Controlled Differential Equations on Long Sequences via Non-standard Wavelets

Sourav Pal [1] **Zhanpeng Zeng** [1] **Sathya N. Ravi** [2] **Vikas Singh** [1]

Abstract

Neural Controlled Differential equations (NCDE) are a powerful mechanism to model the dynamics in temporal sequences, e.g., applications involving physiological measures, where apart from the initial condition, the dynamics also depend on subsequent measures or even a different "control" sequence. But NCDEs do not scale well to longer sequences. Existing strategies adapt rough path theory, and instead model the dynamics over summaries known as *log signatures*. While rigorous and elegant, invertibility of these summaries is difficult, and limits the scope of problems where these ideas can offer strong benefits (reconstruction, generative modeling). For tasks where it is sensible to assume that the (long) sequences in the training data are a *fixed* length of temporal measurements – this assumption holds in most experiments tackled in the literature – we describe an efficient simplification. First, we recast the regression/classification task as an integral transform. We then show how restricting the class of operators (permissible in the integral transform), allows the use of a known algorithm that leverages non-standard Wavelets to decompose the operator. Thereby, our task (learning the operator) radically simplifies. A neural variant of this idea yields consistent improvements across a wide gamut of use cases tackled in existing works. We also describe a novel application on modeling tasks involving coupled differential equations.

1. Introduction

In the last few years, the function approximation capabilities of modern deep neural networks models (DNNs) (Liang & Srikant, 2016; Hanson & Raginsky, 2020) have been successfully exploited towards new results exploring the interface of DNNs with the rich extant literature on differential equations, a widely used tool in much of science. Consequently, new capabilities have emerged (Li et al., 2020b; Salvi et al., 2022). To see this, consider the task of modeling the relationship between variables and their derivatives, using a first order Ordinary Differential Equation (ODE) along with initial conditions of the form

$$z(0) = z_0; \qquad \frac{\mathrm{d}z}{\mathrm{d}t} = f(z(t)) = f(z_t) \tag{1}$$

where $z_t \in \mathbb{R}^d$. In their prevalent use in simulation and physics, the analytical form of f is known and hence the de-facto approach would use numerical initial value problem (IVP) solvers. But one may also hope to learn f parameterized as neural networks using observed data, and Neural Ordinary Differential Equation (NODE) (Chen et al., 2018) is a prominent example of such a model. Given a sequence of observations $x_0, x_1, \cdots, x_T$, with each $x_i \in \mathbb{R}^D$, where D is the dimensionality of the observed space, NODE can be written

$$z_0 = S_\theta(x_0); \qquad z_t = z_0 + \int_0^t f_\theta(z_s)\mathrm{d}s \tag{2}$$

where S_θ and f_θ are neural networks pertaining to the initial condition and latent dynamics in a latent space of dimensionality d respectively. Here, s is introduced as the variable of integration. In (1) and (2), $z_t = z(t)$ is the value of the

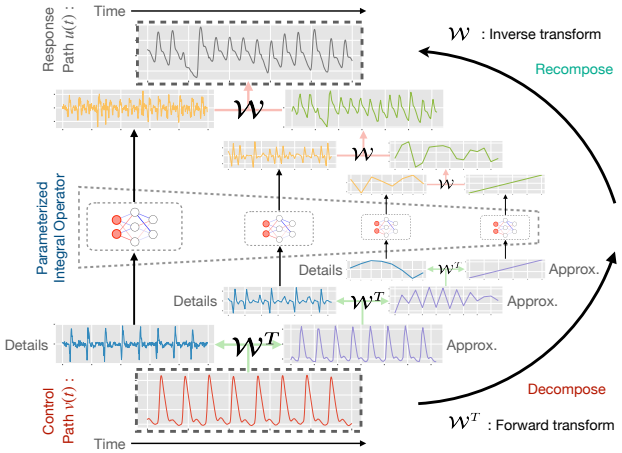

Figure 1. A control path driving a response. Multi-resolution analysis allows the proposed parameterization of an integral operator.

[1]University of Wisconsin-Madison [2]University of Illinois Chicago. Correspondence to: Sourav Pal <spal9@wisc.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

latent variable z when the variable of integration is varied up to t. Similarly, $z_0 = z(0)$ denotes the initial latent variable determined from the initial observation of $x_0 = x(0)$. We will use z_t to denote $z(t)$ (similarly for other variables).

The solution to an ODE/NODE is characterized by its initial conditions and underlying dynamics. However, in many cases (e.g., molecular dynamics), one must also model the noise and perturbations, often due to coupling with the environment leading to solutions influenced by conditions distinct from the ones at the start of the trajectory. Stochastic Differential Equations (SDE) nicely capture this setting, where the solutions correspond to stochastic processes. Similar to ODEs, the fusion of SDEs and DNNs have been studied, and finds use in score-based generative models (Song et al., 2020; Song & Ermon, 2020; Vahdat et al., 2021), data augmentation (Meng et al., 2021) among others (Jia & Benson, 2019; Li et al., 2020a; Nazarovs et al., 2021).

Beyond ODEs/SDEs: In many other cases, the dynamics may be influenced by yet another source (separate from noise) – and the Controlled Differential Equations (CDE) literature (Salvi, 2021) corresponds to this general setting. To allow learning in this regime, (Kidger et al., 2020) proposed Neural Controlled Differential Equation (NCDE) where the key change from NODE (2) is the use of Riemann–Stieltjes integral (instead of the Riemann integral),

$$z_0 = S_\theta(x_0); \qquad z_t = z_0 + \int_0^t f_\theta(z_s)\mathrm{d}\mathbf{X}_s \tag{3}$$

Here, $\mathbf{X} : t \to \mathbb{R}^D$ is a continuous function of bounded variation, say, a *continuous* approximation to the discrete observations of x and serves as the "control" for the solution to (3). If $\mathbf{X}$ is differentiable, we write the integral in (3) as:

$$z_t = z_0 + \int_0^t f_\theta(z_s)\frac{\mathrm{d}\mathbf{X}}{\mathrm{d}s}(s)\mathrm{d}s \tag{4}$$

which is in the Riemann integral form of an NODE. So, a NCDE can use the same solvers as NODE. We point out that t is the dimension along which the function evolves, it is convenient to think of t as time (which can be appended as a channel as in (Kidger et al., 2020)).

CDEs for long time series: While NCDEs are quite versatile, long time series poses challenges in terms of training time. The result in (Morrill et al., 2021), called Neural Rough Differential Equation (NRDE), uses *log-signatures* as summary measures of the control path to solve NCDEs, thereby circumventing the expensive point-wise calculation, achieving runtime and performance improvements. But inversion of the signature transform (Chang et al., 2017; Lyons & Xu, 2018) within a DNN model is challenging (Kidger et al., 2019). Also, the signatures are obtained via a pre-processing step. Finally, while numerical IVP solvers are excellent, when operating on representations obtained via upstream layers (or a pre-processing step), dealing with stiffness and/or abrupt changes requires care (Hairer & Wanner, 1999; Curtiss & Hirschfelder, 1952; Holt et al., 2022).

Simplifications for long (but fixed length) time sequences: At this time, NRDEs are an effective way to solve CDEs for long sequences. But we notice that in most use cases tackled in the literature, e.g., regression, reconstruction and classification tasks, the training data corresponds to samples whose length is fixed. And even if not, artificially adjusting the data to satisfy this requirement, does not lead to any disconnect between the task and the model's functionality. So, if the full time sequence is visible to the model (note that this is also true when calculating log-signatures), to uncover the latent dynamics, rather than step through incrementally, one may *unroll* the entire sequence of (a very long sequence of) steps, and attempt to solve it all at once. On its face, our proposal to *unroll* appears to be a poor design choice, especially for long sequences (it could increase the compute footprint). It is not clear why this is a "simplification".

Our contributions: Let us review the setting where the model can access the entire fixed length sequence. Viewing both x and z as two distinct functions, learning the latent dynamics can be interpreted as learning a transform between these function spaces. This can be written using the associated kernel and its discretized operator. On its own, this perspective does not endow any obvious advantages just yet. Interestingly, we find that for nearly all applications tackled so far via NRDE/NCDE approaches (as well as for coupled differential equations), restricting the aforementioned operator to a class (e.g., Calderon-Zygmund(CZ)) which admits a structured decomposition suffices. In particular, a key result from Beylkin, Coifman and Rokhlin referred to as the BCR algorithm (Beylkin et al., 1991) is effective for rapid calculation of integral transforms. We describe how a neural variant, derived from the BCR algorithm, yields excellent performance on long sequence tasks, achieving sizable runtime benefits (up to a few orders of magnitude). An overview of the proposed framework is shown in Fig. 1. Further, tasks such as autoencoding and reconstruction that are difficult using log-signatures (due to invertibility) are possible and quite efficient. Finally, new capabilities that emerge include modeling coupled differential equations.

2. Review: Calderon-Zygmund Operators and Non-standard Wavelet Forms

CZ operators: In harmonic analysis, singular integrals can be considered as an integral operator (transform) $\mathcal{A}$:

$$\mathcal{A}(f)(t) = \int a(t,s)f(s)\mathrm{d}s \tag{5}$$

where the kernel function/operator $a(t,s)$ is singular along the diagonal $t = s$. We briefly review Calderon-Zygmund

(CZ) operators which are central to our model. CZ operators are singular integrals with the property that the associated $a(t,s)$ is smooth away from the diagonal and satisfies:

$$\begin{aligned} |a(t,s)| &\leq \frac{1}{|t-s|} \\ |\partial_t^M a(t,s)| + |\partial_s^M a(t,s)| &\leq \frac{C_0}{|t-s|^{1+M}} \end{aligned} \tag{6}$$

where C_0 is some constant greater than zero and M is an integer greater than 1 which corresponds to the M-th partial derivative. This property enables compression of integral operators when expressed in their non-standard form, described next.

Multi-resolution analysis (MRA): Consider a scaling function (or father wavelet) $\phi(\tau)$ which is smooth, where τ is the dimension over which we seek to perform MRA. The scaled and translated copies of the scaling function can be represented as:

$$\phi_k^l(\tau) = 2^{l/2}\phi(2^l\tau - k) \tag{7}$$

where $l \in \{0, 1, \cdots\}$ denotes the level of decomposition (scaling) associated with the MRA scheme and $k \in \mathbb{Z}$ denotes the translation along the dimension of analysis. At each level l, the set of scaling functions ϕ_k^l forms the basis for the vector space V^l. We get a nested sequence of subspaces with $V^l \subset V^{l+1}$. Hence, $\phi(\tau) \in V^0$ has a representation in terms of a linear combination of the basis functions in V^1, known as the two-scale equation (or refinement equation or dilation relation) as follows:

$$\phi(\tau) = \sum_{k\in\mathbb{Z}} h_k \sqrt{2}\phi(2\tau - k) \tag{8}$$

where h_k are coefficients; they are nonzero only in the associated support of the wavelet family under consideration. It is this compact support which enables "local" analysis through wavelets otherwise absent in Fourier based schemes. The wavelet function (or mother wavelet) is defined in terms of the scaling function as:

$$\psi(\tau) = \sum_{k\in\mathbb{Z}} g_k \sqrt{2}\phi(2\tau - k) \tag{9}$$

with associated coefficients g_k, which are related to h_k by the quadrature mirror relation:

$$g_k = (-1)^{1-k} h_{1-k} \tag{10}$$

Similar to (7), the scaled and translated versions of the wavelet function can be represented as:

$$\psi_k^l(\tau) = 2^{l/2}\psi(2^l\tau - k) \tag{11}$$

for $l \in \{0, 1, \cdots\}$ (corresponding to scale) and $k \in \mathbb{Z}$ (corresponding to translation). At a given level l, the wavelet functions ψ_k^l form a basis for the difference space, $W^l = V^{l+1} \ominus V^l$. This material is covered in detail in (Daubechies, 1992; Mallat, 1999; Alexander & Poularikas, 1998) .

Non-standard form: Wavelets in higher dimensions are often composed by combining the MRA scheme for 1D, defined in terms of scaling functions ϕ_k^l and wavelet functions ψ_k^l along multiple dimensions. We will focus on the 2D case, where there are two ways to arrive at a 2D decomposition scheme: **(a)** standard form and **(b)** non-standard form. For a given number of levels L of analysis, the standard form involves the repeated application of a 1D wavelet transform in the first dimension, L times, followed by the repeated application of 1D wavelet transform in the second dimension. In contrast, the non-standard form involves the alternative application of 1D wavelet transform in either dimension until the desired level, L is reached. The standard basis functions are obtained by the Cartesian product of the 1D wavelets and the father wavelet, whereas the non-standard basis functions at a particular level are obtained from the Cartesian product of scaling and wavelet functions at the *same* level (Kopp & Purgathofer, 1998). *Why is this relevant?* Like in (Fan et al., 2019), we will use a non-standard 2D MRA to compress CZ operators. The beauty of the *non-standard* form lies in the decoupling among different levels, achieved during decomposition followed by a simple coupling mechanism during reconstruction, this will manifest in a recursive relation (Beylkin et al., 1991). On the other hand, the *standard* form leads to interactions between different levels.

3. From CDEs to Integral Transform

Notations: We consider our observations to be sequential measurements, i.e., each sample can be represented as $\mathbf{x} \coloneqq (x_0, x_1, x_2, \ldots, x_T)$, where T denotes the sequence length for the time series. Since the samples may not be univariate, for $(x_0, \cdots, x_i, \ldots, x_T)$, we use D to denote the dimensionality (e.g., number of channels) i.e., $x_i \in \mathbb{R}^D$. Let $\mathbf{X} : t \to \mathbb{R}^D$ be a "path" – a continuous differentiable function of bounded variation $t \in [0, \cdots, T]$ (which may represent time). Note that $\mathbf{x}$ is often assumed to be a discretization of an underlying process, observed only through $\mathbf{x}$, and $\mathbf{X}$ is an approximation to this underlying process, usually obtained via interpolation (Kidger et al., 2020).

Latent space modeling of CDEs: NCDEs (Kidger et al., 2020) model the dynamics in some latent space, say of dimensionality d, that lead to the observations provided. The trajectory in the latent space will be $\mathbf{z} \coloneqq (z_0, \cdots, z_i, \cdots, z_T)$, where $z_i \in \mathbb{R}^d$. Let

$$g_\theta : \mathbb{R}^D \to \mathbb{R}^d \tag{12}$$

be a neural network parameterized by θ, such that, we have,

the initial latent variable (at the start of the process) as:

$$z_0 = g_\theta(\mathbf{X}(0)) \equiv g_\theta(x_0); \quad z_0 \in \mathbb{R}^{d\times 1}, \mathbf{X}(0) \in \mathbb{R}^{D\times 1} \tag{13}$$

where $x_0 \in \mathbb{R}^D$ is the input observed at the first time step. For simplicity, we use $t = 0$ for the first time instance in the trajectory (but in general, it can be arbitrary). Next, let

$$f_\theta : \mathbb{R}^d \to \mathbb{R}^{d\times D} \tag{14}$$

be any neural network depending on model parameters θ. We use θ as a generic placeholder variable to denote the parameterization of the corresponding neural network. Then, as described in (3) the NCDE model is:

$$z_t = z_0 + \int_0^t f_\theta\left(z_s\right) \mathrm{d}\mathbf{X}_s \quad ; t > 0 \tag{15}$$

Assume $\mathbf{X}$ is differentiable and using $\frac{\mathrm{d}\mathbf{X}}{\mathrm{d}s}(s) \equiv \mathbf{X}'(s)$,

$$z_t = z_0 + \int_0^t f_\theta\left(z_s\right) \frac{\mathrm{d}\mathbf{X}}{\mathrm{d}s}(s)\mathrm{d}s = z_0 + \int_0^t f_\theta\left(z_s\right) \mathbf{X}'(s)\mathrm{d}s \tag{16}$$

The identity in (16) above is the NCDE in (Kidger et al., 2020). It is in the form of a definite integral, where the limits of integration are from start of trajectory $t_0 = 0$ to the point of evaluation t, and solved using numerical solvers.

Transformation between function spaces: Let us define another neural network parameterized by θ in the following way,

$$h_\theta : \mathbb{R}^d \to \mathbb{R}^{D\times k} \tag{17}$$

For the f_θ term in (16), we propose to define:

$$f_\theta(z_s) = a_\theta(t,s) \times (h_\theta(z_s))^T; \quad a_\theta(t,s) \in \mathbb{R}^{d\times k} \tag{18}$$

where $a_\theta(t,s)$ is parameterized by θ (using θ as a generic placeholder), $f_\theta(\cdot) \in \mathbb{R}^{d\times D}$ and $h_\theta(\cdot) \in \mathbb{R}^{D\times k}$. The non-linearities in f can be absorbed within a_θ and/or h_θ. Then, (16) can be cast as:

$$z_t = z_0 + \int_0^t a_\theta(t,s)\ (h_\theta(z_s))^T\ \mathbf{X}'(s)\mathrm{d}s \tag{19}$$

Now, since s is a dummy variable of integration, we may combine the derivative of the path and a function of the hidden state, and denote the result as v_s,

$$(h_\theta(z_s))^T \times X'(s) := v(s) \equiv v_s \tag{20}$$

where $X'(s) \in \mathbb{R}^{D\times 1}$ and $v_s \in \mathbb{R}^{k\times 1}$. Substituting back,

$$z_t = z_0 + \int_0^t a_\theta(t,s)v(s)\mathrm{d}s \tag{21}$$

Simply by defining $u_t = z_t - z_0$ gives us

$$u_t = \int_0^t a_\theta(t,s)v(s)\mathrm{d}s \tag{22}$$

Instead of viewing the integral in $s \in [0,t]$, we can view that the integration is carried out over the entire domain $s \in \Omega$ with appropriate restrictions on the operator, presented shortly. This gives us a parameterized integral transform,

$$u_t = \int_{s\in\Omega} a_\theta(t,s)v(s)\mathrm{d}s \tag{23}$$

Takeaway: With (23) we have written the CDE as an integral transform, taking us from $v(s)$ to $u(t)$ with kernel $a_\theta(t,s)$. Do the simple manipulations above offer any benefits? In fact, it may even be computationally *more* challenging to solve (23) than to directly deploy mature numerical solvers on (16). We will see shortly that we must constrain the class of the operator a_θ we will consider – and if we do so, significant benefits are available. This observation is what makes the approach efficient and practical.

Remark 3.1. If $d = k = 1$, then the kernel is a scalar valued kernel, otherwise it will be a tensor-valued kernel (which needs some specific implementation strategies). For simplicity, we will consider the scalar valued kernel for the remainder of the presentation.

4. Leveraging the Non-standard form

Computationally, to apply the integral transform (23), we must solve it via discretization. If we consider discretization using the (given) frequency/sampling rate of the observations, i.e., T discrete values, we can represent (23) as:

$$\mathbf{u} = \mathbf{A}\mathbf{v} \tag{24}$$

where $\mathbf{u} \in \mathbb{R}^T$, $\mathbf{v} \in \mathbb{R}^T$ and $\mathbf{A} \in \mathbb{R}^{T\times T}$. This matrix-vector product involves $O(T^2)$ operations. For large values of T (long sequences), the situation does not look promising.

4.1. MRA using the BCR algorithm

Unrolling the differential equation along time and solving it at once appears difficult, but a closer look reveals extensive structure that can be exploited. In a seminal work (Beylkin et al., 1991), Beylkin, Coifman and Rokhlin (BCR) exploited sparse representations and corresponding algorithms for rapidly performing integral transforms, alternatively viewed as application of dense matrices to vectors as we have in (24). Depending on the type of operator, the BCR scheme can yield remarkable efficiency. The BCR algorithm uses a class of orthonormal wavelet bases, to be specific, the ones constructed by I. Daubechies (Daubechies, 1988), popularly referred to as "db$n$" wavelets, where n corresponds to the number of vanishing moments ($2n$ is the

corresponding support), e.g., "db2". The main observation in the result was that for the class of orthonormal wavelets, the corresponding ***non-standard*** 2D multi-resolution analysis (MRA) of CZ integral operators exhibit a diagonally banded structure and hence can facilitate the fast application of dense matrices to arbitrary vectors.

Consider, the non-parameterized version of (23) with kernel $a(t,s)$ and the corresponding integral operator denoted by A. Then, at any level l of 2D MRA of A, we have:

$$\begin{aligned}\alpha^l_{km} &= \iint \psi^l_k(t)a(t,s)\psi^l_m(s)\mathrm{d}t\mathrm{d}s\\ \beta^l_{km} &= \iint \psi^l_k(t)a(t,s)\phi^l_m(s)\mathrm{d}t\mathrm{d}s\\ \gamma^l_{km} &= \iint \phi^l_k(t)a(t,s)\psi^l_m(s)\mathrm{d}t\mathrm{d}s\\ A^l_{km} &= \iint \phi^l_k(t)a(t,s)\phi^l_m(s)\mathrm{d}t\mathrm{d}s\end{aligned} \tag{25}$$

where $\alpha^l_{km}, \beta^l_{km}, \gamma^l_{km}, A^l_{km}$ are the coefficients obtained by projecting the integral operator A along the wavelet bases of level l.

Use in our case: We observe that when modeling temporal data, it is critical to preserve memory of the measurements for the time points that are far away from the current time point in $[0,T]$. However, it is also sensible to assume that the magnitude of information flow, both through zeroth-order and higher order derivatives, across the sequence has a decreasing but non-zero value when the distance between time points of interest is large. This is precisely the property associated with Calderon-Zygmund kernels (CZ) (6). This informs the choice of our kernel – we assume that our data will admit modeling using the CZ operators in our integral transform in (23).

Due to BCR (Beylkin et al., 1991), for the MRA scheme, we will have the following estimates $\forall |k-m| \geq 2M$

$$|\alpha^l_{km}|+|\beta^l_{km}|+|\gamma^l_{km}| \leq \frac{C_M}{1+|k-m|^{1+M}} \tag{26}$$

Hence, we can consider the coefficient matrices at level l, i.e., $\alpha^l, \beta^l, \gamma^l$ to be diagonally banded with a band length of $B \geq 2M$. This is precisely the sparse representation we exploit; it admits an error bound of $\frac{C}{B^M}\log_2(T)$, where C is a constant dependent on the kernel $a(t,s)$ and the original operator A in its discrete form (has a dimension of $T \times T$). The approximation in (26) holds for all k,m pairs via an extra boundedness condition, see (Beylkin et al., 1991). When using the BCR algorithm to compute (24) the computational complexity is $O(T)$ (Beylkin et al., 1991; Fan et al., 2019).

Remark 4.1. The non-standard form of BCR has been used in the context of neural networks in (Fan et al., 2019), albeit for a completely different purpose. In our work, we use a different parameterization which is efficient and circumvents issues like over-fitting, and we will give details later. Moreover, a closer look at (Fan et al., 2019) suggest other overfitting-related issues in the prior work, which our proposal resolves, the specifics of which are described next.

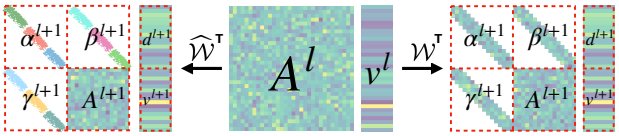

Figure 2. The application of dense matrix to an arbitrary vector via BCR based decomposition is shown towards right. Note the diagonally banded structure (26,28). Towards the left, we have our approximate efficient re-parameterization of the diagonally banded matrix via Partially Un-shared Convolution (PUC). Note that segments of same color in the banded diagonals of $\alpha^{l+1}, \beta^{l+1}, \gamma^{l+1}$ share parameters.

5. Deriving an Efficient (Deep) BCR Algorithm

Consider a $1D$ forward/inverse transform of orthogonal wavelets $\mathcal{W}^T$ and $\mathcal{W}$ respectively denoting the filters (both scaling and wavelet functions). Then, the $2D$ MRA of the operator A can be written as (Beylkin et al., 1991; Fan et al., 2019):

$$\mathcal{W}^T A^l \mathcal{W} = \begin{bmatrix}\alpha^{l+1} & \beta^{l+1}\\ \gamma^{l+1} & A^{l+1}\end{bmatrix}; A^l = \mathcal{W}\begin{bmatrix}\alpha^{l+1} & \beta^{l+1}\\ \gamma^{l+1} & A^{l+1}\end{bmatrix}\mathcal{W}^T \tag{27}$$

Throughout, we assume that $l=0$ corresponds to the finest resolution, and increasing l denotes coarser resolutions. While there is no restriction for the wavelets to be fixed at all levels, for simplicity of presentation, we assume that the same pair of forward and inverse wavelet transform are used at all levels. Hence, the application of the operator A on any vector v at a finer level l, can be written in terms of a coarser level $l+1$ as (Beylkin et al., 1991; Fan et al., 2019):

$$\begin{aligned}u^l = A^l v^l &= \mathcal{W}\begin{bmatrix}\alpha^{l+1} & \beta^{l+1}\\ \gamma^{l+1} & A^{l+1}\end{bmatrix}(\mathcal{W}^T v^l)\\ &= \mathcal{W}\left(\begin{bmatrix}\alpha^{l+1} & \beta^{l+1}\\ \gamma^{l+1} & 0\end{bmatrix}\begin{bmatrix}d^{l+1}\\ v^{l+1}\end{bmatrix} + \begin{bmatrix}0\\ u^{l+1}\end{bmatrix}\right)\end{aligned} \tag{28}$$

Remark 5.1 (Sparsity/MRA). This yields a recursive relation. We should not instantiate the dense operator A at the finest level. The dense operations are relegated to a much coarser level where the dimensionality is small. Also, note from (26) that the matrix of coefficients $\alpha^l, \beta^l, \gamma^l$ are all diagonally banded, so we have avoided the computational burden present in a naive unrolling, see Fig. 2. In doing so, the parameterization is required only at the coarsest level, L, denoted by d_θ, signifying that this is a dense FC layer.

Other modules needed: The other pieces of parameterization involve the diagonally banded matrices $\alpha^l, \beta^l, \gamma^l$ at

each level l of the MRA. These are convolutions with filters depending on the position called Locally Connected Layers in (Fan et al., 2019) (unshared convolution). Essentially, these are diagonally banded matrices corresponding to linear layers. However, we find that this is heavily over-parameterized and performs poorly, leads to over-fitting, and other problems. We instead propose, a *Partially Un-shared Convolution* (**PUC**) layer, which is neither a linear layer (weights vary across the sequence length), nor a pure convolution (weights are shared). In PUC, weights are shared for a partial length of the sequence before they are unshared, this is repeated a small number of times. While it may seem that this will reduce the capacity of the model, we find that this small change is highly effective. One explanation is that the kernel should not change drastically for every time point in the sequence (i.e., there is some continuity in the function that we are trying to estimate).

Algorithm 1 BCR-DE

1: **Input:** Set of sequence $\{\mathbf{x}^n\}_{n=1}^N$, where each $\mathbf{x}^n := (x_0^n, x_1^n, \dots x_T^n)$, with $x_i^n \in \mathbb{R}^D$, T is sequence length.
2: **Input:** L, the number of levels of decomposition
3: **Input:** $\mathcal{W}^T$ and $\mathcal{W}$, $1D$ forward and inverse wavelet transform respectively.
4: **Input:** $g_\theta, h_\theta, d_\theta, \mathcal{L}_\theta, r_\theta$ all learnable modules
5: **Compute:** Continuous approximation of each observation $\mathbf{x}^n$ and denote it by $\mathbf{X}^n$
6: **Compute:** $\mathbf{z}^n = g_\theta(\mathbf{x}^n)$
7: **Compute:** $\mathbf{X}'^n = derivative(\mathbf{X}^n)$
8: **Compute:** $\mathbf{v}^n = (h_\theta(\mathbf{z}^n))^T \times \mathbf{X}'^n$
9: **Initialize:** approx = [], detail = []
10: **for** $l = 1$ **to** $l = L$ **do**
11: $\quad \mathbf{a}^n, \mathbf{d}^n = \mathcal{W}^T(\mathbf{v}^n)$ // a:approx, d:detail coefficient
12: $\quad$ approx.append($\mathbf{a}^n$), detail.append($\mathbf{d}^n$)
13: $\quad \mathbf{v}^n = \mathbf{a}^n$
14: **end for**
15: **Compute:**$\mathbf{u}^n = d_\theta(\mathbf{a}^n)$ // u:transformed sequence
16: **for** $l = L$ **to** $l = 1$ **do**
17: $\quad \begin{bmatrix}\mathbf{d}^n \\ \mathbf{a}^n\end{bmatrix} = \mathbf{PUC}(\mathcal{L}_\theta[l], \begin{bmatrix} detail[l] \\ approx[l]\end{bmatrix}) + \begin{bmatrix} 0 \\ \mathbf{u}^n\end{bmatrix}$
18: $\quad \mathbf{u}^n = \mathcal{W}\left(\begin{bmatrix}\mathbf{d}^n \\ \mathbf{a}^n\end{bmatrix}\right)$
19: **end for**
20: **Output:** $r_\theta(\mathbf{u}^n)$

Algorithm 2 Partially Un-shared Convolution **(PUC)**

1: **Input:** $\mathcal{L}_\theta[l], \begin{bmatrix}\mathbf{d}^n \\ \mathbf{a}^n\end{bmatrix}$
2: **Compute:** $\mathbf{d}'^n = \mathcal{L}_\theta[l][\alpha]\mathbf{d}^n + \mathcal{L}_\theta[l][\beta]\mathbf{a}^n$
3: **Compute:** $\mathbf{a}'^n = \mathcal{L}_\theta[l][\gamma]\mathbf{d}^n$
4: **Output:** $\begin{bmatrix}\mathbf{d}'^n \\ \mathbf{a}'^n\end{bmatrix}$)

Our algorithm: With all modules in hand, we can now present Algorithm 1. In lines 5–8, we compute the function in the v space using the above procedure. Thereafter, lines 9–14 represent the forward wavelet transform, where the details and approximations are stored for future computations. Note that the parameters in $\mathcal{L}_\theta$ can be indexed by the corresponding level l, which further contains parameters for the diagonally banded matrices represented by $\alpha^l, \beta^l, \gamma^l$, (28). Partially Un-shared convolution (PUC) layer is best understood through Fig. 2 and Algorithm 2. Line 18 corresponds to the inverse wavelet transform. The final output is obtained after applying a learnable transform.

Implementation detail: For all experiments shown here, we parameterized g_θ, h_θ and d_θ using fully connected layers, along with $\tanh$ non-linearities. For each level l of $\mathcal{L}_\theta$, we parameterize using our PUC layer. Finally, the last module r_θ is also parameterized using fully connected layers.

5.1. Discussion/Wrapping up

In Section 3, we discussed how apart from restricting our operator to be a CZ operator, one may impose some additional restrictions. These restrictions are particularly relevant when one wants to update the latent variables *only* based on the past values (for some time point) and not future time points ($> t$). This boils down to the discretized operator being lower triangular. In our non-standard multi-resolution decomposition of the operator, this roughly translates to *band diagonal* matrices $\alpha^l, \beta^l, \gamma^l$ at various levels l, except for the very coarse matrix A_L, where it is lower triangular (although this does not offer any noticeable benefits), see Appendix A.

Remark 5.2. Our choice of operators results in diagonally banded matrices based on (6 , 26). It is convenient to think of them as diagonally dominant matrices, common in finite element methods used to solve differential equations.

Remark 5.3. CDEs require the differentiable approximation. We assume a differentiable path, but the requirement is weaker since our discretization is fixed and we only need derivatives at the known points. Even if the path is not differentiable, in practice, we can use finite differences. In CDEs, during training, the sequence length is fixed but because of the use of adaptive step size solvers in most cases, a different number of steps may be needed. BCR-DE is solver-free because it is fixed length and also fixed (although not necessarily uniform) discretization.

A note on Wavelet Basis. Recall that RDEs involve the log-signature transform for the specific choice of the basis function (pre-processing step). In our model, the use of wavelets to model the dynamics allows the basis functions to be learnable. Further, both the forward wavelet transform and its accompanying inverse transform are efficient and allow circumventing invertibility issues, which offers benefits for auto-encoders and coupled differential equations experiments.

Model	RMSE			Time (hrs)		
	RR	HR	SpO_2	RR	HR	SpO_2
ODE-RNN (s512)	1.66 ± 0.06	6.75 ± 0.9	1.98 ± 0.31	0.0	0.1	0.1
NCDE (s1)	2.79 ± 0.04	9.82 ± 0.34	2.83 ± 0.27	23.8	22.1	28.1
NCDE (s512)	2.53 ± 0.03	12.22 ± 0.11	2.98 ± 0.04	0.1	0.0	0.1
NRDE (d3s8)	2.42 ± 0.19	7.67 ± 0.40	2.55 ± 0.13	2.9	3.2	3.1
NRDE (d3s128)	$\mathbf{1.51 \pm 0.08}$	$\mathbf{2.97 \pm 0.45}$	1.37 ± 0.22	0.5	1.7	1.7
NRDE (d3s512)	$\mathbf{1.49 \pm 0.08}$	3.46 ± 0.13	$\mathbf{1.29 \pm 0.15}$	0.3	0.4	0.4
BCR-DE	$\mathbf{1.53 \pm 0.09}$	$\mathbf{3.27 \pm 0.16}$	$\mathbf{1.18 \pm 0.15}$	0.4	0.5	0.9

Table 1. Mean and standard deviation of RMSE over three runs and mean training time for prediction of vital signs RR, HR, SpO2. Note: Contents in parenthesis beside baseline models depict their variants; 's' for step size and 'd' for depth whenever applicable.

6. Experimental evaluations

We present evaluations on a wide variety of experimental settings from prediction to autoencoding to modeling coupled differential equations, to assess the effectiveness and capability of our proposed method.

6.1. Prediction using Medium length/Long Sequences

6.1.1. Physiological Measurements: Regression

(a)–(b) Dataset and Setup. We evaluate a regression task on data from Beth Israel Deaconess Medical Centre (BIDMC), see (Tan et al., 2020). The three different tasks involve predicting the average values of (i) Respiratory Rate (RR), (ii) Heart Rate (HR) and (iii) oxygen saturation (SpO2) based on a participant's PPG and ECG data over a long time interval. The original data as collected by BIDMC was sampled at 125Hz, and (Tan et al., 2020) converts/provides a time series of length 4000 for a sliding window of 32 seconds which we use in our experiments.

(c) Main findings. We report baseline results for ODE-RNN (Rubanova et al., 2019), NCDE, NRDE from (Morrill et al., 2021) in Table 1. For each baseline, we report both its best performing variant and its most efficient variant (training time). The data are moderately long sequence lengths (4000), so we should only expect parity. Our model (BCR-DE, in blue) achieves comparable performance relative to NRDE (with occasional runtime benefits) while improving performance (Root Mean Square Error (RMSE) on test set) over alternatives by a noticeable margin.

6.1.2. EigenWorms: Classification

(a) Dataset. We use the EigenWorms dataset from (Bagnall et al., 2017) with a much longer sequence length of 17984 where the task is to classify a worm into one of the 5 types, one wild and four other mutant types based on motion capture data originally collected in (Brown et al., 2013).

(b) Setup. *Baselines.* As in §6.1.1, we report baseline results for ODE-RNN (Rubanova et al., 2019), NCDE, NRDE from (Morrill et al., 2021) in Table 2 including both best and the most efficient variants. *Our model setup.* Our model works in the latent space of wavelet coefficients at multiple resolutions. Perturbing these coefficients during training helps regularize the model. We train a variant of our model, BCR-DE (Noise), and report both models but with no hyperparameter search for our model as in (Morrill et al., 2021).

Model	Accuracy (%)	Time (hrs)
ODE-RNN (s128)	47.9 ± 5.3	0.01
NCDE (s4)	66.7 ± 11.8	5.5
NCDE (s128)	48.7 ± 2.6	0.1
NRDE (d2s4)	$\mathbf{83.8 \pm 3.0}$	2.4
NRDE (d3s128)	68.4 ± 8.2	0.1
BCR-DE	77.8 ± 1.2	0.01
BCR-DE (Noise)	78.7 ± 2.4	0.01

Table 2. Mean (and s.d.) of test set accuracy over three repeats and average training time. Parenthesized text indicates baseline model variants; 's' for step size and 'd' for depth.

(c) Main findings. Results are reported in Table 2. BCR-DE achieves a comparable performance to the best baseline of NRDE but with a runtime that is two orders of magnitude faster. We see there is a sizable variance in performance due to the small dataset size (181 train/40 test samples).

6.2. Auto-Encoder for Medium Length Sequences

We consider several use-cases of sequence to sequence modelling from the perspective of auto-encoders for time-series.

(a) Dataset. Similar to §6.1.1, we use the BIDMC32 (Tan et al., 2020) data (sequence length of 4000).

(b) Setup. We evaluate the performance metric of Mean Squared Error (MSE) along with training time for NCDE, NRDE and BCR-DE for six different setups. In all cases, we train until convergence, or a maximum budget of 100 epochs, whichever is smaller. A couple of adjustments to baselines were needed, described next.

The experimental settings in NCDE (Kidger et al., 2020) and NRDE (Morrill et al., 2021) did not involve sequence to sequence tasks, hence we made some adjustments for fair comparisons. **(i)** To give each model the same expressive power, we follow (Morrill et al., 2021) and compare models with similar number of parameters. **(ii)** We adjusted the output of NCDE/NRDE to output sequences (small adjustments to the code). **(iii)** NRDE creates summaries over segments of the temporal sequence. So, the effective length of the sequence is reduced due to log-signatures. We report performance on the sub-sampled data with the caveat that we do not use NRDE's full trajectory (for the full sequence length). This is because of invertibility of the signature transform, see (Kidger et al., 2019). We also present the baseline models with time as an additional channel in the input as used in (Kidger et al., 2020; Morrill et al., 2021). Our model (BCR-DE) does not need this channel. **(iv)** To use NRDE as a

Task	Dataset	NCDE		NRDE		BCR-DE	
		MSE	Time (hrs)	MSE	Time (hrs)	MSE	Time (hrs)
AE	PPG	**6.05e-5**	3.63	0.014	0.67	0.012	0.2
	ECG	**6.06e-5**	3.03	0.014	0.57	0.024	0.19
DAE	PPG	**0.008**	4.1	0.023	0.92	**0.009**	0.18
	ECG	**0.008**	3.04	0.023	0.73	0.02	0.18
MAE	PPG	0.28	2.23	0.106	5.47	**0.024**	0.22
	ECG	0.29	1.5	0.106	3.76	**0.097**	0.23

Table 3. MSE on test set and training time for auto-encoding, denoising auto-encoding and masked auto-encoding for PPG/ECG data. BCR-DE gives comparable or better performance but needs a much lower runtime.

baseline, we fix a depth of 3 and a reasonable step size of 50. **(v)** We use the adjoint method for the baselines. Otherwise, for medium length/long sequences, memory becomes prohibitive. Avoiding the adjoint method offers some speed-up but running medium length/long sequences is challenging. **(vi)** We use the fixed step size solver from the Runge–Kutta family "RK4" (Press et al., 1992) in all of our baseline experiments. Adaptive step size solvers like Dormand-Prince method "DOPRI5" (Dormand & Prince, 1980) yield better results, but the compute budget for running the full set of experiments we present here was excessive.

6.2.1. Reconstruction

This is a vanilla auto-encoder for time-series data. We consider both PPG and ECG data separately.

(c) Main findings. As can be seen in Table 3, NCDE can achieve good performance but with an order of magnitude higher training time. NRDEs can achieve it a bit faster but note that the reconstruction is only evaluated on a subsample of the original length. BCR-DE achieves good reconstruction in the shortest time. Note that for NCDE, the error is very small and it was achieved in only 2 epochs.

6.2.2. Denoising

Next, we check the model in a denoising autoencoding setup. We add standard Gaussian noise scaled by 0.1. Ablation studies (different scales of noise) are in the Appendix.

(c) Main findings. Table 3 shows that our model achieves good denoising, but in orders of magnitude less time.

6.2.3. Masked reconstruction

We now check the model's efficacy in performing masked reconstruction, where we mask a fixed length (50) segment of the data . In many continuous sensor measurement settings, the sensor may get switched off or deactivated. We leverage the fact that we model the dynamics using wavelet coefficients, and so we can adjust our loss to minimize total variation (TV) and ℓ_1-norm of the details apart from the standard reconstruction loss (Ding & Selesnick, 2015). In doing so, we are able to faithfully capture the overall dynamics and produce reasonable reconstructions.

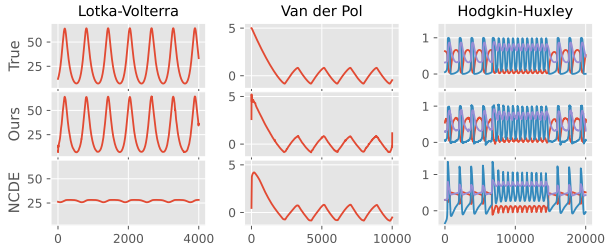

Figure 3. Comparing predicted trajectories for different coupled differential equations with simulated ground truth. As one can see in all the cases, our method (BCR-DE) can almost perfectly match the true trajectory. It is better than NCDE and does so in much less time. Note: Visualization of the trajectory from NRDE is omitted as the output is a sub-sample of the original sequence length.

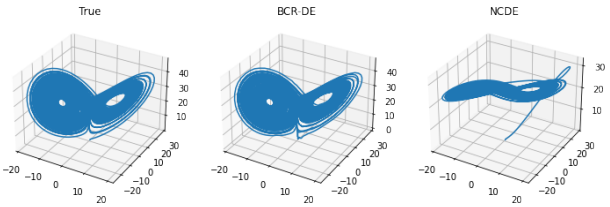

Figure 4. Comparison of trajectories for a chaotic Lorenz system. BCR-DE can match the original trajectory almost exactly.

(c) Main findings. From Table 3, we see that BCR-DE achieves the best performance in the least time.

6.3. Coupled Differential Equations

In many common dynamical systems, we may have multiple dependent variables and one independent variable (time). The measurements may be described by a system of differential equations, or coupled differential equations. Since we model the operator of an integral transform, we hypothesize that BCR-DE may model the behavior in coupled differential equations. Here, we check if this is the case.

(a)-(b) Dataset and Setup. We now list the various instances used in our experiments: (i) Toy DE; (ii) Lotka-Volterra; (iii) Van der Pol Oscillator; (iv) Lorenz System; (v) Hodgkin-Huxley model; (vi) Temperature/Relative-Humidity sequence. Details are given in Appendix C. In all cases for simulated data, we use 2000 training samples and 1000 samples each for validation and test. Adam (Kingma & Ba, 2014) is used as an optimizer with weight decay of 0.0001 and we use a learning rate of 0.01. We reduce the learning rate when the validation loss plateaus. We use MSE as the loss function.

(c) Main findings. The MSE and training time for all settings is reported in Table 4. We see that BCR-DE is very efficient and gives small MSE values. Comparisons with ground truth trajectories are shown in Fig. 3 and Fig. 4.

Setting (Seq Len)	NCDE		NRDE		BCR_DE	
	MSE	Time (hrs)	MSE	Time (hrs)	MSE	Time (hrs)
Toy Coupled DE (4k)	1e-4	0.62	**6e-5**	0.02	3e-4	0.009
Lotka-Volterra (4k)	377.9	43.74	365.4	3.04	**0.19**	0.134
Van der Pol (10k)	0.023	43.6	0.94	7.43	**1e-3**	0.34
Chaotic Lorenz (10k)	66.15	42.9	133.3	3.7	**0.05**	0.35
Hodgkin-Huxley (20k)	0.02	45.35	1.24	4.28	**4e-4**	0.35
Benzene Conc. (240)	250.3	3.64	725.4	1.17	**212.9**	0.046

Table 4. MSE on test set and train time for different settings of coupled differential equations. In almost all cases, BCR-DE achieves the best performance (MSE) in significantly less time.

6.4. Performance trade-offs w.r.t. decomposition levels

We also studied the dependence between the levels of decomposition in BCR-DE with the model performance using the Hodgkin-Huxley coupled differential equation as an example. For a sequence length of 20K, with only two levels of decomposition, the dense matrix is too large to be instantiated (due to GPU memory limits). For level 3, the model has 225M parameters and does not perform well. For level 6, the model has roughly 3M parameters and we can achieve good MSE of about 1e-2 or lower. When increasing the number of levels to 8 or more, the number of parameters in the model are about 100K and we can achieve, an MSE of about 1e-4. This shows that as the number of levels of resolution is increased, we can achieve the benefits of BCR-DE, however when the levels are too few, the multi-resolution is similar to naive unrolling and faces a computational burden.

7. Related Work

Apart from works described in Section 1, the interface of neural networks and ordinary differential equations is an active topic of research with a number of nice results in the last year alone (Rodriguez et al., 2022; Meunier et al., 2022; Irie et al., 2022). More results are also appearing marrying partial differential equations and neural networks (Li et al., 2020b;c; Brandstetter et al., 2022; Long et al., 2018; Berg & Nyström, 2019). Slightly distinct from this thrust, there is an evolving body of work which takes into account physics based constraints to model dynamical systems (Raissi et al., 2019; Li et al., 2021; Krishnapriyan et al., 2021) as PDEs, often utilizing learning in various ways. The family of generative models popularly known as diffusion models (Croitoru et al., 2022) exploit the dynamics modelled as SDE, and have radically expanded the types of applications that can benefit from results in this area.

Wavelets, which inform our operator compression, have been heavily studied since the 1970s, but also more recently vis-à-vis deep neural networks (Xu et al., 2019; Guo et al., 2017; Hy & Kondor, 2022). Wavelets, similar to their use in §4.1, are emerging as important tools for efficiency gains. For example, (Zeng et al., 2022) provided an approximation mechanism for self-attention via a multi-resolution/wavelet analysis. Recently (Guth et al., 2022) accelerated score-based generative models by leveraging wavelet coefficients across multiple scales. Ideas like (Michau et al., 2022; Pedersen et al.) have proposed parameterizing the wavelet functions and learning them based on data.

8. Conclusions

We give a strategy for strong efficiency gains for a subclass of learning tasks involving long temporal sequences where Neural CDE/Neural RDE models are appropriate. For these problems, especially when the task corresponds to regression or classification, if each longitudinal sequence in the training data is a fixed length of temporal samples, we show that unrolling the dynamics to obtain an integral transform followed by restricting the class of operators, allows the use of a simple neural interpretation of BCR algorithm. We should note that our results do not involve any new insights that allow generic speed-ups for CDEs. Rather, for most use-cases shown in the literature, promising improvements in run-time are possible. We suspect that these advantages may hold in some other cases where learning the dynamics (as accomplished in a model such as NCDE/NRDE) is an intermediary step towards a subsequent prediction/classification goal. **Limitations.** We point out scenarios where the use of BCR-DE offers minimal benefits. First, since we compute the integral transform of the entire sequence at once, it is 2-3$\times$ more memory intensive than baselines. This was not a problem for the experiments reported here (commodity GPUs were sufficient), but may turn out to be a bottleneck for sequences that are an order of magnitude longer. For shorter sequences, there are minimal advantages, if any, since the NCDE runtime/performance is quite good, as shown in our first experiment. In an actual online setting, when new data arrives continually, for each sequence, the model must be retrained. Here, at best, we can choose to retrain lazily and so in the extreme case, the compute benefits will take a large hit. Code is available at https://github.com/sourav-roni/BCR-DE.

Acknowledgments

This work was partially supported by funds from the Vilas Board of Trustees and NIH grant RF1 AG059312. Ravi was supported by UIC startup funds. We thank Cindy Orozco Bohorquez for answering questions regarding (Fan et al., 2019) and Lopa Mukherjee for offering feedback on multiple drafts of the paper. The reviewers were particularly generous with their time in offering numerous detailed suggestions that have significantly improved the paper overall.

References

Alexander, E. and Poularikas, D. The handbook of formulas and tables for signal processing. *Boca Raton, Florida*, 33431, 1998.

Bagnall, A., Lines, J., Bostrom, A., Large, J., and Keogh, E. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data mining and knowledge discovery*, 31(3):606–660, 2017.

Berg, J. and Nyström, K. Data-driven discovery of pdes in complex datasets. *Journal of Computational Physics*, 384:239–252, 2019.

Beylkin, G., Coifman, R., and Rokhlin, V. Fast wavelet transforms and numerical algorithms i. *Communications on pure and applied mathematics*, 44(2):141–183, 1991.

Brandstetter, J., Worrall, D., and Welling, M. Message passing neural pde solvers. *arXiv preprint arXiv:2202.03376*, 2022.

Brown, A. E., Yemini, E. I., Grundy, L. J., Jucikas, T., and Schafer, W. R. A dictionary of behavioral motifs reveals clusters of genes affecting caenorhabditis elegans locomotion. *Proceedings of the National Academy of Sciences*, 110(2):791–796, 2013.

Chang, J., Duffield, N., Ni, H., and Xu, W. Signature inversion for monotone paths. *Electronic Communications in Probability*, 22:1–11, 2017.

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

Croitoru, F.-A., Hondru, V., Ionescu, R. T., and Shah, M. Diffusion models in vision: A survey. *arXiv preprint arXiv:2209.04747*, 2022.

Curtiss, C. F. and Hirschfelder, J. O. Integration of stiff equations. *Proceedings of the National Academy of Sciences*, 38(3):235–243, 1952.

Daubechies, I. Orthonormal bases of compactly supported wavelets. *Communications on pure and applied mathematics*, 41(7):909–996, 1988.

Daubechies, I. *Ten lectures on wavelets*. SIAM, 1992.

Ding, Y. and Selesnick, I. W. Artifact-free wavelet denoising: non-convex sparse regularization, convex optimization. *IEEE signal processing letters*, 22(9):1364–1368, 2015.

Dormand, J. R. and Prince, P. J. A family of embedded runge-kutta formulae. *Journal of computational and applied mathematics*, 6(1):19–26, 1980.

Fan, Y., Bohorquez, C. O., and Ying, L. Bcr-net: A neural network based on the nonstandard wavelet form. *Journal of Computational Physics*, 384:1–15, 2019.

Guo, T., Seyed Mousavi, H., Huu Vu, T., and Monga, V. Deep wavelet prediction for image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 104–113, 2017.

Guth, F., Coste, S., De Bortoli, V., and Mallat, S. Wavelet score-based generative modeling. *arXiv preprint arXiv:2208.05003*, 2022.

Hairer, E. and Wanner, G. Stiff differential equations solved by radau methods. *Journal of Computational and Applied Mathematics*, 111(1-2):93–111, 1999.

Hanson, J. and Raginsky, M. Universal simulation of stable dynamical systems by recurrent neural nets. In *Learning for Dynamics and Control*, pp. 384–392. PMLR, 2020.

Hodgkin, A. L. and Huxley, A. F. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117 (4):500, 1952.

Holt, S. I., Qian, Z., and van der Schaar, M. Neural laplace: Learning diverse classes of differential equations in the laplace domain. In *International Conference on Machine Learning*, pp. 8811–8832. PMLR, 2022.

Hy, T. S. and Kondor, R. Multiresolution matrix factorization and wavelet networks on graphs. In *Topological, Algebraic and Geometric Learning Workshops 2022*, pp. 172–182. PMLR, 2022.

Irie, K., Faccio, F., and Schmidhuber, J. Neural differential equations for learning to program neural nets through continuous learning rules. *arXiv preprint arXiv:2206.01649*, 2022.

ISyE8843A, B. V. H. 1 basics of wavelets.

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

Kidger, P., Bonnier, P., Perez Arribas, I., Salvi, C., and Lyons, T. Deep signature transforms. *Advances in Neural Information Processing Systems*, 32, 2019.

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

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kopp, M. and Purgathofer, W. Interleaved dimension decomposition: A new decomposition method for wavelets and its application to computer graphics. 1998.

Krishnapriyan, A., Gholami, A., Zhe, S., Kirby, R., and Mahoney, M. W. Characterizing possible failure modes in physics-informed neural networks. *Advances in Neural Information Processing Systems*, 34:26548–26560, 2021.

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*, pp. 3870–3882. PMLR, 2020a.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020b.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020c.

Li, Z., Zheng, H., Kovachki, N., Jin, D., Chen, H., Liu, B., Azizzadenesheli, K., and Anandkumar, A. Physics-informed neural operator for learning partial differential equations. *arXiv preprint arXiv:2111.03794*, 2021.

Liang, S. and Srikant, R. Why deep neural networks for function approximation? *arXiv preprint arXiv:1610.04161*, 2016.

Long, Z., Lu, Y., Ma, X., and Dong, B. Pde-net: Learning pdes from data. In *International Conference on Machine Learning*, pp. 3208–3216. PMLR, 2018.

Lyons, T. J. and Xu, W. Inverting the signature of a path. *Journal of the European Mathematical Society*, 20(7): 1655–1687, 2018.

Mallat, S. *A wavelet tour of signal processing*. Elsevier, 1999.

Meng, Z., Singh, V., and Ravi, S. N. Neural tmdlayer: Modeling instantaneous flow of features via sde generators. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 11635–11644, 2021.

Meunier, L., Delattre, B. J., Araujo, A., and Allauzen, A. A dynamical system perspective for lipschitz neural networks. In *International Conference on Machine Learning*, pp. 15484–15500. PMLR, 2022.

Michau, G., Frusque, G., and Fink, O. Fully learnable deep wavelet transform for unsupervised monitoring of high-frequency time series. *Proceedings of the National Academy of Sciences*, 119(8):e2106598119, 2022.

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

Nazarovs, J., Chakraborty, R., Tasneeyapant, S., Ravi, S., and Singh, V. A variational approximation for analyzing the dynamics of panel data. In *Uncertainty in Artificial Intelligence*, pp. 107–117. PMLR, 2021.

Pedersen, C., Eickenberg, M., and Ho, S. Learnable wavelet neural networks for cosmological inference.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. Numerical recipes in c. 2. *Cambrige University*, 1992.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

Rodriguez, I. D. J., Ames, A., and Yue, Y. Lyanet: A lyapunov framework for training neural odes. In *International Conference on Machine Learning*, pp. 18687–18703. PMLR, 2022.

Rubanova, Y., Chen, R. T. Q., and Duvenaud, D. K. Latent ordinary differential equations for irregularly-sampled time series. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/42a6845a557bef704ad8ac9cb4461d43-Paper.pdf.

Salvi, C. *Rough paths, kernels, differential equations and an algebra of functions on streams*. PhD thesis, University of Oxford, 2021.

Salvi, C., Lemercier, M., and Gerasimovics, A. Neural stochastic pdes: Resolution-invariant learning of continuous spatiotemporal dynamics. In *Advances in Neural Information Processing Systems*, 2022.

Song, Y. and Ermon, S. Improved techniques for training score-based generative models. *Advances in neural information processing systems*, 33:12438–12448, 2020.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Tan, C. W., Bergmeir, C., Petitjean, F., and Webb, G. I. Monash university, uea, ucr time series regression archive. *arXiv preprint arXiv:2006.10996*, 2020.

Vahdat, A., Kreis, K., and Kautz, J. Score-based generative modeling in latent space. *Advances in Neural Information Processing Systems*, 34:11287–11302, 2021.

Xu, B., Shen, H., Cao, Q., Qiu, Y., and Cheng, X. Graph wavelet neural network. *arXiv preprint arXiv:1904.07785*, 2019.

Zeng, Z., Pal, S., Kline, J., Fung, G. M., and Singh, V. Multi resolution analysis (mra) for approximate self-attention. In *International Conference on Machine Learning*, pp. 25955–25972. PMLR, 2022.

A. Lower triangular kernel approximations

In this section, we will look at the structure of coefficient matrices α, β, γ from 25 when the kernel is assumed to be lower triangular.

We will work with Daubechies wavelets, which are compactly supported and orthogonal. Note that the analysis will depend on how the compact support is identified, but the procedure will remain the same, with limits being adjusted.

Let, p be a positive integer, the scaling function $\phi(x)$ is supported in $[0, 2p-1]$. Note that here x is a generic variable denoting the dimension over which the wavelet transform is applied, this was denoted as τ in review of MRA in Section 2. Given the scaling function, the mother wavelet function $\psi(x)$ is supported in $[-p+1, p]$. Also, the scaled and translated version of the scaling and wavelet functions can be described as below:

$$\phi_k^{(\ell)}(x) = 2^{\ell/2}\phi\left(2^\ell x - k\right), \quad \ell = 0, 1, 2, \ldots, \quad k \in \mathbb{Z} \tag{29}$$

$$\psi_k^{(\ell)}(x) = 2^{\ell/2}\psi\left(2^\ell x - k\right), \quad \ell = 0, 1, 2, \ldots, \quad k \in \mathbb{Z} \tag{30}$$

We consider the integral operator with the kernel $a(x, y)$ in the periodic interval $[0, T]$. The nonstandarad form is essentially a data sparse representaiton of A using $2D$ multiresolution wavelet basis. We consider lower triangular kernel, which means, we can further write $a(x, y)$ as follows:

$$a(x,y) = \begin{cases} a(x,y) & x \le y \\ 0 & x > y \end{cases} \tag{31}$$

$$A^{(\ell)}_{k_1,k_2} := \int_{y=0}^{T}\int_{x=0}^{T} \phi_{k_1}^{(\ell)}(x)a(x,y)\phi_{k_2}^{(\ell)}(y)\mathrm{d}x\,\mathrm{d}y \tag{32}$$

Using the lower triangular property and arranging the integral, we have:

$$A^{(\ell)}_{k_1,k_2} := \int_{y=0}^{T} \phi_{k_2}^{(\ell)}(y)\left(\int_{x=0}^{y} \phi_{k_1}^{(\ell)}(x)a(x,y)\mathrm{d}x\right)\,\mathrm{d}y \tag{33}$$

$$A^{(\ell)}_{k_1,k_2} := \int_{y=0}^{T} 2^{\ell/2}\phi\left(2^\ell y - k_2\right)\left(\int_{x=0}^{y} 2^{\ell/2}\phi\left(2^\ell x - k_1\right)a(x,y)\mathrm{d}x\right)\,\mathrm{d}y \tag{34}$$

Now, since $\phi(x)$ has support in $[0, 2p-1]$, we can write the following two conditions, where integrands are non zero.

$$\begin{aligned} 0 \le 2^\ell x - k_1 &\le 2p-1 \\ k_1/2^\ell \le x &\le (2p-1+k_1)/2^\ell \end{aligned} \tag{35}$$

Similarly,

$$\begin{aligned} 0 \le 2^\ell y - k_2 &\le 2p-1 \\ k_2/2^\ell \le y &\le (2p-1+k_2)/2^\ell \end{aligned} \tag{36}$$

So, for $A^{(l)}_{k_1,k_2} == 0$, we have:

$$\begin{aligned} (2p-1+k_2)/2^\ell &< k_1/2^\ell \\ 2p-1+k_2 &< k_1 \\ k_1 &> k_2 + (2p-1) \end{aligned} \tag{37}$$

This says, $A^{(l)}$ will primarily be an lower triangular matrix with $2p-1$ diagonals above the lower triangle (potentially non-zero).

Next, for α, we have:

$$\alpha^{(\ell)}_{k_1,k_2} := \int_{y=0}^{T}\int_{x=0}^{T} \psi^{(\ell)}_{k_1}(x)a(x,y)\psi^{(\ell)}_{k_2}(y)\mathrm{d}x\ \mathrm{d}y \tag{38}$$

Using lower triangular property and arrangement, we can write:

$$\alpha^{(\ell)}_{k_1,k_2} := \int_{y=0}^{T} 2^{\ell/2}\psi\left(2^\ell y - k_2\right)\left(\int_{x=0}^{y} 2^{\ell/2}\psi\left(2^\ell x - k_1\right)a(x,y)\mathrm{d}x\right)\ \mathrm{d}y \tag{39}$$

Since, $\psi(x)$ in our case of Daubechies wavelets is supported in $[-p+1,p]$, we can write the following 2 conditions, where integrands are non zero:

$$\begin{aligned} -p+1 \le 2^\ell x - k_1 \le p \\ (-p+1+k_1)/2^\ell \le x \le (p+k_1)/2^\ell \end{aligned} \tag{40}$$

Similarly,

$$\begin{aligned} -p+1 \le 2^\ell y - k_2 \le p \\ (-p+1+k_2)/2^\ell \le y \le (p+k_2)/2^\ell \end{aligned} \tag{41}$$

Then, for $\alpha^\ell_{k_1,k_2} == 0$, we have:

$$\begin{aligned} (p+k_2)/2^\ell &< (-p+1+k_1)/2^\ell \\ p+k_2 &< -p+1+k_1 \\ k_1 &> k_2 + (2p-1) \end{aligned} \tag{42}$$

Again, as before for A^ℓ; α^ℓ will be primarily lower triangular with $2p-1$ diagonals above the lower triangle (potentially non-zero).

Next, for β, we have:

$$\beta^{(\ell)}_{k_1,k_2} := \int_{y=0}^{T}\int_{x=0}^{T} \psi^{(\ell)}_{k_1}(x)a(x,y)\phi^{(\ell)}_{k_2}(y)\mathrm{d}x\ \mathrm{d}y \tag{43}$$

Using lower triangular property and arrangement, we can write:

$$\beta^{(\ell)}_{k_1,k_2} := \int_{y=0}^{T} 2^{\ell/2}\phi\left(2^\ell y - k_2\right)\left(\int_{x=0}^{y} 2^{\ell/2}\psi\left(2^\ell x - k_1\right)a(x,y)\mathrm{d}x\right)\ \mathrm{d}y \tag{44}$$

Based, on the support of ϕ and ψ, we have the following two conditions, where integrand is non zero:

$$\begin{aligned} -p+1 \le 2^\ell x - k_1 \le p \\ (-p+1+k_1)/2^\ell \le x \le (p+k_1)/2^\ell \end{aligned} \tag{45}$$

And,

$$\begin{aligned} 0 \le 2^\ell y - k_2 \le 2p-1 \\ k_2/2^\ell \le y \le (2p-1+k_2)/2^\ell \end{aligned} \tag{46}$$

So, for $\beta^{(\ell)}_{k_1,k_2} == 0$, we have:

$$\begin{aligned} (2p-1+k_2)/2^\ell &< (-p+1+k_1)/2^\ell \\ 2p-1+k_2 &< -p+1+k_1 \\ k_1 &> k_2 + (3p-2) \end{aligned} \tag{47}$$

Again, we have β^ℓ be primarily lower triangular with $3p-2$ diagonals above the lower triangle (potentially non-zero).

Next, for γ^ℓ, we have:

$$\gamma^{(\ell)}_{k_1,k_2} := \int_{y=0}^{T}\int_{x=0}^{T} \phi^{(\ell)}_{k_1}(x)a(x,y)\psi^{(\ell)}_{k_2}(y)\mathrm{d}x\,\mathrm{d}y \tag{48}$$

Using lower triangular property and arrangement, we can write:

$$\gamma^{(\ell)}_{k_1,k_2} := \int_{y=0}^{T} 2^{\ell/2}\psi\left(2^\ell y - k_2\right)\left(\int_{x=0}^{y} 2^{\ell/2}\phi\left(2^\ell x - k_1\right)a(x,y)\mathrm{d}x\right)\,\mathrm{d}y \tag{49}$$

Based, on the support of ϕ and ψ, we have the following two conditions, where integrand is non zero:

$$\begin{aligned} 0 \le 2^\ell x - k_1 &\le 2p-1 \\ k_1/2^\ell \le x &\le (2p-1+k_1)/2^\ell \end{aligned} \tag{50}$$

$$\begin{aligned} -p+1 \le 2^\ell y - k_2 &\le p \\ (-p+1+k_2)/2^\ell \le y &\le (p+k_2)/2^\ell \end{aligned} \tag{51}$$

So, for $\gamma^{(\ell)}_{k_1,k_2} == 0$, we have:

$$\begin{aligned} (p+k_2)/2^\ell &< k_1/2^\ell \\ p+k_2 &< k_1 \\ k_1 &> k_2 + (p) \end{aligned} \tag{52}$$

Hence, we have γ^ℓ be primarily lower triangular with p diagonals above the lower triangle (potentially non-zero).

In the treatment of the lower triangular property above we see that α, β, γ are also lower triangular with few elements above the diagonal non-zero. This coupled with our assumption that the operator is a Calderon-Zygmund operator, provides the banded diagonal approximation for $\alpha^\ell, \beta^\ell, \gamma^\ell$.

B. Experiment Details

B.1. EigenWorms and BIDMC32

The training is performed with learning rate of 0.01, with Adam optimizer and an weight decay of 0.0001. We use learning rate scheduler to reduce learning rate when validation loss plateaus with a patience of 5 and factor of 0.5.

B.2. Long Sequence Autoencoder

Here, we consider the BIDMC32 data presented in (Tan et al., 2020). In all cases we train for 100 epochs with initial lr of 0.01 and use mean squared error as the loss function. Adam as an optimizer with weight decay of 0.0001. Learning rate is reduced when validation loss plateaus with a patience of 5 and factor of 0.5. Size of training set is 5508, while validation and test set have 1181 samples each. Below, we outline, the details of the model(s) which provide the best performance for BCR-DE.

B.2.1. Reconstruction

- **PPG:** Params: 133680; Memory 1445.43 MB, time: 1360.01s, test loss: 0.01039
  Params: 59416; Memory 654.28 MB, time: 726.46s, test loss: 0.012086

- **ECG:** Params: 133680; Memory 1445.43 MB, time: 1331.89s, test loss: 0.019030
  Params: 59416; Memory 654.28 MB, time: 677.89s, test loss: 0.024419

B.2.2. DENOISING (ABLATION)

Next, we demonstrate the capability of our mode in a denoising autoencoding setup, we add standard Gaussian noise at three different scales: 0.01, 0.05, 0.1 and report the best performance of our model below.:

- **PPG (Noise scale: 0.01)** Params: 59416; Memory 655.25 MB, time: 663.46s, test loss: 0.0069
- **PPG (Noise scale: 0.05)** Params: 59416; Memory 655.25 MB, time: 675.24s, test loss: 0.0071
- **PPG (Noise scale: 0.1)** Params: 59416; Memory 655.25 MB, time: 661.70s, test loss: 0.0086
- **ECG (Noise scale: 0.01)** Params: 59416; Memory 655.25 MB, time: 662.42s, test loss: 0.0203
- **ECG (Noise scale: 0.05)** Params: 59416; Memory 655.25 MB, time: 684.86s, test loss: 0.0189
- **ECG (Noise scale: 0.1)** Params: 59416; Memory 655.25 MB, time: 661.64s, test loss: 0.0205

As can be seen from above, our model is robust for considerably high magnitude of noise, and can achieve equivalent performance in all cases irrespective of the amount of denoising.

B.2.3. MASKED RECONSTRUCTION (ABLATION)

Below, we present the best performance of BCR-DE for two different lengths of masked segments, 50 and 100 respectively for both PPG and ECG.

- **PPG (Masked50):** Params: 59416; Memory 704.12 MB, time: 796.62s, test loss: 0.02414
- **PPG (Masked100):** Params: 59416; Memory 704.12 MB, time: 1355.28s, test (recon) loss: 0.02396
- **ECG (Masked50):** Params: 59416; Memory 704.12 MB, time: 831.14s, test loss: 0.09736
- **ECG (Masked100):** Params: 59416; Memory 704.12 MB, time: 2242.7s, test loss: 0.1130

C. Coupled Differential Equation

(i) Toy DE: We consider the following simple system of differential equations with initial conditions:

$$\frac{\mathrm{d}y}{\mathrm{d}t} = -y + 3x; \quad \frac{\mathrm{d}x}{\mathrm{d}t} = 4x - 2y; \quad y(0) = 2 \quad x(0) = 1 \tag{53}$$

which has an analytical solution given by: $y(t) = 3e^t - e^{2t}$ and $x(t) = 2e^t - e^{2t}$. We simulate the data for a sequence length of 4000. We intend to model the dynamics of one stream given the other.

(ii) Lotka-Volterra: We consider the well known predator-prey interactions between Snowshoe Hare and Canadian Lynx involving nonlinear dynamics and given by:

$$\frac{\mathrm{d}H}{\mathrm{d}t} = rH\big(1 - \frac{H}{k}\big) - \frac{aHL}{c+H}; \quad \frac{\mathrm{d}L}{\mathrm{d}t} = a\frac{bHL}{c+H} - dL \tag{54}$$

where we use $a = 3.2, b = 0.6, c = 50, d = 0.56, k = 125, r = 1.6$. We simulate the data using scipy odeint for sequence length of 10000. The goal is to predict the population trajectory of Hare given that of Lynx.

(iii) Van der Pol Oscillator: We consider Van der Pol oscillator which is a non-linear system involving damping and is not conservative. It is a 2^{nd} order differential equation whose 2-D form is:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \mu\big(x - \frac{x^3}{3} - y\big); \quad \frac{\mathrm{d}y}{\mathrm{d}t} = \frac{x}{\mu} \tag{55}$$

We use $\mu = 6$ and scipy odeint to generate solutions with sequence length 10000 and the model is trained to generate dynamics in one dimension given the other.

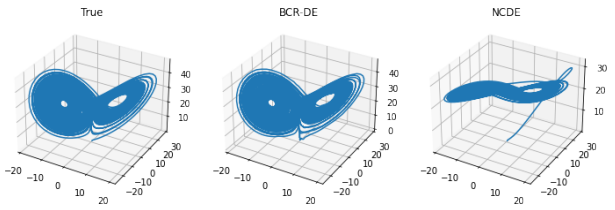

Figure 5. Comparison of trajectories for a chaotic Lorenz system. BCR-DE can be seen to match the original trajectory almost exactly.

(iv) Lorenz System: We consider the Lorenz system which is a nonlinear, aperiodic, three-dimensional system of ordinary differential equations and is known to exhibit chaotic behavior, and was originally developed to model atmospheric convection. In its simplest form Lorenz equations are:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \sigma(y - x); \quad \frac{\mathrm{d}y}{\mathrm{d}t} = x(\rho - z) - y; \quad \frac{\mathrm{d}z}{\mathrm{d}t} = xy - \beta z \tag{56}$$

We use $\sigma = 10, \rho = 28, \beta = 8/3$ for our experiments and use scipy odeint to generate solutions with sequence length 10000. The task is the model the chaotic trajectory in the third dimension given trajectories in the first two dimension.

(v) Hodgkin–Huxley model We consider the Hudgkin-Huxley model ((Hodgkin & Huxley, 1952)) which is a set of nonlinear differential equations used to describe electrical phenomenon of cells such as neurons and muscle cells. The system of ODEs is:

$$\begin{aligned}\frac{\mathrm{d}V_m}{\mathrm{d}t} &= \frac{I}{C_m} - \frac{\bar{g}_K n^4}{C_m}\left(V_m - V_K\right) \\ &\quad - \frac{\bar{g}_{Na} m^3 h}{C_m}\left(V_m - V_{Na}\right) - \frac{\bar{g}_l}{C_m}\left(V_m - V_l\right) \\ \frac{\mathrm{d}*}{\mathrm{d}t} &= \alpha_*\left(V_m\right)(1 - *) - \beta_*\left(V_m\right) *; \quad * \in \{n, m, h\}\end{aligned}$$

We used open source implementation from [1] to simulate our data. We model n, m, h which resemble the probability of opening of the ion channel being dependent on the type of the channel (either Sodium or Potassium in this case) based on the potential V_m and external stimulus I.

(vi) Temperature/Pressure/Humidity: Although this dataset is small in sequence length (so BCR-DE is not ideal), we want to assess performance. We focus on data where a natural coupling is expected, for example between temperature and relative humidity in the dataset of "BenzeneConcentration" from (Tan et al., 2020). In this case, there are 8 different variables for the time series out of which we work with two of them and the goal is to recover relative humidity based on observed temperature.

(vii) Character Trajectory: We consider the character trajectory dataset from (Bagnall et al., 2017). The data consists of 3 channels corresponding to the x,y coordinates and the pen tip force while drawing one of the 20 characters from English alphabet. We model the two dimensions via a coupled differential equation and seek to reconstruct one from the other.

In this case our model achieves MSE of 0.12624 in 0.052 hours of training time.

[1] https://github.com/swharden/pyHH