DualDynamics: Synergizing Implicit and Explicit Methods for Robust Irregular Time Series Analysis

YongKyung Oh[1*], **Dong-Young Lim**[2,3*], **Sungil Kim**[2,3†]

[1]Medical & Imaging Informatics (MII) Group, University of California, Los Angeles (UCLA), CA, USA
[2]Department of Industrial Engineering, Ulsan National Institute of Science and Technology (UNIST), Republic of Korea
[3]Artificial Intelligence Graduate School, Ulsan National Institute of Science and Technology (UNIST), Republic of Korea
yongkyungoh@mednet.ucla.edu, {dlim, sungil.kim}@unist.ac.kr

Abstract

Real-world time series analysis faces significant challenges when dealing with irregular and incomplete data. While Neural Differential Equation (NDE) based methods have shown promise, they struggle with limited expressiveness, scalability issues, and stability concerns. Conversely, Neural Flows offer stability but falter with irregular data. We introduce DualDynamics, a novel framework that synergistically combines NDE-based method and Neural Flow-based method. This approach enhances expressive power while balancing computational demands, addressing critical limitations of existing techniques. We demonstrate DualDynamics' effectiveness across diverse tasks: classification of robustness to dataset shift, irregularly-sampled series analysis, interpolation of missing data, and forecasting with partial observations. Our results show consistent outperformance over state-of-the-art methods, indicating DualDynamics' potential to advance irregular time series analysis significantly.

Code — https://github.com/yongkyung-oh/DualDynamics
Extended version — https://arxiv.org/abs/2401.04979

Introduction

Effectively modeling time series data is a cornerstone in machine learning, supporting numerous applications across diverse sectors. This significance has driven substantial research endeavors aimed at developing novel methodologies capable of accurately capturing the complexities of continuous latent processes (Chen et al. 2018; Kidger et al. 2020; Kidger 2022; Oh, Lim, and Kim 2024).

Recent advancements have led to two distinct approaches: *implicit* methods, such as Neural Differential Equation (NDE)-based techniques, and *explicit* methods, such as Neural Flows. Neural Ordinary Differential Equations (Neural ODEs) (Chen et al. 2018), Neural Controlled Differential Equations (Neural CDEs) (Kidger et al. 2020) and Neural Stochastic Differential Equations (Neural SDEs) (Oh, Lim, and Kim 2024) have emerged as prominent implicit methods, recognized for their ability to learn continuous-time dynamics and underlying temporal structures. These approaches directly learn continuous latent representations based on vector fields parameterized by neural networks. However, NDE-based methods face challenges including limited expressive power (Dupont, Doucet, and Teh 2019; Massaroli et al. 2020b; Chalvidal et al. 2021; Kidger 2022) and scalability issues when analyzing irregular or complex time series data (Norcliffe et al. 2020; Morrill et al. 2021, 2022; Irie, Faccio, and Schmidhuber 2022; Pal et al. 2023). These limitations are attributed to data complexity, sequence length variations, and the stability constraints of numerical solvers (He and Semnani 2023; Westny et al. 2023).

In parallel, explicit methods have been developed, focusing on directly mapping solution curves using neural networks (Lu et al. 2018; Sonoda and Murata 2019; Massaroli et al. 2020a; Biloš et al. 2021). These approaches offer invertible solutions and enhanced stability through the change of variables formula (Kobyzev, Prince, and Brubaker 2020; Papamakarios et al. 2021). However, they struggle with irregularly-sampled time series and are sensitive to initial state and corresponding initial value problem.

To address the limitations of both implicit and explicit methods, we introduce DualDynamics, a novel framework that synergistically combines NDE-based model and Neural Flow-based model. This approach aims to leverage the strengths of both paradigms: the flexibility of implicit methods in handling irregular data and the stability and computational efficiency of explicit methods. By integrating these complementary approaches, DualDynamics seeks to enhance expressive power, improve scalability, and maintain stability in modeling complex, irregular time series data.

Related Works

The landscape of neural differential equations for time series modeling has been dominated by two distinct approaches: implicit methods, exemplified by Neural ODEs, Neural CDEs and Neural SDEs, and explicit methods, such as Neural Flows. These approaches fundamentally differ in how they represent and solve the underlying temporal dynamics.

Implicit Methods for Continuous Dynamics

Implicit methods in deep learning for time series modeling solve differential equations numerically, allowing for

*These two authors are equal contributors to this work and designated as co-first authors.
†Corresponding Author

adaptive computation and flexible handling of irregular data. These methods have gained significant attention due to their ability to model complex dynamics.

Let $\boldsymbol{x} = (x_0, x_1, \ldots, x_n)$ be a vector of original irregularly-sampled observations and $\boldsymbol{z}(t)$ denote a latent state at time t. Continuous latent dynamics $\boldsymbol{z}(t)$ can be achieved by integration of $\mathrm{d}\boldsymbol{z}(t)$ over time in $[0, t]$. Neural Differential Equations can be expressed as:

$$\text{Neural ODEs:} \quad \mathrm{d}\boldsymbol{z}(t) = f(s, \boldsymbol{z}(s); \theta_f)\,\mathrm{d}s \tag{1}$$

$$\text{Neural CDEs:} \quad \mathrm{d}\boldsymbol{z}(t) = f(s, \boldsymbol{z}(s); \theta_f)\,\mathrm{d}X(s) \tag{2}$$

$$\text{Neural SDEs:} \quad \mathrm{d}\boldsymbol{z}(t) = f(s, \boldsymbol{z}(s); \theta_f)\,\mathrm{d}s + g(s, \boldsymbol{z}(s); \theta_g)\,\mathrm{d}W(s) \tag{3}$$

with initial condition $\boldsymbol{z}(0) = h(\boldsymbol{x}; \theta_h)$, where $h : \mathbb{R}^{d_x} \to \mathbb{R}^{d_z}$ is an affine function. f and g are neural networks parameterized by θ_f and θ_g, respectively. $X(t)$ is a control path for Neural CDEs, and $W(t)$ is a Wiener process (or Brownian motion) for Neural SDEs.

Neural ODEs (Chen et al. 2018) model deterministic dynamics but may struggle with expressiveness and stability for complex trajectories (Dupont, Doucet, and Teh 2019; Massaroli et al. 2020b). Furthermore, Neural CDEs (Kidger et al. 2020) handle irregular time series effectively but can be computationally intensive for long sequences (Morrill et al. 2021). Therefore, recent works introduced extended architecture using Neural CDE. On the other hand, Neural SDEs (Li et al. 2020) incorporate uncertainty modeling but face challenges in training and inference due to their stochastic nature (Jia and Benson 2019; Oh, Lim, and Kim 2024). These methods offer varying approaches to modeling continuous-time dynamics in time series data, each with its own strengths and limitations in handling complexity, irregularity, and uncertainty.

Explicit Methods for Continuous Dynamics

Explicit methods directly model the solution of differential equations, often providing faster computation and improved stability, albeit potentially at the cost of some flexibility. Neural Flows, introduced by Biloš et al. (2021), present a novel approach to directly model the trajectory of the latent state $\boldsymbol{z}(t)$, in contrast to Neural ODEs which represent the rate of change of $\boldsymbol{z}(t)$ over time in an integration form. Specifically, $\boldsymbol{z}(t)$ is expressed as:

$$\boldsymbol{z}(t) = \mathcal{F}(t, \boldsymbol{z}(0); \theta_{\mathcal{F}}), \tag{4}$$

with the initial value $\boldsymbol{z}(0) = h(\boldsymbol{x}; \theta_h)$, where $\mathcal{F}$ is a neural network parameterized by $\theta_{\mathcal{F}}$. Note that while $\mathcal{F}$ represents the solution to the initial value problem of Neural ODEs, as described in Eq. (1), the formulation of Neural Flows in Eq. (4) does not require the use of an ODE solver. However, the requirement for the Neural Flow $\mathcal{F}(\cdot, \boldsymbol{z}(0))$ to be invertible restricts the type of neural network architectures, which may not be suitable if the initial value is unknown or unstable to estimate. In practice, determining the exact flow is challenging since many real-world problems do not have analytical solutions, which can lead to training instability.

Methodology

Proposed Framework

To illustrate our `DualDynamics` framework, we utilize Neural CDEs as the primary example in this section. Neural CDEs serve as an effective representation of the implicit component in our dual approach, offering a robust foundation for modeling irregular time series data.

Figure 1 shows the conceptual overview of the proposed architecture using Neural CDE (implicit) and Flow-based model (explicit) for classification task.

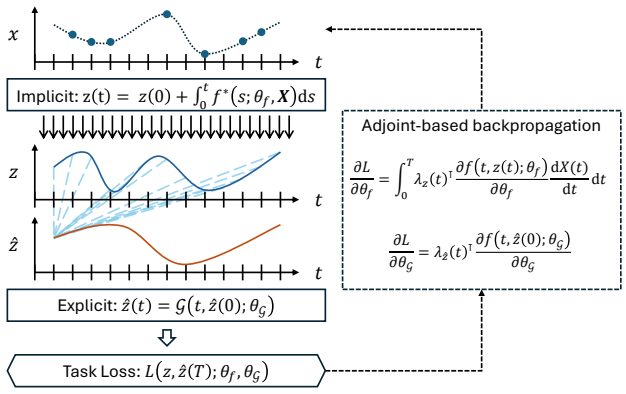


Figure 1: Overview of the proposed `DualDynamics`

The proposed method combines the stability of flow models with NDE-based model to achieve a better representation of the irregular time series. The flow model, known for its capability to smoothly transform complex distributions, offers inherent stability when dealing with time-varying dynamics. To achieve stability in the latent representation $\boldsymbol{z}(t)$, we introduce a secondary latent space, denoted by $\hat{\boldsymbol{z}}(t)$. This secondary latent space is inspired by the principles of Neural Flow and is designed to produce a regularized representation of the original latent variable. From Eq. (2), the latent representation $\boldsymbol{z}(t)$ can be derived from an initial vector $\boldsymbol{z}(0)$, which is transformed from the raw input x_0. Thus, the proposed dual latent variable evolves in time as:

$$\boldsymbol{z}(t) = \boldsymbol{z}(0) + \int_0^t f(s, \boldsymbol{z}(s); \theta_f)\,\mathrm{d}X(s), \tag{5}$$

where the initial value of the first latent representation $\boldsymbol{z}(0) = h(x_0; \theta_h)$, and $h : \mathbb{R}^{d_x} \to \mathbb{R}^{d_z}$. As stated in Chen et al. (2018) and Kidger et al. (2020), we focus on the time-evolving state $\boldsymbol{z}$ from the linear mapping from $\boldsymbol{x}$ using h. Eq. (5) can be rewritten as follows:

$$\boldsymbol{z}(t) = \boldsymbol{z}(0) + \int_0^t f^*(s; \theta_f, X)\,\mathrm{d}s, \tag{6}$$

where $f^*(s; \theta_f, X) := f(s, \boldsymbol{z}(s); \theta_f)\frac{\mathrm{d}X(s)}{\mathrm{d}s}$. Recall that X is generated from $\boldsymbol{x}$. Here, we implement X for 'Hermite Cubic splines with a backward difference', which is smooth and unique, as discussed by Morrill et al. (2022). Through this part, we find $\boldsymbol{z} \in \mathbb{R}^{d_z}$, and then plug it into the following flow model $\mathcal{G}$ as:

$$\hat{\boldsymbol{z}}(t) = \mathcal{G}(t, \hat{\boldsymbol{z}}(0); \theta_{\mathcal{G}}), \tag{7}$$

with $\hat{\boldsymbol{z}}(0) = k(\boldsymbol{z}; \theta_k)$, and $k : \mathbb{R}^{d_z \times T} \to \mathbb{R}^{d_z}$. In other words, $\hat{\boldsymbol{z}}(0)$ is determined by $\boldsymbol{z}(t)$ for $t \in [0, T]$.

Here, $\mathcal{G}$ satisfies the ODE defined in Eq. (6), i.e., $\frac{\mathrm{d}\mathcal{G}(t,\hat{\boldsymbol{z}}(0))}{\mathrm{d}t} = f^*(t; \theta_f, X)$ and $\mathcal{G}(\cdot, \hat{\boldsymbol{z}}(0))$ is invertible. In summary, after obtaining the primary latent space $\boldsymbol{z}(t)$, we compute $\mathcal{G}$ with the initial value $\hat{\boldsymbol{z}}(0)$. This model generates the secondary latent state $\hat{\boldsymbol{z}}(t)$ without the need to directly determine its derivative $\frac{\mathrm{d}\hat{\boldsymbol{z}}(t)}{\mathrm{d}t}$.

Expressive Power of Flow Model $\mathcal{G}$ Flow-based model, which is invertible and differentiable (known as 'diffeomorphisms'), has the capability of representing any distribution. Given the Neural Flow $\mathcal{G}$, there exists a unique input $\hat{\boldsymbol{z}}(0)$ such that, $\hat{\boldsymbol{z}}(0) = \mathcal{G}^{-1}(t, \hat{\boldsymbol{z}}(t); \theta_{\mathcal{G}})$. This invertibility ensures a one-to-one mapping between the initial and transformed states, preserving information content. The change of variables formula for probability densities leads to:

$$p_{\hat{\boldsymbol{z}}(t)}(\hat{\boldsymbol{z}}(t)) = p_{\hat{\boldsymbol{z}}(0)}(\hat{\boldsymbol{z}}(0)) \left|\det \boldsymbol{J}_{\mathcal{G}}(\hat{\boldsymbol{z}}(0))\right|^{-1}, \tag{8}$$

where $\boldsymbol{J}_{\mathcal{G}}(\hat{\boldsymbol{z}}(0))$ is the Jacobian matrix of $\mathcal{G}$ at $\hat{\boldsymbol{z}}(0)$, and $|\det \boldsymbol{J}_{\mathcal{G}}(\hat{\boldsymbol{z}}(0))|$ is its determinant.

Suppose $\hat{\boldsymbol{z}}(0) \sim \mathcal{U}([0,1]^{d_z})$, meaning each component of $\hat{\boldsymbol{z}}(0)$ is uniformly distributed between 0 and 1. The probability density function of $\hat{\boldsymbol{z}}(0)$ in this multi-dimensional space is constant, $p_{\hat{\boldsymbol{z}}(0)}(y) = 1$ if $y \in [0,1]^{d_z}$. Then, Eq. (8) can be simplified to:

$$p_{\hat{\boldsymbol{z}}(t)}(\hat{\boldsymbol{z}}(t)) = \left|\det \boldsymbol{J}_{\mathcal{G}}(\hat{\boldsymbol{z}}(0))\right|^{-1}. \tag{9}$$

The determinant of the Jacobian $|\det \boldsymbol{J}_{\mathcal{G}}(\hat{\boldsymbol{z}}(0))|$ accounts for the 'stretching' or 'squeezing' of the volume elements during the transformation. By adjusting $\theta_{\mathcal{G}}$, the model can control how much each part of the space $[0,1]^{d_z}$ expands, contracts or projects, allowing $\mathcal{G}$ to mold the uniform distribution into any desired complex distribution $p_{\hat{\boldsymbol{z}}(t)}$.

Properties of the Proposed Method

Preservation of Probability Density It is crucial for ensuring that the total probability across transformations remains constant, enabling accurate modeling of dynamic systems' evolution without loss or gain of information (Li and Chen 2008; Dinh, Sohl-Dickstein, and Bengio 2016). For each fixed t, $p_{\boldsymbol{z}(t)}$ and $p_{\hat{\boldsymbol{z}}(t)}$ be the probability density functions of $\boldsymbol{z}(t)$ and $\hat{\boldsymbol{z}}(t)$, respectively. From the *instantaneous change of variables formula* (Chen et al. 2018), we observe that:

$$\frac{\partial \log p_{\boldsymbol{z}(t)}(\boldsymbol{z}(t))}{\partial t} = -\mathrm{Tr}\left(\boldsymbol{J}_{f^*}(\boldsymbol{z}(t))\right), \tag{10}$$

$$\frac{\partial \log p_{\hat{\boldsymbol{z}}(t)}(\hat{\boldsymbol{z}}(t))}{\partial t} = -\mathrm{Tr}\left(\boldsymbol{J}_{\frac{\mathrm{d}\mathcal{G}}{\mathrm{d}t}}(\hat{\boldsymbol{z}}(t))\right), \tag{11}$$

where $Tr(A)$ is the trace of matrix A and $\boldsymbol{J}_{f^*}$ is the Jacobian matrix of f^*. $\boldsymbol{J}_{\frac{\mathrm{d}\mathcal{G}}{\mathrm{d}t}}$ is the Jacobian matrix of $\frac{\mathrm{d}\mathcal{G}}{\mathrm{d}t}$ and we have used $\frac{\mathrm{d}\mathcal{G}(t,\hat{\boldsymbol{z}}(0))}{\mathrm{d}t} = f^*(t; \theta_f, X)$ for all t. Therefore, the total changes in log-density for $\boldsymbol{z}$ and $\hat{\boldsymbol{z}}$ from time 0 to t are given by:

$$\begin{aligned}
&\log p_{\boldsymbol{z}(t)}(\boldsymbol{z}(t)) - \log p_{\boldsymbol{z}(0)}(\boldsymbol{z}(0)) \\
&\quad = -\int_0^t \mathrm{Tr}\left(\boldsymbol{J}_{f^*}(\boldsymbol{z}(s))\right) \mathrm{d}s, &(12)\\
&\log p_{\hat{\boldsymbol{z}}(t)}(\hat{\boldsymbol{z}}(t)) - \log p_{\hat{\boldsymbol{z}}(0)}(\hat{\boldsymbol{z}}(0)) \\
&\quad = -\int_0^t \mathrm{Tr}\left(\boldsymbol{J}_{\frac{\mathrm{d}\mathcal{G}}{\mathrm{d}s}}(\hat{\boldsymbol{z}}(s))\right) \mathrm{d}s, &(13)\\
&\quad = -\int_0^t \mathrm{Tr}\left(\boldsymbol{J}_{f^*}(\hat{\boldsymbol{z}}(s))\right) \mathrm{d}s, &(14)
\end{aligned}$$

Eq. (12) and Eq. (14) imply that the probability content of corresponding regions in the primary latent space $\boldsymbol{z}$ and the second latent space $\hat{\boldsymbol{z}}$ must be the same when the Neural Flow $\mathcal{G}$ is perfectly estimated (Kingma and Dhariwal 2018; Onken et al. 2021).

Computational Advantage of Flow Model $\mathcal{G}$ We are interested in computing the trace of $\boldsymbol{J}_{\frac{\mathrm{d}\mathcal{G}}{\mathrm{d}s}}$ in Eq. (13), which is a key quantity to understand the change in log-density. We use Hutchinson's trace estimator[1] (Hutchinson 1989) used to estimate the trace of a matrix efficiently, especially in situations where the matrix is large and computing the trace directly is computationally expensive[2] (Adams et al. 2018; Han, Malioutov, and Shin 2015; Grathwohl et al. 2018).

Applying Hutchinson's trace estimator to $\boldsymbol{J}_{\frac{\mathrm{d}\mathcal{G}}{\mathrm{d}s}}$, we have:

$$\mathrm{Tr}\left(\boldsymbol{J}_{\frac{\mathrm{d}\mathcal{G}}{\mathrm{d}t}}(\hat{\boldsymbol{z}}(t))\right) \approx \mathbb{E}_{\boldsymbol{v}}\left[\boldsymbol{v}^\top \boldsymbol{J}_{\frac{\mathrm{d}\mathcal{G}}{\mathrm{d}t}}(\hat{\boldsymbol{z}}(t))\boldsymbol{v}\right]. \tag{15}$$

Then, Eq. (13) can be rewritten as follows:

$$\begin{aligned}
&\log p_{\hat{\boldsymbol{z}}(t)}(\hat{\boldsymbol{z}}(t)) - \log p_{\hat{\boldsymbol{z}}(0)}(\hat{\boldsymbol{z}}(0)) \\
&\quad \approx -\mathbb{E}_{\boldsymbol{v}}\left[\int_0^t \boldsymbol{v}^\top \boldsymbol{J}_{\frac{\mathrm{d}\mathcal{G}}{\mathrm{d}t}}(\hat{\boldsymbol{z}}(t))\boldsymbol{v}\,\mathrm{d}s\right] \\
&\quad = -\mathbb{E}_{\boldsymbol{v}}\left[\int_0^t \boldsymbol{v}^\top \frac{\partial \boldsymbol{J}_{\mathcal{G}}(\hat{\boldsymbol{z}}(t))}{\partial t}\boldsymbol{v}\,\mathrm{d}s\right]. &(16)
\end{aligned}$$

where the last inequality is obtained by changing the order of partial derivatives. Incorporating the flow model $\mathcal{G}$ into the original NDE-based model enhances its expressive power without significantly increasing the computational burden, as opposed to the addition of an extra NDE formulation.

Parameter Optimization

To optimize parameters involved in the proposed method, we employ adjoint-based backpropagation (Chen et al. 2018; Kidger et al. 2020; Kidger 2022; Oh, Lim, and Kim 2024). This method allows for efficient gradient computation, ensuring that the information from one step influences and refines the subsequent step, leading to a more unified

[1] Hutchinson's trace estimator states that for a square matrix A, the trace of A can be estimated as: $\mathrm{Tr}(A) \approx \mathbb{E}_{\boldsymbol{v}}[\boldsymbol{v}^\top A \boldsymbol{v}]$, where $\boldsymbol{v}$ is a random vector with each element drawn independently from a distribution with zero mean and unit variance.

[2] Hutchinson's trace estimator significantly reduces computational complexity from $\mathcal{O}({d_z}^2)$ for direct trace computation to $\mathcal{O}(d_z)$ for matrix-vector multiplications.

optimization process. The adjoint method is a powerful technique often used to calculate the gradients of systems governed by differential equations. The basic idea is to recast the derivative computation problem as the adjoint equation (Pontryagin 2018; Pollini, Lavan, and Amir 2018).

Consider an objective function $\mathcal{L}(\boldsymbol{z}, \hat{\boldsymbol{z}}(T); \theta_f, \theta_{\mathcal{G}})$ that depends on the final value of the second latent state, $\hat{\boldsymbol{z}}(T)$. Then, one can define the adjoint state $\boldsymbol{\lambda}_{\boldsymbol{z}}(t)$ and $\boldsymbol{\lambda}_{\hat{\boldsymbol{z}}}(t)$ as the gradient of the loss function with respect to the state $\boldsymbol{z}(t)$ and $\hat{\boldsymbol{z}}(t)$:

$$\boldsymbol{\lambda}_{\boldsymbol{z}}(t) = \frac{\partial \mathcal{L}}{\partial \boldsymbol{z}(t)}, \qquad \boldsymbol{\lambda}_{\hat{\boldsymbol{z}}}(t) = \frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{z}}(t)}. \tag{17}$$

To estimate the model parameters θ_f and $\theta_{\mathcal{G}}$, we compute the gradients of the loss function with respect to these parameters simultaneously:

$$\frac{\partial \mathcal{L}}{\partial \theta_f} = \int_0^T \boldsymbol{\lambda}_{\boldsymbol{z}}(t)^\top \frac{\partial f(t, \boldsymbol{z}(t); \theta_f)}{\partial \theta_f} \frac{\mathrm{d}X(t)}{\mathrm{d}t} \mathrm{d}t, \tag{18}$$

$$\frac{\partial \mathcal{L}}{\partial \theta_{\mathcal{G}}} = \boldsymbol{\lambda}_{\hat{\boldsymbol{z}}}(T)^\top \frac{\partial \mathcal{G}(T, \hat{\boldsymbol{z}}(0); \theta_{\mathcal{G}})}{\partial \theta_{\mathcal{G}}}. \tag{19}$$

Note that $\boldsymbol{\lambda}_{\boldsymbol{z}}(t)$ can be derived by integration of adjoint dynamics backward in time from T to 0, which is the continuous-time analog to the backpropagation algorithm (Rumelhart, Hinton, and Williams 1986; Kidger, Chen, and Lyons 2021). On the other hand, $\boldsymbol{\lambda}_{\hat{\boldsymbol{z}}}(t)$ is derived by backpropagation algorithm directly.

The `DualDynamics` framework integrates implicit and explicit components into a unified architecture, enabling concurrent optimization of both elements. This joint learning process ensures that the implicit Neural Differential Equation and the explicit Neural Flow components evolve synergistically, rather than sequentially, thereby capitalizing on their complementary strengths in a single, cohesive optimization procedure.

Experiments

All experiments were performed using a server on Ubuntu 22.04 LTS, equipped with an Intel(R) Xeon(R) Gold 6242 CPU and a cluster of NVIDIA A100 40GB GPUs. The source code for our experiments and datasets can be accessed at https://github.com/yongkyung-oh/DualDynamics. Please refer the supplementary material for the experiment settings and implementation details.

Proposed Method

The implicit component in our framework primarily utilizes Neural CDE, with Neural ODE and Neural SDE explored as alternatives in our ablation study. For the explicit component, we evaluated three distinct flow models: ResNet Flow, GRU Flow, and Coupling Flow. These flow architectures, as outlined by Biloš et al. (2021), were incorporated into our proposed framework. To assess the impact of using flow models, we also implemented a conventional multilayer perceptron (MLP) as a baseline for comparison. In practice, optimal flow model is selected by hyperparameter tuning stage.

Benchmark Methods

In experiments, we employed a variety of benchmark models for classification and forecasting, incorporating conventional RNN (Rumelhart, Hinton, and Williams 1986; Medsker and Jain 1999), LSTM (Hochreiter and Schmidhuber 1997), and GRU (Chung et al. 2014), alongside their notable variants, including GRU-Δt (Choi et al. 2016), and GRU-D (Che et al. 2018). Also, we used attention-based method including transformer (Vaswani et al. 2017), Multi-Time Attention Networks (MTAN) (Shukla and Marlin 2021), and Multi-Integration Attention Module (MIAM) (Lee et al. 2022). ODE-based models like, GRU-ODE (De Brouwer et al. 2019), ODE-RNN (Rubanova, Chen, and Duvenaud 2019) ODE-LSTM (Lechner and Hasani 2020), Latent-ODE (Rubanova, Chen, and Duvenaud 2019), Augmented-ODE (Dupont, Doucet, and Teh 2019), and Attentive co-evolving neural ordinary differential equations (ACE-NODE) (Jhin et al. 2021) are considerd. Furthermore, Neural CDE (Kidger et al. 2020), and Neural Rough Differential Equation (Neural RDE) (Morrill et al. 2021), were included in our comparative study. Additionally, recent advances in Neural CDE are considered, including Attentive Neural Controlled Differential Equation (ANCDE) (Jhin et al. 2024), EXtrapolation and InTerpolation-based model (EXIT) (Jhin et al. 2022), and LEArnable Path-based model (LEAP) (Jhin et al. 2023). Lastly, variations of Neural Flow (Biloš et al. 2021) are also included.

In the interpolation task, encoder-decoder architecture is implemented by Variational Auto-Encoder (VAE) scheme with the evidence lower bound (ELBO). RNN-VAE (Chen et al. 2018), L-ODE-RNN (Chen et al. 2018), L-ODE-ODE (Rubanova, Chen, and Duvenaud 2019), and mTAND-Full (MTAN encoder–MTAN decoder model) (Shukla and Marlin 2021), were used as benchmark methods, based on the suggestion of Shukla and Marlin (2021). In our model, the proposed method is utilized for the encoder component, while the conventional RNN is employed as the decoder.

Robustness to Dataset Shift

Descriptive Example. To provide comprehensive insights, we conducted a descriptive analysis of model performances using the 'BasicMotions' dataset as a case study. We focused on five key methods: a standard RNN, GRU-ODE, ODE-LSTM, Neural CDE, Neural Flow, and our proposed method. Our investigation covered four different settings, including regular (0% missingness), and three missing rates including 30%, 50%, and 70%, respectively.

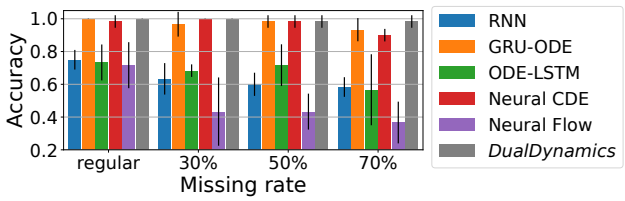
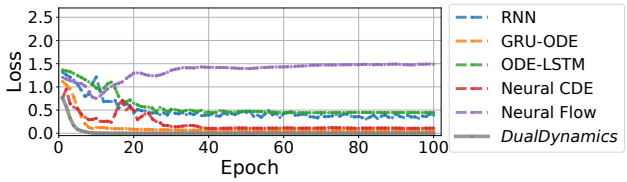


Figure 2: Classification performance comparison on the 'BasicMotions' dataset with different scenarios

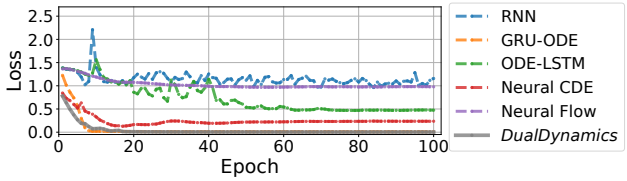
Methods	Regular		30% Missing		50% Missing		70% Missing		Average	
	Accuracy	Rank	Accuracy	Rank	Accuracy	Rank	Accuracy	Rank	Accuracy	Rank
RNN	0.560 (0.072)	11.9	0.484 (0.075)	14.7	0.471 (0.082)	14.0	0.453 (0.068)	14.4	0.492 (0.074)	13.8
LSTM	0.588 (0.067)	11.0	0.552 (0.075)	10.2	0.516 (0.073)	11.4	0.505 (0.067)	11.3	0.540 (0.071)	11.0
GRU	0.674 (0.080)	7.3	0.639 (0.065)	8.4	0.611 (0.076)	9.0	0.606 (0.088)	8.6	0.633 (0.077)	8.3
GRU-Δt	0.629 (0.065)	9.9	0.636 (0.069)	8.2	0.651 (0.068)	7.2	0.649 (0.074)	7.9	0.641 (0.069)	8.3
GRU-D	0.593 (0.088)	10.9	0.579 (0.087)	10.6	0.580 (0.075)	10.6	0.599 (0.062)	10.1	0.588 (0.078)	10.6
Transformer	0.720 (0.063)	6.8	0.663 (0.066)	8.0	0.669 (0.075)	7.4	0.643 (0.079)	8.2	0.674 (0.071)	7.6
MTAN	0.654 (0.088)	9.4	0.634 (0.098)	7.4	0.631 (0.089)	6.8	0.642 (0.073)	6.8	0.640 (0.087)	7.6
MIAM	0.573 (0.079)	11.9	0.585 (0.086)	10.0	0.572 (0.071)	9.9	0.544 (0.063)	11.1	0.568 (0.075)	10.7
GRU-ODE	0.663 (0.072)	7.6	0.661 (0.069)	7.4	0.664 (0.069)	6.8	0.659 (0.081)	6.9	0.662 (0.073)	7.2
ODE-RNN	0.652 (0.085)	7.3	0.632 (0.076)	7.8	0.626 (0.086)	7.8	0.653 (0.059)	6.5	0.641 (0.076)	7.4
ODE-LSTM	0.566 (0.074)	11.4	0.518 (0.069)	12.7	0.501 (0.068)	13.3	0.474 (0.068)	13.4	0.515 (0.070)	12.7
Neural CDE	0.681 (0.073)	7.6	0.672 (0.068)	7.4	0.661 (0.070)	7.3	0.652 (0.091)	7.3	0.667 (0.075)	7.4
Neural RDE	0.649 (0.082)	8.6	0.648 (0.071)	7.4	0.633 (0.078)	8.2	0.607 (0.079)	8.5	0.634 (0.078)	8.2
ANCDE	0.662 (0.083)	7.9	0.661 (0.083)	7.1	0.639 (0.080)	8.2	0.631 (0.073)	7.3	0.649 (0.080)	7.6
EXIT	0.595 (0.087)	10.1	0.580 (0.088)	10.5	0.578 (0.086)	10.1	0.564 (0.072)	10.4	0.579 (0.083)	10.3
LEAP	0.490 (0.062)	14.4	0.459 (0.070)	15.1	0.466 (0.074)	13.6	0.451 (0.074)	13.7	0.466 (0.070)	14.2
Neural Flow	0.545 (0.059)	11.9	0.485 (0.067)	13.4	0.455 (0.058)	14.3	0.438 (0.054)	14.1	0.481 (0.059)	13.4
Proposed method	**0.724 (0.090)**	**5.1**	**0.720 (0.088)**	**4.9**	**0.691 (0.091)**	**5.1**	**0.697 (0.098)**	**4.6**	**0.708 (0.092)**	**4.9**

Table 1: Average classification performance on 18 datasets under regular and three missing rates (Values in parentheses show the average of 18 standard deviations. The **top-ranking** and runner-up methods are emphasized.)

Figure 2 illustrates the classification performance trend of each model concerning varying missing data rates. While the conventional RNN's performance decreases as the missing rate increases, GRU-ODE and ODE-LSTM outperform conventional RNNs but become less stability with increasing missing data. Neural CDE demonstrates superior performance, whereas Neural Flow shows performance worse than that of the conventional RNN.



(a) Test loss without missingness (regular)



(b) Test loss with 50% missing (irregular)

Figure 3: Stability comparison on the 'BasicMotions' dataset, monitored over 100 epochs without early stopping

Our proposed method consistently demonstrates remarkable performance regardless of missing data rates. Further exploration is presented in Figure 3 (a) and (b), illustrating the test loss dynamics of the 'BasicMotions' dataset, both with and without missingness. Despite its simplicity, Neural Flow shows limited convergence in both cases, whereas our proposed method achieves stability, outperforming both conventional approaches and Neural CDE. Furthermore, we included detailed results and computational time analysis in the supplementary material.

Configuration	Accuracy	Loss
Proposed method (Neural CDE)	0.708 (0.092)	0.686 (0.168)
– Baseline	0.667 (0.075)	0.765 (0.264)
– Conventional MLP	0.582 (0.087)	0.862 (0.148)
– ResNet Flow	0.651 (0.091)	0.777 (0.155)
– GRU Flow	0.660 (0.079)	0.768 (0.172)
– Coupling Flow	0.669 (0.097)	0.747 (0.186)
Proposed method (Neural ODE)	0.535 (0.066)	0.968 (0.072)
– Baseline	0.502 (0.069)	0.981 (0.078)
– Conventional MLP	0.504 (0.066)	0.980 (0.076)
– ResNet Flow	0.510 (0.060)	0.978 (0.077)
– GRU Flow	0.523 (0.071)	0.981 (0.080)
– Coupling Flow	0.523 (0.064)	0.970 (0.070)
Proposed method (Neural SDE)	0.538 (0.066)	0.968 (0.071)
– Baseline	0.509 (0.061)	0.978 (0.073)
– Conventional MLP	0.506 (0.069)	0.981 (0.077)
– ResNet Flow	0.520 (0.064)	0.969 (0.073)
– GRU Flow	0.521 (0.068)	0.981 (0.073)
– Coupling Flow	0.519 (0.068)	0.975 (0.079)

Table 2: Ablation study of the proposed method

Comprehensive Analysis. Based on our initial findings, we extended our experiments to 18 public benchmark datasets from the the University of East Anglia (UEA) and

Methods	Test MSE $\times 10^{-3}$				
	50%	**60%**	**70%**	**80%**	**90%**
RNN-VAE	13.418 ± 0.008	12.594 ± 0.004	11.887 ± 0.005	11.133 ± 0.007	11.470 ± 0.006
L-ODE-RNN	8.132 ± 0.020	8.140 ± 0.018	8.171 ± 0.030	8.143 ± 0.025	8.402 ± 0.022
L-ODE-ODE	6.721 ± 0.109	6.816 ± 0.045	6.798 ± 0.143	6.850 ± 0.066	7.142 ± 0.066
mTAND-Full	4.139 ± 0.029	4.018 ± 0.048	4.157 ± 0.053	4.410 ± 0.149	4.798 ± 0.036
Proposed method	**3.631 ± 0.049**	**3.659 ± 0.028**	**3.463 ± 0.032**	**3.224 ± 0.044**	**3.114 ± 0.050**

Table 3: Performance of interpolation relative to observed percentage on the PhysioNet Mortality dataset

the University of California Riverside (UCR) Time Series Classification Repository (Bagnall et al. 2018; Dau et al. 2019), encompassing Motion & Human Activity Recognition (HAR), Electrocardiogram (ECG) & Electroencephalogram (EEG), and Sensor domains. Following Kidger et al. (2020) and Oh, Lim, and Kim (2024), we generated subsets with 30%, 50%, and 70% missingness rates, creating four distinct scenarios. Data was split 70:15:15 for training, validation, and testing. We employed five repetition, evaluating mean performance and ranking across iterations.

In the results presented in Table 1, our method distinctly stands out, demonstrating superiority when compared with other contemporary techniques. Considering the volatile nature of accuracy scores across diverse datasets, we prioritize rank statistics as a more consistent metric for comparative analysis. Across the four missing rate scenarios considered, our proposed method consistently achieves top-tier accuracy, confirming its robustness and effectiveness.

We conducted an in-depth comparative analysis of the classification efficacy and cross-entropy loss across various flow model designs, as detailed in Table 2. For implicit component, Neural CDE outperformed compared to Neural ODE and Neural SDE. The proposed method selected the flow configuration from ResNet flow, GRU flow, and Coupling flow during hyperparameter tuning. For the comparison, we employed conventional MLP, which notably failed to meet specific flow criteria. Empirical evaluations indicate that the performance of conventional MLP is worse compared to the flow-based approach. This observation confirms the superiority of our proposed method.

Interpolation of Missing Data

The interpolation experiment utilized the 2012 PhysioNet Mortality dataset (Silva et al. 2012), comprising 37 variables from ICU records over 48 hours post-admission. Following Rubanova, Chen, and Duvenaud (2019), timestamps were adjusted to the nearest minute, yielding up to 2880 intervals per series. We adhered to Shukla and Marlin (2021)'s methodology, varying observation frequency from 50% to 90% to predict remaining samples using 8000 instances.

In our interpolation experiments, we followed the experimental protocol advocated by Shukla and Marlin (2021). This strategy involved generating interpolated values based on a selected subset of data points to predict values for the unobserved time intervals. The interpolation spanned an observational range from 50% to 90% of the total data points. During the test phase, models were designed to interpolate values for the remaining intervals in the test set. The efficacy of the models was quantitatively assessed using Mean Squared Error (MSE), supported by five cross-validations.

Table 3 demonstrates that the interpolation performance of our proposed methods is superior compared to the benchmark models across all three flow configurations. The flow configuration achieved lower MSE errors compared to the non-flow neural network, a consistent and robust observation across various levels of observed data, ranging from 50% to 90%. Detailed experiment protocols and additional results are included in the supplementary material.

Classification of Irregularly-Sampled Data

We utilized the PhysioNet Sepsis dataset (Reyna et al. 2019), comprising 40,335 patient instances with 34 temporal variables. Following Kidger et al. (2020), we addressed data irregularity using two classification approaches: with and without observation intensity (OI). Observation intensity (OI) indicates patient condition severity, augmenting each time series point with an intensity index when utilized. Model performance was evaluated using Area Under the Receiver Operating Characteristic Curve (AUROC), with benchmark comparisons from Jhin et al. (2022).

Methods	Test AUROC		Memory (MB)	
	OI	**No OI**	**OI**	**No OI**
GRU-Δt	0.878 ± 0.006	0.840 ± 0.007	837	826
GRU-D	0.871 ± 0.022	0.850 ± 0.013	889	878
GRU-ODE	0.852 ± 0.010	0.771 ± 0.024	454	273
ODE-RNN	0.874 ± 0.016	0.833 ± 0.020	696	686
Latent-ODE	0.787 ± 0.011	0.495 ± 0.002	133	126
ACE-NODE	0.804 ± 0.010	0.514 ± 0.003	194	218
Neural CDE	0.880 ± 0.006	0.776 ± 0.009	244	122
ANCDE	0.900 ± 0.002	0.823 ± 0.003	285	129
EXIT	0.913 ± 0.002	0.836 ± 0.003	257	127
Proposed method	**0.918 ± 0.003**	**0.873 ± 0.004**	453	233

Table 4: AUROC on PhysioNet Sepsis dataset

In Table 4, our method demonstrates superior AUROC scores in both scenarios. Consistent with prior experimental findings, models based on flow exhibit enhanced efficacy relative to their non-flow neural network counterparts. These results confirm the effectiveness of our proposed methodologies in addressing challenges associated with time series irregularity and missingness.

Forecasting with the Partial Observations

We utilized two datasets, MuJoCo and Google, with different levels of observation for the forecasting task. Detailed explanations of each dataset, implementation details, and results are provided in the supplementary material.

MuJoCo. We utilized the MuJoCo dataset (Todorov, Erez, and Tassa 2012), based on the Hopper configuration from the DeepMind control suite (Tassa et al. 2018). Following Jhin et al. (2023, 2024), we used 50 initial points to predict the next 10, introducing 30%, 50%, and 70% missing data scenarios. Table 5 illustrates the forecasting performance across diverse conditions, with our method consistently demonstrating minimal MSE in all settings.

Methods	Test MSE			
	Regular	30% Dropped	50% Dropped	70% Dropped
GRU-Δt	0.223 ± 0.020	0.198 ± 0.036	0.193 ± 0.015	0.196 ± 0.028
GRU-D	0.578 ± 0.042	0.608 ± 0.032	0.587 ± 0.039	0.579 ± 0.052
GRU-ODE	0.856 ± 0.016	0.857 ± 0.015	0.852 ± 0.015	0.861 ± 0.015
ODE-RNN	0.328 ± 0.225	0.274 ± 0.213	0.237 ± 0.110	0.267 ± 0.217
Latent-ODE	0.029 ± 0.011	0.056 ± 0.001	0.055 ± 0.004	0.058 ± 0.003
Augmented-ODE	0.055 ± 0.004	0.056 ± 0.004	0.057 ± 0.005	0.057 ± 0.005
ACE-NODE	0.039 ± 0.003	0.053 ± 0.007	0.053 ± 0.005	0.052 ± 0.006
Neural CDE	0.028 ± 0.002	0.027 ± 0.000	0.027 ± 0.001	0.026 ± 0.001
ANCDE	0.026 ± 0.001	0.025 ± 0.001	0.025 ± 0.001	0.024 ± 0.001
EXIT	0.026 ± 0.000	0.025 ± 0.004	0.026 ± 0.000	0.026 ± 0.001
LEAP	0.022 ± 0.002	0.022 ± 0.001	0.022 ± 0.002	0.022 ± 0.001
Proposed method	**0.006 ± 0.001**	**0.008 ± 0.001**	**0.008 ± 0.001**	**0.008 ± 0.001**

Table 5: Forecasting performance comparison on the MuJoCo dataset with varying percentages of data dropped

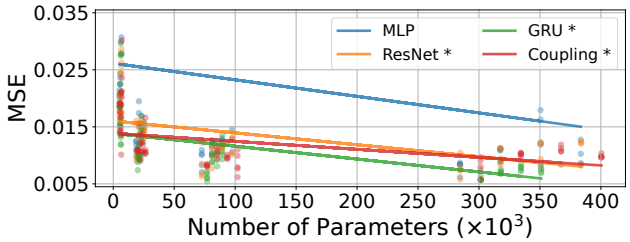


Figure 4: Forecasting performance versus the number of parameters on MuJoCo dataset with regular scenario (different flow model (with remarked $*$) versus conventional MLP)

Google. We used the Google stock data (2011-2021)(Jhin et al. 2022), forecasting 5 price metrics for the next 10 days based on 50 days of history. Following Jhin et al. (2022)'s protocol, we introduced 30%, 50%, and 70% missing data scenarios. Table 6 presents the outcomes of forecasting stock data with partial observations. When comparing with benchmark methods, our method consistently presents lower MSE in the four different scenarios.

Methods	Test MSE			
	Regular	30% Dropped	50% Dropped	70% Dropped
GRU-Δt	0.0406 ± 0.002	0.0043 ± 0.003	0.0041 ± 0.002	0.0041 ± 0.001
GRU-D	0.0040 ± 0.004	0.0058 ± 0.018	0.0039 ± 0.001	0.0040 ± 0.001
GRU-ODE	0.0028 ± 0.016	0.0029 ± 0.001	0.0031 ± 0.004	0.0030 ± 0.002
ODE-RNN	0.0311 ± 0.044	0.0318 ± 0.066	0.0322 ± 0.056	0.0324 ± 0.053
Latent-ODE	0.0030 ± 0.011	0.0032 ± 0.001	0.0033 ± 0.002	0.0032 ± 0.003
Augmented-ODE	0.0023 ± 0.002	0.0025 ± 0.002	0.0029 ± 0.006	0.0023 ± 0.005
ACE-NODE	0.0022 ± 0.003	0.0024 ± 0.001	0.0022 ± 0.002	0.0025 ± 0.002
Neural CDE	0.0037 ± 0.062	0.0038 ± 0.073	0.0032 ± 0.035	0.0039 ± 0.038
ANCDE	0.0020 ± 0.002	0.0022 ± 0.001	0.0021 ± 0.002	0.0021 ± 0.001
EXIT	0.0016 ± 0.002	0.0016 ± 0.001	0.0017 ± 0.001	0.0019 ± 0.003
Proposed method	**0.0010 ± 0.0001**	**0.0010 ± 0.0001**	**0.0009 ± 0.0001**	**0.0013 ± 0.0002**

Table 6: Forecasting performance comparison on the Google dataset with varying percentages of data dropped

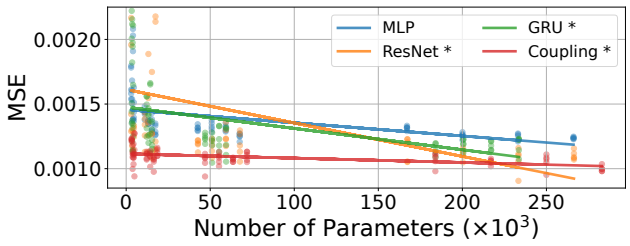


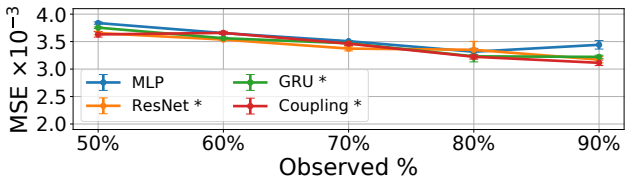
Figure 5: Forecasting performance versus the number of parameters on Google dataset with regular scenario (different flow model (with remarked $*$) versus conventional MLP)

Discussion

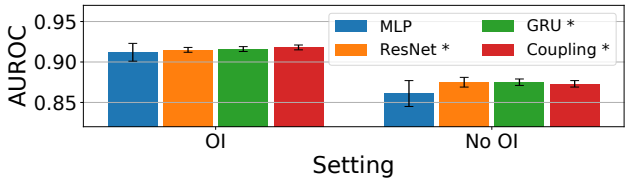
Performance versus Network Capacity. Our analysis reveals that the performance improvements of `DualDynamics` are not merely a result of increased network capacity, but rather stem from the synergistic integration of implicit and explicit methods in our framework.

To validate DualDynamics' effectiveness, we conducted controlled experiments comparing it against models with similar parameter counts. Figures 4 and 5 illustrate the relationship between parameter count and MSE for our flow architecture versus conventional MLPs. Figure 4 shows that increasing MLP capacity yields limited improvement on the MuJoCo dataset, while DualDynamics maintains superior performance. Figure 5 demonstrates that with optimal flow configuration, DualDynamics consistently outperforms MLPs on the Google dataset across all model sizes.

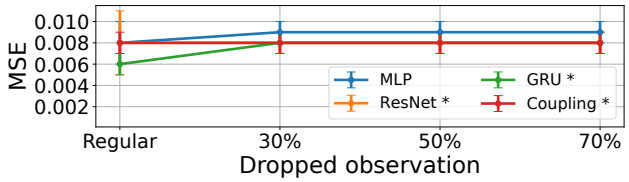
This performance advantage persists even when adjusting baseline model complexity to match DualDynamics, indicating that our framework's superiority stems from its architectural design rather than network capacity. These results highlight DualDynamics' efficiency in capturing complex temporal dynamics in irregular time series.



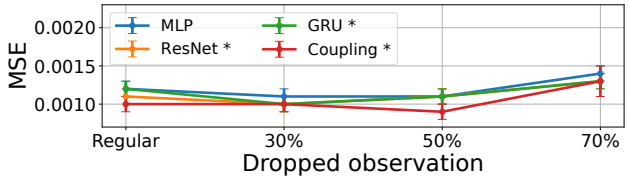
(a) Physionet Mortality



(b) Physionet Sepsis



(c) MuJoCo



(d) Google

Figure 6: Ablation study of different flow model (with remarked $*$) versus conventional MLP

Ablation Study of Flow Model Configurations. Figure 6 presents a comprehensive overview of our ablation study with four datasets, illustrating the performance of various flow configurations across different tasks, as detailed in the preceding sections. The results demonstrate a clear advantage of flow-based architectures over the conventional MLP approach, refer to the detailed results in the supplementary material. These results suggest that incorporating implicit and explicit could lead to significant improvements in model performance and capabilities.

Variations in Our Framework. Our framework's architecture allows for several potential variations, each with distinct trade-offs. One possible modification is to utilize only $\boldsymbol{z}(t)$ instead of $\hat{\boldsymbol{z}}(t)$ after training the unified framework. However, this approach significantly limits the model's expressiveness, as it fails to leverage the unique properties of the flow model. Another consideration is the replacement of the flow model with more sophisticated architectures, such as feedforward networks (Hornik, Stinchcombe, and White 1989), gated recurrent units (Chung et al. 2014), temporal convolutional networks (Lea et al. 2017), or transformer-based attention mechanisms (Vaswani et al. 2017). For further details and quantitative comparisons, we refer the reader to the supplementary material.

Conclusion

We introduce DualDynamics, an innovative methodology that integrates implicit and explicit methods for modeling irregularly-sampled time series data. This sophisticated paradigm excels in capturing intricate temporal characteristics inherent in continuous-time processes, offering an enhanced framework for representation learning in time series analysis. By synergistically combining these approaches, our method strikes a balance between expressive power and computational efficiency, addressing key challenges in temporal modeling for the irregular time series.

Our approach consistently surpasses existing benchmarks in classification, interpolation, and forecasting tasks, significantly elevating the precision and depth of understanding in temporal dynamics. These outstanding results highlight DualDynamics' potential for diverse real-world applications. Its adaptability to various temporal modeling challenges positions it as a significant advancement in the irregular time series analysis in practices.

Ethics Statement

We commit to conducting our research with integrity, ensuring ethical practices and the responsible use of technology, fully aligning our efforts with established academic and scientific standards to promote trust and accountability.

Acknowledgments

We thank the teams and individuals for their efforts in the real-world dataset preparation and curation for our research, especially the UEA & UCR repository for the numerous datasets that we extensively analyzed.

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (RS-2024-00407852); Korea Health Technology R&D Project through the Korea Health Industry Development Institute (KHIDI),

funded by the Ministry of Health and Welfare, Republic of Korea (HI19C1095); the Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2020-0-01336, Artificial Intelligence Graduate School Program (UNIST)); and the Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.RS-2021-II212068, Artificial Intelligence Innovation Hub).

References

Adams, R. P.; Pennington, J.; Johnson, M. J.; Smith, J.; Ovadia, Y.; Patton, B.; and Saunderson, J. 2018. Estimating the spectral density of large implicit matrices. *arXiv preprint arXiv:1802.03451*.

Bagnall, A.; Dau, H. A.; Lines, J.; Flynn, M.; Large, J.; Bostrom, A.; Southam, P.; and Keogh, E. 2018. The UEA multivariate time series classification archive, 2018. *arXiv preprint arXiv:1811.00075*.

Behrmann, J.; Grathwohl, W.; Chen, R. T.; Duvenaud, D.; and Jacobsen, J.-H. 2019. Invertible residual networks. In *International Conference on Machine Learning*, 573–582. PMLR.

Biloš, M.; Sommer, J.; Rangapuram, S. S.; Januschowski, T.; and Günnemann, S. 2021. Neural flows: Efficient alternative to neural ODEs. *Advances in Neural Information Processing Systems*, 34: 21325–21337.

Chalvidal, M.; Ricci, M.; Vanrullen, R.; and Serre, T. 2021. Go with the Flow: Adaptive Control for Neural ODEs. In *ICLR 2021: International Conference on Learning Representations*.

Che, Z.; Purushotham, S.; Cho, K.; Sontag, D.; and Liu, Y. 2018. Recurrent neural networks for multivariate time series with missing values. *Scientific Reports*, 8(1): 6085.

Chen, R. T.; Rubanova, Y.; Bettencourt, J.; and Duvenaud, D. K. 2018. Neural ordinary differential equations. *Advances in neural information processing systems*, 31.

Choi, E.; Bahadori, M. T.; Schuetz, A.; Stewart, W. F.; and Sun, J. 2016. Doctor ai: Predicting clinical events via recurrent neural networks. In *Machine learning for healthcare conference*, 301–318. PMLR.

Chung, J.; Gulcehre, C.; Cho, K.; and Bengio, Y. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Dau, H. A.; Bagnall, A.; Kamgar, K.; Yeh, C.-C. M.; Zhu, Y.; Gharghabi, S.; Ratanamahatana, C. A.; and Keogh, E. 2019. The UCR time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6): 1293–1305.

De Brouwer, E.; Simm, J.; Arany, A.; and Moreau, Y. 2019. GRU-ODE-Bayes: Continuous modeling of sporadically-observed time series. *Advances in neural information processing systems*, 32.

Dinh, L.; Krueger, D.; and Bengio, Y. 2014. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*.

Dinh, L.; Sohl-Dickstein, J.; and Bengio, S. 2016. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*.

Dupont, E.; Doucet, A.; and Teh, Y. W. 2019. Augmented neural odes. *Advances in Neural Information Processing Systems*, 32.

Gao, Y.; and Lin, J. 2018. Efficient discovery of variable-length time series motifs with large length range in million scale time series. *arXiv preprint arXiv:1802.04883*.

Gouk, H.; Frank, E.; Pfahringer, B.; and Cree, M. J. 2021. Regularisation of neural networks by enforcing lipschitz continuity. *Machine Learning*, 110: 393–416.

Grathwohl, W.; Chen, R. T.; Bettencourt, J.; Sutskever, I.; and Duvenaud, D. 2018. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*.

Han, I.; Malioutov, D.; and Shin, J. 2015. Large-scale log-determinant computation through stochastic Chebyshev expansions. In *International Conference on Machine Learning*, 908–917. PMLR.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

He, Y.; and Semnani, S. J. 2023. Incremental Neural Controlled Differential Equations for Modeling of Path-dependent Materials. *arXiv preprint arXiv:2311.17336*.

Hochreiter, S.; and Schmidhuber, J. 1997. Long short-term memory. *Neural computation*, 9(8): 1735–1780.

Hornik, K.; Stinchcombe, M.; and White, H. 1989. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5): 359–366.

Hutchinson, M. F. 1989. A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3): 1059–1076.

Irie, K.; Faccio, F.; and Schmidhuber, J. 2022. Neural differential equations for learning to program neural nets through continuous learning rules. *Advances in Neural Information Processing Systems*, 35: 38614–38628.

Jhin, S. Y.; Jo, M.; Kong, T.; Jeon, J.; and Park, N. 2021. Acenode: Attentive co-evolving neural ordinary differential equations. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 736–745.

Jhin, S. Y.; Jo, M.; Kook, S.; Park, N.; Woo, S.; and Lim, S. 2023. Learnable Path in Neural Controlled Differential Equations. *arXiv preprint arXiv:2301.04333*.

Jhin, S. Y.; Lee, J.; Jo, M.; Kook, S.; Jeon, J.; Hyeong, J.; Kim, J.; and Park, N. 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.

Jhin, S. Y.; Shin, H.; Kim, S.; Hong, S.; Jo, M.; Park, S.; Park, N.; Lee, S.; Maeng, H.; and Jeon, S. 2024. Attentive neural controlled differential equations for time-series classification and forecasting. *Knowledge and Information Systems*, 66(3): 1885–1915.

Jia, J.; and Benson, A. R. 2019. Neural jump stochastic differential equations. *Advances in Neural Information Processing Systems*, 32.

Keogh, E. 2003. Efficiently finding arbitrarily scaled patterns in massive time series databases. In *European Conference on Principles of Data Mining and Knowledge Discovery*, 253–265. Springer.

Kidger, P. 2022. On neural differential equations. *arXiv preprint arXiv:2202.02435*.

Kidger, P.; Chen, R. T.; and Lyons, T. J. 2021. " Hey, that's not an ODE": Faster ODE Adjoints via Seminorms. In *ICML*, 5443–5452.

Kidger, P.; Morrill, J.; Foster, J.; and Lyons, T. 2020. Neural controlled differential equations for irregular time series. *Advances in Neural Information Processing Systems*, 33: 6696–6707.

Kingma, D. P.; and Dhariwal, P. 2018. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31.

Kobyzev, I.; Prince, S. J.; and Brubaker, M. A. 2020. Normalizing flows: An introduction and review of current methods. *IEEE*

transactions on pattern analysis and machine intelligence, 43(11): 3964–3979.

Lea, C.; Flynn, M. D.; Vidal, R.; Reiter, A.; and Hager, G. D. 2017. Temporal convolutional networks for action segmentation and detection. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 156–165.

Lechner, M.; and Hasani, R. 2020. Learning long-term dependencies in irregularly-sampled time series. *arXiv preprint arXiv:2006.04418*.

Lee, Y.; Jun, E.; Choi, J.; and Suk, H.-I. 2022. Multi-view Integrative Attention-based Deep Representation Learning for Irregular Clinical Time-series Data. *IEEE Journal of Biomedical and Health Informatics*.

Li, J.; and Chen, J. 2008. The principle of preservation of probability and the generalized density evolution equation. *Structural Safety*, 30(1): 65–77.

Li, X.; Wong, T.-K. L.; Chen, R. T.; and Duvenaud, D. 2020. Scalable gradients for stochastic differential equations. In *International Conference on Artificial Intelligence and Statistics*, 3870–3882. PMLR.

Liaw, R.; Liang, E.; Nishihara, R.; Moritz, P.; Gonzalez, J. E.; and Stoica, I. 2018. Tune: A Research Platform for Distributed Model Selection and Training. *arXiv preprint arXiv:1807.05118*.

Löning, M.; Bagnall, A.; Ganesh, S.; Kazakov, V.; Lines, J.; and Király, F. J. 2019. sktime: A unified interface for machine learning with time series. *arXiv preprint arXiv:1909.07872*.

Lu, Y.; Zhong, A.; Li, Q.; and Dong, B. 2018. Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. In *International Conference on Machine Learning*, 3276–3285. PMLR.

Massaroli, S.; Poli, M.; Bin, M.; Park, J.; Yamashita, A.; and Asama, H. 2020a. Stable neural flows. *arXiv preprint arXiv:2003.08063*.

Massaroli, S.; Poli, M.; Park, J.; Yamashita, A.; and Asama, H. 2020b. Dissecting neural odes. *Advances in Neural Information Processing Systems*, 33: 3952–3963.

Medsker, L.; and Jain, L. C. 1999. *Recurrent neural networks: design and applications*. CRC press.

Moritz, P.; Nishihara, R.; Wang, S.; Tumanov, A.; Liaw, R.; Liang, E.; Elibol, M.; Yang, Z.; Paul, W.; Jordan, M. I.; et al. 2018. Ray: A distributed framework for emerging {AI} applications. In *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, 561–577.

Morrill, J.; Kidger, P.; Yang, L.; and Lyons, T. 2022. On the choice of interpolation scheme for neural CDEs. *Transactions on Machine Learning Research*, 2022(9).

Morrill, J.; Salvi, C.; Kidger, P.; and Foster, J. 2021. Neural rough differential equations for long time series. In *International Conference on Machine Learning*, 7829–7838. PMLR.

Norcliffe, A.; Bodnar, C.; Day, B.; Moss, J.; and Liò, P. 2020. Neural ODE Processes. In *International Conference on Learning Representations*.

Oh, Y.; Lim, D.; and Kim, S. 2024. Stable Neural Stochastic Differential Equations in Analyzing Irregular Time Series Data. In *The Twelfth International Conference on Learning Representations*.

Onken, D.; Fung, S. W.; Li, X.; and Ruthotto, L. 2021. Ot-flow: Fast and accurate continuous normalizing flows via optimal transport. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 9223–9232.

Pal, S.; Zeng, Z.; Ravi, S. N.; and Singh, V. 2023. Controlled differential equations on long sequences via non-standard wavelets. In *International Conference on Machine Learning*, 26820–26836. PMLR.

Papamakarios, G.; Nalisnick, E.; Rezende, D. J.; Mohamed, S.; and Lakshminarayanan, B. 2021. Normalizing flows for probabilistic modeling and inference. *The Journal of Machine Learning Research*, 22(1): 2617–2680.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Pollini, N.; Lavan, O.; and Amir, O. 2018. Adjoint sensitivity analysis and optimization of hysteretic dynamic systems with nonlinear viscous dampers. *Structural and Multidisciplinary Optimization*, 57: 2273–2289.

Pontryagin, L. S. 2018. *Mathematical theory of optimal processes*. Routledge.

Reyna, M. A.; Josef, C.; Seyedi, S.; Jeter, R.; Shashikumar, S. P.; Westover, M. B.; Sharma, A.; Nemati, S.; and Clifford, G. D. 2019. Early prediction of sepsis from clinical data: the PhysioNet/Computing in Cardiology Challenge 2019. In *2019 Computing in Cardiology (CinC)*, Page–1. IEEE.

Rubanova, Y.; Chen, R. T.; and Duvenaud, D. K. 2019. Latent ordinary differential equations for irregularly-sampled time series. *Advances in neural information processing systems*, 32.

Rumelhart, D. E.; Hinton, G. E.; and Williams, R. J. 1986. Learning representations by back-propagating errors. *nature*, 323(6088): 533–536.

Shukla, S. N.; and Marlin, B. M. 2021. Multi-time attention networks for irregularly sampled time series. *arXiv preprint arXiv:2101.10318*.

Silva, I.; Moody, G.; Scott, D. J.; Celi, L. A.; and Mark, R. G. 2012. Predicting in-hospital mortality of icu patients: The physionet/computing in cardiology challenge 2012. In *2012 Computing in Cardiology*, 245–248. IEEE.

Sonoda, S.; and Murata, N. 2019. Transport analysis of infinitely deep neural network. *The Journal of Machine Learning Research*, 20(1): 31–82.

Tan, C. W.; Petitjean, F.; Keogh, E.; and Webb, G. I. 2019. Time series classification for varying length series. *arXiv preprint arXiv:1910.04341*.

Tassa, Y.; Doron, Y.; Muldal, A.; Erez, T.; Li, Y.; Casas, D. d. L.; Budden, D.; Abdolmaleki, A.; Merel, J.; Lefrancq, A.; et al. 2018. Deepmind control suite. *arXiv preprint arXiv:1801.00690*.

Todorov, E.; Erez, T.; and Tassa, Y. 2012. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, 5026–5033. IEEE.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Westny, T.; Mohammadi, A.; Jung, D.; and Frisk, E. 2023. Stability-Informed Initialization of Neural Ordinary Differential Equations. *arXiv preprint arXiv:2311.15890*.

Yankov, D.; Keogh, E.; Medina, J.; Chiu, B.; and Zordan, V. 2007. Detecting time series motifs under uniform scaling. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, 844–853.