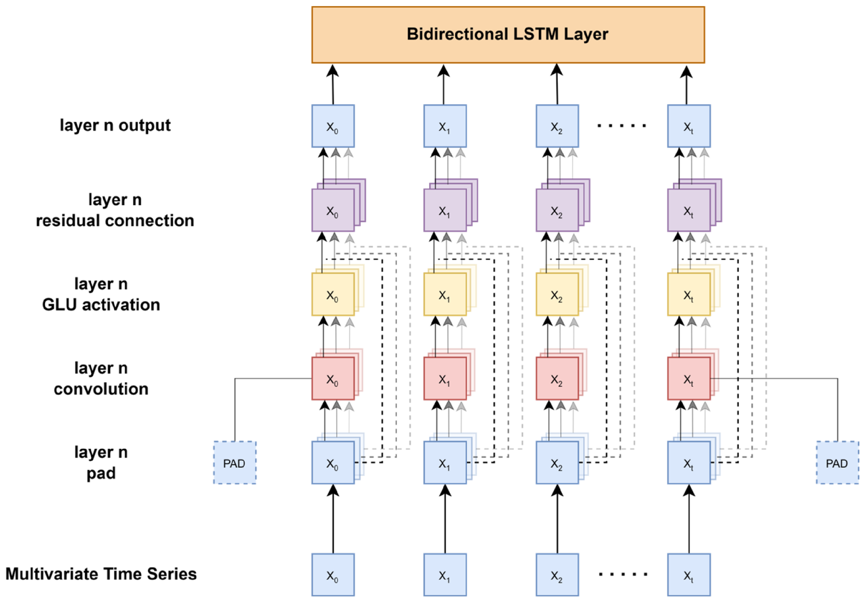


Fig. 2 Convolutional blocks layer

Furthermore, the activation function σ used in this study is GLU (Gated Linear Units). The GLUs activation function is widely used in tasks where the input length should keep the same after forwarding the convolution layer. This activation captures complex and non-linear relationships, ensures smooth and continuous activation, and provides improved gradient flow and convergence properties compared to the ReLU activation function. GLUs have gating mechanisms (similar to LSTMs and GRUs) contained within the sigmoid activation function, which usually keep the hidden dimension the same size. The activation function is defined in (2).

$$GLU(x) = x \cdot \left(\frac{1}{1+e^{-x}}\right) \quad (2)$$

The CNNs architecture is commonly divided into two types of layers: convolutional and pooling layers. However, the main drawback of traditional pool operations, such as max pooling or average pooling, is the potential loss of sequential information [35]. In the context of time series forecasting, whereas each multivariate input is essential to forecast, applying pooling layers could discard some detailed information needed for making accurate predictions. Commonly, the length of the input coming out of a convolutional layer will be $filter_{size} - 1$ shorter than the input entering the convolution layer. Therefore, to handling this problem, we set padding in the input before apply convolution. It will reduce the length of input and the input coming into the convolutional blocks to equal the length of coming out.

2) Bidirectional Long Short-Term Memory (Bi-LSTM)

The second layer of our proposed Encoder Module is Bidirectional Long Short-Term Memory (Bi-LSTM), as shown in Fig. 3. This layer comprises the forward and backward sub-layers that can access past and future information when predicting a specific time step. Considering the context from both directions, the model can capture complete dependencies and valuable information missed by a unidirectional, which tends to reduce the efficiency of LSTM, particularly in time series data [36]. An LSTM unit typically consists of three gates: input, forget, and output which are useful for forgetting and storing the information in the memory cells (cell states). Suppose x_t and i_t represent the time series input and an input gate, respectively, at the time step t, U and W are weight matrices corresponding to the input and hidden states, and h_{t-1} is the hidden state from the previous time step. Therefore, to calculate the input i_t, forget f_t, and output o_t gates as seen in (3)-(5).

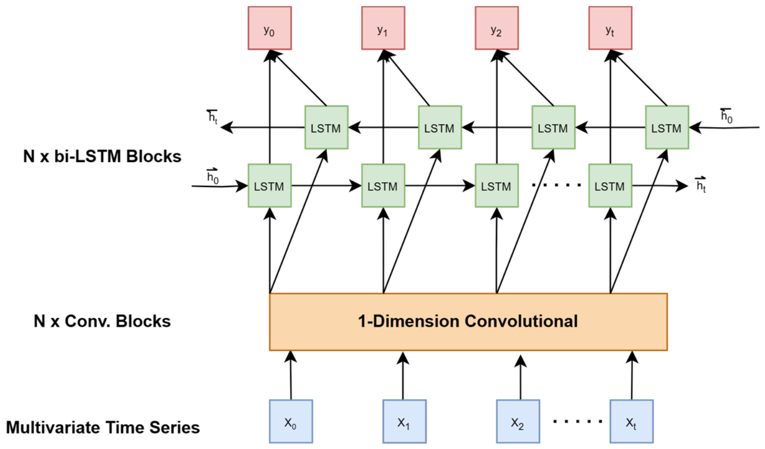


Fig. 3 The Bidirectional Long Short-Term Memory blocks

$$i_t = \sigma\left(b_i + \sum_j U_{ij}x_{tj} + \sum_j W_{ij}h_{t-1j}\right) \quad (3)$$

$$f_t = \sigma\left(b_f + \sum_j U_{fij}x_{tj} + \sum_j W_{fij}h_{t-1j}\right) \quad (4)$$

$$o_t = \sigma\left(b_o + \sum_j U_{oij}x_{tj} + \sum_j W_{oij}h_{t-1j}\right) \quad (5)$$

After the calculation of those gates, we then calculate the cell c_t and hidden h_t states at the time step t as defined in (6) and (7), respectively. The output of each cell aims to get the relevant and meaningful information, discard irrelevant or redundant information, and as the representation of the LSTM output each time step.

$$c_t = f_t \cdot c_{t-1} + i_t \cdot tanh\left(b_c + \sum_j U_{cij}x_{tj} + \sum_j W_{cij}h_{t-1j}\right) \quad (6)$$

$$h_t = o_t \cdot tanh(c_t) \quad (7)$$

Since we used forward and backward layers, we must compute each hidden state from both directions into a single representation or create a unified representation. Let say h_t represents the combined hidden state at time step t, h_t^f and h_t^b are the backward and forward hidden states, then we can obtain the combined hidden state through concatenation as seen in (8).

$$h_t = concat([h_t^f, h_t^b]) \quad (8)$$

In addition, we used the hyperbolic tangent activation function to ensure that the combined hidden state captures a complex relationship. This activation function makes the hidden representations should bound within a specific range, especially between -1 and 1. It is the common choice for activation functions due to its smooth and continuous nature. The updated equations for obtaining a combined hidden state with a hyperbolic tangent function are seen in (9).

$$h_t = tanh\left(concat\left(\left[h_t^f, h_t^b\right]\right)\right) \quad (9)$$

The combined representation from the last layer in the Encoder Module, which consists of hidden and cell bi-LSTM representation, will propagate through the Decoder module to generate the outputs of each time step. Those memory cells will produce a fixed-size context representing a semantic summary of the multivariate input sequence. By using the context, the decoder will clearly understand the data patterns before learning to produce the output.

B. Decoder Module

The decoder module, as shown in Fig. 4, consists of multiple LSTM and Linear transformation layers. It allows the model to capture the temporal dependencies and interactions between different output variables at each time step and transform them into the desired outputs for multiple time series.

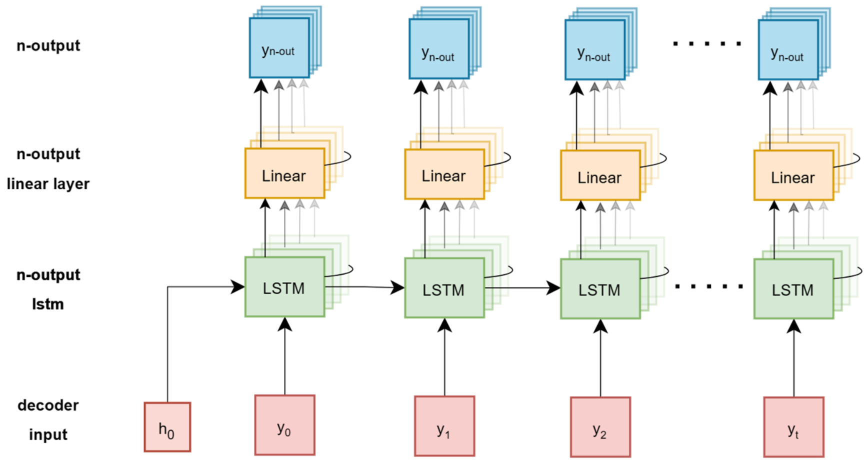


Fig. 4 Proposed decoder module

The number of LSTM modules will depend on the number of output dimensions. Each LSTM module will be responsible for specific outputs or targets. By using multiple LSTM modules concerning the number of output dimensions, each dedicated to capturing temporal dependencies and patterns within specific output dimensions. As a result, the decoder can effectively learn the complex dynamics of the multi-dimensional time series data and leverage sequential information to make accurate predictions. Let denotes $LSTM_i$, $hidden_i$, $cell_i$, is LSTM module, hidden, and cell for each i output, then the computation be represented in (10).

$$LSTM_i\left(input, (hidden_{i-1}, cell_{i-1})\right) \rightarrow output_i, (hidden_i, cell_i) \quad (10)$$

Furthermore, we shared the hidden state across multiple LSTM modules. Based on the formula above, the hidden and cell for the current output dimension is the hidden and cell from the previous LSTM module in specific output. It ensures that all output dimension have access to the same encoded historical information while having specific context representations in different output dimensions. The shared representation enables leveraging the learned information from one output dimension, capturing the interdependencies and sharing common patterns in each output.

Meanwhile, the subsequent multiple linear layers provide the flexibility to transform the LSTM outputs into the desired output dimension for forecasting. These linear transformations allow the decoder to map the learned representations from the LSTM module to the specific format and scale required for multi-output time series forecasting. Furthermore, it can effectively adapt the learned representations from the LSTM module to match specific requirements and characteristics of each output dimension. Doing this enhances the forecasting performance for each output dimension.

Each context vector in each time step will propagate through the feed-forward layer to produce the time series outputs. These processes ensure that cross-variable interactions occur when the model predicts each output, capturing different characteristics, complexity levels, or varying degrees of uncertainty in the multi-outputs.

C. Training Strategies & Evaluation Metrics

There are two standard procedures in training sequential data. The recursive and teacher-forcing strategies. The recursive works by using the output in the previous step $t-1$ as input at time $t+1$. In our study, especially in real-world scenarios, where the model will not have access to the ground-truth output y_t, the model will use their prediction in the previous step as the input to predict the future step. It is common in the forecasting task. While the teacher-forcing strategy is a strategy that receives the ground-truth output y_t as input at time $t+1$ to produce the future output. This procedure would apply during the training and enhance the training process. Furthermore, it could teach intricate patterns more efficiently and effectively when forecasting in the future step.

Accordingly, we use teacher forcing in the training and recursive in the forecasting. The illustration of both training types is shown in Fig. 5. In detail, we set the teacher-forcing ratio during training. Instead of using only the ground truth output y_t as input in the next step, we combine both procedures by a ratio. The ratio will be conditioned with a random number, deciding whether teacher-forcing or recursive will be used in each step. Therefore, in Fig. 4, we can see that specific steps will use teacher forcing and others will not.

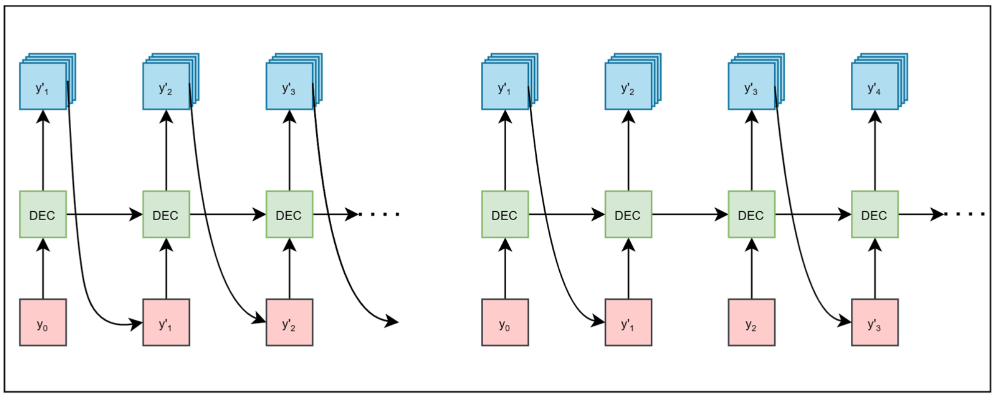


Fig. 5 The training strategies. **Left**: Recursive training type receives the output prediction of each step as input to the next step. **Right:** Mixed teacher-forcing type mixed recursive and teacher-forcing types determined by a ratio

Lastly, we evaluated our model with standard metrics that aim to measure the model's performance. We used Mean Squared Error (MSE) and Root Mean Squared Error (RMSE). The MSE measures the average squared difference between the actual values y_i and the predicted values $\hat{y}_i$. Meanwhile, the RMSE takes the square root of the mean squared error to obtain the mean root value. These metrics' formulations are defined in (11) and (12).

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \quad (11)$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \quad (12)$$

We used Mean Absolute Error (MAE) metrics to evaluate and measure the model's performance for comparison purposes. Mean Absolute Error (MAE) measures the average absolute difference between the predicted and actual values without considering the direction of the errors. It measures the average magnitude of errors in the same unit as the original data and is beneficial for understating the scale of the errors. The formula of MAE is expressed in (13), where, y_i and $\hat{y}_i$ represent the actual and predicted values of the data i, and n is the total number of data.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \quad (13)$$

IV. RESULTS

This section provides the experiment results on a public multivariate time series dataset from Beijing Multi-Site Air-Quality Data [37], which contains multi-output or multi-target data. We also demonstrated the superiority of the proposed model in handling multi-output time series forecasting by performing transfer learning in different datasets.

A. Dataset

Beijing Multi-Site Air Quality Data is a public dataset donated in 2019 by Beijing Municipal Environmental Monitoring Center. The dataset contains multivariate air pollutants and relevant meteorological variables related to internal factors affecting air quality. Many researchers and projects have already used the dataset in time-series forecasting tasks [19], [38]–[41]. It is a 1-hour interval with 11 features for 12 weather stations in China administrations. The data was gathered from the 1st of March 2013 to the 28th of February 2017, with 420,768 instances or 35,064 instances of each of the 12 weather stations. However, this study only considers two weather stations: Gucheng stations for modelling and Wanshouxigong stations for transfer learning purposes.

The Gucheng station data consists of hourly data on six primary air pollutants and six relevant meteorological features. These six air pollutant concentrations are Particulate Matter (PM) with diameters of 2.5 and 10 micrometers, Sulfur Dioxide (SO2), Nitrogen Dioxide (NO2), Carbon Monoxide (CO), and Ozone (O3). Meanwhile, the Temperature (TEMP), Pressure (PRES), Dew Point Temperature (DEWP), Precipitation (RAIN), Wind Direction (wd), and Wind Speed (WSPM) are compared as the meteorological variables. The statistical summary of each feature is shown in Table 1. Each feature has a certain number of missing values. Even though these missing values are less than 2%, we must address them before modelling the data. Therefore, we performed an imputation by applying linear interpolation to fill in the remaining missing values with the median value.

TABLE 1
STATISTICAL SUMMARY OF THE GUCHENG DATASET

Feature*	Unit	Missing Value	Mean	Standard Deviation	Median
PM2.5	ug/m^3	646	83.31	82.46	59.0
PM10	ug/m^3	381	114.48	96.08	93.0
SO2	ug/m^3	507	16.36	22.046	8.0
NO2	ug/m^3	668	57.57	36.83	52.0
CO	ug/m^3	1401	1293.63	1215.55	900.0
O3	ug/m^3	729	57.03	57.46	43.0
TEMP	degree celcius	51	13.72	11.34	14.7
PRES	hPa	50	1010.33	10.36	1010.0
DEWP	degree celcius	51	2.86	13.73	3.4
RAIN	mm	43	0.0659	0.87	0.0
WSPM	m/s	42	1.52	1.19	1.2

*Excluded the unuseful features (e.g., year, month, day, index, stations, and wd.)

We examine the previous study on forecasting time series multi-output in air quality data to choose the targets [19], [42], [43]. Based on our observation, we consider the four outputs commonly used in forecasting tasks: PM2.5, PM10, SO2, and NO2. Fig. 6 shows how complex the output will be forecast. These outputs have become standard for targeting pollutants with significant health implications, environment, and regulatory relevance. Besides that, we also evaluate the correlation of each feature to decide the target. We can conclude that there is a relationship in each output. The relationship could have a strong positive correlation, such as PM2.5 and PM10, and a moderate positive correlation, such as PM2.5, with SO2 and NO2. It indicates essential insights regarding forecasting the multi-output time series.

It is commonly used to normalize time series data to ensure that all feature values fall within a standardized range, regardless of the original data with varying range values. Therefore, we preprocessed the feature to have the same scale or a standardized range. We used a data normalization technique called MinMax Scaler, which scales the values within a specific range, typically between 0 and 1.

We also use the Wanshouxigong station dataset to examine our proposed model by performing transfer learning. The term "transfer learning" here means that we used the previous model, which has been trained with a specific dataset, to forecast the dataset that the model had never seen before in a specific dataset. It allows models on related tasks or domains to improve the learning process and the generalization ability of new models on specific targets[44]. Furthermore, the Gucheng station will be a model builder, and the Wanshouxigong station will examine the model performance on new data and have specific characteristics and intricate patterns. By doing that, we can measure the model generalization in another task within the same domain.

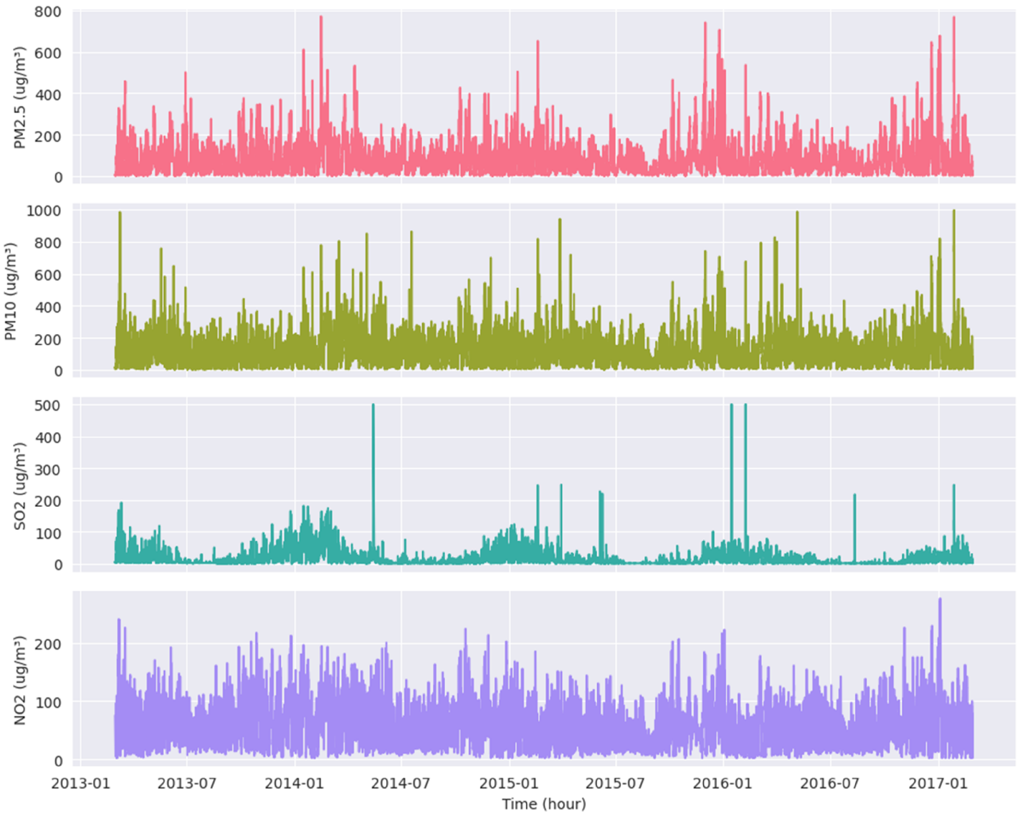


Fig. 6 Visual representation of each output in the Gucheng station dataset

B. Experimental Setup

Three essential steps are commonly used to train the machine learning model: Splitting Data, Model Training, and Model Evaluation.

Splitting the data is an important step commonly used to train the machine learning model. In the machine learning model context, dividing the data into three portions is better: training, testing, and validation. However, in this study, we only split our data into train and test data with a portion of 2% of the test set. The reason is that we will examine the model with another dataset. It might be a good approach for this case.

The exact number of each set will depend on the sequence (e.g., steps, hour, timestep, or horizon) we determined. We use the word sequence here to describe the time series in the data, which refers to the ordered arrangement of data points over time. Each data point in the sequence represents a measurement or observation taken at a specific time or interval. It is crucial because it captures the temporal dependency and potential patterns or trends in data. It can be seen as a history of information that the model should learn. Specifically, in the context of our air quality data, it points out how many hours we considered in one feature. Therefore, we experimented with sequences or hours: 1-hour, 6-hour, 18-hour, and 24-hour.

In the model training and evaluation, we used different strategies for sequencing data. We combined different training and evaluation strategies to build the model. Each strategy is described in the method section above. We used teacher-forcing in the training mode and mixed teacher-forcing in the testing mode. It allows leveraging the benefits of teacher-forcing for training while introducing a more realistic validation setting and enabling the model to forecast based on its predictions during forecasting. After that, to examine the model's performance, we set the recursive strategies to demonstrate the model in real-world data, which does not have the ground truth or target.

To determine the most optimal hyperparameter configuration, we experiment with such a trial to determine the best value. A tuned combination of hyperparameters will be applied to the model with the value details in Table 2.

Primarily, our hyperparameter is based on such components. First is the model architecture, which determines the structure and depth of the model. Regularization and dropout hyperparameters showed us the regularization control to prevent model overfitting. The CNN and scale hyperparameters are specific to the convolutional layer and the contribution of the residual path in the model. Furthermore, training configuration hyperparameters define the setting for training the model, such as batch_size, learning_rate, and optimizer choice.

TABLE 2
HYPERPARAMETER VALUES

Hyperparameter	Value	Description
n_layers	4	Number of LSTM layers in both encoder and decoder layer
hidden_size	64	Number of LSTM hidden sizes in both encoder and decoder layer
dropout	0.2	Number of dropouts in Linear layers
cnn_layers	2	Number of CNN layer
kernel_size	3	Number of kernel sizes in CNN layer
batch_size	32	Number of batch sizes in the data loader
learning_rate	0.001	Number of learning rates of the optimizer
optimizer	AdamW	Variants of Adam with weight decay
scale	0.5	Number of contributions of the residual path

C. Result Analysis

We experimented with two scenarios to assess the model performance in handling time series forecasting tasks. The first is to show the model performance in the single-output and the second is the multi-output experiment, which is the primary experiment of this study. We used these scenarios to evaluate the proposed model performance in single and multi-output to demonstrate that the model works well in any conditions, even though not in their first aims.

1) Single-Output

On the single-output side, we only considered 24-hours of sequence, which means the model will have 24 hours of history information and try to forecast the 24-hours ahead. Computational efficiency and the paramount importance of real-world weather forecasting applications are why we used 24-hour sequence in a single-output experiment. It is more important to assess the model performance in the long-term horizon, specifically only including one feature, rather than in short-step forecasting. Meanwhile, PM2.5 is our feature due to the crucial environmental parameter directly impacting air quality and public health. In addition, PM2.5 has been used by many researchers as an essential pollution level to forecast air quality [39], [40], [45], [46]. The result showed that the model achieved a relatively low training loss with the 0.0014 value. Meanwhile, the test cost 0.0012 indicates relatively low testing loss and better performance of how well the model generalizes to unseen data. In addition, we also calculated RMSE and MAE scores and got 0.0305 and 0.0197, respectively. These cost metrics suggest the model makes accurate predictions with minor errors and deviations from the actual values. It indicates that the model is performing well in minimizing prediction errors and capturing the intricate patterns in the data.

However, the objective evaluation of the model performance should be in real-world scenarios where the model faces unseen data or does not have ground truth in the time series data. Therefore, we evaluated or tested the data with unseen data and the model's performance in single-output. Note that our evaluation is based on a recursive technique. This approach aligns with forecasting in practice, where predictions are made sequentially based on available historical information. It ensures that the model prediction relies solely on historical data. As we can see in Figure 7, the model is quite successful in forecasting and fitting the target data. Even though there is still a specific point that the model could not forecast, it could be an outlier or the extreme values present in the data that make it challenging to forecast. Overall, the model can identify the intricate patterns of the data.

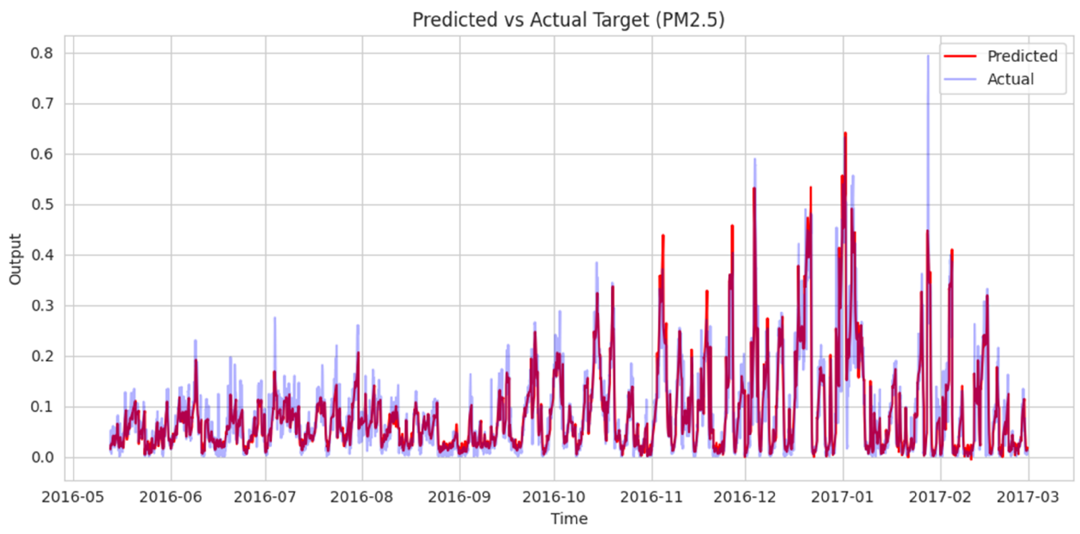


Fig. 7 Comparison of predicted and actual in the PM2.5 single-output model

2) *Multi-Output*

In this main experiment, we used four horizons or sequences, as mentioned before, which are 1-hours, 6-hours, 12-hours, and 24-hours. The metric score for each horizon showed in Table 3. Using these different horizons not only looks at which horizons performed better, but we can inspect the model performance on the long sequence and the model performance when the information provided is short. We choose 1-hours to examine the lack of information and 24-hours if the model tends to have information loss.

Based on Table 2, the 6-hours of horizon generally perform better in accuracy and precision along each output. The loss values in MSE, RMSE, and MAE are relatively lower compared to the longer horizon (12-hours and 24-hours). The value also has no significant variations when the model predicts different air pollutants, with relatively small errors compared with the actual values. It demonstrated the capability to capture short-term patterns effectively.

TABLE 3
LOSSES FOR EACH HORIZON

Horizon	PM2.5			PM10			SO2			NO2		
	MSE	RMSE	MAE	MSE	RMSE	MAE	MSE	RMSE	MAE	MSE	RMSE	MAE
1-hours	0.0046	0.0390	0.0319	0.0038	0.0436	0.0223	0.0007	0.0023	0.0189	0.0055	0.0552	0.0422
6-hours	0.0010	0.0293	0.0198	0.0013	0.0347	0.0240	0.0007	0.0236	0.0172	0.0024	0.0468	0.0320
12-hours	0.0016	0.0378	0.0245	0.0021	0.0442	0.0307	0.0010	0.0290	0.0208	0.0042	0.0635	0.0429
24-hours	0.0018	0.0400	0.0270	0.0022	0.0456	0.0324	0.0008	0.0278	0.0204	0.0044	0.0651	0.0445

The loss over the training time and epoch is shown in Figure 8. The figure provides insight into the model performance and its ability to converge toward optimal parameters along the epochs. The smoothness of the loss curve indicates how well the model performs and effectively learns from the training data and gradually improves its performance. The model also appears to generalize well to unseen data while capturing the important patterns in the training data. The model produced 815,548 trainable and non-trainable parameters in training. It should increase the complexity and capacity to learn from the data and be more prone to overfitting. However, it showed that the model was suitable for handling overfitting or underfitting. It makes the model capable of capturing subtle and nuanced patterns and relationships in the input data.

In addition, we did not set up the value for the epoch, but we used an early stopping technique. Early stopping prevents overfitting by stopping the training process before the model becomes too specialized to the training data and underperforms in the unseen data. Therefore, we only initiate a number as early stopping criteria, using 20 as a threshold for performance degradation.

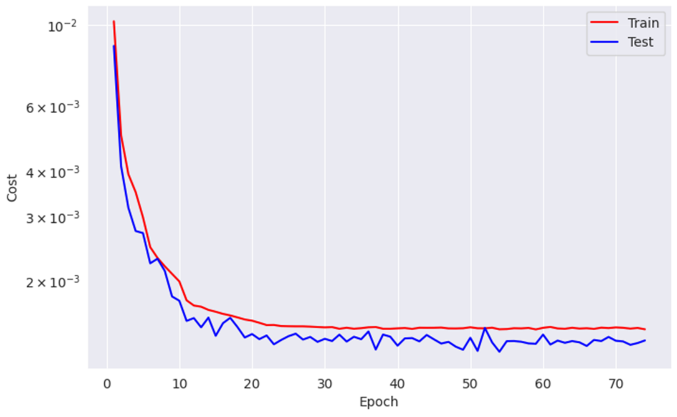


Fig. 8 The losses in training and testing for each epoch

Furthermore, the complete visualization of each output has presented in Figure 9. As we can see, there is a specific point that the model could not forecast along the outputs. We have inspected them as extreme values or outliers in the data. In this study, we purposely chose to keep the extreme values or outliers in the data instead of removing them. By keeping these challenging data points in the analysis, we aim to evaluate the model's robustness and ability in real-world data, even in the presence of outliers. As a result, it seems like the model can learn the pattern in the presence of an outlier. The model predictions align with the general trend of the data, capturing the fluctuation and overall behavior. However, the model still struggles to accurately predict the peak value in the extreme values.

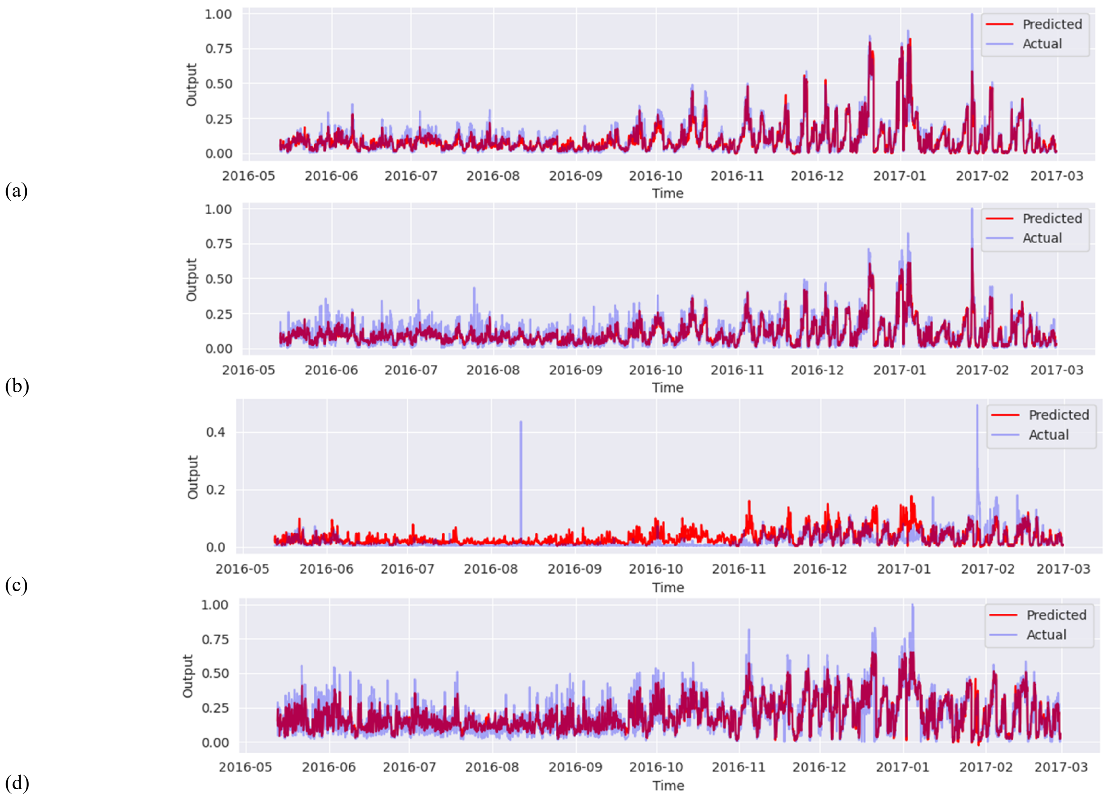
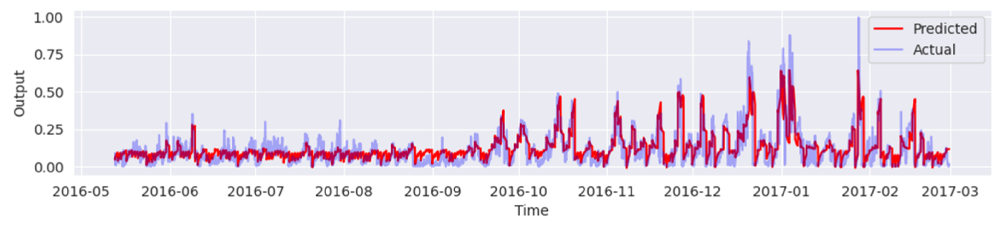
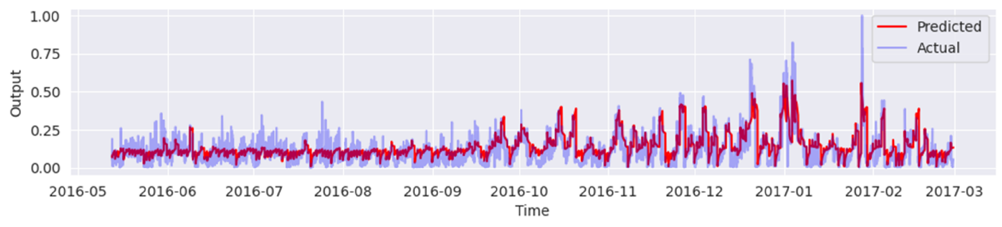


Fig. 9 Visual representation of predicted and target each output in 6-hour horizon (a) PM2.5, (b) PM10, (c) SO2, and (d) NO2

Nevertheless, the loss value tends to increase by extending the horizon or the sequence, even though there are no significant differences between the 6-hours horizon. It will indicate that the model performance struggles as it attempts to forecast over longer time spans. Figure 10 visually compares the predicted and actual values in the 24-hour horizon, especially in the target PM2.5 and PM10. Based on the visualization, we can infer that the model can still capture the underlying patterns and dynamics of the data, albeit with reduced accuracy. Although its performance may not be as strong for the 6-hours horizon, the model can still capture specific trends and make predictions that are informative to some extent. As long as this study purpose is to examine the model in multi-output capability, the overall performance in each output expresses the superiority of the model. It provides meaningful forecasts with acceptable accuracy for multiple air pollutants over extended periods, signifying its multi-output forecasting capability.



(a)



(b)

Fig. 10 Predicted and target in 24-hour horizon (a) PM2.5 and (b) PM10

Meanwhile, if we only use 1-hours horizon, the information is insufficient to learn from the model. As a result, the loss increased three times compared with 6-hours horizon. It suggests accurately forecasting air pollutant levels, as such a short horizon might be challenging for the model. The data's limited time horizon, dynamics, and short-term fluctuations restrict the model from learning and incorporating basic patterns and trends beyond the immediate next hour.

3) *Transfer Learning*

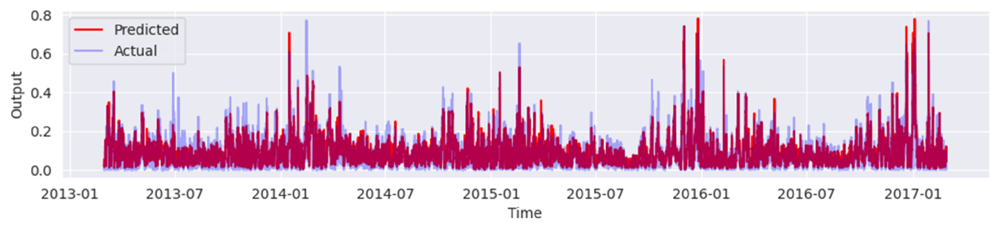
To assess the robustness of the proposed model, we experimented with transfer the learning model Gucheng 12-hour horizon into the 12-hours horizon in the Wanshouxigong dataset. Table 4 shows the losses for the 12-hours horizon in the Wanshouxigong dataset using the transfer learning model. Overall, the model demonstrated exemplary performance in forecasting PM2.5, PM10, and SO2 and may struggle to accurately predict NO2 concentration as indicated by the higher losses.

TABLE 4
LOSSES FOR 12-HOURS HORIZON IN WANSHOUXIGONG

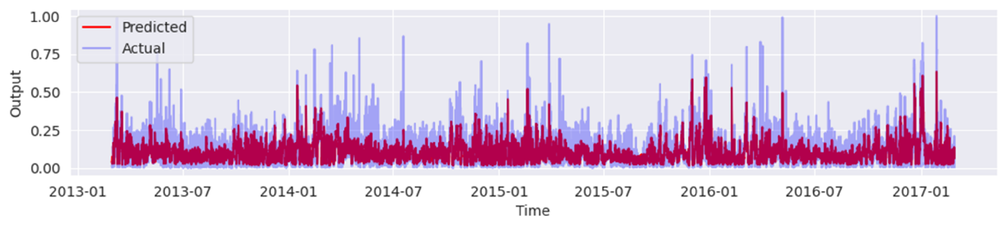
Metric	Output			
	PM2.5	PM10	SO2	NO2
MSE	0.0054	0.0055	0.0008	0.0130
RMSE	0.0737	0.0744	0.0285	0.1141
MAE	0.0548	0.0548	0.0211	0.0892

The model achieves relatively low losses for the PM2.5 and PM10 outputs with MSE, RMSE, and MAE, indicating a slight average difference between the predicted and actual values. In the case of SO2, the model performs well with an MSE of 0.0008, indicating low overall error prediction SO concentrations. The RMSE and MAE values also confirm the model accuracy in capturing the SO2 patterns. However, for the NO2 output, the model exhibits a higher loss compared to the other loss. The MSE had a relatively higher error in predicting NO2 concentrations, the same as RMSE, and MAE also has a more significant difference between the predicted and actual values for NO2.

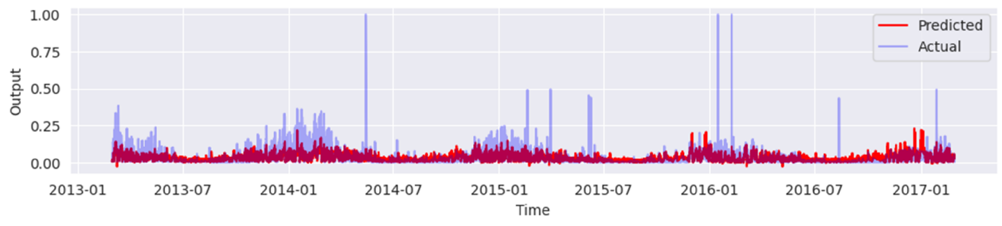
Figure 11 visually represents the model predicted and the actual target values for the Wanshouxigong dataset. The model generally fits the trend and captures the main features of the actual data. However, it is worth noting that there are discrepancies between the predicted values and the actual data, particularly in the case of NO2. Despite these errors, the model still captures the general pattern and approximates the actual values, albeit with deviations. It indicates that the model understands the underlying patterns in the data and can adapt and learn from specific datasets, even if it cannot perfectly fit the actual values.



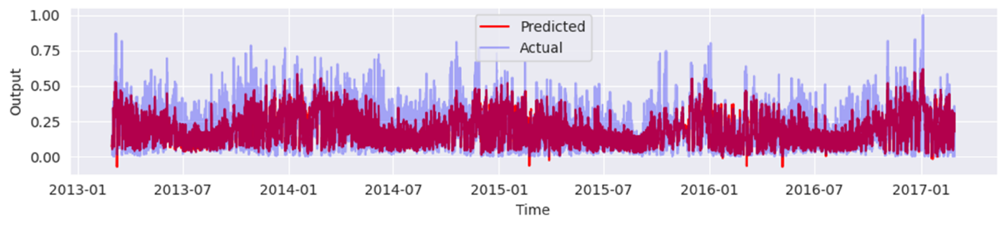
(a)



(b)



(c)



(d)

Fig. 11 Visual representation of predicted and target each output in transfer learning 6-hour horizon (a) PM2.5, (b) PM10, (c) SO2, and (d) NO2

V. Discussion

This study examined the Encoder-Decoder networks by proposing a model architecture combining LSTM, CNN, and Linear layers, especially in multi-output time series forecasting. The experiments have been conducted using two open public datasets, Gucheng and Wanshouxigong stations datasets from Beijing Multi-Site Air Quality Data. The first experiment assesses the proposed model in single-output tasks; the second is the main experiment, which evaluates the multi-output in Gucheng and then transfers the learning to evaluate the Wanshouxigong stations dataset.

In the single-output analysis, the model has trained using 24-hours of historical data to forecast the 24-hours of PM2.5 feature. The model achieved a relatively low error in training and testing data based on the result. Overall, each metric score confirms the high accuracy and demonstrates its ability to identify intricate patterns in the time series data, even though it is designed for multi-output forecasting. We compared our proposed model, as shown in Table 5, to show how the model competes with other models. Our proposed model performed better than the prior study[39], which builds the model solely on a single output. It is three times lower than the VMD+BiLSTM model in the MAE score and reached 3.050% in the RMSE score. In addition, our proposed model also outperformed the previous study [45], with the MSE error reduced by 14.56% in the training and 12.68% in the testing compared with 13.38% and 55.30%, respectively.

TABLE 5
COMPARISON OF GUCHENG STATION PM2.5 MODELS

Algorithms	MAE	RMSE%
Decision Tree	16.155	22.973
BiLSTM	9.716	14.204
EMD+BiLSTM	4.768	6.497
VMD+LSTM	4.453	5.947
VMD+BiLSTM	3.481	5.121
Proposed	1.970	3.050

We experimented with four horizons in the multi-output analysis to examine the proposed model performance on long and short sequences. The result showed that the 6-hour horizon performs better than other horizons. The value also has no significant variations when the model predicts different air pollutants, with relatively small errors compared with the actual values. It demonstrated the capability to capture short-term patterns effectively, making the model less error-sensitive [31]. Shorter-term forecasts rely on more immediate past data, which can be easier to capture accurately. We compared our proposed model with the previous study [19]. However, they consider only two outputs, PM2.5 and PM10, making it unable to comprehend all output comparisons. Our proposed model outperforms overall scores, especially in the RMSE and MAE metrics, reaching under 40 compared with 40 and 32 in PM2.5 and 54 and 47 in PM10. Meanwhile, the model also tends to have relatively low values across each loss metric in transfer learning. It indicates that the model performs well in forecasting the Wanshouxigong stations dataset and makes it accurate with minor errors and deviations from the actual values. It also measures how well the proposed model transfers the learning into another specific dataset.

This study faced some limitations in the design of the multi-output model. The horizon length effect has become the main problem of this. As we increase the sequence, the model tends to increase the loss, which makes the model difficult to forecast. It suggests that longer-term forecasts are less accurate compared to shorter-term forecasts. Therefore, it is crucial to consider when using the model for longer-term prediction. However, based on our evaluation, the model can still capture the data patterns, even though it does not fit perfectly. Future work can explore and develop a specialized technique to improve long-term prediction, particularly in multi-output scenarios. A possible integration is to use Informer model which based on transformer architecture or attention layer in both encoder and decoder layers. It has proved to handle long sequence text in the single-output time series forecasting model. Another limitation is the variability across outputs. Some outputs, such as PM10 and NO2, exhibit higher prediction errors than others, such as SO2 and PM2.5. It indicates that the model might have more difficulty accurately forecasting specific outputs and may require further improvement specifically for those outputs. Future research can concentrate on output-specific model enhancement to improve the forecasting accuracy of specific outputs using feature engineering, model adjustment, or specialized training approaches. For example, parallel multi-head attention mechanism in transformer architecture has shown great performance to handle feature or output-specific using their multi self-attention mechanism.

VI. CONCLUSIONS

In this study, we introduced an encoder-decoder model capable of handling multi-output time series forecasting tasks. The model could generate multiple outputs simultaneously for each step in the sequences. Beijing Multi-Site Air Quality Data has been used to evaluate the model performance. The experiment showed that the model could effectively forecast the multiple outputs in different horizons. It can be seen in the relatively low error for MSE, RMSE, and MAE metric loss. The model also shows robustness in the forecast of the single-output, even though the model is built for multi-output purposes. Therefore, it can be concluded that the proposed model can effectively handle the time series forecasting tasks.

Author Contributions: *Kristoko Dwi Hartomo*: Conceptualization, Methodology, Writing - Original Draft. *Joanito Agili Lopo*: Code Experiment, Methodology, Data Curation, Writing - Original Draft. *Hindriyanto Dwi Purnomo*: Writing - Review & Editing, Supervision.

All authors have read and agreed to the published version of the manuscript.

Funding: This research was financially supported by the Internal Research of Satya Wacana Christian University.

Acknowledgments: The authors sincerely thank Radius Tanone for reviewing and providing valuable feedback on this paper.

Conflicts of Interest: The authors declare no conflict of interest.

Data Availability: The source code and project materials for this research are openly accessible in our GitHub repository at https://github.com/joanitolopo/MultiTSF-Encoder-Decoder. Researchers and interested parties are encouraged to use these resources for replication, validation, and further analysis. The data used in this study is sourced from the UCI Irvine Machine Learning Repository's dataset titled "Beijing Multi-Site Air Quality Data," which is available through the following DOI: https://doi.org/10.24432/C5RK5G. While the source code is openly shared, it is important to note that the data is publicly accessible from its original repository.

Informed Consent: There were no human subjects.

Animal Subjects: There were no animal subjects.

ORCID:
Kristoko Dwi Hartomo: https://orcid.org/0000-0003-0237-851X
Joanito Agili Lopo: https://orcid.org/0009-0001-3183-7132
Hindriyanto Dwi Purnomo: https://orcid.org/0000-0001-6728-7868

REFERENCES

[1] Q.Q. He, P.C.I. Pang, and Y.W.Si, “Transfer Learning for Financial Time Series Forecasting,” in *PRICAI 2019: Trends in Artificial Intelligence,* pp. 24–36, 2019.

[2] A. Mahmoud and A. Mohammed, “A Survey on Deep Learning for Time-Series Forecasting,” in *Machine Learning and Big Data Analytics Paradigms: Analysis, Applications and Challenges*, A. E. Hassanien and A. Darwish, Eds. Cham: Springer International Publishing, pp. 365-392, 2021, doi: 10.1007/978-3-030-59338-4_19.

[3] S. Sengupta *et al.*, “A review of deep learning with special emphasis on architectures, applications and recent trends,” *Knowledge-Based Syst.*, vol. 194, 105596, 2020, doi: 10.1016/j.knosys.2020.105596.

[4] H. Wang, Y. Zhao, and S. Tan, “Short-Term load forecasting of power system based on time convolutional network,” *2019 8th Int. Symp. Next Gener. Electron. ISNE 2019*, pp. 1-3, 2019, doi: 10.1109/ISNE.2019.8896684.

[5] K. Bandara, C. Bergmeir, and S. Smyl, “Forecasting across time series databases using recurrent neural networks on groups of similar series: A clustering approach,” *Expert Syst. Appl.*, vol. 140, 112896, 2020, doi: 10.1016/j.eswa.2019.112896.

[6] G. Zerveas, S. Jayaraman, D. Patel, A. Bhamidipaty, and C. Eickhoff, “A Transformer-based Framework for Multivariate Time Series Representation Learning,” *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, pp. 2114–2124, 2021, doi: 10.1145/3447548.3467401.

[7] S. Ranathunga, E.S. A. Lee, M. Prifti Skenduli, R. Shekhar, M. Alam, and R. Kaur, “Neural Machine translation for low-resource languages: A survey,” *ACM Comput. Surv.*, vol. 55, no. 11, pp. 1-37, 2023, doi: 10.1145/3567592.

[8] Y. Wang *et al.*, “Lednet: A lightweight encoder-decoder network for real-time semantic segmentation,” in *2019 IEEE International Conference on Image Processing (ICIP)*, pp. 1860–1864, 2019, doi: 10.1109/ICIP.2019.8803154.

[9] S. Wu and Y. Wang, “Attention-based Encoder-Decoder Recurrent Neural Networks for HTTP Payload Anomaly Detection,” in *2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*, pp. 1452–1459, 2021, doi: 10.1109/ISPA-BDCloud-SocialCom-SustainCom52081.2021.00196.

[10] N. P. Winkler, H. Matsukura, P. P. Neumann, E. Schaffernicht, H. Ishida, and A. J. Lilienthal, “Super-Resolution for Gas Distribution Mapping: Convolutional Encoder-Decoder Network,” in *2022 IEEE International Symposium on Olfaction and Electronic Nose (ISOEN)*, pp. 1-3, 2022, doi: 10.1109/ISOEN54820.2022.9789555.

[11] Y. Sarker, M.N. Islam Mondal, S.R. Fahim, S. Shahriar, S.K. Sarker, and S.K. Das, “A Novel Diagnosis System Using Regularized Encoder-Decoder Based Generative Probabilistic Network for Parkinson’s Disease,” in *IEEE Region 10 Symposium (TENSYMP)*, pp. 1444-1447, 2020, doi: 10.1109/TENSYMP50017.2020.9230792.

[12] S. Du, T. Li, Y. Yang, and S.J. Horng, “Multivariate time series forecasting via attention-based encoder–decoder framework,” *Neurocomputing*, vol. 388, pp. 269–279, 2020, doi: 10.1016/j.neucom.2019.12.118.

[13] R. Laubscher, “Time-series forecasting of coal-fired power plant reheater metal temperatures using encoder-decoder recurrent neural networks,” *Energy*, vol. 189, 116187, 2019, doi: 10.1016/j.energy.2019.116187.

[14] H. Zhou *et al.*, "Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting," *35th AAAI Conf. Artif. Intell. AAAI 2021*, vol. 12B, pp. 11106-11115, 2021, doi: 10.1609/aaai.v35i12.17325.

[15] X.B. Jin, W.Z. Zheng, J.L. Kong, X.Y. Wang, Y.T. Bai, T.L. Su, and S. Lin, , "Deep-learning forecasting method for electric power load via attention-based encoder-decoder with bayesian optimization," *Energies*, vol. 14, no. 6, 1596, 2021, doi: 10.3390/en14061596.

[16] P. Lyu, N. Chen, S. Mao, and M. Li, "LSTM based encoder-decoder for short-term predictions of gas concentration using multi-sensor fusion," *Process Saf. Environ. Prot.*, vol. 137, pp. 93-105, 2020, doi: 10.1016/j.psep.2020.02.021.

[17] D. Xu, Y. Shi, I. W. Tsang, Y.S. Ong, C. Gong, and X. Shen, "Survey on Multi-Output Learning," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 31, no. 7, pp. 2409-2429, 2020, doi: 10.1109/TNNLS.2019.2945133.

[18] K.K.R. Samal, K.S. Babu, and S.K. Das, "Time Series Forecasting of Air Pollution using Deep Neural Network with Multi-output Learning," in *IEEE 18th India Council International Conference (INDICON)*, pp. 1-5, 2021, doi: 10.1109/INDICON52576.2021.9691669.

[19] K.K.R. Samal, A.K. Panda, K.S. Babu, and S.K. Das, "Multi-output TCN autoencoder for long-term pollution forecasting for multiple sites," *Urban Clim.*, vol. 39, 100943, 2021, doi: https://doi.org/10.1016/j.uclim.2021.100943.

[20] 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. Clean. Prod.*, vol. 209, pp. 134-145, 2019, doi: https://doi.org/10.1016/j.jclepro.2018.10.243.

[21] J.S. Chou, D.N. Truong, and T.L. Le, "Interval Forecasting of Financial Time Series by Accelerated Particle Swarm-Optimized Multi-Output Machine Learning System," *IEEE Access*, vol. 8, no. 2008, pp. 14798-14808, 2020, doi: 10.1109/ACCESS.2020.2965598.

[22] R. Talavera-Llames, R. Pérez-Chacón, A. Troncoso, and F. Martínez-Álvarez, "MV-kWNN: A novel multivariate and multi-output weighted nearest neighbours algorithm for big data time series forecasting," *Neurocomputing*, vol. 353, pp. 56–73, 2019, doi: 10.1016/j.neucom.2018.07.092.

[23] P. Jiang, Y. Nie, J. Wang, and X. Huang, "Multivariable short-term electricity price forecasting using artificial intelligence and multi-input multi-output scheme," *Energy Econ.*, vol. 117, 106471, 2023, doi: https://doi.org/10.1016/j.eneco.2022.106471.

[24] X. Zhan, S. Zhang, W.Y. Szeto, and X. Chen, "Multi-step-ahead traffic speed forecasting using multi-output gradient boosting regression tree," *J. Intell. Transp. Syst. Technol. Planning, Oper.*, vol. 24, no. 2, pp. 125-141, 2020, doi: 10.1080/15472450.2019.1582950.

[25] Z. Al Sadeque and F.M. Bui, "A deep learning approach to predict weather data using cascaded LSTM network," *Can. Conf. Electr. Comput. Eng.*, pp. 1-5, 2020, doi: 10.1109/CCECE47787.2020.9255716.

[26] N. Azizi, M. Yaghoubirad, M. Farajollahi, and A. Ahmadi, "Deep learning based long-term global solar irradiance and temperature forecasting using time series with multi-step multivariate output," *Renew. Energy*, vol. 206, pp. 135-147, 2023, doi: https://doi.org/10.1016/j.renene.2023.01.102.

[27] Z. Qu *et al.*, "Temperature forecasting of grain in storage: A multi-output and spatiotemporal approach based on deep learning," *Comput. Electron. Agric.*, vol. 208, 107785, 2023, doi: https://doi.org/10.1016/j.compag.2023.107785.

[28] Z. Zhang, H. Tang, H. Qin, B. Luo, C. Zhou, and H. Zhou, "Multi-step ahead probabilistic forecasting of multiple hydrological variables for multiple stations," *J. Hydrol.*, vol. 617, 129094, 2023, doi: https://doi.org/10.1016/j.jhydrol.2023.129094.

[29] P. Lu *et al.*, "A novel spatio-temporal wind power forecasting framework based on multi-output support vector machine and optimization strategy," *J. Clean. Prod.*, vol. 254, 119993, 2020, doi: 10.1016/j.jclepro.2020.119993.

[30] X. He, S. Shi, X. Geng, and L. Xu, "Dynamic Co-Attention Networks for multi-horizon forecasting in multivariate time series," *Futur. Gener. Comput. Syst.*, vol. 135, pp. 72-84, 2022, doi: https://doi.org/10.1016/j.future.2022.04.029.

[31] Y. Wang, T. Li, W. Lu, and Q. Cao, "Attention-inspired RNN Encoder-Decoder for Sensory Time Series Forecasting," *Procedia Comput. Sci.*, vol. 209, pp. 112-121, 2022, doi: 10.1016/j.procs.2022.10.104.

[32] I. Lloret, J. A. Troyano, F. Enríquez, and J. J. González-de-la-Rosa, "Two deep learning approaches to forecasting disaggregated freight flows: convolutional and encoder–decoder recurrent," *Soft Comput.*, vol. 25, no. 12, pp. 7769-7784, 2021, doi: 10.1007/s00500-021-05678-5.

[33] G. Shi and C. Wang, "Multivariate Multi-Step Agrometeorological Forecast Model for Rapid Spray," *IEEE Access*, vol. 9, pp. 159271-159282, 2021, doi: 10.1109/ACCESS.2021.3131649.

[34] H. Yin, Z. Ou, S. Huang, and A. Meng, "A cascaded deep learning wind power prediction approach based on a two-layer of mode decomposition," *Energy*, vol. 189, 116316, 2019, doi: 10.1016/j.energy.2019.116316.

[35] R. Wan, S. Mei, J. Wang, M. Liu, and F. Yang, "Multivariate temporal convolutional network: A deep neural networks approach for multivariate time series forecasting," *Electron.*, vol. 8, no. 8, 876, 2019, doi: 10.3390/electronics8080876.

[36] S. Khullar and N. Singh, "Water quality assessment of a river using deep learning Bi-LSTM methodology: forecasting and validation," *Environ. Sci. Pollut. Res.*, vol. 29, no. 9, pp. 12875-12889, 2022, doi: 10.1007/s11356-021-13875-w.

[37] S. Chen, "Beijing Multi-Site Air-Quality Data." 2019.

[38] R. Yan, J. Liao, J. Yang, W. Sun, M. Nong, and F. Li, "Multi-hour and multi-site air quality index forecasting in Beijing using CNN, LSTM, CNN-LSTM, and spatiotemporal clustering," *Expert Syst. Appl.*, vol. 169, 114513, 2021, doi: https://doi.org/10.1016/j.eswa.2020.114513.

[39] Z. Zhang, Y. Zeng, and K. Yan, "A hybrid deep learning technology for PM2.5 air quality forecasting," *Environ. Sci. Pollut. Res.*, vol. 28, no. 29, pp. 39409-39422, 2021, doi: 10.1007/s11356-021-12657-8.

[40] M. Niu, Y. Zhang, and Z. Ren, "Deep Learning-Based PM2.5 Long Time-Series Prediction by Fusing Multisource Data-A Case Study of Beijing," *Atmosphere (Basel).*, vol. 14, no. 2, 340, 2023, doi: 10.3390/atmos14020340.

[41] X. Cheng, W. Zhang, A. Wenzel, and J. Chen, "Stacked ResNet-LSTM and CORAL model for multi-site air quality prediction," *Neural Comput. Appl.*, vol. 34, no. 16, pp. 13849-13866, 2022, doi: 10.1007/s00521-022-07175-8.

[42] R. Rakholia, Q. Le, B. Quoc Ho, K. Vu, and R. Simon Carbajo, "Multi-output machine learning model for regional air pollution forecasting in Ho Chi Minh City, Vietnam," *Environ. Int.*, vol. 173, 107848, 2023, doi: 10.1016/j.envint.2023.107848.

[43] M. Benhaddi and J. Ouarzazi, "Multivariate Time Series Forecasting with Dilated Residual Convolutional Neural Networks for Urban Air Quality Prediction," *Arab. J. Sci. Eng.*, vol. 46, no. 4, pp. 3423-3442, 2021, doi: 10.1007/s13369-020-05109-x.

[44] E. J. Prasetyo and K. D. Hartomo, "Multi-industry stock forecasting using GRU-LSTM deep transfer learning method," vol. 15, no. 2, pp. 30-43, 2023.

[45] C. Erden, "Genetic algorithm-based hyperparameter optimization of deep learning models for PM2.5 time-series prediction," *Int. J. Environ. Sci. Technol.*, vol. 20, no. 3, pp. 2959-2982, 2023, doi: 10.1007/s13762-023-04763-6.

[46] K. Liu *et al.*, "Time series prediction of the chemical components of PM2.5 based on a deep learning model," *Chemosphere*, vol. 342, 140153, 2023, doi: https://doi.org/10.1016/j.chemosphere.2023.140153.