NEURAL ODE WITH DIFFERENTIABLE HIDDEN STATE FOR IRREGULAR TIME SERIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Capturing the continuous underlying dynamics of irregular time series is essential for accurately reflecting the ongoing evolution and intricate correlations within the data. The discrete nature of current models, including RNN-based models and transformer variants, poses challenges when it comes to generalizing to the continuous-time data paradigms, which is necessary for capturing ongoing dynamics of irregular time series. Neural Ordinary Differential Equations (NODEs) assume a continuous latent dynamic and provide an elegant framework for irregular time series analysis. However, integrating new information while maintaining the continuity of latent dynamics remains challenging. To tackle this problem, we introduce Differentiable Hidden State (DHS) enhanced neural ODE, a data-dependent framework that is capable of effectively capturing temporal dependencies and ensuring the continuity of the hidden process. We leverage the theory of generalized inverses to innovatively compute attention mechanism in reverse and obtain a continuous representation. To capture more accurate temporal relationships, we introduce Hoyer metric and maximize the sparsity of it. Experimental results on both synthetic and real-world datasets demonstrate the effectiveness of our model. The code is provided on anonymous link https://anonymous.4open.science/status/DHS-5F24.

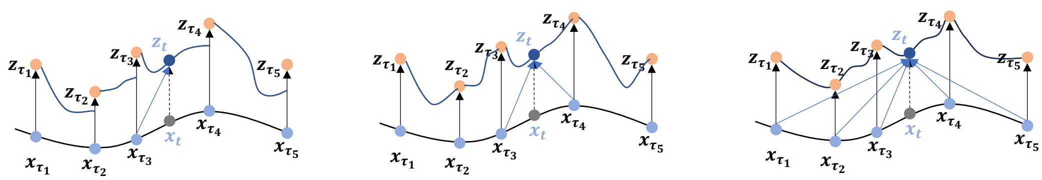
1 INTRODUCTION

Irregular time series data are ubiquitous across a variety of real-world applications, including disease prevention, financial decision-making, and earthquake prediction Bauer et al. (2016); Jia & Benson (2019); Zuo et al. (2020). Irregular time series data are characterized by non-uniform sampling, with observations occurring at variable time intervals Chen et al. (2024); Che et al. (2018). This irregularity, coupled with frequent missing data due to technical issues or data quality concerns, poses challenges for existing time series analysis methods, including RNN-based models Rangapuram et al. (2018); Salinas et al. (2020); Chung et al. (2014) and Transformer variants Zhou et al. (2021); Child et al. (2019); Li et al. (2019); Wu et al. (2021); Zhou et al. (2022).

Neural Ordinary Differential Equations (NODEs) have become a favored and promising approach for irregular time series modeling, due to their sequential processing capabilities and ability to manage irregularly sampled data Chen et al. (2018). By employing appropriate Ordinary Differential Equations (ODEs) to model the dynamics of irregular time series, it becomes feasible to reconstruct a continuous and complete time series from the irregularly sampled data through the application of integration techniques to the ODEs.

NODE-based methods Rubanova et al. (2019); Lechner & Hasani (2020); Chien & Chen (2022); De Brouwer et al. (2019); Herrera et al. (2020); Poli et al. (2019); Oskarsson et al. (2023); De Brouwer & Krishnan (2023) face a fundamental challenge on irregular time series modeling. They integrate from an initial value to derive all subsequent values, without considering observed data points later than the initial point. They integrate latent state at each time points with observations, i.e., having different initial value at different time intervals. Though such mechanism can achieve a certain accuracy, it considers only one observation at each time point, and the correlations among observations are ignored. Meanwhile, such mechanism results in a fragmented latent process that may not accurately reflect the true dynamics, as shown in Fig. 1 (a). Tackling the issue of fragmented latent state of NODEs, NCDE approaches Kidger et al. (2020); Chen et al. (2024); Li et al. (2024)

involves interpolating the observed values to estimate the latent process, for example, the natural cubic spline interpolation used in Kidger et al. (2020). This estimated process then guides the integration path, allowing the model to incorporate subsequent observations. Despite its simplicity, this method fails to leverage the full informational content of the data. As shown in Fig. 1 (b), such methods take only two nearest observations at a given time point. And using interpolation algorithm can not well model the temporal correlations in time series.



(a) Neural ODE (b) Neural CDE (c) Ours

Figure 1: The sketch of Neural ODE, Neural CDE and our method. For a given time point, NODE integrate from the last observation and has a fragmented latent process. NCDE employs interpolation algorithm to calculate a continues path, but fails to fully utilize informational content of the data. Our method introduces an attention-based differential hidden state, which adeptly captures temporal dynamics while ensuring the seamless continuity of the latent process

In response to the limitations inherent in existing solutions, this paper presents the Differentiable Hidden State (DHS) enhanced neural ODE framework, a data-driven approach designed to adeptly capture temporal dynamics while ensuring the seamless continuity of the latent process. An attention-based differential hidden state is introduced, which considers irregular sampled observations as a projection matrix mapping time series into hidden state space. Since the projection is linear, the hidden states preserve the continuity of original time series. Our methodology harnesses the power of generalized inverses to innovatively reverse-engineer the attention mechanism, yielding ODEs describing the dynamics of hidden states. To enhance the precision of temporal relationships, we integrate the Hoyer metric Hurley & Rickard (2009), an advanced tool of sparsity metric. By strategically maximizing Hoyer metric, our framework refines the model's ability to discern subtle yet significant temporal shifts, thereby improving the accuracy and reliability of predictions.

Our contribution is summarized as follows,

- We propose an attention-based differential hidden state to maintain the continuity of time series and applies generalized inverses to innovatively derive ODEs describing the dynamics of the hidden states of irregular time series.
- A deep model is constructed based on the derived ODEs, and Hoyer metric is integrated to enhance the precision of temporal correlation modeling.
- Extensive experiments are conducted on both synthetic and real-world datasets, and the result demonstrates the effectiveness of the proposed model.

2 Related Work

Deep models for time series modeling have been extensively studied. Time series modeling, with its significant practical applications, has been a focal point of research for many years Box & Jenkins (1968). The recent surge in deep learning has led to the introduction of numerous deep learning-based models Salinas et al. (2020); Qin et al. (2017); Sen et al. (2019); Tuncel & Baydogan (2018); Kalpakis et al. (2001), many of which employ recurrent neural networks (RNNs) Bai et al. (2020) and temporal convolutional networks (CNNs) Yu et al. (2018) for series modeling. However, RNNs are known for their limited parallelization capabilities, and both RNN- and CNN-based models struggle to effectively capture long-term temporal dependencies. The advent of the self-attention mechanism in fields like natural language processing Vaswani et al. (2017) and computer vision Rao et al. (2021) has inspired substantial efforts to adapt and apply transformers to time series forecasting Zeng et al. (2022); Wu et al. (2021); Zhou et al. (2022; 2021). Nevertheless, all the above methods are designed for regular time series data and fail to be extended to the problem of irregular time series analysis.

Owing to their exceptional ability to capture temporal dynamics, Neural Ordinary Differential Equations (NODEs) have recently gained popularity in the analysis of irregular time series. Existing works of NODE family can be divided into two kinds. In the first kind of methods, NCDE Kidger et al. (2020) is a pioneering work that introduces controlled differential equations into NODE, obtaining a rough estimate of the latent process using simple numerical differentiation. Subsequent work builds on this foundation, with NRDE Morrill et al. (2021) further utilizing rough path theory to model long-term sequence dependencies. ContiFormer Chen et al. (2024) constructs a continuous extension of the Transformer, where the query is obtained by interpolation. Neural Lad Li et al. (2024) models periodicity, trend information, local information, and multidimensional structural information, with the modeling of local information based on the differential of the interpolated sequence. However, such kind of methods generate the whole complete time series with only a given initial value, and fail to fully leverage the contextual information of the data. In the second kind of methods, ODE-RNN Rubanova et al. (2019) and ODE-LSTM Lechner & Hasani (2020) both use gating mechanisms in the update step to encode new information. CADN Chien & Chen (2022) is based on the ODE-RNN framework and incorporates an attention mechanism to strengthen modeling. GRU-ODE-Bayes De Brouwer et al. (2019) and Neural Jump ODE Jia & Benson (2019) use Bayesian estimation methods in the update step. GNODE Poli et al. (2019) models potential state changes using the extension of ODE on graphs, GDE, in the integral step, while TGNN4I Oskarsson et al. (2023) assumes that ODE changes are linear; both use a GNN+GRU approach for updates. ANDE De Brouwer & Krishnan (2023) introduces the HIPPO matrix in the integral step, working in conjunction with NODE to strengthen the modeling of historical information, and updates directly by assignment in the update step. Nevertheless, the process of these methods results in fragmented latent representations, which can not accurately represent the true dynamics of continuous time series.

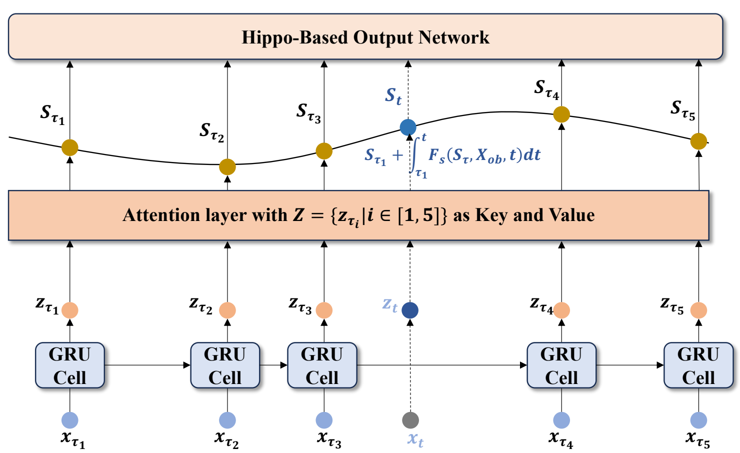


Figure 2: Solution overview. Irregular sampled observations are fed into a neural network to generate Z, which serves as key and value of the attention layer to generate differentiable hidden state. A Hippo-based output network is employed to generate output of the whole framework.

3 METHOD

3.1 MODELING TIME SERIES WITH ODE

We denote the irregular time series of interest as $X_{ob} = \{(x_t, t) | x_t \in \widetilde{X}, t \in T_{ob}\}$, where observations $\widetilde{X} = \{x_{\tau_1}, x_{\tau_2}, \cdots, x_{\tau_n}\}$ are sampled irregularly at time points $T_{ob} = \{t_1, t_2, \cdots, t_n\}$, n is the number of observations. Notably, considering irregular time series is sampled from continuous time series, we may have continuous time series as $X_{co} = \{x_t | x_t, t \in \mathbb{R}\}$.

We model the continuous dynamics of hidden state of irregular time series with ordinary differential equation.

$$\frac{\mathrm{d}S_t}{\mathrm{d}t} = F_s(S_t, X_{ob}, t) \tag{1}$$

where S_t denotes the hidden state of the time series at time t and $F_s(\cdot)$ specifies the dynamics of the hidden state. Given any time t, the hidden state S_t could be attained by the integration of Eq.1 over time as,

$$S_t = S_{\tau_1} + \int_{\tau_1}^{t} F_s(S_\tau, X_{ob}, \tau) d\tau \tag{2}$$

where S_{τ_1} is the initial value of the hidden state. And a readout function can be applied to generate the output of time series at time t.

$$y_t = f_{out}(S_t) \tag{3}$$

3.2 ATTENTIVE DIFFERENTIABLE HIDDEN STATE BASED ON DISCRETE OBSERVATIONS

In this paper, we propose **D**ifferentiable **H**idden **S**tate (DHS) S_t as the continuous dynamics of time series in Eq.1. The proposed DHS is generated from the latent representations of time series, which encodes values of time series and their corresponding time points. Specifically, given any time point t and corresponding data x_t, the latent representation z_t is obtained by a neural network,

$$\psi : (x_t, t, E(x_t)) \rightarrow z_t \tag{4}$$

where $E(x_t)$ refers to the external features corresponding to x_t. In practice, we find introducing historical observations of x_t when obtaining z_t leads to better performance, i.e., we have $E(x_t)$ as $\{x_i | i < t\}$. Therefore, latent representations on all observation time points can be denoted as $Z = [z_{\tau_1}, z_{\tau_2}, \cdots, z_{\tau_n}]^T \in \mathbb{R}^{n \times d}$.

Attention mechanism is applied to generate the differentiable hidden state. Let z_t being Query, and Z being Key and Value. Then we define DHS as

$$\begin{aligned} a_t &= \frac{z_t Z^T}{\sqrt{d}} \\ p_t &= \mathrm{softmax}(a_t) \\ S_t &= p_t Z \end{aligned} \tag{5}$$

where $a_t, p_t \in \mathbb{R}^n, S_t \in \mathbb{R}^d$ and we always have $n > d$. a_t is attention score and indicates the correlations between data at time t and other time, and p_t is the normalization of it. DHS is defined on all observations according to correlations with them.

The above definition of DHS suggests that one can obtain a continuous state space of a time series as in Figure 2, where the hidden state S_t at any time t is correlated to the latent representation z_t of time series at t and the latent representations Z of all irregularly sampled observations X_{ob}. Based on the definition of DHS, the derivative of DHS can be calculated and the differential equation describing the dynamics of DHS can be achieved.

3.3 DERIVATIVE OF DHS

In this section, we aim at achieving the differential equation of DHS as in Eq.1, while giving the detailed form of F_s. According to Eq.5, the derivative of DHS S_t to time t can be calculated using chain rule as,

$$\frac{\mathrm{d}S_t}{\mathrm{d}t} = \frac{\mathrm{d}z_t}{\mathrm{d}t} \frac{Z^T (P_{diag} - p_t^T p_t) Z}{\sqrt{d}} \tag{6}$$

where $P_{diag} = \mathrm{Diag}(p_t)$, $p_t = [p_{t,1}, p_{t,2}, \cdots, p_{t,n}]$ corresponds to normalized attention score of z_t to all observations Z as in Eq.5. Refer to Appendix B.1 for detailed calculation process.

The first term in Eq.6 is intractable. Following NODE, we may apply another neural network ϕ to model it,

$$\frac{\mathrm{d}z_t}{\mathrm{d}t} = \phi(z_t, t) \tag{7}$$

Therefore, we have the derivative of S_t as,

$$\frac{\mathrm{d}S_t}{\mathrm{d}t} = \phi(z_t, t) \frac{Z^T (P_{diag} - p_t^T p_t) Z}{\sqrt{d}} \tag{8}$$

To achieve a differential equation of S_t as in Eq.1, given observations X_{ob}, the derivative of S_t should be only dependent on S_t and t. However, in Eq.8, while Z is transformation of X_{ob}, $\frac{\mathrm{d}S_t}{\mathrm{d}t}$ are dependent on p_t and z_t. In the following, we further transform p_t and z_t into S_t by innovatively computing attention mechanism backwards.

Note that in Eq.5, the dimension of p_t is higher than that of S_t, thus the information is compressed in this step. If we consider equation Eq.5 as a linear system and solve it directly, we will get infinite solutions. To attain a proper S_t, we introduce the theory of generalized inverse. Generalized inverse allows for a unified approach to obtaining solutions for linear system, no matter how many solutions it may have. See detailed introduction in Appendix A.1. In our case, the solution for p_t could be expressed as

$$p_t^T = (Z^T)^\dagger S_t^T + (I_n - (Z^T)^\dagger Z^T)h \tag{9}$$

where h is a random vector of dimension n, and $(Z^T)^\dagger$ is the Moore-Penrose inverse Moore (1920) of Z^T. In most cases, we have $n \gg d$ holds, so we can assume that Z^T has full row rank and thus have $(Z^T)^\dagger = Z(Z^T Z)^{-1}$.

According to the theory of generalized inverse, we could readily obtain the minimum-norm solution $p_t^T = (Z^T)^\dagger S_t^T$. However, a more appropriate solution could be attained by considering the properties of p_t.

In attention mechanism, p_t is always sparse so as to concentrate on certain important time points. We introduce Hoyer Hurley & Rickard (2009) to measure the sparsity of p_t.

Definition 1. *Given a vector $x \in \mathbb{R}^N$, Hoyer could be defined as*

$$\text{Hoyer}(x) = \frac{1}{\sqrt{N} - 1}(\sqrt{N} - \frac{\sum_{i=1}^N x_i}{\sqrt{\sum_{i=1}^N x_i^2}}) \tag{10}$$

As a measure of sparsity, Hoyer has several excellent properties. The larger the Hoyer, the sparser the vector. For specific details, see in Appendix A.2.

From Eq.8, a proper vector h is required to get a sparse p_t. We construct an optimization problem based on Hoyer. Noting that p_t is the result of softmax normalization, so the elements are all positive and the sum of them is 1. Let $J_{1,n}$ and $J_{n,1}$ denote all-one matrices of dimension $1 \times n$ and $n \times 1$ respectively. The sparsity optimization problem is expressed as,

$$\begin{aligned} \max_h \quad & \text{Hoyer}(p_t) \\ s.t. \quad & p \geq 0 \\ & pJ_{n,1} = 1 \end{aligned} \tag{11}$$

Theorem 1. *Optimization problem in Eq.11 could be precisely solved using the KKT conditions. And the time complexity is $\mathcal{O}(2^n)$.*

The detailed proof is given in Appendix B.2. Note in Eq.1, we have to compute p_t at each integration step t, leading to unacceptably high time consumption. In addition, Eq.11 could be approximately solved using iterative methods such as gradient descent. However, the time complexity is still intolerable. Therefore, we relax the conditions to allow for negative values.

Theorem 2. *By introducing negative probability, the optimization problem turns into Eq.12, and could be precisely solved by Lagrange multipliers. The time complexity could be reduced from $\mathcal{O}(2^n)$ to $\mathcal{O}(n)$.*

By relaxing the conditions to allow for negative values, the sparsity optimization problem turns to,

$$\begin{aligned} \max_h \quad & \text{Hoyer}(p_t) \\ s.t. \quad & pJ_{n,1} = 1 \end{aligned} \tag{12}$$

The new problem can be solved precisely using Lagrange multipliers. Detailed proof and calculating process can be found in Appendix B.3. The final result of the above optimization problem is,

$$p_t^T = b_p - \frac{(J_{1,n} b_p - 1) A_p J_{n,1}}{J_{1,n} A_p J_{n,1}} \tag{13}$$

where $b_p = (Z^T)^\dagger S_t^T$ and $A_p = I_n - (Z^T)^\dagger Z^T$.

Next, we describe how to express z_t as a function of S_t. As softmax is too complex to be directly given an algebraic expression, we perform a first-order Taylor expansion for it,

$$p_t = \frac{a + J_{1,n}}{(a + J_{1,n})J_{n,1}} \tag{14}$$

Combining Eq.5, Eq.13 and Eq.14, we have

$$\begin{gathered} z_t = \sqrt{d} \cdot a_h (Z^T)^\dagger \\ a_h = h_2^T (I_n - (J_{n,1}p - I_n)(J_{n,1}p - I_n)^\dagger) - J_{1,n} \end{gathered} \tag{15}$$

h_2 is a random vector and could be trained together with the neural network.

Finally, we apply Eq.13 and Eq.15 to Eq.8, then obtain the differential equation of DHS S_t.

3.4 Output

DHS provides a continuous hidden embedding, which could be conveniently used for downstream tasks. Following the conventions of NODE-based methods, one straightforward approach is to directly map DHS through a simple neural network, that is

$$y = f_{out}(S) \tag{16}$$

In the classification tasks, y refers to the label of the time series and S refers to DHS at all integration time points. In the interpolation and extrapolation tasks, y_t at any given time is obtained from the corresponding S_t at the same time point.

DHS can also be easily combined with other methods. Hippo Gu et al. (2020) is an effective representation for time series and able to update through integration. However, Hippo requires a continuous sequence as input, which is exactly what DHS offers. We construct the following system of equations:

$$\begin{aligned} \frac{dr_t}{dt} &= f_r(S_t || c_t || r_t) \\ \frac{dc_t}{dt} &= Ac_t + B(W_r r_t) \\ \frac{dS_t}{dt} &= F_s(S_t, X_{ob}, t) \end{aligned} \tag{17}$$

where c_t is Hippo representation. The information is concentrated on r_t and then output through a simple neural network just like Eq.16.

4 Experiment

In this section, we evaluate our model on synthetic and real-world datasets for classification, interpolation and extrapolation tasks. We compare the performance of DHS to state-of-the-art methods and validate the effectiveness of DHS for irregular time series.

4.1 Datasets

We implement our approach on four datasets, namely synthetic periodic dataset, dynamical systems, USHCN and Physionet.

Synthetic periodic dataset is generated by the algebraic equation $x(t) = \sin(t + \phi) * \cos(3 * (t + \phi))$ with time $t \in (0, 10)$ and phase $\phi \sim N(0, 2\pi)$. We simulate 1000 time series and create a binary label $y = \mathrm{I}(\mathrm{x}(5) > 0.5)$. To make the time series irregular, we sample from them according a Poisson process with rate 70%. The dataset is divided into training, testing and validation sets with ratio of 50% : 25% : 25%.

Dynamical systems are a widely studied type of time series that require models to learn the underlying dynamics of the processes. We consider one of the representation of the most complex dynamical

Model	Synthetic	Lorenz63	Lorenz96
attention-based			
mTAN	0.757 ± 0.030	0.718 ± 0.066	0.713 ± 0.072
ContiFormer	0.992 ± 0.006	0.989 ± 0.004	0.987 ± 0.004
SSM-based			
HiPPO-obs	0.758 ± 0.023	0.837 ± 0.034	0.949 ± 0.007
HiPPO-RNN	0.742 ± 0.008	0.804 ± 0.023	0.944 ± 0.008
S4	0.994 ± 0.003	0.911 ± 0.005	0.948 ± 0.016
RNN-based			
GRU	0.848 ± 0.044	0.805 ± 0.017	0.834 ± 0.058
GRU-D	0.897 ± 0.028	0.859 ± 0.015	0.864 ± 0.048
ODE-based			
ODE-RNN	0.870 ± 0.032	0.813 ± 0.013	0.954 ± 0.012
Latent-ODE	0.782 ± 0.014	0.713 ± 0.021	0.762 ± 0.024
GRU-ODE-Bayes	0.968 ± 0.004	0.825 ± 0.031	0.925 ± 0.004
NRDE	0.773 ± 0.111	0.604 ± 0.046	0.606 ± 0.112
PolyODE	0.994 ± 0.003	0.992 ± 0.000	0.984 ± 0.002
Ours			
DHS	$\mathbf{0.997 \pm 0.001}$	$\mathbf{0.993 \pm 0.001}$	$\mathbf{0.991 \pm 0.003}$

Table 1: Classification performance on synthetic dataset and dynamical systems. Top-1 accuracy is reported.

systems, chaotic attracters. Chaotic attracters are sensitive to initial conditions and small noises might result in exponentially diverging trajectories. We construct Lorenz63 and Lorenz96 systems and remove the last dimension to make it never fully observed. To make it more irregular, we further sample from them using a Poisson process with rate 30%. Similarly, the dataset is divided into training, testing and validation sets with ratio of $50\% : 25\% : 25\%$.

United States Historical Climatology Network (USHCN) Menne et al. (2009) contains over 150 years of daily climate data from the United States, including five different variables (precipitation, snowfall, snow depth, minimum and maximum temperature) from 1218 weather stations. Following the preprocessing procedure of GRU-ODE-Bayes, we select the data of 1168 stations over 4 years. Due to equipment failure or the occasional collection of certain metrics (e.g. snow depth), the dataset is very sparse. We further increase the irregularity by removing half of the time points and randomly removing 20% of the observations. Divide the dataset into 60% for training, 20% for testing, and 20% for validation.

PhysioNet Challenge 2012 (Physionet) Citi & Barbieri (2012) includes the physical conditions of 8000 patients in the ICU during the first 48 hours, including 37 different indicators, such as serum glucose, heart rate, platelets, etc. Following the preprocessing procedure in ODE-RNN Rubanova et al. (2019), we round the observations to 6 minutes. Divide the dataset into 60% for training, 20% for testing, and 20% for validation.

4.2 Baselines

We compare the performance of DHS with a variety of baselines, including attention-based model (mTAN Shukla & Marlin (2021), ContiFormer Chen et al. (2024)), SSM-based models (HIPPO-obs, HIPPO-RNN Gu et al. (2020), S4 Gu et al. (2021)), RNN-based models (GRU Chung et al. (2014), GRU-D Che et al. (2018)) and ODE-based models (Latent-ODE, ODE-RNN Rubanova et al. (2019), GRU-ODE-Bayes De Brouwer et al. (2019), NRDE Morrill et al. (2021), PolyODE De Brouwer & Krishnan (2023)).

Model	USHCN interpolation	USHCN extrapolation	Physionet interpolation	Physionet extrapolation
attention-based				
mTAN	1.766 ± 0.009	2.360 ± 0.038	0.208 ± 0.025	0.340 ± 0.020
ContiFormer	0.837 ± 0.057	1.634 ± 0.082	0.212 ± 0.023	0.376 ± 0.034
SSM-based				
HiPPO-obs	1.268 ± 0.051	2.417 ± 0.068	0.323 ± 0.061	0.855 ± 0.024
HiPPO-RNN	1.172 ± 0.061	2.324 ± 0.031	0.293 ± 0.068	0.769 ± 0.053
S4	0.823 ± 0.016	1.504 ± 0.063	0.229 ± 0.023	0.535 ± 0.067
RNN-based				
GRU	1.068 ± 0.073	2.071 ± 0.015	0.364 ± 0.088	0.880 ± 0.140
GRU-D	0.994 ± 0.011	1.718 ± 0.015	0.338 ± 0.027	0.873 ± 0.071
ODE-based				
ODE-RNN	0.831 ± 0.008	1.955 ± 0.467	0.236 ± 0.009	0.467 ± 0.006
Latent-ODE	1.798 ± 0.009	2.034 ± 0.005	0.212 ± 0.027	0.725 ± 0.072
GRU-ODE-Bayes	0.841 ± 0.142	5.437 ± 1.020	0.521 ± 0.038	0.798 ± 0.071
NRDE	0.961 ± 0.051	1.923 ± 0.607	0.434 ± 0.077	0.819 ± 0.037
PolyODE	0.806 ± 0.017	1.842 ± 0.440	0.205 ± 0.041	0.598 ± 0.034
Ours				
DHS	$\mathbf{0.775 \pm 0.023}$	$\mathbf{0.869 \pm 0.043}$	$\mathbf{0.182 \pm 0.074}$	$\mathbf{0.328 \pm 0.054}$

Table 2: Interpolation and extrapolation performance on USHCN and Physionet. Mean square error is reported.

mTAN is a generative method based on variational auto-encoder and use attention mechanism to produce a fixed-length representation for time series of arbitrary length. ContiFormer designs a continuous extension of Transformer, based on multiple smooth curves starting from each observation. HIPPO-obs model directly on observations with the dynamics of the HIPPO matrix, while HIPPO-RNN combine a gated RNN with HIPPO matrix dynamics. S4 extends HIPPO to a higher dimension and obtains an efficient training approach. GRU designs an effective gating mechanism and has been widely applied. GRU-D extends GRU with an exponential decay between observations to accommodate irregular time intervals. ODE-based models models underlying dynamics directly and has been detailed in related works.

4.3 Irregularly Sampled Time Series Classification

Classification is an important application of irregular time series analysis. In our evaluation, we subjected a variety of models to rigorous testing using both synthetic periodic dataset and dynamical systems, employing cross-entropy loss for training purposes. The results, presented in Table 1, reveal that our proposed model, DHS, surpasses a diverse array of existing methods, achieving state-of-the-art performance across all tested datasets. Notably, the attention-based method mTAN, along with the RNN-based methods GRU and GRU-D, were unable to surpass other approaches. This underperformance is attributed to their discrete frameworks, underscoring the significant advantage offered by our model's continuous hidden state representation. While the recent PolyODE model demonstrates a general capacity to extract temporal information from time series, it falls short in accurately capturing the subtleties of the underlying dynamics when compared to the robust capabilities of our proposed DHS.

4.4 Interpolation and Extrapolation

We employ the USHCN and Physionet datasets to evaluate the performance of models on interpolation and extrapolation tasks. For interpolation, our goal is to reconstruct the complete time series from a subset of available observations. Conversely, in the extrapolation task, we divide the time series into two equal parts: the first half is utilized for model training, while the full sequence is employed for making predictions.

The results, detailed in Table 2, are presented in terms of mean squared error (MSE), scaled by a factor of 10^{-2}. Our proposed model, DHS, not only outperforms alternative methods but also excels particularly in the extrapolation task. This superior performance suggests that DHS is adept at capturing the intrinsic dynamics of the time series, a capability that significantly aids in the model's ability to forecast future trends accurately.

4.5 Analysis of time consumption

We compare the time complexity of our method with representative baselines, see in the following table. Also, the time consumption of our model and baselines in one training epoch on USHCN dataset is listed in Table 3. In the table, we have n denoted the number of time points with observations, d denoted the dimension of feature of observations, d_c denoted the dimension of Hippo matrix and L denoted the integration steps. The scale of d_c is typically similar to that of d, and L is always less to n. SSM-based models, e.g., HiPPO-obs, are efficient linear models and usually needs at least $O(d_c^2 L)$ time. RNN-based models are simple but less efficient, which usually need only $O(d^2 n)$ time. ODE-based models needs at least $O(d^2 L)$ time, and extra time consumption related to the specific design of the model. Methods combining attention mechanism with NODE usually needs $O(d^2 n^2 L)$ time consumption. Our model designs reduce it to $O(d^2 nL)$, with the similar time complexity as normal attention-based models. We can find that, our model achieves impressive performance gain by introducing continuous attention mechanism while requiring acceptable additional time consumption.

Model	Complexity	Time (s/epoch)
ContiFormer	$O(d^2 n^2 L)$	154
HiPPO-obs	$O(d_c^2 L)$	86
GRU-D	$O(d^2 n)$	232
ODE-RNN	$O(d^2 L)$	91
Latent-ODE	$O(d^2 L)$	110
PolyODE	$O(d_c^2 d^2 L)$	131
DHS	$O(d_c^2 nL)$	126

Table 3: Time consumption comparison.

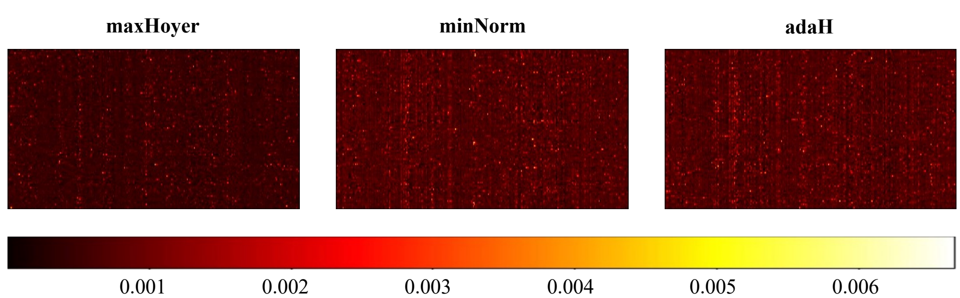


Figure 3: Visualization of attention scores obtained by different methods. Fewer points of lighter color means greater sparsity.

4.6 Analysis of Hoyer metric

To assess the impact of maximizing the Hoyer metric (maxHoyer) on model performance, we conducted a comparative analysis with two alternative approaches for determining p_t: one that employs p_t with the minimum norm (minNorm), and another that treats h in Eq. 9 as an adaptable parameter co-trained with the neural network (adaH). Figure 3 illustrates the gray-scale maps of p_t as derived from these various methods, while Table 4 presents the mean squared error (MSE), scaled by 10^{-2}.

The results indicate that p_t obtained through the maximization of the Hoyer metric not only exhibits greater sparsity but also delivers superior performance on the dataset. This finding emphasizes the Hoyer metric's efficacy in promoting sparsity, which in turn is beneficial for capturing the complex interdependencies among highly correlated points within a time series.

Model	USHCN interpolation	USHCN extrapolation	Physionet interpolation	Physionet extrapolation
maxHoyer	**0.775 ± 0.023**	**0.869 ± 0.043**	**0.182 ± 0.074**	**0.328 ± 0.054**
minNorm	0.804 ± 0.020	0.922 ± 0.034	0.201 ± 0.076	0.346 ± 0.049
adaH	0.798 ± 0.038	0.913 ± 0.081	0.197 ± 0.094	0.351 ± 0.063

Table 4: DHS performance with p_t obtained in different approaches on USHCN and Physionet. maxH, minN, trainP respectively refers to p_t calculated by maximization of Hoyer, minimization of norm and training as a parameter.

Interestingly, the performance of p_t derived from both the minimum norm approach (minNorm) and the adaptive parameter training (adaH) is quite comparable. This similarity in performance might stem from the necessity for h to be closely aligned with the data characteristics for each batch. If h is not well-correlated with the data, its capacity to absorb meaningful information is constrained. The p_t resulting from the Hoyer metric maximization is inherently connected to S_t, aligning with the requirement for data-sensitive h values and thus explaining its enhanced performance.

4.7 Ablation study

We come up with three more ablation studies on the input neural network, the output mechanism, and multi-head attention in this section. For the input neural network, we compare the performance of GRU and MLP on dynamical systems. When using MLP, we actually have $E(x_t)$ in expression 4 as $\varnothing$. For the output mechanism, we compare the performance of using and not using Hippo mechanism. The result is shown in Fig. 5. Synthetic, Lorenz96 and USHCN are employed here. It is shown that using GRU as input layer could better capture the information over time and Hippo is even more important on generating prediction. For multi-head attention, we compare the performance of model with different heads on Physionet dataset. The result is shown in Fig. 4, which illustrates that the improvement from multi-head attention is limited, but it incurs additional time consumption overhead.

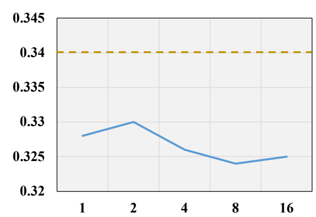
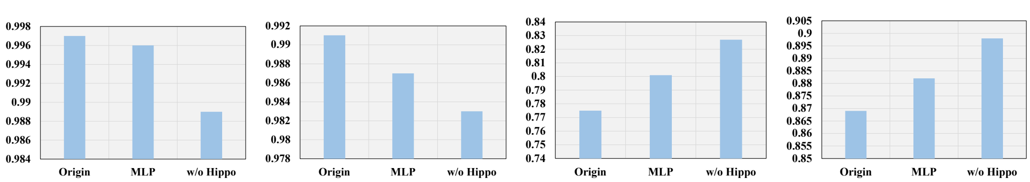


Figure 4: Extrapolation performance on Physionet with different number of heads in attention.



(a) Synthetic (b) Lorenz96 (c) USHCN interpolation (d) USHCN extrapolation

Figure 5: Ablation study of input neural network and output mechanism. Synthetic, Lorenz96 and USHCN are employed here.

5 Conclusion

This paper tackles a significant challenge faced by current neural ODE methods: their inability to seamlessly integrate contextual information while preserving the continuity of the latent dynamics of irregular time series. To overcome this, we introduce an attention-based differential hidden state space, leveraging irregularly sampled observations as Key and Value matrices to enrich the model's context awareness. Building upon this novel hidden state space, we employ the theory of generalized inverses to formulate an ODE that encapsulates the dynamics of the hidden states over time. To enhance the precision of temporal relationships, we incorporate the Hoyer metric, aiming to maximize the sparsity of attention scores during the generation of hidden states. Our approach has been rigorously compared with existing state-of-the-art methods on both synthetic and real-world datasets, with experimental results consistently showcasing the superior effectiveness of our model in irregular time series analysis.

REFERENCES

Lei Bai, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. Adaptive graph convolutional recurrent network for traffic forecasting. *Advances in neural information processing systems*, 33:17804–17815, 2020.

Stefan Bauer, Bernhard Schölkopf, and Jonas Peters. The arrow of time in multivariate time series. In *International Conference on Machine Learning*, pp. 2043–2051. PMLR, 2016.

George EP Box and Gwilym M Jenkins. Some recent advances in forecasting and control. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 17(2):91–109, 1968.

Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. Recurrent neural networks for multivariate time series with missing values. *Scientific reports*, 8(1):6085, 2018.

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

Yuqi Chen, Kan Ren, Yansen Wang, Yuchen Fang, Weiwei Sun, and Dongsheng Li. Contiformer: Continuous-time transformer for irregular time series modeling. *Advances in Neural Information Processing Systems*, 36, 2024.

Jen-Tzung Chien and Yi-Hsiang Chen. Learning continuous-time dynamics with attention. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(2):1906–1918, 2022.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Luca Citi and Riccardo Barbieri. Physionet 2012 challenge: Predicting mortality of icu patients using a cascaded svm-glm paradigm. In *2012 Computing in Cardiology*, pp. 257–260. IEEE, 2012.

Edward De Brouwer and Rahul G Krishnan. Anamnesic neural differential equations with orthogonal polynomial projections. *arXiv preprint arXiv:2303.01841*, 2023.

Edward De Brouwer, Jaak Simm, Adam Arany, and Yves Moreau. Gru-ode-bayes: Continuous modeling of sporadically-observed time series. *Advances in neural information processing systems*, 32, 2019.

Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. Hippo: Recurrent memory with optimal polynomial projections. *Advances in neural information processing systems*, 33: 1474–1487, 2020.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

Calypso Herrera, Florian Krach, and Josef Teichmann. Neural jump ordinary differential equations: Consistent continuous-time prediction and filtering. *arXiv preprint arXiv:2006.04727*, 2020.

Niall Hurley and Scott Rickard. Comparing measures of sparsity. *IEEE Transactions on Information Theory*, 55(10):4723–4741, 2009.

Junteng Jia and Austin R Benson. Neural jump stochastic differential equations. *Advances in Neural Information Processing Systems*, 32, 2019.

Konstantinos Kalpakis, Dhiral Gada, and Vasundhara Puttagunta. Distance measures for effective clustering of arima time-series. In *Proceedings 2001 IEEE international conference on data mining*, pp. 273–280. IEEE, 2001.

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

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

Jianguo Li, Zhanxing Zhu, et al. Neural lad: A neural latent dynamics framework for times series modeling. *Advances in Neural Information Processing Systems*, 36, 2024.

Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in neural information processing systems*, 32, 2019.

Matthew J Menne, Claude N Williams Jr, and Russell S Vose. The us historical climatology network monthly temperature data, version 2. *Bulletin of the American Meteorological Society*, 90(7): 993–1008, 2009.

Eliakim H Moore. On the reciprocal of the general algebraic matrix. *Bulletin of the american mathematical society*, 26:294–295, 1920.

James Morrill, Cristopher Salvi, Patrick Kidger, and James Foster. Neural rough differential equations for long time series. In *International Conference on Machine Learning*, pp. 7829–7838. PMLR, 2021.

Joel Oskarsson, Per Sidén, and Fredrik Lindsten. Temporal graph neural networks for irregular data. In *International Conference on Artificial Intelligence and Statistics*, pp. 4515–4531. PMLR, 2023.

Michael Poli, Stefano Massaroli, Junyoung Park, Atsushi Yamashita, Hajime Asama, and Jinkyoo Park. Graph neural ordinary differential equations. *arXiv preprint arXiv:1911.07532*, 2019.

Yao Qin, Dongjin Song, Haifeng Chen, Wei Cheng, Guofei Jiang, and Garrison Cottrell. A dual-stage attention-based recurrent neural network for time series prediction. *arXiv preprint arXiv:1704.02971*, 2017.

Syama Sundar Rangapuram, Matthias W Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. Deep state space models for time series forecasting. *Advances in neural information processing systems*, 31, 2018.

Yongming Rao, Wenliang Zhao, Zheng Zhu, Jiwen Lu, and Jie Zhou. Global filter networks for image classification. *Advances in neural information processing systems*, 34:980–993, 2021.

Yulia Rubanova, Ricky TQ Chen, and David K Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. *Advances in neural information processing systems*, 32, 2019.

David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International journal of forecasting*, 36(3): 1181–1191, 2020.

Rajat Sen, Hsiang-Fu Yu, and Inderjit S Dhillon. Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting. *Advances in neural information processing systems*, 32, 2019.

Satya Narayan Shukla and Benjamin M Marlin. Multi-time attention networks for irregularly sampled time series. *arXiv preprint arXiv:2101.10318*, 2021.

Kerem Sinan Tuncel and Mustafa Gokce Baydogan. Autoregressive forests for multivariate time series modeling. *Pattern recognition*, 73:202–215, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, 34:22419–22430, 2021.

Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, pp. 3634–3640, Stockholm, Sweden, July 2018. International Joint Conferences on Artificial Intelligence Organization. ISBN 978-0-9992411-2-7. doi: 10.24963/ijcai.2018/505. URL `https://www.ijcai.org/proceedings/2018/505`.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? *arXiv preprint arXiv:2205.13504*, 2022.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 11106–11115, 2021.

Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International conference on machine learning*, pp. 27268–27286. PMLR, 2022.

Simiao Zuo, Haoming Jiang, Zichong Li, Tuo Zhao, and Hongyuan Zha. Transformer hawkes process. In *International conference on machine learning*, pp. 11692–11702. PMLR, 2020.

A Background

A.1 Generalized Inverse

A.1.1 Definition of Generalized Inverse

The purpose of constructing a generalized inverse matrix is to obtain a matrix that can serve as an inverse in some sense for a wider class of matrices than invertible matrices. Suppose $A \in C^{m\times n}$ is any complex matrix, if there exists a complex matrix $G \in C^{n\times m}$ such that at least one of the following conditions holds,

- $AGA = A$
- $GAG = G$
- $(AG)^H = AG$
- $(GA)^H = GA$

then G is called a generalized inverse matrix, and the four equations above are called Moore-Penrose(M-P) equations. Furthermore, G is called the Moore-Penrose inverse of A if G satisfies all of the four M-P equations, denoted as $G \in A\{1,2,3,4\}$. In general, if G satisfies the i_1-th, i_2-th, $\cdots$, i_k-th ($1 \le k \le 4$) one of the four M-P equations, then G is a weak inverse of A, denoted as $G \in A\{i_1, i_2, \cdots, i_k\}$.

Usually there exists different notations for the commonly used generalized inverse.

- $A\{1\}$ is called the minus sign inverse, denoted as A^-
- $A\{1,2\}$ is called the reflecsive minus sign inverse, denoted as A^-_r
- $A\{1,3\}$ is called the least square generalized inverse, denoted as A^-_l
- $A\{1,4\}$ is called the least norm generalized inverse, denoted as A^-_m
- $A\{1,2,3,4\}$ is called the Moore-Penrose inverse, denoted as $A^\dagger$

A.1.2 Application in Linear Equation Systems

Consider a non-homogeneous system of linear equations $Ax = b$, where $A \in C^{m\times n}$, $b \in C^m$ are given, and $x \in C^n$ is an unknown vector. If $rank(A|b) = rank(A)$, then the system $Ax = b$ has a solution and we say the system is compatible. If $rank(A|b) \neq rank(A)$, then the system $Ax = b$ has no solution and the system is incompatible.

Given a system $Ax = b$, whether it is solvable or not, we can discuss the solution using the Moore-Penrose inverse $A^\dagger$. We assume a random $n \times 1$ vector h. When $Ax = b$ is compatible, $x = A^\dagger b + (I - A^\dagger A)h$ is the general solution, and $x = A^\dagger b$ is the least norm solution. When $Ax = b$ is incompatible, $x = A^\dagger b + (I - A^\dagger A)h$ is the least square solution.

A.1.3 Computing Moore-Penrose Inverse

A common approach to compute Moore-Penrose Inverse is through singular value decomposition. If A is not a zero matrix, then A has a singular value decomposition $A = VDU^T$. Let $G = UD^\dagger V^T$, where $D^\dagger$ is D but changing all the non-zero elements into their reciprocals. It's easy to verify that G is the Moore-Penrose inverse of A.

Another method to calculate the Moore-Penrose inverse is through full-rank decomposition. Suppose rank($A_{n\times m}$)=r, then there exists two full rank matrices $B_{n\times r}$ and $C_{r\times m}$ such that $A = BC$. The Moore-Penrose inverse of A can be expressed as

$$A^\dagger = C^T(CC^T)^{-1}(B^TB)^{-1}B^T$$

From the equation above, we may easily conclude that A is a full-rank square matrix if and only if $A^\dagger = A^{-1}$. $A_{n\times m}$ is a column full-rank matrix if and only if $A^\dagger A = I_m$, then $A^\dagger = (A^TA)^{-1}A^T$. $A_{n\times m}$ is a row full-rank matrix if and only if $AA^\dagger = I_m$, then $A^\dagger = A^T(AA^T)^{-1}$.

A.2 Hoyer Sparsity Metric

Given a vector $x \in \mathbb{R}^N$, Hoyer is a sparsity metric defined as

$$\text{Hoyer}(x) = \frac{1}{\sqrt{N}-1}(\sqrt{N} - \frac{\sum_{i=1}^N x_i}{\sqrt{\sum_{i=1}^N x_i^2}}) \tag{18}$$

It has been proved that Hoyer satisfies following sparse criteria Hurley & Rickard (2009),

(a) $\forall \alpha, x_i, x_j$ such that $x_i > x_j, 0 < \alpha < \frac{x_i - x_j}{2}$, we have $\text{Hoyer}([x_1, \cdots, x_i - \alpha, \cdots, x_j + \alpha, \cdots]) < \text{Hoyer}(x)$.

(b) $\forall \alpha \in \mathbb{R}, \alpha > 0$, we have $\text{Hoyer}(\alpha x) = \text{Hoyer}(x)$.

(c) $\forall i, \exists \beta > 0$, such that $\forall \alpha > 0$, we have $\text{Hoyer}([x_1, \cdots, x_i + \beta + \alpha, \cdots]) > \text{Hoyer}([x_1, \cdots, x_i + \beta, \cdots])$.

(d) $\text{Hoyer}(x||0) > \text{Hoyer}(x)$, where $||$ denotes concatenation.

Criteria (a) implies that if the sum of the vector remains constant, then the more uniformly distributed, the less sparse the vector will become. Criteria (b) suggests that sparsity is a relative property. Multiplying all elements by the same factor does not alter the sparsity. Criteria (c) finds a main element. When the main element is large enough, it is able to determine the sparsity of the vector. Criteria (d) naturally follows from the definition of sparsity.

B Proof

B.1 Derivative of DHS

S_t satisfies the following equations

$$
\begin{aligned}
a_t &= \frac{z_t Z^T}{\sqrt{d}} \\
p_t &= softmax(a_t) \\
S_t &= p_t Z
\end{aligned}
$$

We want to compute $\frac{dS_t}{dt}$. For convenience, let $z_{\tau_i} = z_i$, then $Z = (z_1^T, z_2^T, \cdots, z_n^T)^T$. Noting that $\forall i$, z_i is independent of t. The derivative of softmax is

$$
\frac{\partial p_j}{\partial a_i} = \begin{cases} p_j(1-p_j), & i = j \\ -p_i p_j, & i \neq j \end{cases}
$$

Then

$$
\begin{aligned}
\frac{dS_t}{dt} &= \sum_{j=1}^{n} (\frac{dp_j}{dt} z_j + p_j \frac{dz_j}{dt}) \\
&= \sum_{j=1}^{n} \frac{dp_j}{dt} z_j
\end{aligned}
$$

where

$$
\begin{aligned}
\frac{dp_j}{dt} &= \sum_{i=1}^{n} \frac{\partial p_j}{\partial a_i} \frac{da_i}{dt} \\
&= p_j(1-p_j)\frac{da_j}{dt} - \sum_{i \neq j} p_i p_j \frac{da_i}{dt} \\
&= p_j \frac{da_j}{dt} - \sum_{i=1}^{n} p_i p_j \frac{da_i}{dt} \\
&= p_j \frac{dz_t}{dt} \frac{z_j^T}{\sqrt{d}} - \sum_{i=1}^{n} p_i p_j \frac{dz_t}{dt} \frac{z_i^T}{\sqrt{d}}
\end{aligned}
$$

Then

$$
\begin{aligned}
\frac{dS_t}{dt} &= \frac{dz_t}{dt} (\sum_{i=1}^{n} p_i \frac{z_i^T z_i}{\sqrt{d}} - \sum_{i=1}^{n} \sum_{j=1}^{n} p_i p_j \frac{z_i^T z_j}{\sqrt{d}}) \\
&= \frac{dz_t}{dt} \frac{1}{\sqrt{d}} (Z^T (P_{diag} - p_t^T p_t) Z)
\end{aligned}
$$

where $P_{diag} = \begin{pmatrix} p_1 & & \\ & \ddots & \\ & & p_n \end{pmatrix}$.

B.2 Precise Solution of Sparsity Optimization Problem

The sum of p is 1, so the optimization problem could be simplified as

$$
\begin{aligned}
\max_{h} \quad & pp^T \\
s.t. \quad & p \geq 0 \\
& J_{1,n} p = 1
\end{aligned}
$$

where

$$
p^T = (Z^T)^\dagger S_t^T + (I_n - (Z^T)^\dagger Z^T) h
$$

For simplicity, let $b = (Z^T)^\dagger S_t^T$, $A = I_n - (Z^T)^\dagger Z^T$. The standard form of problem could be written as

$$\begin{aligned} \min_h \quad & -b^T b - h^T A h \\ s.t. \quad & -b - Ah \leq 0 \\ & J_{1,n}(b + Ah) = 1 \end{aligned}$$

The Lagrange function is defined as

$$L(h, \lambda, \mu) = -b^T b - h^T A h + \lambda(1 - J_{1,n}(b + Ah)) + \mu(-b - Ah)$$

Then the KKT conditions are

$$\begin{aligned} & \nabla_h L = -2Ah - \lambda A J_{n,1} - A\mu = 0 \\ & 1 - J_{1,n}(b + Ah) = 0 \\ & -b - Ah \leq 0 \\ & \mu \geq 0 \\ & \mu_{diag}(-b - Ah) = 0 \end{aligned}$$

where $\mu_{diag} = \begin{pmatrix} \mu_1 & & \\ & \ddots & \\ & & \mu_n \end{pmatrix}$. Let $b = (b_1, \cdots, b_n)$, $A = \begin{pmatrix} A_1 \\ \vdots \\ A_n \end{pmatrix}$. We have

$$\begin{aligned} \mu_i(b_i + A_i h) = 0, \quad i = 1, \cdots, n \\ A_i(2h + \mu + \lambda J_{n,1}) = 0, \quad i = 1, \cdots, n \\ \sum_{i=1}^{n} A_i h + \sum_{i=1}^{n} b_i - 1 = 0. \end{aligned}$$

Suppose there are k non-zero elements in μ, indexes as $\mathfrak{N} = \{n_1, n_2, \cdots, n_k\}$. Let $\alpha_i = sum(A_i)$, $\alpha = sum(A)$. We have

$$2b_{n_i} - A_{n_i}\mu - \lambda\alpha_{n_i} = 0\lambda\alpha = 2(\sum_{i=1}^{n} b_i - 1 - \frac{1}{2}\sum_{i=1}^{k} \mu_{n_i}\alpha_{n_i})$$

Further simplify them into the form that only involves the non-zero terms

$$b_{\mathfrak{N}} = \frac{1}{2}(A_{\mathfrak{N}\mathfrak{N}}\mu_{\mathfrak{N}} + \lambda\alpha_{\mathfrak{N}})\lambda = \frac{2}{\alpha}(J_{1,n}b - 1 - \frac{1}{2}\alpha_{\mathfrak{N}}^T\mu_{\mathfrak{N}})$$

Substitute λ into $b_{\mathfrak{N}}$

$$\frac{1}{2}(A_{\mathfrak{N}\mathfrak{N}} - \frac{1}{\alpha}\alpha_{\mathfrak{N}}\alpha_{\mathfrak{N}}^T)\mu_{\mathfrak{N}} = b_{\mathfrak{N}} - \frac{J_{1,n}b - 1}{\alpha}\alpha_{\mathfrak{N}}$$

Then we can obtain μ, λ, h sequentially. Substitute the results into the inequality constraints of the KTT conditions and verify. If the constraints are satisfied, we fortunately find the solution.

Noting that we have to decide some elements of μ to zero each time. In the worst case, we need to try 2^n times.

B.3 Solution of Relaxed Sparsity Optimization Problem

The optimization problem

$$\begin{aligned} \min_h \quad & -b^T b - h^T A h \\ s.t. \quad & J_{1,n}(b + Ah) = 1 \end{aligned}$$

The Lagrange function is defined as

$$L(h, \lambda) = -b^T b - h^T A h + \lambda(J_{1,n}(b + Ah) - 1)$$

Let derivatives equal 0

$$\nabla_h L = -2Ah + \lambda(J_{1,n}A)^T = 0\nabla_\lambda L = J_{1,n}(b + Ah) - 1 = 0$$

Noting that $A = A^T$, we have

$$2Ah = \lambda A J_{n,1}$$

Substituting it into the second equation, we have

$$\lambda = \frac{2 - 2J_{1,n}b}{J_{1,n}AJ_{n,1}}$$

Then

$$Ah = \frac{(1 - J_{1,n}b)AJ_{n,1}}{J_{1,n}AJ_{n,1}}$$

Finally, we obtain p as

$$p^T = b - \frac{(J_{1,n}b - 1)AJ_{n,1}}{J_{1,n}AJ_{n,1}}$$

The most time-consuming part is the matrix summation of A, which can be computed in $O(n)$ time on modern GPUs optimized for matrix operations.

C Implementation Details

There are three small neural networks in DHS, namely the input map from observations, the output map and the one in the derivative of DHS. We use a one-layer GRU to map the observations into latent states. An MLP with one hidden layer is used for model the dynamics of DHS. For output, we use an MLP with 1 hidden layer. For all datasets, the hidden size of MLPs is set to 32. Integration method is implicit adams, an adaptive method with tiny numerical error. We use early stopping when the validation loss has not increase in 20 epochs. Learning rate is set to 0.001 and weight decay is set to 0.001.

For classification tasks, batch size is set to 128 and the dimension of DHS and information state r_t is set to 16. The integration step of ODE solution is set to 0.05. When we train the model, we have 250 max epochs. For interpolation and extrapolation tasks, batch size is set to 32. The dimension of DHS and information state r_t is set to 32. The integration step of ODE solution is set to 5. Max epochs is set to 100.