IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 26, 2021, accepted October 1, 2021, date of publication October 27, 2021, date of current version November 2, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3122910

StackDA: A Stacked Dual Attention Neural Network for Multivariate Time-Series Forecasting

JUNGSOO HONG, JINUK PARK, AND SANGHYUN PARK, (Member, IEEE)
Department of Computer Science, Yonsei University, Seoul 03722, South Korea

Corresponding author: Sanghyun Park (sanghyun@yonsei.ac.kr)

This work was supported by the Institute of Information and Communications Technology Planning and Evaluation (IITP) Grant by the Korean Government through Ministry of Science and ICT (MSIT) [(SW Star Lab) Research and Development of the high performance in-memory distributed DBMS based on flash memory storage in an IoT environment] under Grant IITP-2017-0-00477.

ABSTRACT Multivariate time-series forecasting derives key seasonality from past patterns to predict future time-series. Multi-step forecasting is crucial in the industrial sector because a continuous perspective leads to more effective decisions. However, because it depends on previous prediction values, multi-step forecasting is highly unstable. To mitigate this problem, we introduce a novel model, named stacked dual attention neural network (StackDA), based on an encoder-decoder. In dual attention, the initial attention is for the time dependency between the encoder and decoder, and the second attention is for the time dependency in the decoder time steps. We stack dual attention to stabilize the long-term dependency and multi-step forecasting problem. We add an autoregression component to resolve the lack of linear properties because our method is based on a nonlinear neural network model. Unlike the conventional autoregressive model, we propose skip autoregressive to deal with multiple seasonalities. Furthermore, we propose a denoising training method to take advantage of both the teacher forcing and without teacher forcing methods. We adopt multi-head fully connected layers for the variable-specific modeling owing to our multivariate time-series data. We add positional encoding to provide the model with time information to recognize seasonality more accurately. We compare our model performance with that of machine learning and deep learning models to verify our approach. Finally, we conduct various experiments, including an ablation study, a seasonality determination test, and a stack attention test, to demonstrate the performance of StackDA.

INDEX TERMS Attention mechanism, autoregressive model, denoising training, multi-step forecasting, multivariate time-series forecasting.

I. INTRODUCTION

Time-series forecasting is a task that predicts future time-series using historical time-series data. It is used to analyze and predict patterns to make decisions regarding the future or to detect abnormal situations. Furthermore, it can be divided into univariate and multivariate time-series forecasting based on the number of variables used in the modeling. In the modern era, most data consist of many variables because multiple sensors are used, for example, to sample data from multiple plants, record data from multiple clients, and collect measured data on freeways [1]. Therefore, multivariate time-series forecasting has been studied extensively in various contexts, such as the industrial sector [2], energy sector, electricity consumption [3], and financial markets [4], [5]. However, the different seasonalities among multiple variables make multivariate forecasting more challenging than univariate forecasting. Moreover, traditional time-series forecasting tends to conduct one-step forecasting rather than multi-step forecasting, particularly in deep learning methods. One-step forecasting predicts only one time step, whereas multi-step forecasting predicts a complete cycle of future data over a relatively long time range. In the actual industrial sector, multi-step forecasting is highly significant because decisions are made based on future perspectives using multi-step forecasting. However, multi-step forecasting has to accurately describe future time-series patterns, which is a challenging task because the cumulative prediction increases

The associate editor coordinating the review of this manuscript and approving it for publication was Manuel Rosa-Zurera.

the variance. Furthermore, there is a problem that prediction is performed by simply following the actual time-series or predicting its shape differently [6].

Moreover, time-series forecasting experiences not only the multi-step forecasting problem, but also the multi-seasonalities problem. In time-series forecasting, the time-series are composed of multiple iterative patterns. Multiple seasonalities can be distinguished by short- and long-term patterns [1]. In real-world datasets, short-term patterns exist within a day and long-term patterns exist in daily seasonalities such as weekdays and weekends. As these long- and short-term patterns are interrelated, pattern prediction becomes complicated when multiple seasonalities exist [7]. In this study, we alleviate the described problems and focus on multi-step forecasting in a multivariate time-series.

Statistical methods for time-series forecasting have been used extensively because they are linear models that can identify the linear features of data [7]. However, these statistical methods cannot capture nonlinearity, and it is challenging to capture multiple seasonalities. Furthermore, many hyperparameters are necessary that require knowledge from experts in associated fields. However, deep neural networks are specialized in capturing nonlinearities. The recurrent neural network (RNN) [8] has exhibited structural advantages in time-series modeling owing to its recurrent structure; however, it has limitations in terms of long-term dependencies. The convolutional neural network (CNN) [9] and Attention [10], [11] can address the long-term dependency problem; nevertheless, they exhibit the limitation that several convolutional layers are required to capture long-term seasonality. Sequence-to-sequence (Seq2Seq) [12] solves the shortcomings of the previous method; however, it has the limitation that multi-step forecasting is challenging. Moreover, the multivariate modeling of existing deep neural networks includes all the variables simultaneously; hence, it is difficult to obtain variable-specific models.

In this study, we propose a novel model—stacked dual attention neural network (StackDA)—to solve the limitations of previous studies. The proposed model uses a stacked dual attention mechanism to solve long-term dependency problems. We propose a novel autoregressive model, namely skip autoregressive (Skip AR), for capturing complex seasonalities. Moreover, we propose a method to improve the forecasting performance by modeling multivariate time-series data as a model that is specialized for each variable using multi-head neural networks (Multi Head). Finally, we propose a denoising training method that eliminates the exposure bias problem by accelerating the training and positional encoding (PE), which effectively performs multi-step forecasting by providing the positional information of the time-series.

The primary contributions of this study can be summarized as follows:

- Skip AR for multiple seasonalities: We introduce the novel Skip AR model that differs from the conventional autoregressive model. It focuses on the corresponding time step to capture the seasonalities of the time-series. This provides satisfactory performance when the time-series contain diverse multiple seasonalities.
- Stacked dual attention mechanism for multi-step forecasting: We propose dual attention to learn the time dependency of the encoder-decoder and the dependency among the time steps of the decoder. Moreover, we stack attention to stabilize the multi-step forecasting.
- Denoising training method for stable forecasting: We propose a novel denoising training method for stable multi-step forecasting of the time-series.
- Variable-specific forecasting method: We improve the multivariate time-series forecasting performance by extracting and predicting variable-specific features through Multi Head.

The remainder of this paper is organized as follows. Section II reviews the related studies, and Section III defines the problem formulation. Section IV presents the proposed methods and model, and Section V introduces the details of the experimental environment and evaluation metrics. Section VI presents an experimental comparison between our results and those presented in the existing studies. Section VII draws the conclusions.

II. RELATED WORKS

One of the traditional models for univariate time-series forecasting is the autoregressive integrated moving average (ARIMA) model [7]. The ARIMA model and its variants consist of a linear combination of autoregression and a moving average. These statistical methods guarantee time-series stationarity, such as differencing. As these models are composed of linear combinations, they have the advantage of performing multi-step forecasting while exhibiting high interpretability. However, they are inappropriate for high-dimensional multivariate time-series forecasting because they require many hyperparameters for autoregression or moving average modeling and domain knowledge to set appropriate hyperparameters. Vector autoregression (VAR) [13]—a model commonly used in multivariate time-series forecasting—is a generalization of the AR model that inherits the advantage of clear and concise to interpretation. Nevertheless, VAR does not capture complex patterns because it lacks the capability to model nonlinearity. The Gaussian process (GP) [14]—a nonparametric method with relatively few hyperparameters—is also frequently used for time-series forecasting; however, it requires distribution assumptions and massive computations.

Prophet [15] and TBATS [16] are commercial APIs that can perform multi-step forecasting using time-series modeling. Prophet is an open-source framework for time-series forecasting that was developed by Facebook. It uses curve fitting with Fourier transform (FT) to solve the time-series modeling problem. Although Prophet can obtain flexible models and facilitate rapid learning, it is not easy to set hyperparameters according to the data. TBATS is a framework for fitting the ARMA model, which is a variant of ARIMA, based

on FT. It can be used to set arbitrary multiple seasonalities for hyperparameters; however, it is difficult to implement TBATS with insufficient domain knowledge regarding the data. In addition to these problems, it requires powerful computing performance, such as the Box-Cox transformation for time-series stationarity, as well as a long training time.

In recent years, deep neural networks have garnered considerable attention owing to their impressive capability to capture the nonlinearities of time-series [17], [18]. In particular, the RNN has exhibited structural advantages in modeling time-series owing to its recurrent structure. However, RNN and its variants [8], [19] have limited long-term time dependency drawbacks. LSTNet has been proposed to improve the time dependency problem by modeling the seasonality characteristics [1]. LSTNet uses both convolutional and recurrent layers, and performs one-step forecasting by capturing the local dependencies with the convolutional layer and the long-term dependencies with the recurrent layer. Recurrent-skip, which is used to capture the long-term dependencies in LSTNet, exhibits disadvantages when the time-series seasonality length is dynamic over time because it requires a predefined hyperparameter. To compensate for this, MTNet [20] incorporates a memory component into the model to store the long-term data and solve the long-term dependency problems. However, MTNet cannot yet perform multi-step forecastings.

All of the preceding deep neural network models have been one-step forecasting models. In practice, multi-step forecasting is more critical than one-step forecasting in many fields, such as electricity load forecasting [3] and stock market forecasting [21]. The Seq2Seq model [12], [22], [23], which has been successful in machine translation, is of particular interest in such multi-step forecastings. However, Seq2Seq has theoretical limitations because time-series variations have non-stationarity, the variance of which is not constant with the time flow [24]. Owing to the high variance and unstable training of multi-step forecasting, the model simply follows the actual time-series for prediction, or predicts the time-series shape itself differently. DILATE [6] was proposed as a solution to this problem. The loss function that has been proposed for DILATE uses dynamic time warping to calculate the loss occurring in the time and pattern. It can stabilize the multi-step forecasting; however, it exhibits the disadvantage of slow training owing to its high computational complexity. Accordingly, this study proposes a method for training to perform denoising on the model itself, rather than the conventional method for modifying the loss function. Moreover, we introduce the Skip AR for capturing multiple seasonality, and propose stacked dual attention to enable stable decoding for multi-step forecasting.

III. PROBLEM FORMULATION

This study focuses on the multi-step forecasting of $\{1, 2, \ldots, T\}$ time steps, rather than the one-step forecasting of the Y_t value at time step t. We do not use exogenous variables other than the provided time-series, and only model the seasonality of the provided time-series to predict the future time-series. The input time-series X is defined as a time-series up to time step L with n dimensions, as follows:

$$X = \{X_1, X_2, \ldots, X_L\}, \quad X \in \mathbb{R}^{n \times L}. \tag{1}$$

As this model does not use exogenous variables, we define it as an output time-series Y of length T with the same dimension as the input time-series as follows:

$$Y = \{Y_1, Y_2, \ldots, Y_T\}, \quad Y \in \mathbb{R}^{n \times T}. \tag{2}$$

IV. METHODS

In this study, we propose a novel method based on the encoder-decoder for multi-step time-series modeling. Fig. 1 depicts the overall structure of this method. The proposed method consists of a convolutional encoder, decoder using stacked dual attention, Skip AR, and a Multi Head. Furthermore, we propose a PE that can represent the time position of the time-series and a denoising training method for effective multi-step forecasting. In each of the following sections, the components are described in detail, and an optimization method for performing the final prediction and training is described.

A. CONVOLUTIONAL ENCODER

In this study, information of the time-series data is extracted using a one-dimensional convolution encoder cell. Feature vectors using the k^{th} convolution filter are calculated using (3).

$$e_k = ReLU\left(W_k * X + b_k\right). \tag{3}$$

As expressed in (3), the feature vector e_k is obtained using the convolution encoder and the $ReLU$ activation function for the input time-series X. The $*$ operation represents a convolution operation, and W_k represents the k^{th} filter. Moreover, W_k and b_k are learnable parameters. An $e_k \in \mathbb{R}^{\mathrm{L}}$ dimension using zero padding exists. Finally, the output value $E = \{e_1, e_2, \ldots, e_k\}$ is $E \in \mathbb{R}^{D_K \times L}$, where D_k denotes the number of filters.

B. DECODER USING STACKED DUAL ATTENTION

The decoder cell receives the previous prediction value as the input and executes multi-step forecasting. The decoder cell uses the gated recurrent unit (GRU) [25], which is an RNN-based model, and M decoder cells are stacked. As illustrated in Fig. 2 and Fig. 3, the decoder uses two attention mechanisms to complement the long-term dependencies, which is a disadvantage of RNN models. (1) Temporal attention (TA): The hidden state is obtained by the dot product of the encoder feature vector with PE and the decoder hidden state with added noise. (2) Masked causal attention (CA): This derives the result value, including the information between the decoder time steps. This mechanism models complex seasonalities. We stack attention to stabilize the multi-step forecasting, and construct a Multi Head to perform variable-specific decoding.

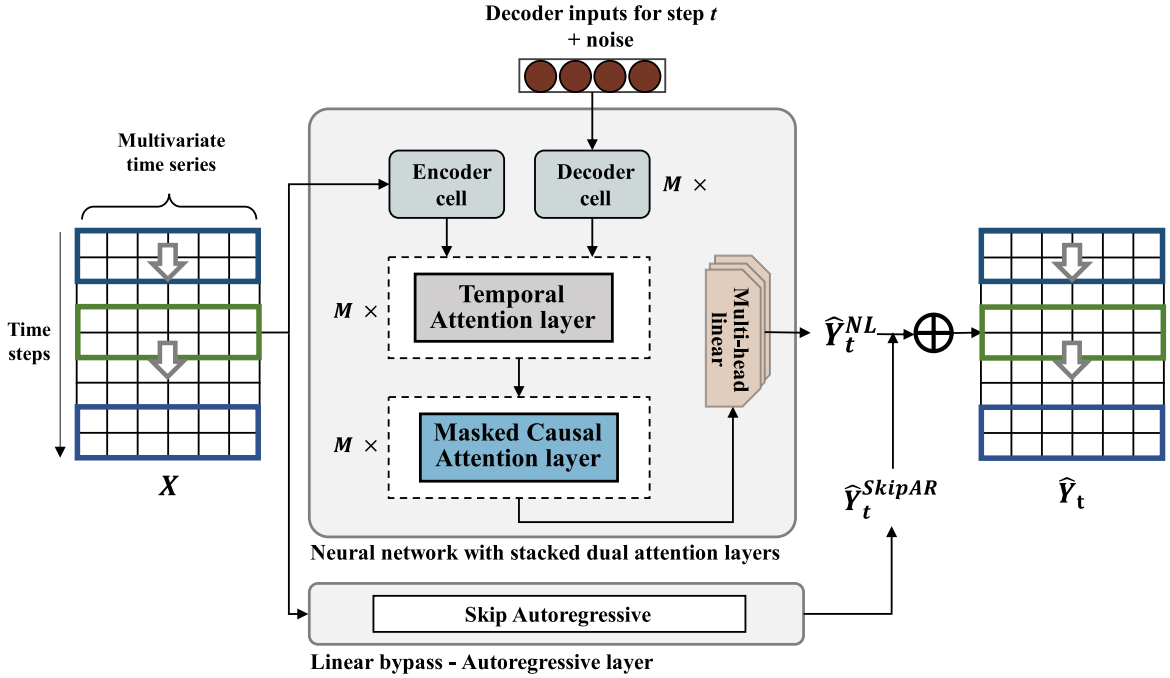


FIGURE 1. Overview of the proposed model, StackDA.

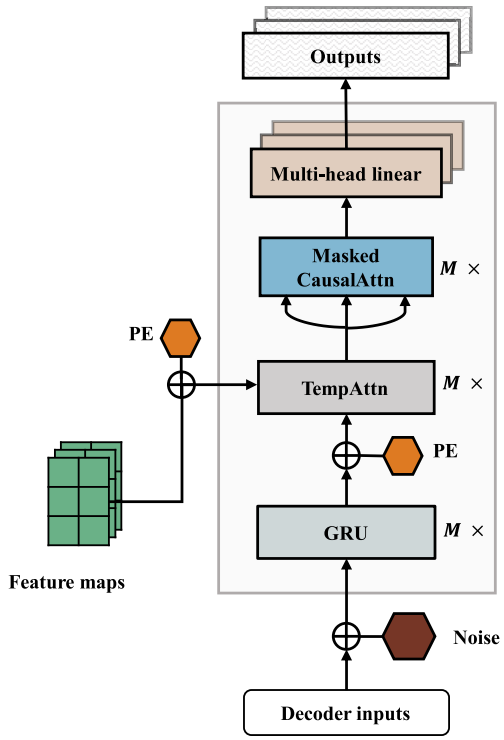


FIGURE 2. Details of the decoder architecture using stacked dual attention and multi head.

1) TEMPORAL ATTENTION

TA calculates the importance of the observed values in all of the past time steps at the prediction time step. It assembles past information according to the calculated importance, and dot product attention is used. At the decoding time step t of this model, the hidden state is extracted using the GRU decoder cell and the TA is calculated as follows:

$$d_t = GRU^{(M)}\left(d_{t-1}, \hat{y}_{t-1}\right), \tag{4}$$

$$TempAttn\left(h_i, h_j\right) = \text{softmax}\left(h_i^{\mathrm{T}} h_j\right), \tag{5}$$

$$\alpha_{t,k} = TempAttn(d_t, e_k), \tag{6}$$

$$c_t = \sum\nolimits_k \alpha_{t,k} e_k. \tag{7}$$

The GRU hidden state $d_t \in \mathbb{R}^{D_{model}}$ at time step t is calculated using the GRU decoder cell, as indicated in (4), where $GRU^{(M)}$ represents M stacked GRUs. The attention score α_t is calculated with the attention function M stacked $TempAttn\,(*)$ which uses the dot product between d_t and the encoder feature map E, as expressed in (5) and (6). Subsequently, the context vector c_t is calculated by the product of the encoder feature map E and attention score α_t at time step t, as expressed in (7). Thereafter, in (8), h_t is obtained through a linear transformation layer using c_t and d_t, as follows:

$$h_t = \sigma\left(W^{TA}\left[c_t; d_t\right] + b^{TA}\right). \tag{8}$$

2) CAUSAL ATTENTION

CA considers the dependency between time steps when performing predictions. As mentioned in Subsection IV.B.1, TA is a vector that reflects the time relationship between the time step of the encoder input time-series and time step t when executing the prediction. At $\{1, 2, \ldots, T\}$ time steps, when the output of the decoder is to be predicted, it depends only on the recurrent structure of the GRU immediately

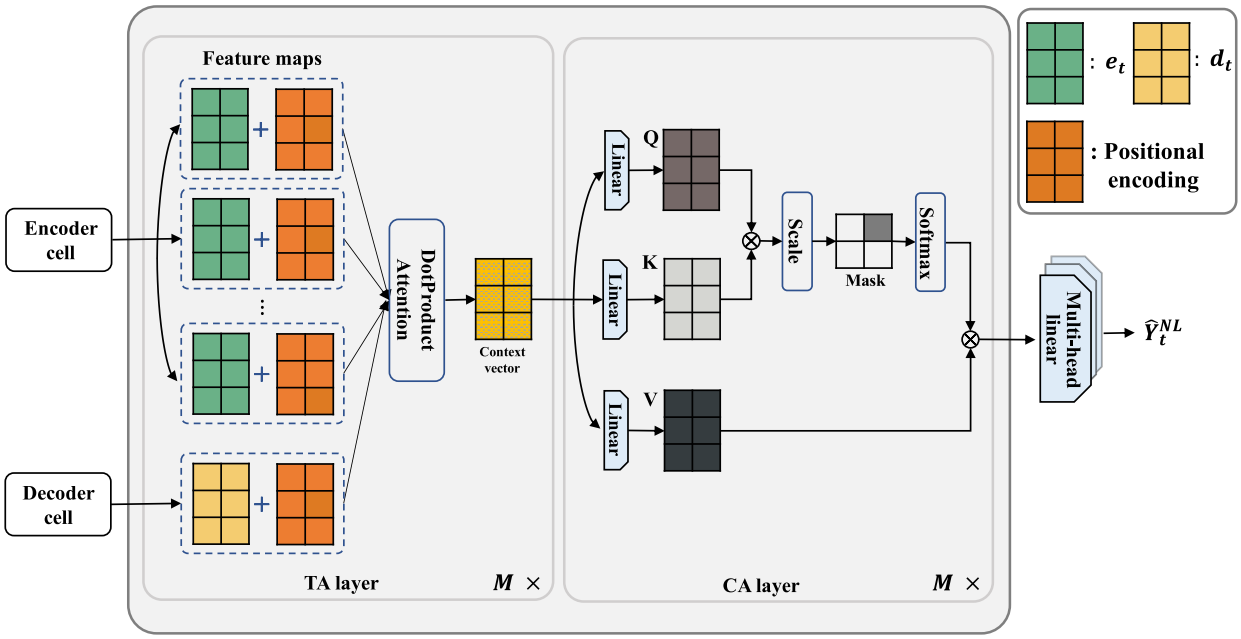


FIGURE 3. Details of the TA and masked CA layers.

before the information of the time step. Therefore, long-term dependency problems can arise when forecasting multiple steps rather than one time step.

To complement this, we apply self-attention [26] to express the information regarding the relationship within the decoder time steps:

$$S = CausalAttn\left(q(H), k(H), v(H)\right), \tag{9}$$

$$CausalAttn\left(Q, K, V\right) = \text{softmax}\left(\frac{QK^T}{\sqrt{D_{model}}}\right)V, \tag{10}$$

where $H = \{h_1, h_2, \ldots, h_T\}$ is the output of Subsection VI.B.1, and H is used in the attention function M stacked $CausalAttn\left(*\right)$. In (9) and (10), we obtain $S \in \mathbb{R}^{D_{model} \times T}$ using $CausalAttn\left(*\right)$, which reflects the information between each decoding time step. The functions $q\left(.\right), k\left(.\right)$, and $v\left(.\right)$ represent a learnable linear transformation and D_{model} is the hidden layer size.

During training, we apply the mask to prevent attention from being applied to the future time step values. That is, masks are used to make predictions to guarantee the causality of time. At time step t, we mask the hidden values of the $\{t+1, \ldots, T\}$ time steps and only use the hidden values of the $\{1, 2, \ldots, t\}$ time steps to implement the attention. A mask is not used in the test because the future prediction value is not provided in the test time as the decoder input value.

C. SKIP AUTOREGRESSIVE LAYER

As our method is based on a nonlinear neural network model, the scale of the output value does not adequately reflect the scale of the input time-series. Owing to the nonperiodic scale change in the input time-series, the scale reflection failure significantly degrades the performance of the prediction model [7].

To address this problem, we adopt the linear additive model, as in existing studies [1]. This component solves the problem of the lack of linear characteristics and enhances the scale reflection of the model [27], [28]. Similar to the highway network concept [29], this model composes the final prediction values by decomposing them into linear and nonlinear parts, as described in Subsection IV.G. Therefore, the lack of scale is addressed in the linear part, and the capturing of multiple seasonalities is modeled in the nonlinear part. Moreover, we modify the conventional autoregressive model. The proposed Skip AR model is designed to use the characteristics of multiple seasonalities in the datasets. Therefore, Skip AR can also capture multiple seasonalities in the linear part, thereby improving the time-series forecasting performance. Skip AR learns for each variable independently according to the following equation:

$$\hat{y}_t^{SkipAR} = \sum_{k=1}^{q} w_k^{SkipAR} x_{(m \times k)} + b^{SkipAR}. \tag{11}$$

Here, Skip AR fits linearly with q past time steps, which are the corresponding time steps to be predicted. Fig. 4 depicts the reason for selecting the name Skip AR. It skips the other time steps to model the Skip AR only with the corresponding time step. In (11), m is the parameter according to the granularity of the time-series. As the majority of the time-series includes complex seasonal patterns, it is difficult to parameterize m without explicit seasonal information. To mitigate this limitation, we use FT to identify the frequency. This frequency can determine the length of periodicity; thus, we can use FT when the seasonality is not shown in the time-series [30]. Therefore, we identify the

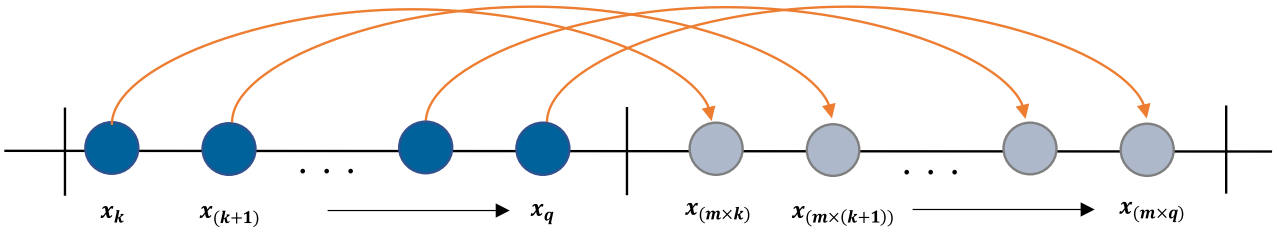


FIGURE 4. Details of the Skip AR.

seasonality in our datasets using FT and adapt it to m in Skip AR. In the experiment, it was set as the number of time-series data for *unit length*, as described in Subsection V.C.

D. MULTI-HEAD FULLY CONNECTED LAYER

In this model, the predicted values of the nonlinear model are derived using fully connected layers along with dual attention. To make a variable-specialized predictive model, the hidden layer is divided into several heads ($\tilde{h}_{t,i}$), and an independent fully connected neural network is established for each variable as follows:

$$\hat{y}_{t,i} = W^O \tilde{h}_{t,i} + b^O, \tag{12}$$

$$\tilde{h}_{t,i} = \sigma\left(W_i^I s_t + b_i^I\right), \tag{13}$$

where $i = \{1, 2, \ldots, n\}$ represents the variable dimension, and the hidden size of the head ($\tilde{h}_i$) is set to D_{head}. That is, the final output value $\hat{Y}_t^{NL} \in \mathbb{R}^n$ of the nonlinear model is obtained using the D_{head} sized fully connected neural network (13) and the output layer (12), which is a linear transformation. Furthermore, W and b are learnable parameters.

E. POSITIONAL ENCODING

Typical time-series data do not contain information regarding time steps. As time-series data are affected significantly by seasonality, an accurate prediction can be made if information concerning the time step is available when the model executes the prediction. However, it is challenging to learn the prediction time step by simply assigning the time information a numerical value such as an integer. In this study, we use PE to provide additional time information and to enable the model to recognize the seasonality more accurately.

The PE method was proposed in [26], which includes information regarding the relative and absolute position of natural language to word embeddings in machine translation. In this study, we propose the unique PE for each time point used for the time-series modeling, as follows:

$$PE_{(pos,2i)} = \sin\left(pos/10000^{2i/D_{model}}\right),$$

$$PE_{(pos,2i+1)} = \cos\left(pos/10000^{2i/D_{model}}\right), \tag{14}$$

$$\tilde{E} = E + PE, \quad \tilde{D} = D + PE. \tag{15}$$

The size of the PE vector is the same as that of D_{model}. i represents the index in the PE vector, and pos represents the absolute position of the corresponding time step t. As expressed in (14), PE of time step t is generated by intersecting the *sin* and *cos* functions as the dimensional index in the vector increases. Finally, PE is added to the output value E of the encoder cell and output value D of the decoder cell, as indicated in (15).

F. DENOISING TRAINING METHOD

The modeling method significantly affects the model performance in machine learning and deep neural networks. In this study, we propose a denoising training method that can capitalize on all of the advantages of the teacher forcing (TF) and without teacher forcing (w/o TF) methods using Gaussian noise.

Models that are based on encoder-decoder structures, such as Seq2Seq, primarily implement the TF method [31], [32]. The TF training method considers the step $t-1$ ground truth as the step t decoder input value. Therefore, the decoder can predict accurately during training, thereby increasing the initial learning speed [33]. However, a discrepancy exists between the inference and learning stages because the TF method does not make predictions based on the output from step $t-1$. This leads to an exposure bias problem that degrades the model performance and stability [34].

In contrast, the w/o TF method uses the value that is predicted in step $t-1$ ($\hat{Y}_{t-1}$) as the input value to obtain the output in t steps even during training [35]. If $\hat{Y}_{t-1}$ is significantly incorrect, this may result in high computational complexity owing to a decrease in the initial learning speed. However, as no difference exists between the learning and inferencing in the w/o TF method, exposure bias does not occur, which means that the model is relatively stable. Furthermore, owing to the characteristics of this training method, the predicted values in step t can be trained and predicted appropriately, considering the incorrect predicted values of step $t-1$.

$$\tilde{Y}_{t-1} = Y_{t-1} + \gamma \times scale\,(Y) \times \varepsilon, \tag{16}$$

$$\varepsilon \sim N(0, 1). \tag{17}$$

The proposed denoising training method offers the advantages of using noise, providing the rapid learning of the TF method via Y_{t-1}, and leveraging w/o TF, which eliminates the exposure bias problem via $\tilde{Y}_{t-1}$. As expressed in (16), the noise is calculated by multiplying ε, γ, and the standard

deviation of Y, where ε denotes Gaussian noise and γ denotes a hyperparameter that determines the intensity of the noise reflection. Subsequently, $\tilde{Y}_{t-1}$, which is noise added, is set as the decoder input value. A higher γ indicates that the more noise is added to the ground truth. Consequently, the use of Y_{t-1} with added small noise ε eliminates the exposure deflection problem, and the performance and stability of the model are enhanced by training to correct the incorrect prediction of the previous time step. However, the noise that is added to the model is only used during training.

The MAE loss function is used during training, which represents the absolute error of (18). Outliers frequently occur in time-series forecasting and the absolute error offers the advantage of reacting insensitively to outliers [36]. We use the Adam [37] optimization during the training process.

$$L_1 = \frac{1}{|T|\,|n|} \sum_t \sum_n |Y - \hat{Y}|. \tag{18}$$

Algorithm 1 : Flow of StackDA

Input: X (input time-series, $X = \{X_1, X_2, \ldots, X_L\}$)
Output: $\hat{Y}_t$

/∗ Step 1. Encoder ∗/
$E \leftarrow Conv1d(X)$
$E \leftarrow E + PE$
/∗ Step2. Skip AR ∗/
$\hat{y}_t^{SkipAR} \leftarrow \sum_{k=1}^{q} w_k^{SkipAR} x_{(m\times k)} + b^{SkipAR}$ // make prediction using past observations at corresponding time step (m interval)
/∗ Step 3. Decoder ∗/
$\tilde{Y}_{t-1} \leftarrow Noise(Y_{t-1})$ // decoder inputs
$d_t \leftarrow GRU(d_{t-1}, \hat{y}_{t-1})$ // stack M times
$D = D + PE$
$c_t, a_{t,k} \leftarrow TempAttn(d_t, e_k)$ // dot product attention, stack M times
$h_t \leftarrow FuseLayer(c_t, d_t)$
$S \leftarrow CausalAttn\,(H, H, H, mask)$ // self-attention, stack M times
$\hat{Y}_t^{NL} \leftarrow MultiHead(S)$ // final output value of nonlinear model
$\hat{Y}_t = \hat{Y}_t^{NL} + \hat{Y}_t^{SkipAR}$

G. FINAL PREDICTION

Algorithm 1 describes the model flow in detail. As indicated in (19), our model adds two output values to obtain the predicted value $\hat{Y}_t$ at step t. $\hat{Y}_t^{NL}$ is the nonlinear output value of the convolutional encoder and decoder when stacked dual attention is used. Moreover, $\hat{Y}_t^{SkipAR}$ is the output value of the Skip AR model:

$$\hat{Y}_t = \hat{Y}_t^{NL} + \hat{Y}_t^{SkipAR}. \tag{19}$$

The loss is calculated using the final $\hat{Y}_t$ and real value Y_t. According to the deep learning process, $\hat{Y}_t^{NL}$ and $\hat{Y}_t^{SkipAR}$ are learned such that $\hat{Y}_t$ is similar to the real value. This aids the model in learning the degree of training of the linear element $\hat{Y}_t^{SkipAR}$ and nonlinear element $\hat{Y}_t^{NL}$ during backpropagation.

TABLE 1. Statistics of all datasets.

Dataset	# of variables	# of time steps	Granularity
Electricity	321	26,304	1 hour
Traffic	862	17,544	1 hour
Solar energy	137	52,560	10 minutes

V. EXPERIMENTAL WORK

A. DATASET

We use three public multivariate time-series benchmark datasets in our experiments, excluding exogenous variables. The number of time-series data generated within one day is defined as the *unit length m*. We compare our results with those presented in several existing studies; thus, we use all preprocessed datasets presented in [1]. Table 1 summarizes the statistics of the datasets.

- *Electricity*: The electricity consumption for 321 locations was recorded in kWh every 15 min from 2012 to 2014. The data were converted into the consumption per hour, with $m = 24$.
- *Traffic*: Traffic data were acquired from the hourly highway share (0-1) dataset over 48 months (2015–2016) provided by the California Department of Transportation, with $m = 24$.
- *Solar energy*: Solar energy data were acquired by recording the photovoltaic production from 137 plants in Alabama State every 10 min in 2006, with $m = 144$.

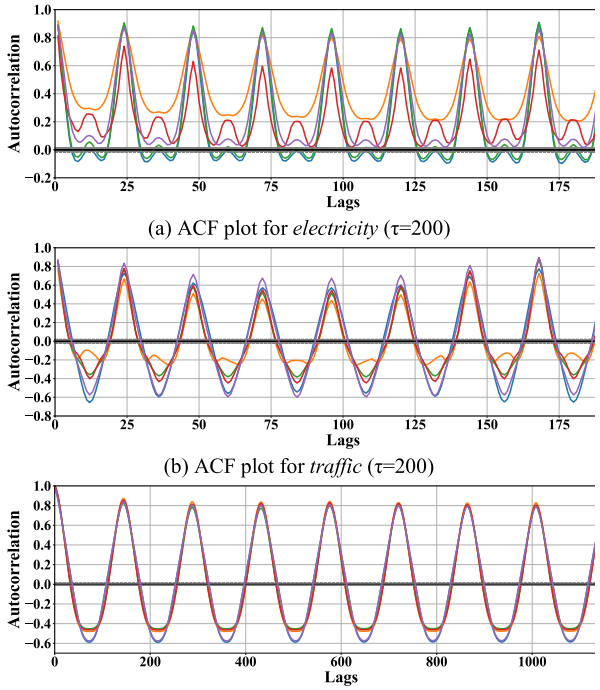
In this experiment, the last 31 days of a provided time-series dataset are used as the test set. In total, eighty percent of the datasets, excluding the test sets, are randomly used for training datasets, and twenty percent are used for verification.

A crucial aspect of time-series forecasting is to determine the seasonality. Thus, the autocorrelation function (ACF) is calculated for each variable in this dataset to determine the seasonalities of the time-series. The ACF, calculated using (20), is a function for determining the autocorrelation of time-series data, and it indicates the correlation between the current and lag times.

$$\rho\,(\tau) = \frac{E[(X_t - \mu)(X_{t+\tau} - \mu)]}{\sigma^2}, \tag{20}$$

where τ is the amount of delayed time, μ is the mean value and σ^2 is the variance.

Fig. 5 depicts the ACFs for five randomly selected variables that existed in each dataset. As the granularity of the solar energy is 10 min, we calculated a more extended time step to explore the long-term dependency (electricity and traffic: $\tau = 200$, solar energy: $\tau = 1{,}200$). Fig. 5 indicates that all datasets have repetitive patterns with a high autocorrelation. In addition to the largest autocorrelation value,



(a) ACF plot for *electricity* (τ=200)

(b) ACF plot for *traffic* (τ=200)

(c) ACF plot for *solar energy* (τ=1200)

FIGURE 5. ACF plots for all datasets.

time steps with extremely high correlation coefficients occur periodically.

B. EXPLANATION OF COMPARISON MODELS

We use four widely used machine learning and five deep learning comparison models to evaluate the performance of our proposed model. Univariate time-series models such as AR, GP, Prophet, TBATS, and N-BEATS perform independent training by generating models for each variable. Multivariate time-series models such as Seq2Seq-w/o-attn, Seq2Seq-w-attn, LSTNet-rec, and DeepAR combine variables to generate and train models. In the case of LSTNet-rec, multi-step forecasting is performed after training using one-step forecasting.

- AR is an autoregressive model that provides simple univariate linear regression and has traditionally been used in time-series forecasting [7].
- GP is the Gaussian process for univariate time-series forecasting using the distributed assumption [14].
- Prophet is a univariate model that performs time-series modeling in curve adaptation using FT [15].
- TBATS is a univariate framework that fits ARMA models based on FT [16].
- Seq2Seq-w/o-attn is a general encoder-decoder structure that uses a GRU.
- Seq2Seq-w-attn uses the above encoder-decoder structure and attention mechanism.
- LSTNet-rec is a multivariate time-series model that modifies LSTNet [1], which performs one-step forecasting for multi-step forecasting.
- DeepAR performs probabilistic forecasting based on an autoregressive RNN model [38].
- N-BEATS is an interpretable deep neural architecture based on residual links and deep stacked fully connected layers [39].

C. EXPERIMENTAL SETTINGS

In this section, we discuss the hyperparameter settings of our model and the comparative models. The input time-series length L that is used in all models limits the *unit length* $m \times 7$ to *max length*; however, it differs depending on the restrictions of the comparison model. The length T of the output time-series for multi-step forecasting is the same as *unit length* m.

1) SETTING OF MACHINE LEARNING MODELS

For AR, the maximum time steps, which indicates the past observation period, is set to the *max length* above. That is, X is used for fitting, which is the value of $m \times 7$. Unlike the other models, GP fit the data, which are the data for the seven days closest to the date for prediction.

Prophet uses *max length* as a window and adds seasonality information to make predictions. Daily, weekly, and monthly seasonalities are used. Each seasonality uses a Fourier series to obtain an approximation of the pattern. The Fourier series order of the daily periodicity is 20, and the weekly periodicity is 20. In the prediction step, the upper and lower values are set to 1 and 0, respectively.

TBATS also uses the input window length as *max length*, and the seasonality period is $(m, m \times 7)$. The prediction is performed by deciding whether to apply the Box-Cox transformation based on the AIC.

2) SETTING OF DEEP LEARNING MODELS

Our model uses an input window that is equal to *max length*. Moreover, we set an identical size for D_k and D_{model}, which represent the number of kernels in the encoder convolution and the hidden size of the decoder, respectively. We select the size in $\{2^7, 2^8, \ldots, 2^{10}\}$ using a grid search based on the validation dataset. The encoder kernel size is set to 5, and D^{head} in the independent fully connected neural network is set to 8. The dropout rate for regularization is set to 0.5, and the learning rate is 0.001. The stack size M is in $\{0, 2, 4, \ldots, 20\}$.

The window size in Seq2Seq is selected based on the validation dataset of $\{m, m \times 5, m \times 7\}$. The hidden size of the encoder and decoder GRU is set to 100. The same hyperparameter setting is used for Seq2Seq-w/o-attn and Seq2Seq-w-attn. Seq2Seq-w-attn is trained by adding the inner product attention mechanism to the hidden layers in the encoder and decoder.

LSTNet [1] follows the hyperparameters in the original paper as far as possible and modifies them to fit the training data. A grid search is performed based on the validation dataset in all of the provided grids. The grid of the input time-series length L is $\{m \times 2^3, m \times 2^4, \ldots, m \times 2^{10}\}$. The grid of the CNN and RNN layers is {50, 100, 200}.

TABLE 2. Evaluation results of all methods on the three datasets, where the best performance is highlighted in bold.

	Electricity			Traffic			Solar energy		
	RRSE	CORR	NRSME	RRSE	CORR	NRMSE	RRSE	CORR	NRMSE
AR	0.4554	0.8000	0.7986	0.7010	0.7485	0.6419	**0.5375**	0.8440	**0.9792**
GP	0.9433	0.3608	1.6542	0.9650	0.3020	0.8842	0.9899	0.3539	1.8037
Prophet	0.3277	0.8296	0.5779	0.6485	0.8046	0.5938	0.5886	0.8265	1.0725
TBATS	0.3724	0.7826	0.6530	0.9078	0.6504	0.8313	0.5968	0.8132	1.0874
S2S-w/o-attn	0.5023	0.7357	0.8808	0.7679	0.7424	0.7031	1.2273	0.2326	2.2361
S2S-w-attn	0.6647	0.7116	1.1655	0.7844	0.7260	0.7182	1.6416	-0.1176	2.9909
LSTNet-rec	1.0036	0.5390	1.7599	0.9297	0.5603	0.8513	1.1583	0.1176	2.1104
DeepAR	0.3393	0.8342	0.5950	0.6718	0.8065	0.6152	0.8614	0.7305	1.6895
NBEATS	0.3331	0.8387	0.5841	0.8197	0.6870	0.7505	0.6400	0.8021	1.2551
Ours	**0.2960**	**0.8388**	**0.5191**	**0.5750**	**0.8385**	**0.5265**	0.5387	**0.8479**	0.9815

The hidden layer of the RNN-skip layer is 10 for electricity and traffic, and {20, 50, 100} for solar energy, which is selected based on the validation dataset. The skip rate of the RNN-skip layer for solar energy is selected from $\{2, 2^2, \ldots, 2^6\}$, and 24 is used for electricity and traffic. LSTNet-rec is trained with one-step forecasting; however, multi-step forecasting was performed in the evaluation.

In addition, DeepAR [38] and N-BEATS [39] use the input window length as the *max length*. In DeepAR, the number of RNN layers is 3, and the number of RNN cells for each layer is 40. In N-BEATS, the number of stacks in the network is 2.

D. EVALUATION METRICS

In this study, we use the root relative squared error (RRSE), normalized root mean square error (NRMSE), and empirical correlation coefficient (CORR), which are three evaluation metrics that are used extensively for time-series forecasting. The metrics are formulated as follows, where Y and $\hat{Y}$ indicate the true and predicted values, respectively:

$$RRSE = \sqrt{\frac{\Sigma_t \left(y_t - \hat{y}_t\right)^2}{\sum_t (y_t - mean(Y))^2}} \tag{21}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_t (y_t - \hat{y}_t)^2} \tag{22}$$

$$NRMSE = \frac{RMSE}{mean(Y)} \tag{23}$$

$$CORR = \frac{1}{n}\sum \frac{\sum_t (y_t - mean(Y))(\hat{y}_t - mean(\hat{Y}))}{\sqrt{\sum_t (y_t - mean(Y))^2}\sqrt{(\hat{y}_t - mean(\hat{Y}))^2}}. \tag{24}$$

The range of the measured values of each time-series is large because the multivariate time-series consists of different scales. Therefore, we use RRSE (21), as the evaluation metric, which is scaled RRSE. RRSE is designed not to be affected by the data size. RMSE (22) is a widely used indicator in time-series; however, the data scale is not considered. In multivariate time-series forecasting, NRMSE (23) is commonly used because it considers the scale of the data. The NRMSE is calculated by dividing the RMSE by the mean of the real values. The empirical CORR (24) can be considered as the average of the correlation coefficients for the time steps of all variables. The prediction performance is higher for a lower RRSE and NRMSE, and a higher CORR.

VI. RESULTS AND DISCUSSION

In this section, we verify the superiority of the proposed model and the effectiveness of its components using various experimental results. Subsection VI.A demonstrates the performance of our model compared to nine machine learning and deep learning models. Subsequently, Subsection VI.B presents an in-depth analysis of each component of the model to confirm its efficiency.

A. MAIN RESULTS

A comparative experiment is conducted between our model and nine comparison models for time-series forecasting using the same dataset to verify the performance of the proposed method. For our model, the test procedure with multi-step one-month forecasting is performed by iterating the predicted observations occurring over one day.

Table 2 lists the experimental results for all models. The results reveal that our model outperforms the machine learning and deep learning models in terms of seven out of nine indicators. Therefore, the proposed method effectively models the time-series on datasets with several complex seasonalities. As illustrated in Fig. 5, all three datasets are characterized by clear and repetitive patterns. As our method firmly captures various patterns under the effect of Skip AR and stacked dual attention, it exhibit a significant performance on the electricity and traffic datasets. The variables in the solar energy dataset have patterns with similar seasonality, whereas the other two datasets have patterns in various forms. AR performs slightly better on the solar energy dataset owing to capturing the linearity, whereas our model performs the second best on this dataset.

Regarding decoding, our model exhibits superior performance to the existing deep learning methods. In particular,

TABLE 3. Comparison of LSTNet, LSTNet-rec, and our model. Note that LSTNet follows one-step forecasting, whereas StackDA and LSTNet-rec conduct multi-step forecasting.

	Electricity			Traffic			Solar-energy		
	RRSE	CORR	NRMSE	RRSE	CORR	NRMSE	RRSE	CORR	NRMSE
LSTNet	0.0917	0.9155	0.1381	0.4995	0.8520	0.4453	0.2010	0.9819	0.3325
LSTNet-rec	1.0036	0.5390	1.7599	0.9297	0.5603	0.8513	1.1583	0.1176	2.1104
Ours	0.2960	0.8388	0.5191	0.5750	0.8385	0.5265	0.5387	0.8479	0.9815

TABLE 4. Ablation study on each component of StackDA, where the best performance is highlighted in bold.

	Electricity			Traffic			Solar-energy		
	RRSE	CORR	NRMSE	RRSE	CORR	NRMSE	RRSE	CORR	NRMSE
Ours	**0.2960**	**0.8388**	**0.5191**	**0.5750**	**0.8385**	**0.5265**	**0.5387**	**0.8479**	**0.9815**
- PE	0.3288	0.8161	0.5767	0.6426	0.8007	0.5885	0.6380	0.8021	1.1625
- TA	0.3222	0.8260	0.5650	0.6204	0.8138	0.5681	0.6189	0.7974	1.1277
- CA	0.3295	0.8371	0.5779	0.6115	0.8190	0.5599	0.6719	0.7848	1.2242
- MultiHead	0.3198	0.8042	0.5608	0.5929	0.8277	0.5429	0.5835	0.8379	1.0631
- SkipAR	0.5081	0.7606	0.8911	0.9935	0.4278	0.9097	1.1639	0.0108	2.1206

Table 2 indicates that all of the deep learning methods except for our model perform less effectively than the machine learning methods. Furthermore, we compared the performance of the LSTNet and our model, as outlined in Subsection VI.B.

B. COMPARISON BETWEEN MULTI-STEP AND ONE-STEP FORECASTING

Table 3 presents the evaluation metrics based on the forecasting method of the LSTNet model. LSTNet performs one-step forecasting that predicts only the next time step, whereas LSTNet-rec performs multi-step forecasting that predicts $\hat{Y}_t$ using the predicted value $\hat{Y}_{t-1}$ again.

The results in Table 3 demonstrate that the one-step forecasting is significantly better than the multi-step forecasting. Furthermore, the one-step forecasting of LSTNet outperforms the multi-step forecasting of our model. However, the multi-step forecasting performance of LSTNet-rec is inferior to the one-step prediction of LSTNet because the decoding stage of LSTNet-rec is unstable owing to the continuously predicted procedure. That is, both the LSTNet and LSTNet-rec models are modeling the time-series based on feature extraction in deep learning techniques; however, LSTNet-rec can be interpreted as unstable in decoding because the variance for the incorrect predicted values in the previous time step is accumulated. Therefore, multi-step forecasting using deep learning is more complicated than one-step forecasting. Moreover, even in the Seq2Seq model, which is an encoder-decoder-based method, the difficulty of multi-step decoding can be identified, even when the feature vector can be adequately extracted from the encoder.

However, the proposed method overcome the limitations of existing deep learning models and constructs a denoising training method that stabilizes the multi-step forecasting. Moreover, our proposed stacked dual attention makes the learning highly stable in the decoding step, and thus, it exhibits significant performance in multi-step forecasting. Therefore, whereas the existing deep learning models exhibit a considerable degradation in multi-step forecasting, our model performs excellently by overcoming the multi-step forecasting limitations.

C. COMPARISON EXPERIMENT

1) ABLATION STUDY

We conduct comparative experiments by alternately removing the key components of our model to verify the performance of PE, TA, masked CA, Multi Head, and Skip AR.

As indicated in Table 4, when Skip AR is removed from this model, all indicators underwent the highest performance degradation. As the scale of the input time-series is corrected owing to the autoregressive element, which is the effect of the linear autoregressive model, the error is the highest when the scale is not corrected.

Furthermore, TA and CA exhibit a remarkable performance degradation when they are removed from the model. Therefore, TA and CA are not limited by the time step in the multi-step decoding; they solve the long-term dependency problem with the GRU decoder cell, and the complementary information flow leads to high performance.

Moreover, we confirm that PE, which adds a unique time location encoding for each time step in the time-series data, effectively solves the multiple seasonalities problem, and Multi Head, which is a fully connected neural network with multiple heads for variable-wise forecasting, positively affects the time-series forecasting.

2) COMPARISON BETWEEN SKIP AR AND TYPICAL AR

As the neural network model has nonlinear characteristics, the scale of the output value does not accurately reflect the

TABLE 5. Comparison of general AR and proposed Skip AR, where the best performance is highlighted in bold.

	Electricity			Traffic			Solar-energy		
	RRSE	CORR	NRMSE	RRSE	CORR	NRMSE	RRSE	CORR	NRMSE
AR	0.4145	0.8272	0.7268	0.7007	0.7851	0.6417	0.5840	0.8293	1.0640
SkipAR	**0.2960**	**0.8388**	**0.5191**	**0.5750**	**0.8385**	**0.5265**	**0.5387**	**0.8479**	**0.9815**

TABLE 6. Effect of the proposed denoising training method, where the best performance is highlighted in bold.

	Electricity			Traffic			Solar-energy		
	RRSE	CORR	NRMSE	RRSE	CORR	NRMSE	RRSE	CORR	NRMSE
$\gamma = 0.00$ (TF)	0.3169	0.8288	0.5557	0.5912	0.8302	0.5414	0.6030	0.8363	1.0986
$\gamma = 0.03$	0.3032	0.8278	0.5316	0.5848	0.8341	0.5355	**0.5387**	**0.8479**	**0.9815**
$\gamma = 0.05$	0.2990	0.8286	0.5243	0.5793	0.8366	0.5304	0.5388	**0.8479**	0.9816
$\gamma = 0.07$	**0.2960**	**0.8388**	**0.5191**	**0.5750**	**0.8385**	**0.5265**	0.5389	0.8478	0.9818
$\gamma = 0.10$	0.3040	0.8378	0.5330	0.5873	0.8323	0.5378	0.5998	0.8292	1.0928

input time-series scale, and it is difficult to capture the linear characteristics. To compensate for this limitation, existing studies have incorporated autoregression as a linear component. Similarly, in our model, autoregression components are added to capture the linear characteristics. Thus, we propose Skip AR, which only contains information at the same time step, rather than a typical AR. Skip AR fits linearly using historical time steps that are identical to the time step to be predicted. Table 5 compares the typical AR and our proposed Skip AR models. Skip AR captures multiple seasonalities satisfactorily with linear fitting using the same historical time steps. Consequently, Skip AR performs significantly better than typical AR.

Furthermore, we use the FT method to identify the appropriate m in Skip AR, which is the seasonality in the time-series. FT converts the time-series from the time domain to the frequency domain to determine the period, and the frequency is divided by 1 to obtain the period [30]. As illustrated in Fig. 6, the granularity of electricity and traffic is 1 h, whereas that of solar energy is 10 min. Hence, the top seasonality of all datasets was one day, and we adopted one day for m. Table 5 demonstrates that Skip AR performs significantly better than typical AR. In particular, it is superior for electricity and traffic datasets. Fig. 5 illustrates that electricity and traffic have diverse multiple seasonalities compared to solar energy in the ACF plot. That is, Skip AR captures the seasonality more precisely and delicately than typical AR owing to its skip characteristics.

D. EFFECT OF NOISE INTENSITY (γ)

We propose a denoising training method using Gaussian noise, considering both the TF and w/o TF training methods. The Gaussian noise ε is adjusted by the noise intensity (γ), as expressed in (16). A comparison is conducted considering γ to determine the efficiency of the denoising training method and optimal noise intensity.

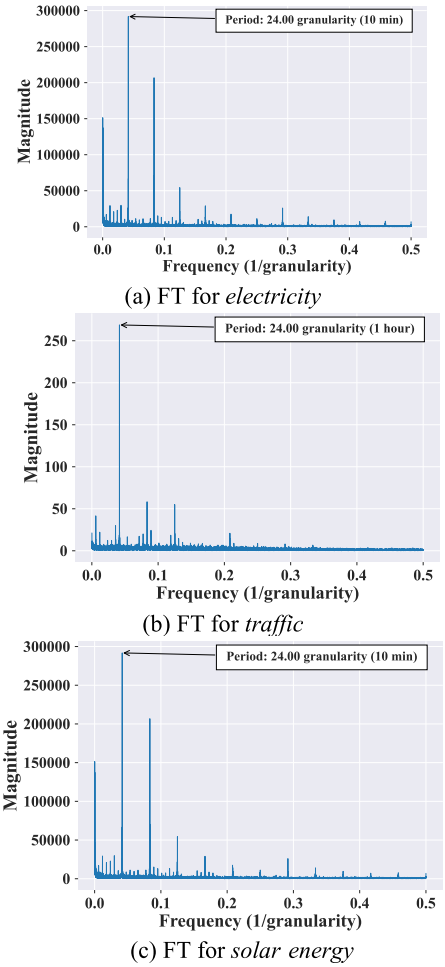


FIGURE 6. Fourier transform to determine seasonality for all datasets. Note that the term "period" means "seasonality."

Table 6 presents the results of the comparison to determine the optimum noise intensity ($\gamma = 0.00, 0.03, 0.05,$

TABLE 7. Effect of the proposed stack attentions, where the best performance is highlighted in bold.

	Electricity			Traffic			Solar energy		
	RRSE	CORR	NRMSE	RRSE	CORR	NRMSE	RRSE	CORR	NRMSE
Non-stacked	0.2992	0.8355	0.5247	0.5830	0.8346	0.5376	0.5388	**0.8479**	**0.9817**
Stacked	**0.2960**	**0.8388**	**0.5191**	**0.5750**	**0.8385**	**0.5265**	**0.5387**	**0.8479**	**0.9815**

0.07, 0.10). For $\gamma = 0.00$, the noise intensity is not provided, and it is equal to that of the TF training method. The optimum noise intensity differs based on the variability in the provided time-series data. We empirically confirm that $\gamma = 0.07, 0.07,$ and 0.03 are the most appropriate values for the electricity, traffic, and solar energy time-series datasets, respectively. The sensitivity to noise shows that solar energy is the strongest, and relatively sensitive than electricity and traffic. Moreover, the performance improves with increase in the noise; however, when the noise value is considerably large (0.10), the training quality degrades. Moreover, the exposure bias problem is mitigated when an appropriate noise proportional to the scale of the time-series is added. This makes the model perform a stable multi-step prediction, which is an advantage of the denoising training method.

E. THE EFFECT OF STACK ATTENTION

We propose a dual-attention structure with TA and CA. TA is the time dependency between the encoder and decoder, and CA is the time dependency in the decoder time steps. In addition, we attempted to determine multi-seasonality by stacking each attention for more detailed seasonality modeling. According to the results listed in Table 7, our model outperforms other comparison methods in five out of six indicators, compared to the standard dual attention mechanism. Consequently, the necessary information is well extracted from each time step when attention is stacked.

VII. CONCLUSION

Time-series forecasting predicts future time steps by deriving seasonality from past observations. It exhibits the limitations of multiple pattern problems and multi-step prediction difficulties. In this study, we proposed the novel Skip AR to solve the multiple seasonality problem. Owing to the characteristics of Skip AR, the performance exhibited on the datasets with diverse multiple seasonalities in the time-series is satisfactory. Furthermore, we introduced TA mechanisms between the encoder and decoder, and CA mechanisms within the decoder. We stacked the attention to capture the seasonality for solving the multiple seasonalities problem more precisely. Moreover, we proposed a model that is optimized for multi-step forecasting using a denoising training method by implementing both the TF and w/o TF methods. We also established variable-specific modeling using independent decoder modules and Multi Head, and used the PE method for time-series forecasting.

We conducted experiments using three widely used datasets for time-series forecasting. According to the experimental results, our model exhibited higher performance than that of the machine learning and deep learning methods, which are used extensively in time-series forecasting. Furthermore, the superiority of the proposed method was demonstrated using in-depth comparative experiments, such as ablation, attention stack, FT, and denoising degree experiments.

Our model effectively performed multi-step predictions for multivariate time-series with complex seasonalities using Skip AR, stacked dual attention, denoising training, variable-specific modeling, and PE. In the future, we can effectively model variables in high-dimensional multivariate time-series data with a relatively large numbers of variables. Furthermore, clustering variables with similar roles and advancing the modeling methods are interesting research topics to be considered.

REFERENCES

[1] 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.*, Jun. 2018, pp. 95–104.
[2] Z. Liu and M. Hauskrecht, "A regularized linear dynamical system framework for multivariate time series analysis," in *Proc. 29th AAAI Conf. Artif. Intell. (AAAI)*, vol. 15, 2015, pp. 1798–1804.
[3] J. Zheng, C. Xu, Z. Zhang, and X. Li, "Electric load forecasting in smart grids using long-short-term-memory based recurrent neural network," in *Proc. 51st Annu. Conf. Inf. Sci. Syst. (CISS)*, Mar. 2017, pp. 1–6.
[4] A. Tsantekidis, N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Forecasting stock prices from the limit order book using convolutional neural networks," in *Proc. IEEE 19th Conf. Bus. Inform. (CBI)*, Jul. 2017, pp. 7–12.
[5] Y. Wu, J. M. H. Lobato, and Z. Ghahramani, "Dynamic covariance models for multivariate financial time series," in *Proc. 30th Int. Conf. Mach. Learn.*, vol. 28, 2013, pp. 558–566.
[6] V. L. E. Guen and N. Thome, "Shape and time distortion loss for training deep time series forecasting models," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 32, 2019, pp. 4189–4201.
[7] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis: Forecasting and Control*, 5th ed. Hoboken, NJ, USA: Wiley, 2015.
[8] J. T. Connor, R. D. Martin, and L. E. Atlas, "Recurrent neural networks and robust time series prediction," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 240–254, Mar. 1994.
[9] A. Borovykh, S. Bohte, and C. W. Oosterlee, "Conditional time series forecasting with convolutional neural networks," 2017, *arXiv:1703.04691*. [Online]. Available: http://arxiv.org/abs/1703.04691
[10] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. 3rd Int. Conf. Learn. Represent.*, 2016, pp. 1–11.
[11] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, vol. 15, 2015, pp. 1412–1421.
[12] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 27, 2014, pp. 3104–3112.
[13] I. Melnyk and A. Banerjee, "Estimating structured vector autoregressive models," in *Proc. 33rd Int. Conf. Mach. Learn.*, vol. 48, pp. 830–839, 2016.

[14] S. Roberts, M. Osborne, M. Ebden, S. Reece, N. Gibson, and S. Aigrain, "Gaussian processes for time-series modelling," *Philos. Trans. Roy. Soc. A, Math., Phys. Eng. Sci.*, vol. 371, no. 1984, 2013, Art. no. 20110550.
[15] S. J. Taylor and B. Letham, "Forecasting at scale," *Amer. Statistician*, vol. 72, no. 1, pp. 37–45, 2018.
[16] A. M. De Livera, R. J. Hyndman, and R. D. Snyder, "Forecasting time series with complex seasonal patterns using exponential smoothing," *J. Amer. Statist. Assoc.*, vol. 106, no. 496, pp. 1513–1527, 2011.
[17] R. Yu, Y. Li, C. Shahabi, U. Demiryurek, and Y. Liu, "Deep learning: A generic approach for extreme condition traffic forecasting," in *Proc. SIAM Int. Conf. Data Mining (SIAM)*, 2017, pp. 777–785.
[18] Y. Zhu, H. Li, Y. Liao, B. Wang, Z. Guan, H. Liu, and D. Cai, "What to do next: Modeling user behaviors by time-LSTM," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, Aug. 2017, pp. 3602–3608.
[19] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[20] Y.-Y. Chang, F.-Y. Sun, Y.-H. Wu, and S.-D. Lin, "A memory-network based solution for multivariate time-series forecasting," 2018, *arXiv:1809.02105*. [Online]. Available: http://arxiv.org/abs/1809.02105
[21] X. Ding, Y. Zhang, T. Liu, and J. Duan, "Deep learning for event-driven stock prediction," in *Proc. 24th Int. Conf. Artif. Intell. (IJCAI)*, vol. 15, 2015, pp. 2327–2333.
[22] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," in *Proc. 34th Int. Conf. Mach. Learn. (ICML)*, vol. 70, 2017, pp. 1243–1252.
[23] M. Akram and C. El, "Sequence to sequence weather forecasting with long short-term memory recurrent neural networks," *Int. J. Comput. Appl.*, vol. 143, no. 11, pp. 7–11, Jun. 2016.
[24] Z. Mariet and V. Kuznetsov, "Foundations of sequence-to-sequence modeling for time series," in *Proc. 22nd Int. Conf. Artif. Intell. Statist.*, vol. 89, Apr. 2019, pp. 408–417.
[25] 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," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, vol. 14, Oct. 2014, pp. 1724–1734.
[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. U. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 30, 2017, pp. 5998–6008.
[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. (CIKM)*, vol. 19, 2019, pp. 2129–2132.
[28] G. P. Zhang, "Time series forecasting using a hybrid ARIMA and neural network model," *Neurocomputing*, vol. 50, pp. 159–175, Jan. 2003.
[29] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 28, 2015, pp. 2377–2385.
[30] H. Musbah and M. El-Hawary, "SARIMA model forecasting of short-term electrical load data augmented by fast Fourier transform seasonality detection," in *Proc. IEEE Can. Conf. Electr. Comput. Eng. (CCECE)*, May 2019, pp. 1–4.
[31] Y. Keneshloo, T. Shi, N. Ramakrishnan, and C. K. Reddy, "Deep reinforcement learning for sequence-to-sequence models," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 7, pp. 2469–2489, Jul. 2019.
[32] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Comput.*, vol. 1, no. 2, pp. 270–280, 1989.
[33] T. Mihaylova and A. F. T. Martins, "Scheduled sampling for transformers," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics: Student Res. Workshop*, 2019, pp. 351–356.
[34] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba, "Sequence level training with recurrent neural networks," in *Proc. 4th Int. Conf. Learn. Represent.*, 2016, pp. 1–12.
[35] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 28, 2015, pp. 1171–1179.
[36] C. J. Willmott and K. Matsuura, "Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance," *Climate Res.*, vol. 30, no. 1, pp. 79–82, Dec. 2005.
[37] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent.*, 2015, pp. 1–11.
[38] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, "DeepAR: Probabilistic forecasting with autoregressive recurrent networks," *Int. J. Forecasting*, vol. 36, no. 3, pp. 1181–1191, Jul. 2020.
[39] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, "N-BEATS: Neural basis expansion analysis for interpretable time series forecasting," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–12. [Online]. Available: https://openreview.net/forum?id=r1ecqn4YwB

JUNGSOO HONG received the B.S. degree in computer science and engineering from Ewha University, Seoul, South Korea, in 2019, and the M.S. degree in computer science from Yonsei University, Seoul, in 2021. Her current research interests include machine learning, time-series modeling, and neural networks.

JINUK PARK received the B.S. degree in statistics from the University of Seoul, Seoul, South Korea, in 2016. He is currently pursuing the Ph.D. degree in computer science with Yonsei University, Seoul. His current research interests include machine learning, time-series modeling, and neural networks.

SANGHYUN PARK (Member, IEEE) received the B.S. and M.S. degrees in computer engineering from Seoul National University, in 1989 and 1991, respectively, and the Ph.D. degree from the Department of Computer Science, University of California at Los Angeles (UCLA), in 2001. He is currently a Professor with the Department of Computer Science, Yonsei University, Seoul, South Korea. His current research interests include databases, data mining, bioinformatics, and flash memory.