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 sub-systems 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 meta-system capturing the interaction between entities over time. Specifically, we argue that the meta-system 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
We note, however, despite the ubiquity of this setting, there 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 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.

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., 2020; 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. (2020) 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.
Learning to Decouple Complex Systems

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 \( \dot{z}(t) = f(z(t), t) \), where \( z \in \mathbb{R}^d \) and \( t \) are d-dimension state and the time, respectively. Function \( f : \mathbb{R}^d \times \mathbb{R} \rightarrow \mathbb{R}^d \) governs the evolution of the dynamics. Given the initial state \( z(t_0) \), the state at any time \( t_1 \) can be evaluated with:

\[
\begin{align*}
\dot{z}(t_1) &= z(t_0) + \int_{t_0}^{t_1} f(z(s), s) ds 
\end{align*}
\]

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

\[
\begin{align*}
\dot{z}(t_1) &= z(t_0) + \int_{t_0}^{t_1} f_\theta(z(s), x(s), s) ds 
\end{align*}
\]

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

\[
\begin{align*}
z(t_1) &= z(t_0) + \int_{t_0}^{t_1} F_\theta(z(s)) \cdot x'(s) ds 
\end{align*}
\]

where \( F_\theta : \mathbb{R}^d \rightarrow \mathbb{R}^{d \times b} \) is a learnable vector field and \( x'(s) = dx/ds \) is the derivative of signal \( x \) w.r.t. time \( s \), thus \( F_\theta(z(s)) \cdot x'(s) \) is a matrix-vector multiplication. During implementation, Kidger et al. (2020) argued that a simple cubic spline interpolation on \( x \) allows dense calculation of \( x'(t) \) 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 \( i \)-th unit at depth \( l \) reads:

\[
\begin{align*}
\hat{z}_{l,i} &= z_{l,i} + \text{MHAtt}_{W_{l,i}^\text{att}}(z_{l,i}, [z_{l,1}, z_{l,2}, ..., z_{l,n}]) \\
\hat{z}_{l+1,i} &= \hat{z}_{l,i} + \text{FFN}_{W_{l,i}^\text{ffn}}(\hat{z}_{l,i}) 
\end{align*}
\]

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

\[
\begin{align*}
\frac{dz_i(t)}{dt} &= F(z_i(t), [z_1(t), ..., z_n(t)], t) + G(z_i(t)) 
\end{align*}
\]

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 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 \( K \subset \mathbb{R}^n \) and a mapping \( H : \mathbb{R}^n \rightarrow \mathbb{R}^n \), we can introduce an operator \( \Pi_K : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}^n \) which is defined by means of directional derivatives as:

\[
\begin{align*}
\Pi_K(a, H(a)) &= \lim_{\alpha \rightarrow 0} \frac{P_K(a + \alpha H(a)) - a}{\alpha} 
\end{align*}
\]
where $P_K(\cdot)$ is a projection onto $K$ in terms of Euclidean distance:

$$
\|P_K(a) - a\|_2 = \inf_{y \in K} \|y - a\|_2 
$$

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

$$
\frac{d a(t)}{dt} = \Pi_K(a, H(a))
$$

which allows for discontinuous dynamics on $a$.

### 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/controls be $c(t) \in \mathbb{R}^k$ at time $t$ for $t = 1, ..., T$, where $T$ is the time horizon. We employ $n$ distinct mappings (e.g., MLP) to obtain respective controls to each sub-system: $x_i(t) = p_i(c(t)) \in \mathbb{R}^m$ for $i = 1, ..., n$. A generic dynamics of the proposed method can be written as:

$$
\begin{align}
\frac{d z_i(t)}{dt} &= f_i(z_i(t), [z_1(t), ..., z_n(t)], a(t)) \tag{9a} \\
\frac{d a(t)}{dt} &= \Pi_S(a(t), g(a(t), [z_1(t), ..., z_n(t)])) \tag{9b}
\end{align}
$$

where Eq. (9a) and Eq. (9b) refer to the $i$th sub-system describing the evolution of a single latent entity and meta-system depicting the interactions, respectively. $z_i(t) \in \mathbb{R}^q$ is the hidden state for the $i$th subsystem, and $a$ is a tensor governing the dynamics of the interactions. Here $\Pi_S(\cdot)$ is a projection operator, which projects the evolving trajectory into set $S$. 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 $\Pi_S(\cdot)$ and the set $S$ play important roles in Eq. (9b). For $\Pi_S(\cdot)$, while previous works of ProjDEs only consider L2-induced projection, we propose novel interpretation and extension under Bregman divergence. For $S$, we consider a probabilistic simplex following the setting in Lu et al. (2020), Vuckovic et al. (2020), though it can be any polyhedron.

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 $x_i$s into Eq. (9a) in a controlled dynamical fashion as in the state-of-the-art method (Kidger et al., 2020).

$$
\frac{d z_i(t)}{dt} = F_i(z_i(t), a(t), [A(t) \cdot z(t)]_i) \; dx_i(t) \tag{10}
$$

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

$$
\frac{d z_i(t)}{dt} = F_i([A(t) \cdot z(t)]_i) \; dx_i(t) \tag{11}
$$

where $[\cdot]_i$ takes the $i$th slice from a tensor. Note timestamp $t$ can be arbitrary, resulting in irregularly sampled data. To address this, we follow the strategy in Kidger et al. (2020) by performing cubic spline interpolation on $x_i$ over observed timestamp $t$, resulting in $x_i(t)$ at dense time $t$.

Note that for all sub-systems, different from Eq. (10) we utilize an identical function/network $F(\cdot)$ as in Eq. (11), but with different control sequence $x_i(t) = p_i(c(t))$. 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 $S$ and the projection operator $\Pi$. In the context of ProjDEs, a system is constrained as $a(t) \in S$ for any $t$. In terms of interactions, a common choice of $S$ is the stochastic simplex which can be interpreted as a transition kernel (Vuckovic et al., 2020). We allow follow this setting by defining $S$ be a row-wise stochastic $(n - 1)$-simplices:

$$
S \triangleq \{ A \in \mathbb{R}^{m \times n} | A1 = 1, A_{ij} \geq 0 \} \tag{12}
$$

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

$$
\frac{d A(t)}{dt} = \Pi_S(A(t), g(A(t), [z_1(t), ..., z_n(t)])) \tag{13}
$$

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.
Learning to Decouple Complex Systems

We further discuss some neural-friendly features of Eq. (14a), which has a well-known closed-form solution $\text{softmax}(\cdot)$ appearing in attention mechanism. In Eq. (14b), we adopt a standard L2-induced projection identical to Eq. (15), which leads to sparse solutions (Wainwright et al., 2008). Intuitively, the projection of a point onto a simplex in terms of $L_2$ distance tends to lie on a facet or a vertex of a simplex, thus being sparse.

$$P_{S^{\text{soft}}}(A_{i,:}) = \arg\min_{B \in S} A_{i,:}^\top B_{i,:} - \mathbb{H}_{\text{entr}}(B_{i,:}) \quad (14a)$$

$$P_{S^{\text{sparse}}}(A_{i,:}) = \arg\min_{B \in S} A_{i,:}^\top B_{i,:} - \mathbb{H}_{\text{gini}}(B_{i,:})$$

$$= \arg\min_{B \in S} |A_{i,:} - B_{i,:}|^2 \quad (14b)$$

where $\mathbb{H}_{\text{entr}}(\cdot)$ and $\mathbb{H}_{\text{gini}}(\cdot) = \frac{1}{2} \sum_i y_i(y_i - 1)$ are the standard entropy and the gini-entropy, respectively. $A_{i,:}$ and $B_{i,:}$ are the $i$th row and column of $A$ and $B$, respectively. While the solution to Eq. (14b) is $\text{softmax}(A)$, 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 $A$ into the interior of $(n - 1)$-simplex. We visualize the two versions of projections in Eq. (14a) 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, $\text{softmax}$ 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. (15) using projection Eq. (14a) defines a projected dynamical system directly on $A$, we switch to update the system using $L$ 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 $\text{softmax}(\cdot)$:

$$A(t) = \text{Softmax}(L(t)) \quad (15a)$$

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

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

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

2. Second, both the solution to projection Eq. (14b) and its gradient w.r.t. $A$ 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 $z$ and $A$ simultaneously with a sufficiently small time step. We term our approach a decoupling-based neural system (DNS) using projection Eq. (14a) and 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 \((\times 10^{-6})\).

<table>
<thead>
<tr>
<th>MODEL</th>
<th>REGULAR</th>
<th>IRREGULAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>CT-GRU</td>
<td>1.8272</td>
<td>2.4811</td>
</tr>
<tr>
<td>NEURALCDE</td>
<td>3.3297</td>
<td>5.0077</td>
</tr>
<tr>
<td>RIM</td>
<td>2.4510</td>
<td>-</td>
</tr>
<tr>
<td>DNS</td>
<td>1.7573</td>
<td>2.2164</td>
</tr>
</tbody>
</table>

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 \(x_i(t)\) with natural smoothing spline [Green & Silverman 1993], in consideration of incorporating smoother controls and alleviating data noise. This version is termed as DNS\(_n\).

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 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\), A remains stable as the three bodies are apart from each other without intensive interactions. At \(t = 7\), 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.\textsuperscript{7} and Tab.\textsuperscript{8} in the appendix.

<table>
<thead>
<tr>
<th>Model</th>
<th>CLEAN</th>
<th>NOISY</th>
<th>SHORT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>REGULAR</td>
<td>IRREGULAR</td>
<td>TRAIN &amp; TEST</td>
</tr>
<tr>
<td>CT-GRU</td>
<td>92.89±0.52</td>
<td>88.47±0.34</td>
<td>92.71±0.55</td>
</tr>
<tr>
<td>NEURALCDE</td>
<td>92.47±0.06</td>
<td>89.74±0.18</td>
<td>90.76±0.08</td>
</tr>
<tr>
<td>RIM</td>
<td>89.73±0.07</td>
<td>-</td>
<td>89.65±0.14</td>
</tr>
<tr>
<td>DNS\textsubscript{0}</td>
<td>94.31±0.48</td>
<td>94.25±0.29</td>
<td>93.76±0.36</td>
</tr>
<tr>
<td>DNS\textsubscript{5}</td>
<td>94.44±0.69</td>
<td>93.60±1.21</td>
<td>93.47±0.57</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Control</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO ENCODING</td>
<td>91.57</td>
</tr>
<tr>
<td>MLP(2×INPUT)</td>
<td>91.51</td>
</tr>
<tr>
<td>MLP(16×INPUT)</td>
<td>91.17</td>
</tr>
<tr>
<td>DNS (8×MLP(2×INPUT))</td>
<td>95.38</td>
</tr>
</tbody>
</table>

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 \cite{Kuramoto1975,Kipf2018}. 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\textsuperscript{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\textsuperscript{2}

A. When flying apart, one body seems more independent, while another two keep entangled together. These are well reflected via the meta-system \textsuperscript{A}. To further see how the holistic 9-dimensional input is decoupled into sub-systems \(z_i\), we visualize the sub-system focus in Fig.\textsuperscript{3} (also see Appendix A.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.

**Clean Spring**. From CLEAN part of Table\textsuperscript{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\textsubscript{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\textsuperscript{2} DNS\textsubscript{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 \textsuperscript{A} of meta-systems over time in Fig.\textsuperscript{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\textsubscript{G}. One interesting thing we note is that the interactions in DNS\textsubscript{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 \textsuperscript{A} tends to be diagonal. We suppose this is because DNS and DNS\textsubscript{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\textsuperscript{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.
Learning to Decouple Complex Systems

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.

<table>
<thead>
<tr>
<th>MODEL</th>
<th>NORM IRREG</th>
<th>NORM REG</th>
<th>UNNORM IRREG</th>
<th>UNNORM REG</th>
</tr>
</thead>
<tbody>
<tr>
<td>CT-GRU</td>
<td>67.30±6.19†</td>
<td>60.33‡</td>
<td>66.67‡</td>
<td></td>
</tr>
<tr>
<td>NEURALCDE</td>
<td>89.73±3.38†</td>
<td>70.33‡</td>
<td>59.17‡</td>
<td></td>
</tr>
<tr>
<td>RIM</td>
<td>-</td>
<td>55.50‡</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>DNS</td>
<td>91.35±3.48†</td>
<td>97.00‡</td>
<td>95.33‡</td>
<td></td>
</tr>
</tbody>
</table>

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 9 (detailed in Table 10). 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 8. 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).

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-
Learning to Decouple Complex Systems

by the average edge connectivity $\lambda$ 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 $\lambda$ springs and result in $\lambda$ 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 sub-systems 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


Learning to Decouple Complex Systems


Learning to Decouple Complex Systems

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 pre-trained 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 $\frac{dA}{dt}$ and $\frac{dL}{dt}$

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

$$L(t + \Delta t) = L(t) + \Delta t \cdot \frac{dL}{dt} \tag{16}$$

Apply the softmax function on both sides of the equation, and we have

$$A(t + \Delta t) = \text{softmax}(L(t) + \Delta t \cdot \frac{dL}{dt}) + A(t) - A(t)$$

$$= A(t) + \text{softmax}(L(t) + \Delta t \cdot \frac{dL}{dt}) - \text{softmax}(L(t))$$

Reorder the equation, we have

$$\frac{A(t + \Delta t) - A(t)}{\Delta t} = \frac{\text{softmax}(L(t) + \Delta t \cdot \frac{dL}{dt}) - \text{softmax}(L(t))}{\Delta t \cdot \frac{dL}{dt}} \cdot \frac{dL}{dt} \tag{17}$$

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

$$\frac{dA}{dt} = \frac{d\text{softmax}(L(t))}{dL} \cdot \frac{dL}{dt},$$

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

A.3. softmax and sparsemax

In [Wainwright et al., 2008], authors find a few similarities between softmax and sparsemax functions.

softmax operator: a projection operator with entropic regularization

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

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

12
sparsemax operator: a projection operator with Gini entropy regularization

\[
\text{sparsemax}(z) = \arg \min_{\mathbb{P} \in \Delta^n} z^\top y - \mathbb{H}^{\text{gini}}(y) \tag{18a}
\]

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

where \(\mathbb{H}^{\text{gini}}(y) = \frac{1}{2} \sum_i y_i (y_i - 1)\).

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

\[
\text{sparsemax}_i(z) = [z_i - \tau(z)]_+ \tag{19}
\]

where \(\tau : \mathbb{R}^K \to \mathbb{R}\) is the unique function that satisfies \(\sum_j |z_j - \tau(z)|_+ = 1\) for every \(z\). Furthermore, \(\tau\) can be expressed as follows. Let \(z_1 \geq z_2 \geq \cdots \geq z_K\) be the sorted coordinates of \(z\), and define \([K] := \{1, 2, \ldots, K\}\) and \(k(z) := \max\{k \in [K] | 1 + k z(k) > \sum_{j \leq k} z(j)\}\). Then,

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

, where \(S(z) := \{j \in [K] | \text{sparsemax}_j(z) > 0\}\) is the support of \(\text{sparsemax}(z)\) \cite{martins2016}.

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

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

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

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

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

\[
\mu_i^* y_i^* = 0, \forall i \in [K]. \tag{22c}
\]

If \(y_i^* > 0\) for \(i \in [K]\), then from Eq. (22c), we must have \(\mu_i^* = 0\), which from Eq. (22a) implies \(y_i^* = z_i - \tau^*\). Let \(S(z) := \{j \in [K] | y_j^* > 0\}\). From Eq. (22a), we obtain \(\sum_{j \in S(z)} (z_j - \tau^*) = 1\), which yields the right hand side of Eq. (20). Again from Eq. (22c), we have that \(\mu_i^* > 0\) implies \(y_i^* = 0\), which from Eq. (22a) implies \(\mu_i^* = \tau^* - z_i \geq 0\), i.e., \(z_i \leq \tau^*\) for \(i \notin S(z)\). Therefore, we have that \(k(z) = |S(z)|\), which proves the first equality of Eq. (20). Another way to prove the above proposition using Moreau’s identity \cite{combettes2005} can be found in Chen & Ye \cite{chen2011}.

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

\[
\frac{\partial \text{sparsemax}_i(z)}{\partial z_j} = \begin{cases} 
\delta_{ij} - \frac{1}{|S(z)|} & \text{if } i, j \in S(z) \\
0 & \text{otherwise} 
\end{cases} \tag{23}
\]

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

\[
J_{\text{sparsemax}}(z) = \text{diag}(s) - \frac{s s^\top}{|S(z)|} \tag{24a}
\]

\[
J_{\text{sparsemax}}(z) \cdot v = s \odot (v - \hat{v} 1), \text{ with } \hat{v} := \frac{\sum_{j \in S(z)} v_j}{|S(z)|} \tag{24b}
\]

where \(\odot\) denotes the Hadamard product \cite{martins2016}.
Proof. From Eq. (19), we have
\[
\frac{\partial \text{sparsemax}_i(z)}{\partial z_j} = \begin{cases} 
\delta_{ij} - \frac{\partial \tau(z)}{\partial z_j} & \text{if } z_i > \tau(z) \\
0 & \text{otherwise}
\end{cases} 
\] (25)

From Eq. (20), we have
\[
\frac{\partial \tau(z)}{\partial z_j} = \begin{cases} 
\frac{1}{|S(z)|} & \text{if } j \in S(z) \\
0 & \text{otherwise}
\end{cases} 
\] (26)

Note that \( j \in S(z) \iff z_j > \tau(z) \). Therefore, we have
\[
\frac{\partial \text{sparsemax}_i(z)}{\partial z_j} = \begin{cases} 
\delta_{ij} - \frac{1}{|S(z)|} & \text{if } i, j \in S(z) \\
0 & \text{otherwise}
\end{cases} 
\] (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 \(2 \times \text{input size}\). When initializing the weight matrix of the key and query layer, control encoder, and initial hidden state encoder, we use \(0.01 \times \text{torch.rand}\) 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 \((\tau)\) in \([0.5, 1, 2]\) and the number of traces with log-linear spaced time scales \((M)\) in \([5, 8]\).

A.4.3. \textsc{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 \(\text{eta}_\text{min}=1e^{-4}\) (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 \( U(0.995, 1.005) \). 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.

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

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 pretrained 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 \( x \ y \)” to indicate settings, where \( x \) and \( y \) are the dimension of the hidden size and the number of the underlying modules (e.g., \( \tau \) 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 in general.

<table>
<thead>
<tr>
<th>Model</th>
<th>Square-Error ($\times 10^{-2}$)</th>
<th>Control</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Regular</td>
<td>Irregular</td>
<td></td>
</tr>
<tr>
<td>CT-GRU 512 1 8</td>
<td>2.0659</td>
<td>2.7449</td>
<td></td>
</tr>
<tr>
<td>CT-GRU 1024 1 8</td>
<td>1.9509</td>
<td>2.5653</td>
<td></td>
</tr>
<tr>
<td>CT-GRU 2048 1 8</td>
<td>1.8272</td>
<td>2.4811</td>
<td></td>
</tr>
<tr>
<td>NeuralCDE 512 2</td>
<td>3.8252</td>
<td>5.0077</td>
<td></td>
</tr>
<tr>
<td>NeuralCDE 1024 2</td>
<td>4.3028</td>
<td>5.4811</td>
<td></td>
</tr>
<tr>
<td>NeuralCDE 2048 2</td>
<td>3.3297</td>
<td>failure</td>
<td></td>
</tr>
<tr>
<td>RIM 512 5 8</td>
<td>2.7900</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>RIM 1024 5 8</td>
<td>2.4510</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>RIM 2048 5 8</td>
<td>failure</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>DNS 512 3 2</td>
<td>2.0265</td>
<td>2.5574</td>
<td></td>
</tr>
<tr>
<td>DNS 1024 3 2</td>
<td>2.0804</td>
<td>2.4735</td>
<td></td>
</tr>
<tr>
<td>DNS 2048 3 2</td>
<td><strong>1.7573</strong></td>
<td><strong>2.2164</strong></td>
<td></td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy (%)</th>
<th>Model</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Regular</td>
<td>Irregular</td>
<td></td>
</tr>
<tr>
<td>CT-GRU 128 0.5</td>
<td>88.76 ± 0.09</td>
<td>86.24 ± 0.19</td>
<td></td>
</tr>
<tr>
<td>CT-GRU 128 0.5</td>
<td>88.70 ± 0.13</td>
<td>86.21 ± 0.15</td>
<td></td>
</tr>
<tr>
<td>CT-GRU 128 1.0</td>
<td>88.58 ± 0.20</td>
<td>86.38 ± 0.15</td>
<td>RIM 128 3</td>
</tr>
<tr>
<td>CT-GRU 128 1.0</td>
<td>88.64 ± 0.11</td>
<td>86.35 ± 0.23</td>
<td>RIM 128 5</td>
</tr>
<tr>
<td>CT-GRU 128 2.0</td>
<td>89.81 ± 0.48</td>
<td>86.72 ± 0.24</td>
<td>RIM 256 3</td>
</tr>
<tr>
<td>CT-GRU 128 2.0</td>
<td>89.99 ± 0.84</td>
<td>86.68 ± 0.07</td>
<td>RIM 256 3</td>
</tr>
<tr>
<td>CT-GRU 256 0.5</td>
<td>89.52 ± 0.09</td>
<td>86.20 ± 0.15</td>
<td>RIM 256 5</td>
</tr>
<tr>
<td>CT-GRU 256 0.5</td>
<td>89.41 ± 0.23</td>
<td>86.26 ± 0.13</td>
<td>RIM 512 3</td>
</tr>
<tr>
<td>CT-GRU 256 1.0</td>
<td>89.55 ± 0.17</td>
<td>86.43 ± 0.22</td>
<td>RIM 512 3</td>
</tr>
<tr>
<td>CT-GRU 256 1.0</td>
<td>89.53 ± 0.20</td>
<td>86.49 ± 0.15</td>
<td>RIM 512 5</td>
</tr>
<tr>
<td>CT-GRU 256 2.0</td>
<td>90.57 ± 0.48</td>
<td>87.06 ± 0.12</td>
<td>DNS 128 5</td>
</tr>
<tr>
<td>CT-GRU 256 2.0</td>
<td>90.41 ± 0.37</td>
<td>87.28 ± 0.12</td>
<td>DNS 128 8</td>
</tr>
<tr>
<td>CT-GRU 512 0.5</td>
<td>90.21 ± 0.38</td>
<td>87.22 ± 0.47</td>
<td>DNS 128 10</td>
</tr>
<tr>
<td>CT-GRU 512 0.5</td>
<td>90.70 ± 0.82</td>
<td>86.96 ± 0.31</td>
<td>DNS 256 5</td>
</tr>
<tr>
<td>CT-GRU 512 1.0</td>
<td>90.64 ± 0.48</td>
<td>87.06 ± 0.23</td>
<td>DNS 256 8</td>
</tr>
<tr>
<td>CT-GRU 512 1.0</td>
<td>90.99 ± 0.90</td>
<td>87.02 ± 0.25</td>
<td>DNS 256 10</td>
</tr>
<tr>
<td>CT-GRU 512 2.0</td>
<td>92.50 ± 0.46</td>
<td>88.18 ± 0.26</td>
<td>DNS 512 5</td>
</tr>
<tr>
<td>CT-GRU 512 2.0</td>
<td>92.89 ± 0.52</td>
<td>88.47 ± 0.34</td>
<td>DNS 512 8</td>
</tr>
<tr>
<td>NeuralCDE 128 2</td>
<td>90.74 ± 0.11</td>
<td>88.59 ± 0.11</td>
<td>DNS 512 10</td>
</tr>
<tr>
<td>NeuralCDE 128 3</td>
<td>89.23 ± 0.24</td>
<td>87.24 ± 0.40</td>
<td>DNS 128 5</td>
</tr>
<tr>
<td>NeuralCDE 128 4</td>
<td>88.95 ± 0.09</td>
<td>84.64 ± 0.78</td>
<td>DNS 128 8</td>
</tr>
<tr>
<td>NeuralCDE 256 2</td>
<td>92.11 ± 0.06</td>
<td>89.45 ± 0.10</td>
<td>DNS 128 10</td>
</tr>
<tr>
<td>NeuralCDE 256 3</td>
<td>91.08 ± 0.07</td>
<td>88.13 ± 0.13</td>
<td>DNS 256 5</td>
</tr>
<tr>
<td>NeuralCDE 256 4</td>
<td>90.18 ± 0.08</td>
<td>84.52 ± 0.59</td>
<td>DNS 256 8</td>
</tr>
<tr>
<td>NeuralCDE 512 2</td>
<td>92.47 ± 0.06</td>
<td>89.74 ± 0.18</td>
<td>DNS 256 10</td>
</tr>
<tr>
<td>NeuralCDE 512 3</td>
<td>91.56 ± 0.09</td>
<td>87.85 ± 0.22</td>
<td>DNS 512 5</td>
</tr>
<tr>
<td>NeuralCDE 512 4</td>
<td>90.89 ± 0.08</td>
<td>83.92 ± 0.16</td>
<td>DNS 512 8</td>
</tr>
<tr>
<td>NeuralCDE 1024 2</td>
<td>91.69 ± 0.13</td>
<td>89.12 ± 0.39</td>
<td>DNS 512 10</td>
</tr>
<tr>
<td>NeuralCDE 1024 3</td>
<td>91.35 ± 0.08</td>
<td>87.35 ± 0.22</td>
<td>DNS 512 10</td>
</tr>
<tr>
<td>Model</td>
<td>Accuracy (%)</td>
<td>Model</td>
<td>Accuracy (%)</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------</td>
<td>-------------</td>
<td>--------------</td>
</tr>
<tr>
<td></td>
<td>Train&amp;Test</td>
<td>Test</td>
<td>Train&amp;Test</td>
</tr>
<tr>
<td>CT-GRU 128 0.5 5</td>
<td>88.73±0.20</td>
<td>88.66±0.08</td>
<td>NeuralCDE 1024 4</td>
</tr>
<tr>
<td>CT-GRU 128 0.5 8</td>
<td>88.62±0.15</td>
<td>88.63±0.14</td>
<td>RIM 128 3 5</td>
</tr>
<tr>
<td>CT-GRU 128 1.0 5</td>
<td>88.58±0.11</td>
<td>88.53±0.17</td>
<td>RIM 128 3 8</td>
</tr>
<tr>
<td>CT-GRU 128 1.0 8</td>
<td>88.54±0.09</td>
<td>88.62±0.10</td>
<td>RIM 128 5 8</td>
</tr>
<tr>
<td>CT-GRU 128 2.0 5</td>
<td>89.88±0.38</td>
<td>89.72±0.48</td>
<td>RIM 256 3 5</td>
</tr>
<tr>
<td>CT-GRU 128 2.0 8</td>
<td>89.74±0.34</td>
<td>89.93±0.79</td>
<td>RIM 256 3 8</td>
</tr>
<tr>
<td>CT-GRU 256 0.5 5</td>
<td>89.42±0.11</td>
<td>89.43±0.12</td>
<td>RIM 256 5 8</td>
</tr>
<tr>
<td>CT-GRU 256 0.5 8</td>
<td>89.41±0.08</td>
<td>89.33±0.21</td>
<td>RIM 512 3 5</td>
</tr>
<tr>
<td>CT-GRU 256 1.0 5</td>
<td>89.34±0.07</td>
<td>89.47±0.21</td>
<td>RIM 512 3 8</td>
</tr>
<tr>
<td>CT-GRU 256 1.0 8</td>
<td>89.30±0.22</td>
<td>89.47±0.19</td>
<td>RIM 512 5 8</td>
</tr>
<tr>
<td>CT-GRU 256 2.0 5</td>
<td>89.87±0.15</td>
<td>90.48±0.50</td>
<td>DNS 128 5 2</td>
</tr>
<tr>
<td>CT-GRU 256 2.0 8</td>
<td>90.32±0.61</td>
<td>90.32±0.40</td>
<td>DNS 128 8 2</td>
</tr>
<tr>
<td>CT-GRU 512 0.5 5</td>
<td>91.12±0.66</td>
<td>90.10±0.38</td>
<td>DNS 128 10 2</td>
</tr>
<tr>
<td>CT-GRU 512 0.5 8</td>
<td>90.89±0.58</td>
<td>90.64±0.86</td>
<td>DNS 256 5 2</td>
</tr>
<tr>
<td>CT-GRU 512 1.0 5</td>
<td>90.88±0.79</td>
<td>90.57±0.46</td>
<td>DNS 256 8 2</td>
</tr>
<tr>
<td>CT-GRU 512 1.0 8</td>
<td>91.10±0.71</td>
<td>90.92±0.90</td>
<td>DNS 256 10 2</td>
</tr>
<tr>
<td>CT-GRU 512 2.0 5</td>
<td>92.35±0.36</td>
<td>92.39±0.45</td>
<td>DNS 512 5 2</td>
</tr>
<tr>
<td>CT-GRU 512 2.0 8</td>
<td>92.71±0.55</td>
<td>92.80±0.53</td>
<td>DNS 512 8 2</td>
</tr>
<tr>
<td>NeuralCDE 128 2</td>
<td>89.22±0.14</td>
<td>88.09±0.12</td>
<td>DNS 512 10 2</td>
</tr>
<tr>
<td>NeuralCDE 128 3</td>
<td>87.73±0.10</td>
<td>86.76±0.27</td>
<td>DNS 512 5 2</td>
</tr>
<tr>
<td>NeuralCDE 128 4</td>
<td>87.30±0.14</td>
<td>86.86±0.11</td>
<td>DNS 512 8 2</td>
</tr>
<tr>
<td>NeuralCDE 256 2</td>
<td>90.26±0.05</td>
<td>88.74±0.10</td>
<td>DNS 512 10 2</td>
</tr>
<tr>
<td>NeuralCDE 256 3</td>
<td>89.43±0.09</td>
<td>88.34±0.13</td>
<td>DNS 512 5 2</td>
</tr>
<tr>
<td>NeuralCDE 256 4</td>
<td>88.56±0.13</td>
<td>88.06±0.11</td>
<td>DNS 512 8 2</td>
</tr>
<tr>
<td>NeuralCDE 512 2</td>
<td><strong>90.76±0.08</strong></td>
<td>89.27±0.10</td>
<td>DNS 512 10 2</td>
</tr>
<tr>
<td>NeuralCDE 512 3</td>
<td>90.09±0.10</td>
<td>89.00±0.13</td>
<td>DNS 512 5 2</td>
</tr>
<tr>
<td>NeuralCDE 512 4</td>
<td>89.27±0.11</td>
<td>88.84±0.05</td>
<td>DNS 512 8 2</td>
</tr>
<tr>
<td>NeuralCDE 1024 2</td>
<td>90.20±0.06</td>
<td><strong>89.61±0.09</strong></td>
<td>DNS 512 10 2</td>
</tr>
<tr>
<td>NeuralCDE 1024 3</td>
<td>89.89±0.23</td>
<td>89.42±0.09</td>
<td></td>
</tr>
</tbody>
</table>

*Table 8. Link Prediction. Noisy Spring Dataset.*
### Table 9. Action Classification. Accuracy on Normalized data of Human Action.

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

### Table 10. Action Classification. Accuracy on Unnormalized data of Human Action. (%)

<table>
<thead>
<tr>
<th>MODEL</th>
<th>UNNORMALIZED REGULAR</th>
<th>UNNORMALIZED IRREGULAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>CT-GRU 32 1.0 8</td>
<td>58.33</td>
<td>56.00</td>
</tr>
<tr>
<td>CT-GRU 64 1.0 8</td>
<td>60.33</td>
<td>66.67</td>
</tr>
<tr>
<td>NEURALCDE 32 2</td>
<td>52.47</td>
<td>57.83</td>
</tr>
<tr>
<td>NEURALCDE 64 2</td>
<td>70.33</td>
<td>59.17</td>
</tr>
<tr>
<td>RIM 32 3 6</td>
<td>55.50</td>
<td>-</td>
</tr>
<tr>
<td>RIM 64 3 6</td>
<td>44.83</td>
<td>-</td>
</tr>
<tr>
<td>DNS 32 6 2</td>
<td>95.00</td>
<td>95.33</td>
</tr>
<tr>
<td>DNS 64 6 2</td>
<td><strong>97.00</strong></td>
<td>93.17</td>
</tr>
</tbody>
</table>