Invertible Solution of Neural Differential Equations for Analysis of Irregularly-Sampled Time Series

YongKyung Oh[1], **Dongyoung Lim**[2,3], **Sungil Kim**[2,3]
[1]Industry Intelligentization Institute, UNIST
[2]Department of Industrial Engineering, UNIST
[3]Graduate school of Artificial Intelligence, UNIST
Ulsan, Republic of Korea
{yongkyungoh, dlim, sungil.kim}@unist.ac.kr

Abstract

To handle the complexities of irregular and incomplete time series data, we propose an invertible solution of Neural Differential Equations (NDE)-based method. While NDE-based methods are a powerful method for analyzing irregularly-sampled time series, they typically do not guarantee reversible transformations in their standard form. Our method suggests the variation of Neural Controlled Differential Equations (Neural CDEs) with Neural Flow, which ensures invertibility while maintaining a lower computational burden. Additionally, it enables the training of a dual latent space, enhancing the modeling of dynamic temporal dynamics. Our research presents an advanced framework that excels in both classification and interpolation tasks. At the core of our approach is an enhanced dual latent states architecture, carefully designed for high precision across various time series tasks. Empirical analysis demonstrates that our method significantly outperforms existing models. This work significantly advances irregular time series analysis, introducing innovative techniques and offering a versatile tool for diverse practical applications.

Introduction

Effectively modeling time series data stands as a foundational pursuit in machine learning, anchoring countless applications. Among the myriad of innovations, Neural Ordinary Differential Equations (Neural ODEs) have distinguished themselves, garnering widespread acclaim for their prowess in encapsulating complex temporal dynamics (Chen et al. 2018). Neural Controlled Differential Equations (Neural CDEs) represent a noteworthy enhancement, ingeniously integrating controlled paths. This integration facilitates dynamic adjustment of continuous trajectories, catering to incoming observations, thereby imbuing the model with heightened adaptability (Kidger et al. 2020).

Neural differential equation (NDE)-based methods, despite their powerful expressiveness, confront limitations in scenarios with sparse or irregularly sampled data. These methods often struggle to encapsulate the inherent uncertainty in underlying dynamics, a critical aspect in many real-world time-series data (Norcliffe et al. 2020). Additionally, NDE-based methods exhibit instability in stiff problems, where the solver of the differential equation necessitates minuscule integration steps, a phenomenon observed in both known and training-induced stiff latent dynamics (Massaroli et al. 2020b; Biloš et al. 2021).

In parallel with the evolution of NDE-based methods, the other researches have focused on the continuum limit of neural networks. This approach realizes the input-output mapping through the solution of NDEs (Lu et al. 2018; Sonoda and Murata 2019; Massaroli et al. 2020a; Biloš et al. 2021). Unlike methods that rely on ODE solvers, Neural Flows prioritize capturing the solution curves, or "flows," inherent to an ODE using neural networks, thereby offering a distinct approach to modeling. Flow model can provide the invertible solution that leverage the change of variables formula to compute probability density functions, enhancing solution stability (Kobyzev, Prince, and Brubaker 2020; Papamakarios et al. 2021). The effectiveness of the naïve implementation of neural flow is significantly influenced by the choice of initial values, necessitating meticulous design and fine-tuning for optimal performance.

Motivated by the complementary strengths of these paradigms, our work introduces an innovative methodology that incorporates Neural CDEs and Neural Flows within a unified framework, all underpinned by a dual latent space architecture for the irregularly-sampled time series classification. This synthesis endows our model with the dexterity to harness the combined might of both approaches.

Related Works

Neural ODEs

Neural ODEs present a fascinating synergy between continuous-time dynamics and the flexibility of neural architectures (Chen et al. 2018). These models exploit ODE solvers to address the challenges associated with integral formulations, as detailed in eq. (1). For a given time t, let us denote a hidden representation as $\boldsymbol{z}(t)$, which is derived from an original irregularly-sampled observation, $\boldsymbol{x} = (x_0, x_1, \ldots, x_n)$. Neural ODEs can be formally expressed as:

$$\boldsymbol{z}(t) = \boldsymbol{z}(0) + \int_0^t f(s, \boldsymbol{z}(s); \theta_f)\mathrm{d}s, \tag{1}$$

where the initial value $\boldsymbol{z}(0) = h(\boldsymbol{x}; \theta_h)$, and $h : \mathbb{R}^{d_x} \to \mathbb{R}^{d_z}$ characterized by its learnable parameters, θ_h. The differential function, $f(s, \boldsymbol{z}(s); \theta_f)$, which approximates the rate of

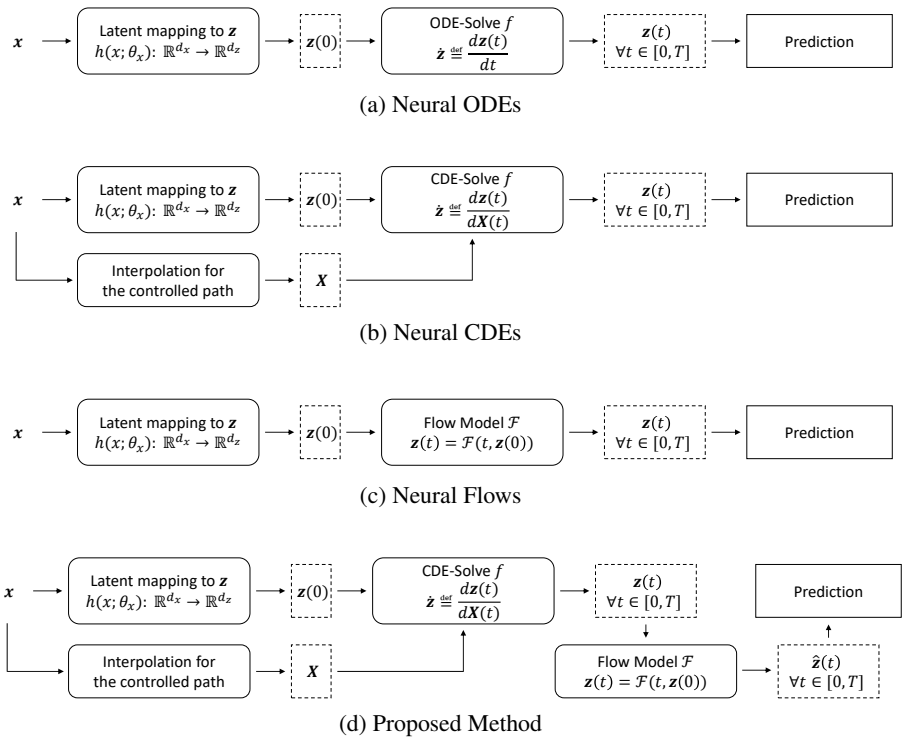


Figure 1: The overall architecture of conventional Neural ODE, Neural CDE, Neural Flow, and the proposed method.

change, $\dot{\boldsymbol{z}} \equiv \frac{\mathrm{d}\boldsymbol{z}(t)}{\mathrm{d}t}$, is modelled using a neural network by the parameters θ_f. However, a limitation of Neural ODEs is their dependency on initial conditions, which sometimes hampers their efficacy in capturing temporal nuances (Kidger 2022).

Neural CDEs

Neural CDEs provide a framework for constructing a continuous-time representation by formulating a controlled path $\boldsymbol{X}(t)$, generated through natural cubic spline interpolation of observed data (Kidger et al. 2020). The formation of Neural CDEs is expressed via the Riemann–Stieltjes integral:

$$\boldsymbol{z}(t) = \boldsymbol{z}(0) + \int_0^t f(s, \boldsymbol{z}(s); \theta_f)\, \mathrm{d}\boldsymbol{X}(s), \tag{2}$$

where the initial value $\boldsymbol{z}(0) = h(x_0; \theta_h)$, and the CDE function $f(s, \boldsymbol{z}(s); \theta_f)$, parameterized by θ_f, provides an approximation to $\frac{\mathrm{d}\boldsymbol{z}(t)}{\mathrm{d}\boldsymbol{X}(t)}$, differentiating Neural CDEs from Neural ODEs in their approximation approach. To evaluate the integral in eq. (2), one may utilize the conventional ODE solvers by recognizing that $\dot{\boldsymbol{z}}(t) \equiv \frac{\mathrm{d}\boldsymbol{z}(t)}{\mathrm{d}\boldsymbol{X}(t)}$.

Neural Flows

Introduced by Biloš et al. (2021), Neural Flows present a novel approach to directly model the solution trajectory of the ODEs through neural networks. Given a state $\boldsymbol{z}(t)$, it can be expressed as:

$$\boldsymbol{z}(t) = \mathcal{F}(t, \boldsymbol{z}(0); \theta_{\mathcal{F}}), \tag{3}$$

where the initial value $\boldsymbol{z}(0) = h(\boldsymbol{x}; \theta_h)$, and the function $\mathcal{F}$ represents the solution to the initial value problem characterized by $\dot{\boldsymbol{z}} \equiv \frac{\mathrm{d}\boldsymbol{z}(t)}{\mathrm{d}t}$. Neural Flows bypass conventional ODE solvers by directly approximating $\mathcal{F}$ with a neural network, from which the parameters $\theta_{\mathcal{F}}$ are learned. Given the reference point at $t = 0$, we derive the initial condition as $\boldsymbol{z}(0) = \mathcal{F}^{-1}(0, \boldsymbol{z}(0))$, as "analytical inverse". This leads to the sought-after solution $\mathcal{F}(\cdot, \boldsymbol{z}(0))$, in line with the initial value problem of the original ODE. However, it is not always possible to determine the precise flow, while many NDE-based methods lack closed-form solutions.

Methodology

Model architecture

The central idea of our approach revolves around integrating the principles of flow models into the Neural CDE framework. The flow model, known for its capability to smoothly transform complex distributions, offers inherent stability when dealing with time-varying dynamics. The proposed method combines the stability of flow models with Neural CDEs to establish a complementary relationship where the strengths of each address the weakness of the other.

In order 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 equation 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 dual latent variable evolves in time as:

$$\begin{aligned} \boldsymbol{z}(t) &= \boldsymbol{z}(0) + \int_0^t f(s, \boldsymbol{z}(s); \theta_f)\, \mathrm{d}\boldsymbol{X}(s), \\ \hat{\boldsymbol{z}}(t) &= \mathcal{G}(t, \hat{\boldsymbol{z}}(0); \theta_{\mathcal{G}}), \end{aligned} \tag{4}$$

where the initial value of the first latent representation $\boldsymbol{z}(0) = h(x_0; \theta_h)$, and the initial value of the second representation space $\hat{\boldsymbol{z}}(0) = k(\boldsymbol{z}; \theta_k)$ with $\boldsymbol{z}$ as continuous value over the given time. The transformation functions, represented as $h(\cdot; \theta_h)$ and $k(\cdot; \theta_k)$. capture the intrinsic features from the input representation to latent representation.

After obtaining the primary latent space z using the Riemann–Stieltjes integral with an initial value $\boldsymbol{z}(0) = h(x_0; \theta_h)$, we introduce the model $\mathcal{G}$. Let us denote a smooth flow function as $\mathcal{G}$, which can be expressed as $\mathcal{G} : [0, T] \times \mathbb{R}^{d_z} \to \mathbb{R}^{d_z}$. This model generates the secondary latent state $\hat{\boldsymbol{z}}(t)$ without the need to directly determine its derivative $\dot{\hat{\boldsymbol{z}}} \equiv \frac{\mathrm{d}\hat{\boldsymbol{z}}}{\mathrm{d}t}$ with $\hat{\boldsymbol{z}}(0) = k(\boldsymbol{z}; \theta_k)$.

Given the latent variable $\boldsymbol{z}$ drawn from the distribution $p(\boldsymbol{z})$, and considering the invertible function $\mathcal{G}$, the probability density function of $\hat{\boldsymbol{z}} = \mathcal{G}(\boldsymbol{z})$ is computable via the change of variables formula: $p(\boldsymbol{z}) = q(\hat{\boldsymbol{z}}) \cdot |\det \boldsymbol{J}(\hat{\boldsymbol{z}})|^{-1}$, where $\boldsymbol{J}$ denotes the Jacobian matrix of $\mathcal{G}$. This formulation encapsulates a core principle of flow-based models in transforming probability densities (Kobyzev, Prince, and Brubaker 2020; Papamakarios et al. 2021; Biloš et al. 2021).

Flow model architecture

We integrate three distinct flow architectures proposed in Biloš et al. (2021) within the proposed framework.

ResNet flow. Originating from the concept of Residual Networks (ResNets) (He et al. 2016), the ResNet flow model can be conceptualized as a continuous-time analogue of ResNets. The core dynamics of $\boldsymbol{x}(t)$ in ResNet flow are described by the equation:

$$\mathcal{G}(t, \boldsymbol{x}) = \boldsymbol{x} + \varphi(t) g(t, \boldsymbol{x}),$$

where φ represents a temporal embedding function and g denotes a continuous, nonlinear transformation parameterized by θ_g. To address the inherent non-invertibility in standard ResNets, spectral normalization techniques (Gouk et al. 2021) are employed, ensuring a bounded Lipschitz constant and thus, invertibility (Behrmann et al. 2019).

GRU flow. The GRU flow model extends the Gated Recurrent Unit (GRU) architecture (Chung et al. 2014) to a continuous-time setting. In this model, the hidden state $\boldsymbol{h}$ evolves continuously according to an ordinary differential equation, akin to the updates in traditional discrete GRUs. De Brouwer et al. (2019) introduced GRU-ODE, and Biloš et al. (2021) further refined it to an invertible form within the Neural Flow context, as expressed by:

$$\mathcal{G}(t, \boldsymbol{h}) = \boldsymbol{h} + \varphi(t)(1 - z(t, \boldsymbol{h})) \odot (c(t, \boldsymbol{h}) - \boldsymbol{h}),$$

where $\varphi(t)$ is a time-dependent embedding function, and $z(t, \boldsymbol{h})$, $c(t, \boldsymbol{h})$ are adapted from the GRU framework.

Coupling flow The concept of Coupling flow, initially introduced by Dinh, Krueger, and Bengio (2014) and Dinh, Sohl-Dickstein, and Bengio (2016), involves a bijective transformation characterized by its computational efficiency and flexibility. In this model, input dimensions are partitioned into two disjoint subsets d_1 and d_2, leading to the transformation:

$$\mathcal{G}(t)_{d_1} = \boldsymbol{x}_{d_1} \exp(u(t, \boldsymbol{x}_{d_2})\varphi_u(t)) + v(t, \boldsymbol{x}_{d_2})\varphi_v(t),$$

with u and v as neural networks, and φ_u, φ_v as time-dependent embedding functions.

Each flow model exhibits its own strengths and limitations. The choice of a flow model should be contingent upon the dataset's nature, the problem's specific requirements, and computational constraints. Therefore, the selection of a flow model is empirically refined during hyperparameter tuning.

Parameter optimization

While the explicit solving of Neural CDE and the implicit solving of Neural Flow occur in a sequential manner, they are linked in the training. To optimize parameters involving in both steps, we employ adjoint-based backpropagation (Kidger et al. 2020; Kidger 2022). 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 optimization process.

In contrast to Neural CDEs, our method ensures an invertible solution by the properties of the flow model. Diverging from the naïve Neural Flow, which relies on partial observations $\boldsymbol{x}$, our approach introduces a dual latent representation with variables $\boldsymbol{z}$ and $\hat{\boldsymbol{z}}$. This dual structure allows the Flow model $\mathcal{G}$ to handle $\hat{\boldsymbol{z}}$ with a complete initial value derived from $\boldsymbol{z}$. When faced with irregularly-sampled data, $\boldsymbol{z}(0)$ is inferred from the partial observation $\boldsymbol{x}$, and $\hat{\boldsymbol{z}}$ represents the complete observation of $\boldsymbol{z}$ within a specified timeframe. This method effectively addresses distribution shifts caused by partial observations (versus complete observations), a significant challenge for model robustness (Li et al. 2021; Zhou, Balakrishnan, and Lipton 2023). For a comprehensive comparison, both conventional approaches and our method are detailed in Figure 1 and Table 1.

Experiment

We undertook two experimental investigations to probe the effects of irregular sampling and data missingness.

Robust classification with the missing data As guided by the protocol established by Kidger et al. (2020), we enhanced our dataset by generating three additional subsets, each marked by distinct missingness rates: 30%, 50%, and 70%. This led to the formation of four discrete experimental scenarios. Data partitioning was executed in a 70:15:15 ratio for training, validation, and testing phases, respectively.

Interpolation for the missing data The interpolation experiment was conducted on the 2012 PhysioNet Mortality dataset (Silva et al. 2012). This dataset, a conglomeration of multivariate time series data from ICU records, encompasses 37 varied variables collected from Intensive Care Unit (ICU) records. The conducted experiments conformed to the

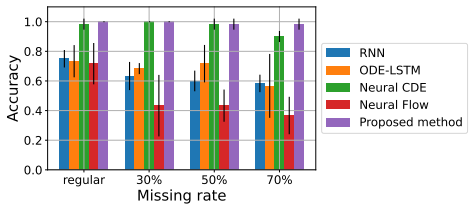
Table 1: Comparison of methodology

Model	Derivative Computation	Solving Method	Completness of Observations	Invertibility of Solution
Neural ODE	Explicit	ODE Solver	Partial $\boldsymbol{x}$	No
Neural CDE	Explicit	CDE Solver	Interpolated $\boldsymbol{X}$	No
Neural Flow	Implicit	Flow-based	Partial $\boldsymbol{x}$	Yes
Proposed Method	Mixed (Explicit & Implicit)	Combined	Interpolated $\boldsymbol{X}$ & Complete $\boldsymbol{z}$	Yes

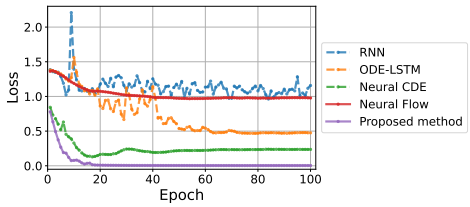
methodology proposed by Shukla and Marlin (2021), which involved altering the observation frequency from 50% to 90% for the purpose of predicting the remaining data samples.

Descriptive example

First of all, we conduct a descriptive analysis of model performances using the 'BasicMotions' dataset as example. For the sake of clarity and depth, our investigation is streamlined to five pivotal methods: a standard RNN, ODE-LSTM, (naïve) Neural CDE, (naïve) Neural Flow, and our proposed method.



(a) Classification performance



(b) Test loss for 100 epochs

Figure 2: Performance and stability comparison using 'BasicMotions' dataset (without early-stopping strategy)

Figure 2 (a) showcases the trajectory of each model's performance, especially in the context of escalating missing data rates. The conventional RNN's performance trajectory visibly slopes downward with the upsurge of missing data. ODE-LSTM, while exhibiting a more robust grasp over temporal intricacies compared to the conventional RNN, manifests erratic behaviors as missing data rates soar. Neural CDE shows superior performances, while Neural Flow shows similar performance as the conventional RNN.

Our proposed method shows remarkable performance irrespective of the missing data dynamics. Such resilience underpins its structural robustness and adaptability. Taking our exploration further, Figure 2 (b) presents a comparative lens into the test loss dynamics of the 'BasicMotions' dataset, specifically under a 50% data missing scenario. Both Neural CDE and our proposed model delineate rapid convergence trajectories. The standout attribute of the proposed method is its immediate and consistent stability.

Robust classification with the missing data

Our experiments utilize 18 public time series datasets spanning three domains: Motion & Human Activity Recognition (HAR), Electrocardiogram (ECG) & Electroencephalogram (EEG), and Sensor dataset. These datasets are sourced from the University of East Anglia (UEA) and the University of California Riverside (UCR) Time Series Classification Repository[1](Bagnall et al. 2018). Please see Appendix for an in-depth exploration of dataset and implementation details.

We employed a variety of benchmark models, incorporating conventional RNN (Rumelhart, Hinton, and Williams 1986; Medsker and Jain 1999), LSTM (Hochreiter and Schmidhuber 1997), and GRU architectures (Chung et al. 2014), alongside their notable variants (GRU-Δt (Choi et al. 2016), GRU-D (Che et al. 2018), GRU-ODE (De Brouwer et al. 2019)). ODE-based models like ODE-RNN (Rubanova, Chen, and Duvenaud 2019) and ODE-LSTM (Lechner and Hasani 2020), as well as state-of-the-art models such as Neural CDE (Kidger et al. 2020), Neural Rough Differential Equation (Neural RDE) (Morrill et al. 2021), and Neural Flow (Biloš et al. 2021), were also included in our comparative study. For Neural CDE and Neural RDE, we used 'hermite cubic splines' interpolation. We used Neural Flow without further imputation, while Oh, Lim, and Kim (2023) used mean imputation. The model's efficacy was gauged using a five-fold cross-validation method, emphasizing the mean performance and ranking across these validations.

In the results presented in Table 2, our method distinctly stands out, displaying marked superiority when juxtaposed with other contemporary techniques. Recognizing the volatile nature of accuracy scores spread across diverse datasets, we give precedence to rank statistics, deeming it to be a more consistent metric for comparative analysis. Particularly in a head-to-head comparison with Neural CDE, our approach unmistakably underscores its preeminence across all the test scenarios we evaluated. Moreover, in the context of the four missing rate scenarios that we considered, our proposed method consistently achieves top-tier accuracy, reaffirming its robustness and effectiveness.

In addition to the previous analysis, Table 3 provides a comprehensive summary of the performance across all domains as well as for three specific domains. This summary was obtained by calculating the average performance across four different scenarios of missing data rates. Upon examin-

[1]https://www.timeseriesclassification.com/

Table 2: Average classification performance on 18 datasets under regular and three missing rates (Values in parentheses show average of 18 standard deviations. **best** and second best highlighted).

Methods	Regular datasets		Missing datasets (30%)		Missing datasets (50%)		Missing datasets (70%)	
	Accuracy	Rank	Accuracy	Rank	Accuracy	Rank	Accuracy	Rank
RNN	0.560 (0.072)	8.3	0.484 (0.075)	10.2	0.471 (0.082)	9.6	0.453 (0.068)	9.9
LSTM	0.588 (0.067)	7.8	0.552 (0.075)	6.9	0.516 (0.073)	7.7	0.505 (0.067)	7.7
GRU	0.674 (0.080)	5.3	0.639 (0.065)	5.9	0.611 (0.076)	5.9	0.606 (0.088)	5.7
GRU-Δt	0.629 (0.065)	7.0	0.636 (0.069)	5.4	0.651 (0.068)	4.8	0.649 (0.074)	5.3
GRU-D	0.593 (0.088)	7.7	0.579 (0.087)	7.4	0.580 (0.075)	7.0	0.599 (0.062)	6.8
GRU-ODE	0.663 (0.072)	5.6	0.661 (0.069)	5.3	0.664 (0.069)	4.8	0.659 (0.081)	4.8
ODE-RNN	0.652 (0.085)	4.9	0.632 (0.076)	5.3	0.626 (0.086)	5.1	0.653 (0.059)	4.1
ODE-LSTM	0.566 (0.074)	8.0	0.518 (0.069)	8.7	0.501 (0.068)	9.1	0.474 (0.068)	9.1
Neural CDE	0.681 (0.073)	5.4	0.672 (0.068)	5.3	0.661 (0.070)	5.3	0.652 (0.091)	5.3
Neural RDE	0.649 (0.082)	5.9	0.648 (0.071)	5.1	0.633 (0.078)	5.6	0.607 (0.079)	5.8
Neural Flow	0.543 (0.059)	8.8	0.486 (0.069)	9.1	0.456 (0.058)	9.7	0.435 (0.056)	9.7
Proposed method	**0.726 (0.093)**	**3.3**	**0.721 (0.090)**	**3.5**	**0.691 (0.092)**	**3.4**	**0.693 (0.104)**	**3.6**

Table 3: Average classification performance on all datasets and three different domains

Methods	All domains		Motion & HAR		ECG & EEG		Sensor	
	Accuracy	Rank	Accuracy	Rank	Accuracy	Rank	Accuracy	Rank
RNN	0.492 (0.074)	9.5	0.451 (0.067)	10.1	0.547 (0.063)	8.7	0.478 (0.092)	9.8
LSTM	0.540 (0.071)	7.5	0.513 (0.091)	8.1	0.579 (0.056)	6.7	0.529 (0.065)	7.8
GRU	0.633 (0.077)	5.7	0.642 (0.093)	6.0	0.679 (0.067)	4.3	0.577 (0.072)	6.9
GRU-Δt	0.641 (0.069)	5.6	0.631 (0.074)	5.4	0.614 (0.057)	7.2	0.679 (0.076)	4.3
GRU-D	0.588 (0.078)	7.2	0.563 (0.069)	7.9	0.608 (0.060)	7.2	0.592 (0.105)	6.5
GRU-ODE	0.662 (0.073)	5.1	0.656 (0.082)	5.7	0.649 (0.064)	5.6	0.680 (0.072)	4.1
ODE-RNN	0.641 (0.076)	4.8	0.610 (0.081)	5.5	0.651 (0.054)	4.6	0.662 (0.094)	4.4
ODE-LSTM	0.515 (0.070)	8.7	0.489 (0.089)	8.8	0.555 (0.053)	8.1	0.500 (0.067)	9.2
Neural CDE	0.667 (0.075)	5.3	**0.733 (0.097)**	3.9	0.624 (0.047)	6.1	0.643 (0.083)	6.0
Neural RDE	0.634 (0.078)	5.6	0.693 (0.088)	4.2	0.598 (0.067)	6.9	0.612 (0.078)	5.7
Neural Flow	0.480 (0.061)	9.3	0.424 (0.076)	9.2	0.538 (0.048)	8.8	0.479 (0.057)	10.0
Proposed method	**0.708 (0.094)**	**3.4**	0.705 (0.099)	**3.3**	**0.710 (0.097)**	**3.8**	**0.708 (0.088)**	**3.3**

ing each individual domain, it is evident that our proposed method demonstrates generally robust performance. Compared to that, some benchmarks only outperform in the specific domain. This suggests that the method maintains its effectiveness across varying conditions and domains, indicating its adaptability and reliability in diverse scenarios.

Table 4: Ablation study of different flow model design

Methods	Accuracy	Cross-entropy loss
Neural Flow	0.480 ± 0.061	1.103 ± 0.114
– ResNet Flow	0.468 ± 0.061	1.097 ± 0.092
– GRU Flow	0.470 ± 0.065	1.109 ± 0.117
– Coupling Flow	0.475 ± 0.060	1.106 ± 0.115
Proposed method	0.708 ± 0.094	0.687 ± 0.176
– 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

The scope of our investigation encompassed 18 datasets, which were systematically classified into three domain-specific clusters. Acknowledging the diverse nature of time series data, we noted that performance metrics exhibit variability in response to the unique characteristics inherent to each domain. A salient observation, as delineated in Figure 3, underscores the strong interdependence between the efficacy of our proposed approach and the distinctive attributes of the datasets under scrutiny. Detailed analytical results, encompassing both domain-centric and individual dataset perspectives, are comprehensively documented in Appendix.

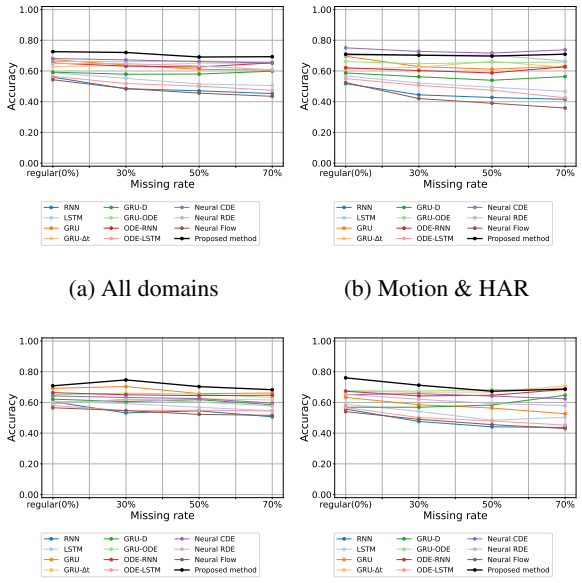
We conducted an in-depth comparative analysis of the classification efficacy and cross-entropy loss across various flow model designs, as detailed in Table 4. This comparison was predicated on the premise that performance metrics can exhibit significant variability contingent upon the dataset utilized. Consequently, to ascertain the most suitable flow model for each specific dataset, we employed a hyperparameter tuning process to select the flow model configuration.

Interpolation for the missing data

In our interpolation experiments, we follow the experimental protocol advocated by Shukla and Marlin (2021) using 8000 instances. The strategy involved generating interpolated val-

Table 5: Performance of interpolation relative to percentage of observations in the PhysioNet dataset, measured by Mean Squared Error (MSE) in Units of 10^{-3}.

Methods		**Observed %**				
		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
MTAN		4.139 ± 0.029	4.018 ± 0.048	4.157 ± 0.053	4.410 ± 0.149	4.798 ± 0.036
Proposed method	– ResNet Flow	3.655 ± 0.031	**3.537 ± 0.020**	**3.373 ± 0.042**	3.351 ± 0.154	3.171 ± 0.057
	– GRU Flow	3.754 ± 0.014	3.562 ± 0.019	3.470 ± 0.031	3.232 ± 0.101	3.222 ± 0.030
	– Coupling Flow	**3.631 ± 0.049**	3.659 ± 0.028	3.700 ± 0.142	**3.224 ± 0.044**	**3.114 ± 0.050**



(a) All domains (b) Motion & HAR

(c) ECG & EEG (d) Sensor

Figure 3: Comparison of performance across diverse domains and four missing rate scenarios

ues based on a selected subset of data points, with the aim to accurately 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 conditioned on the observed data to extrapolate values for the remaining intervals in the test set. The models' efficacy was quantitatively assessed using Mean Squared Error (MSE), underpinned by five cross-validations.

A comparative analysis was conducted against the benchmark established in Shukla and Marlin (2021), encompassing several models: the RNN-VAE model (Chen et al. 2018), which employs a VAE framework with both encoder and decoder instantiated as conventional RNNs; the L-ODE-RNN model (Chen et al. 2018), a Latent ODE variant with an RNN encoder and Neural ODE decoder; the L-ODE-ODE model (Rubanova, Chen, and Duvenaud 2019), another Latent ODE variant with both encoder and decoder grounded in ODE-RNN and Neural ODE, respectively; and the MTAN model (Shukla and Marlin 2021), which integrates a time attention mechanism alongside Bidirectional RNNs for temporal feature encapsulation and data interpolation.

In our model, the proposed method is utilized for the encoder component, while the conventional RNN is employed as the decoder. In Table 5, the interpolation performance of our proposed methods is markedly superior compared to the benchmark models across all three flow configurations. This observation is consistent and robust across various levels of observed data, ranging from 50% to 90%. These results highlight the effectiveness of our proposed methodologies in handling challenges associated with irregularly-sampled time series data or time series with missingness.

Conclusion

In this study, we have unveiled an innovative methodology that synergistically integrates the capabilities of Neural CDE and Neural Flows, thereby establishing a sophisticated paradigm for irregularly-sampled time series data modeling. This method excels in accurately encapsulating the intricate temporal characteristics inherent in continuous-time processes, offering a formidable and enhanced framework for representation learning in time series analysis.

Our approach not only surpasses existing benchmarks in the realms of classification and interpolation of irregularly-sampled time series data, but it also significantly elevates the precision and depth of understanding in temporal dynamics. Such remarkable performance metrics underscore the extensive potential of our methodology for practical application in diverse real-world time series analysis scenarios.

Acknowledgements

This work was partly supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT)(No.RS-2023-00218913), the National Research Foundation of Korea (NRF) Grant funded by the Korea government (MSIT) (NRF-2021R1F1A1061038), and the 2023 Research Fund (1.230035.01) of Ulsan National Institute of Science and Technology (UNIST). This study was also supported by the 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 (Grant number HI19C1095).

References

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.

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*.

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*.

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

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.

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

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

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.

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.

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

Li, A.; Boyd, A.; Smyth, P.; and Mandt, S. 2021. Detecting and adapting to irregular distribution shifts in bayesian online learning. *Advances in neural information processing systems*.

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*.

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.; 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. 2023. Unifying Neural Controlled Differential Equations and Neural Flow for Irregular Time Series Classification. In *The Symbiosis of Deep Learning and Differential Equations III*.

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.

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.

Zhou, H.; Balakrishnan, S.; and Lipton, Z. 2023. Domain adaptation under missingness shift. *International Conference on Artificial Intelligence and Statistics*.

Experiment details

All experiments were performed using a server on Ubuntu 22.04 LTS, equipped with an Intel(R) Xeon(R) Gold 6242 CPU and six NVIDIA A100 40GB GPUs. The source code for our experiments can be accessed at the repository[2]. It is important to note that our preliminary results have been discussed in Oh, Lim, and Kim (2023).

In our study, we conducted classification experiments across 18 diverse datasets in three distinct domains, as outlined in Table 6. Among these, the datasets in the Sensor domain are univariate, whereas the others encompass a mix of univariate and multivariate data. We utilized the original source codes for Neural CDE[3] (Kidger et al. 2020) and Neural Flow[4] (Biloš et al. 2021). The benchmark methods were directly implemented using their respective original source codes.

The interpolation task was executed utilizing the source code provided by Shukla and Marlin (2021), retrieved from their publicly accessible GitHub repository[5]. The 2012 PhysioNet Mortality dataset (Silva et al. 2012) aggregates a diverse array of multivariate time series data from ICU records, featuring 37 unique variables. This dataset is characterized by its collection of sporadic and less frequent measurements recorded within the first 48 hours following a patient's admission to the ICU. Detailed insights into the dataset employed and the specific experimental protocols followed can be gleaned by referring to the original research article.

Table 6: Description of datasets for the classification task

Domain	Dataset	Total number of samples	Number of classes	Dimension of time series	Length of time series
Motion & HAR	**BasicMotions**	80	4	6	100
	Epilepsy	275	4	3	206
	PickupGestureWiimoteZ	100	10	1	29-361
	ShakeGestureWiimoteZ	100	10	1	41-385
	ToeSegmentation1	268	2	1	277
	ToeSegmentation2	166	2	1	343
ECG & EEG	**Blink**	950	2	4	510
	ECG200	200	2	1	96
	SelfRegulationSCP1	561	2	6	896
	SelfRegulationSCP2	380	2	7	1152
	StandWalkJump	27	3	4	2500
	TwoLeadECG	1162	2	1	82
Sensor	**DodgerLoopDay**	158	7	1	288
	DodgerLoopGame	158	2	1	288
	DodgerLoopWeekend	158	2	1	288
	Lightning2	121	2	1	637
	Lightning7	143	7	1	319
	Trace	200	4	1	275

To ensure robust performance, we employed the `ray` library (Moritz et al. 2018; Liaw et al. 2018) for systematic hyperparameter tuning. Specifically, the learning rate lr is optimized in the range of 10^{-4} to 10^{-1} through a log-uniform search. The number of layers n_l is determined from the set $\{1, 2, 3, 4\}$ and the hidden vector dimensions n_h from $\{16, 32, 64, 128\}$, both utilizing a grid search methodology. Depending on the total size of the dataset, batch sizes were chosen from the set $\{16, 32, 64, 128\}$. For models based on Neural Flows, including both the naïve Neural Flow and our proposed approach, we choose from among the ResNet flow, GRU flow, and Coupling flow using a consistent procedure. Training of all considered models spans 100 epochs, with the final model selection criteria being the minimal validation loss. This ensures that our chosen model exhibits strong generalization capabilities. Our methodological choices and training regimen are geared towards achieving an optimal classifier for the task at hand.

Detailed results for each domain

We collate the outcomes across three domains: Motion & HAR, ECG & EEG, and Sensor datasets, in Tables 7, 8, and 9. In each domain, our approach consistently secures either the top or runner-up rank. This underscores the proficiency and resilience of our method in addressing real-world time series classification challenges.

Detailed results for individual dataset

Upon thorough empirical analysis, it's evident that employing this technique substantially amplifies the model's proficiency in grasping and decoding the intricate dynamics inherent in the data. In particular, for classification endeavors rooted in irregularly-sampled time series, our methodology offers a steadfast means to discern trends, resulting in heightened classification precision and a more profound comprehension of the inherent temporal frameworks. Nevertheless, the optimal model might differ for each dataset, reflecting the unique characteristics of individual time series. Therefore we represent the individual results of our experiment in Figure 4, 5, and 6.

[2]https://github.com/yongkyung-oh/torch-ists

[3]https://github.com/patrick-kidger/NeuralCDE

[4]https://github.com/mbilos/neural-flows-experiments

[5]https://github.com/reml-lab/mTAN

Table 7: Average classification performance on 6 datasets in Motion & HAR

Methods	Regular datasets		Missing datasets (30%)		Missing datasets (50%)		Missing datasets (70%)	
	Accuracy	Rank	Accuracy	Rank	Accuracy	Rank	Accuracy	Rank
RNN	0.518 (0.062)	9.3	0.445 (0.077)	10.7	0.427 (0.061)	10.4	0.415 (0.070)	9.9
LSTM	0.568 (0.089)	8.0	0.522 (0.092)	7.8	0.494 (0.094)	7.8	0.467 (0.090)	9.0
GRU	0.696 (0.095)	5.3	0.629 (0.076)	7.2	0.612 (0.089)	6.2	0.631 (0.111)	5.2
GRU-Δt	0.613 (0.076)	6.2	0.624 (0.079)	4.8	0.663 (0.064)	4.4	0.624 (0.077)	6.4
GRU-D	0.587 (0.077)	8.0	0.563 (0.089)	7.8	0.540 (0.056)	8.5	0.564 (0.054)	7.4
GRU-ODE	0.662 (0.081)	6.3	0.647 (0.091)	6.3	0.656 (0.072)	5.7	0.657 (0.084)	4.4
ODE-RNN	0.621 (0.087)	6.5	0.603 (0.078)	5.7	0.588 (0.095)	5.2	0.627 (0.064)	4.7
ODE-LSTM	0.550 (0.083)	8.9	0.507 (0.080)	8.3	0.475 (0.102)	9.3	0.426 (0.094)	8.8
Neural CDE	**0.751 (0.096)**	3.7	**0.728 (0.080)**	3.6	**0.717 (0.100)**	4.1	**0.738 (0.111)**	4.1
Neural RDE	0.702 (0.091)	4.0	0.703 (0.077)	3.8	0.704 (0.088)	4.2	0.665 (0.095)	4.8
Neural Flow	0.524 (0.068)	8.5	0.420 (0.084)	8.8	0.390 (0.082)	9.4	0.359 (0.070)	10.1
Proposed method	0.709 (0.089)	**3.4**	0.703 (0.077)	**3.4**	0.697 (0.090)	**3.0**	0.709 (0.138)	**3.2**

Table 8: Average classification performance on 6 datasets in ECG & EEG

Methods	Regular datasets		Missing datasets (30%)		Missing datasets (50%)		Missing datasets (70%)	
	Accuracy	Rank	Accuracy	Rank	Accuracy	Rank	Accuracy	Rank
RNN	0.604 (0.075)	6.8	0.532 (0.058)	10.1	0.545 (0.062)	8.3	0.506 (0.056)	9.7
LSTM	0.609 (0.054)	7.8	0.592 (0.066)	5.8	0.568 (0.057)	6.6	0.546 (0.048)	6.6
GRU	0.692 (0.073)	3.8	0.704 (0.059)	**3.7**	0.658 (0.068)	5.1	0.661 (0.068)	4.5
GRU-Δt	0.602 (0.047)	9.1	0.616 (0.052)	7.5	0.624 (0.062)	5.6	0.616 (0.069)	6.5
GRU-D	0.621 (0.056)	6.9	0.606 (0.066)	7.6	0.618 (0.055)	6.5	0.585 (0.061)	7.9
GRU-ODE	0.650 (0.057)	6.2	0.661 (0.057)	5.0	0.653 (0.063)	5.4	0.633 (0.080)	6.0
ODE-RNN	0.664 (0.068)	4.0	0.650 (0.050)	4.9	0.644 (0.048)	5.8	0.647 (0.050)	**3.7**
ODE-LSTM	0.580 (0.058)	6.8	0.546 (0.065)	8.4	0.550 (0.054)	8.9	0.545 (0.036)	8.4
Neural CDE	0.643 (0.040)	6.2	0.632 (0.038)	6.2	0.625 (0.041)	6.4	0.596 (0.069)	5.8
Neural RDE	0.591 (0.065)	8.9	0.621 (0.060)	5.9	0.602 (0.070)	6.6	0.577 (0.073)	6.3
Neural Flow	0.565 (0.047)	8.4	0.547 (0.053)	8.8	0.522 (0.046)	9.4	0.516 (0.047)	8.3
Proposed method	**0.708 (0.119)**	**3.2**	**0.747 (0.086)**	4.2	**0.703 (0.086)**	**3.3**	**0.683 (0.096)**	4.3

Table 9: Average classification performance on 6 datasets in Sensor

Methods	Regular datasets		Missing datasets (30%)		Missing datasets (50%)		Missing datasets (70%)	
	Accuracy	Rank	Accuracy	Rank	Accuracy	Rank	Accuracy	Rank
RNN	0.557 (0.078)	8.8	0.476 (0.089)	9.9	0.441 (0.123)	10.2	0.437 (0.079)	10.3
LSTM	0.588 (0.059)	7.6	0.542 (0.067)	7.2	0.484 (0.069)	8.7	0.503 (0.065)	7.6
GRU	0.633 (0.073)	6.6	0.585 (0.059)	6.8	0.563 (0.072)	6.5	0.526 (0.084)	7.5
GRU-Δt	0.673 (0.072)	5.8	0.666 (0.075)	3.9	0.668 (0.080)	4.4	**0.708 (0.075)**	**3.1**
GRU-D	0.569 (0.130)	8.1	0.569 (0.105)	7.0	0.583 (0.115)	6.1	0.647 (0.070)	5.0
GRU-ODE	0.677 (0.079)	4.5	0.674 (0.059)	4.4	**0.684 (0.072)**	**3.3**	0.687 (0.079)	4.0
ODE-RNN	0.673 (0.098)	4.3	0.642 (0.101)	5.2	0.647 (0.115)	4.2	0.684 (0.062)	4.0
ODE-LSTM	0.567 (0.083)	8.3	0.502 (0.061)	9.3	0.479 (0.049)	9.2	0.453 (0.075)	10.1
Neural CDE	0.650 (0.082)	6.5	0.657 (0.087)	6.3	0.641 (0.069)	5.4	0.623 (0.094)	6.0
Neural RDE	0.653 (0.089)	4.8	0.620 (0.076)	5.5	0.594 (0.077)	6.2	0.580 (0.069)	6.3
Neural Flow	0.540 (0.063)	9.4	0.490 (0.068)	9.6	0.455 (0.047)	10.2	0.430 (0.050)	10.8
Proposed method	**0.760 (0.070)**	**3.3**	**0.712 (0.106)**	**2.9**	0.673 (0.100)	3.8	0.687 (0.076)	3.4

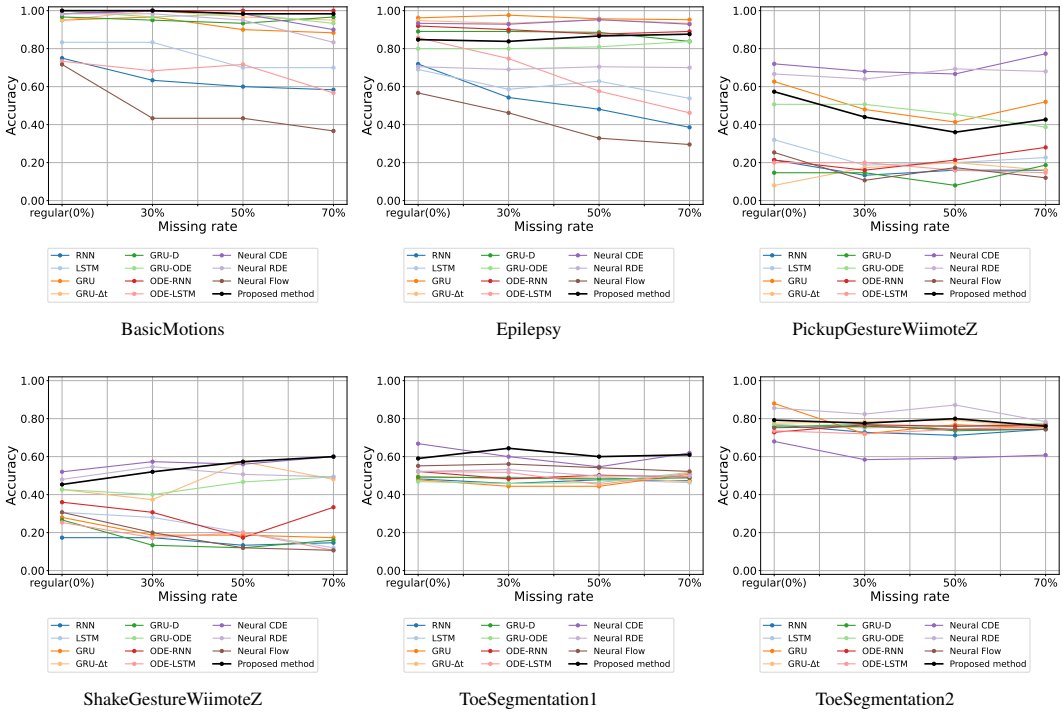


Figure 4: Classification result of the 6 datasets in Motion & HAR

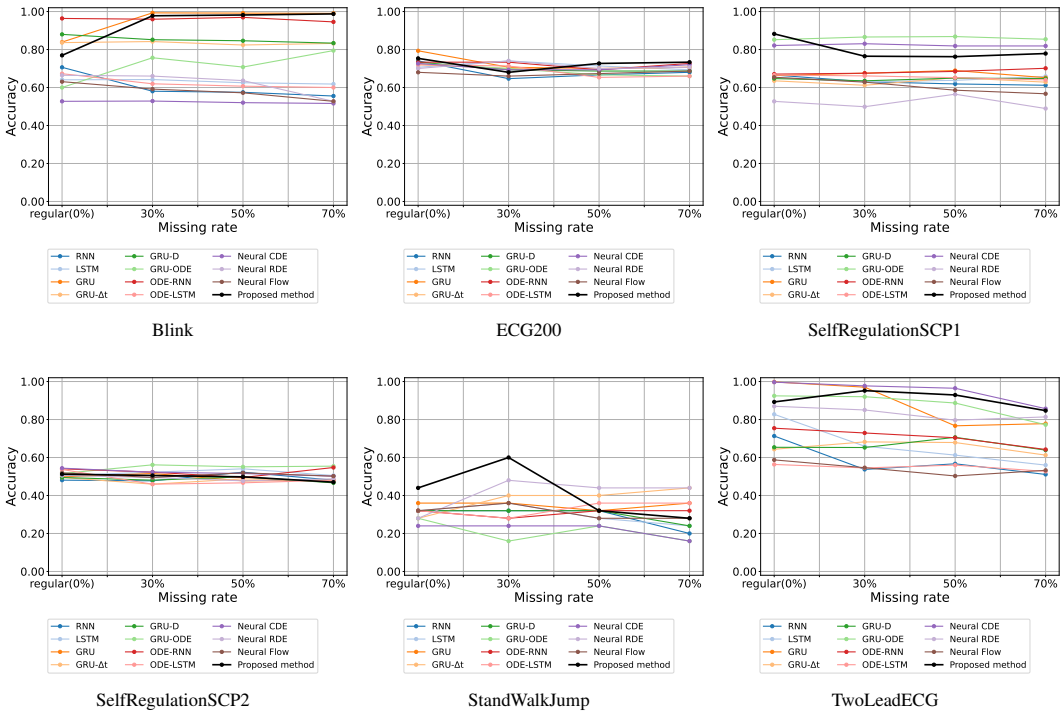


Figure 5: Classification result of the 6 datasets in ECG & EEG

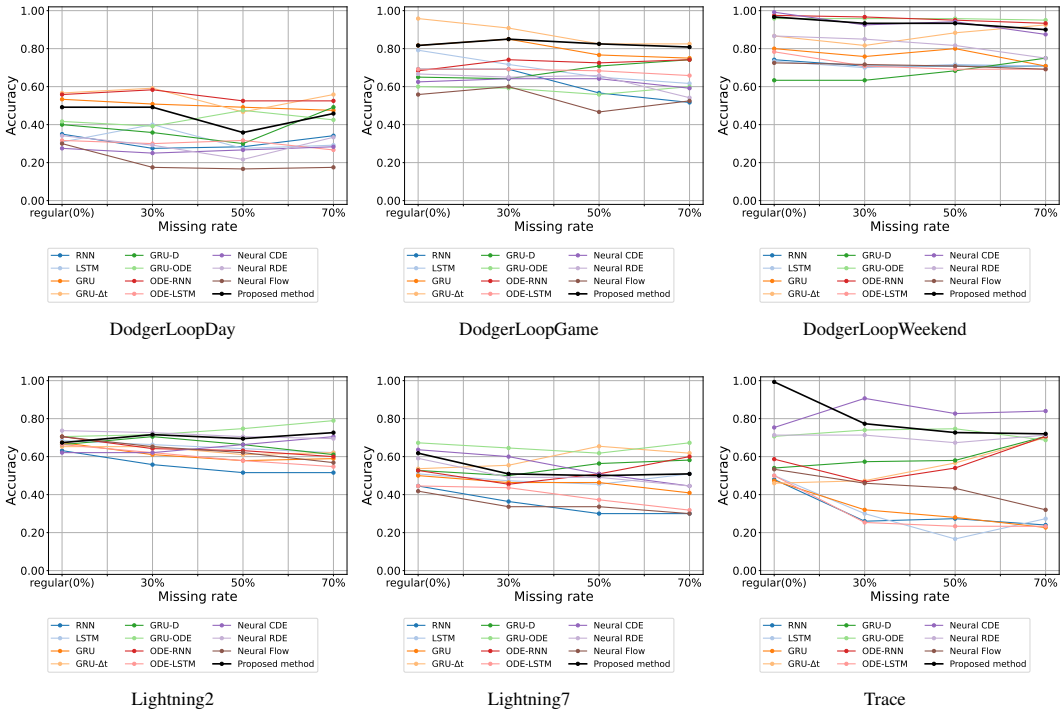


Figure 6: Classification result of the 6 datasets in Sensor