Learning to Decouple Complex Systems

Zihan Zhou[1] **Tianshu Yu**[1 2]

Abstract

A complex system with cluttered observations may be a coupled mixture of multiple simple subsystems corresponding to *latent entities*. Such sub-systems may hold distinct dynamics in the continuous-time domain; therein, complicated interactions between sub-systems also evolve over time. This setting is fairly common in the real world but has been less considered. In this paper, we propose a sequential learning approach under this setting by decoupling a complex system for handling irregularly sampled and cluttered sequential observations. Such decoupling brings about not only subsystems describing the dynamics of each latent entity but also a metasystem capturing the interaction between entities over time. Specifically, we argue that the metasystem evolving within a simplex is governed by *projected differential equations (ProjDEs)*. We further analyze and provide neural-friendly projection operators in the context of Bregman divergence. Experimental results on synthetic and real-world datasets show the advantages of our approach when facing complex and cluttered sequential data compared to the state-of-the-art.

1. Introduction

Discovering hidden rules from sequential observations has been an essential topic in machine learning, with a large variety of applications such as physics simulation (Sanchez-Gonzalez et al., 2020), autonomous driving (Diehl et al., 2019), ECG analysis (Golany et al., 2021) and event analysis (Chen et al., 2021), to name a few. A standard scheme is to consider sequential data at each timestamp to be holistic and homogeneous under some ideal assumptions (i.e., only the temporal behavior of one entity is involved in a sequence), under which data/observation is treated as a collection of slices at a different time from a unified system. A series of sequential learning models fall into this category, including variants of recurrent neural networks (RNNs) (Cho et al., 2014; Hochreiter & Schmidhuber, 1997), neural differential equations (DEs) (Chen et al., 2018; Kidger et al., 2020; Rusch & Mishra, 2021; Zhu et al., 2021) and spatial/temporal attention-based approaches (Vaswani et al., 2017; Fan et al., 2019; Song et al., 2017). These variants fit well into the scenarios agreeing with the aforementioned assumptions and are proved effective in learning or modeling for relatively simple applications with clean data sources.

In the real world, a system may not only describe a single and holistic entity but also consist of several *distinguishable* interacting but simple subsystems, where each subsystem corresponds to a physical entity. For example, we can think of the movement of a solar system as the mixture of distinguishable subsystems of the sun and surrounding planets, while interactions between these celestial bodies over time are governed by the laws of gravity. Back centuries ago, physicists and astronomers made enormous efforts to discover the rule of celestial movements from the records of every single body and eventually delivered the neat yet elegant differential equations (DEs) depicting principles of moving bodies and interactions therein. Likewise, nowadays, researchers also developed a series of machine learning models for sequential data with distinguishable partitions (Qin et al., 2017). Two widely adopted strategies for learning the interactions between subsystems are graph neural networks (Iakovlev et al., 2021; Ha & Jeong, 2021; Kipf et al., 2018; Yıldız et al., 2022; Xhonneux et al., 2020) and attention mechanism (Vaswani et al., 2017; Lu et al., 2020; Goyal et al., 2021), while the interactions are typically encoded with "messages" between nodes and pair-wise "attention scores", respectively.

It is worth noting an even more difficult scenario:

- *The data/observation is so cluttered that cannot be readily distinguished into separate parts.*

This can be either due to the way of data collection (e.g., videos consisting of multiple objects) or because there are no explicit physical entities originally (e.g., weather time series). To tackle this, a fair assumption can be introduced that complex observations can be decoupled into several

[1] The Chinese University of Hong Kong, Shenzhen [2] Shenzhen Institute of Artificial Intelligence and Robotics for Society. Correspondence to: Tianshu Yu <yutianshu@cuhk.edu.cn>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

relatively independent modules in the feature space, where each module corresponds to a *latent entity*. Latent entities may not have exact physical meanings, but learning procedures can greatly benefit from such decoupling, as this assumption can be viewed as strong regularization to the system. This assumption has been successfully incorporated in several models for learning from *regularly* sampled sequential data by emphasizing "independence" to some extent between channels or groups in the feature space (Li et al., 2018; Yu et al., 2020; Goyal et al., 2021; Madan et al., 2021). Another successful counterpart in parallel benefiting from this assumption is transformer (Vaswani et al., 2017) which stacks multiple layers of self-attention and point-wise feedforward networks. In transformers, each attention head can be viewed as a relatively independent module, and interaction happens throughout the head re-weighting procedure following the attention scores. Lu et al. (2020) presented an interpretation from a dynamic point of view by regarding a basic layer in the transformer as one step of integration governed by differential equations derived from interacting particles. Vuckovic et al. (2020) extended this interpretation with more solid mathematical support by viewing the forward pass of the transformer as applying successive Markov kernels in a particle-based dynamic system.

We note, however, despite the ubiquity of this setting, there is barely any previous investigation focusing on learning for *irregularly sampled* and *cluttered* sequential data. The aforementioned works either fail to handle the irregularity (Goyal et al., 2021; Li et al., 2018) or neglect the independence/modularity assumption in the latent space (Chen et al., 2018; Kidger et al., 2020). In this paper, inspired by recent advances of neural controlled dynamics (Kidger et al., 2020) and novel interpretation of attention mechanism (Vuckovic et al., 2020), we take a step to propose an effective approach addressing this problem under the dynamic setting. To this end, our approach explicitly learned to decouple a complex system into several latent sub-systems and utilizes an additional meta-system capturing the evolution of interactions over time. Specifically, taking into account the meta-system capturing interactions evolving in a constrained set (e.g., simplex), we further characterized such interactions using projected differential equations (ProjDEs) with neural-friendly projection operators. We argued our **contributions** as follows:

- We provide a novel modeling strategy for sequential data from a system decoupling perspective;
- We propose a novel and natural interpretation of evolving interactions as a ProjDE-based meta-system, with insights into projection operators in the sense of Bregman divergence;
- Our approach is parameter-insensitive and more compatible with other modules and data, thus being flexible to be integrated into various tasks.

Extensive experiments were conducted on either regularly or irregularly sampled sequential data, including both synthetic and real-world settings. It was observed that our approach achieved prominent performance compared to the state-of-the-art on a wide spectrum of tasks. Our code is available at https://github.com/LOGO-CUHKSZ/DNS.

2. Related Work

Sequential Learning. Traditionally, learning with sequential data can be performed using variants of recurrent neural networks (RNNs) (Hochreiter & Schmidhuber, 1997; Cho et al., 2014; Li et al., 2018) under the Markov setting. While such RNNs are generally designed for regular sampling frequency, a more natural line of counterparts lies in the continuous time domain allowing irregularly sampled time series as input. As such, a variety of RNN-based methods are developed by introducing exponential decay on observations (Che et al., 2018; Mei & Eisner, 2017), incorporating an underlying Gaussian process (Li & Marlin, 2016; Futoma et al., 2017), or integrating some latent evolution under ODEs (Rubanova et al., 2019; De Brouwer et al., 2019). A seminal work interpreting forward passing in neural networks as an integration of ODEs was proposed in Chen et al. (2018), followed by a series of relevant works (Liu et al., 2019; Li et al., 2020a; Dupont et al., 2019). As integration over ODEs allows for arbitrary step length, it is natural modeling of irregular time series and proved powerful in many machine learning tasks (e.g., bioinformatics (Golany et al., 2021), physics (Nardini et al., 2021) and computer vision (Park et al., 2021)). (Kidger et al., 2020) studied a more effective way of injecting observations into the system via a mathematical tool called Controlled differential Equation, achieving state-of-the-art performance on several benchmarks. Some variants of neural ODEs have also been extended to discrete structure (Chamberlain et al., 2021b; Xhonneux et al., 2020) and non-Euclidean setting (Chamberlain et al., 2021a).

Learning with Independence. Independence or modular property serves as strong regularization or prior in some learning tasks under static setting (Wang et al., 2020; Liu et al., 2020). In the sequential case, some early attempts over RNNs emphasized implicit "independence" in the feature space between dimensions or channels (Li et al., 2018; Yu et al., 2020). As independence assumption commonly holds in vision tasks (with distinguishable objects), Pang et al. (2020); Li et al. (2020b) proposed video understanding schemes by decoupling the spatiotemporal patterns. For a more generic case where the observations are collected without any prior, Goyal et al. (2021) devised a sequential learning scheme called recurrent independence mechanism

(RIM), and its generalization ability was extensively studied in Madan et al. (2021). Lu et al. (2020) investigated self-attention mechanism (Vaswani et al., 2017) and interpreted it as a nearly independent multi-particle system with interactions therein. Vuckovic et al. (2020) further provided more solid mathematical analysis with the tool of Markov kernel. The study of such a mechanism in the dynamical setting was barely observed.

Learning Dynamics under Constraints. It is practically significant as a series of real-world systems evolve within some manifolds, such as fluid (Vinuesa & Brunton, 2022), coarse-grained dynamics (Kaltenbach & Koutsourelakis, 2020), and molecule modeling (Chmiela et al., 2020). While some previous research incorporates constraints from a physical perspective (Kaltenbach & Koutsourelakis, 2020; Linot & Graham, 2020), an emerging line is empowered by machine learning to integrate or even discover the constraints (Kolter & Manek, 2019; Lou et al., 2020; Goldt et al., 2020). To ensure a system evolves in constraints, efficient projections or pseudo-projections are required, about which Bregman divergence provides rich insights (Martins & Astudillo, 2016; Krichene et al., 2015; Lim & Wright, 2016). Despite these results, to our best knowledge, there is barely any related investigation about neural-friendly projections.

3. Methodology

3.1. Background

In this section, we briefly review three aspects related to our approach. Our approach is built upon the basic sub-system derived from *Neural Controlled Dynamics* (Kidger et al., 2020), while the interactions are modeled at an additional meta-system analogous to *Self-attention* (Lu et al., 2020; Vuckovic et al., 2020), and further interpreted and generalized using the tool of *Projected Differential Equations* (Dupuis & Nagurney, 1993).

Neural Controlled Dynamics. Continuous-time dynamics can be expressed using differential equations $\mathbf{z}'(t) = d\mathbf{z}/dt = f(\mathbf{z}(t), t)$, where $\mathbf{z} \in \mathbb{R}^d$ and t are a d-dimension state and the time, respectively. Function $f : \mathbb{R}^d \times \mathbb{R}_+ \to \mathbb{R}^d$ governs the evolution of the dynamics. Given the initial state $\mathbf{z}(t_0)$, the state at any time t_1 can be evaluated with:

$$\mathbf{z}(t_1) = \mathbf{z}(t_0) + \int_{t_0}^{t_1} f(\mathbf{z}(s), s)\mathrm{d}s \qquad (1)$$

In practice, we aim at learning the dynamics from a series of observations or controls $\{\mathbf{x}(t_k) \in \mathbb{R}^b | k = 0, 1, ...\}$ by parameterizing the dynamics with $f_\theta(\cdot)$ where θ is the unknown parameter to be learned. Thus, a generic dynamics incorporating outer signals $\mathbf{x}$ can be written as:

$$\mathbf{z}(t_1) = \mathbf{z}(t_0) + \int_{t_0}^{t_1} f_\theta(\mathbf{z}(s), \mathbf{x}(s), s)\mathrm{d}s \qquad (2)$$

Rather than directly injecting $\mathbf{x}$ as in Eq. (2), Neural Controlled Differential Equation (CDE) proposed to deal with outer signals with a Riemann–Stieltjes integral (Kidger et al., 2020):

$$\mathbf{z}(t_1) = \mathbf{z}(t_0) + \int_{t_0}^{t_1} \mathbf{F}_\theta(\mathbf{z}(s))\mathbf{x}'(s)\mathrm{d}s \qquad (3)$$

where $\mathbf{F}_\theta : \mathbb{R}^d \to \mathbb{R}^{d \times b}$ is a learnable vector field and $\mathbf{x}'(s) = \mathrm{d}\mathbf{x}/\mathrm{d}s$ is the derivative of signal $\mathbf{x}$ w.r.t. time s, thus "$\mathbf{F}_\theta(\mathbf{z}(s))\mathbf{x}'(s)$" is a matrix-vector multiplication. During implementation, Kidger et al. (2020) argued that a simple cubic spline interpolation on $\mathbf{x}$ allows dense calculation of $\mathbf{x}'(t)$ at any time t and exhibits promising performance. In (Kidger et al., 2020), it is also mathematically shown that incorporating observations/controls following Eq. (3) is with greater representation ability compared to Eq. (2), hence achieving state-of-the-art performance on several public tasks.

Self-attention. It is argued in Lu et al. (2020); Vuckovic et al. (2020) that a basic unit in Transformer (Vaswani et al., 2017) with self-link consisting of one self-attention layer and point-wise feedforward layer amounts to simulating a multi-particle dynamical system. Considering such a layer with n attention-heads (corresponding to n particles), given an attention head index $i \in \{1, 2, ..., n\}$, the update rule of the ith unit at depth l reads:

$$\tilde{\mathbf{z}}_{l,i} = \mathbf{z}_{l,i} + \mathrm{MHAtt}_{W^l_{\mathrm{att}}}(\mathbf{z}_{l,i}, [\mathbf{z}_{l,1}, ..., \mathbf{z}_{l,n}]) \qquad (4a)$$

$$\mathbf{z}_{l+1,i} = \tilde{\mathbf{z}}_{l,i} + \mathrm{FFN}_{W^l_{\mathrm{ffn}}}(\tilde{\mathbf{z}}_{l,i}) \qquad (4b)$$

where $\mathrm{MHAtt}_{W^l_{\mathrm{att}}}$ and $\mathrm{FFN}_{W^l_{\mathrm{ffn}}}$ are multi-head attention layer and feedforward layer with parameters W^l_{att} and W^l_{ffn}, respectively. Eq. (4) can then be interpreted as an interacting multi-particle system:

$$\frac{\mathrm{d}\mathbf{z}_i(t)}{\mathrm{d}t} = F(\mathbf{z}_i(t), [\mathbf{z}_1(t), ..., \mathbf{z}_n(t)], t) + G(\mathbf{z}_i(t)) \qquad (5)$$

where function F corresponding to Eq. (4a) represents the diffusion term and G corresponding to Eq. (4b) stands for the convection term. Notably, the attention score obtained via $\mathrm{softmax}$ in Eq. (4a) is regarded as a Markov kernel. Readers are referred to Lu et al. (2020); Vuckovic et al. (2020) for more details.

Projected DEs. It is a tool depicting the behavior of dynamics where solutions are constrained within a (convex) set. Concretely, given a closed polyhedral $\mathcal{K} \subset \mathbb{R}^n$ and a mapping $H : \mathcal{K} \to \mathbb{R}^n$, we can introduce an operator $\Pi_\mathcal{K} : \mathbb{R}^n \times \mathcal{K} \to \mathbb{R}^n$ which is defined by means of directional derivatives as:

$$\Pi_\mathcal{K}(\mathbf{a}, H(\mathbf{a})) = \lim_{\alpha \to 0_+} \frac{P_\mathcal{K}(\mathbf{a} + \alpha H(\mathbf{a})) - \mathbf{a}}{\alpha} \qquad (6)$$

where $P_{\mathcal{K}}(\cdot)$ is a projection onto $\mathcal{K}$ in terms of Euclidean distance:

$$\|P_{\mathcal{K}}(\mathbf{a})-\mathbf{a}\|_2=\inf_{\mathbf{y}\in\mathcal{K}}\|\mathbf{y}-\mathbf{a}\|_2 \tag{7}$$

Intuitively, Eq. (6) pictures the dynamics of $\mathbf{a}$ driven by function H, but constrained within $\mathcal{K}$. Whenever $\mathbf{a}$ reaches beyond $\mathcal{K}$, it would be projected back using Eq. (7). By extending Eq. (6), (Dupuis & Nagurney, 1993; Zhang & Nagurney, 1995) considered the projected differential equations as follows:

$$\frac{\mathrm{d}\mathbf{a}(t)}{\mathrm{d}t}=\Pi_{\mathcal{K}}(\mathbf{a},H(\mathbf{a})) \tag{8}$$

which allows for discontinuous dynamics on $\mathbf{a}$.

3.2. Learning to Decouple

Our method is built upon the assumption that cluttered sequential observations are composed of several relatively independent sub-systems and, therefore, explicitly learns to decouple them as well as to capture the mutual interactions with a meta-system in parallel. Let the cluttered observations/controlls be $\mathbf{c}(t)\in\mathbb{R}^k$ at time t for $t=1,...,T$, where T is the time horizon. We employ n distinct mappings with learnable parameters (e.g., MLP) to obtain respective controls to each sub-system: $\mathbf{x}_i(t)=p_i(\mathbf{c}(t))\in\mathbb{R}^m$ for $i=1,...,n$. A generic dynamics of the proposed method can be written as:

$$\frac{\mathrm{d}\mathbf{z}_i(t)}{\mathrm{d}t}=f_i\left(\mathbf{z}_i(t),\left[\mathbf{z}_1(t),...,\mathbf{z}_n(t)\right],\mathbf{x}_i(t),\mathbf{a}(t)\right) \tag{9a}$$

$$\frac{\mathrm{d}\mathbf{a}(t)}{\mathrm{d}t}=\Pi_{\mathcal{S}}\left(\mathbf{a}(t),g(\mathbf{a}(t),[\mathbf{z}_1(t),...,\mathbf{z}_n(t)])\right) \tag{9b}$$

where Eq. (9a) and Eq. (9b) refer to the ith sub-system describing the evolution of a single latent entity and meta-system depicting the interactions, respectively. $\mathbf{z}_i(t)\in\mathbb{R}^q$ is the hidden state for the ith subsystem, and $\mathbf{a}$ is a tensor governs the dynamics of the interactions. Here $\Pi_{\mathcal{S}}(\cdot)$ is a projection operator, which projects the evolving trajectory into set $\mathcal{S}$. We introduce such an operator as it is assumed that interactions among latent entities should be constrained following some latent manifold structure. $f_i(\cdot)$ and $g(\cdot)$ are both learnable functions and also the essential roles for capturing the underlying complex dynamics.

Remark 1. It is seen the projection operator $\Pi_{\mathcal{S}}(\cdot)$ and the set $\mathcal{S}$ play important roles in Eq. (9b). For $\Pi_{\mathcal{S}}(\cdot)$, while previous works of ProjDEs only consider L2-induced projection, we propose novel interpretation and extension under Bregman divergence. For $\mathcal{S}$, we consider a probabilistic simplex following the setting in Lu et al. (2020); Vuckovic et al. (2020), though it can be any polyhedral.

According to Eq. (9), we fully decouple a complex system into several components. Although we found some decoupling counterparts in the context of RNNs (Li et al., 2018; Yu et al., 2020) and attention-like mechanism (Lu et al., 2020; Goyal et al., 2021), their decoupling could not be applied to our problem. We elaborate on the details of implementing Eq. (9) in the following.

Learning Sub-systems. Sub-systems corresponding to the latent entities seek to model relatively independent dynamics separately. Specifically, we employ the way of integrating $\mathbf{x}_i$s into Eq. (9a) in a controlled dynamical fashion as in the state-of-the-art method (Kidger et al., 2020):

$$\mathrm{d}\mathbf{z}_i(t)=\mathbf{F}_i\left(\mathbf{z}_i(t),\mathbf{a}(t),[\mathbf{z}_1(t),...,\mathbf{z}_n(t)]\right)\mathrm{d}\mathbf{x}_i(t) \tag{10}$$

where $\mathbf{F}_i(\cdot)\in\mathbb{R}^{q\times m}$ is a learnable vector field. Concretely, if we let $\mathbf{z}(t)=[\mathbf{z}_i(t),...,\mathbf{z}_n(t)]$ be the tensor collecting all sub-systems, the ith sub-system in a self-attention fashion reads:

$$\mathrm{d}\mathbf{z}_i(t)=\mathbf{F}([\mathbf{A}(t)\cdot\mathbf{z}(t)]_i)\mathrm{d}\mathbf{x}_i(t) \tag{11}$$

where $[\cdot]_i$ takes the ith slice from a tensor. Note timestamp t can be arbitrary, resulting in irregularly sampled sequential data. To address this, we follow the strategy in Kidger et al. (2020) by performing cubic spline interpolation on $\mathbf{x}_i$ over observed timestamp t, resulting in $\mathbf{x}_i(t)$ at dense time t. Note that for all sub-systems, different from Eq. (10) we utilize an identical function/network $\mathbf{F}(\cdot)$ as in Eq. (11), but with different control sequence $\mathbf{x}_i(t)=p_i(\mathbf{c}(t))$. Since in our implementation, $p_i(\cdot)$ is a lightweight network such as MLP, this can significantly reduce the parameter size.

Learning Interactions. In our approach, interactions between latent entities are modeled separately as another meta-system. This is quite different from some related methods (Lu et al., 2020; Vuckovic et al., 2020) where sub-systems and interactions are treated as one holistic step of forward integration. For the meta-system describing the interactions in Eq. (9b), two essential components are involved: domain $\mathcal{S}$ and the projection operator Π. In the context of ProjDEs, a system is constrained as $\mathbf{a}(t)\in\mathcal{S}$ for any t. In terms of interactions, a common choice of $\mathcal{S}$ is the stochastic simplex which can be interpreted as a transition kernel (Vuckovic et al., 2020). We allow follow this setting by defining $\mathcal{S}$ be a row-wise stochastic $(n-1)$-simplices:

$$\mathcal{S}\triangleq\{\mathbf{A}\in\mathbb{R}^{n\times n}|\mathbf{A}\mathbf{1}=\mathbf{1},\mathbf{A}_{ij}\geq 0\} \tag{12}$$

where $\mathbf{1}$ is a vector with all 1 entries. $\mathbf{A}=\mathrm{mat}(\mathbf{a})$ is a $n\times n$ matrix. In the sequel, we will use the notation $\mathbf{A}$ throughout. Thus the meta-system capturing the interactions can be implemented as follows:

$$\frac{d\mathbf{A}(t)}{dt}=\Pi_{\mathcal{S}}\left(\mathbf{A}(t),g(\mathbf{A}(t),[\mathbf{z}_1(t),...,\mathbf{z}_n(t)])\right) \tag{13}$$

For the projection operator, we consider two versions shown in Eq. (14). In Eq. (14a), we give a row-wise projection onto the $(n-1)$-simplex with entropic regularization

(Amos, 2019), which has a well-known closed-form solution $\mathrm{softmax}(\cdot)$ appearing in attention mechanism. In Eq. (14b), we adopt a standard L2-induced projection identical to Eq. (7), which leads to sparse solutions (Wainwright et al., 2008). Intuitively, the projection of a point onto a simplex in terms of L2 distance tends to lie on a facet or a vertex of a simplex, thus being sparse.

$$P_{\mathcal{S}}^{\text{soft}}(\mathbf{A}_{j,:}) = \underset{\mathbf{B}\in\mathcal{S}}{\arg\min}\, \mathbf{A}_{j,:}^{\top}\mathbf{B}_{:,j} - \mathbb{H}^{\text{entr}}(\mathbf{B}_{:,j}) \tag{14a}$$

$$\begin{aligned} P_{\mathcal{S}}^{\text{sparse}}(\mathbf{A}_{j,:}) &= \underset{\mathbf{B}\in\mathcal{S}}{\arg\min}\, \mathbf{A}_{j,:}^{\top}\mathbf{B}_{:,j} - \mathbb{H}^{\text{gini}}(\mathbf{B}_{:,j}) \\ &= \underset{\mathbf{B}\in\mathcal{S}}{\arg\min}\, |\mathbf{A}_{j,:} - \mathbf{B}_{:,j}|^2 \end{aligned} \tag{14b}$$

where $\mathbb{H}^{\text{entr}}(\cdot)$ and $\mathbb{H}^{\text{gini}}(\mathbf{y}) = \frac{1}{2}\sum_i \mathbf{y}_i(\mathbf{y}_i - 1)$ are the standard entropy and the gini-entropy, respectively. $\mathbf{A}_{j,:}$ and $\mathbf{B}_{:,j}$ are the ith row and column of $\mathbf{A}$ and $\mathbf{B}$, respectively. While the solution to Eq. (14a) is $\mathrm{softmax}(\mathbf{A})$, Eq. (14b) also has closed-form solution shown in Appendix A.3. Comparing Eq. (14a) to the standard Euclidean projection in Eq. (14b), we note the entropic regularization $\mathbb{H}(\cdot)$ in Eq. (14a) allows for a smoother trajectory by projecting any $\mathbf{A}$ into the interior of $(n-1)$-simplex. We visualize the two versions of projections in Eq. (14) onto 1-simplex from some random points in Fig. 1. One can readily see that Eq. (14b) is an exact projection such that points far from the simplex are projected onto the boundary. However, $\mathrm{softmax}$ is smoother by projecting all points onto a relative interior of 1-simplex without sudden change. In the context of Bregman divergence, different distances can facilitate efficient convergence under different "L-relative smoothness" (Dragomir et al., 2021), which can potentially accelerate the learning of dynamics. We leave this to our future work.

We further discuss some neural-friendly features of Eq. (14a) and (14b) facilitating the neural computation:

(1) First, the neural computational graph can be simplified using projection Eq. (14a). Though Eq. (13) using projection Eq. (14a) defines a projected dynamical system directly on $\mathbf{A}$, we switch to update the system using $\mathbf{L}$ as follows, which is considered to further ease the forward integration. This is achieved by instead modeling the dynamics of the feature before fed into $\mathrm{softmax}(\cdot)$:

$$\mathbf{A}(t) = \mathrm{Softmax}(\mathbf{L}(t)) \tag{15a}$$

$$\mathbf{L}(t) = \mathbf{L}(0) + \int_0^t \frac{\mathrm{d}}{\mathrm{d}s}\frac{\mathbf{Q}(\mathbf{z}(s))\cdot\mathbf{K}^{\top}(\mathbf{z}(s))}{\sqrt{d_k}}\mathrm{d}s, \tag{15b}$$

$$\mathbf{L}(t+\Delta t) = \mathbf{L}(t) + \Delta t \cdot \frac{\mathrm{d}}{\mathrm{d}s}\frac{\mathbf{Q}(\mathbf{z}(s))\cdot\mathbf{K}^{\top}(\mathbf{z}(s))}{\sqrt{d_k}}\bigg|_{s=t} \tag{15c}$$

where $\mathbf{Q}(\cdot)$ and $\mathbf{K}(\cdot)$ correspond to the query and key in the attention mechanism, respectively. $\mathbf{L}(0) = \mathbf{Q}(\mathbf{z}(0)) \cdot \mathbf{K}^{\top}(\mathbf{z}(0))/\sqrt{d_k}$. We show that updating the dynamic of $\mathbf{L}$ following Eq. (15) is equivalent to directly updating $\mathbf{A}$ in Appendix A.2.

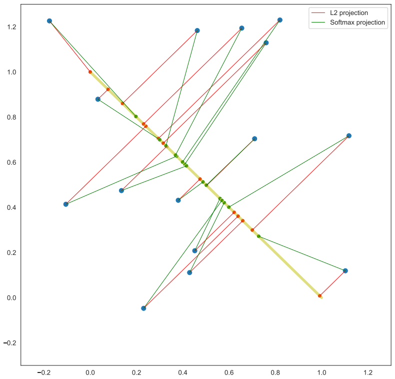

Figure 1. Comparsion of softmax and L2 projection onto a simplex. We see that the softmax projection trends to project onto the "center" of the simplex while the L2 projection trends to project onto the corner.

(2) Second, both the solution to projection Eq. (14b) and its gradient w.r.t. $\mathbf{A}$ are in closed form. See Proposition 1 and Proposition 2 in Appendix A.3 for more details. This, in turn, eases the computational flow in the neural architecture with high efficiency and stability.

Though only two versions of projections are discussed under Bregman divergence, we believe they are sufficiently distinguishable for analyzing the behavior of ProjDEs. For generic neural-friendly projections, we leave them to our future work.

Integration. We employ the standard Euler's discretization for performing the forward integration by updating $\mathbf{z}$ and $\mathbf{A}$ simultaneously with a sufficiently small time step. We term our approach a **d**ecoupling-based **n**eural **s**ystem (**DNS**) using projection Eq. (14a) and $\mathbf{DNS_G}$ using projection Eq. (14b), respectively.

4. Experiments

Sheard & Mostashari (2011) categorized the origins and characteristics of complex systems as dynamic complexity, socio-political complexity, and structural complexity. We carefully select datasets involving the above complexities. The three-body dataset contains rapidly changing interaction patterns (dynamic complexity), the spring dataset stimulates how an individual behaves according to hidden interrelationships (socio-political complexity), and in the human action video dataset where CNNs are frozen, system elements are clustered and required to adapt by RNN to adapt to external needs (structural complexity). We evaluate the performance of DNS on the above synthetic and real-world datasets. More details about the dataset and implementation details can be found in Appendix A.4 and A.6. Throughout all the tables consisting of the results, "-" indicates "not applicable" since RIM cannot handle irregular cases.

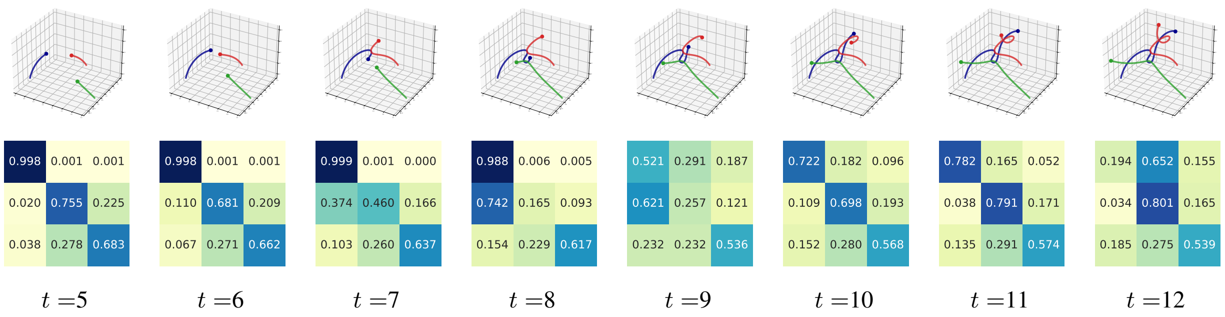

Figure 2. A figure showing the corresponding three-body trajectory (on the top), as well as the evolution over time on interactions (at the bottom) between three **latent sub-systems** in a Three-Body environment. Timestamp from 5 to 12.

Table 1. **Trajectory prediction.** MSE loss of the three body dataset ($\times 10^{-2}$).

MODEL	REGULAR	IRREGULAR
CT-GRU	1.8272	2.4811
NEURALCDE	3.3297	5.0077
RIM	2.4510	-
DNS	**1.7573**	**2.2164**

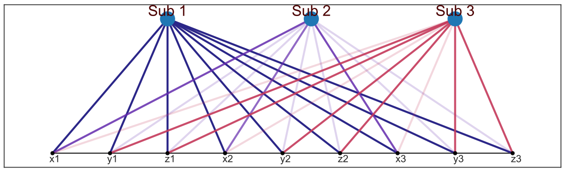

Figure 3. A figure showing the focus of 3 sub-systems on 9-dimensional input of Three Body. The strength of focus is reflected by the thickness of the lines.

Remark 2. In all the experiments, the input feature is treated holistically without any distinguishable parts. For example, in the Three Body dataset, the input is a 9-dimensional vector, with every 3 dimensions (coordinates) from a single object. However, this prior is not fed into any models in comparison. Thus, we do not compare to models integrated with strong prior such as Kipf et al. (2018).

Baselines. We compare DNS with several selected models capturing interactions or modeling irregular time series, including **CT-GRU** (Mozer et al., 2017) using state-decay decay mechanisms, **RIM** (Goyal et al., 2021) updating almost independent modules discretely, and **NeuralCDE** (Kidger et al., 2020) which reports state-of-the-art performance on several benchmarks.

Adapting DNS to the Noisy Case. To allow DNS fitting to noisy and uncertain circumstances, we create a variant by slightly modifying it. This variant is obtained by replacing cubic spline interpolation over $\mathbf{x}_i(t)$ with natural smoothing spline (Green & Silverman, 1993), in consideration of incorporating smoother controls and alleviating data noise. This version is termed as $\textbf{DNS}_\textbf{S}$.

4.1. Three Body

The three-body problem is characterized by a chaotic dynamical system for most randomly initial conditions. A small perturbation may cause drastic changes in the movement. Taking into account the problem's complexity, it is particularly suitable for testing our approach. In this experiment, we consider a trajectory predicting problem given the noisy historical motion of three masses, where gravity causes interactions between them. Therefore, models need to (implicitly) learn both Newton's laws of motion for modeling sub-system dynamics and Newton's law of universal gravitation to decouple the latent interaction. This dataset consists of 50k training samples and 5k test samples. For each sample, 8 historical locations for the regular setting and 6 historical locations (randomly sampled from 8) for the irregular setting in the 3-dimensional space of three bodies are given to predict 3 subsequent locations. To equip with the cluttered setting, the correspondence between dimensions and bodies will not be fed into the learning models, hence a 9-dimensional observation at each time stamp. Models' performance is summarized in Table 1. We can conclude that DNS outperformed all the selected counterparts in both regular and irregular settings. Notably, although our method is built on NeuralCDE, with the decoupling, the performance can be significantly improved. See Table 5 in Appendix A.7.2 for more detailed results.

Visualization and Analysis. We visualize dynamics $\mathbf{A}$ of DNS along the movements of three body system. See Fig. 2 for results. We set the time stamps starting from 5 to 12 to make visualization more informative. It is seen in the beginning ($t = 5, 6$ or even earlier), $\mathbf{A}$ remains stable as the three bodies are apart from each other without intensive interactions. At $t = 7$, $\mathbf{A}$ demonstrates obvious change when two bodies start to the coil. Another body joins in this party at $t = 8$, yielding another moderate change of

Table 2. **Link prediction**. Accuracy on Spring (%). Clean, Noisy, and Short correspond to settings with clean, noisy, and short portion data, respectively. Detailed results for Clean and Noisy are separately summarized in Tab. 7 and Tab. 8 in the appendix.

Model	Clean		Noisy		Short	
	Regular	Irregular	Train&Test	Test	50%	25%
CT-GRU	92.89±0.52	88.47±0.34	92.71±0.55	92.80±0.53	88.67	78.00
NeuralCDE	92.47±0.06	89.74±0.18	90.76±0.08	89.61±0.09	90.75	87.51
RIM	89.73±0.07	-	89.65±0.14	89.64±0.10	80.00	71.26
DNS$_{\text{G}}$	94.31±0.48	**94.25±0.29**	**93.76±0.36**	87.86±0.46	**92.58**	**92.31**
DNS$_{\text{S}}$	**94.44±0.69**	93.60±1.21	93.67±0.57	**92.99±1.30**	91.11	92.13
DNS			93.42±1.05	89.56±0.42		

Table 3. **Link prediction**. Ablation study. (%).

Control	Accuracy (%)
No encoding	91.57
MLP(2×input)	91.51
MLP(16×input)	91.17
DNS (8×MLP(2×input))	**95.38**

$\mathbf{A}$. When flying apart, one body seems more independent, while another two keep entangled together. These are well reflected via the meta-system $\mathbf{A}$. To further see how the holistic 9-dimensional input is decoupled into sub-systems $\mathbf{z}_i$, we visualize the sub-system focus in Fig. 3 (also see Appendix A.1.1). Interestingly, latent entities (sub-systems) do not correspond to physical entities (three bodies). Instead, the first sub-system puts more focus on the whole input, but the remaining two sub-systems concentrate on the x-axis and y/z-axis, respectively. Though counterintuitive, this unexpected decoupling exhibits good performance. We will investigate how to decouple out physical entities from cluttered observations in our future work.

4.2. Spring

We experiment with the capability of DNS in decoupling the independence in complex dynamics controlled by simple physics rules. We use a simulated system in which particles are connected by (invisible) springs (Kuramoto, 1975; Kipf et al., 2018). Each pair of particles has an equal probability of having an interaction or not. Our task is to use observed trajectory to predict whether there are springs between any pair of two particles, which is analogous to the task of link prediction under a dynamical setting. This can be inferred from whether two trajectories change coherently. The spring dataset consists of 50k training examples and 10k test examples. Each sample has a length of 49. We test a variety of combinations of the number of sub-systems and dimensions of the hidden state. Experimental results are in Table 2. To test the models' noise resistance, we add Gaussian noise to the spring dataset and obtain the noisy spring dataset. We set two scenarios, "Train&Test" and "Test", corresponding to injecting noise at both training and test phases and only at testing phases, respectively. Experimental results are in Table 2.

Clean Spring. From Clean part of Table 2, we see variants of DNS stably outperform all the selected counterparts by a large margin. Especially, under the irregularly sampled data, DNS and DNS$_{\text{G}}$ have a remarkable performance gap with all other methods and maintain reliability as in the regular setting. We believe this is significant since learning from irregularly sampled data is typically much more difficult than learning from normal data.

Noisy Spring. According to Noisy part of Table 2, DNS$_{\text{S}}$ is quite reliable in noisy cases. It seems a smoothing procedure on the controls can be helpful under massive uncertainty. Also, we see that adding noise tends to damage the performance of all methods. This also raises one of our future research directions to investigate how to handle different controls. Without applying a smooth cubic spline, DNS can still have a good performance, which indicates that by decoupling, the model focuses on learning latent interaction patterns, and patterns are less susceptible to noise.

Visualization and Analysis. We also visualize state $\mathbf{A}$ of meta-systems over time in Fig. 4 for Spring. From top to bottom, the first, second and third rows correspond to the trajectory of particles, meta-system state of DNS, and meta-system state of DNS$_{\text{G}}$. One interesting thing we note is that the interactions in DNS$_{\text{G}}$ almost concentrate on the starting portion of all the time stamps. At $t = 8$ and after, there is no interaction at all. Though not obvious, this also happens to DNS in the sense that $\mathbf{A}$ tends to be diagonal. We suppose this is because DNS and DNS$_{\text{G}}$ only need a portion of data from the start to determine the existence of a link rather than looking into all the redundant time stamps.

Short Spring. We thus verify this by training and testing both variants with 50% and 25% of data cropped from the starting time stamp and summarize results in Short part of Table 2. It is seen that incomplete data in this task only slightly impact the performance. And this can be surprisingly reflected in the evolution of meta-systems. This also aligns with the intuition that *Link prediction* needs fewer data than *Trajectory prediction* as in Three Body.

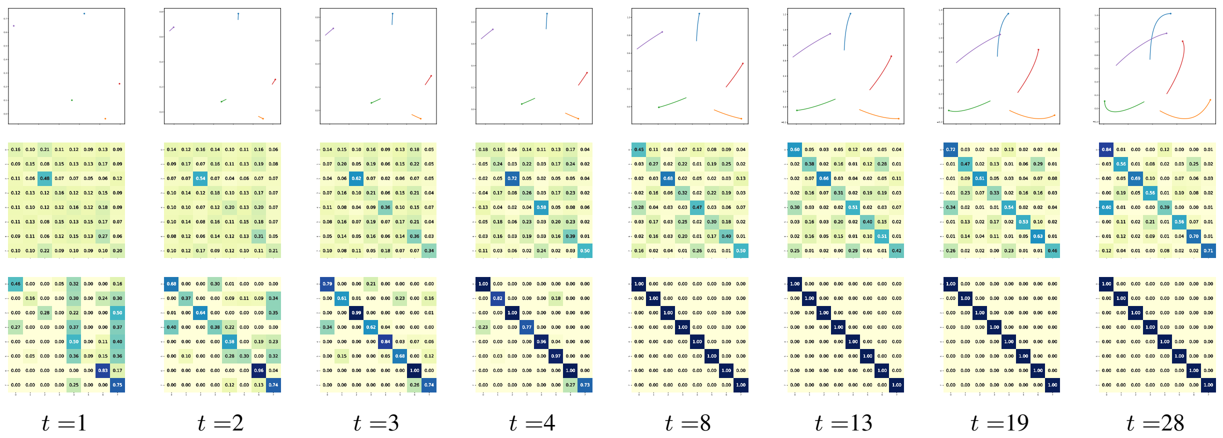

Figure 4. Visualization of the evolution of the meta-systems of DNS and DNS$_G$ on Spring dataset. On each time stamp t, from top to bottom, we show the trajectory of the 5 balls, the meta-system state of DNS, and the meta-system state of DNS$_G$, respectively.

Table 4. **Video classification**. Accuracy of the human actions dataset (%). Norm and Unnorm refer to normalized and unnormalized inputs, respectively. Detailed results with superscript † and ‡ are in Tab. 9 and Tab. 10, respectively.

Model	Norm	Unnorm	
	Irreg	Reg	Irreg
CT-GRU	67.30±6.19†	60.33‡	66.67‡
NeuralCDE	89.73±3.38†	70.33‡	59.17‡
RIM	-	55.50‡	-
DNS	**91.35±3.48**†	**97.00**‡	**95.33**‡

Ablation Study. Since our method merely incorporates an extra meta-system and a control encoder for modeling the interaction compared to standard NeuralCDE, we conduct experiments under different settings to see how different encoders and hidden state dimensions can contribute to improving NeuralCDE. To ensure fairness, we cast a 2-layer MLP with different output sizes (2 and 16 times of input size) as in DNS to obtain varying sizes of controls. Results are summarized in Table 3 (detailed in Tab. 6). We see that with an extra control encoder, there is no obvious performance difference among these settings. However, once the interaction meta-system is imposed, DNS can achieve quite significant performance gain. This, in turn, shows the necessity of the proposed meta-system for explicitly modeling the evolving interactions.

4.3. Human Actions

The recognition of human actions dataset contains three types of human actions, which are hand clapping, hand waving, and jogging (Schuldt et al., 2004). For this dataset, we consider the limbs of the character as subsystems. When the character does one kind of action, subsystems interact in a specific pattern. We test the performance of all the selected models with the learnable backbone Resnet18 (He et al., 2016). We also test the compatibility of all methods with different dynamical ranges: Norm and Unnorm indicate pixel value in $[0, 1]$ and $[0, 255]$, respectively. Experimental results are summarized in Table 4. DNS consistently outperforms all other methods and exhibits strong compatibility to drastically changed ranges under Unnorm setting. Thus it is potentially more flexible to be integrated into various tasks with a large dynamical range (e.g., earthquake).

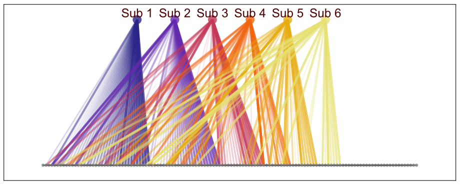

Figure 5. A figure showing the importance of each feature vector entry for subsystems

To view how the decoupling works for video recognition tasks, we visualize the strength of the learned parameters by mapping the 128-D feature into 6 latent sub-systems in Figure 5 with re-ordered indices for better view. It can be seen that there are some obvious latent structures in the grouping of the parameters 128-D control to the system. Each sub-system mainly focuses on a small portion of the control, based on which we can infer that each sub-system models different components in inputted images.

4.4. Impact of Subsystem Number

For complex systems, the number of latent entities in the systems is hard to define. For example, in the spring dataset, there are 5 particles randomly connected to each other. One may imagine the best number of subsystems to be 5. But a more reasonable approach is to define the number of sub-systems by the average edge connectivity λ of the particle graph whose vertices are 5 particles and edges being the invisible spring. This approach is based on the assumption that to remove interactions by cutting the minimum number of the spring, we should cut at least λ springs and result in λ independent subsystems. Hence, the optimal settings of the number of subsystems may not determine by the number of physical entities. An approach for tuning this hyperparameter is to use a grid search. From the experiment results on the spring dataset, we can see that DNS still has a satisfying performance when this hyperparameter is not optimal.

5. Conclusion

In this paper, we propose a method for modeling cluttered and irregularly sampled sequential data. Our method is built upon the assumption that complex observation may be derived from relatively simple and independent latent sub-systems, wherein the interactions also evolve over time. We devise a strategy to explicitly decouple such latent subsystems and a meta-system governing the interaction. Inspired by recent findings of projected differential equations and the tool of Bregman divergence, we present a novel interpretation of our model and pose some potential future directions. Experiments on various tasks demonstrate the prominent performance of our method over previous state-of-the-art methods.

References

Amos, B. Differentiable optimization-based modeling for machine learning. *Ph. D. thesis*, 2019.

Chamberlain, B., Rowbottom, J., Eynard, D., Di Giovanni, F., Dong, X., and Bronstein, M. Beltrami flow and neural diffusion on graphs. *NeurIPS*, 2021a.

Chamberlain, B., Rowbottom, J., Gorinova, M. I., Bronstein, M., Webb, S., and Rossi, E. Grand: Graph neural diffusion. In *ICML*, 2021b.

Che, Z., Purushotham, S., Cho, K., Sontag, D., and Liu, Y. Recurrent neural networks for multivariate time series with missing values. *Scientific reports*, 8(1):1–12, 2018.

Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. In *NeurIPS*, 2018.

Chen, R. T., Amos, B., and Nickel, M. Learning neural event functions for ordinary differential equations. In *ICLR*, 2021.

Chen, Y. and Ye, X. Projection onto a simplex. *arXiv preprint arXiv:1101.6081*, 2011.

Chmiela, S., Sauceda, H. E., Tkatchenko, A., and Müller, K.-R. Accurate molecular dynamics enabled by efficient physically constrained machine learning approaches. In *Machine Learning Meets Quantum Physics*, pp. 129–154. Springer, 2020.

Cho, K., Van Merriënboer, B., Bahdanau, D., and Bengio, Y. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

Combettes, P. L. and Wajs, V. R. Signal recovery by proximal forward-backward splitting. *Multiscale modeling & simulation*, 4(4):1168–1200, 2005.

De Brouwer, E., Simm, J., Arany, A., and Moreau, Y. Gru-ode-bayes: Continuous modeling of sporadically-observed time series. *NeurIPS*, 2019.

Diehl, F., Brunner, T., Le, M. T., and Knoll, A. Graph neural networks for modelling traffic participant interaction. In *IEEE Intelligent Vehicles Symposium (IV)*, 2019.

Dragomir, R. A., Even, M., and Hendrikx, H. Fast stochastic bregman gradient methods: Sharp analysis and variance reduction. In *ICML*, 2021.

Dupont, E., Doucet, A., and Teh, Y. W. Augmented neural odes. *NeurIPS*, 2019.

Dupuis, P. and Nagurney, A. Dynamical systems and variational inequalities. *Annals of Operations Research*, 44 (1):7–42, 1993.

Fan, C., Zhang, Y., Pan, Y., Li, X., Zhang, C., Yuan, R., Wu, D., Wang, W., Pei, J., and Huang, H. Multi-horizon time series forecasting with temporal attention learning. In *ACM SIGKDD*, 2019.

Futoma, J., Hariharan, S., and Heller, K. Learning to detect sepsis with a multitask gaussian process rnn classifier. In *ICML*, 2017.

Golany, T., Freedman, D., and Radinsky, K. Ecg ode-gan: Learning ordinary differential equations of ecg dynamics via generative adversarial learning. In *AAAI*, 2021.

Goldt, S., Mézard, M., Krzakala, F., and Zdeborová, L. Modeling the influence of data structure on learning in neural networks: The hidden manifold model. *Physical Review X*, 10(4):041044, 2020.

Goyal, A., Lamb, A., Hoffmann, J., Sodhani, S., Levine, S., Bengio, Y., and Schölkopf, B. Recurrent independent mechanisms. In *ICLR*, 2021.

Green, P. J. and Silverman, B. W. *Nonparametric regression and generalized linear models: a roughness penalty approach*. Crc Press, 1993.

Ha, S. and Jeong, H. Unraveling hidden interactions in complex systems with deep learning. *Scientific reports*, 11(1):1–13, 2021.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, 2016.

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

Iakovlev, V., Heinonen, M., and Lähdesmäki, H. Learning continuous-time pdes from sparse data with graph neural networks. In *ICLR*, 2021.

Kaltenbach, S. and Koutsourelakis, P.-S. Incorporating physical constraints in a deep probabilistic machine learning framework for coarse-graining dynamical systems. *Journal of Computational Physics*, 419:109673, 2020.

Kidger, P., Morrill, J., Foster, J., and Lyons, T. Neural controlled differential equations for irregular time series. In *NeurIPS*, 2020.

Kipf, T., Fetaya, E., Wang, K.-C., Welling, M., and Zemel, R. Neural relational inference for interacting systems. In *ICML*, 2018.

Kolter, J. Z. and Manek, G. Learning stable deep dynamics models. *NeurIPS*, 2019.

Krichene, W., Krichene, S., and Bayen, A. Efficient bregman projections onto the simplex. In *2015 54th IEEE Conference on Decision and Control (CDC)*, 2015.

Kuramoto, Y. Self-entrainment of a population of coupled non-linear oscillators. In *International symposium on mathematical problems in theoretical physics*, pp. 420–422, 1975.

Li, S., Li, W., Cook, C., Zhu, C., and Gao, Y. Independently recurrent neural network (indrnn): Building a longer and deeper rnn. In *CVPR*, 2018.

Li, S. C.-X. and Marlin, B. M. A scalable end-to-end gaussian process adapter for irregularly sampled time series classification. *NeurIPS*, 2016.

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

Li, Y.-L., Liu, X., Wu, X., Li, Y., and Lu, C. Hoi analysis: Integrating and decomposing human-object interaction. *NeurIPS*, 2020b.

Lim, C. H. and Wright, S. J. Efficient bregman projections onto the permutahedron and related polytopes. In *Artificial Intelligence and Statistics*, 2016.

Linot, A. J. and Graham, M. D. Deep learning to discover and predict dynamics on an inertial manifold. *Physical Review E*, 101(6):062209, 2020.

Liu, X., Xiao, T., Si, S., Cao, Q., Kumar, S., and Hsieh, C.-J. Neural sde: Stabilizing neural ode networks with stochastic noise. *arXiv preprint arXiv:1906.02355*, 2019.

Liu, Y., Wang, X., Wu, S., and Xiao, Z. Independence promoted graph disentangled networks. In *AAAI*, 2020.

Lou, A., Lim, D., Katsman, I., Huang, L., Jiang, Q., Lim, S. N., and De Sa, C. M. Neural manifold ordinary differential equations. In *NeurIPS*, 2020.

Lu, Y., Li, Z., He, D., Sun, Z., Dong, B., Qin, T., Wang, L., and Liu, T.-Y. Understanding and improving transformer from a multi-particle dynamic system point of view. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2020.

Madan, K., Ke, N. R., Goyal, A., Schölkopf, B., and Bengio, Y. Fast and slow learning of recurrent independent mechanisms. In *ICLR*, 2021.

Martins, A. and Astudillo, R. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *ICML*, 2016.

Mei, H. and Eisner, J. M. The neural hawkes process: A neurally self-modulating multivariate point process. In *NeurIPS*, 2017.

Mozer, M. C., Kazakov, D., and Lindsey, R. V. Discrete event, continuous time rnns. *arXiv preprint arXiv:1710.04110*, 2017.

Nardini, J. T., Baker, R. E., Simpson, M. J., and Flores, K. B. Learning differential equation models from stochastic agent-based model simulations. *Journal of the Royal Society Interface*, 18(176):20200987, 2021.

Pang, B., Zha, K., Cao, H., Tang, J., Yu, M., and Lu, C. Complex sequential understanding through the awareness of spatial and temporal concepts. *Nature Machine Intelligence*, 2(5):245–253, 2020.

Park, S., Kim, K., Lee, J., Choo, J., Lee, J., Kim, S., and Choi, E. Vid-ode: Continuous-time video generation with neural ordinary differential equation. In *AAAI*, 2021.

Peters, B., Niculae, V., and Martins, A. F. Sparse sequence-to-sequence models. *arXiv preprint arXiv:1905.05702*, 2019.

Qin, Y., Song, D., Chen, H., Cheng, W., Jiang, G., and Cottrell, G. A dual-stage attention-based recurrent neural network for time series prediction. In *IJCAI*, 2017.

Rubanova, Y., Chen, R. T., and Duvenaud, D. K. Latent ordinary differential equations for irregularly-sampled time series. *NeurIPS*, 2019.

Rusch, T. K. and Mishra, S. Unicornn: A recurrent model for learning very long time dependencies. In *ICML*, 2021.

Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., and Battaglia, P. Learning to simulate complex physics with graph networks. In *ICML*, 2020.

Schuldt, C., Laptev, I., and Caputo, B. Recognizing human actions: a local svm approach. In *ICPR*, 2004.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*, 2017.

Sheard, S. A. and Mostashari, A. 6.2. 1 complexity types: From science to systems engineering. In *INCOSE International Symposium*, volume 21, pp. 673–682. Wiley Online Library, 2011.

Song, S., Lan, C., Xing, J., Zeng, W., and Liu, J. An end-to-end spatio-temporal attention model for human action recognition from skeleton data. In *AAAI*, 2017.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *NeurIPS*, 2017.

Vinuesa, R. and Brunton, S. L. Enhancing computational fluid dynamics with machine learning. *Nature Computational Science*, 2(6):358–366, 2022.

Vuckovic, J., Baratin, A., and Combes, R. T. d. A mathematical theory of attention. *arXiv preprint arXiv:2007.02876*, 2020.

Wainwright, M. J., Jordan, M. I., et al. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008.

Wang, Y., Bao, J., Liu, G., Wu, Y., He, X., Zhou, B., and Zhao, T. Learning to decouple relations: Few-shot relation classification with entity-guided attention and confusion-aware training. *arXiv preprint arXiv:2010.10894*, 2020.

Xhonneux, L.-P., Qu, M., and Tang, J. Continuous graph neural networks. In *ICML*, 2020.

Yıldız, Ç., Kandemir, M., and Rakitsch, B. Learning interacting dynamical systems with latent gaussian process odes. *arXiv preprint arXiv:2205.11894*, 2022.

Yu, T., Li, Y., and Li, B. Rhyrnn: Rhythmic rnn for recognizing events in long and complex videos. In *ECCV*, 2020.

Zhang, D. and Nagurney, A. On the stability of projected dynamical systems. *Journal of Optimization Theory and Applications*, 85(1):97–124, 1995.

Zhu, Q., Guo, Y., and Lin, W. Neural delay differential equations. In *ICLR*, 2021.

A. Appendix

A.1. Details about Finding the Attention of Each Subsystem

A.1.1. Model's Decouple of the Three Body System

Inspired by Grad-CAM (Selvaraju et al., 2017), we compute the sensitivity of the control signal with respect to input vectors. Such sensitivity is evaluated by the control's gradient with respect to input vectors. If the control signal of a subsystem is more sensitive to an entry of input vectors, we conclude that the subsystem focuses on this entry. We investigate the model's attention on all training samples at timestamps where the mutual gravity force of three celestial entities is strong. The results show that for all samples, without loss of generality, the first subsystem focuses on all the entries of input vectors, the second subsystem focuses on the motions on the x-axis, and the last subsystem focuses on the motions on the y-axis and z-axis.

A.1.2. Details about Figure 5

We replace the fully connected layer in the pretained Resnet18 with another neural network whose output size equals 64. Image feature vectors are fed forward by a linear layer of size 64 by 128 and activated by the ReLu function. Then, feature vectors are fed forward by distinct linear layers, and we obtain different control signals for each subsystem. In Figure 5, gray points on the second line denote entries of the 128-dimensional feature vector after reordering. For each subsystem, we plot the top 40 entries which have the greatest impact on control signals.

A.2. On the Equivalence of Modeling $\frac{\mathrm{d}\mathbf{A}}{\mathrm{d}t}$ and $\frac{\mathrm{d}\mathbf{L}}{\mathrm{d}t}$

Let $\mathbf{L}(t)$ denotes the multiplication of key and query, i.e., $\mathbf{L}(t) = \frac{\mathbf{Q}(t)\mathbf{K}^\top(t)}{\sqrt{d_k}}$ and $\mathbf{A} = \mathrm{softmax}(\mathbf{L})$. If we model the dynamics of $\mathbf{L}(t)$, we obtain

$$\mathbf{L}(t+\Delta t) = \mathbf{L}(t) + \Delta t \cdot \frac{\mathrm{d}\mathbf{L}}{\mathrm{d}t}, \tag{16}$$

Apply the $\mathrm{softmax}$ function on both sides of the equation, and we have

$$\begin{aligned} \mathbf{A}(t+\Delta t) &= \mathrm{softmax}(\mathbf{L}(t) + \Delta t \cdot \frac{\mathrm{d}\mathbf{L}}{\mathrm{d}t}) + \mathbf{A}(t) - \mathbf{A}(t) \\ &= \mathbf{A}(t) + \mathrm{softmax}(\mathbf{L}(t) + \Delta t \cdot \frac{\mathrm{d}\mathbf{L}}{\mathrm{d}t}) - \mathrm{softmax}(\mathbf{L}(t)) \end{aligned}$$

Reorder the equation, we have

$$\begin{aligned} \frac{\mathbf{A}(t+\Delta t) - \mathbf{A}(t)}{\Delta t} &= \frac{\mathrm{softmax}(\mathbf{L}(t) + \Delta t \cdot \frac{\mathrm{d}\mathbf{L}}{\mathrm{d}t}) - \mathrm{softmax}(\mathbf{L}(t))}{\Delta t} \\ &= \frac{\mathrm{softmax}(\mathbf{L}(t) + \Delta t \cdot \frac{\mathrm{d}\mathbf{L}}{\mathrm{d}t}) - \mathrm{softmax}(\mathbf{L}(t))}{\Delta t \cdot \frac{\mathrm{d}\mathbf{L}}{\mathrm{d}t}} \cdot \frac{\mathrm{d}\mathbf{L}}{\mathrm{d}t} \end{aligned}$$

Take $\Delta t \to 0$, we have

$$\frac{\mathrm{d}\mathbf{A}}{\mathrm{d}t} = \frac{\mathrm{dsoftmax}(\mathbf{L}(t))}{\mathrm{d}\mathbf{L}} \cdot \frac{\mathrm{d}\mathbf{L}}{\mathrm{d}t}, \tag{17}$$

which is equivalent to the update step in Eq. (15).

A.3. $\mathrm{softmax}$ and $\mathrm{sparsemax}$

In Wainwright et al. (2008), authors find a few similarities between $\mathrm{softmax}$ and $\mathrm{sparsemax}$ functions.

$\mathrm{softmax}$ operator: a projection operator with entropic regularization

$$\mathrm{softmax}(\mathbf{z}) = \arg\min_{\mathbf{y}\in\Delta^n} \mathbf{z}^\top \mathbf{y} - \mathbb{H}^{\text{entr}}(\mathbf{y})$$

where $\mathbb{H}^{\text{entr}}(\mathbf{y}) = \sum_i \mathbf{y}_i \log \mathbf{y}_i$.

$\mathrm{sparsemax}$ operator: a projection operator with Gini entropy regularization

$$\mathrm{sparsemax}(\mathbf{z}) = \arg\min_{\mathbf{p}\in\Delta^n} \mathbf{z}^\top \mathbf{y} - \mathbb{H}^{\mathrm{gini}}(\mathbf{y}) \tag{18a}$$

$$= \arg\min_{\mathbf{y}\in\Delta^n} ||\mathbf{z}-\mathbf{y}||^2 \tag{18b}$$

where $\mathbb{H}^{\mathrm{gini}}(\mathbf{y}) = \frac{1}{2}\sum_i \mathbf{y}_i(\mathbf{y}_i - 1)$.

Proposition 1. *The solution of Eq. (18a) is of the form:*

$$\mathrm{sparsemax}_i(\mathbf{z}) = [\mathbf{z}_i - \tau(\mathbf{z})]_+, \tag{19}$$

where $\tau : \mathbb{R}^K \to \mathbb{R}$ is the unique function that satisfies $\sum_j [\mathbf{z}_j - \tau(\mathbf{z})]_+ = 1$ for every $\mathbf{z}$. Furthermore, τ can be expressed as follows. Let $\mathbf{z}_{(1)} \geq \mathbf{z}_{(2)} \geq \cdots \geq \mathbf{z}_{(K)}$ be the sorted coordinates of $\mathbf{z}$, and define $[K] := \{1, 2, ..., K\}$ and $k(\mathbf{z}) := \max\{k \in [K] | 1 + k\mathbf{z}_{(k)} > \sum_{j\leq k} \mathbf{z}_{(j)}\}$. Then,

$$\tau(\mathbf{z}) = \frac{(\sum_{j\leq k(\mathbf{z})} \mathbf{z}_{(j)}) - 1}{k(\mathbf{z})} = \frac{(\sum_{j\in S(\mathbf{z})} \mathbf{z}_{(j)}) - 1}{|S(\mathbf{z})|} \tag{20}$$

, where $S(\mathbf{z}) := \{j \in [K] | \mathrm{sparesemax}_j(\mathbf{z}) > 0\}$ is the support of $\mathrm{sparsemax}(\mathbf{z})$ (Martins & Astudillo, 2016).

Proof. The Lagrangian of the optimization problem in Eq. (18a) is:

$$\mathcal{L}(\mathbf{z}, \mu, \tau) = \frac{1}{2}||\mathbf{y}-\mathbf{z}||^2 - \mu^\top \mathbf{y} + \tau(\mathbf{1}^\top \mathbf{y} - 1). \tag{21}$$

The optimal $(\mathbf{y}^*, \mu^*, \tau^*)$ must satisfy the following KKT conditions:

$$\mathbf{y}^* - \mathbf{z} - \mu^* + \tau^*\mathbf{1} = 0, \tag{22a}$$

$$\mathbf{1}^\top \mathbf{y}^* = 1, \mathbf{y}^* \geq 0, \mu^* \geq 0, \tag{22b}$$

$$\mu_i^* \mathbf{y}_i^* = 0, \forall i \in [K]. \tag{22c}$$

If $\mathbf{y}_i^* > 0$ for $i \in [K]$, then from Eq. (22c), we must have $\mu_i^* = 0$, which from Eq. 22a implies $\mathbf{y}_i^* = z_i - \tau^*$. Let $S(\mathbf{z}) := \{j \in [K] | \mathbf{y}_j^* > 0\}$. From Eq. (22b), we obtain $\sum_{j\in S(\mathbf{z})}(z_j - \tau^*) = 1$, which yields the right hand side of Eq. (20). Again from Eq. (22c), we have that $\mu_i^* > 0$ implies $\mathbf{y}_i^* = 0$, which from Eq. (22a) implies $\mu_i^* = \tau^* - \mathbf{z}_i \geq 0$, i.e., $\mathbf{z}_i \leq \tau^*$ for $i \notin S(\mathbf{z})$. Therefore, we have that $k(\mathbf{z}) = |S(\mathbf{z})|$, which proves the first equality of Eq. (20). Another way to prove the above proposition using Moreau's identity (Combettes & Wajs, 2005) can be found in Chen & Ye (2011). □

Proposition 2. $\mathrm{sparsemax}$ *is differentiable everywhere except at splitting points $\mathbf{z}$ where the support set $S(\mathbf{z})$ changes, i.e., where $S(\mathbf{z}) \neq S(\mathbf{z} + \epsilon\mathbf{d})$ for some $\mathbf{d}$ and infinitesimal ϵ and we have that*

$$\frac{\partial \mathrm{sparsemax}_i(\mathbf{z})}{\partial \mathbf{z}_j} = \begin{cases} \delta_{ij} - \dfrac{1}{|S(\mathbf{z})|} & \textit{if} \quad i, j \in S(\mathbf{z}) \\ 0 & \textit{otherwise} \end{cases} \tag{23}$$

where δ_{ij} is the Kronecker delta, which evaluates to 1 if $i = j$ and 0 otherwise. Let $\mathbf{s}$ be an indicator vector whose ith entry is 1 if $i \in S(\mathbf{z})$, and 0 otherwise. We can write the Jacobian matrix as

$$\mathbf{J}_{\mathrm{sparsemax}}(\mathbf{z}) = \mathrm{diag}(\mathbf{s}) - \frac{\mathbf{s}\mathbf{s}^\top}{|S(\mathbf{z})|} \tag{24a}$$

$$\mathbf{J}_{\mathrm{sparsemax}}(\mathbf{z}) \cdot \mathbf{v} = \mathbf{s} \odot (\mathbf{v} - \hat{v}\mathbf{1}), \quad \textit{with} \quad \hat{v} := \frac{\sum_{j\in S(\mathbf{z})} v_j}{|S(\mathbf{z})|} \tag{24b}$$

where $\odot$ denotes the Hadamard product (Martins & Astudillo, 2016).

Proof. From Eq. (19), we have

$$\frac{\partial \text{sparsemax}_i(\mathbf{z})}{\partial \mathbf{z}_j} = \begin{cases} \delta_{ij} - \dfrac{\partial \tau(\mathbf{z})}{\partial z_j} & \text{if} \quad \mathbf{z}_i > \tau(\mathbf{z}) \\ 0 & \text{otherwise} \end{cases} \tag{25}$$

From Eq. (20), we have

$$\frac{\partial \tau(\mathbf{z})}{\partial \mathbf{z}_j} = \begin{cases} \dfrac{1}{|S(\mathbf{z})|} & \text{if} \quad j \in S(\mathbf{z}) \\ 0 & \text{otherwise} \end{cases} \tag{26}$$

Note that $j \in S(\mathbf{z}) \iff \mathbf{z}_j > \tau(\mathbf{z})$. Therefore, we have

$$\frac{\partial \text{sparsemax}_i(\mathbf{z})}{\partial \mathbf{z}_j} = \begin{cases} \delta_{ij} - \dfrac{1}{|S(\mathbf{z})|} & \text{if} \quad i, j \in S(\mathbf{z}) \\ 0 & \text{otherwise} \end{cases} \tag{27}$$

□

A.4. Experiment Details

A.4.1. DNS

For implementation simplicity, DNS with batch input requires each sample to be observed at the first and last timestamp. Default control signal dimension equals 2× input_size. When initializing the Weight matrix of the key and query layer, control encoder, and initial hidden state encoder, we use $0.01\times$ torch.rand and set bias equals 0. We grid-search hyperparameters of the layer number of neural networks that parameterize the dynamics in $[2, 3, 4]$ ($[2]$ for the spring dataset) and the number of subsystems in $[5, 8, 10]$ ($[6, 8]$ for the human action dataset).

A.4.2. CT-GRU

We grid-search hyperparameters of the time for the state to decay to a proportion e^{-1} of its initial level (τ) in $[0.5, 1, 2]$ and the number of traces with log-linear spaced time scales (M) in $[5, 8]$.

A.4.3. NeuralCDE

We use the Euler method to integrate the CDE. We grid-search hyperparameters of the layer number of neural networks that parameterize the dynamics in $[2, 3, 4]$.

A.4.4. RIM

We set relatively unimportant hyperparameters to the default values in the original paper. Key size input:64, value size input: 400, query size input 64, number of input heads: 1, number of common heads: 1, input dropout: 0.1, common dropout: 0.1, key size common: 32, value size common: 100, query size common: 32. We grid-search hyperparameters of the number of blocks and the number of blocks to be updated in $[(5, 3), (8, 3), (8, 5)]$.

A.5. Training Hyperparameters

We use 5-fold cross-validation (except for the three-body dataset because training processes of all models are very stable) and early stop if the validation accuracy is not improved for 10 epochs. We use the Adam optimizer and set the learning rate to 1e-3 with a cosine annealing scheduler with eta_min=1e-4 (5e-5 on the three-body dataset). Except for the spring dataset, we apply gradient clipping with the max gradient norm equal to 0.1. We use cumulative gradients on the three body dataset with batch size equal to 1 and update after 128 times forward. We set the batch size to 128 and 1 on the spring and human action datasets, respectively.

A.6. Dataset Settings

A.6.1. Three Body Dataset

We use Python to simulate the motion of three bodies. We add a multiplication noise from a uniform distribution $\mathcal{U}(0.995, 1.005)$. We generate 50k training samples, 5k validation samples, and 5k test samples. Three celestial bodies in all samples have a fixed initial position, and each pair has the sample distance. We randomly initialize the velocity so that in most samples, all three bodies have strong interactions, and it is also possible that only two celestial bodies have strong interactions, and the rest moves almost in a straight line. The dataset contains the locations of three bodies in three-dimensional space, so the input size equals 9. All samples have a length of 8. For the partially observed dataset, all samples have a length of 6, and the locations at the last timestamp are always observed. We use the historical motion of three bodies to predict 3 subsequent motions. We train each model with hidden size in [512, 1024, 2048] and report the MSE loss on the test set.

A.6.2. Spring

We follow the experiment setting in Kipf et al. (2018). We generate 50k/40k (regular/irregular) training samples and 10k test samples and use 5-fold cross-validation. We test the models' noise resistance ability on the noisy spring dataset. The noise level can be seen in Figure 6. We set the number of particles to 5. The input contains the current location and velocity of each particle in two dimensions, so the input size is 20. All samples have lengths of 49 and 19 for regular and irregular spring datasets, respectively. Feature vectors at the first and last timestamp are always observed. The task is to predict whether there are springs connecting two particles. We search the hidden size in $[128, 256, 512]$ for CTGRU, RIM and DNS and in $[128, 256, 512, 1024]$ for NeuralCDE. Models' sizes are at the relatively same magnitude level.

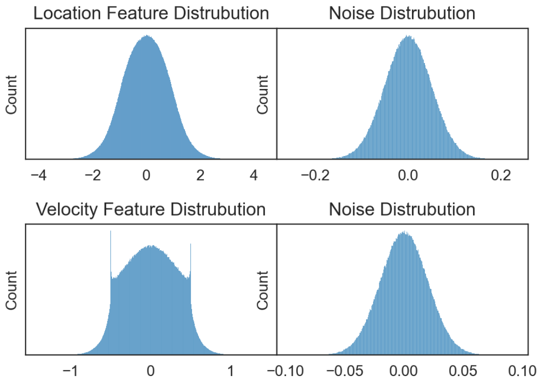

Figure 6. Noise level. Figures on the left-hand side plot feature magnitude levels, and figures on the right-hand side plot the additional noise level added to the corresponding feature vectors in each entry.

A.6.3. Human Action

The human action dataset contains three types of human actions. There are 99 videos for hand clapping, 100 videos for hand waving, and 100 videos for jogging. Videos have a length of 15 seconds on average, and all videos were taken over homogeneous backgrounds with a static camera. We take 50 equispaced frames from each video and downsample the resolution of each frame to 224×224 pixels. For the irregular human action dataset, each video has a length of 36 to 50 frames. We normalize images with mean and std equal to 0.5 and use Resnet18 pretained on ImageNet (He et al., 2016) as feature extractors. We set the output size of the fully connected layer to 64. Models need to use clustered image features for action recognition. We grid search the best hidden size in $[64, 128, 256]$.

A.7. Experiment Results Supplement

We use "method x $\mathbf{y}$ " to indicate settings, where x and $\mathbf{y}$ are the dimension of the hidden size and the number of the underlying modules (e.g., τ and M in CT-GRU, the layer number of neural network in NeuralCDE, the number of blocks to be updated and the number of blocks in RIM, and the number of subsystems and the layer number of neural network in DNS).

A.7.1. Noisy Spring

Because of the huge performance gaps among models, we do not run the cross-validation. Results are shown in Table 8. DNS slightly surpasses DNS_D in general.

Table 5. **Trajectory prediction**. MSE loss of the three body dataset ($\times 10^{-2}$)

Model	Square-Error ($\times 10^{-2}$)	
	Regular	Irregular
CT-GRU 512 1 8	2.0659	2.7449
CT-GRU 1024 1 8	1.9509	2.5653
CT-GRU 2048 1 8	1.8272	2.4811
NeuralCDE 512 2	3.8252	5.0077
NeuralCDE 1024 2	4.3028	5.4811
NeuralCDE 2048 2	3.3297	failure
RIM 512 5 8	2.7900	-
RIM 1024 5 8	2.4510	-
RIM 2048 5 8	failure	-
DNS 512 3 2	2.0265	2.5574
DNS 1024 3 2	2.0804	2.4735
DNS 2048 3 2	**1.7573**	**2.2164**

Table 6. **Link prediction**. Accuracy of the spring dataset ($\times$%). We can see that the control encoder does not have a significant impact on the performance.

Control	Accuracy (%)
No encoding + 128	90.02
No encoding + 256	91.06
No encoding + 512	91.57
MLP(2×input) + 128	87.01
MLP(2×input) + 256	90.87
MLP(2×input) + 512	91.51
MLP(16×input) + 128	91.17
MLP(16×input) + 256	91.08
MLP(16×input) + 512	90.70
DNS (8×MLP(2×input))	**95.38**

A.7.2. Three Body

In Table 5, we show models' performance under the same training strategy. For NeuralCDE and RIM, there are two "failure" cases that cannot be trained by all means.

A.7.3. Spring

More detailed results of the Clean setting of Spring dataset can be found in Table 7. Results under the Noisy Spring setting are summarized in Table 8.

A.7.4. Human Action

Detailed results on Norm and Unnorm setting can be found in Table 9 and 10.

Table 7. **Link Prediction**. Spring Dataset.

Model	Accuracy (%)		Model	Accuracy (%)	
	Regular	Irregular		Regular	Irregular
CT-GRU 128 0.5 5	88.76±0.09	86.24±0.19	NeuralCDE 1024 4	90.46±0.26	82.95±0.19
CT-GRU 128 0.5 8	88.70±0.13	86.21±0.15	RIM 128 3 5	89.62±0.23	-
CT-GRU 128 1.0 5	88.58±0.20	86.38±0.15	RIM 128 3 8	84.76±0.14	-
CT-GRU 128 1.0 8	88.64±0.11	86.35±0.23	RIM 128 5 8	89.25±0.08	-
CT-GRU 128 2.0 5	89.81±0.48	86.72±0.24	RIM 256 3 5	89.34±0.12	-
CT-GRU 128 2.0 8	89.99±0.84	86.68±0.07	RIM 256 3 8	84.72±0.12	-
CT-GRU 256 0.5 5	89.52±0.09	86.20±0.15	RIM 256 5 8	**89.73±0.07**	-
CT-GRU 256 0.5 8	89.41±0.23	86.26±0.13	RIM 512 3 5	80.44±0.42	-
CT-GRU 256 1.0 5	89.55±0.17	86.43±0.22	RIM 512 3 8	74.00±0.11	-
CT-GRU 256 1.0 8	89.53±0.20	86.49±0.15	RIM 512 5 8	83.03±0.29	-
CT-GRU 256 2.0 5	90.57±0.48	87.06±0.12	DNS 128 5 2	90.50±1.78	91.63±0.49
CT-GRU 256 2.0 8	90.41±0.37	87.28±0.12	DNS 128 8 2	93.93±0.66	93.42±0.51
CT-GRU 512 0.5 5	90.21±0.38	87.22±0.47	DNS 128 10 2	92.92±1.31	92.94±0.28
CT-GRU 512 0.5 8	90.70±0.82	86.96±0.31	DNS 256 5 2	92.34±1.53	91.05±1.69
CT-GRU 512 1.0 5	90.64±0.48	87.06±0.23	DNS 256 8 2	93.79±1.81	92.32±1.95
CT-GRU 512 1.0 8	90.99±0.90	87.02±0.25	DNS 256 10 2	**94.44±0.69**	92.98±1.05
CT-GRU 512 2.0 5	92.50±0.46	88.18±0.26	DNS 512 5 2	90.55±1.95	90.30±2.42
CT-GRU 512 2.0 8	**92.89±0.52**	**88.47±0.34**	DNS 512 8 2	94.38±0.95	93.57±0.55
NeuralCDE 128 2	90.74±0.11	88.59±0.11	DNS 512 10 2	94.37±1.21	93.60±1.21
NeuralCDE 128 3	89.23±0.24	87.24±0.40	DNS$_G$ 128 5 2	91.48±1.26	91.28±1.66
NeuralCDE 128 4	88.95±0.09	84.64±0.78	DNS$_G$ 128 8 2	94.00±0.55	93.11±0.83
NeuralCDE 256 2	92.11±0.06	89.45±0.10	DNS$_G$ 128 10 2	92.92±1.31	93.67±0.75
NeuralCDE 256 3	91.08±0.07	88.13±0.13	DNS$_G$ 256 5 2	91.77±1.07	91.78±1.39
NeuralCDE 256 4	90.18±0.08	84.52±0.59	DNS$_G$ 256 8 2	94.31±0.48	91.99±2.73
NeuralCDE 512 2	**92.47±0.06**	**89.74±0.18**	DNS$_G$ 256 10 2	92.82±1.21	**94.25±0.29**
NeuralCDE 512 3	91.56±0.09	87.85±0.22	DNS$_G$ 512 5 2	92.14±1.79	90.98±1.90
NeuralCDE 512 4	90.89±0.08	83.92±0.16	DNS$_G$ 512 8 2	93.11±0.27	92.20±0.47
NeuralCDE 1024 2	91.69±0.13	89.12±0.39	DNS$_G$ 512 10 2	94.24±0.49	93.33±1.07
NeuralCDE 1024 3	91.35±0.08	87.35±0.22			

Table 8. **Link Prediction**. Noisy Spring Dataset.

Model	Accuracy (%)		Model	Accuracy (%)	
	Train&Test	Test		Train&Test	Test
CT-GRU 128 0.5 5	88.73±0.20	88.66±0.08	NeuralCDE 1024 4	88.96±0.41	88.66±0.33
CT-GRU 128 0.5 8	88.62±0.15	88.63±0.14	RIM 128 3 5	89.48±0.23	89.59±0.20
CT-GRU 128 1.0 5	88.58±0.11	88.53±0.17	RIM 128 3 8	84.91±0.19	84.81±0.10
CT-GRU 128 1.0 8	88.54±0.09	88.62±0.10	RIM 128 5 8	89.30±0.08	89.19±0.04
CT-GRU 128 2.0 5	89.88±0.38	89.72±0.48	RIM 256 3 5	89.42±0.12	89.31±0.12
CT-GRU 128 2.0 8	89.74±0.34	89.93±0.79	RIM 256 3 8	84.99±0.10	84.86±0.09
CT-GRU 256 0.5 5	89.42±0.11	89.43±0.12	RIM 256 5 8	**89.65±0.14**	**89.64±0.10**
CT-GRU 256 0.5 8	89.41±0.08	89.33±0.21	RIM 512 3 5	80.91±0.35	80.50±0.36
CT-GRU 256 1.0 5	89.34±0.07	89.47±0.21	RIM 512 3 8	74.11±0.01	74.18±0.10
CT-GRU 256 1.0 8	89.30±0.22	89.47±0.19	RIM 512 5 8	83.29±0.19	83.05±0.36
CT-GRU 256 2.0 5	89.87±0.15	90.48±0.50	DNS 128 5 2	85.23±8.21	84.17±2.96
CT-GRU 256 2.0 8	90.32±0.61	90.32±0.40	DNS 128 8 2	92.55±0.13	88.23±0.62
CT-GRU 512 0.5 5	91.12±0.66	90.10±0.38	DNS 128 10 2	92.67±0.85	85.74±0.94
CT-GRU 512 0.5 8	90.89±0.58	90.64±0.86	DNS 256 5 2	86.53±5.93	86.16±2.89
CT-GRU 512 1.0 5	90.88±0.79	90.57±0.46	DNS 256 8 2	92.92±0.43	87.49±2.47
CT-GRU 512 1.0 8	91.10±0.71	90.92±0.90	DNS 256 10 2	92.82±0.75	88.22±1.49
CT-GRU 512 2.0 5	92.35±0.36	92.39±0.45	DNS 512 5 2	89.68±2.84	85.84±2.86
CT-GRU 512 2.0 8	**92.71±0.55**	**92.80±0.53**	DNS 512 8 2	93.37±0.97	89.56±0.42
NeuralCDE 128 2	89.22±0.14	88.09±0.12	DNS 512 10 2	93.42±1.05	87.20±3.36
NeuralCDE 128 3	87.73±0.10	86.76±0.27	DNS_S 128 5 2	89.30±1.70	88.99±1.80
NeuralCDE 128 4	87.30±0.14	86.86±0.11	DNS_S 128 8 2	93.22±0.39	92.08±0.68
NeuralCDE 256 2	90.26±0.05	88.74±0.10	DNS_S 128 10 2	92.83±1.09	92.13±0.66
NeuralCDE 256 3	89.43±0.09	88.34±0.13	DNS_S 256 5 2	92.13±1.03	89.17±1.43
NeuralCDE 256 4	88.56±0.13	88.06±0.11	DNS_S 256 8 2	93.08±1.11	91.49±2.10
NeuralCDE 512 2	**90.76±0.08**	89.27±0.10	DNS_S 256 10 2	93.47±1.60	92.10±1.36
NeuralCDE 512 3	90.09±0.10	89.00±0.13	DNS_S 512 5 2	89.68±1.79	90.62±2.42
NeuralCDE 512 4	89.27±0.11	88.84±0.05	DNS_S 512 8 2	92.77±1.86	**92.99±1.30**
NeuralCDE 1024 2	90.20±0.06	**89.61±0.09**	DNS_S 512 10 2	**93.67±0.57**	92.10±1.36
NeuralCDE 1024 3	89.89±0.23	89.42±0.09			

Table 9. **Action Classification**. Accuracy on **Nomarlized** data of Human Action.

Model	Accuracy (%)	Model	Accuracy (%)
CT-GRU 64 0.5 5	61.89±4.71	NeuralCDE 128 4	68.11±11.74
CT-GRU 64 0.5 8	65.68±12.92	NeuralCDE 256 2	82.16±2.32
CT-GRU 64 1.0 5	60.54±4.39	NeuralCDE 256 3	64.59±12.51
CT-GRU 64 1.0 8	60.81±4.10	NeuralCDE 256 4	73.24±11.7
CT-GRU 64 2.0 5	57.84±5.88	DNS 64 6 2	83.51±14.84
CT-GRU 64 2.0 8	61.35±2.78	DNS 64 6 3	89.73±8.40
CT-GRU 128 0.5 5	57.03±8.65	DNS 64 6 4	85.68±7.86
CT-GRU 128 0.5 8	55.41±14.38	DNS 64 8 2	80.27±15.70
CT-GRU 128 1.0 5	61.08±8.08	DNS 64 8 3	90.81±3.01
CT-GRU 128 1.0 8	63.24±12.8	DNS 64 8 4	**91.35±3.48**
CT-GRU 128 2.0 5	58.92±6.87	DNS 128 6 2	68.38±11.64
CT-GRU 128 2.0 8	59.73±7.71	DNS 128 6 3	87.03±4.49
CT-GRU 256 0.5 5	62.97±9.92	DNS 128 6 4	90.54±2.09
CT-GRU 256 0.5 8	60.81±4.91	DNS 128 8 2	75.41±18.12
CT-GRU 256 1.0 5	58.65±6.49	DNS 128 8 3	90.54±2.09
CT-GRU 256 1.0 8	60.27±6.97	DNS 128 8 4	72.70±16.78
CT-GRU 256 2.0 5	60.81±4.10	DNS 256 6 2	79.73±9.01
CT-GRU 256 2.0 8	**67.30±6.19**	DNS 256 6 3	79.73±10.64
NeuralCDE 64 2	71.89±12.13	DNS 256 6 4	84.32±8.74
NeuralCDE 64 3	**89.73±3.38**	DNS 256 8 2	87.84±4.83
NeuralCDE 64 4	72.16±5.83	DNS 256 8 3	82.97±12.52
NeuralCDE 128 2	82.43±4.60	DNS 256 8 4	83.78±14.38
NeuralCDE 128 3	70.54±11.51		

Table 10. **Action Classification**. Accuracy on **Unnomarlized** data of Human Action. (%)

Model	Unnormalized	
	Regular	Irregular
CT-GRU 32 1.0 8	58.33	56.00
CT-GRU 64 1.0 8	60.33	66.67
NeuralCDE 32 2	52.47	57.83
NeuralCDE 64 2	70.33	59.17
RIM 32 3 6	55.50	-
RIM 64 3 6	44.83	-
DNS 32 6 2	95.00	**95.33**
DNS 64 6 2	**97.00**	93.17