Addressing Prediction Delays in Time Series Forecasting: A Continuous GRU Approach with Derivative Regularization

Sheo Yon Jhin
Yonsei University
Seoul, South Korea
sheoyonj@yonsei.ac.kr

Seojin Kim
Yonsei University
Seoul, South Korea
bwnebs1@yonsei.ac.kr

Noseong Park
KAIST
Daejeon, South Korea
noseong@kaist.ac.kr

ABSTRACT

Time series forecasting has been an essential field in many different application areas, including economic analysis, meteorology, and so forth. The majority of time series forecasting models are trained using the mean squared error (MSE). However, this training based on MSE causes a limitation known as ***prediction delay***. The prediction delay, which implies the ground-truth precedes the prediction, can cause serious problems in a variety of fields, e.g., finance and weather forecasting — as a matter of fact, predictions succeeding ground-truth observations are not practically meaningful although their MSEs can be low. This paper proposes a new perspective on traditional time series forecasting tasks and introduces a new solution to mitigate the prediction delay. We introduce a continuous-time gated recurrent unit (GRU) based on the neural ordinary differential equation (NODE) which can supervise explicit time-derivatives. We generalize the GRU architecture in a continuous-time manner and minimize the prediction delay through our time-derivative regularization. Our method outperforms in metrics such as MSE, Dynamic Time Warping (DTW) and Time Distortion Index (TDI). In addition, we demonstrate the low prediction delay of our method in a variety of datasets.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning algorithms**; **Artificial intelligence**.

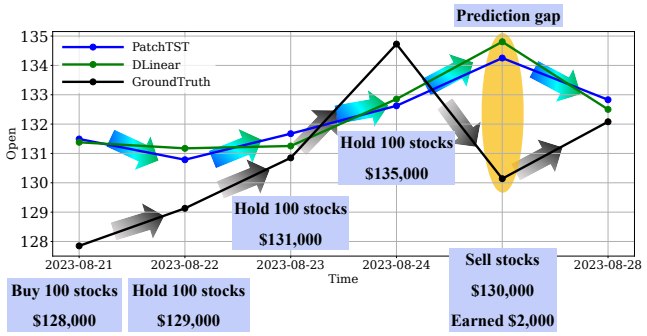
KEYWORDS

Time-series forecasting, Prediction delay, Neural ODE

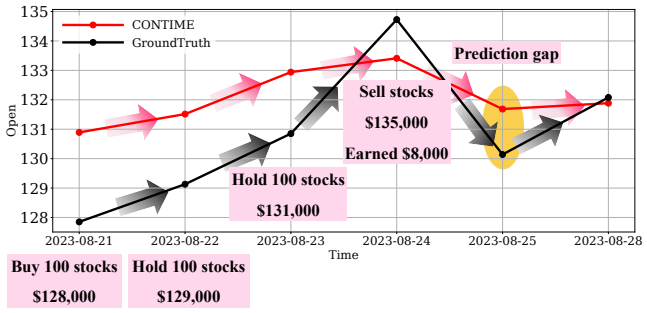
ACM Reference Format:
Sheo Yon Jhin, Seojin Kim, and Noseong Park. 2024. Addressing Prediction Delays in Time Series Forecasting: A Continuous GRU Approach with Derivative Regularization. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24), August 25–29, 2024, Barcelona, Spain.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3637528.3671969

1 INTRODUCTION

Time series forecasting is important in diverse domains, such as weather prediction, stock prediction, and so forth, and has several challenges [4, 15, 22, 29]. The necessity to tackle these practical challenges has spurred numerous proposed studies investigating the intricacies of short- and long-term time series forecasting. Within this realm, a spectrum of models has been suggested, ranging from simple linear networks to advanced transformer-based architectures [19, 23, 34]. Traditionally, the predominant evaluation metrics in most studies have been Mean Squared Error (MSE) or Mean Absolute Error (MAE), with ongoing research endeavors striving to showcase state-of-the-art outcomes through learning based on these metrics. However, the remarkable success achieved in time series forecasting using MSE highlights a limitation related to the prediction delay, as illustrated in Figure 1.(a). In this context, we define ***prediction delay*** as a phenomenon where the actual observations precede the prediction in the time series forecasting task — in other words, a model is trained to output an observation similar



(a) Prediction results from PatchTST, DLinear



(b) Prediction results from CONTIME

Figure 1: Visualization of Table 1 (experimental results for GOOG stock prediction from August 21 to August 28, 2023)

KDD '24, August 25–29, 2024, Barcelona, Spain

ACM ISBN 979-8-4007-0490-1/24/08
https://doi.org/10.1145/3637528.3671969

to the most recent observation, which can lead to reasonable MSE or MAE values but is, in practice, rather meaningless [6, 10, 12, 21].

Table 1: Experimental results on GOOG on $P = 36$

Models	TDI ↓	DTW ↓	MSE ↓
DLinear	4.835	2.229	0.199
PatchTST	4.882	2.766	0.191
CONTIME	**4.712**	**2.189**	**0.189**

Figure 1 visually presents the experimental results detailed in Table 1. Notably, a prediction delay is discernible in state-of-the-art (SOTA) models, exemplified by PatchTST and DLinear [27, 33], despite their relatively small Mean Squared Error (MSE). Conversely, CONTIME, characterized by a comparatively similar MSE, does not exhibit a prediction delay. This study seeks to provide a comprehensive interpretation of time series forecasting by introducing additional metrics, namely Temporal Distortion Index (TDI) and Dynamic Time Warping (DTW), aimed at elucidating the observed phenomenon. Furthermore, Figure 1 underscores the significance of prediction delay in a straightforward scenario. Investors relying on SOTA model forecasts for GOOG stocks anticipate the upper price limit on August 25th, 2023. However, due to a one-day delay in the forecast results, this leads to stock sales (See Figure 1.(a)). In contrast, investors relying on CONTIME, free from prediction delay, predict a stock price decline on August 25th, 2023, prompting them to initiate stock sales on August 24th, 2023 (See Figure 1.(b)). Assuming an investor trades 100 shares of stock, those relying on CONTIME stand to make a profit of approximately $6,000. This achieves more accurate and timely forecasts in real-world applications and provides beneficial forecast results to investors. The example in Figure 1 highlights the importance of mitigating the prediction delay in time series forecasting.

Beyond the financial domain, the aforementioned prediction delay assumes a significant role in areas intricately interwoven with daily life, such as weather forecasting. Despite discussions on these limitations dating back to 1998 [1, 6, 9, 10, 12, 21], recent state-of-the-art studies have predominantly concentrated on evaluating the performance of metrics like MSE, MAE, etc., in the context of time series forecasting. As evident from [21], it is imperative that a model's prediction accurately captures both the *shape* and *temporal* trends within the time series. Dynamic Time Warping (DTW) emerges as a method capable of discerning differences in shape between time series. Additionally, Temporal Distortion Index (TDI) serves as an extra metric to explore the temporal lag between two sequences. The incorporation of these two metrics enables a comprehensive assessment of comparability between the respective time series. Consequently, our intention is to subject the model to evaluation using these novel metrics.

This paper introduces an innovative approach to mitigate the prediction delay in time series forecasting. In this paper, we redefine GRU as differential equation that reflect past observations to the current hidden state for processing continuously generalizing GRU. We propose a continuous-time bi-directional gated recurrent unit (GRU) network based on neural ordinary differential equation (NODE) and train it with explicit time-derivative regularizations, thus addressing the inherent prediction delay observed in various time series forecasting models. We extend the bi-directional GRU to efficiently capture the temporal dependencies within time-series sequences with minimal delays. Our contributions can be summarized as follows:

(1) We propose **CON**tinuous GRU to address the prediction delay in **TIME** series forecasting, i.e., **CONTIME**. By continuously extending the bi-directional GRU, we present a novel architecture that facilitates the supervision of the time-derivative of observations in the continuous time domain.
(2) In Section 3.2, we compute the time-derivatives of the hidden state $\mathbf{h}(t)$, the reset gate $\mathbf{r}(t)$, the update gate $\mathbf{z}(t)$, and the update vector $\mathbf{g}(t)$ of GRU. We strategically employ the bi-directional GRU structure to generate more effective hidden representations for downstream task.
(3) We conduct time series forecasting with minimal prediction delays through our proposed time-derivative regularization.
(4) **CONTIME** demonstrates outstanding performance in addressing the prediction delay across all 6 benchmark datasets. In addition to minimizing the prediction delay, it excels in all three metrics (TDI, DTW, and MSE).
(5) Our code is available at this link [1], and we refer readers to Appendix I for the information on reproducibility.

2 BACKGROUNDS

2.1 Time series forecasting models

In this section, we introduce various time series forecasting models from ODE-based models to recent models.

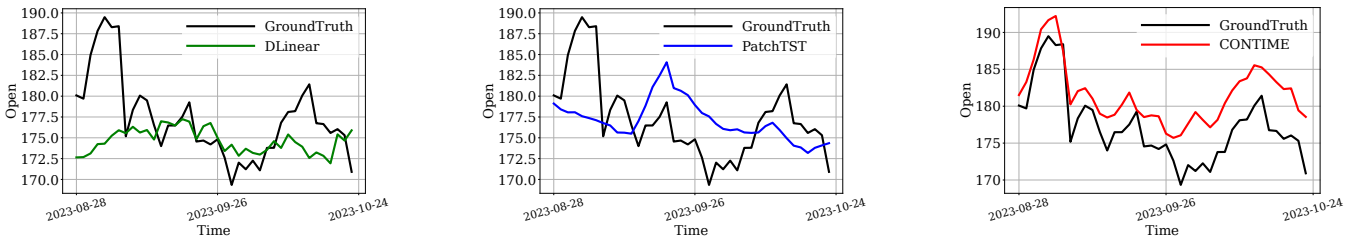
ODE-based Models: Neural ODE enable the processing of time-series data in a *continuous* manner, allowing them to read and write values at any arbitrary time-point t through the differential equation presented in Equation (1).

$$\mathbf{h}(T) = \mathbf{h}(0) + \int_0^T f(\mathbf{h}(t), t; \theta_f)dt, \tag{1}$$

where $\mathbf{h}(t) \in \mathbb{R}^D$, $t \in [0, T]$, represents a D-dimensional vector (with boldface denoting vectors and matrices). The derivative $\dot{\mathbf{h}}(t) \stackrel{\text{def}}{=} \frac{d\mathbf{h}(t)}{dt}$ is approximated by the neural network $f(\mathbf{h}(t), t; \theta_f)$, and solving the initial value problem yields the final value $\mathbf{h}(T)$ from the initial value $\mathbf{h}(0)$. The ODE-based neural network learns by estimating the differential values of the data function $f(\mathbf{h}(t), t; \theta_f)$ using ODE solvers such as the explicit Euler method, the 4th order Runge-Kutta (RK4) method, the Dormand-Prince (DOPRI) technique, and similar approaches [5].

There also exist prominent time-series processing models based on NODE, such as Neural Controlled Differential Equation (NCDE). NCDE, an advanced network of NODE, utilizes the Riemann–Stieltjes integral, as shown in Equation (2). Unlike NODE, which employs the Riemann integral, NCDE can continuously read $X(t)$ values over time. Thus, NCDE can overcome the limitations of NODE that

[1] https://github.com/sheoyon-jhin/CONTIME



(a) DLinear (b) PatchTST (c) CONTIME

Figure 2: Visualization for comparing characteristics of each metric (MSE, DTW, TDI). Forecasting results on AAPL from August 28th, 2023 to October 24th, 2023.

depend on initial values [17].

$$\begin{aligned}\mathbf{h}(T) &= \mathbf{h}(0) + \int_0^T f(\mathbf{h}(t);\theta_f)dX(t) \\ &= \mathbf{h}(0) + \int_0^T f(\mathbf{h}(t);\theta_f)\frac{dX(t)}{dt}dt,\end{aligned} \tag{2}$$

Moreover, NCDE creates a continuous path $X(t)$ by employing interpolation techniques like the natural cubic spline or Hermite cubic spline. Since the natural cubic spline uses all time observations to form a continuous path $X(t)$, so in the context of time-series forecasting, the natural cubic spline method may not be suitable for forecasting tasks. Therefore, in this paper, we opt for the Hermite cubic spline method to generate the continuous path $X(t)$ [26].

Transformer-based models: Subsequent advancements have introduced transformer architectures, originally devised for natural language processing[31], to the domain of time series forecasting, thereby incorporating self-attention mechanisms. These models, utilizing self-attention, have demonstrated remarkable efficacy in capturing overarching dependencies within sequential data, leading to the development of significant transformer-based studies such as Autoformer and FEDformer [32, 35]. While Autoformer employs auto-correlation attention for periodic patterns, it falls short in series decomposition, overly depending on a basic moving average for detrending, which may constrain its ability to capture intricate trend patterns. On the other hand, FEDformer [35] integrates the Transformer with seasonal trend decomposition, utilizing decomposition for global profiles and Transformers for detailed structures. Despite these notable accomplishments, it is crucial to acknowledge that transformer-based architectures exhibit inefficiencies in capturing local dependencies and temporal information. This constraint has spurred continuous research endeavors aimed at addressing and improving the effectiveness of transformer-based models in comprehensively capturing both global and local intricacies within time series data.

Recent state-of-the-art models: The innovative introduction of PatchTST [27] represents a groundbreaking approach that employs patch-based representations to enhance the capture of both local and global patterns within time series data. Building upon this, PatchTST further enhances its methodology by segmenting time series before utilizing a Transformer, demonstrating superior performance compared to existing models. Despite being rooted in the foundational Transformer architecture, innovations are focused on transitioning from self-attention to sparse self-attention, often overlooking a comprehensive global view of time series data. DLinear [33] has significantly contributed to the field by exploring linear models for time series forecasting. In defiance of the prevalent assumption that only highly complex nonlinear models excel in this context, DLinear has exhibited competitive performance with a linear layer, emphasizing efficiency and interoperability.

In summary, the progression from Neural ODE to PatchTST and DLinear signifies an ongoing quest for more effective and efficient deep learning models in the domain of time series forecasting. Each model brings unique features, methodologies, and challenges that challenge prevailing assumptions, with a notable emphasis on a novel approach for model evaluation based on the MSE.

2.2 Evaluation and training metrics

In the realm of evaluating and training deep models for time series forecasting, prevalent approaches heavily depend on metrics such as MAE, MSE, and their variants, including SMAPE. While these metrics effectively gauge overall model performance, evaluating shape and temporal location is crucial for a more comprehensive assessment. Techniques like Dynamic Time Warping (DTW) [28] are employed to capture shape-related metrics, and Temporal Distortion Index (TDI) [11, 30] is utilized for prediction delay estimation. However, due to their non-differentiability, these evaluation metrics are unsuitable as loss functions for training deep neural networks.

Addressing the challenge of optimizing non-differentiable evaluation metrics directly, the development of surrogate losses has been explored across various domains, including computer vision. Recently, alternatives to MSE have been investigated, with a focus on seamless approximations of DTW [8] to train deep neural networks. Despite its effectiveness in assessing shape errors, the inherent design of DTW, i.e., the invariance to elastic distortions, overlooks crucial considerations about the temporal localization of changes. Le Guen and Thome [21] attempted to train models with a loss that combines DTW and TDI to account for both the shape and temporal distortion.

Figure 2 and Table 2 provide examples where each metric (MSE, DTW, TDI) excels in analyzing experimental results (refer to Table 3). By examining the relationship between metric values and visualization results, we gain insight into the role of each metric. In Figure 2, the results of DLinear in Figure 2.(a) demonstrate a relatively

Table 2: Effect of Metrics on Figure 2. *Time* and *Shape* respectively denote the resemblance in timing and shape between the ground-truth and the prediction. (O/X means whether each result displays good output when scrutinized with visualization)

Models	*Time*		*Shape*		Score	
	TDI	Vis	DTW	Vis	MSE	Vis
DLinear	3.810	X	1.409	X	0.084	O
PatchTST	3.166	X	1.253	O	0.091	O
CONTIME	2.378	O	1.114	O	0.074	O

small MSE of 0.084, yet exhibit a lack of superiority in terms of prediction shape and timing. This observation is further supported by the TDI and DTW metrics. It illustrates that a good MSE score does not necessarily guarantee accurate time series prediction. Figure 2.(b) presents predictions with a more accurate shape than Figure 2.(a), but entails a prediction delay in determining the direction of movement. Consequently, TDI and MSE values are large compared to smaller DTW values. This indicates that time series forecasts cannot be evaluated solely using DTW and MSE. Figure 2.(c) showcases the prediction results of CONTIME, demonstrating excellent performance in terms of time series shape and timing, naturally leading to small MSE values. These analyses emphasize the necessity to evaluate time series forecasts from diverse perspectives.

This paper advances this approach by directly computing the gradient of sequences. To enable the instantaneous prediction of rises and declines, we incorporate a regularization component that utilizes time-derivatives. This strategy addresses the gap by introducing time-derivative regularization to the traditional MSE loss. By decoupling the training for prediction delay and the MSE criterion, this paper aims to provide a robust framework for training deep neural networks on real-time series data.

2.3 The prediction delay in time series forecasting

Time series forecasting is a crucial task spanning diverse domains, including finance and environmental science. A significant challenge in this domain is prediction delay, where models may struggle to provide accurate and timely predictions. This subsection explores existing research addressing prediction delay in time series forecasting, emphasizing neural network approaches and other relevant methodologies. In [6], challenges in time series forecasting using neural networks are investigated, and strategies to mitigate forecast delays are proposed. The study assesses the impact of delay on prediction accuracy and explores techniques to enhance prediction timeliness using neural network architectures. In addition, three studies in [10, 12, 24] focus on applying artificial neural networks to rainfall-runoff modeling, economic models, traffic forecasting, and so on and investigating constraints related to forecast delays. One of the studies evaluates the trade-off between hydrological state representation and model evaluation, emphasizing the challenges posed by delays in hydrological forecasting. Beyond the applications like rainfall runoff or wave height predictions, the delay phenomenon is also observed in the economic field.

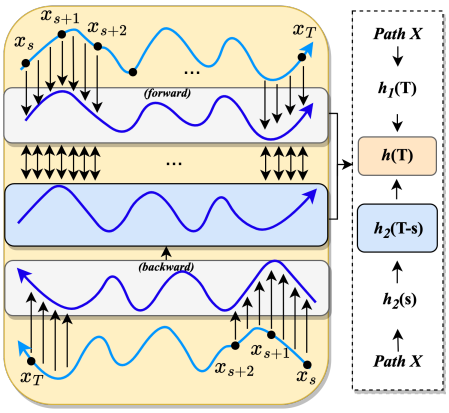


Figure 3: Overall Architecture

Causes of the prediction delay: We categorize two common causes of prediction delay in time series prediction models:

(1) Time series data often exhibit temporal dependence, where current values are influenced by past observations. The prediction delay can occur if the model fails to accurately capture these dependencies or experiences a delay in incorporating relevant historical information.
(2) The prediction delay may arise from MSE-based forecasting models' limited ability to adjust to sudden changes in the time series because their primary objective is to minimize the mean square difference between predicted and actual values [21].

In this paper, we propose CONTIME, a model architecture for supervising time-derivatives to eliminate prediction delays. We add a time-derivative regularization to the main task-dependent loss function to effectively handle prediction delays.

3 PROPOSED METHOD

In this section, we describe our proposed method to address the prediction delay that is common in time series forecasting. In other words, the model we propose is a NODE-based bi-directional continuous GRU that can be explicitly supervised for time-derivatives to address the prediction delay.

3.1 Overall workflow

Figure 3 shows the detailed design of our method, CONTIME. The overall workflow is as follows:

(1) The path X in Figure 3 is created from a discrete time-series sample by Hermite-cubic-spline algorithm.
(2) CONTIME has a bi-directional structure. As shown in Figure 3, we perform bi-directional integral operations (highlighted in the gray box in Figure 3) are conducted both forward ($s \longrightarrow T$) and backward ($T \longrightarrow s$).
(3) After the forward and backward operations are performed, the hidden vector $\mathbf{h}_2(s)$ from the backward operation is rearranged in the forward direction ($\mathbf{h}_2(T-s)$) through the reverse layer (highlighted in the blue box in Figure 3).

(4) We can get final hidden vector $\mathbf{h}(T)$ by adding $\mathbf{h}_1(T)$ and $\mathbf{h}_2(T-s)$.
(5) From the hidden vector $\mathbf{h}(T)$, the linear network produces the future predictions.
(6) We explicitly calculate $\frac{d\hat{Y}}{dt}$ from the forecasting prediction $\hat{Y}$ to supervise the time-derivative ($L_{\Delta t}$).
(7) There is a loss to maintain the accuracy of the existing time series predictions (L_{Task}) and the time-derivative regulation term ($L_{\Delta t}$) to prevent prediction delay.

We describe each part in detail, followed by a theoretical result that shows training the proposed model is well-posed problem.

3.2 Bi-directional CONTIME

We first introduce our formulation to define the proposed CONTIME. The entire module can be written, when we adopt the proposed bi-directional continuous GRU strategy to supervise time-derivative, as follows:

$$\begin{aligned}\mathbf{h}_1(T) &= \mathbf{h}_1(s) + \int_s^T f_1(\mathbf{h}_1(t), t; \theta_{f_1})dt,\\ \mathbf{h}_2(s) &= \mathbf{h}_2(T) + \int_T^s f_2(\mathbf{h}_2(t), t; \theta_{f_2})dt,\end{aligned} \tag{3}$$

where s denotes initial point in time-series sample $X = (X_s, ..., X_T) \in \mathbb{R}^{(T-s)\times F}$, where F means the number of features. $\mathbf{h}_1(s) = \Phi_{\mathbf{h}_1(X_s)}$, $\mathbf{h}_2(T) = \Phi_{\mathbf{h}_2(X_T)}$ and $\Phi_{\mathbf{h}_1}, \Phi_{\mathbf{h}_2}$ is a fully-connected layer-based feature extractor. In Equation (3), we use bi-directional integral operations in the forward ($s \to T$) and backward ($T \to s$) directions to generate a more useful hidden representation in long sequences. After reverse $\mathbf{h}_2(s)$ to $\mathbf{h}_2(T-s)$, we can write our final hidden representation $\mathbf{h}(T)$ as follows:

$$\mathbf{h}(T) = \mathbf{h}_1(T) + \mathbf{h}_2(T-s). \tag{4}$$

In the integration of Equation (3), we use ODE function f_1 and f_2 which can be interpreted as $\frac{d\mathbf{h}_1(t)}{dt}$ and $\frac{d\mathbf{h}_2(t)}{dt}$, thereby explicitly calculating the hidden vector $\mathbf{h}(t)$ of GRUs.

Time-derivative of $\mathbf{h}(t)$*:* GRUs can be written as follows:

$$\begin{aligned}\mathbf{h}(t) &:= \mathbf{z}(t) \odot \mathbf{h}(t-\tau) + (1-\mathbf{z}(t)) \odot \mathbf{g}(t),\\ \mathbf{z}(t) &:= \sigma\big(\mathbf{W}_z X(t) + \mathbf{U}_z \mathbf{h}(t-\tau) + \mathbf{b}_z\big),\\ \mathbf{r}(t) &:= \sigma\big(\mathbf{W}_r X(t) + \mathbf{U}_r \mathbf{h}(t-\tau) + \mathbf{b}_r\big),\end{aligned} \tag{5}$$

where $\mathbf{W} \in \mathbb{R}^{\dim(\mathbf{h})\times\dim(\mathbf{x})}$ and $\mathbf{U} \in \mathbb{R}^{\dim(\mathbf{h})\times\dim(\mathbf{h})}$ are weight matrices, and $\mathbf{b} \in \mathbb{R}^{\dim(\mathbf{h})}$ is a bias vector. σ is a sigmoid function and ϕ is a hyperbolic tangent function. $\tau > 0$ is a delay factor — note that $\tau = 1$ in the original design of GRUs whereas we interpret it in a continuous manner. Since the hidden state $\mathbf{h}(t)$ is a composite function of $\mathbf{r}(t)$, $\mathbf{z}(t)$, and $\mathbf{g}(t)$, the derivative of $\mathbf{h}(t)$ can be written as follows:

$$\begin{aligned}\frac{d\mathbf{h}(t)}{dt} &= \frac{d\mathbf{z}(t)}{dt} \odot \mathbf{h}(t-\tau) + \mathbf{z}(t) \odot \frac{d\mathbf{h}(t-\tau)}{dt}\\ &\quad - \frac{d\mathbf{z}(t)}{dt} \odot \mathbf{g}(t) + (1-\mathbf{z}(t)) \odot \frac{d\mathbf{g}(t)}{dt},\\ &= \frac{d\mathbf{z}(t)}{dt} \odot \big(\mathbf{h}(t-\tau) - \mathbf{g}(t)\big)\\ &\quad + \mathbf{z}(t) \odot \big(\frac{d\mathbf{h}(t-\tau)}{dt} - \frac{d\mathbf{g}(t)}{dt}\big) + \frac{d\mathbf{g}(t)}{dt},\\ &= \frac{d\mathbf{z}(t)}{dt} \odot \zeta(t, t-\tau)\\ &\quad + \mathbf{z}(t) \odot \frac{d\zeta(t, t-\tau)}{dt} + \frac{d\mathbf{g}(t)}{dt},\end{aligned} \tag{6}$$

where $\zeta(t, t-\tau) = \mathbf{h}(t-\tau) - \mathbf{g}(t)$. So, we can write $\frac{d\mathbf{h}(t)}{dt}$ as follows:

$$\frac{d\mathbf{h}(t)}{dt} = \frac{d(\mathbf{z}(t) \odot \zeta(t, t-\tau))}{dt} + \frac{d\mathbf{g}(t)}{dt}. \tag{7}$$

Finally, Equation (3) can be rewritten as follows:

$$\begin{aligned}\mathbf{h}_1(T) &= \mathbf{h}_1(s) + \int_s^T \frac{d(\mathbf{z}_1(t) \odot \zeta_1(t, t-\tau))}{dt} + \frac{d\mathbf{g}_1(t)}{dt}dt,\\ \mathbf{h}_2(s) &= \mathbf{h}_2(T) + \int_T^s \frac{d(\mathbf{z}_2(t) \odot \zeta_2(t, t-\tau))}{dt} + \frac{d\mathbf{g}_2(t)}{dt}dt.\end{aligned} \tag{8}$$

Other derivatives for $\mathbf{z}(t)$, $\mathbf{g}(t)$, and $\mathbf{r}(t)$ are in Appendix A. Finally, the time-derivatives of $\mathbf{h}(t)$, $\mathbf{z}(t)$, $\mathbf{g}(t)$, and $\mathbf{r}(t)$ is written as follows:

$$\frac{d}{dt}\begin{bmatrix}\mathbf{h}(t)\\ \mathbf{z}(t)\\ \mathbf{g}(t)\\ \mathbf{r}(t)\end{bmatrix} := \begin{bmatrix}\frac{d(\mathbf{z}(t)\odot\zeta(t,t-\tau))}{dt} + \frac{d\mathbf{g}(t)}{dt}\\ \sigma\big(\mathbf{A}(t,t-\tau)\big)\big(1-\sigma(\mathbf{A}(t,t-\tau))\big)\frac{d\mathbf{A}(t,t-\tau)}{dt}\\ \big(1-\phi^2(\mathbf{B}(t,t-\tau)\big)\frac{d\mathbf{B}(t,t-\tau)}{dt}\\ \sigma\big(\mathbf{C}(t,t-\tau)\big)\big(1-\sigma(\mathbf{C}(t,t-\tau))\big)\frac{d\mathbf{C}(t,t-\tau)}{dt}\end{bmatrix}. \tag{9}$$

$\frac{dX(t)}{dt}$ contained by the derivatives of $\mathbf{A}, \mathbf{B}$, and $\mathbf{C}$ can also be calculated since we use an interpolation method to construct continuous path $X(t)$ (see Appendix C).

We derive the final prediction result $\hat{Y}$ by passing $\mathbf{h}(T)$ to a single fully-connected layer $\mathtt{FC}_{\theta_p}$:

$$\hat{Y} = \mathtt{FC}_{\theta_p}(\mathbf{h}(T)), \tag{10}$$

where $\hat{Y} := (\hat{Y}_1, ..., \hat{Y}_P) \in \mathbb{R}^{P\times F}$ with the prediction length P.

3.3 Why Continuous GRU?

In this section, we outline the rationale behind selecting GRU as the primary network architecture for CONTIME.

GRU-based network: The hidden representation $\mathbf{h}(t)$ in the GRU (Equation 5) comprises hidden vectors at time t and $t-\tau$. We propose a GRU-based network, called CONTIME, embracing the benefits of modeling past hidden representations in reducing prediction delay.

$\mathbf{h}(t)$ *to* $\frac{d\mathbf{h}(t)}{dt}$*:* Due to the GRU Equation (5) including both time t and time $t-\tau$, $h(t)$ is redefined as the derivative of $h(t)$ w.r.t. time in multiple papers proposing GRU-based networks [3, 24]. Similarly, we redefine GRU Equation (5) as $h(t)$ for time, albeit with $\tau < 1$ compared to the conventional GRU where $\tau = 1$.

Algorithm 1: How to train CONTIME

Input: Training data D_{train}, Validating data D_{val}, Maximum iteration number max_iter

Initialize θ_{f_1}, θ_{f_2}, and other parameters θ_{others} if any, e.g., the parameters of the feature extractor;
Create a continuous path $X(t)$;
$k \leftarrow 0$;
while $k < max\_iter$ **do**
  Train θ_{f_1} and θ_{f_2} and using $L_{CONTIME}$;
  Validate and update the best parameters, $\theta^*_{f_1}$, $\theta^*_{f_2}$, and θ^*_{others}, with D_{val};
  $k \leftarrow k + 1$;
return $\theta^*_{f_1}$, $\theta^*_{f_2}$, and θ^*_{others};

Discrete to continuous: Essentially, we introduce a continuous ODE-based GRU as opposed to the discrete GRU. This continuous approach facilitates detailed and continuous modeling between discrete time points, allowing a more comprehensive representation of the value of $h(t+\tau)$ between $h(t)$ and $h(t+1)$. This sublime design well aligns with our objective of supervising time-derivatives and eliminating prediction delays.

3.4 How to train

Our proposed model, CONTIME, uses a loss based on MSE and a time-derivative regularization to accurately predict time series and prevent prediction delays. The final loss $L_{CONTime}$ is the sum of L_{Task} and our time-derivative loss $L_{\Delta t}$.

$$\begin{aligned} L_{Task} &= \text{MSE}(Y, \hat{Y}), \\ L_{\Delta t} &= \text{MSE}(Y_{\Delta t}, \hat{Y}_{\Delta t}), \end{aligned} \tag{11}$$

where Y is a ground-truth time series and $\hat{Y}$ is an inferred time series. Δt denotes their time-derivatives.

Δt Loss: The purpose of the Δt loss function is to oversee time differentiation. Essentially, it ensures that accurate time series predictions are achieved without any delay by adjusting the increment or decrement pattern. To solve this problem, we explicitly calculate $\frac{d\hat{Y}}{dt}$ as follows:

$$\hat{Y}_{\Delta t} = \frac{d(\texttt{FC}_{\theta_p}(\mathbf{h}(T)))}{dt} = \frac{d(\mathbf{W}_{\theta_P}(\mathbf{h}(T)) + \mathbf{b}_{\theta_p})}{dt} = \mathbf{W}_{\theta_p}\frac{d\mathbf{h}(T)}{dt}, \tag{12}$$

since $\texttt{FC}_{\theta_p}$ is a fully connected layer, $\texttt{FC}_{\theta_p}(\mathbf{h}(T))$ can be written as $\mathbf{W}_{\theta_p}(\mathbf{h}(T)) + \mathbf{b}_{\theta_p})$ and $\frac{d\mathbf{h}(T)}{dt}$ is defined by the ODE function f_1. So, Equation (12) can be easily calculated by the automatic differentiation method. Therefore, we use the MSE loss between $\hat{Y}_{\Delta t}$ and $Y_{\Delta t}$ to supervise the time-derivative, where $Y_{\Delta t} := Y_{t_i} - Y_{t_{i-1}}$ — in other words, we use the difference $Y_{\Delta t}$ to supervise the derivative $\hat{Y}_{\Delta t}$, which is reasonable since we do not know the explicit time-derivative of Y. Our loss function can be summarized as follows:

$$L_{CONTIME} = \alpha L_{Task} + \beta L_{\Delta t}, \tag{13}$$

where α and β are the coefficients of the two terms. Finally, we can summarize our training algorithm in Algorithm 1.

Well-posedness: The well-posedness[2] of NODE was already proved in [25, Theorem 1.3] under the mild condition of the Lipschitz continuity. We show that our CONTIME is also well-posed. Almost all activations, such as ReLU, Leaky ReLU, Tanh, Sigmoid, ArcTan, and Softsign, have a Lipschitz constant of 1. Other common neural network layers, such as dropout, batch normalization, and other pooling methods, have explicit Lipschitz constant values. Therefore, the Lipschitz continuity of $\frac{d\mathbf{h}(t)}{dt}$, $\frac{d\mathbf{r}(t)}{dt}$, $\frac{d\mathbf{z}(t)}{dt}$, and $\frac{d\mathbf{g}(t)}{dt}$ can be fulfilled in our case. Accordingly, it is a well-posed problem. Thus, its training process is stable in practice.

4 EXPERIMENTS

In this section, we describe our experimental environments and results. We conduct experiments on multivariate time series forecasting. All experiments were conducted in the same software and hardware environments. Ubuntu 18.04 LTS, Python 3.8.0, Numpy 1.22.3, Scipy 1.10.1, Matplotlib 3.6.2, PyTorch 2.0.1, CUDA 11.4, NVIDIA Driver 470.182.03 i9 CPU, and NVIDIA RTX A5000. We repeat training and testing procedures with three different random seeds and report their mean scores. We report standard deviations of all 6 datasets in the arXiv version.

4.1 Experimental settings

We list all the descriptions of datasets and detailed experimental settings in Appendix, D, and I.

Baselines: We test the following state-of-the-art baselines to compare our proposed CONTIME with 6 baseline models. (1) DLinear [33] is a simple linear network with time series decomposition method and shows state-of-the-art performance. (2) Neural ODE (NODE [5]) is a continuous-time model that defines the hidden state $\mathbf{h}(t)$ with an initial value problem (IVP). (3) Neural CDE (NCDE [17]) is a conceptually enhanced model of NODE based on the theory of controlled differential equations. (4) Autoformer [32] is a transformer-based method which uses an auto-correlation attention for periodic patterns. (5) FEDformer [35] is a transformer-based method which integrates transformer with seasonal trend decomposition, leveraging decomposition for global profiles and transformers for detailed structures. (6) PatchTST [27] is a time series forecasting technique that makes use of patch-based processing to improve the model's capacity to grasp complex patterns and relationships by segmenting temporal sequences into smaller patches.

Datasets: We evaluate the performance of the proposed CONTIME on six benchmarked datasets, including weather, exchange, and four Stock datasets (AAPL, AMZN, GOOG, MSFT). Among the benchmarked datasets used, weather and exchange are widely utilized and are publicly available at [32]. The Stock dataset has been actively used in [16]. The following is a description of the six experimental data sets. (1) The **Stocks** dataset [2, 7, 13, 14] contains stock prices of four companies (APPLE, AMAZON, Google, and Microsoft). All four datasets measure 6 stock indicators (Open price, High price, Low price, Close price, Adj Close price, and Volume) of each company from January 17th, 2019 to January 4th, 2024.

[2] A well-posed problem means i) its solution uniquely exists, and ii) its solution continuously changes as input data changes.

(2) **Exchange** contains exchange data among 8 countries [20]. (3) **Weather** is data that measures 21 weather indicators, including temperature and humidity, every 10 minutes throughout 2020.

Evaluation metrics: As time-series forecasting is a complicated task, evaluating the prediction result only with MSE or MAE is insufficient. Thus, we include DTW and TDI as additional metrics, which can be interpreted as follows, to analyze the time-series forecasting task from multiple perspectives:

(1) **DTW:** We evaluate the difference of the overall shape between Y and $\hat{Y}$ via DTW. In particular, the more volatile the data is, the more emphasis is placed on using these metrics. Small DTW values mean the overall shapes of Y and $\hat{Y}$. However, one pitfall of DTW is that it ignores delays.
(2) **TDI:** TDI quantifies the disparity between the optimal paths of Y and $\hat{Y}$. Further details on TDI can be found in Appendix F. Utilizing TDI, a metric for assessing temporal distortion, is critical for precise predictions. Smaller TDI values indicate minimal prediction delays, aligning with the objectives of this paper.

4.2 Experimental results

In this subsection, we analyze the experimental results of six datasets by dividing them into a total of three evaluation metrics (MSE, DTW, and TDI) [21]. Table 3 introduces our experimental results for time-series forecasting with 6 datasets from various fields. We also report our time complexity and model usage in Appendix H.

Stocks: The past five years of stock data for AAPL, AMZN, GOOG, and MSFT exhibit both sharp rises and sharp falls, rendering them suitable for assessing accurate time series predictions across various aspects. PatchTST demonstrates specialization in MSE and surpasses other models based on differential equations and transformers. Autoformer exhibits reasonable DTW scores. Among the differential equation-based models, including CONTIME, the lowest TDI scores are observed. Notably, most baseline models specialize in a single metric, such as MSE or DTW, whereas CONTIME outperforms across all metrics with the lowest standard deviation.

Exchange: Table 3 presents the experimental findings for the Exchange dataset. Most models exhibit small MSE values; conversely, the NODE and NCDE models demonstrate superior TDI performance compared to others, indicating the efficacy of models based on differential equations in addressing prediction delays. Significantly, our suggested model, CONTIME, performs second-best with an MSE difference of only about 0.005 when compared to DLinear. In addition, CONTIME performs fairly well in TDI, indicating its capacity to efficiently reduce prediction delays.

Weather: In Table 3, our proposed model demonstrates superiority over all other models across all metrics. While DLinear and PatchTST exhibit reasonable performance in MSE and DTW, CONTIME consistently outperforms them. Specifically, CONTIME shows an average decrease of 0.168 in TDI compared to the second-best model across all prediction lengths. Furthermore, NODE and NCDE exhibit remarkable performance, reaffirming the efficiency of differential-equation-based models in mitigating prediction delays. Unlike baselines that excel in only one of the three metrics, CONTIME clearly demonstrates the best performance across all.

4.3 Visualization

Figure 4 provides a visualization of AAPL, AMZN, Exchange, and Weather forecasting results that prove CONTIME's outstanding performance over various aspects compared to state-of-the-art (SOTA) models, such as PatchTST and DLinear. For instance, focusing on the highlighted section in yellow in Figure 4.(a), the SOTA models (blue and green line) predict the opposite of the stock price fluctuation due to delays in the prediction. Specifically, unlike the ground truth, which starts to decline around August, 28th, 2023, state-of-the-art (SOTA) models fail to promptly recognize this change due to a delay. In contrast, CONTIME (red line) accurately captures the actual stock price (black line) in terms of shape, and timing. Figure 4.(b) illustrates the visualization results for the AMZN stock. Across the entire time, CONTIME closely matches the shape of the ground truth and makes predictions without any noticeable delays. Similarly, in Figure 4.(c), while the SOTA models exhibit similarities in terms of shape, their results are delayed; compared to the ground-truth with a high OT value on January 6, 2010, SOTA models predict a high OT value on January 11 due to a delay while CONTIME predicts on time. In Figure 4(d), most models exhibit a shape similar to the ground-truth. Notably, in the highlighted sections of our model (indicated in yellow), the fluctuations of T (degC) are predicted in detail. Conversely, the baseline models made predictions with a slight delay.

4.4 Sensitivity analysis & ablation study

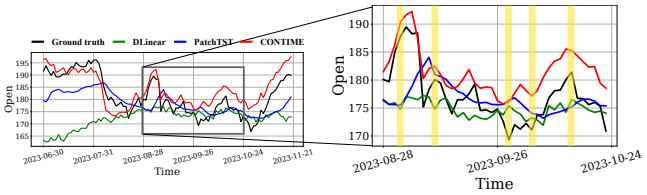
4.4.1 Ablation study on loss function. Table 4 summarizes the results of the ablation study for the loss functions applied to CONTIME. Three types of loss functions are utilized. L_{Task} is a plain MSE loss function to compare Y and $\hat{Y}$. L_{TDI} uses the TDI-based regularization proposed in [21] to address the prediction delay problem. This regularization minimizes TDI using Equation (28) of Appendix F. The Δt loss function aims to reduce the computed time-derivative explicitly. Through this ablation study, we evaluate the effectiveness of the time-derivative regularization process in alleviating prediction delays.

CONTIME (Only L_{task}), which trains with the task loss, exhibits reasonable performance in terms of MSE. However, it becomes apparent that it does not effectively learn in other aspects such as TDI and DTW. On the other hand, L_{TDI} and $L_{\Delta t}$, employing different types of regularization respectively, demonstrate exceptional performance in terms of TDI. Though CONTIME (L_{task} + L_{TDI}) performs well on TDI, it exhibits unstable performance in terms of MSE or DTW. However, CONTIME (L_{task} + $L_{\Delta t}$) excels the others in all three evaluation metrics, proving efficacy of our Δt loss.

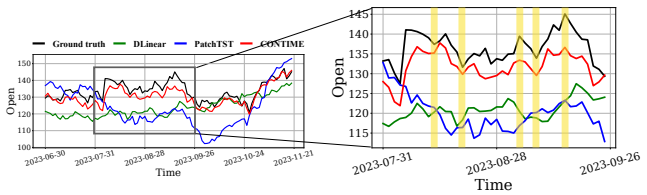
Relationship between L_{TDI} and $L_{\Delta t}$. The experimental results in Table 4 of the paper show that CONTIME ($L_{\Delta t}$) has superior TDI values compared to CONTIME (L_{TDI}). The TDI loss, calculated from the optimal DTW path A between $\hat{y}$ and y, exhibits good performance in TDI, which is slghtly inferior to our methodology. Unlike the TDI loss, solely focusing on aligning the timing of the DTW path, our methodology improves performance on both DTW

Table 3: Experimental results on 6 datasets. The best results are in bold and the second best are underlined.

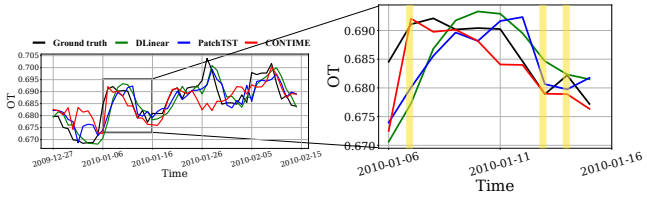
Datasets		APPL			AMZN			GOOG			MSFT			Exchange			Weather		
	P	TDI	DTW	MSE	TDI	DTW	MSE	TDI	DTW	MSE	TDI	DTW	MSE	TDI	DTW	MSE	TDI	DTW	MSE
DLinear	24	3.180	1.409	0.084	3.855	2.239	0.265	3.766	1.297	0.166	4.327	**1.430**	0.197	3.629	**0.533**	**0.044**	3.505	1.894	0.119
	36	5.106	1.940	0.187	5.396	2.726	0.372	4.835	2.229	0.199	6.103	2.385	0.319	5.638	0.781	0.065	5.944	1.436	0.144
	48	7.751	2.323	0.213	8.915	2.964	0.408	7.518	2.568	0.262	7.324	3.738	0.468	7.989	1.742	**0.084**	8.208	1.817	0.161
	60	10.84	2.907	0.258	9.252	3.017	0.347	12.39	2.848	0.294	12.10	4.247	0.492	11.01	2.304	0.107	10.16	**1.771**	**0.174**
NODE	24	3.739	4.330	0.168	3.063	3.275	0.397	3.684	6.399	1.298	4.596	3.389	0.359	2.085	2.855	0.525	2.758	4.262	0.336
	36	4.911	2.916	0.328	5.479	4.893	0.464	5.793	4.223	0.646	6.769	4.329	0.496	4.055	9.289	1.137	4.314	6.193	1.261
	48	7.482	4.203	0.535	7.149	6.436	0.813	7.795	5.112	0.794	8.868	4.656	0.504	6.104	6.028	1.100	6.827	6.294	1.261
	60	8.702	10.25	1.149	8.954	6.333	1.033	9.513	5.648	0.874	10.72	7.963	0.618	9.822	6.621	1.056	10.54	7.652	1.506
NCDE	24	5.039	4.555	0.227	2.984	5.493	0.261	3.719	4.601	0.517	4.842	2.809	0.445	1.874	3.689	0.576	2.489	5.609	0.854
	36	6.651	3.199	0.462	5.829	4.022	0.335	4.946	3.541	0.582	6.687	2.902	0.628	4.184	8.137	0.542	4.661	4.059	0.799
	48	7.303	4.028	0.440	7.113	5.817	0.711	8.132	6.161	0.756	9.018	4.327	0.690	6.012	7.957	0.874	6.922	4.682	0.783
	60	11.47	3.882	0.459	9.041	7.936	1.352	10.02	5.637	0.771	12.35	5.221	0.766	8.105	6.516	0.604	9.900	5.882	0.989
Autoformer	24	3.085	1.551	0.150	3.576	**1.485**	0.174	3.289	**1.239**	0.167	4.222	1.690	0.246	3.158	1.120	0.098	2.586	1.938	0.327
	36	6.561	1.882	0.171	5.541	2.032	0.203	5.782	2.210	0.199	**5.111**	2.474	0.288	4.724	1.516	0.125	4.662	2.393	9.349
	48	9.814	2.307	0.170	6.941	2.388	0.219	7.606	2.943	0.289	7.335	2.810	0.287	8.245	1.760	0.129	6.955	2.855	0.415
	60	13.82	2.651	0.188	9.414	**2.723**	0.275	10.80	3.248	0.279	12.14	3.668	0.380	10.53	**2.026**	0.139	9.944	2.854	0.415
FEDformer	24	3.417	1.396	0.129	3.108	1.764	0.232	3.154	1.587	0.204	4.335	1.754	0.243	3.311	0.887	0.079	2.872	1.506	0.215
	36	6.335	1.826	0.149	5.878	2.201	0.249	5.311	2.203	0.215	6.794	2.505	0.304	5.638	1.079	0.085	5.108	1.801	0.313
	48	12.64	1.932	0.135	7.664	2.691	0.289	8.489	2.312	0.225	8.203	2.891	0.308	7.952	1.692	0.108	6.342	2.053	0.226
	60	16.39	2.642	0.204	12.84	2.980	0.354	12.13	2.785	0.244	12.76	**3.209**	0.321	10.68	2.714	0.128	9.495	2.083	0.199
PatchTST	24	3.166	1.253	0.091	3.969	1.574	0.177	3.706	1.554	**0.165**	4.222	1.529	0.215	3.658	0.903	0.056	3.089	1.796	0.119
	36	5.358	**1.417**	0.118	6.679	**1.733**	**0.168**	4.882	2.766	0.191	6.388	**2.154**	**0.234**	5.603	**0.766**	0.078	4.849	1.128	0.149
	48	7.984	**1.809**	0.130	8.706	2.521	0.220	7.840	2.342	0.203	10.49	3.075	0.356	8.083	1.701	0.099	6.687	**1.473**	0.181
	60	11.00	2.626	0.202	12.24	3.475	0.275	10.64	**2.673**	0.244	14.09	3.883	0.693	11.32	2.210	**0.106**	10.40	1.988	0.229
CONTIME	24	**2.378**	**1.114**	**0.074**	**2.866**	1.529	**0.167**	**3.052**	1.541	**0.165**	**4.218**	1.528	**0.184**	**1.761**	0.884	0.049	**2.254**	**1.023**	**0.117**
	36	**4.807**	1.541	**0.089**	**5.275**	1.881	0.193	**4.712**	**2.189**	**0.189**	5.371	2.334	0.256	**3.488**	1.221	**0.063**	**4.120**	**1.390**	**0.136**
	48	**7.300**	1.912	**0.114**	**6.844**	**2.300**	**0.209**	**7.364**	**2.297**	**0.188**	**7.296**	**2.755**	**0.262**	**5.366**	**1.683**	0.097	**6.226**	1.805	**0.159**
	60	**7.932**	**2.625**	**0.147**	**8.885**	2.873	**0.239**	**9.271**	2.741	**0.210**	**11.83**	3.261	**0.292**	**7.452**	2.139	0.125	**9.366**	2.121	**0.174**



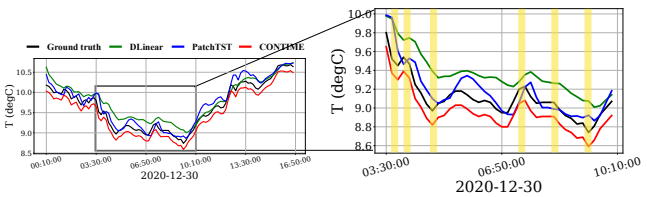
(a) AAPL from June 30th,2023 to November 21th,2023



(b) AMZN from June 30th,2023 to November 21th,2023



(c) Exchange from December 27th,2009 to Feburary 15th, 2010



(d) Weather from 6 hours at December 30th, 2020

Figure 4: Forecasting visualization on 4 datasets. More figures are in Appendix G

(shape) and TDI (timing) through the explicit gradient modeling at each time t.

4.4.2 Sensitivity to α, β. In Figure 5, we discern the impact of our Δt loss on TDI, DTW, and MSE with the sensitivity curve w.r.t β (varying from 0.1 to 0.001), compared to the top 3 baselines in each metric. Across all settings, CONTIME consistently outperforms the baselines in terms of DTW and TDI, demonstrating the efficacy of our model. Regarding MSE, CONTIME exhibits reasonable performance across all settings. These results indicate stable performance of our model trained with Δt loss, thereby leading to more effective elimination of prediction delays and it also shows stable performance in DTW and MSE.

4.4.3 Additional experiments on other 3 datasets. To evaluate the performance of our model in different domains, we evaluate the model on ILL (National Disease) and ETTh1 and ETTh2. Compared to PatchTST and DLinear, we show slightly better performance in

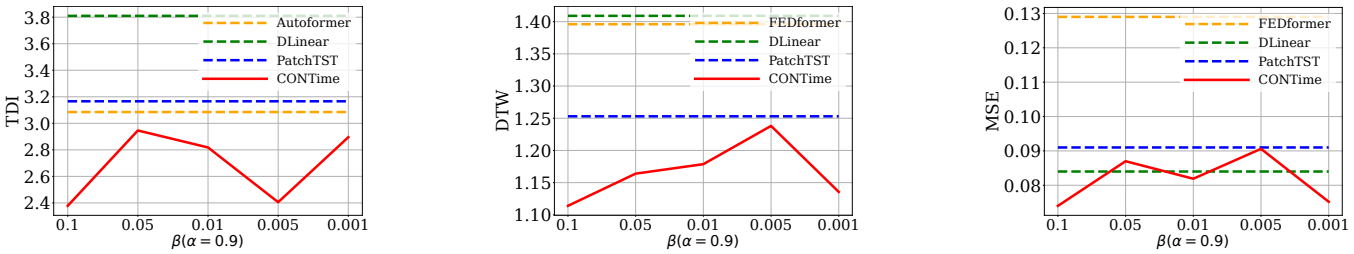


Figure 5: Sensitivity to α, β in AAPL

Table 4: Comparison on TDI loss and Δt loss

Models	P	AMZN			Exchange		
		TDI	DTW	MSE	TDI	DTW	MSE
CONTIME (Only L_{Task})	24	3.481	2.122	0.202	1.887	1.343	0.105
	36	5.854	1.947	0.194	3.825	1.970	0.113
	48	7.775	2.255	0.214	6.301	2.836	0.242
	60	11.99	2.695	0.221	8.716	1.859	**0.102**
CONTIME ($L_{Task} + L_{TDI}$)	24	**2.816**	3.263	1.269	1.782	3.264	0.325
	36	**4.715**	2.936	0.355	3.557	4.992	0.843
	48	6.923	4.341	0.535	**5.229**	3.596	0.852
	60	8.924	5.456	0.826	7.522	2.857	0.257
CONTIME ($L_{Task} + L_{\Delta t}$)	24	2.866	**1.529**	**0.167**	**1.761**	**0.884**	**0.049**
	36	5.275	**1.881**	**0.193**	**3.488**	**1.221**	**0.063**
	48	**6.844**	**2.300**	**0.209**	5.366	**1.683**	**0.097**
	60	**8.885**	**2.873**	**0.239**	**7.452**	**2.139**	0.125

Table 5: Additional experimental results on 3 datasets

Datasets	P	CONTIME			PatchTST			DLinear		
		TDI	DTW	MSE	TDI	DTW	MSE	TDI	DTW	MSE
ILL	24	1.552	4.122	1.357	1.744	4.311	1.449	2.013	4.278	1.980
	36	1.739	5.108	1.501	1.927	5.258	1.541	3.124	6.018	1.873
	48	1.042	4.916	0.778	1.224	5.986	1.673	4.171	9.701	2.296
	60	2.404	6.626	1.688	2.421	8.376	1.549	5.281	8.240	2.334
ETTh1	24	1.199	2.235	0.445	1.921	2.204	0.329	1.709	2.907	0.398
	36	1.443	2.389	0.441	2.205	2.426	0.364	2.231	3.294	0.388
	48	1.872	2.832	0.437	2.475	2.758	0.338	2.621	3.907	0.378
	60	2.201	4.008	0.453	2.674	4.071	0.354	2.988	4.227	0.386
ETTh2	24	1.952	1.571	0.177	2.162	2.319	0.187	2.266	2.359	0.181
	36	2.231	1.957	0.199	2.479	2.675	0.202	2.509	2.883	0.217
	48	2.417	2.087	0.219	2.537	2.014	0.246	3.120	2.912	0.228
	60	3.370	2.481	0.233	3.485	2.528	0.271	3.839	3.387	0.263

terms of MSE (value), but excellent performance in TDI and DTW metrics that measure timing and shape.

4.4.4 To deal with distribution shift problem. The most difficult part of predicting time series benchmarked datasets is the distribution shift problem [18]. In this paper, we use the shift method as in NLinear [33] to solve this situation.

$$\begin{aligned} \texttt{shift term} &= \hat{Y}(0) - \mathbf{X}(T), \\ \hat{Y} &= \hat{Y} + \texttt{shift term}, \end{aligned} \tag{14}$$

, where $\mathbf{X}(T)$ refers to the last observations of the input sequences. We calculate the difference between the last observation $\mathbf{X}(T)$ and the first prediction value $\hat{Y}(0)$. By simply adding `shift term` to the forecasting result $\hat{Y}$, We can reduce the distribution shift problem,

Table 6: Comparison between CONTIME and CONTIME (shift term)

Models	P	AAPL			Weather		
		TDI	DTW	MSE	TDI	DTW	MSE
CONTIME	24	2.378	1.114	0.074	2.323	1.115	0.129
	36	4.807	1.541	0.089	4.094	1.563	0.154
	48	7.338	1.941	0.112	6.273	2.043	0.183
	60	7.932	2.625	0.147	9.243	2.253	0.194
CONTIME (`shift term`)	24	2.917	1.111	0.074	2.254	1.023	0.117
	36	4.802	1.602	0.094	4.120	1.390	0.136
	48	7.300	1.912	0.114	6.226	1.805	0.159
	60	7.964	2.625	0.148	9.366	2.121	0.174

5 CONCLUSIONS

This paper suggests yet another view on the time series forecasting research in other perspectives. To mitigate the prediction latency in time series forecasting, we suggest CONTIME, a unique architecture that enables the explicit supervision of the time-derivative of observations in the continuous time domain by continuously generalizing the bi-directional GRU. With this distinctive architecture, we effectively addressed the prediction delay problem, which has long been an obstacle of time series forecasting. By applying the continuous bi-directional GRU and Δt loss to naturally supervise the time-derivative, CONTIME alleviates the prediction delay problem. We quantify these phenomena by measuring not just MSE but also TDI and DTW as evaluation metrics. As a result, we exhibit superior overall performance when compared to 6 state-of-the-art baselines for 6 datasets from various fields.

ACKNOWLEDGEMENTS

This work was partly supported by the Korea Advanced Institute of Science and Technology (KAIST) grant funded by the Korea government (MSIT) (No. G04240001, Physics-inspired Deep Learning, 10%), and Institute for Information & Communications Technology Planning & Evaluation (IITP) grants funded by the Korea government (MSIT) (No. RS-2020-II201361, Artificial Intelligence Graduate School Program (Yonsei University),5%), and (No.RS-2022-II220113, Developing a Sustainable Collaborative Multi-modal Lifelong Learning Framework,80%) and Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No.RS-2019-II190075 Artificial Intelligence Graduate School Program(KAIST), 5%)

REFERENCES

[1] Robert J Abrahart, Alison J Heppenstall, and Linda M See. 2007. Timing error correction procedure applied to neural network rainfall—runoff modelling. *Hydrological sciences journal* 52, 3 (2007), 414–431.

[2] Inc. Amazon.com. 2024. Amazon Stock. https://finance.yahoo.com/quote/AMZN/history?p=AMZN.

[3] Edward De Brouwer, Jaak Simm, Adam Arany, and Yves Moreau. 2019. GRU-ODE-Bayes: Continuous modeling of sporadically-observed time series. arXiv:1905.12374 [cs.LG]

[4] Jian Cao, Zhi Li, and Jian Li. 2019. Financial time series forecasting model based on CEEMDAN and LSTM. *Physica A: Statistical mechanics and its applications* 519 (2019), 127–139.

[5] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. 2018. Neural ordinary differential equations. *Advances in neural information processing systems* 31 (2018).

[6] Andrew J Conway, Keith P Macpherson, and John C Brown. 1998. Delayed time series predictions with neural networks. *Neurocomputing* 18, 1-3 (1998), 81–89.

[7] Microsoft Corporation. 2024. MSFT Stock. https://finance.yahoo.com/quote/MSFT/history?p=MSFT.

[8] Marco Cuturi and Mathieu Blondel. 2017. Soft-dtw: a differentiable loss function for time-series. In *International conference on machine learning*. PMLR, 894–903.

[9] NJ De Vos and THM Rientjes. 2005. Constraints of artificial neural networks for rainfall-runoff modelling: trade-offs in hydrological state representation and model evaluation. *Hydrology and earth system sciences* 9, 1/2 (2005), 111–126.

[10] Pradnya Dixit, Shreenivas Londhe, and Yogesh Dandawate. 2015. Removing prediction lag in wave height forecasting using Neuro-Wavelet modeling technique. *Ocean Engineering* 93 (2015), 74–83.

[11] Laura Frías-Paredes, Fermín Mallor, Martín Gastón-Romeo, and Teresa León. 2017. Assessing energy forecasting inaccuracy by simultaneously considering temporal and absolute errors. *Energy Conversion and Management* 142 (2017), 533–546.

[12] Yang Han, Ying Tian, Liangliang Yu, and Yuning Gao. 2023. Economic system forecasting based on temporal fusion transformers: Multi-dimensional evaluation and cross-model comparative analysis. *Neurocomputing* 552 (2023), 126500.

[13] Apple Inc. 2024. Apple Stock. https://finance.yahoo.com/quote/AAPL/history?p=AAPL.

[14] Alphabet Inc. 2024. Google Stock. https://finance.yahoo.com/quote/GOOG/history?p=GOOG.

[15] Sheo Yon Jhin, Jaehoon Lee, Minju Jo, Seungji Kook, Jinsung Jeon, Jihyeon Hyeong, Jayoung Kim, and Noseong Park. 2022. Exit: Extrapolation and interpolation-based neural controlled differential equations for time-series classification and forecasting. In *Proceedings of the ACM Web Conference 2022*. 3102–3112.

[16] Sheo Yon Jhin, Heejoo Shin, Sujie Kim, Seoyoung Hong, Minju Jo, Solhee Park, Noseong Park, Seungbeom Lee, Hwiyoung Maeng, and Seungmin Jeon. 2023. Attentive neural controlled differential equations for time-series classification and forecasting. *Knowledge and Information Systems* (2023), 1–31.

[17] Patrick Kidger, James Morrill, James Foster, and Terry Lyons. 2020. Neural controlled differential equations for irregular time series. *Advances in Neural Information Processing Systems* 33 (2020), 6696–6707.

[18] Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi, and Jaegul Choo. 2021. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International Conference on Learning Representations*.

[19] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451* (2020).

[20] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. 2018. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*. 95–104.

[21] Vincent Le Guen and Nicolas Thome. 2019. Shape and time distortion loss for training deep time series forecasting models. *Advances in neural information processing systems* 32 (2019).

[22] Bryan Lim and Stefan Zohren. 2021. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A* 379, 2194 (2021), 20200209.

[23] Shizhan Liu, Hang Yu, Cong Liao, Jianguo Li, Weiyao Lin, Alex X Liu, and Schahram Dustdar. 2021. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *International conference on learning representations*.

[24] Qingqing Long, Zheng Fang, Chen Fang, Chong Chen, Pengfei Wang, and Yuanchun Zhou. 2024. Unveiling Delay Effects in Traffic Forecasting: A Perspective from Spatial-Temporal Delay Differential Equations. arXiv:2402.01231 [cs.LG]

[25] Terry Lyons, M. Caruana, and T. Lévy. 2004. *Differential Equations Driven by Rough Paths*. Springer. École D'Eté de Probabilités de Saint-Flour XXXIV - 2004.

[26] James Morrill, Patrick Kidger, Lingyi Yang, and Terry Lyons. 2021. Neural controlled differential equations for online prediction tasks. *arXiv preprint arXiv:2106.11028* (2021).

[27] Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. 2023. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. In *International Conference on Learning Representations*.

[28] H. Sakoe and S. Chiba. 1978. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 26, 1 (1978), 43–49. https://doi.org/10.1109/TASSP.1978.1163055

[29] Afan Galih Salman, Bayu Kanigoro, and Yaya Heryadi. 2015. Weather forecasting using deep learning techniques. In *2015 international conference on advanced computer science and information systems (ICACSIS)*. Ieee, 281–285.

[30] Loïc Vallance, Bruno Charbonnier, Nicolas Paul, Stéphanie Dubost, and Philippe Blanc. 2017. Towards a standardized procedure to assess solar forecast accuracy: A new ramp and time alignment metric. *Solar Energy* 150 (2017), 408–422.

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention Is All You Need. arXiv:1706.03762 [cs.CL]

[32] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. 2021. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems* 34 (2021).

[33] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. 2023. Are transformers effective for time series forecasting?. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 37. 11121–11128.

[34] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35.

[35] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. 2022. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International Conference on Machine Learning*. PMLR, 27268–27286.

A DERIVATIVES OF $\mathbf{z}(t), \mathbf{g}(t), \mathbf{r}(t)$

time-derivative of $\mathbf{z}(t)$*:* The *continuous* update gate is written as $\mathbf{z}(t) = \sigma\big(\mathbf{W}_z\mathbf{x}(t) + \mathbf{U}_z\mathbf{h}(t-\tau) + \mathbf{b}_z\big) = \sigma(\mathbf{A}(t, t-\tau))$, and its derivative, denoted $\frac{d\mathbf{z}(t)}{dt}$, is as follows:

$$\frac{d\mathbf{z}(t)}{dt} = \sigma\big(\mathbf{A}(t,t-\tau)\big)\big(1-\sigma(\mathbf{A}(t,t-\tau))\big)\frac{d\mathbf{A}(t,t-\tau)}{dt}, \tag{15}$$

where $\mathbf{A}(t,t-\tau) = \mathbf{W}_z\mathbf{x}(t) + \mathbf{U}_z\mathbf{h}(t-\tau) + \mathbf{b}_z$, and $\frac{d\mathbf{A}(t,t-\tau)}{dt} = \mathbf{W}_z\frac{d\mathbf{x}(t)}{dt} + \mathbf{U}_z\frac{d\mathbf{h}(t-\tau)}{dt}$.

time-derivative of $\mathbf{g}(t)$*:* The *continuous* update vector has the form of $\mathbf{g}(t) = \phi\big(\mathbf{W}_g\mathbf{x}(t) + \mathbf{U}_g\big(\mathbf{r}(t)\odot\mathbf{h}(t-\tau)\big) + \mathbf{b}_g\big) = \phi(\mathbf{B}(t,t-\tau)$, and its derivative, $\frac{d\mathbf{g}(t)}{dt}$, can be calculate as follows:

$$\frac{d\mathbf{g}(t)}{dt} = \big(1-\phi^2(\mathbf{B}(t,t-\tau)\big)\frac{d\mathbf{B}(t,t-\tau)}{dt}, \tag{16}$$

where $\mathbf{B}(t,t-\tau) = \mathbf{W}_g\mathbf{x}(t) + \mathbf{U}_g\big(\mathbf{r}(t)\odot\mathbf{h}(t-\tau)\big) + \mathbf{b}_g$, and $\frac{d\mathbf{B}(t,t-\tau)}{dt} = \mathbf{W}_g\frac{d\mathbf{x}(t)}{dt} + \mathbf{U}_g\frac{d\mathbf{r}(t)}{dt}\mathbf{h}(t-\tau) + \mathbf{U}_g\mathbf{r}(t)\frac{d\mathbf{h}(t-\tau)}{dt}$.

time-derivative of $\mathbf{r}(t)$*:* The *continuous* reset gate is defined as $\mathbf{r}(t) = \sigma\big(\mathbf{W}_r\mathbf{x}(t) + \mathbf{U}_r\mathbf{h}(t-\tau) + \mathbf{b}_r\big)$, and its derivative $\frac{d\mathbf{r}(t)}{dt}$ is derived as follows:

$$\frac{d\mathbf{r}(t)}{dt} = \sigma\big(\mathbf{C}(t)\big)\big(1-\sigma(\mathbf{C}(t,t-\tau))\big)\frac{d\mathbf{C}(t,t-\tau)}{dt}, \tag{17}$$

where $\mathbf{C}(t,t-\tau) = \mathbf{W}_r\mathbf{x}(t) + \mathbf{U}_r\mathbf{h}(t-\tau) + \mathbf{b}_r$, and $\frac{d\mathbf{C}(t,t-\tau)}{dt} = \mathbf{W}_r\frac{d\mathbf{x}(t)}{dt} + \mathbf{U}_r\frac{d\mathbf{h}(t-\tau)}{dt}$.

B PROOF OF EQUATION.9

First, let $\mathbf{z}(t)$, $\mathbf{g}(t)$, and $\mathbf{r}(t)$ be the update gate, the update vector, and the reset gate of GRU:

$$\begin{aligned}\mathbf{z}(t) &= \sigma\big(\mathbf{W}_z\mathbf{x}(t) + \mathbf{U}_z\mathbf{h}(t-\tau) + \mathbf{b}_z\big),\\ \mathbf{g}(t) &= \phi\big(\mathbf{W}_g\mathbf{x}(t) + \mathbf{U}_g\big(\mathbf{r}(t)\odot\mathbf{h}(t-\tau)\big) + \mathbf{b}_g\big),\\ \mathbf{r}(t) &= \sigma\big(\mathbf{W}_r\mathbf{x}(t) + \mathbf{U}_r\mathbf{h}(t-\tau) + \mathbf{b}_r\big),\end{aligned} \tag{18}$$

To simplify the equations, we will define them as follows:

$$\begin{aligned}\mathbf{z}(t) &= \sigma\big(\mathbf{A}(t,t-\tau)\big),\\ \mathbf{g}(t) &= \phi\big(\mathbf{B}(t,t-\tau)\big),\\ \mathbf{r}(t) &= \sigma\big(\mathbf{C}(t,t-\tau)\big),\end{aligned} \tag{19}$$

where $\mathbf{A}(t,t-\tau) = \mathbf{W}_z\mathbf{x}(t) + \mathbf{U}_z\mathbf{h}(t-\tau) + \mathbf{b}_z$, $\mathbf{B}(t,t-\tau) = \mathbf{W}_h\mathbf{x}(t) + \mathbf{U}_h\big(\mathbf{r}(t)\odot\mathbf{h}(t-\tau)\big) + \mathbf{b}_h$, and $\mathbf{C}(t,t-\tau) = \mathbf{W}_r\mathbf{x}(t) + \mathbf{U}_r\mathbf{h}(t-\tau) + \mathbf{b}_r$. The derivatives of $\mathbf{z}(t)$, $\mathbf{g}(t)$, and $\mathbf{r}(t)$ are defined as follows:

$$\begin{aligned}\frac{d\mathbf{z}(t)}{dt} &= \sigma(\mathbf{A}(t,t-\tau))(1-\sigma(\mathbf{A}(t,t-\tau)))\frac{d\mathbf{A}(t,t-\tau)}{dt}\\ \frac{d\mathbf{g}(t)}{dt} &= (1-\phi^2(\mathbf{B}(t,t-\tau)))\frac{d\mathbf{B}(t,t-\tau)}{dt}\\ \frac{d\mathbf{r}(t)}{dt} &= \sigma(\mathbf{C}(t,t-\tau))(1-\sigma(\mathbf{C}(t,t-\tau)))\frac{d\mathbf{C}(t,t-\tau)}{dt}\end{aligned} \tag{20}$$

Lastly, the hidden state $\mathbf{h}(t)$ of GRU is written as follows:

$$\mathbf{h}(t) = \mathbf{z}(t)\odot\mathbf{h}(t-\tau) + (1-\mathbf{z}(t))\odot\mathbf{g}(t). \tag{21}$$

The derivative of the hidden state $\mathbf{h}(t)$ is defined by the chain rule as follows:

$$\begin{aligned}\frac{d\mathbf{h}(t)}{dt} &= \frac{d\mathbf{z}(t)}{dt}\odot\mathbf{h}(t-\tau) + \mathbf{z}(t)\odot\frac{d\mathbf{h}(t-\tau)}{dt}\\ &\quad - \frac{d\mathbf{z}(t)}{dt}\odot\mathbf{g}(t) + (1-\mathbf{z}(t))\odot\frac{d\mathbf{g}(t)}{dt},\\ &= \frac{d\mathbf{z}(t)}{dt}\odot\big(\mathbf{h}(t-\tau) - \mathbf{g}(t)\big)\\ &\quad + \mathbf{z}(t)\odot\Big(\frac{d\mathbf{h}(t-\tau)}{dt} - \frac{d\mathbf{g}(t)}{dt}\Big) + \frac{d\mathbf{g}(t)}{dt},\\ &= \frac{d\mathbf{z}(t)}{dt}\odot\zeta(t,t-\tau) + \mathbf{z}(t)\odot\frac{d\zeta(t,t-\tau)}{dt} + \frac{d\mathbf{g}(t)}{dt},\end{aligned} \tag{22}$$

where $\zeta(t,t-\tau) = \mathbf{h}(t-\tau) - \mathbf{g}(t)$. So, we can rewrite $\frac{d\mathbf{h}(t)}{dt}$ as follows:

$$\frac{d\mathbf{h}(t)}{dt} = \frac{d(\mathbf{z}(t)\odot\zeta(t,t-\tau))}{dt} + \frac{d\mathbf{g}(t)}{dt} \tag{23}$$

C DETAILED DESCRIPTIONS OF INTERPOLATION METHODS

In Section. 3.2, we calculated $\frac{dX(t)}{dt}$ by using Cubic Hermite spline method. In this section, we describe why we choose the Cubic Hermite spline method not the Natural cubic spline which creates the continuous path $X(t)$. There are two interpolation methods that create continuous path $X(t)$, Natural cubic splines and Cubic Hermite splines.

Natural cubic splines: Natural cubic splines used in Neural CDE [17] require the entire time series to be used as a control signal. That is, a change in future time step may interfere past time steps, thereby making interpolated result unreliable. In other words, it is an interpolation method that cannot be used in online prediction.

Cubic Hermite splines: This approach mitigates the discontinuity of linear control while maintaining the same online properties by joining adjacent viewpoints with cubic splines that use additional degrees of freedom to smooth out gradient discontinuities. This results in faster integration times than linear control. The main difference from natural cubic splines is that Cubic Hermite splines solve a single equation for each $[i, i+1)$ piece independently. As a result, it changes more quickly than the natural cubic spline and therefore has a slower integration time than the natural cubic spline [26].

Due to the above two differences, we believe that the Cubic Hermite spline is more suitable for real-world time series forecasting, so we use this method to create a continuous path $X(t)$.

D DATASETS

The datasets used in our experiments are publicly available and can be downloaded at the following locations:

(1) AAPL: https://finance.yahoo.com/quote/AAPL/history?p=AAPL,
(2) AMZN: https://finance.yahoo.com/quote/AMZN/history?p=AMZN,
(3) MSFT: https://finance.yahoo.com/quote/MSFT/history?p=MSFT,
(4) GOOG: https://finance.yahoo.com/quote/GOOG/history?p=GOOG,
(5) Exchange: https://drive.google.com/drive/folders/1ZOYpTUa82_jCcxIdTmyr0LXQfvaM9vIy,
(6) Weather: https://drive.google.com/drive/folders/1ZOYpTUa82_jCcxIdTmyr0LXQfvaM9vIy,

We split the entire dataset into training/validating/testing parts. The first 70% of the data is used as training, 10% is used for validating, and the last 20% is used for testing.

E HYPERPARAMETER

All of the models follow the same experimental setup with prediction horizon $P \in \{24, 36, 48, 60\}$ for all 6 datasets.

E.1 Hyperparameter for CONTIME

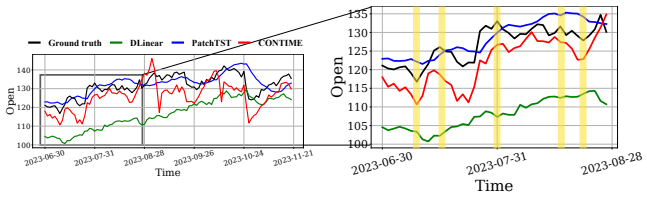
In Table 10, we showed our best hyperparameter for all 6 datasets.

F HOW TO CALCULATE TDI

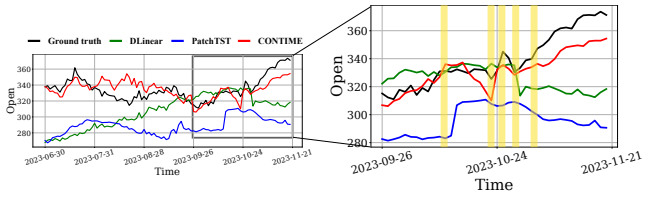
We adopted TDI loss calculation for time-series sequence from [21]. The calculation below is applied to each feature of data X defined in Section. 3.2.

We define $\mathbf{A}$ as a binary warping path of prediction length P, i.e. $\mathbf{A} \subset \{0,1\}^{P\times P}$, with $A_{h,j} = 1$ if $\hat{Y}_h$ is associated with Y_j and otherwise 0, where h, j are a time point of each sequence. $\Delta(\hat{Y}, Y) := \big[(\hat{Y}_h - Y_j)^2\big]_{h,j}$, which means measuring dissimilarity of two sequences by euclidean distance. We calculate TDI from the optimal path matrix $\mathbf{A}^\star$ of DTW as follows:

$$DTW(\hat{Y}_i, Y_i) = \min_{\mathbf{A}\in A_{P,P}} \langle \mathbf{A}, \Delta(\hat{Y}_i, Y_i)\rangle \tag{24}$$



(a) GOOG



(b) MSFT

Figure 6: Forecasting visualization on 2 datasets

Table 9: Model usage

Models	AAPL	AMZN	GOOG	MSFT	Exchange	Weather
DLinear	179.1	179.1	179.1	179.1	173.2	180.7
NODE	26.05	26.05	26.05	26.05	27.37	35.39
NCDE	7.027	7.027	7.027	7.027	10.78	50.69
Autoformer	112.4	112.4	112.4	112.4	1,243	1,244
FEDformer	496.7	496.7	496.7	496.7	635.1	804.1
PatchTST	102.3	102.3	102.3	102.3	116.8	625.2
CONTIME	198.7	198.7	198.7	198.7	117.7	131.8

Table 7: Best hyperparameter for CONTIME

Hyperparameter	P	AAPL	AMZN	GOOG	MSFT	Exchange	Weather
λ	24	0.005	0.005	0.005	0.005	0.01	0.001
	36	0.005	0.001	0.005	0.001	0.005	0.001
	48	0.005	0.005	0.005	0.005	0.005	0.001
	60	0.005	0.005	0.005	0.005	0.005	0.001
α	24	0.9	0.8	0.8	0.9	0.9	0.9
	36	0.8	0.8	0.8	0.8	0.9	0.9
	48	0.9	0.9	0.8	0.9	0.9	0.9
	60	0.9	0.9	0.8	1.0	0.9	0.9
β	24	0.1	0.1	0.05	0.001	0.1	0.001
	36	0.01	0.01	0.01	0.001	0.1	0.001
	48	0.001	0.1	0.05	0.001	0.1	0.001
	60	0.1	0.005	0.1	0.001	0.01	0.001
T	24	144	104	144	144	60	60
	36	144	144	144	104	60	60
	48	144	144	144	104	60	60
	60	144	104	144	144	60	60

$$\mathbf{A}^\star := \arg\min_{\mathbf{A}\in A_{P,P}} \langle \mathbf{A}, \Delta(\hat{Y}_i, Y_i)\rangle \tag{25}$$

$$TDI(\hat{Y}_i, Y_i) := \langle \mathbf{A}^\star, \Omega\rangle = \left\langle \arg\min_{\mathbf{A}\in A_{p,p}} \langle \mathbf{A}, \Delta(\hat{Y}_i, Y_i)\rangle, \Omega \right\rangle \tag{26}$$

where Ω is a square matrix of size $P \times P$ penalizing each element Y_h being associated to an $\hat{Y}_j$, for $h \neq j$: e.g. $\Omega(h, j) = \frac{(h-j)^2}{P^2}$.

To make TDI differentiable, we approximate $\mathbf{A}^\star$ with $\mathbf{A}^\star_\gamma$ using the fact that $\mathbf{A}^\star = \nabla_\Delta DTW(\hat{Y}_i, Y_i)$:

$$\mathbf{A}^\star_\gamma := \nabla_\Delta DTW_\gamma(\hat{Y}_i, Y_i) = \frac{1}{Z}\sum_{\mathbf{A}\in A_{P,P}} \mathbf{A}\exp -\frac{\langle \mathbf{A}, \Delta(\hat{Y}_i, Y_i)\rangle}{\gamma} \tag{27}$$

where $Z = \exp -\frac{\langle \mathbf{A}, \Delta(\hat{Y}_i, Y_i)\rangle}{\gamma}$. The resulting TDI loss is:

$$L_{TDI} := TDI(\hat{Y}_i, Y_i) := \langle \mathbf{A}^\star_\gamma, \Omega\rangle \tag{28}$$

G FORECASTING VISUALIZATION

In this section, we additionally visualize forecasting results on all 6 datasets.

H COMPUTATIONAL TIME AND MODEL USAGE

In this section, we report computational time of our model and model usage for all 6 datasets.

Table 8: Computational time

Models	AAPL	AMZN	GOOG	MSFT	Exchange	Weather
DLinear	6.436	5.653	6.289	6.371	41.25	240.9
NODE	14.08	13.85	13.67	13.81	43.40	9.935
NCDE	80.68	82.72	77.69	36.52	34.94	86.29
Autoformer	8.414	7.949	8.704	8.191	31.99	350.2
FEDformer	14.58	10.03	9.598	11.20	39.18	312.83
PatchTST	2.495	2.240	2.751	2.236	22.62	289.6
CONTIME	23.25	21.79	23.65	32.37	33.68	29.93

I HYPERPARAMETER

All of the models follow the same experimental setup with prediction horizon $P \in \{24, 36, 48, 60\}$ for all 6 datasets.

I.1 Hyperparameter for baselines

For the best outcome of baselines and our method, we conduct a hyperparameter search for them based on the recommended hyperparameter set from each paper. Considered hyperparameter sets are as follows:

Stocks:

(1) DLinear : We train for 100 epochs with a learning rate λ in $\{0.01, 0.05, 0.001, 0.005, 0.0001\}$. Input sequence length T in $\{96, 104, 144\}$
(2) Differential-equation based models: For Neural ODE and Neural CDE, we train for 100 epochs with a learning rate λ in $\{0.01, 0.05, 0.001, 0.005, 0.0001\}$. Hidden size in $\{39, 49, 59\}$.
(3) Transformer-based models: For Autoformer and FEDformer, we train for 50 epochs with a learning rate λ in $\{0.01, 0.05, 0.001, 0.005, 0.0001\}$. Input sequence length T in $\{96, 104, 144\}$
(4) PatchTST: We train for 50 epochs with a learning rate λ in $\{0.01, 0.05, 0.001, 0.005, 0.0001\}$. Input sequence length T in $\{96, 104, 144\}$. Other settings follow the same experimental settings in the baseline.

Table 10: Best hyperparameter for CONTIME

Hyperparameter	P	AAPL	AMZN	GOOG	MSFT	Exchange	Weather
λ	24	0.005	0.005	0.005	0.005	0.01	0.001
	36	0.005	0.001	0.005	0.001	0.005	0.001
	48	0.005	0.005	0.005	0.005	0.005	0.001
	60	0.005	0.005	0.005	0.005	0.005	0.001
α	24	0.9	0.8	0.8	0.9	0.9	0.9
	36	0.8	0.8	0.8	0.8	0.9	0.9
	48	0.9	0.9	0.8	0.9	0.9	0.9
	60	0.9	0.9	0.8	1.0	0.9	0.9
β	24	0.1	0.1	0.05	0.001	0.1	0.001
	36	0.01	0.01	0.01	0.001	0.1	0.001
	48	0.001	0.1	0.05	0.001	0.1	0.001
	60	0.1	0.005	0.1	0.001	0.01	0.001
T	24	144	104	144	144	60	60
	36	144	144	144	104	60	60
	48	144	144	144	104	60	60
	60	144	104	144	144	60	60

Exchange:

(1) DLinear : We train for 100 epochs with a learning rate λ in $\{0.01, 0.05, 0.001, 0.005, 0.0001\}$. Input sequence length T in $\{60, 72, 96\}$.
(2) Differential-equation based models: For Neural ODE and Neural CDE, we train for 100 epochs with a learning rate λ in $\{0.01, 0.05, 0.001, 0.005, 0.0001\}$. Hidden size in $\{39, 49, 59\}$. Input sequence length T in $\{60, 72, 96\}$.
(3) Transformer-based models: For Autoformer and FEDformer, we train for 50 epochs with a learning rate λ in $\{0.01, 0.05, 0.001, 0.005, 0.0001\}$. Input sequence length T in $\{60, 72, 96\}$. Other settings follow the same experimental settings in the baseline.
(4) PatchTST: We train for 50 epochs with a learning rate λ in $\{0.01, 0.05, 0.001, 0.005, 0.0001\}$. Input sequence length T in $\{60, 72, 96\}$. Other settings follow the same experimental settings in the baseline.

Weather:

(1) DLinear : We train for 100 epochs with a learning rate λ in $\{0.01, 0.05, 0.001, 0.005, 0.0001\}$. Input sequence length T in $\{96, 104, 144\}$
(2) Differential-equation based models: For Neural ODE and Neural CDE, we train for 100 epochs with a learning rate λ in $\{0.01, 0.05, 0.001, 0.005, 0.0001\}$. Hidden size in $\{39, 49, 59\}$. Input sequence length T in $\{60, 72, 96\}$.
(3) Transformer-based models: For Autoformer and FEDformer, we train for 50 epochs with a learning rate λ in $\{0.01, 0.05, 0.001, 0.005, 0.0001\}$. Input sequence length T in $\{60, 72, 96\}$. Other settings follow the same experimental settings in the baseline.
(4) PatchTST: We train for 50 epochs with a learning rate λ in $\{0.01, 0.05, 0.001, 0.005, 0.0001\}$. Input sequence length T in $\{60, 72, 96\}$. Other settings follow the same experimental settings in the baseline.

I.2 Hyperparameter for CONTIME

For reproducibility, we report the hyperparameters search range as follows:

Stocks: We train for 100 epochs with a batch size of 256. A learning rate λ in $\{0.001, 0.005, 0.01, 0.05\}$ are used. Coefficient of $L_{CONTIME}$ α in $\{0.7, 0.8, 0.9, 1.0\}$ and β in $\{0.001, 0.005, 0.01, 0.05, 0.1\}$. We used rk4 as an ODE solver. Our input sequence length T in $\{96, 104, 144\}$.

Exchange: We train for 100 epochs with a batch size of 256. A learning rate λ in $\{0.001, 0.005, 0.01, 0.05\}$ are used. Coefficient of $L_{CONTIME}$ α in $\{0.7, 0.8, 0.9, 1.0\}$ and β in $\{0.001, 0.005, 0.01, 0.05, 0.1\}$. We used rk4 as an ODE solver. Our input sequence length T in $\{60, 72, 96\}$.

Weather: We train for 150 epochs with a batch size of 256. A learning rate λ in $\{0.001, 0.005, 0.01, 0.05\}$ are used. Coefficient of $L_{CONTIME}$ α in $\{0.7, 0.8, 0.9, 1.0\}$ and β in $\{0.001, 0.005, 0.01, 0.05, 0.1\}$. We used rk4 as an ODE solver. Our input sequence length T in $\{60, 72, 96\}$.

In Table 10, we showed our best hyperparameter for all 6 datasets.

J EXPERIMENTAL RESULTS WITH STANDARD DEVIATION

In Table 11 and Table 12, we repeat training and testing procedures with three different random seeds and report their mean squared error and standard deviations of all 6 datasets.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009

Table 11: Experimental results on 3 datasets. The best results are in bold and the second best are underlined.

Datasets		APPL			AMZN			GOOG		
	P	TDI	DTW	MSE	TDI	DTW	MSE	TDI	DTW	MSE
DLinear	24	3.810 ± 0.031	1.409 ± 0.034	0.105 ± 0.004	3.855 ± 0.048	2.239 ± 0.050	0.265 ± 0.008	3.766 ± 0.031	1.297 ± 0.006	0.166 ± 0.008
	36	5.106 ± 0.028	1.940 ± 0.049	0.187 ± 0.008	5.396 ± 0.051	2.726 ± 0.054	0.372 ± 0.006	4.835 ± 0.044	2.229 ± 0.084	0.199 ± 0.008
	48	7.751 ± 1.060	2.323 ± 0.033	0.213 ± 0.008	8.915 ± 0.056	2.964 ± 0.073	0.408 ± 0.008	7.518 ± 0.077	2.568 ± 0.034	0.262 ± 0.003
	60	10.84 ± 1.276	2.907 ± 0.143	0.258 ± 0.010	9.252 ± 0.047	3.017 ± 0.055	0.347 ± 0.009	12.39 ± 0.038	2.848 ± 0.061	0.294 ± 0.009
NODE	24	3.739 ± 0.047	4.330 ± 0.008	0.168 ± 0.007	3.063 ± 0.021	3.275 ± 0.009	0.397 ± 0.001	3.684 ± 0.055	6.399 ± 0.010	1.298 ± 0.091
	36	4.911 ± 0.071	2.916 ± 0.008	0.328 ± 0.008	5.479 ± 0.033	4.893 ± 0.009	0.464 ± 0.001	5.793 ± 0.167	4.223 ± 0.012	0.646 ± 0.001
	48	7.482 ± 0.028	4.203 ± 0.010	0.535 ± 0.001	7.149 ± 0.092	6.436 ± 0.009	0.813 ± 0.087	7.795 ± 0.041	5.112 ± 0.006	0.794 ± 0.001
	60	8.702 ± 0.030	10.25 ± 0.007	1.149 ± 0.021	8.954 ± 0.073	6.333 ± 0.011	1.033 ± 0.001	9.513 ± 0.065	5.648 ± 0.014	0.874 ± 0.001
NCDE	24	5.039 ± 0.020	4.555 ± 0.033	0.227 ± 0.014	2.984 ± 0.037	5.493 ± 0.041	0.261 ± 0.012	3.719 ± 0.046	4.601 ± 0.040	0.517 ± 0.017
	36	6.651 ± 0.032	3.199 ± 0.074	0.462 ± 0.082	5.829 ± 0.036	4.022 ± 0.032	0.335 ± 0.016	4.946 ± 0.057	3.541 ± 0.042	0.582 ± 0.009
	48	7.303 ± 0.044	4.028 ± 0.056	0.440 ± 0.019	7.113 ± 0.038	5.817 ± 0.063	0.711 ± 0.011	8.132 ± 0.053	6.161 ± 0.073	0.756 ± 0.011
	60	11.47 ± 0.093	3.882 ± 0.002	0.459 ± 0.015	9.041 ± 0.043	7.936 ± 0.062	1.352 ± 0.014	10.02 ± 0.025	5.637 ± 0.053	0.771 ± 0.011
Autoformer	24	3.085 ± 0.024	1.551 ± 0.023	0.150 ± 0.024	3.576 ± 0.037	**1.485** ± 0.022	0.174 ± 0.001	3.289 ± 0.031	**1.239** ± 0.021	0.167 ± 0.001
	36	6.561 ± 0.025	1.882 ± 0.029	0.171 ± 0.001	5.541 ± 0.087	2.032 ± 0.018	0.203 ± 0.011	5.782 ± 0.061	2.210 ± 0.021	0.199 ± 0.034
	48	9.814 ± 0.031	2.307 ± 0.019	0.170 ± 0.001	6.941 ± 0.063	2.388 ± 0.013	0.219 ± 0.017	7.606 ± 0.040	2.943 ± 0.027	0.289 ± 0.061
	60	13.82 ± 0.052	2.651 ± 0.019	0.188 ± 0.001	9.414 ± 0.035	**2.723** ± 0.021	0.275 ± 0.001	10.80 ± 0.021	3.248 ± 0.023	0.279 ± 0.001
FEDformer	24	3.417 ± 0.045	1.396 ± 0.063	0.129 ± 0.009	3.108 ± 0.100	1.764 ± 0.056	0.232 ± 0.004	3.154 ± 0.034	1.587 ± 0.041	0.204 ± 0.005
	36	6.335 ± 0.031	1.826 ± 0.052	0.149 ± 0.005	5.878 ± 0.078	2.201 ± 0.043	0.249 ± 0.079	5.311 ± 0.071	2.203 ± 0.030	0.215 ± 0.004
	48	12.64 ± 0.074	1.932 ± 0.031	0.135 ± 0.004	7.664 ± 0.103	2.691 ± 0.071	0.289 ± 0.003	8.489 ± 0.080	2.312 ± 0.016	0.225 ± 0.003
	60	16.39 ± 0.081	2.642 ± 0.102	0.204 ± 0.007	12.84 ± 0.108	2.980 ± 0.027	0.354 ± 0.007	12.13 ± 0.081	2.785 ± 0.021	0.244 ± 0.008
PatchTST	24	3.166 ± 0.149	1.253 ± 0.046	0.084 ± 0.006	3.969 ± 0.078	1.574 ± 0.051	0.177 ± 0.006	3.706 ± 0.066	1.554 ± 0.043	**0.165** ± 0.007
	36	5.358 ± 0.123	**1.417** ± 0.556	0.118 ± 0.006	6.679 ± 0.089	**1.733** ± 0.022	**0.168** ± 0.008	4.882 ± 0.054	2.766 ± 0.037	0.191 ± 0.009
	48	7.984 ± 0.693	**1.809** ± 0.459	0.130 ± 0.063	8.706 ± 0.062	2.521 ± 0.045	0.220 ± 0.011	7.840 ± 0.061	2.342 ± 0.051	0.203 ± 0.006
	60	11.00 ± 1.025	2.626 ± 0.042	0.202 ± 0.057	12.24 ± 0.042	3.475 ± 0.076	0.275 ± 0.009	10.64 ± 0.043	**2.673** ± 0.061	0.244 ± 0.007
CONTIME	24	**2.378** ± 0.019	**1.114** ± 0.013	**0.074** ± 0.008	**2.866** ± 0.022	1.529 ± 0.017	**0.167** ± 0.001	**3.052** ± 0.011	1.541 ± 0.017	**0.165** ± 0.004
	36	**4.807** ± 0.033	1.541 ± 0.016	**0.089** ± 0.002	**5.275** ± 0.008	1.881 ± 0.035	0.193 ± 0.090	**4.712** ± 0.021	**2.189** ± 0.010	**0.189** ± 0.007
	48	**7.300** ± 0.011	1.912 ± 0.008	**0.114** ± 0.002	**6.844** ± 0.012	**2.300** ± 0.016	**0.209** ± 0.001	**7.364** ± 0.020	**2.297** ± 0.016	**0.188** ± 0.003
	60	**7.932** ± 0.017	**2.625** ± 0.016	**0.147** ± 0.009	**8.885** ± 0.018	2.873 ± 0.013	**0.239** ± 0.006	**9.271** ± 0.015	2.741 ± 0.012	**0.210** ± 0.001

Table 12: Experimnetal results on 3 datasets.

Datasets		MSFT			Exchange Rate			Weather		
	P	TDI	DTW	MSE	TDI	DTW	MSE	TDI	DTW	MSE
DLinear	24	4.327 ± 0.044	**1.430** ± 0.091	0.197 ± 0.071	3.629 ± 0.052	**0.533** ± 0.003	**0.044** ± 0.091	3.505 ± 0.071	1.894 ± 0.838	0.119 ± 0.041
	36	6.103 ± 0.083	2.385 ± 0.044	0.319 ± 0.062	5.638 ± 0.088	0.781 ± 0.023	0.065 ± 0.071	5.944 ± 0.091	1.436 ± 0.064	0.144 ± 0.838
	48	7.324 ± 0.087	3.738 ± 0.014	0.468 ± 0.071	7.989 ± 0.852	1.742 ± 0.041	**0.084** ± 0.052	8.208 ± 0.041	1.817 ± 0.002	0.161 ± 0.064
	60	12.10 ± 0.062	4.247 ± 0.062	0.492 ± 0.048	11.01 ± 0.174	2.304 ± 0.029	0.107 ± 0.064	10.16 ± 0.057	**1.771** ± 0.041	**0.174** ± 0.014
NODE	24	4.596 ± 0.071	3.389 ± 0.002	0.359 ± 0.012	2.085 ± 0.031	2.855 ± 0.020	0.525 ± 0.014	2.758 ± 0.084	4.262 ± 0.052	0.336 ± 0.013
	36	6.769 ± 0.022	4.329 ± 0.012	0.496 ± 0.019	4.055 ± 0.081	9.289 ± 0.051	1.137 ± 0.011	4.314 ± 0.090	6.193 ± 0.031	0.489 ± 0.012
	48	8.868 ± 0.101	4.656 ± 0.030	0.504 ± 0.013	6.104 ± 0.073	6.028 ± 0.031	1.100 ± 0.018	6.827 ± 0.082	6.294 ± 0.022	1.261 ± 0.018
	60	10.72 ± 0.073	7.973 ± 0.025	0.618 ± 0.031	9.822 ± 0.091	6.621 ± 0.021	1.056 ± 0.021	10.54 ± 0.080	7.652 ± 0.032	1.506 ± 0.020
NCDE	24	4.842 ± 0.022	2.809 ± 0.081	0.445 ± 0.064	1.874 ± 0.081	3.689 ± 0.092	0.576 ± 0.030	2.489 ± 0.031	5.609 ± 0.061	0.854 ± 0.081
	36	6.687 ± 0.082	2.902 ± 0.034	0.628 ± 0.016	4.184 ± 0.045	8.137 ± 0.033	0.542 ± 0.087	4.661 ± 0.027	4.059 ± 0.061	0.799 ± 0.038
	48	9.018 ± 0.071	4.327 ± 0.065	0.690 ± 0.063	6.012 ± 0.082	7.957 ± 0.088	0.874 ± 0.024	6.922 ± 0.075	4.682 ± 0.046	0.783 ± 0.076
	60	12.35 ± 0.091	5.221 ± 0.022	0.766 ± 0.012	8.105 ± 0.049	6.516 ± 0.029	0.604 ± 0.094	9.900 ± 0.039	5.882 ± 0.081	0.989 ± 0.053
Autoformer	24	4.222 ± 0.041	1.690 ± 0.037	0.246 ± 0.009	3.158 ± 0.158	1.120 ± 0.158	0.098 ± 0.000	2.586 ± 0.000	1.938 ± 0.000	0.327 ± 0.000
	36	**5.111** ± 0.001	2.474 ± 0.022	0.288 ± 0.015	4.724 ± 0.046	1.516 ± 0.132	0.125 ± 0.019	4.662 ± 0.042	2.393 ± 0.016	0.349 ± 0.036
	48	7.335 ± 0.037	2.810 ± 0.029	0.287 ± 0.009	8.245 ± 0.407	1.760 ± 0.041	0.129 ± 0.001	6.955 ± 0.507	2.855 ± 0.067	0.415 ± 0.077
	60	12.14 ± 0.066	3.668 ± 0.102	0.380 ± 0.071	10.53 ± 0.132	**2.026** ± 0.022	0.139 ± 0.002	9.944 ± 0.132	2.854 ± 0.132	0.415 ± 0.132
FEDformer	24	4.335 ± 0.021	1.754 ± 0.051	0.243 ± 0.009	3.311 ± 0.064	0.887 ± 0.036	0.079 ± 0.006	2.872 ± 0.027	1.506 ± 0.058	0.215 ± 0.007
	36	6.794 ± 0.022	2.505 ± 0.063	0.304 ± 0.010	5.638 ± 0.093	1.079 ± 0.014	0.085 ± 0.009	5.108 ± 0.023	1.801 ± 0.039	0.313 ± 0.003
	48	8.203 ± 0.033	2.891 ± 0.101	0.308 ± 0.000	7.952 ± 0.077	1.692 ± 0.019	0.108 ± 0.002	6.342 ± 0.096	2.053 ± 0.033	0.226 ± 0.004
	60	12.76 ± 0.054	**3.209** ± 0.091	0.321 ± 0.000	10.68 ± 0.226	2.714 ± 0.054	0.128 ± 0.009	9.495 ± 0.132	2.083 ± 0.132	0.199 ± 0.132
PatchTST	24	4.222 ± 0.098	1.529 ± 0.102	0.215 ± 0.009	3.658 ± 0.138	0.903 ± 0.122	0.056 ± 0.064	3.089 ± 0.132	1.796 ± 0.132	0.119 ± 0.132
	36	6.388 ± 0.094	**2.154** ± 0.099	**0.234** ± 0.007	5.603 ± 0.436	**0.776** ± 0.026	0.078 ± 0.034	4.849 ± 0.436	1.428 ± 0.436	0.149 ± 0.036
	48	10.49 ± 0.090	3.075 ± 0.110	0.356 ± 0.088	8.083 ± 0.647	1.701 ± 0.027	0.099 ± 0.046	6.687 ± 0.647	**1.473** ± 0.647	0.181 ± 0.017
	60	14.09 ± 0.088	3.883 ± 0.098	0.693 ± 0.009	11.32 ± 0.169	2.210 ± 0.016	**0.106** ± 0.006	10.40 ± 0.049	1.988 ± 0.029	0.229 ± 0.169
CONTIME	24	**4.218** ± 0.021	1.528 ± 0.041	**0.184** ± 0.002	**1.761** ± 0.027	0.884 ± 0.013	0.049 ± 0.007	**2.254** ± 0.011	**1.023** ± 0.021	**0.117** ± 0.002
	36	5.371 ± 0.018	2.334 ± 0.007	0.256 ± 0.003	**3.488** ± 0.017	1.221 ± 0.007	**0.063** ± 0.009	**4.120** ± 0.015	**1.390** ± 0.012	**0.136** ± 0.003
	48	**7.296** ± 0.024	**2.755** ± 0.010	**0.262** ± 0.005	**5.366** ± 0.037	**1.683** ± 0.071	0.097 ± 0.011	**6.226** ± 0.033	1.805 ± 0.010	**0.159** ± 0.005
	60	**11.83** ± 0.041	3.261 ± 0.009	**0.292** ± 0.007	**7.452** ± 0.080	2.139 ± 0.001	0.125 ± 0.004	**9.366** ± 0.019	2.121 ± 0.091	**0.174** ± 0.002