

# Long-term Time Series Forecasting based on Decomposition and Neural Ordinary Differential Equations

Seonkyu Lim<sup>\*†§</sup>, Jaehyeon Park<sup>\*§</sup>, Seojin Kim<sup>\*§</sup>, Hyowon Wi<sup>\*</sup>, Haksoo Lim<sup>\*</sup>,  
Jinsung Jeon<sup>\*</sup>, Jeongwhan Choi<sup>\*</sup> and Noseong Park<sup>\*</sup>  
Yonsei University<sup>\*</sup>, Seoul, South Korea

Korea Financial Telecommunications & Clearings Institute<sup>†</sup>, Seoul, South Korea

sklim@kftc.or.kr<sup>†</sup>, {jaehyun9907, bwnebs1, wihyowon, limhaksoo96, jjsjjs0902, jeongwhan.choi, noseong}@yonsei.ac.kr<sup>\*</sup>

**Abstract**—Long-term time series forecasting (LTSF) is a challenging task that has been investigated in various domains such as finance investment, health care, traffic, and weather forecasting. In recent years, Linear-based LTSF models showed better performance, pointing out the problem of Transformer-based approaches causing temporal information loss. However, Linear-based approach has also limitations that the model is too simple to comprehensively exploit the characteristics of the dataset. To solve these limitations, we propose LTSF-DNODE, which applies a model based on linear ordinary differential equations (ODEs) and a time series decomposition method according to data statistical characteristics. We show that LTSF-DNODE outperforms the baselines on various real-world datasets. In addition, for each dataset, we explore the impacts of regularization in the neural ordinary differential equation (NODE) framework.

**Index Terms**—long-term time series forecasting, time series decomposition, instance normalization, neural ordinary differential equations

## I. INTRODUCTION

Time series data is continuously generated from various real-world applications. To utilize this data, numerous approaches have been developed in the fields such as forecasting [1]–[9], classification [2], [5], [6], [10]–[13], and generation [14]–[17]. Among them, time series forecasting is one of the most important research topics in deep learning. For time series forecasting, recurrent neural network (RNN)-based models were used, such as LSTM [18] and GRU [19]. These models performed well in time series forecasting but suffered from error accumulation due to their iterative multi-step forecasting approach, especially when dealing with long-term time series forecasting (LTSF).

To overcome this challenge, there have been various attempts in the past few years. Among them, the Transformer-based approaches [2], which enable direct multi-step forecasting, show significant performance improvement. Transformer [20] has demonstrated remarkable performance in various natural language processing tasks, and its ability to

effectively capture long-range dependencies and interactions in sequential data makes it suitable for application in LTSF.

Despite the impressive achievements of Transformer-based models in LTSF, they have struggled with temporal information loss caused by the self-attention mechanism as a result of permutation invariant and anti-order properties [21]. Furthermore, it was demonstrated that the error upper bound of Transformer-based models, which is one of the non-linear deep learning approaches, is higher than linear regression [22].

Recently, simple Linear-based methods [21], [22], as non-Transformer-based models, show better performance compared to complex Transformer-based models. These approaches are novel attempts in LTSF where Transformer architecture has recently taken the lead. However, they have limits to comprehensively exploit the intricate characteristics of the time series dataset as the models are too simple.

Inspired by these insights, we aim to design a model that takes into account the characteristics of each dataset by introducing some sophistication to the model architecture. Our proposed model, LTSF-DNODE, adeptly harnesses temporal information by employing time series decomposition and the neural ordinary differential equation (NODE) framework. The NODE framework transforms a single linear layer into time-derivative modeling. It uses a structure composed of the same single linear layer, but can better capture the complex dynamics of time series data, providing various advantages within time series processing.

Our contributions can be summarized as follows:

- We analyze the temporal information of each real-world time series datasets with exploratory data analysis. This investigation helps us detect the presence of seasonality, guiding us to perform decomposition appropriately.
- Based on the NODE framework, we demonstrate the following benefits of time-derivative modeling: i) It is suitable for time series tasks by interpreting discrete linear layers as continuous linear layers, and ii) more advanced regularization is available.
- Through various empirical experiments, we demonstrate that the NODE framework and decomposition according to data characteristics are effective in LTSF.

<sup>§</sup>These authors contributed equally to this research.

## II. PRELIMINARIES

In this section, we review the decomposition method and neural ordinary differential equations. Following that, we conduct empirical explorations to investigate the effectiveness of these techniques on LTSF.

### A. Problem Formulation

The objective of LTSF is to forecast from an input sequence of historical time series data to a corresponding future sequence. Given the input historical data  $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_L]^\top \in \mathbb{R}^{L \times F}$ , LTSF models forecast  $\mathbf{Y} = [\mathbf{y}_1, \dots, \mathbf{y}_H]^\top \in \mathbb{R}^{H \times F}$ , where  $L$  is the look-back window size,  $H$  is the forecasting horizon and  $F$  is the feature dimension. The LTSF problem deals with cases where  $H$  is longer than 1, and  $F$  is not restricted to univariate cases.

### B. Time Series Decomposition

Data preprocessing is used to enhance data quality as an input to the LTSF method, enabling it to deliver better outcomes. According to [23], suitable preprocessing methods for non-stationary time series increased forecasting accuracy by more than 10% on over 95% of the temporal data.

Time series decomposition is a pivotal technique in time series preprocessing [24]. The decomposition methods (e.g., STL [25] and SEAT [26]) typically decompose the time series into three components: trend, seasonality, and residual component. The decomposition can be represented as follows:

$$\mathbf{X} = \mathbf{T} + \mathbf{S} + \mathbf{R}, \quad (1)$$

where  $\mathbf{T}, \mathbf{S}, \mathbf{R} \in \mathbb{R}^{L \times F}$  respectively represent the trend, seasonality, and residual component. The trend is a general systematic linear or nonlinear component that changes over time and does not repeat within the given timeframe, usually identified by the two-sided simple moving average [27]. Seasonality means a pattern with a particular cycle in a time series. To extract seasonality, the classic additive decomposition method averages the detrended series  $(\mathbf{X} - \mathbf{T})$  across a predetermined period. Residual is the remaining value after removing the trend and seasonality from a series.

We use this method to provide an overall understanding of time series datasets with exploratory data analysis.

### C. Neural Ordinary Differential Equations

Neural ordinary differential equations (NODEs) [28] can handle time series data in a continuous manner, using the differential equation as follows:

$$\mathbf{z}(T) = \mathbf{z}(0) + \int_0^T f(\mathbf{z}(t), t; \theta_f) dt, \quad (2)$$

where  $f(\mathbf{z}(t), t; \theta_f)$ , called an ODE function, is a neural network to approximate the derivative of  $\mathbf{z}(t)$  with respect to  $t$  (denoted as  $\frac{d\mathbf{z}(t)}{dt}$ ).

To solve the integral problem, the NODEs use the ODE solver. The ODE solvers divide the integral time domain  $[0, T]$  in Eq. (2) into small steps and convert the integral into many

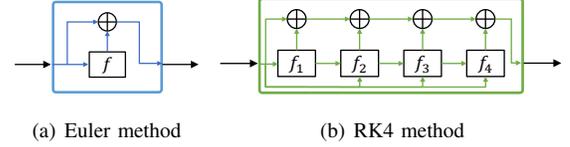


Fig. 1. The (a) explicit Euler method and (b) RK4 method in a step. We note that the Euler method creates the residual connection and RK4 makes the dense connection given the ODE function  $f$  parameterized by  $\theta_f$ .

steps of additions. For example, one step of the explicit Euler method, a typical ODE solver, is as follows:

$$\mathbf{z}(t + s) = \mathbf{z}(t) + s \cdot f(\mathbf{z}(t), t; \theta_f), \quad (3)$$

where  $s \in (0, 1]$  is step size of the Euler method. A more sophisticated method, such as the 4-th order Runge-Kutta (RK4) method, is as follows:

$$\mathbf{z}(t + s) = \mathbf{z}(t) + \frac{s}{6} (f_1 + 2f_2 + 2f_3 + f_4), \quad (4)$$

where  $f_1 = f(\mathbf{z}(t), t; \theta_f)$ ,  $f_2 = f(\mathbf{z}(t) + \frac{s}{2}f_1, t + \frac{s}{2}; \theta_f)$ ,  $f_3 = f(\mathbf{z}(t) + \frac{s}{2}f_2, t + \frac{s}{2}; \theta_f)$ , and  $f_4 = f(\mathbf{z}(t) + sf_3, t + s; \theta_f)$ .

Fig. 1 denotes the Euler and RK4 methods. These are some of the explicit methods that have a fixed step size. The RK4 method requires four times as much work as the Euler method in a single step. When the step size is 1, the Euler method is equivalent to the residual connection. Similarly, when  $f$  represents a neural network layer, the RK4 method is analogous to the dense connection. On the other hand, one of the most advanced methods, Dormand-Prince (DOPRI) [29], uses an adaptive step size. Recently, the Memory-efficient ALF Integrator [30] guaranteeing constant memory cost shows good performance. However, when NODE learns a complex dataset, the step size of the ODE solver often becomes extremely small. Consequently, this results in dynamics equivalent to a substantial number of layers, thereby increasing the training time significantly. To address this issue, Jacobian and kinetic regularizations [31] are introduced, simplifying the dynamics, increasing the step size, and reducing the training time.

The NODEs demonstrated superior performance in various tasks, including time series and others, by employing continuous modeling [32]–[34]. Specifically, the neural rough differential equations applied to the NODEs using rough-path theory showed good performance in the long-term time series task [35], [36].

In our model, we use a single linear layer as  $f$  with various ODE solvers and regularizers to model times series dynamics effectively.

### D. Empirical Explorations on Decomposition and NODEs

We conduct ablation studies to explore the efficacy of decomposition and NODE in LTSF. We implement simple variants of a single linear layer model by applying the decomposition method and NODE framework:

- “Linear” is a single linear layer identical to the one introduced in [21]. This model predicts future values based on

TABLE I  
MSE OF FORECASTING RESULTS BASED ON DECOMPOSITION AND NODE.  
WE SET THE LOOK-BACK WINDOW SIZE AS 336.

Datasets	Forecasting horizon	Linear	Linear with T/R	Linear with T/S/R	Linear with NODE
ETTh1	96	0.375	0.375	0.378	<b>0.371</b>
	192	0.418	0.405	<b>0.404</b>	0.406
	336	0.479	0.439	<b>0.437</b>	<b>0.437</b>
	720	0.624	<b>0.472</b>	<b>0.472</b>	0.475
ETTh2	96	0.288	0.289	0.288	<b>0.280</b>
	192	0.377	0.383	<b>0.359</b>	0.364
	336	0.452	0.448	<b>0.422</b>	0.438
	720	0.698	0.605	0.570	<b>0.568</b>

past values via a weighted summation. The single linear layer is mathematically expressed as  $\hat{\mathbf{Y}} = \mathbf{W}\mathbf{X}$ , where  $\mathbf{W} \in \mathbb{R}^{P \times L}$  is a weight matrix.

- “Linear with T/R” decomposes time series into trend and residual components and then individually learns them using single linear layers, as proposed in [21].
- “Linear with T/S/R” decomposes time series into trend, seasonality, and residual components, and then individually learns them through single linear layers.
- “Linear with NODE” also uses a single linear layer with the same structure as “Linear” to build the ODE function in the NODE framework. It employs the Euler method as the ODE solver.

The mathematical expression of “Linear” can also be depicted using the time variable  $t$  as follows:

$$\hat{\mathbf{x}}(t_1) = \mathbf{W}\mathbf{x}(t_0), \quad (5)$$

where  $\hat{\mathbf{x}}(t_1)$  is the future sequence  $\hat{\mathbf{Y}}$ ,  $\mathbf{x}(t_0)$  is the historical sequence  $\mathbf{X}$ .  $\mathbf{W}$  is a single linear layer matrix.

“Linear with NODE” formulation based on Eq. (5) is as follows:

$$\frac{d\mathbf{x}(t)}{dt} = \frac{1}{t_1 - t_0}(\log \mathbf{W})\mathbf{x}(t), \quad (6)$$

$$\mathbf{x}(t_1) = \mathbf{x}(t_0) + \int_{t_0}^{t_1} \frac{1}{t_1 - t_0}(\log \mathbf{W})\mathbf{x}(t)dt. \quad (7)$$

The parameters are trained using the adjoint sensitivity method of the NODE framework.

Table I shows the LTSF results for each variant model. Based on these observations, we infer the following findings:

- 1) Applying time series decomposition methods enhances the performance of LTSF. This assertion is corroborated by the empirical observation that “Linear with T/R” shows better performance compared to the “Linear”.
- 2) In the ETTh2 dataset, “Linear with T/S/R” demonstrates a discernible advantage over “Linear with T/R”. This suggests that the merits of a finer-grained decomposition, particularly the extraction of seasonality, might vary depending on the characteristics of the datasets.

- 3) The better performance of “Linear with NODE” as compared to “Linear” demonstrates the benefits of incorporating the NODE framework in the domain of LTSF.

From these observations, we can infer that time-derivate modeling based on the NODE framework and more refined time series decomposition have a positive impact on LTSF.

### III. PROPOSED METHOD

We explain the detailed information about our proposed model, LTSF-DNODE, in this section. We first describe an overview of how our model works, followed by detailed components and how they contribute to time series forecasting.

#### A. Overall Workflow

Fig. 2 illustrates the forecasting procedure of LTSF-DNODE. It consists of three main blocks: the decomposition block (DCMP), the normalizing and denormalizing blocks (NORM & DENORM), and the NODE block (NODE). These blocks are sequentially applied to make predictions as follows:

- 1) Find data characteristics with exploratory data analysis.
- 2) An observed series  $\mathbf{X}$  is given as input. The DCMP block decomposes  $\mathbf{X}$  depending on data characteristics.
- 3) For datasets with distribution discrepancy problems, we apply the NORM block to address them.
- 4) Then, the NODE block forecasts the future patterns of each decomposed component.
- 5) In the case of the dataset normalized in (2), denormalization is performed in the DENORM block.
- 6) Finally, the forecasting series  $\hat{\mathbf{Y}}$  is reconstructed by addition of the future decomposed components.

#### B. Exploratory Data Analysis

1) *Properties*: To obtain insights from the datasets, we analyze their characteristics. The results of this analysis are presented in Table IV. The fundamental structure and methodology of this analysis are derived from [37]. The detailed methods used to acquire statistical properties are as follows:

- “Forecastability” [38] refers to a measure calculated by subtracting the entropy of the Fourier decomposition of the time series from one.
- “Trend” is the slope of the linear regression applied to the time series, adjusted according to its magnitude.
- “Seasonality” is the ratio of noticeable patterns quantified by the ACF test [39].
- “Stationarity” is measured by the ADF test on the residual component after removing trend and seasonality components from the time series.

2) *Decomposition*: DCMP block requires a pre-defined kernel size and seasonality period during the decomposition process. In order to achieve a suitable decomposition, we need to find the optimal parameters depending on the dataset. We consider various parameter combinations as candidates and conduct two tests, the auto-correlation function (ACF) test [39] and the augmented Dickey-Fuller (ADF) test, to find the optimal values among them. The ACF test examines if

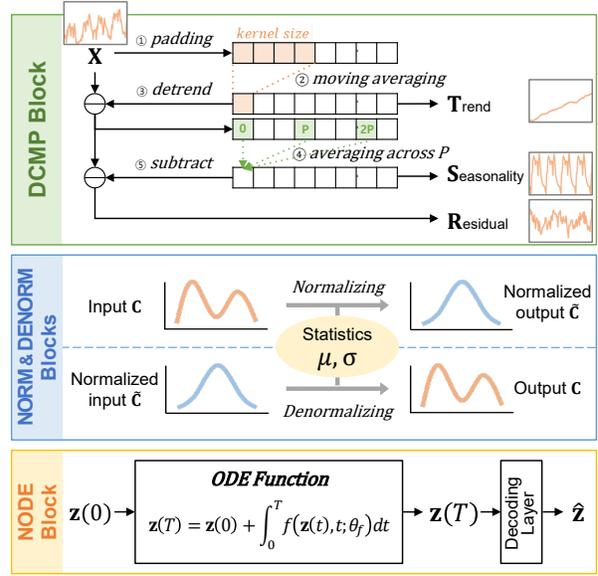
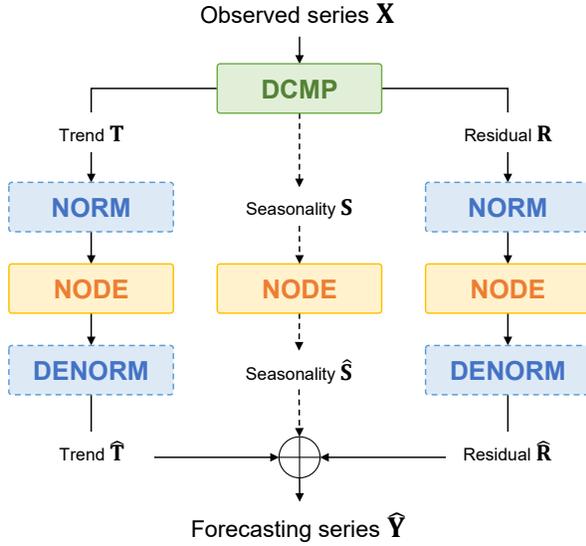


Fig. 2. The overall workflow of LTSF-DNODE. LTSF-DNODE consists of three blocks: the decomposition block (DCMP), the normalizing and denormalizing blocks (NORM & DENORM), and the neural ODE block (NODE). DCMP block decomposes the observed time series into trend, seasonality, and residual components. NORM block normalizes the observed series based on its mean and variance. NODE block forecasts of the decomposed component based on each ODE. The processes indicated by dotted lines are optionally applied, considering the characteristics of the dataset.

a sequence follows a repeating seasonal pattern with a pre-defined cycle, while the ADF test is used to check whether the sequence is stationary or not.

We first divide the time series into non-overlapping window-sized sequences and then decompose them. The ACF test is conducted for analyzing the seasonality components, while the ADF test is utilized to assess the residual components. Following the tests, we can quantify both seasonality and stationarity by calculating the proportion of sequences meeting the criteria relative to the total number of sequences.

This procedure is applied to all the datasets. An overview of the experimental parameters selected for each dataset and a summary of the results derived from the exploratory analysis are presented in Table IV.

3) *Normalization*: We investigate the distribution of trend components in both training and testing data. If there exists a noticeable discrepancy between these distributions, we employ the NORM block to perform instance normalization. The same procedure is also applied to the residual components.

The instance normalization method aids in mitigating distribution disparities and aligning the overall trend. Additional findings can be found in Section IV.

### C. DCMP Block

The DCMP block decomposes the time series into three, trend  $\mathbf{T}$ , seasonality  $\mathbf{S}$ , and residual  $\mathbf{R}$  components, or two, without seasonality depending on the characteristics of the observed series  $\mathbf{X}$ . In the DCMP block, in order to extract the trend from the  $\mathbf{X}$ , we use the moving average method. To align the length of the derived trend with  $\mathbf{X}$ , we first apply  $padding(\cdot)$ , which involves pre-padding with the first value

and post-padding with the last value of the input. After that, we perform an average pooling operation to extract the trend as follows:

$$\mathbf{T} = AvgPool(padding(\mathbf{X})). \quad (8)$$

If the time series has a low significance of seasonality, extracting seasonality is omitted. In this case, the detrended  $(\mathbf{X} - \mathbf{T})$  is regarded as the residual component, and the decomposition process is completed. However, if the series exhibits significant seasonality, we extract the seasonality from  $(\mathbf{X} - \mathbf{T})$ . We first obtain seasonal fragments by averaging  $(\mathbf{X} - \mathbf{T})$  across the pre-defined period  $P$ . In the fragments, each element is calculated as follows:

$$\mathbf{S}_i = \frac{1}{m} \sum_{k=0}^{m-1} (\mathbf{X}_{i+kP} - \mathbf{T}_{i+kP}), \quad (9)$$

where  $m$  is the smallest integer satisfying  $L < i + mP$  for  $0 \leq i < P$  and sequence length  $L$ ,  $\mathbf{S}_i$ ,  $\mathbf{T}_{i+kP}$  and  $\mathbf{X}_{i+kP}$  denote the  $i$ -th element of the seasonal fragments and the  $(i + kP)$ -th element of  $\mathbf{T}$  and  $\mathbf{X}$ .

We produce seasonal fragments with a period  $P$  using Eq. (9). Then we construct the overall seasonality component  $\mathbf{S}$  of length  $L$  by tiling the obtained seasonal fragments. The residual component  $\mathbf{R}$  is the remaining part of the observed series  $\mathbf{X}$ , obtained by subtracting the trend (and seasonality, depending on the dataset).

We effectively capture the temporal information from the time series using the decomposition method, which allows us to enhance the forecasting capabilities of our model. The analysis about the significance of seasonality for the datasets we use is in Section IV-C.

#### D. NODE Block

For the decomposition of the time series into trend, seasonality and residual, the NODEs can be written as follows and solved by the ODE solvers:

$$\begin{aligned}\mathbf{T}(T) &= \mathbf{T}(0) + \int_0^T f_{\mathbf{T}}(\mathbf{T}(t); \boldsymbol{\theta}_{\mathbf{T}}) dt, \\ \mathbf{S}(T) &= \mathbf{S}(0) + \int_0^T f_{\mathbf{S}}(\mathbf{S}(t); \boldsymbol{\theta}_{\mathbf{S}}) dt, \\ \mathbf{R}(T) &= \mathbf{R}(0) + \int_0^T f_{\mathbf{R}}(\mathbf{R}(t); \boldsymbol{\theta}_{\mathbf{R}}) dt,\end{aligned}\quad (10)$$

where the function  $f_{\mathbf{T}} : \mathbb{R}^L \rightarrow \mathbb{R}^L$  is a neural network with parameters  $\boldsymbol{\theta}_{\mathbf{T}}$  that are learned from the data and captures the dynamics of the data with the trend. The function  $f_{\mathbf{T}}$  is defined as  $f_{\mathbf{T}} := \frac{d\mathbf{T}(t)}{dt} = \mathbf{W}_{\mathbf{T}}\mathbf{T}(t)$ , where  $\mathbf{W}_{\mathbf{T}} \in \mathbb{R}^{L \times L}$  is a weight matrix associated with the trend component, modeled as a single linear layer. Starting with an initial value,  $\mathbf{T}(0) = \mathbf{T}$ , the NODE block computes the output,  $\mathbf{T}(T)$ , at the terminal time  $T$  by solving the initial value problems. The  $f_{\mathbf{S}}$  and  $f_{\mathbf{R}}$  are defined and parameterized in the same manner as  $f_{\mathbf{T}}$ .

Note that we use various ODE solvers with the Jacobian and kinetic regularization [31]. Leveraging these regularizers allows for more accurate regularization of the learned dynamics at each time point, which exhibits diverse patterns.

Since NODE requires the same dimension size of input and output, the results of the ODE solvers are decoded into the forecasting horizon through the decoding layer, as follows:

$$\begin{aligned}\widehat{\mathbf{T}} &= \text{FC}_{L \rightarrow H}^{\mathbf{T}}(\mathbf{T}(T)), \\ \widehat{\mathbf{S}} &= \text{FC}_{L \rightarrow H}^{\mathbf{S}}(\mathbf{S}(T)), \\ \widehat{\mathbf{R}} &= \text{FC}_{L \rightarrow H}^{\mathbf{R}}(\mathbf{R}(T)),\end{aligned}\quad (11)$$

where each  $\text{FC}_{L \rightarrow H}$  means a fully-connected layer whose input size is  $L$  and output size is  $H$ .

The decomposed components  $\mathbf{T}$ ,  $\mathbf{S}$ , and  $\mathbf{R}$  of the observed sequence yield the future decomposed components  $\widehat{\mathbf{T}}$ ,  $\widehat{\mathbf{S}}$ , and  $\widehat{\mathbf{R}}$  after passing through the NODE block.

#### E. NORM & DENORM Blocks

If the dataset has distribution discrepancy problems, we apply instance normalization to the trend and residual components using the NORM and DENORM blocks. In Section IV-C, it will be discussed why instance normalization is only done on the trend and residual components. Instance normalization is performed on feature dimensions. Specifically, given the original component  $\mathbf{C}$ , which could be a trend or a residual, represented as  $\mathbf{C}_{ij} \in \mathbb{R}^{L \times F}$ , the normalization procedure can be expressed as follows:

$$\boldsymbol{\mu}_i = \frac{1}{F} \sum_{j=1}^F \mathbf{C}_{ij}, \quad \sigma_i^2 = \frac{1}{F} \sum_{j=1}^F (\mathbf{C}_{ij} - \boldsymbol{\mu}_i)^2, \quad (12)$$

$$\widetilde{\mathbf{C}}_{ij} = \frac{\mathbf{C}_{ij} - \boldsymbol{\mu}_i}{\sigma_i}, \quad (13)$$

TABLE II  
COMPARISON WITH EXISTING BASELINE MODELS, WHICH USE DECOMPOSITION AND NORMALIZATION

Models	Main Architecture	Decomposition	Normalization
FEDformer [40]	Transformer	Frequency	-
Autoformer [41]	Transformer	Fixed (T/S)	-
NLinear [21]	Single Linear	-	Subtraction
DLinear [21]	Single Linear	Fixed (T/R)	-
LTSF-DNODE	Neural ODE	Selective (T/S/R)	Instance Normalization

where  $L$  is the length of the sequence,  $F$  is the number of features,  $\boldsymbol{\mu}_i$  and  $\sigma_i^2$  are the  $i$ -th mean and variance of the original data, and  $\widetilde{\mathbf{C}}_{ij}$  is normalized component.

DENORM block conducts the inverse process of normalization using the preserved mean ( $\boldsymbol{\mu}_i$ ) and standard deviation ( $\sigma_i$ ) obtained during the normalization process. When normalized component  $\widetilde{\mathbf{C}}$  is given, the denormalization procedure performs the reverse of normalization as follows:

$$\mathbf{C}_{ij} = \sigma_i \widetilde{\mathbf{C}}_{ij} + \boldsymbol{\mu}_i, \quad (14)$$

#### F. Forecasting

To forecast future series, if the dataset is normalized, we apply denormalization to restore the original distribution, and we add up  $\widehat{\mathbf{T}}$ ,  $\widehat{\mathbf{S}}$ , and  $\widehat{\mathbf{R}}$  to obtain the final prediction  $\widehat{\mathbf{Y}}$ :

$$\widehat{\mathbf{Y}} = \widehat{\mathbf{T}} + \widehat{\mathbf{S}} + \widehat{\mathbf{R}}. \quad (15)$$

The values used as Trend regularizers are defined as follows:

$$\dot{E}_{\mathbf{T}}(t) = \|f_{\mathbf{T}}(\mathbf{T}(t); \boldsymbol{\theta})\|^2, \quad (16)$$

$$\dot{J}_{\mathbf{T}}(t) = \|\epsilon^{\top} \nabla f_{\mathbf{T}}(\mathbf{T}(t); \boldsymbol{\theta})\|, \quad (17)$$

where ‘ $\cdot$ ’ denotes derivative,  $\epsilon$  is sampled from standard normal distribution, and  $f_{\mathbf{T}}$  refers to the function defined in Eq. (10). The same definitions are applied to the seasonality and residual components as well.

The proposed model is trained using the mean square error (MSE) loss function with regularization terms as follows:

$$\begin{aligned}Loss &= \frac{\sum_{i=1}^n (\mathbf{Y}_i - \widehat{\mathbf{Y}}_i)^2}{n} \\ &+ \sum_{j \in \{\mathbf{T}, \mathbf{S}, \mathbf{R}\}} (\lambda_K E_j(t) + \lambda_J J_j(t))\end{aligned}\quad (18)$$

where  $\mathbf{Y}_i$  is the  $i$ -th element of the ground truth  $\mathbf{Y}$ ,  $n$  is the number of elements, and  $\{\mathbf{T}, \mathbf{S}, \mathbf{R}\}$  each represent the Trend, Seasonality, and Residual in the summation, respectively. The coefficients  $\lambda_K$  and  $\lambda_J$  are used as hyperparameters.

#### G. Comparison with Existing Models

Table II shows the main differences between our model and the baselines. In contrast to the baseline models using single linear and Transformer, our model employs NODE as the main architecture. If we remove the NODE framework from our proposed model, it closely resembles a combination of the NLinear and DLinear. However, our approach employs

TABLE III

MSE AND MAE RESULTS FOR MULTIVARIATE TIME SERIES FORECASTING ON BENCHMARK DATASETS. WE SET THE LOOK-BACK WINDOW SIZE AS 104 FOR ILI AND 336 FOR THE OTHERS. THE ILI DATASET HAS A FORECASTING HORIZON  $H \in \{24, 36, 48, 60\}$ . FOR THE OTHERS,  $H \in \{96, 192, 336, 720\}$ . THE BEST RESULTS ARE HIGHLIGHTED IN **BOLD**.

Datasets	Forecasting horizon	LTSF-DNODE		NLinear		DLinear		FEDformer		Autoformer		Informer		Pyraformer		LogTrans	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Electricity	96	<b>0.140</b>	<b>0.237</b>	0.141	<b>0.237</b>	<b>0.140</b>	<b>0.237</b>	0.193	0.308	0.201	0.317	0.274	0.368	0.386	0.449	0.258	0.357
	192	<b>0.153</b>	0.249	0.154	<b>0.248</b>	<b>0.153</b>	0.249	0.201	0.315	0.222	0.334	0.296	0.386	0.386	0.443	0.266	0.368
	336	<b>0.168</b>	0.267	0.171	<b>0.265</b>	0.169	0.267	0.214	0.329	0.231	0.338	0.300	0.394	0.378	0.443	0.280	0.380
	720	<b>0.203</b>	0.300	0.210	<b>0.297</b>	<b>0.203</b>	0.301	0.246	0.355	0.254	0.361	0.373	0.439	0.376	0.445	0.283	0.376
Exchange	96	<b>0.078</b>	<b>0.200</b>	0.089	0.208	0.081	0.203	0.148	0.278	0.197	0.323	0.847	0.752	0.376	1.105	0.968	0.812
	192	<b>0.155</b>	<b>0.292</b>	0.180	0.300	0.157	0.293	0.271	0.380	0.300	0.369	1.204	0.895	1.748	1.151	1.040	0.851
	336	<b>0.259</b>	<b>0.388</b>	0.331	0.415	0.305	0.414	0.460	0.500	0.509	0.524	1.672	1.036	1.874	1.172	1.659	1.081
	720	<b>0.606</b>	<b>0.591</b>	1.033	0.780	0.643	0.601	1.195	0.841	1.447	0.941	2.478	1.310	1.943	1.206	1.941	1.127
Weather	96	<b>0.174</b>	0.234	0.182	<b>0.232</b>	0.176	0.237	0.217	0.296	0.266	0.336	0.300	0.384	0.896	0.556	0.458	0.490
	192	<b>0.215</b>	0.272	0.225	<b>0.269</b>	0.220	0.282	0.276	0.336	0.307	0.367	0.598	0.544	0.622	0.624	0.658	0.589
	336	<b>0.262</b>	0.311	0.271	<b>0.301</b>	0.265	0.319	0.339	0.380	0.359	0.395	0.578	0.523	0.739	0.753	0.797	0.652
	720	<b>0.323</b>	0.362	0.338	<b>0.348</b>	<b>0.323</b>	0.362	0.403	0.428	0.419	0.428	1.059	0.741	1.004	0.934	0.869	0.675
ILI	24	<b>1.626</b>	<b>0.848</b>	1.683	0.858	2.215	1.081	3.228	1.260	3.483	1.287	5.764	1.677	1.420	2.012	4.480	1.444
	36	<b>1.589</b>	<b>0.845</b>	1.703	0.859	1.963	0.963	2.679	1.080	3.103	1.148	4.755	1.467	7.394	2.031	4.799	1.467
	48	<b>1.566</b>	<b>0.861</b>	1.719	0.884	2.130	1.024	2.622	1.078	2.669	1.085	4.763	1.469	7.551	2.057	4.800	1.468
	60	<b>1.693</b>	<b>0.911</b>	1.819	0.917	2.368	1.096	2.857	1.157	2.770	1.125	5.264	1.564	7.662	2.100	5.278	1.560
ETTh1	96	<b>0.369</b>	<b>0.392</b>	0.374	0.394	0.375	0.399	0.376	0.419	0.449	0.459	0.865	0.713	0.664	0.612	0.878	0.740
	192	<b>0.403</b>	<b>0.411</b>	0.408	0.415	0.405	0.416	0.420	0.448	0.500	0.482	1.008	0.792	0.790	0.681	1.037	0.824
	336	<b>0.423</b>	<b>0.422</b>	0.429	0.427	0.439	0.443	0.459	0.465	0.521	0.496	1.107	0.809	0.891	0.738	1.238	0.932
	720	<b>0.425</b>	<b>0.444</b>	0.440	0.453	0.472	0.490	0.506	0.507	0.514	0.512	1.181	0.865	0.963	0.782	1.135	0.852
ETTh2	96	<b>0.272</b>	<b>0.334</b>	0.277	0.338	0.289	0.353	0.346	0.388	0.358	0.397	3.755	1.525	0.645	0.597	2.116	1.197
	192	<b>0.334</b>	<b>0.375</b>	0.344	0.381	0.383	0.418	0.429	0.439	0.456	0.452	5.602	1.931	0.788	0.683	4.315	1.635
	336	<b>0.341</b>	<b>0.390</b>	0.357	0.400	0.448	0.465	0.496	0.487	0.482	0.486	4.721	1.835	0.907	0.747	1.124	1.604
	720	<b>0.387</b>	<b>0.428</b>	0.394	0.436	0.605	0.551	0.463	0.474	0.515	0.511	3.647	1.625	0.963	0.783	3.188	1.540
ETTh1	96	<b>0.299</b>	<b>0.343</b>	0.306	0.348	<b>0.299</b>	<b>0.343</b>	0.379	0.419	0.505	0.475	0.672	0.571	0.543	0.510	0.600	0.546
	192	<b>0.334</b>	<b>0.364</b>	0.349	0.375	0.335	0.365	0.426	0.441	0.553	0.496	0.795	0.669	0.557	0.537	0.837	0.700
	336	<b>0.369</b>	<b>0.386</b>	0.375	0.388	<b>0.369</b>	<b>0.386</b>	0.445	0.459	0.621	0.537	1.212	0.871	0.754	0.655	1.124	0.832
	720	<b>0.424</b>	<b>0.419</b>	0.433	0.422	0.425	0.421	0.543	0.490	0.671	0.561	1.166	0.823	0.908	0.724	1.153	0.820
ETTh2	96	<b>0.163</b>	<b>0.253</b>	0.167	0.255	0.167	0.260	0.203	0.287	0.255	0.339	0.365	0.453	0.435	0.507	0.768	0.642
	192	<b>0.217</b>	<b>0.291</b>	0.221	0.293	0.224	0.303	0.269	0.328	0.281	0.340	0.533	0.563	0.730	0.673	0.989	0.757
	336	<b>0.270</b>	<b>0.325</b>	0.274	0.327	0.281	0.342	0.325	0.366	0.339	0.372	1.363	0.887	1.201	0.845	1.334	0.872
	720	<b>0.359</b>	<b>0.381</b>	0.368	0.384	0.397	0.421	0.421	0.415	0.433	0.432	3.379	1.338	3.625	1.451	3.048	1.328

both data characteristic-dependent decomposition and normalization methods, as opposed to the consistent method used in other baselines. The normalization and decomposition methods of each model can be found in Table II.

#### IV. EXPERIMENTAL EVALUATIONS

In this section, we compare the performance of LTSF-DNODE for multivariate LTSF on real-world datasets against baselines and provide an analysis of the model’s architecture with respect to data characteristics.

##### A. Experimental Environments

All experiments are conducted in the following software and hardware environments: UBUNTU 18.04.4 LTS, PYTHON 3.8.13, PYTORCH 1.11.0, CUDA V10.0.130, NVIDIA QUADRO RTX 6000/8000.

We select several cases for each dataset by varying hyper-parameters. The learning rates are in  $\{0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001\}$  and the batch sizes are in  $\{8, 16, 32, 64\}$ . The number of training epochs is set to a maximum of 100. With the validation dataset, an early-stop approach with a patience of 10 iterations is applied. We implement the ODE function of the NODE framework using a single linear layer without

an activation function and use Euler, RK4, and DOPRI as the ODE solvers. The coefficients of the Jacobian and kinetic regularizers are in  $\{0.1, 0.2, \dots, 1.0\}$ . The terminal time, denoted as  $T$ , is set to 1.

1) *Datasets*: In order to evaluate LTSF-DNODE, we conduct experiments on real-world datasets with diverse characteristics collected from various domains such as energy, economics, and more. These datasets possess a wide range of features (see Table IV for its detailed descriptions).

- Electricity records the amount of electricity consumption of 321 customers from 2012 to 2014.
- Exchange Rate [42] consists of the daily records of the exchange rates of eight countries from 1990 to 2016.
- Weather consists of data from 21 weather-related features (e.g. air temperature, humidity), recorded in Germany with 10-min interval during 2020.
- ILI (Influenza-like Illness) is compiled by the Centers for Disease Control and Prevention of the United States from 2002 to 2021.
- ETT (Electricity Transformer Temperature) [43] consists of two datasets with hourly granularity and two datasets with 15-minute granularity. Each data shows seven oil

TABLE IV  
THE ANALYSIS RESULTS OF THE BENCHMARK DATASETS. THE SELECTED KERNEL SIZE AND PERIOD YIELDS THE HIGHEST SEASONALITY.

Datasets	Electricity	Exchange	Weather	ILI	ETTh1	ETTh2	ETTM1	ETTM2
Features	321	8	21	7	7	7	7	7
Timesteps	26,304	7,588	52,696	966	17,420	17,420	69,680	69,680
Granularity	1hour	1day	10min	1week	1hour	1hour	15min	15min
Forecastability	0.126	0.159	0.141	0.173	0.148	0.156	0.142	0.144
Trend	$-4.00 \times 10^{-6}$	$1.84 \times 10^{-4}$	$4.00 \times 10^{-6}$	$1.46 \times 10^{-3}$	$-1.90 \times 10^{-5}$	$-7.90 \times 10^{-5}$	$-5.40 \times 10^{-5}$	$-2.00 \times 10^{-4}$
Kernel size	25	10	10	25	10	25	50	25
Period	24	7	6	52	48	24	7	7
Seasonality	98.70%	11.25%	52.38%	22.22%	97.02%	93.45%	94.20%	80.21%
Stationarity	100.00%	100.00%	99.87%	96.83%	100.00%	100.00%	100.00%	100.00%

and load related properties of transformer. The data was aggregated from July 2016 to July 2018.

2) *Baselines*: We consider the following 7 baselines for long-term time series forecasting. These baselines consist of five Transformer-based models and two Linear-based models, the previous state-of-the-art models:

- Linear-based models: NLinear and DLinear [21]
- Transformer-based models: Informer [43], Pyraformer [44], and LogTrans [45]
- Transformer-based models with decomposition method: FEDformer [40], and Autoformer [41]

3) *Metrics*: To evaluate our model, we use MSE and mean absolute error (MAE). These metrics are as follows:

$$\text{MSE} = \frac{\sum_{i=1}^n (\mathbf{Y}_i - \hat{\mathbf{Y}}_i)^2}{n}, \text{MAE} = \frac{1}{n} \sum_{i=1}^n \left| \frac{\mathbf{Y}_i - \hat{\mathbf{Y}}_i}{\mathbf{Y}_i} \right|, \quad (19)$$

where  $\mathbf{Y}_i$  is the  $i$ -th element of the ground truth  $\mathbf{Y}$ ,  $n$  is the number of elements.

## B. Main Results

In Table III, we list the results of the multivariate long-term time series forecasting. The forecasting horizon refers to the length of the sequence that the model aims to predict. LTSF-DNODE shows better performance in most cases than the previous models. This result indicates that LTSF-DNODE effectively utilized the essential temporal information emphasized in [21] for LTSF, leveraging precise preprocessing methods and forecasting modeling based on NODE.

There are performance improvements (e.g., up to 15.08% on MSE and up to 6.28% on MAE) observed in the Exchange, ILI, ETTh1, ETTh2, and ETTm2 datasets. In the Electricity, Weather, and ETTm1 datasets, our model demonstrated performance similar to that of Linear-based models.

These improvements are attributed to decomposition and normalization methods tailored to the data characteristics. In addition, it can be argued that formulating between past and future values as Eq. (6) contributed positively to LTSF by accurately capturing the dynamics of time series. The subsequent section discusses the analysis results of data characteristics and how each component of our proposed model contributed to performance enhancement.

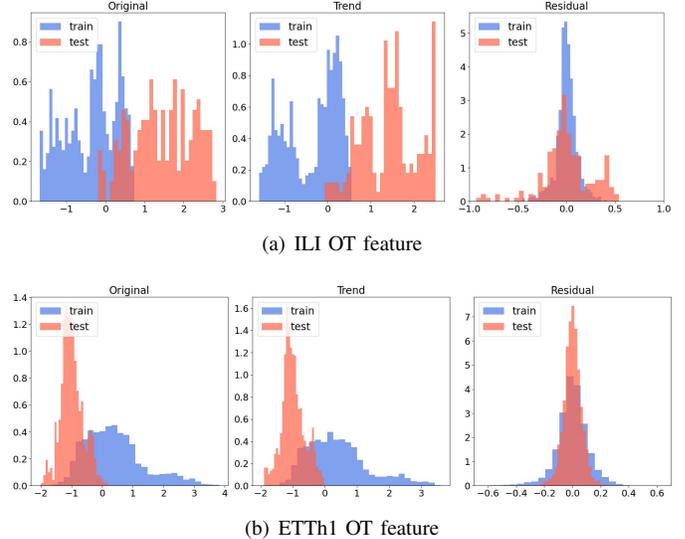


Fig. 3. Distribution of original, trend, and residual for feature OT in (a) ILI and (b) ETTh1. There is a clear discrepancy in the distribution between the train and test datasets.

## C. Data Analysis Results

The result of exploratory analysis on each dataset can be found in Table IV. “Forecastability” and “Trend” metrics are evaluated for the full sequence, with averaging across the features. “Seasonality” is determined based on sub-sequences with a length (104 for ILI and 336 for the others) using kernel sizes of  $\{10, 25, 50\}$  and periods that are determined based on the “Granularity” (e.g., the period is 24 for an hour-based dataset to determine the seasonality of a day, 48 for two days). In Table IV, “Seasonality” is the highest ratio obtained among combinations of kernel sizes and periods. Similarly, we calculate the proportion of sequences exhibiting stationarity for each feature and take the average sequences sliced of length 720 (also 104 for ILI).

Based on the results of the ACF test, we select three candidates with the highest seasonality ratio for each kernel size. Then, using the outcomes of the ADF test, we compare the stationarity ratio of the residuals to identify the best parameter combination. Since the residual should show stationarity, we

TABLE V  
ACCURACY COMPARISON BETWEEN LTSF-DNODE AND ITS VARIANTS.  
(NUMBERS) UNDER DATASETS ARE FORECASTING HORIZONS.

Models	Metrics	Exchange (96)	ILI (24)	ETTh1 (336)	ETTh2 (336)	ETTh2 (720)
<i>w/o.</i> DCOMP	MSE	0.088	1.662	0.429	0.342	0.366
	MAE	0.206	<b>0.844</b>	0.427	0.392	0.382
<i>w/o.</i> NORM	MSE	0.082	1.708	0.433	0.399	0.401
	MAE	0.205	0.894	0.434	0.437	0.423
<i>w/o.</i> NODE	MSE	0.082	<b>1.620</b>	0.426	0.360	0.365
	MAE	0.205	0.893	0.424	0.400	0.383
LTSF-DNODE	MSE	<b>0.078</b>	1.626	<b>0.423</b>	<b>0.341</b>	<b>0.359</b>
	MAE	<b>0.200</b>	0.848	<b>0.422</b>	<b>0.390</b>	<b>0.381</b>

select the parameter combination that shows a higher ratio of stationarity in the residual. If it is ambiguous to differentiate the candidates by stationarity ratios, we compare the p-values obtained from the ADF test for each sequence. Since a lower p-value indicates a stronger indication of stationarity, we select the parameter combination with a greater number of sequences having lower p-values as the optimal choice.

Additionally, we determine whether to extract seasonality during the decomposition process based on the seasonality ratios. The Exchange, ILI, and Weather datasets have lower seasonality ratios compared to other datasets, indicating that these datasets have a lower significance of seasonality. We decompose these datasets without extracting seasonality.

Fig. 3 shows a clear distribution discrepancy between trend components of the training and testing datasets. It was observed in the ETTh1, ETTh2, ETTh2, and ILI datasets. For these datasets, we adopt instance normalization to mitigate distribution discrepancies. There are less significant changes in the distributions of residual components. Nevertheless, we have observed different variances between the training and testing datasets, and to mitigate these, we apply instance normalization to residual components. For the seasonality component, instance normalization is not applied since seasonality repeats values periodically, as defined.

#### D. Ablation Studies

In order to understand the effectiveness of each block of LTSF-DNODE, we conduct ablation studies on the datasets.

1) *Decomposition*: To assess the effectiveness of the DCOMP block (utilizing the classical decomposition method), we compare the performance of the LTSF-DNODE model with the “*w/o.* DCOMP”, where the DCOMP block is omitted. LTSF-DNODE performs better compared to “*w/o.* DCOMP”, with a maximum improvement of 11.26% on MSE and 3.01% on MAE in the Exchange dataset. As a result, we can infer that time series decomposition has a positive impact on forecasting.

Additionally, we conduct an ablation study to investigate alternative decomposition methods, specifically frequency-based approaches, as follows:

- “FTLinear” employs Fourier transform [46] to decompose the time series. It uses low or high-pass filters with

TABLE VI  
MSE AND MAE ON VARIOUS DECOMPOSITION METHODS

Models	Metrics	Exchange (96)	ILI (24)	ETTh1 (336)	ETTh2 (336)	ETTh2 (720)
FTLinear	MSE	0.087	1.978	0.442	0.410	0.409
	MAE	0.215	0.983	0.443	0.436	0.426
WTLinear	MSE	0.086	<b>1.948</b>	0.438	<b>0.408</b>	0.414
	MAE	0.213	<b>0.969</b>	0.439	<b>0.435</b>	0.429
TSRLinear	MSE	<b>0.079</b>	1.981	<b>0.437</b>	0.422	<b>0.385</b>
	MAE	<b>0.201</b>	0.979	<b>0.438</b>	0.444	<b>0.409</b>

frequency criteria of  $1.00 \times 10^{-4}$ . The filtered time series passes through individual single linear layers.

- “WTLinear” employs Wavelet transform [46] to decompose the time series. This model has a structure similar to “FTLinear” and uses a low or high-pass filter.
- “TSRLinear” decomposes time series into trend, seasonality, and residual components using classical methods. This model is identical to the LTSF-DNODE, excluding the NORM and NODE blocks.

Table VI shows the forecasting results of these models. In various benchmark datasets, the classical decomposition method is the better option than other methods.

2) *Normalization*: To assess how NORM and DENORM blocks impact a dataset with distribution disparities, we compare the performance of LTSF-DNODE with “*w/o.* NORM”, which excludes instance normalization. LTSF-DNODE improves performance by up to 14.54% in MSE and 10.76% in MAE compared to “*w/o.* NORM” in the ETTh2 dataset.

Therefore, it can be inferred that instance normalization is effective when there exists a distribution discrepancy between the training and testing datasets.

3) *Neural ODEs*: Finally, we investigate the effectiveness of modeling the relationship between the past and the future in LTSF using linear ODEs, as shown in Eq. (6). We compare two variants: “*w/o.* NODE” and LTSF-DNODE. To investigate the effectiveness of NODE, we compare “*w/o.* NODE” and LTSF-DNODE. Both models utilize decomposition and normalization methods. However “*w/o.* NODE” has a single linear layer, while LTSF-DNODE adopts the NODE framework. The experimental results show that LTSF-DNODE outperforms “*w/o.* NODE” with a maximum performance improvement of 5.28% on MSE and 2.50% on MAE in the ETTh2 dataset.

Also, we explore the effect of various learning techniques, such as ODE solver and regularizer, within the NODE framework. The ODE solver is one of the crucial factors in the NODE framework. Table VII shows the MSE and MAE against various ODE solvers. Additionally, the Jacobian and kinetic regularizers enable smoother learning. It transforms unstable learning into stable learning, leading to an improvement in forecasting accuracy. In Fig. 4 (a), we can see the loss decreases as the Jacobian and kinetic coefficients increase, respectively; however, in Fig. 4 (b), it does not. These findings indicate that the effectiveness of the regularizer depends on the dataset. As a result, the incorporation of a regularizer suggests

TABLE VII  
MSE AND MAE ON VARIOUS ODE SOLVERS. IT WAS CONDUCTED FOR THE ILI DATASET WITH THE FORECASTING HORIZON OF 60.

ODE Solver	MSE	MAE
Euler	1.836	0.942
RK4	1.693	0.911

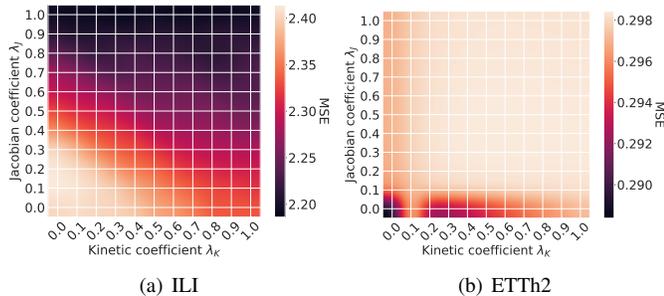


Fig. 4. Analyses about the impact of the Jacobian and kinetic regularization on performance (MSE). The forecasting horizon is 60 in (a) and 96 in (b).

its potential to enhance the learning process of our proposed model.

4) *Model Efficiency Analyses*: Fig. 5 shows the number of parameters and the MSE of models. Our LTSF-DNODE model employs fewer parameters compared to Transformer-based models and slightly more parameters compared to Linear-based models. However, LTSF-DNODE outperforms these models on various datasets.

## V. RELATED WORK

Recently, various Transformer-based models have shown encouraging results in the field of LTSF. Transformer-based LTSF models overcome the limitations of RNNs in LTSF by enabling direct multi-step forecasting, but difficulties remain due to quadratic time complexity, high memory usage, and the inherent limitations of the encoder-decoder architecture.

Transformer-based LTSF models have addressed these issues in various ways. LogTrans [45] utilizes a convolutional self-attention mechanism to address challenges related to locality-agnostic property and memory limitations. Pyraformer [44] uses an interscale tree structure, reflecting a multi-resolution representation of a time series dataset. Informer [43] introduces a ProbSparse self-attention to minimize time complexity and reduce the processing time. Also, it uses a generative decoder to improve performance.

On the other hand, other transformer-based models have attempted to learn the characteristics of time series by combining self-attention structures and decomposition methods. Autoformer [41] aims to enhance forecasting accuracy by capturing auto-correlation and employing the decomposition method, which extracts trend and seasonality. This structure enables progressive decomposition capabilities for intricate time series. FEDformer [40] also utilizes the Fast Fourier

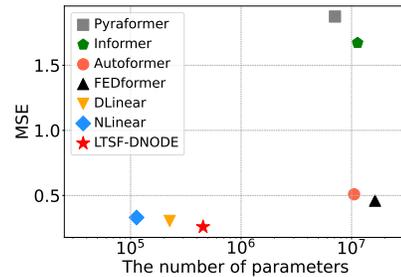


Fig. 5. Model efficiency. It was measured for the Exchange dataset. The bottom left corner is preferred.

transform with a low-rank approximation for preprocessing the temporal data. Moreover, it employs a mixture of expert decomposition, which extracts trend and seasonality, to manage the distribution discrepancy problem.

Due to the issues with Transformers, non-Transformer-based models have also emerged. N-BEATS [47] incorporates fully-connected networks with trend and seasonality decomposition to enhance interpretability. Furthermore, DEPTS [48] advances N-BEATS to propose a more effective model that specializes in learning periodic time series. SNaive [22] uses simple linear regression and demonstrates that a simple statistical model using linear regression could achieve better performance. NLinear and DLinear [21] are Linear-based models that use a single linear layer with simple preprocessing. NLinear utilizes a normalization method that subtracts the last value of the sequence from the input. DLinear utilizes the classical decomposition method that decomposes the input into a trend and a remainder.

## VI. CONCLUSION AND FUTURE WORK

Long-term time series forecasting is an important research topic in deep learning, and simple models such as Linear-based approaches are showing good performance. However, these models are too simple to represent the dynamics of the time series. Our proposed method, LTSF-DNODE, uses a neural ODE (NODE) framework with a simple architecture and utilizes decomposition depending on data characteristics. Our contribution allows the model to not only use temporal information appropriately but also capture time series dynamically. We experimentally demonstrated that LTSF-DNODE outperforms the existing baselines on real-world benchmark datasets. In ablation studies, we analyzed how the main components of LTSF-DNODE affect performance.

In future work, we will refine the modeling of the residual component. Our analysis results demonstrate the evident stationarity of the residual component within the optimal settings we have identified. Therefore, we hypothesize that the performance of LTSF can be further improved through an elaborate modeling of the residual component, potentially via an approach such as stochastic differential equations (SDEs).

## ACKNOWLEDGEMENT

Noseong Park is the corresponding author. This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (No. 2020-0-01361, Artificial Intelligence Graduate School Program at Yonsei University, 10%), and (No.2022-0-00857, Development of AI/data-based financial/economic digital twin platform, 90%)

## REFERENCES

- [1] B. Lim and S. Zohren, "Time-series forecasting with deep learning: a survey," *Philosophical Transactions of the Royal Society A*, vol. 379, no. 2194, p. 20200209, 2021.
- [2] Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, and L. Sun, "Transformers in time series: A survey," *arXiv preprint arXiv:2202.07125*, 2022.
- [3] J. Hwang, J. Choi, H. Choi, K. Lee, D. Lee, and N. Park, "Climate modeling with neural diffusion equations," in *ICDM*. IEEE, 2021, pp. 230–239.
- [4] J. Choi, H. Choi, J. Hwang, and N. Park, "Graph neural controlled differential equations for traffic forecasting," in *AAAI*, vol. 36, no. 6, 2022, pp. 6367–6374.
- [5] S. Y. Jhin, H. Shin, S. Hong, M. Jo, S. Park, N. Park, S. Lee, H. Maeng, and S. Jeon, "Attentive neural controlled differential equations for time-series classification and forecasting," in *ICDM*. IEEE, 2021, pp. 250–259.
- [6] S. Y. Jhin, J. Lee, M. Jo, S. Kook, J. Jeon, J. Hyeong, J. Kim, and N. Park, "Exit: Extrapolation and interpolation-based neural controlled differential equations for time-series classification and forecasting," in *TheWebConf (former WWW)*, 2022, pp. 3102–3112.
- [7] J. Choi and N. Park, "Graph neural rough differential equations for traffic forecasting," *ACM Transactions on Intelligent Systems and Technology*, 2023.
- [8] H. Choi, J. Choi, J. Hwang, K. Lee, D. Lee, and N. Park, "Climate modeling with neural advection–diffusion equation," *Knowledge and Information Systems*, vol. 65, no. 6, pp. 2403–2427, 2023.
- [9] S. Hong, M. Jo, S. Kook, J. Jung, H. Wi, N. Park, and S.-B. Cho, "TimeKit: A time-series forecasting-based upgrade kit for collaborative filtering," in *2022 IEEE International Conference on Big Data (Big Data)*. IEEE, 2022, pp. 565–574.
- [10] A. Alqahtani, M. Ali, X. Xie, and M. W. Jones, "Deep time-series clustering: A review," *Electronics*, vol. 10, no. 23, p. 3001, 2021.
- [11] H. Ismail Fawaz, B. Lucas, G. Forestier, C. Petitjean, D. F. Schmidt, J. Weber, G. I. Webb, L. Idoumghar, P.-A. Muller, and F. Petitjean, "Inceptiontime: Finding alexnet for time series classification," *Data Mining and Knowledge Discovery*, vol. 34, no. 6, pp. 1936–1962, 2020.
- [12] W. Chen and K. Shi, "Multi-scale attention convolutional neural network for time series classification," *Neural Networks*, vol. 136, pp. 126–140, 2021.
- [13] S. Y. Jhin, M. Jo, S. Kook, and N. Park, "Learnable path in neural controlled differential equations," in *AAAI*, 2023.
- [14] J. Jeon, J. Kim, H. Song, S. Cho, and N. Park, "Gt-gan: General purpose time series synthesis with generative adversarial networks," *NeurIPS*, vol. 35, pp. 36999–37010, 2022.
- [15] J. Yoon, D. Jarrett, and M. Van der Schaar, "Time-series generative adversarial networks," *NeurIPS*, vol. 32, 2019.
- [16] A. Alaa, A. J. Chan, and M. van der Schaar, "Generative time-series modeling with fourier flows," in *ICLR*, 2021.
- [17] C. Donahue, J. McAuley, and M. Puckette, "Adversarial audio synthesis," *arXiv preprint arXiv:1802.04208*, 2018.
- [18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [19] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014.
- [20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *NeurIPS*, vol. 30, 2017.
- [21] A. Zeng, M. Chen, L. Zhang, and Q. Xu, "Are transformers effective for time series forecasting?" in *AAAI*, vol. 37, no. 9, 2023, pp. 11121–11128.
- [22] H. Li, J. Shao, K. Liao, and M. Tang, "Do simpler statistical methods perform better in multivariate long sequence time-series forecasting?" in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 2022, pp. 4168–4172.
- [23] R. Salles, K. Belloze, F. Porto, P. H. Gonzalez, and E. Ogasawara, "Nonstationary time series transformation methods: An experimental review," *Knowledge-Based Systems*, vol. 164, pp. 274–291, 2019.
- [24] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*. OTexts, 2018.
- [25] R. B. Cleveland, W. S. Cleveland, J. E. McRae, and I. Terpenning, "Stl: A seasonal-trend decomposition," *J. Off. Stat.*, vol. 6, no. 1, pp. 3–73, 1990.
- [26] E. B. Dagum and S. Bianconcini, *Seasonal adjustment methods and real time trend-cycle estimation*. Springer, 2016.
- [27] R. J. Hyndman, "Moving averages." 2011.
- [28] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," *NeurIPS*, vol. 31, 2018.
- [29] J. R. Dormand and P. J. Prince, "A family of embedded runge-kutta formulae," *Journal of computational and applied mathematics*, vol. 6, no. 1, pp. 19–26, 1980.
- [30] J. Zhuang, N. C. Dvornek, S. Tatikonda, and J. S. Duncan, "Mali: A memory efficient and reverse accurate integrator for neural odes," *arXiv preprint arXiv:2102.04668*, 2021.
- [31] C. Finlay, J.-H. Jacobsen, L. Nurbekyan, and A. Oberman, "How to train your neural ode: the world of jacobian and kinetic regularization," in *ICML*. PMLR, 2020, pp. 3154–3164.
- [32] J. Jeon, S. Kang, M. Jo, S. Cho, N. Park, S. Kim, and C. Song, "Lightmove: A lightweight next-poi recommendation for taxicab rooftop advertising," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 3857–3866.
- [33] S. Y. Jhin, M. Jo, T. Kong, J. Jeon, and N. Park, "Ace-node: Attentive co-evolving neural ordinary differential equations," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 736–745.
- [34] S. Hong, H. Shin, J. Choi, and N. Park, "Prediction-based one-shot dynamic parking pricing," in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 2022, pp. 748–757.
- [35] J. Morrill, C. Salvi, P. Kidger, and J. Foster, "Neural rough differential equations for long time series," in *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2021, pp. 7829–7838.
- [36] J. Lee, J. Jeon, J. Hyeong, J. Kim, M. Jo, K. Seungji, N. Park *et al.*, "Lord: Lower-dimensional embedding of log-signature in neural rough differential equations," *arXiv preprint arXiv:2204.08781*, 2022.
- [37] R. Wang, Y. Dong, S. O. Arik, and R. Yu, "Koopman neural forecaster for time series with temporal distribution shifts," *arXiv preprint arXiv:2210.03675*, 2022.
- [38] G. Goerg, "Forecastable component analysis," in *ICML*. PMLR, 2013, pp. 64–72.
- [39] A. Witt, J. Kurths, and A. Pikovsky, "Testing stationarity in time series," *physical Review E*, vol. 58, no. 2, p. 1800, 1998.
- [40] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin, "Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting," in *ICML*. PMLR, 2022, pp. 27268–27286.
- [41] H. Wu, J. Xu, J. Wang, and M. Long, "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting," *NeurIPS*, vol. 34, pp. 22419–22430, 2021.
- [42] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling long-and short-term temporal patterns with deep neural networks," in *SIGIR*, 2018, pp. 95–104.
- [43] 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 *AAAI*, vol. 35, no. 12, 2021, pp. 11106–11115.
- [44] S. Liu, H. Yu, C. Liao, J. Li, W. Lin, A. X. Liu, and S. Dustdar, "Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting," in *ICLR*, 2021.
- [45] 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," *NeurIPS*, vol. 32, 2019.
- [46] R. A. Roberts and C. T. Mullis, *Digital signal processing*. Addison-Wesley Longman Publishing Co., Inc., 1987.

- [47] B. N. Oreshkin, D. Carпов, N. Chapados, and Y. Bengio, "N-beats: Neural basis expansion analysis for interpretable time series forecasting," *arXiv preprint arXiv:1905.10437*, 2019.
- [48] W. Fan, S. Zheng, X. Yi, W. Cao, Y. Fu, J. Bian, and T.-Y. Liu, "Depts: deep expansion learning for periodic time series forecasting," *arXiv preprint arXiv:2203.07681*, 2022.