

## RESEARCH ARTICLE

# A Convolutional Transformer Model for Multivariate Time Series Prediction

**DONG-KEON KIM AND KWANGSU KIM, (Member, IEEE)**

Department of Software, Sungkyunkwan University, Suwon, Gyeonggi-do 16419, South Korea

Corresponding author: Kwangsu Kim (kim.kwangsu@skku.edu)

This work was supported by the Ministry of Science and ICT (MSIT) under the Grand Information Technology Research Center Support Program Supervised by the Institute for Information & Communications Technology Planning & Evaluation (IITP), South Korea, under Grant IITP-2021-2015-0-00742.

**ABSTRACT** This paper presents a multivariate time series prediction framework based on a transformer model consisting of convolutional neural networks (CNNs). The proposed model has a structure that extracts temporal features of input data through CNN and interprets correlations between variables through an attention mechanism. This framework solves the problem of the inability to simultaneously analyze the temporal features of the input data and the correlation between variables, which is a limitation of the forecasting models presented in existing studies. We designed a forecasting experiment using several time series datasets with various data characteristics to precisely evaluate the proposed model. In addition, comparative experiments were performed between the proposed model and several predictive models proposed in recent studies. Furthermore, we conducted ablation studies on the extent to which the proposed CNN structure in the prediction model affects the forecasting results by substituting a specific layer of the model. The results of the experiments showed that the proposed predictive model exhibited good performance in predicting time series data with a clear cycle and high correlation between variables, and improved the accuracy by approximately 3% to 5% compared with that of previous studies' time series prediction models.

**INDEX TERMS** Artificial neural networks, predictive models, time series prediction.


## I. INTRODUCTION

A multivariate time series is sequential data with values of several variables at a regular time unit. Multivariate time series prediction refers to predicting multiple variables at a future point in time after a certain period based on multivariate time series data in a specific period. This subject has been actively studied because it can be applied to various fields. Precisely forecasting variables can influence the decision-making process and strategy establishment in various applications such as manufacturing [1], medical field [2], tourism [3], and transportation [4]. Therefore, it is crucial to design a framework that efficiently and accurately forecasts future variables.

With the development of AI-based forecasting methods, recent studies have presented a time series prediction framework with deep learning models. Researches on designing a

time series predictive model using recursive models such as recurrent neural network [5] (RNN), long-short term memory (LSTM) [6], and gated recurrent units (GRU) [7] are predominant. Recent studies have proposed the transformer model [8] for a time series prediction task to solve the long-term dependency problem and parallel operation limitations of the recursive models. These deep learning models are superior to conventional regression models and machine learning methods.

Although deep learning-based models show good prediction results, there are structural limitations for directly applying these models to multivariate time series prediction. First, neural network models used in previous studies have specialized structures enabling them to process a single sequence. Early studies using deep learning models mainly compose a multivariate time series prediction framework with a structure in which the same neural networks are collocated in parallel in as many as the number of variables. Recent studies have designed models to deal with these problems, but they

The associate editor coordinating the review of this manuscript and approving it for publication was Baozhen Yao .

cannot simultaneously extract correlations between variables and time-domain features. In addition, multivariate time series data have immense input sizes compared with single time series data, and models in existing studies do not have structures for compressing this massive amount of information. This problem appears as an overfitting or long computation time problem. In a time series prediction task, where the input's longer time-axis data affect the prediction accuracy, increasing the receptive field range with an information compression procedure is essential.

In this study, we propose a convolutional transformer model to handle the limitations of existing time series predictive models. The proposed model has a transformer structure [8] with convolutional neural networks (CNN). The encoder and decoder in the transformer model are composed of a self-attention layer, point-wise CNN layer, and one-dimensional CNN layer. The encoder extracts the compressed spatiotemporal features from the given multivariate time series data by combining referred constituent layers. The prediction result is directly derived by analyzing the spatiotemporal features extracted from the encoder through the decoder layer with a similar structure.

The proposed transformer model has a great potential for contributing to the field of multivariate time series prediction. Our model is unique in that it simultaneously extracts the spatiotemporal features of the given multivariate time series inputs with a deeply designed transformer model, while the models of existing studies analyzed temporal features and inter-variable features through a separated process. Furthermore, the results of the extensive prediction experiments performed with various time series datasets show that the proposed model outperforms other existing time series predictive models in terms of forecasting accuracy.

## II. PREVIOUS WORK

Time series prediction problems have been studied in various fields owing to their utility and importance. Many studies have proposed forecasting models that predict a single variable rather than multiple variables. Previously, simple regression models such as linear regression or autoregressive integrated moving average (ARIMA) model were utilized for time-series forecasting [9], [10], [11]. Machine learning-based models such as support vector regression (SVR) [12] and classification and regression tree (CART) [13] are also used to deal with nonlinear time series data [14], [15], [16]. Lately, time series prediction studies with artificial neural network models such as RNN, CNN, and LSTM have become predominant [17], [18], [19], [20], [21], [22]. As the prediction methodology evolves, the proposed models can now precisely interpret time series data with complex patterns.

The deep learning-based prediction framework presented in the latest studies attentively interprets real-world time series data and shows highly-accurate prediction results. More specifically, forecasting models with artificial neural networks learn various patterns and features that cannot be interpreted by existing machine-learning techniques

and regression models, to output accurate prediction results [23], [24]. However, there is a crucial problem in utilizing the aforementioned univariate predictive models for multivariate predictions. Unlike univariate time series, multivariate time series data include both time-domain features and correlations between variables, which are essential for accurate forecasting. Existing single-variable forecasting models have architectures that specialize in analyzing only time-domain features. Therefore, the forecasting accuracy is inevitably lowered when using a univariate predictive model for a multivariate forecasting case.

Because of the mentioned properties of multivariate time series data, designing multivariate prediction frameworks is more complex than the univariate prediction case. Despite this difficulty, recent studies have suggested multivariate prediction frameworks that utilize deep learning manners. For example, Shih *et al.* devised a temporal pattern attention method to select relevant time series for multivariate forecasting [25]. Du *et al.* combined the attention mechanism in an encoder-decoder structure to extract multivariate correlations [26]. Huang *et al.* adopted the attention mechanism twice to reveal both temporal patterns and dependencies among variables [27]. Like these, several studies have proposed a multivariate prediction framework using the latest deep learning methods [28], [29], [30], [31], [32].

Existing multivariate prediction studies have demonstrated good prediction performance achieved through extensive experiments. However, the mentioned models do not provide a precise manner for implicitly reducing the amount of information in the entire multivariate data. Owing to the nature of time series forecasting, which is advantageous for prediction as the length of the time dimension is longer [33], condensing the extensive data is crucial. To address this issue, we designed a multivariate time series prediction model with an improved transformer structure.

## III. PRELIMINARY

This section introduces the terms and notations used in this study, and defines the multivariate time series prediction problem before explaining the proposed model.

### A. NOTATION

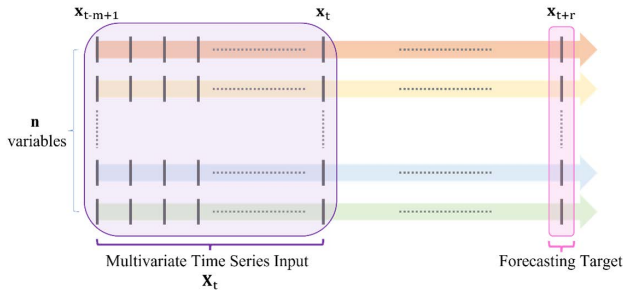
Let  $\mathbf{X}_t$  be the multivariate time series data at time point  $t$  to be inputted into the time series prediction framework. This is defined by (1).

$$\mathbf{X}_t = \{\mathbf{x}_{t-m+1}, \mathbf{x}_{t-m+2}, \dots, \mathbf{x}_t \mid \mathbf{x}_i \in \mathbb{R}^n\} \quad (1)$$

Here,  $m$  is the length of the input multivariate time series data, and  $\mathbf{x}_t$  is the multivariate vector at time point  $t$ . A multivariate vector  $\mathbf{x}_t$  contains  $n$  variables for a single time point. The value of  $n$  depends on the dataset used. This can be expressed by (2).

$$\mathbf{x}_t = \{x_{t(1)}, x_{t(2)}, \dots, x_{t(n)} \mid x_{t(i)} \in \mathbb{R}\} \quad (2)$$

Note that the numbers in parentheses represent enumerated variables that are not based on any specific criteria.



**FIGURE 1.** Description of input data and target data of multivariate time series prediction problem. A multivariate vector  $\mathbf{x}_{t+r} \in \mathbb{R}^n$  is inferred from the real-time series data  $\mathbf{X}_t \in \mathbb{R}^{m \times n}$ .

In summary, the multivariate time series input  $\mathbf{X}_t \in \mathbb{R}^{m \times n}$  represents the historical multivariate data for a certain period from  $(t - m + 1)$  to  $t$ .

**B. PROBLEM STATEMENT**

Time series forecasting problems involve predicting data at a future point in time based on past data. In this paper, we aim to predict a multivariate vector  $\mathbf{x}_{t+r} = \{x_{(t+r)(1)}, x_{(t+r)(2)}, \dots, x_{(t+r)(n)}\}$  of time point  $(t + r)$  with past multivariate time series data  $\mathbf{X}_t = \{\mathbf{x}_{t-m+1}, \mathbf{x}_{t-m+2}, \dots, \mathbf{x}_t\}$  as input. Note that  $r$  indicates the output interval, which is a specific future point. The inputs and targets of the defined problem are shown in Fig. 1.

**IV. PREDICTIVE MODEL**

We propose a novel transformer model for the defined multivariate time series prediction problem. Unlike the existing multivariate prediction models, the designed model extracts spatiotemporal features from a multivariate time series input using a one-forward procedure. In addition, the process of implicitly compressing information to extend the receptive field is also a unique feature of the proposed model. Note that The overall structure of the model is illustrated in Fig. 2.

**A. COMPONENTS**

The encoder and decoder blocks composing the transformer comprise several encoder and decoder layers. These layers consist of several elements. The components are positional encoding, one-dimensional (1D) dilated causal CNN, self-attention, and pointwise CNN. The elements represent pre-processing, temporal feature extraction, correlation extraction between variables, and additional weighted learning. Both the encoder and decoder layers operate through a combination of several components.

**1) POSITIONAL ENCODING**

Prior to inputting the time series data into the predictive model, time information is inserted into the input data. We used the positional encoding method to insert time-domain imprints into a given sequential input. The encoding technique was devised to add positional information to natural language data input in an existing transformer study [8]. The encoding value is a non-duplicate real number that represents each location. A positional encoding function

$PE(\cdot)$  for a given multivariate time series data  $\mathbf{X}_t$  is shown in (3).

$$PE(\mathbf{x}_i, 2k) = \mathbf{x}_i + \sin\left(\frac{\mathbf{x}_i}{10000^{\frac{2k}{d}}}\right)$$

$$PE(\mathbf{x}_i, 2k + 1) = \mathbf{x}_i + \cos\left(\frac{\mathbf{x}_i}{10000^{\frac{2k}{d}}}\right) \quad (3)$$

Note that  $x_i$  indicates the multivariate vector of  $\mathbf{X}_t$  at the  $i$ th time point. This procedure inserts positional information of multiple dimensions ( $d$ ) for a single instance, with the given input data  $\mathbf{X}_t$  shaped  $n \times m$ .

**2) DILATED-CAUSAL CNN**

A 1D CNN is a neural network that performs a convolution operation with a 1D filter. While the conventional CNN (two-dimensional CNN) is mainly used for image processing, the 1D CNN is used for 1D data tasks such as electronic signal processing and audio data analysis.

The 1D CNN is also widely used in time series prediction research. However, the time series forecasting problem differs significantly from other signal processing tasks in that future data cannot be referenced for prediction or long-term dependency owing to their periodic features. For these issues, variations in the 1D CNN are utilized for designing the time series forecasting model. In this study, a variation of the 1D CNN is applied to interpret the temporal features of the multivariate time series input.

The dilated convolution method can be used to handle sequential data. This convolution manner is a variation of the 1D-convolutional operation that compresses long-length information. By taking only a certain portion of the features from the previous layer, the amount of computation of the input values calculated for each layer can be reduced. The concrete formula for the dilated convolution operation is given by (4).

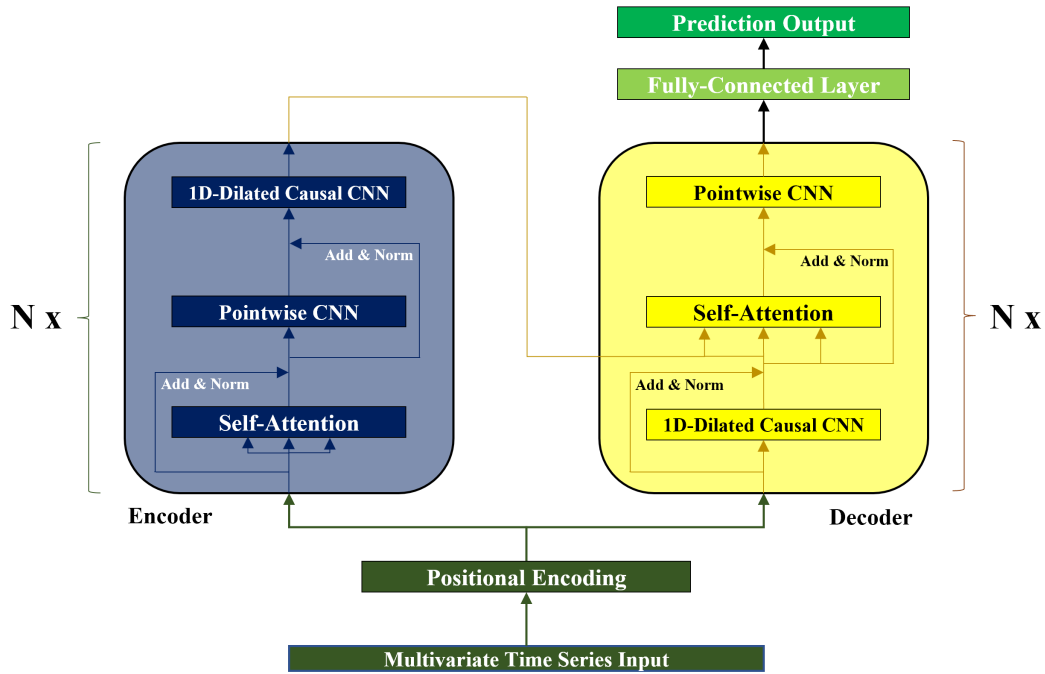
$$(F *_l k)(p) = \sum_{s+l=pt} F(s)k(t) \quad (4)$$

Here,  $F(\cdot)$  indicates the input feature,  $k(\cdot)$  is the convolutional filter, and  $l$  is the dilation rate. Note that  $p, s$  and  $t$  refers to the positions of the features. The number of output features was reduced by  $1/l$  because only  $1/l$  of the total calculation was performed. With a deeper dilation layer, the neural network can cover a wider receptive field, leading to the analysis of a longer sequence input.

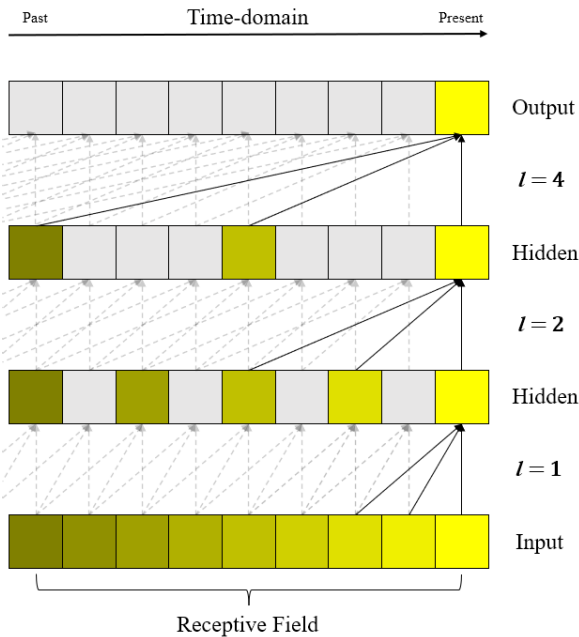
The causal convolution is a 1D CNN technique that considers that current data are only affected by past data. While the normal convolutional network is an operation computes through values adjacent to each other in both filter directions, the causal convolution operation proceeds with values adjacent to each other in one order (past direction) in each filter. The causal convolution operation through the convolution filter is given by (5).

$$(F *_l k)(p) = \sum_{p-t+1}^p F(s)k(t) \quad (5)$$

The notations in (5) are the same as those in (4).



**FIGURE 2.** Overall structure of the proposed convolutional transformer model. The transformer consists of  $N$  encoder layers (left part) and  $N$  decoder layers (right part).



**FIGURE 3.** Illustrated procedure of 1D Dilated-Causal CNN. The kernel size of the CNN is 3 and the stride is 1. The deeper the convolutional layer, the wider is the range of inputs that can be covered in the network.

The proposed model utilizes a dilated-causal convolutional neural network (DCCNN) that combines the two mentioned methods to obtain temporal features of multivariate time series data. Using this convolutional layer, the model can extract compressed features from a given long time series input. The process by which a DCCNN extracts features from a given information is schematically shown in Fig. 3.

### 3) SELF-ATTENTION

The attention mechanism plays a key role in the transformer model. Determining the correlation between elements in a single input is called self-attention. This mechanism can determine the relevance of features that are far from each other in the time domain, which recurrent models structurally unextractable. Therefore, it is mainly used in natural language processing to determine the relevance between elements. This study utilized the self-attention mechanism to analyze the correlation between variables in multivariate time series data.

The initial input of the attention layer is converted into a Query, Key, and Value. A Query indicates the data affected by a particular value and a Key represents the data that affects a particular data value. The Value expresses the weight of the influence. The Query, Key, and Value are computed by multiplying the same initial input by the independent weight matrices for each output. With positional-encoded input data  $\mathbf{X}_t$ , the procedure of gathering the corresponding Query ( $Q_t$ ), Key ( $K_t$ ), and Value ( $V_t$ ) using each weight matrix ( $W_q, W_k, W_v$ ) is shown in (6).

$$Q_t = W_q \mathbf{X}_t \quad K_t = W_k \mathbf{X}_t \quad V_t = W_v \mathbf{X}_t \quad (6)$$

The Query, Key, and Value gathered using (6) are transmitted to the attention layer. The attention layer obtains the attention score using the input Query and Key. The attention score, which refers to the variable having a significant effect on a variable in the multivariate input, is converted into a Query and Key. Note that the dimensions of the  $Q_t, K_t$ , and  $V_t$  are  $n \times m$  which are equal to the initial input. In this study, the dot product method proposed by Luong *et al.* [34] was used for attention score computation. We gathered the attention

score  $\text{Score}(Q_t, K_t)$  through  $Q_t$  and  $K_t$  using (7).

$$\text{Score}(Q_t, K_t) = \text{Tanh}(W'_q Q_t + W'_k K_t + b) \quad (7)$$

Note that  $W'_q$  and  $W'_k$  are weight matrices that are additionally calculated for  $Q_t$  and  $K_t$ , respectively;  $b$  is the bias, and  $\text{Tanh}(\cdot)$  is the activation function, which is a tangent hyperbolic function. The obtained  $\text{Score}(Q_t, K_t)$  has the dimension of  $n \times n$ .

We then obtain the Content  $C_t$  by using the attention score  $\text{Score}(Q_t, K_t)$  and Value  $V_t$ . Content is the final output feature of the attention layer, and indicates the result of the correlation between the input data for each variable. The  $C_t$  value was obtained using (8).

$$C_t = \text{SoftMax}(\text{Score}(Q_t, K_t))V_t \quad (8)$$

The final result of the attention layer,  $C_t$  has a matrix form with the same shape as the positional-encoded initial input, and it represents the variable-wise relevance of the input features.

#### 4) POINTWISE CNN

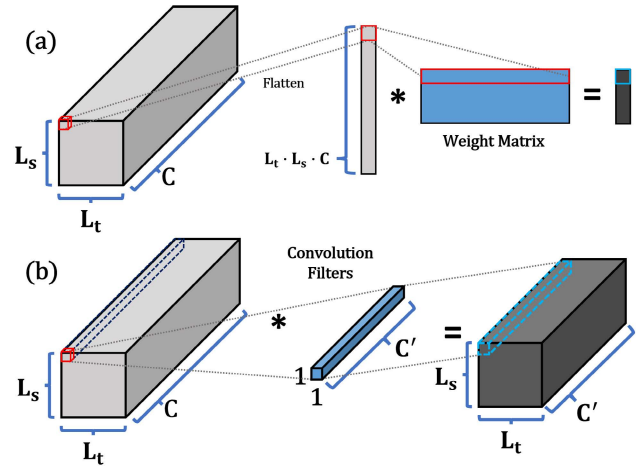
Existing transformer models weigh the attention layer outputs using a fully-connected neural network (feed-forward neural network). Through a fully-connected neural network, features can be better extracted through a weight matrix operation and can be transformed to fit the output shape. However, the fully-connected neural network does not consider the spatiality of multidimensional inputs. Therefore, it has a drawback in interpreting two-dimensional multivariate time series inputs, which include time-domain and variable features.

We leverage a point-wise convolutional neural network (PCNN) to analyze multivariate time series inputs instead of a fully-connected neural network. A PCNN is a variation of the CNN that operates with a convolutional filter of size 1. The PCNN is similar to the fully-connected neural network in that one value is multiplied by one weight. However, it is different in that it operates while maintaining the shape of the input. In addition, this neural network has the advantage of adjusting the channels of the features through the number of convolution filters, as in other convolution operations. The difference between a fully-connected neural network and PCNN is shown in Fig. 4.

#### B. ENCODER

The key role of the encoder layer is to extract spatiotemporal features while compressing the size of a given multivariate time series input. The encoder layer consists of the following layers: Self-attention, PCNN, and DCCNN. Compressed spatiotemporal features are returned by passing through a single encoder layer. Table 1 presents the configuration details of the encoder layers.

The first layer of the encoder block receives positional-encoded multivariate time series data as the input. Next, the self-attention layer analyzes the variables' relevance of the given multivariate data. The extracted features are then subjected to a 2-layer PCNN operation for additional learning



**FIGURE 4.** Two types of neural networks. (a) Fully-connected neural network and (b) PCNN. While the fully-connected neural network loses the spatiality of input features, PCNN can retain the spatiality of input features while adjusting the output size (number of channels).

while maintaining spatial information. Finally, the 2-layer DCCNN compresses the overall feature size while extracting the temporal features from the processed information. By stacking multiple deep encoder layers, the entire encoder block can better interpret the complex patterns of a given multivariate time series input.

Additionally, we considered the Add and Norm process between two neighboring component layers. The Add and Norm process consists of residual connection and layer normalization operations. The residual connection process links the results of each component layer with the features before the component layer. Layer normalization is a normalization method for finding the mean and variance of the features in a batch. Unlike other normalization techniques such as batch normalization and weight normalization, layer normalization is advantageous for handling sequential data features. These two methods prevent the vanishing gradient issue in the stacked encoder networks.

The final output of the entire encoder block is a spatiotemporal feature, with a size much smaller than that of the initial input. This feature was included in the prediction-generating procedure for the decoder layer.

#### C. DECODER

The decoder generates the final prediction result using spatiotemporal features extracted from the encoder. In our transformer model, a single decoder layer consists of components in a different order than that of an encoder. The asymmetric structure is empirically designed to output accurate prediction results from a given input using one-way analysis. One decoder layer consists of the following components: a DCCNN, Self-attention and PCNN. Similar to the encoder block, the decoder block is stacked with multiple decoder layers to generate the final prediction result from the initial input. The detailed specifications of the decoder layer are listed in Table 2.

The first decoder layer receives the positional-encoded multivariate time series data as the input. The 2-layer



**TABLE 1. Structure of a single encoder layer.**

Component	Output Channel	Details
Self-attention	Same as input	-
PCNN-1		$F=1, S=1, P=Same$
PCNN-2		$F=1, S=1, P=Same$
DCCNN-1		$F=3, S=1, P=Same, l=2$
DCCNN-2		$F=3, S=1, P=Same, l=2$

$F$  indicates the convolution filter size,  $S$  refers stride,  $P$  is padding and  $l$  stands for dilation rate.

**TABLE 2. Structure of a single decoder layer.**

Component	Output Channel	Details
DCCNN-1	Same as input	$F=3, S=1, P=Same, l=2$
DCCNN-2	(# of Input channel) / 2	$F=3, S=1, P=Same, l=1$
Self-attention	Same as input	-
PCNN-1		$F=1, S=1, P=Same$
PCNN-2		$F=1, S=1, P=Same$

$F$  indicates the convolution filter size,  $S$  refers stride,  $P$  is padding and  $l$  stands for dilation rate.

DCCNN of the decoder layer compresses the temporal information of the given input. Note that the second DCCNN reduces the number of channels of the input features augmented through positional encoding by half. This procedure is employed for the final output with a single value (one channel). Subsequently, the self-attention layer receives the final output of the encoder as a query and key, and the output of the previous DCCNN layer as a value. Finally, the self-attention layer returns the attention results as the final predictive outcomes. The last 2-layer PCNN additionally extracts features from the output attention result to derive the final prediction result. Note that the internal networks of the decoder layer include the Add and Norm process for dealing with gradient vanishing issues.

The final forecasting results are returned from the features of the decoder block with a fully-connected layer. The fully-connected layer is the last layer of the transformer model, and it consists of a single-layer weight without an activation function or dropout. The outputs of the last decoder layer are transformed to fit the input of the fully-connected layer through the flattening process. The final returns from the last layer are vector-shaped multivariate values  $\hat{\mathbf{x}}_t$ , which indicate the prediction of  $\hat{x}_t$ .

## V. EXPERIMENTS

We performed comparative, ablation, and in-depth experiments to evaluate the designed transformer model. This section describes the details of the prediction experiment, and discusses the performance results.

### A. SETUP

#### 1) DATASET

We considered four multivariate time series datasets to evaluate the performance of the proposed time series prediction model. The multivariate dataset considered datasets of *traffic*, *exchange rate*, *electricity*, and *solar energy* [35]. These public datasets are mainly used as benchmark datasets in time

**TABLE 3. Details of experimental multivariate time series datasets.**

Dataset	Total Length	# of variables	Time Unit
<i>Traffic</i>	17,544	862	1 Hour
<i>Exchange Rate</i>	7,588	8	1 Day
<i>Electricity</i>	26,304	321	1 Hour
<i>Solar Energy</i>	52,560	137	10 Minutes

series prediction research. Details of the datasets are listed in Table 3.

### 2) DATA PREPARATION

#### a: PREPROCESSING

We standardized the datasets. The multivariate time series data were standardized for every time series of a single variable. Model learning was stabilized by fixing variables with different numerical scales to a constant scale (mean 0 and variance 1). Along the time axis of the preprocessed dataset, 80% of the front part was used as the training data, and 20% of the rear part was used as the test data.

#### b: DATA SEGMENTATION

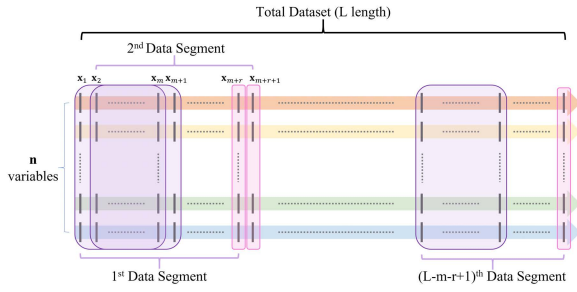
The proposed framework forecasts the expected multivariate with the given historical multivariate time series, as mentioned in Section III-B. We prepared the training and test data by splitting the given dataset into data segments. The data segments, consisting of the input matrix and output vector, were created using the sliding window method. The sliding window method is a data segment generation method that slides the *window* into a single time unit for the entire dataset period. For example, with the length of the given data  $L$  and the size of the window being  $(m+r)$ , a total of  $L-(m+r)+1$  data pieces were created. Note that the first  $m$  multivariate data of a  $(m+r)$ -sized window is the input ( $\mathbf{X}_t$ ), and the  $(m+r)$ -th multivariate vector is the ground truth ( $\mathbf{x}_{t+r}$ ). The process of creating data segments in a sliding window manner for a given dataset is shown in Fig. 5.

### 3) HYPERPARAMETER DETAILS

As in a previous study [8], each of the encoder and decoder block consists of six layers. The number of encoding dimensions mentioned in Section IV-A1 was set to 64. We define the default input length ( $m$ ) and output interval ( $r$ ) as 90 and 90, respectively. The batch size of the input data was set to 32, and the number of training epochs was set to 100. We utilize Adam [36] optimizer with a learning rate of 0.001. The loss function of the model is Mean Squared Error (MSE).

### 4) EVALUATION METRICS

In this study, the root-mean-square error (RMSE), rooted relative squared error (RRSE), and correlation coefficient (CORR) were used as evaluation metrics. These three indicators are evaluation metrics that are mainly used in existing time series prediction and regression studies. The definitions of these indicators are discussed herein.



**FIGURE 5. Illustration showing the process of creating data segments. A data segment consists of an input matrix  $\mathbf{X}_t$  and an output vector  $\mathbf{x}_{t+r}$ . Note that  $\mathbf{X}_t \in \mathbb{R}^{m \times n}$  and  $\mathbf{x}_{t+r} \in \mathbb{R}^n$ .**

The RMSE is the positive square root of the squared error sum divided by the total number of test data points. The loss value increases in proportion to the increase in the error of the individual values because the RMSE is derived by squaring the difference between the actual and predicted values, The RMSE indicates how accurately the predictive model fits the individual values. The RMSE is gathered with the total length of the test data ( $T_{\text{test}}$ ) and the number of variables ( $n$ ) using (9).

$$\text{RMSE} = \sqrt{\frac{1}{T_{\text{test}}} \sum_{i=1}^{T_{\text{test}}} \sum_{j=1}^n (\mathbf{x}_{i(j)} - \hat{\mathbf{x}}_{i(j)})^2} \quad (9)$$

The RRSE differs from RMSE in that it is not divided by the total amount of test data but by the statistic (square deviation) of the actual value of the test data. The RRSE value indicates how similarly the model predicts for that time series variable. Similar to the RMSE, the RRSE is also affected by the error of the individual values. The RRSE is obtained from the total length of the test data ( $T_{\text{test}}$ ) and the number of variables ( $n$ ), using (10).

$$\text{RRSE} = \sqrt{\frac{\frac{1}{n} \sum_{j=1}^n \sum_{i=1}^{T_{\text{test}}} (\mathbf{x}_{i(j)} - \hat{\mathbf{x}}_{i(j)})^2}{\sum_{i=1}^{T_{\text{test}}} \sum_{j=1}^n (\mathbf{x}_{i(j)} - \bar{\mathbf{x}}_j)^2}} \quad (10)$$

Here,  $\bar{\mathbf{x}}_j$  is the average value of the  $j$ th variable.

The CORR is the correlation coefficient between the actual and predicted values. The CORR value indicates whether the overall trend is well predicted for the test data. This metric is less sensitive than the RMSE and RRSE. The formula for calculating CORR is shown in (11).

$$\text{CORR} = \frac{1}{n} \sum_{j=1}^n \frac{\sum_{i=1}^{T_{\text{test}}} (\mathbf{x}_{i(j)} - \bar{\mathbf{x}}_j)(\hat{\mathbf{x}}_{i(j)} - \bar{\hat{\mathbf{x}}}_j)}{\sqrt{\sum_{i=1}^{T_{\text{test}}} (\mathbf{x}_{i(j)} - \bar{\mathbf{x}}_j)^2 (\hat{\mathbf{x}}_{i(j)} - \bar{\hat{\mathbf{x}}}_j)^2}} \quad (11)$$

Note that  $\bar{\hat{\mathbf{x}}}_j$  stands for the average value of the predicted  $j$ th variable.

## B. RESULTS AND DISCUSSIONS

### 1) PREDICTION PERFORMANCE

We first performed a prediction experiment with the hyperparameter specifications mentioned in Section V-A3. The specific metric results are described in comparison with the

experimental results. A comparison of the predicted results and ground-truth data is shown in Fig. 6.

We observed the prediction results based on dataset features. For the prediction results of the *exchange rate* dataset, the model predicted the overall trend with a small error. In addition, it predicted data with periodicity and high accuracy, as seen in the *Electricity\_var301* case. The proposed model showed accurate prediction results even for data where specific values are repeated at regular intervals (*solar energy* dataset). However, the model forecasted the expected value slightly inaccurately when dealing with data with large variance or rapidly-changing data such as *Traffic\_var175* and *Electricity\_var57*.

We also determined that the forecasting model predicted similar values with a slight lag from the actual values, regardless of the data characteristics. This error occurs when a distant time point is inferred from the input data. Nevertheless, the proposed model predicted the result with a time lag smaller than the defined output interval ( $m = 90$ ).

### a: COMPARISON WITH OTHER WORKS

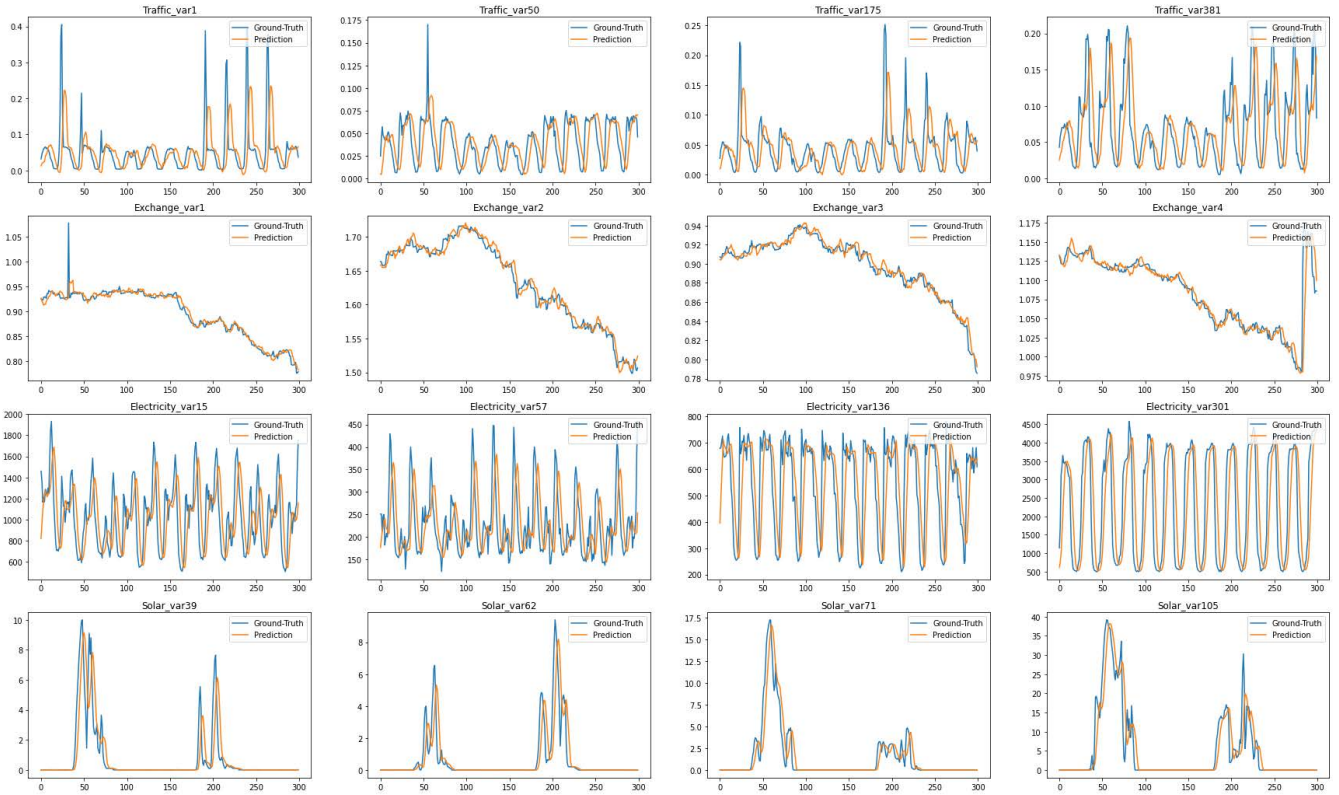
We designed a comparative experiment with existing prediction models to evaluate the objective performance of the proposed forecasting model. The comparison models are the general transformer [8] and the latest models of time series prediction such as LogSparse Transformer [33], Informer [37], LSTNet [35], and SpringNet [38]. The comparative test results were evaluated based on the three evaluation metrics mentioned in Section V-A4. The input/output shape of all models were the same,  $m = 90$  and  $r = 90$ . The experimental results are listed in Table 4.

As shown in Table 4, the prediction performance of the proposed model is almost the same as that of Informer and SpringNet. The *Solar-dataset* dataset result is slightly inferior to other recent models because the proposed model face adversity to predict accurate values (non-zero values) for data in which zero and non-zero values appear periodically. However, we note that our prediction model showed slightly better accuracy than the other prediction models on the *traffic* dataset and the *electricity* dataset. As described in Table 3 and Fig. 6, these two datasets contain many variables, and the patterns of the variables are similar to each other compared with the other two datasets (*exchange rate*, *solar energy*). Therefore, considering the precise forecasting results for this dataset type, it is apparent that the designed prediction model can better interpret multivariate data with a higher correlation between variables than the other existing forecasting models.

### b: COMPUTATION COSTS

We check how much computation cost is required for the designed model. We additionally note the FLOPs (Floating point Operations) of the proposed model and other comparative models by using the exact same computing resources. The results are in Table 5.

We observe that the proposed model has more FLOPs than the general transformer model. Still, the computational cost is slightly lower than the prediction models of the latest studies.



**FIGURE 6.** Multivariate prediction results of the proposed model. They show the prediction results for an arbitrary time period (300 time units). It is not possible to show the results for all variables in one figure, so only the prediction results for four random variables in each dataset are shown.

**TABLE 4.** Prediction results of comparison experiments with existing time series forecasting models. The best results are in bold type, and the second best results are italic type.

Transformer	RMSE	RRSE	CORR	LogSparse	RMSE	RRSE	CORR	Informer	RMSE	RRSE	CORR
<i>Traffic</i>	0.671	0.685	0.489	<i>Traffic</i>	0.603	0.599	0.521	<i>Traffic</i>	0.299	0.323	0.695
<i>Exchange Rate</i>	0.591	0.633	0.491	<i>Exchange Rate</i>	0.571	0.607	0.535	<i>Exchange Rate</i>	<b>0.286</b>	<i>0.291</i>	<b>0.732</b>
<i>Electricity</i>	0.652	0.628	0.475	<i>Electricity</i>	0.538	0.588	0.519	<i>Electricity</i>	<i>0.297</i>	0.290	<i>0.717</i>
<i>Solar Energy</i>	0.679	0.731	0.496	<i>Solar Energy</i>	0.594	0.610	0.498	<i>Solar Energy</i>	<b>0.251</b>	<b>0.262</b>	<b>0.757</b>
LSTNet	RMSE	RRSE	CORR	SpringNet	RMSE	RRSE	CORR	Ours	RMSE	RRSE	CORR
<i>Traffic</i>	0.305	0.398	0.671	<i>Traffic</i>	<i>0.279</i>	<b>0.286</b>	<i>0.708</i>	<i>Traffic</i>	<b>0.278</b>	<i>0.289</i>	<b>0.711</b>
<i>Exchange Rate</i>	0.322	0.357	0.705	<i>Exchange Rate</i>	<i>0.301</i>	0.325	<i>0.720</i>	<i>Exchange Rate</i>	0.303	<b>0.288</b>	0.709
<i>Electricity</i>	0.317	0.371	0.698	<i>Electricity</i>	0.298	<b>0.284</b>	0.715	<i>Electricity</i>	<b>0.271</b>	<i>0.285</i>	<b>0.732</b>
<i>Solar Energy</i>	0.302	0.336	0.723	<i>Solar Energy</i>	0.273	0.278	0.743	<i>Solar Energy</i>	<i>0.269</i>	<i>0.273</i>	<i>0.748</i>

The reason of the lower FLOPs costs is the decoder structure is simplified compared to other transformer-based prediction models.

2) ABLATION STUDY

In this study, we evaluated the performance changes that occur while changing or removing the structure of the designed model.

a: POINTWISE CNN

An ablation experiment was performed on the pointwise CNN layers of the presented model. We replaced the PCNN layer of the proposed model with a feed-forward neural network (FFNN), similar to the existing transformer model. The output of the FFNN was designed to have the same shape as that of the existing PCNN layer. All the other structures and

hyperparameter settings were the same as those of the originally proposed model. The prediction results of this study are shown in Fig. 7.

We observed that the prediction performance of the redesigned model degraded, regardless of the datasets used. In particular, the prediction accuracy for the *exchange rate* dataset, which had a trend rather than periodicity, drastically deteriorated (approximately 13–15%). From this result, it is apparent that the PCNN layers of the proposed model, which preserve the spatiality of the hidden states, have a substantial effect on predicting the overall trend.

b: ENCODER-DECODER STRUCTURE

We proposed an asymmetric transformer model whose layer orders of the encoder and decoder are different. The perfor-



TABLE 5. Computation costs (FLOPs) of the experimental models.

Performance Results (FLOPs)		
Study	Traffic	Exchange Rate
Transformer	$3.16 \times 10^{17}$	$2.97 \times 10^{15}$
LogSparse	$3.33 \times 10^{17}$	$3.08 \times 10^{15}$
Informer	$3.21 \times 10^{17}$	$3.01 \times 10^{15}$
LSTNet	$3.42 \times 10^{17}$	$3.12 \times 10^{15}$
SpringNet	$3.47 \times 10^{17}$	$3.13 \times 10^{15}$
<b>Ours</b>	$3.19 \times 10^{17}$	$2.99 \times 10^{15}$
Study	Electricity	Solar Energy
Transformer	$7.85 \times 10^{16}$	$1.11 \times 10^{17}$
LogSparse	$8.01 \times 10^{16}$	$1.36 \times 10^{17}$
Informer	$7.97 \times 10^{16}$	$1.20 \times 10^{17}$
LSTNet	$8.04 \times 10^{16}$	$1.41 \times 10^{17}$
SpringNet	$8.06 \times 10^{16}$	$1.43 \times 10^{17}$
<b>Ours</b>	$7.95 \times 10^{16}$	$1.18 \times 10^{17}$

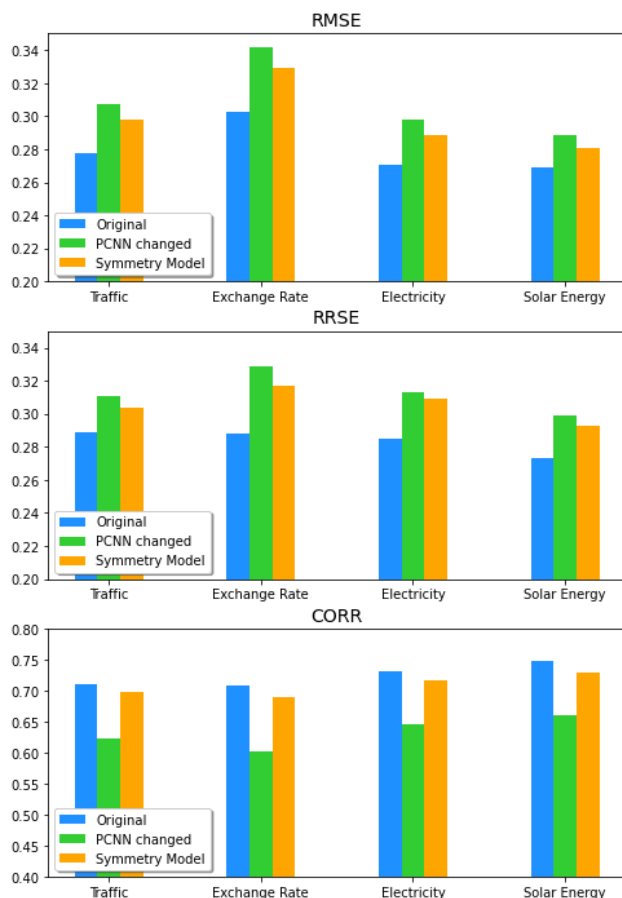


FIGURE 7. Ablation experiment results according to dataset and evaluation metrics.

performance significance of the asymmetric structure was evaluated by designing the encoder and decoder in the same layer order as in other existing transformer studies. A symmetric structure was designed by making the decoder structure identical to the encoder structure. The reordered model was tuned so that all other input/output flows proceeded in the same manner as in the original model. The rest of the model training settings were identical. The change in the prediction results

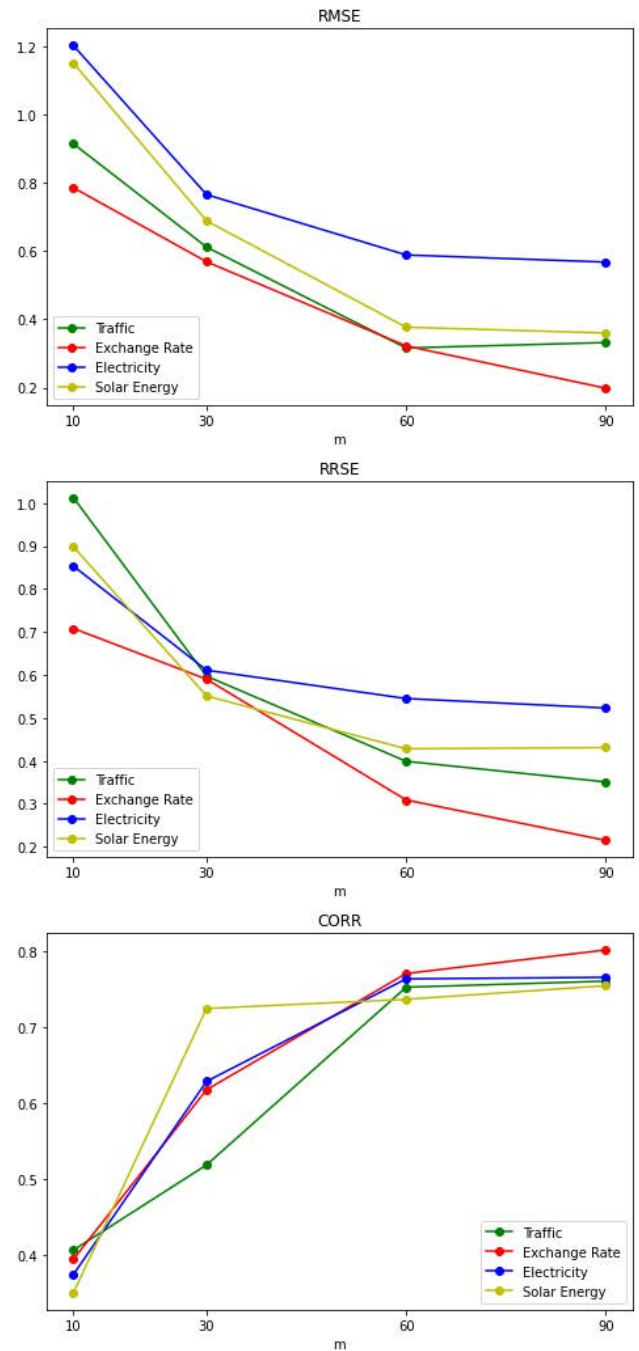


FIGURE 8. Evaluation metrics changes for each dataset according to the input length.

according to the structural transformation of the model is shown in Fig. 7.

The results of the CORR metrics in the referred figure were slightly worse than those of the original models. However, in the cases of the RMSE and RRSE metrics, the performance results were significantly lower than those of the original model (approximately 10–12%). The proposed asymmetric model predicted individual values better than the symmetric-structured model.

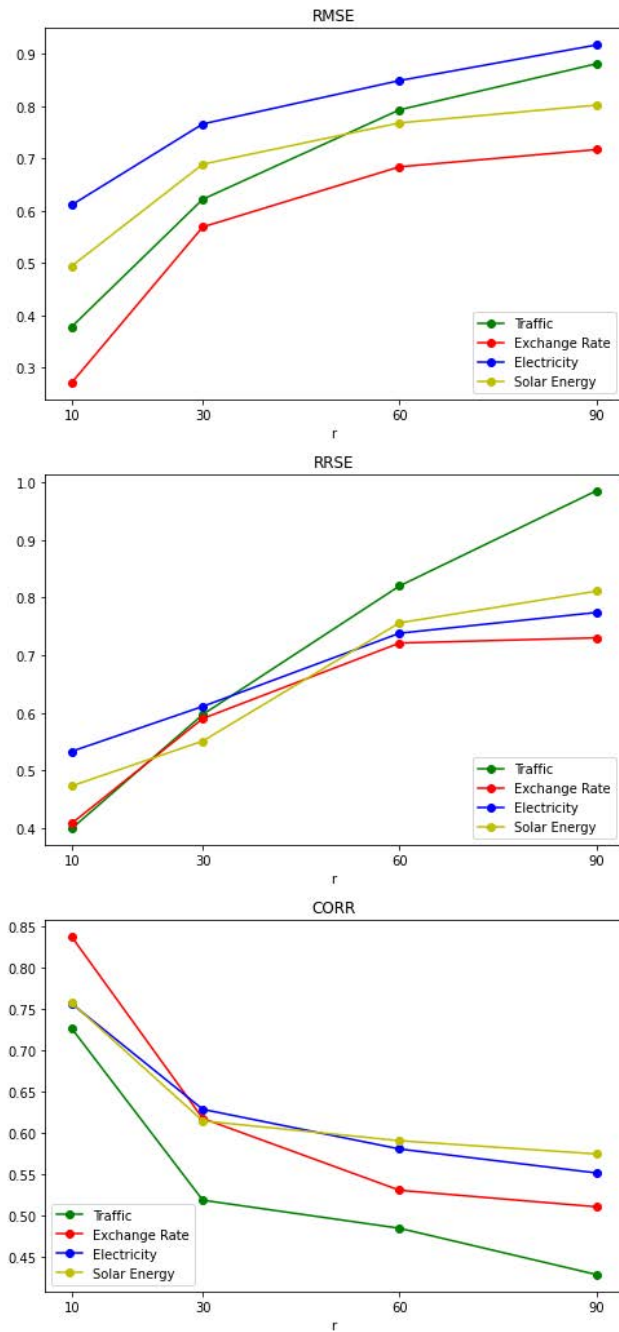


FIGURE 9. Evaluation metrics changes for each dataset according to the output interval.

### 3) IN-DEPTH EXPERIMENT

In addition, we evaluated how the performance of the proposed model changed by adjusting the hyperparameters of the input length ( $m$ ) and output interval ( $r$ ).

#### a: EXPERIMENT WITH INPUT LENGTH

In this experiment, we performed the test while the output interval was fixed and the input length was changed.  $r$  is fixed at 30, and the experiment was performed while changing  $m$  to 10, 30, 60, 90. The experimental results are shown in Fig. 8.

There was no significant difference in the accuracy between the prediction experiments with input time lengths of 60 and 90, except for the *exchange rate* dataset. We observed that the performance improved with a longer input, especially for data with a long-term trend, such as the *exchange rate* dataset. For the *electricity* and *solar energy* datasets, which have a distinct cycle of 24 hours, the prediction outcome with an  $m = 30$  showed a significant performance improvement in all metrics compared with the result with  $m = 10$ .

The results of this experiment show that the proposed model is significantly affected by the periodicity of the input data. In addition, we observed that the input length and performance are directly proportional to the data showing a long-term trend.

#### b: EXPERIMENT WITH OUTPUT INTERVAL

As in the previous experiment, we checked how the performance of the designed model changed as the output interval increased.  $m$  was fixed at 30, and  $r$  was changed to 10, 30, 60, and 90 to compare the forecasting performance of the near future and far future. The results of the prediction tests are shown in Fig. 9.

As shown in Fig. 9, the model shows a good performance in predicting a point in the near future ( $r = 10$ ) compared with the input length, while sharp decreases in performance in the CORR metrics are observed at  $r = 30$ . However, except for the RRSE in *traffic* dataset, there was no significant performance deterioration between  $r = 60$  and  $r = 90$ , which are tasks that predict a future point farther than the input length.

We observed that the performance of the proposed model decreased as the prediction interval increased, particularly in terms of overall trends. Furthermore, compared with the prediction results in Section V-B1, a longer input length is required to forecast distant future points.

## VI. CONCLUSION

We present a multivariate time series prediction model that leverages a convolutional neural network with a transformer model structure. The proposed model simultaneously analyzes the correlation between the input variables and temporal features of a given multivariate time series data in one model while the existing methodologies are difficult to deal with. In addition, we performed experiments using several time series datasets with different data characteristics to demonstrate the superior predictive performance of the designed model. The performance results of the extensive experiments proved that the proposed model enables multivariate prediction at a future time point with high accuracy for many variables and long sequences.

## REFERENCES

- [1] P. Shojaee, Y. Zeng, X. Chen, R. Jin, X. Deng, and C. Zhang, "Deep neural network pipelines for multivariate time series classification in smart manufacturing," in *Proc. 4th IEEE Int. Conf. Ind. Cyber-Phys. Syst. (ICPS)*, May 2021, pp. 98–103.
- [2] M. A. Morid, O. R. L. Sheng, K. Kawamoto, and S. Abdelrahman, "Learning hidden patterns from patient multivariate time series data using convolutional neural networks: A case study of healthcare cost prediction," *J. Biomed. Informat.*, vol. 111, Nov. 2020, Art. no. 103565.

- [3] C. Burger, M. Dohnal, M. Kathrada, and R. Law, "A practitioners guide to time-series methods for tourism demand forecasting—A case study of Durban, South Africa," *Tourism Manage.*, vol. 22, no. 4, pp. 403–409, 2001.
- [4] Y. Yin and P. Shang, "Forecasting traffic time series with multivariate predicting method," *Appl. Math. Comput.*, vol. 291, pp. 266–278, Dec. 2016.
- [5] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986.
- [6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [7] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," 2014, *arXiv:1406.1078*.
- [8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–11.
- [9] W. H. Kruskal and J. M. Tanur, "Linear hypotheses," *Int. Encyclopedia Statist.*, vol. 1, pp. 41–523, 1978.
- [10] F.-M. Tseung and G.-H. Tzeng, "A fuzzy seasonal ARIMA model for forecasting," *Fuzzy Sets Syst.*, vol. 126, no. 3, pp. 367–376, 2002.
- [11] G. Mélard and J.-M. Pasteels, "Automatic ARIMA modeling including interventions, using time series expert software," *Int. J. Forecasting*, vol. 16, no. 4, pp. 497–508, Oct. 2000.
- [12] H. Drucker, C. J. Burges, L. Kaufman, A. Smola, and V. Vapnik, "Support vector regression machines," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 9, 1996, pp. 1–7.
- [13] R. J. Lewis, "An introduction to classification and regression tree (CART) analysis," in *Proc. Annu. Meeting Soc. Academic Emergency Med. San Francisco*, vol. 14, 2000, pp. 1–14.
- [14] Q. Yin, R. Zhang, Y. Liu, and X. Shao, "Forecasting of stock price trend based on CART and similar stock," in *Proc. 4th Int. Conf. Syst. Informat. (ICSAI)*, Nov. 2017, pp. 1503–1508.
- [15] W. Xie, L. Yu, S. Xu, and S. Wang, "A new method for crude oil price forecasting based on support vector machines," in *Proc. Int. Conf. Comput. Sci. Berlin, Germany: Springer*, 2006, pp. 444–451.
- [16] K. J. Kim, "Financial time series forecasting using support vector machines," *Neurocomputing*, vol. 55, nos. 1–2, pp. 307–319, Sep. 2003.
- [17] A. Sagheer and M. Kotb, "Time series forecasting of petroleum production using deep LSTM recurrent networks," *Neurocomputing*, vol. 323, pp. 203–213, Jan. 2019.
- [18] I. E. Livieris, E. Pintelas, and P. Pintelas, "A CNN–LSTM model for gold price time-series forecasting," *Neural Comput. Appl.*, vol. 32, no. 23, pp. 17351–17360, Apr. 2020.
- [19] V. K. R. Chimmula and L. Zhang, "Time series forecasting of COVID-19 transmission in Canada using LSTM networks," *Chaos, Solitons Fractals*, vol. 135, Jun. 2020, Art. no. 109864.
- [20] N. Xue, I. Triguero, G. P. Figueredo, and D. Landa-Silva, "Evolving deep CNN-LSTMs for inventory time series prediction," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2019, pp. 1517–1524.
- [21] Y. Liu, H. Dong, X. Wang, and S. Han, "Time series prediction based on temporal convolutional network," in *Proc. IEEE/ACIS 18th Int. Conf. Comput. Inf. Sci. (ICIS)*, Jun. 2019, pp. 300–305.
- [22] T. Chen, H. Yin, H. Chen, L. Wu, H. Wang, X. Zhou, and X. Li, "TADA: Trend alignment with dual-attention multi-task recurrent neural networks for sales prediction," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2018, pp. 49–58.
- [23] S. Siami-Namini, N. Tavakoli, and A. S. Namin, "A comparison of ARIMA and LSTM in forecasting time series," in *Proc. 17th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2018, pp. 1394–1401.
- [24] D. Karmiani, R. Kazi, A. Nambisan, A. Shah, and V. Kamble, "Comparison of predictive algorithms: Backpropagation, SVM, LSTM and Kalman filter for stock market," in *Proc. Amity Int. Conf. Artif. Intell. (AICAI)*, Feb. 2019, pp. 228–234.
- [25] S.-Y. Shih, F.-K. Sun, and H.-Y. Lee, "Temporal pattern attention for multivariate time series forecasting," *Mach. Learn.*, vol. 108, nos. 8–9, pp. 1421–1441, Sep. 2019.
- [26] 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, May 2020.
- [27] S. Huang, D. Wang, X. Wu, and A. Tang, "DSANet: Dual self-attention network for multivariate time series forecasting," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.*, Nov. 2019, pp. 2129–2132.
- [28] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang, "Connecting the dots: Multivariate time series forecasting with graph neural networks," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2020, pp. 753–763.
- [29] K. Wang, K. Li, L. Zhou, Y. Hu, Z. Cheng, J. Liu, and C. Chen, "Multiple convolutional neural networks for multivariate time series prediction," *Neurocomputing*, vol. 360, pp. 107–119, Sep. 2019.
- [30] M. Han and M. Xu, "Laplacian echo state network for multivariate time series prediction," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 1, pp. 238–244, Jan. 2018.
- [31] S. D. Khan, L. Alarabi, and S. Basalamah, "Toward smart lockdown: A novel approach for COVID-19 hotspots prediction using a deep hybrid neural network," *Computers*, vol. 9, no. 4, p. 99, 2020.
- [32] M. Ullah, M. M. Yamin, A. Mohammed, S. D. Khan, H. Ullah, and F. A. Cheikh, "Attention-based LSTM network for action recognition in sports," *Electron. Imag.*, vol. 2021, no. 6, pp. 1–302, 2021.
- [33] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang, and X. Yan, "Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–11.
- [34] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," 2015, *arXiv:1508.04025*.
- [35] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling long- and short-term temporal patterns with deep neural networks," in *Proc. 41st Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2018, pp. 95–104.
- [36] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
- [37] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Proc. AAAI*, 2021, pp. 1–10.
- [38] Y. Lin, I. Koprinska, and M. Rana, "SpringNet: Transformer and spring DTW for time series forecasting," in *Proc. Int. Conf. Neural Inf. Process. Cham, Switzerland: Springer*, 2020, pp. 616–628.



**DONG-KEON KIM** received the B.S. degree in systems management engineering from the School of Electronic and Electrical Engineering, Sungkyunkwan University, in 2020, and the M.S. degree from the Department of Software, Sungkyunkwan University, in 2022. His research interests include computer vision, machine learning, and time series forecasting.



**KWANGSU KIM** (Member, IEEE) received the Ph.D. degree in computer science from the University of Southern California, in 2007. He served as the Director-General of the Ministry of Science and ICT, South Korea. He is currently a Professor with the Department of Software, Sungkyunkwan University. He is also the Director of Sungkyun AI Research Institute. His research interests include computer vision, domain adaptation, federated learning, AI applications, and explainable AI.