Neural Controlled Differential Equations for Irregular Time Series

Patrick Kidger   **James Morrill**   **James Foster**   **Terry Lyons**

Mathematical Institute, University of Oxford
The Alan Turing Institute, British Library
{kidger, morrill, foster, tlyons}@maths.ox.ac.uk

Abstract

Neural ordinary differential equations are an attractive option for modelling temporal dynamics. However, a fundamental issue is that the solution to an ordinary differential equation is determined by its initial condition, and there is no mechanism for adjusting the trajectory based on subsequent observations. Here, we demonstrate how this may be resolved through the well-understood mathematics of *controlled differential equations*. The resulting *neural controlled differential equation* model is directly applicable to the general setting of partially-observed irregularly-sampled multivariate time series, and (unlike previous work on this problem) it may utilise memory-efficient adjoint-based backpropagation even across observations. We demonstrate that our model achieves state-of-the-art performance against similar (ODE or RNN based) models in empirical studies on a range of datasets. Finally we provide theoretical results demonstrating universal approximation, and that our model subsumes alternative ODE models.

1   Introduction

Recurrent neural networks (RNN) are a popular choice of model for sequential data, such as a time series. The data itself is often assumed to be a sequence of observations from an underlying process, and the RNN may be interpreted as a discrete approximation to some function of this process. Indeed the connection between RNNs and dynamical systems is well-known [1, 2, 3, 4].

However this discretisation typically breaks down if the data is irregularly sampled or partially observed, and the issue is often papered over by binning or imputing data [5].

A more elegant approach is to appreciate that because the underlying process develops in continuous time, so should our models. For example [6, 7, 8, 9] incorporate exponential decay between observations, [10, 11] hybridise a Gaussian process with traditional neural network models, [12] approximate the underlying continuous-time process, and [13, 14] adapt recurrent neural networks by allowing some hidden state to evolve as an ODE. It is this last one that is of most interest to us here.

1.1   Neural ordinary differential equations

Neural ordinary differential equations (Neural ODEs) [3, 15], seek to approximate a map $x \mapsto y$ by learning a function f_θ and linear maps $\ell_\theta^1, \ell_\theta^2$ such that

$$y \approx \ell_\theta^1(z_T), \quad \text{where} \quad z_t = z_0 + \int_0^t f_\theta(z_s)\mathrm{d}s \quad \text{and} \quad z_0 = \ell_\theta^2(x). \tag{1}$$

Note that f_θ does not depend explicitly on s; if desired this can be included as an extra dimension in z_s [15, Appendix B.2].

Neural ODEs are an elegant concept. They provide an interface between machine learning and the other dominant modelling paradigm that is differential equations. Doing so allows for the well-understood tools of that field to be applied. Neural ODEs also interact beautifully with the manifold hypothesis, as they describe a flow along which to evolve the data manifold.

This description has not yet involved sequential data such as time series. The t dimension in equation (1) was introduced and then integrated over, and is just an internal detail of the model.

However the presence of this extra (artificial) dimension motivates the question of whether this model can be extended to sequential data such as time series. Given some ordered data $(x_0, \ldots, x_n)$, the goal is to extend the $z_0 = \ell_\theta^2(x)$ condition of equation (1) to a condition resembling "$z_0 = \ell(x_0), \ldots, z_n = \ell(x_n)$", to align the introduced t dimension with the natural ordering of the data.

The key difficulty is that equation (1) defines an ordinary differential equation; once θ has been learnt, then the solution of equation (1) is determined by the initial condition at z_0, and there is no direct mechanism for incorporating data that arrives later [4].

However, it turns out that the resolution of this issue – how to incorporate incoming information – is already a well-studied problem in mathematics, in the field of rough analysis, which is concerned with the study of *controlled differential equations*.[1] See for example [16, 17, 18, 19]. An excellent introduction is [20]. A comprehensive textbook is [21].

We will not assume familiarity with either controlled differential equations or rough analysis. The only concept we will rely on that may be unfamiliar is that of a Riemann–Stieltjes integral.

1.2 Contributions

We demonstrate how controlled differential equations may extend the Neural ODE model, which we refer to as the *neural controlled differential equation* (Neural CDE) model. Just as Neural ODEs are the continuous analogue of a ResNet, the Neural CDE is the continuous analogue of an RNN.

The Neural CDE model has three key features. One, it is capable of processing incoming data, which may be both irregularly sampled and partially observed. Two (and unlike previous work on this problem) the model may be trained with memory-efficient adjoint-based backpropagation even across observations. Three, it demonstrates state-of-the-art performance against similar (ODE or RNN based) models, which we show in empirical studies on the CharacterTrajectories, PhysioNet sepsis prediction, and Speech Commands datasets.

We provide additional theoretical results showing that our model is a universal approximator, and that it subsumes apparently-similar ODE models in which the vector field depends directly upon continuous data.

Our code is available at `https://github.com/patrick-kidger/NeuralCDE`. We have also released a library `torchcde`, at `https://github.com/patrick-kidger/torchcde`

2 Background

Let $\tau, T \in \mathbb{R}$ with $\tau < T$, and let $v, w \in \mathbb{N}$. Let $X \colon [\tau, T] \to \mathbb{R}^v$ be a continuous function of bounded variation; for example this is implied by X being Lipschitz. Let $\zeta \in \mathbb{R}^w$. Let $f \colon \mathbb{R}^w \to \mathbb{R}^{w \times v}$ be continuous.

Then we may define a continuous path $z \colon [\tau, T] \to \mathbb{R}^w$ by $z_\tau = \zeta$ and

$$z_t = z_\tau + \int_\tau^t f(z_s) \mathrm{d}X_s \quad \text{for } t \in (\tau, T], \tag{2}$$

where the integral is a Riemann–Stieltjes integral. As $f(z_s) \in \mathbb{R}^{w \times v}$ and $X_s \in \mathbb{R}^v$, the notation "$f(z_s)\mathrm{d}X_s$" refers to matrix-vector multiplication. The subscript notation refers to function evaluation, for example as is common in stochastic calculus.

Equation (2) exhibits global existence and uniqueness subject to global Lipschitz conditions on f; see [20, Theorem 1.3]. We say that equation (2) is a controlled differential equation (CDE) which is controlled or driven by X.

---

[1]Not to be confused with the similarly-named but separate field of control theory.
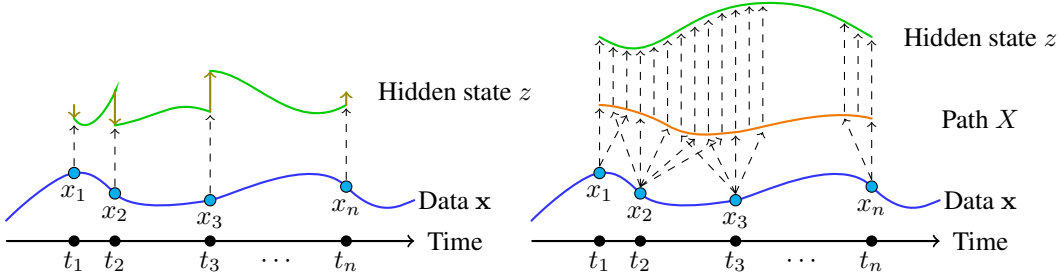
Figure 1: Some data process is observed at times $t_1, \ldots, t_n$ to give observations $x_1, \ldots, x_n$. It is otherwise unobserved. **Left:** Previous work has typically modified hidden state at each observation, and perhaps continuously evolved the hidden state between observations. **Right:** In contrast, the hidden state of the Neural CDE model has continuous dependence on the observed data.

3 Method

Suppose for simplicity that we have a fully-observed but potentially irregularly sampled time series $\mathbf{x} = ((t_0, x_0), (t_1, x_1), \ldots, (t_n, x_n))$, with each $t_i \in \mathbb{R}$ the timestamp of the observation $x_i \in \mathbb{R}^v$, and $t_0 < \cdots < t_n$. (We will consider partially-observed data later.)

Let $X \colon [t_0, t_n] \to \mathbb{R}^{v+1}$ be the natural cubic spline with knots at $t_0, \ldots, t_n$ such that $X_{t_i} = (x_i, t_i)$. As $\mathbf{x}$ is often assumed to be a discretisation of an underlying process, observed only through $\mathbf{x}$, then X is an approximation to this underlying process. Natural cubic splines have essentially the minimum regularity for handling certain edge cases; see Appendix A for the technical details.

Let $f_\theta \colon \mathbb{R}^w \to \mathbb{R}^{w \times (v+1)}$ be any neural network model depending on parameters θ. The value w is a hyperparameter describing the size of the hidden state. Let $\zeta_\theta \colon \mathbb{R}^{v+1} \to \mathbb{R}^w$ be any neural network model depending on parameters θ.

Then we define the *neural controlled differential equation* model as the solution of the CDE

$$z_t = z_{t_0} + \int_{t_0}^{t} f_\theta(z_s) \mathrm{d}X_s \quad \text{for } t \in (t_0, t_n], \tag{3}$$

where $z_{t_0} = \zeta_\theta(x_0, t_0)$. This initial condition is used to avoid translational invariance. Analogous to RNNs, the output of the model may either be taken to be the evolving process z, or the terminal value z_{t_n}, and the final prediction should typically be given by a linear map applied to this output.

The resemblance between equations (1) and (3) is clear. The essential difference is that equation (3) is driven by the data process X, whilst equation (1) is driven only by the identity function $\iota \colon \mathbb{R} \to \mathbb{R}$. In this way, the Neural CDE is naturally adapting to incoming data, as changes in X change the local dynamics of the system. See Figure 1.

3.1 Universal Approximation

It is a famous theorem in CDEs that in some sense they represent general functions on streams [22, Theorem 4.2], [23, Proposition A.6]. This may be applied to show that Neural CDEs are universal approximators, which we summarise in the following informal statement.

Theorem (Informal). *The action of a linear map on the terminal value of a Neural CDE is a universal approximator from* {sequences in $\mathbb{R}^v$} *to* $\mathbb{R}$.

Theorem B.14 in Appendix B gives a formal statement and a proof, which is somewhat technical. The essential idea is that CDEs may be used to approximate bases of functions on path space.

3.2 Evaluating the Neural CDE model

Evaluating the Neural CDE model is straightforward. In our formulation above, X is in fact not just of bounded variation but is differentiable. In this case, we may define

$$g_{\theta, X}(z, s) = f_\theta(z) \frac{\mathrm{d}X}{\mathrm{d}s}(s), \tag{4}$$

so that for $t \in (t_0, t_n]$,

$$z_t = z_{t_0} + \int_{t_0}^{t} f_\theta(z_s)\mathrm{d}X_s = z_{t_0} + \int_{t_0}^{t} f_\theta(z_s)\frac{\mathrm{d}X}{\mathrm{d}s}(s)\mathrm{d}s = z_{t_0} + \int_{t_0}^{t} g_{\theta,X}(z_s, s)\mathrm{d}s. \quad (5)$$

Thus it is possible to solve the Neural CDE using the same techniques as for Neural ODEs. In our experiments, we were able to straightforwardly use the already-existing `torchdiffeq` package [24] without modification.

3.3 Comparison to alternative ODE models

For the reader not familiar with CDEs, it might instead seem more natural to replace $g_{\theta,X}$ with some $h_\theta(z, X_s)$ that is directly applied to and potentially nonlinear in X_s. Indeed, such approaches have been suggested before, in particular to derive a "GRU-ODE" analogous to a GRU [14, 25].

However, it turns out that something is lost by doing so, which we summarise in the following statement.

Theorem (Informal). *Any equation of the form $z_t = z_0 + \int_{t_0}^{t} h_\theta(z_s, X_s)\mathrm{d}s$ may be represented exactly by a Neural CDE of the form $z_t = z_0 + \int_{t_0}^{t} f_\theta(z_s)\mathrm{d}X_s$. However the converse statement is not true.*

Theorem C.1 in Appendix C provides the formal statement and proof. The essential idea is that a Neural CDE can easily represent the identity function between paths, whilst the alternative cannot.

In our experiments, we find that the Neural CDE substantially outperforms the GRU-ODE, which we speculate is a consequence of this result.

3.4 Training via the adjoint method

An attractive part of Neural ODEs is the ability to train via adjoint backpropagation, see [15, 26, 27, 28], which uses only $\mathcal{O}(H)$ memory in the time horizon $L = t_n - t_0$ and the memory footprint H of the vector field. This is contrast to directly backpropagating through the operations of an ODE solver, which requires $\mathcal{O}(LH)$ memory.

Previous work on Neural ODEs for time series, for example [13], has interrupted the ODE to make updates at each observation. Adjoint-based backpropagation cannot be performed across the jump, so this once again requires $\mathcal{O}(LH)$ memory.

In contrast, the $g_{\theta,X}$ defined by equation (4) continuously incorporates incoming data, without interrupting the differential equation, and so adjoint backpropagation may be performed. This requires only $\mathcal{O}(H)$ memory. The underlying data unavoidably uses an additional $\mathcal{O}(L)$ memory. Thus training the Neural CDE has an overall memory footprint of just $\mathcal{O}(L + H)$.

We do remark that the adjoint method should be used with care, as some systems are not stable to evaluate in both the forward and backward directions [29, 30]. The problem of finite-time blow-up is at least not a concern, given global Lipschitz conditions on the vector field [20, Theorem 1.3]. Such a condition will be satisfied if f_θ uses ReLU or tanh nonlinearities, for example.

3.5 Intensity as a channel

It has been observed that the frequency of observations may carry information [6]. For example, doctors may take more frequent measurements of patients they believe to be at greater risk. Some previous work has for example incorporated this information by learning an intensity function [12, 13, 15].

We instead present a simple *non-learnt* procedure, that is compatible with Neural CDEs. Simply concatenate the index i of x_i together with x_i, and then construct a path X from the pair (i, x_i), as before. The channel of X corresponding to these indices then corresponds to the *cumulative* intensity of observations.

As the derivative of X is what is then used when evaluating the Neural CDE model, as in equation (5), then it is the intensity itself that then determines the vector field.

3.6 Partially observed data

One advantage of our formulation is that it naturally adapts to the case of partially observed data. Each channel may independently be interpolated between observations to define X in exactly the same manner as before.

In this case, the procedure for measuring observational intensity in Section 3.5 may be adjusted by instead having a separate observational intensity channel c_i for each original channel o_i, such that c_i increments every time an observation is made in o_i.

3.7 Batching

Given a batch of training samples with observation times drawn from the same interval $[t_0, t_n]$, we may interpolate each $\mathbf{x}$ to produce a continuous X, as already described. Each path X is what may then be batched together, regardless of whether the underlying data is irregularly sampled or partially observed. Batching is thus efficient for the Neural CDE model.

4 Experiments

We benchmark the Neural CDE against a variety of existing models.

These are: GRU-Δt, which is a GRU with the time difference between observations additionally used as an input; GRU-D [6], which modifies the GRU-Δt with learnt exponential decays between observations; GRU-ODE [14, 25], which is an ODE analogous to the operation of a GRU and uses X as its input; ODE-RNN [13], which is a GRU-Δt model which additionally applies a learnt Neural ODE to the hidden state between observations. Every model then used a learnt linear map from the final hidden state to the output, and was trained with cross entropy or binary cross entropy loss.

The GRU-Δt represents a straightforward baseline, the GRU-ODE is an alternative ODE model that is thematically similar to a Neural CDE, and the GRU-D and ODE-RNNs are state-of-the-art models for these types of problems. To avoid unreasonably extensive comparisons we have chosen to focus on demonstrating superiority within the class of ODE and RNN based models to which the Neural CDE belongs. These models were selected to collectively be representative of this class.

Each model is run five times, and we report the mean and standard deviation of the test metrics.

For every problem, the hyperparameters were chosen by performing a grid search to optimise the performance of the baseline ODE-RNN model. Equivalent hyperparameters were then used for every other model, adjusted slightly so that every model has a comparable number of parameters.

Precise experimental details may be found in Appendix D, regarding normalisation, architectures, activation functions, optimisation, hyperparameters, regularisation, and so on.

4.1 Varying amounts of missing data on CharacterTrajectories

We begin by demonstrating the efficacy of Neural CDEs on irregularly sampled time series.

To do this, we consider the CharacterTrajectories dataset from the UEA time series classification archive [31]. This is a dataset of 2858 time series, each of length 182, consisting of the x, y position and pen tip force whilst writing a Latin alphabet character in a single stroke. The goal is to classify which of 20 different characters are written.

We run three experiments, in which we drop either 30%, 50% or 70% of the data. The observations to drop are selected uniformly at random and independently for each time series. Observations are removed across channels, so that the resulting dataset is irregularly sampled but completely observed. The randomly removed data is the same for every model and every repeat.

The results are shown in Table 1. The Neural CDE outperforms every other model considered, and furthermore it does so whilst using an order of magnitude less memory. The GRU-ODE does consistently poorly despite being the most theoretically similar model to a Neural CDE. Furthermore we see that even as the fraction of dropped data increases, the performance of the Neural CDE remains roughly constant, whilst the other models all start to decrease.

Further experimental details may be found in Appendix D.2.

Table 1: Test accuracy (mean $\pm$ std, computed across five runs) and memory usage on CharacterTrajectories. Memory usage is independent of repeats and of amount of data dropped.

Model	Test Accuracy			Memory usage (MB)
	30% dropped	50% dropped	70% dropped	
GRU-ODE	92.6% $\pm$ 1.6%	86.7% $\pm$ 3.9%	89.9% $\pm$ 3.7%	1.5
GRU-Δt	93.6% $\pm$ 2.0%	91.3% $\pm$ 2.1%	90.4% $\pm$ 0.8%	15.8
GRU-D	94.2% $\pm$ 2.1%	90.2% $\pm$ 4.8%	91.9% $\pm$ 1.7%	17.0
ODE-RNN	95.4% $\pm$ 0.6%	96.0% $\pm$ 0.3%	95.3% $\pm$ 0.6%	14.8
Neural CDE (ours)	**98.7% $\pm$ 0.8%**	**98.8% $\pm$ 0.2%**	**98.6% $\pm$ 0.4%**	**1.3**

Table 2: Test AUC (mean $\pm$ std, computed across five runs) and memory usage on PhysioNet sepsis prediction. 'OI' refers to the inclusion of observational intensity, 'No OI' means without it. Memory usage is independent of repeats.

Model	Test AUC		Memory usage (MB)	
	OI	No OI	OI	No OI
GRU-ODE	0.852 $\pm$ 0.010	0.771 $\pm$ 0.024	454	273
GRU-Δt	0.878 $\pm$ 0.006	0.840 $\pm$ 0.007	837	826
GRU-D	0.871 $\pm$ 0.022	**0.850 $\pm$ 0.013**	889	878
ODE-RNN	0.874 $\pm$ 0.016	0.833 $\pm$ 0.020	696	686
Neural CDE (ours)	**0.880 $\pm$ 0.006**	0.776 $\pm$ 0.009	**244**	**122**

4.2 Observational intensity with PhysioNet sepsis prediction

Next we consider a dataset that is both irregularly sampled and partially observed, and investigate the benefits of observational intensity as discussed in Sections 3.5 and 3.6.

We use data from the PhysioNet 2019 challenge on sepsis prediction [32, 33]. This is a dataset of 40335 time series of variable length, describing the stay of patients within an ICU. Measurements are made of 5 static features such as age, and 34 time-dependent features such as respiration rate or creatinine concentration in the blood, down to an hourly resolution. Most values are missing; only 10.3% of values are observed. We consider the first 72 hours of a patient's stay, and consider the binary classification problem of predicting whether they develop sepsis over the course of their entire stay (which is as long as a month for some patients).

We run two experiments, one with observational intensity, and one without. For the Neural CDE and GRU-ODE models, observational intensity is continuous and on a per-channel basis as described in Section 3.6. For the ODE-RNN, GRU-D, and GRU-Δt models, observational intensity is given by appending an observed/not-observed mask to the input at each observation.[23] The initial hidden state of every model is taken to be a function (a small single hidden layer neural network) of the static features.

The results are shown in Table 2. As the dataset is highly imbalanced (5% positive rate), we report AUC rather than accuracy. When observational intensity is used, then the Neural CDE produces the best AUC overall, although the ODE-RNN and GRU-Δt models both perform well. The GRU-ODE continues to perform poorly.

Without observational intensity then every model performs substantially worse, and in particular we see that the benefit of including observational intensity is particularly dramatic for the Neural CDE.

As before, the Neural CDE remains the most memory-efficient model considered.

Further experimental details can be found in Appendix D.3.

---

[2] As our proposed observational intensity goes via a derivative, these each contain the same information.

[3] Note that the ODE-RNN, GRU-D and GRU-Δt models always receive the time difference between observations, Δt, as an input. Thus even in the no observational intensity case, they remain aware of the irregular sampling of the data, and so this case not completely fair to the Neural CDE and GRU-ODE models.

4.3 Regular time series with Speech Commands

Finally we demonstrate the efficacy of Neural CDE models on regularly spaced, fully observed time series, where we might hypothesise that the baseline models will do better.

We used the Speech Commands dataset [34]. This consists of one-second audio recordings of both background noise and spoken words such as 'left', 'right', and so on. We used 34975 time series corresponding to ten spoken words so as to produce a balanced classification problem. We preprocess the dataset by computing mel-frequency cepstrum coefficients so that each time series is then regularly spaced with length 161 and 20 channels.

Table 3: Test Accuracy (mean ± std, computed across five runs) and memory usage on Speech Commands. Memory usage is independent of repeats.

Model	Test Accuracy	Memory usage (GB)
GRU-ODE	47.9% ± 2.9%	**0.164**
GRU-Δt	43.3% ± 33.9%	1.54
GRU-D	32.4% ± 34.8%	1.64
ODE-RNN	65.9% ± 35.6%	1.40
Neural CDE (ours)	**89.8%** ± **2.5%**	0.167

The results are shown in Table 3. We observed that the Neural CDE had the highest performance, whilst using very little memory. The GRU-ODE consistently failed to perform. The other benchmark models surprised us by exhibiting a large variance on this problem, due to sometimes failing to train, and we were unable to resolve this by tweaking the optimiser. The best GRU-Δt, GRU-D and ODE-RNN models did match the performance of the Neural CDE, suggesting that on a regularly spaced problem all approaches can be made to work equally well.

In contrast, the Neural CDE model produced consistently good results every time. Anecdotally this aligns with what we observed over the course of all of our experiments, which is that the Neural CDE model usually trained quickly, and was robust to choice of optimisation hyperparameters. We stress that we did not perform a formal investigation of this phenomenen.

Further experimental details can be found in Appendix D.4.

5 Related work

In [13, 14] the authors consider interrupting a Neural ODE with updates from a recurrent cell at each observation, and were in fact the inspiration for this paper. Earlier work [6, 7, 8, 9] use intra-observation exponential decays, which are a special case. [35] consider something similar by interrupting a Neural ODE with stochastic events.

SDEs and CDEs are closely related, and several authors have introduced Neural SDEs. [36, 37, 38] treat them as generative models for time series and seek to model the data distribution. [39, 40] investigate using stochasticity as a regularizer, and demonstrate better performance by doing so. [41] use random vector fields so as to promote simpler trajectories, but do not use the 'SDE' terminology.

Adjoint backpropagation needs some work to apply to SDEs, and so [42, 43, 44] all propose methods for training Neural SDEs. We would particularly like to highlight the elegant approach of [44], who use the pathwise treatment given by rough analysis to approximate Brownian noise, and thus produce a random Neural ODE which may be trained in the usual way; such approaches may also avoid the poor convergence rates of SDE solvers as compared to ODE solvers.

Other elements of the theory of rough analysis and CDEs have also found machine learning applications. Amongst others, [23, 45, 46, 47, 48, 49, **?**, **?**] discuss applications of the signature transform to time series problems, and [50] investigate the related logsignature transform. [51] develop a kernel for time series using this methodology, and [52] apply this kernel to Gaussian processes. [53] develop software for these approaches tailored for machine learning.

There has been a range of work seeking to improve Neural ODEs. [54, 55] investigate speed-ups to the training proecedure, [56] develop an energy-based Neural ODE framework, and [29] demonstrate potential pitfalls with adjoint backpropagation. [30, 57] consider ways to vary the network parameters over time. [55, 58] consider how a Neural ODE model may be regularised (see also the stochastic regularisation discussed above). This provides a wide variety of techniques, and we are hopeful that some of them may additionally carry over to the Neural CDE case.

6 Discussion

6.1 Considerations

There are two key elements of the Neural CDE construction which are subtle, but important.

Time as a channel CDEs exhibit a *tree-like invariance* property [18]. What this means, roughly, is that a CDE is blind to speed at which X is traversed. Thus merely setting $X_{t_i} = x_i$ would not be enough, as time information is only incorporated via the parameterisation. This is why time is explicitly included as a channel via $X_{t_i} = (x_i, t_i)$.

Initial value networks The initial hidden state z_{t_0} should depend on $X_{t_0} = (x_0, t_0)$. Otherwise, the Neural CDE will depend upon X only through its derivative $\mathrm{d}X/\mathrm{d}t$, and so will be translationally invariant. An alternative would be to append another channel whose first derivative includes translation-sensitive information, for example by setting $X_{t_i} = (x_i, t_i, t_i x_0)$.

6.2 Performance tricks

We make certain (somewhat anecdotal) observations of tricks that seemed to help performance.

Final tanh nonlinearity We found it beneficial to use a tanh as a final nonlinearity for the vector field f_θ of a Neural CDE model. Doing so helps prevent extremely large initial losses, as the tanh constrains the rate of change of the hidden state. This is analogous to RNNs, where the key feature of GRUs and LSTMs are procedures to constrain the rate of change of the hidden state.

Layer-wise learning rates We found it beneficial to use a larger ($\times 10$–100) learning rate for the linear layer on top of the output of the Neural CDE, than for the vector field f_θ of the Neural CDE itself. This was inspired by the observation that the final linear layer has (in isolation) only a convex optimisation problem to solve.[4]

6.3 Limitations

Speed of computation We found that Neural CDEs were typically slightly faster to compute than the ODE-RNN model of [13]. (This is likely to be because in an Neural CDE, steps of the numerical solver can be made across observations, whilst the ODE-RNN must interrupt its solve at each observation.)

However, Neural CDEs were still roughly fives times slower than RNN models. We believe this is largely an implementation issue, as the implementation via `torchdiffeq` is in Python, and by default uses double-precision arithmetic with variable step size solvers, which we suspect is unnecessary for most practical tasks.

Number of parameters If the vector field $f_\theta \colon \mathbb{R}^w \to \mathbb{R}^{w \times (v+1)}$ is a feedforward neural network, with final hidden layer of size ω, then the number of scalars for the final affine transformation is of size $\mathcal{O}(\omega v w)$, which can easily be very large. In our experiments we have to choose small values of w and ω for the Neural CDE to ensure that the number of parameters is the same across models.

We did experiment with representing the final linear layer as an outer product of transformations $\mathbb{R}^w \to \mathbb{R}^w$ and $\mathbb{R}^w \to \mathbb{R}^{v+1}$. This implies that the resulting matrix is rank-one, and reduces the number of parameters to just $\mathcal{O}(\omega(v + w))$, but unfortunately we found that this hindered the classification performance of the model.

6.4 Future work

Vector field design The vector fields f_θ that we consider are feedforward networks. More sophisticated choices may allow for improved performance, in particular to overcome the trilinearity issue just discussed.

---

[4]In our experiments we applied this learning rate to the linear layer on top of every model, not the just the Neural CDE, to ensure a fair comparison.

Modelling uncertainty   As presented here, Neural CDEs do not give any measure of uncertainty about their predictions. Such extensions are likely to be possible, given the close links between CDEs and SDEs, and existing work on Neural SDEs [36, 37, 38, 39, 40, 42, 43, 44, **?**].

Numerical schemes   In this paper we integrated the Neural CDE by reducing it to an ODE. The field of numerical CDEs is relatively small – to the best of our knowledge [17, 59, 60, 61, 62, 63, 64, 65, 66] constitute essentially the entire field, and are largely restricted to rough controls. Other numerical methods may be able to exploit the CDE structure to improve performance.

Choice of X   Natural cubic splines were used to construct the path X from the time series $\mathbf{x}$. However, these are not causal. That is, X_t depends upon the value of x_i for $t < t_i$. This makes it infeasible to apply Neural CDEs in real-time settings, for which X_t is needed before x_i has been observed. Resolving this particular issue is a topic on which we have follow-up work planned.

Other problem types   Our experiments here involved only classification problems. There was no real reason for this choice, and we expect Neural CDEs to be applicable more broadly.

6.5   Related theories

Rough path theory   The field of rough path theory, which deals with the study of CDEs, is much larger than the small slice that we have used here. It is likely that further applications may serve to improve Neural CDEs. A particular focus of rough path theory is how to treat functions that must be sensitive to the order of events in a particular (continuous) way.

Control theory   Despite their similar names, and consideration of similar-looking problems, control theory and controlled differential equations are essentially separate fields. Control theory has clear links and applications that may prove beneficial to models of this type.

RNN theory   Neural CDEs may be interpreted as continuous-time versions of RNNs. CDEs thus offer a theoretical construction through which RNNs may perhaps be better understood. Conversely, what is known about RNNs may have applications to improve Neural CDEs.

7   Conclusion

We have introduced a new class of continuous-time time series models, Neural CDEs. Just as Neural ODEs are the continuous analogue of ResNets, the Neural CDE is the continuous time analogue of an RNN. The model has three key advantages: it operates directly on irregularly sampled and partially observed multivariate time series, it demonstrates state-of-the-art performance, and it benefits from memory-efficient adjoint-based backpropagation even across observations. To the best of our knowledge, no other model combines these three features together. We also provide additional theoretical results demonstrating universal approximation, and that Neural CDEs subsume alternative ODE models.

Broader Impact

We have introduced a new tool for studying irregular time series. As with any tool, it may be used in both positive and negative ways. The authors have a particular interest in electronic health records (an important example of irregularly sampled time-stamped data) and so here at least we hope and expect to see a positive impact from this work. We do not expect any specific negative impacts from this work.

Acknowledgments and Disclosure of Funding

References

[1] K.-i. Funahashi and Y. Nakamura, "Approximation of dynamical systems by continuous time recurrent neural networks," *Neural Networks*, vol. 6, no. 6, pp. 801 – 806, 1993.

[2] C. Bailer-Jones, D. MacKay, and P. Withers, "A recurrent neural network for modelling dynamical systems," *Network: Computation in Neural Systems*, vol. 9, pp. 531–47, 1998.

[3] W. E, "A Proposal on Machine Learning via Dynamical Systems," *Commun. Math. Stat.*, vol. 5, no. 1, pp. 1–11, 2017.

[4] M. Ciccone, M. Gallieri, J. Masci, C. Osendorfer, and F. Gomez, "NAIS-Net: Stable Deep Networks from Non-Autonomous Differential Equations," in *Advances in Neural Information Processing Systems 31*, pp. 3025–3035, Curran Associates, Inc., 2018.

[5] A. Gelman and J. Hill, *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge University Press, 2007.

[6] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu, "Recurrent Neural Networks for Multivariate Time Series with Missing Values," *Scientific Reports*, vol. 8, no. 1, p. 6085, 2018.

[7] W. Cao, D. Wang, J. Li, H. Zhou, L. Li, and Y. Li, "BRITS: Bidirectional Recurrent Imputation for Time Series," in *Advances in Neural Information Processing Systems 31* (S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), pp. 6775–6785, Curran Associates, Inc., 2018.

[8] H. Mei and J. M. Eisner, "The Neural Hawkes Process: A Neurally Self-Modulating Multivariate Point Process," in *Advances in Neural Information Processing Systems 30* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), pp. 6754–6764, Curran Associates, Inc., 2017.

[9] M. Mozer, D. Kazakov, and R. Lindsey, "Discrete Event, Continuous Time RNNs," *arXiv:1710.04110*, 2017.

[10] S. C.-X. Li and B. M. Marlin, "A scalable end-to-end Gaussian process adapter for irregularly sampled time series classification," in *Advances in Neural Information Processing Systems*, pp. 1804–1812, 2016.

[11] J. Futoma, S. Hariharan, and K. Heller, "Learning to Detect Sepsis with a Multitask Gaussian Process RNN Classifier," in *Proceedings of the 34th International Conference on Machine Learning*, pp. 1174–1182, 2017.

[12] S. N. Shukla and B. Marlin, "Interpolation-Prediction Networks for Irregularly Sampled Time Series," in *International Conference on Learning Representations*, 2019.

[13] Y. Rubanova, T. Q. Chen, and D. K. Duvenaud, "Latent Ordinary Differential Equations for Irregularly-Sampled Time Series," in *Advances in Neural Information Processing Systems 32*, pp. 5320–5330, Curran Associates, Inc., 2019.

[14] E. De Brouwer, J. Simm, A. Arany, and Y. Moreau, "GRU-ODE-Bayes: Continuous Modeling of Sporadically-Observed Time Series," in *Advances in Neural Information Processing Systems 32*, pp. 7379–7390, Curran Associates, Inc., 2019.

[15] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural Ordinary Differential Equations," in *Advances in Neural Information Processing Systems 31*, pp. 6571–6583, Curran Associates, Inc., 2018.

[16] T. J. Lyons, "Differential equations driven by rough signals," *Revista Matemática Iberoamericana*, vol. 14, no. 2, pp. 215–310, 1998.

[17] T. Lyons, "Rough paths, signatures and the modelling of functions on streams," *arXiv:1405.4537*, 2014.

[18] B. M. Hambly and T. J. Lyons, "Uniqueness for the signature of a path of bounded variation and the reduced path group," *Annals of Mathematics*, vol. 171, no. 1, pp. 109–167, 2010.

[19] I. Chevyrev and H. Oberhauser, "Signature moments to characterize laws of stochastic processes," *arXiv:1810.10971*, 2018.

[20] T. Lyons, M. Caruana, and T. Levy, *Differential equations driven by rough paths*. Springer, 2004. École d'Été de Probabilités de Saint-Flour XXXIV - 2004.

[21] P. K. Friz and N. B. Victoir, "Multidimensional stochastic processes as rough paths: theory and applications," *Cambridge University Press*, 2010.

[22] I. Perez Arribas, "Derivatives pricing using signature payoffs," *arXiv:1809.09466*, 2018.

[23] P. Bonnier, P. Kidger, I. Perez Arribas, C. Salvi, and T. Lyons, "Deep Signature Transforms," in *Advances in Neural Information Processing Systems*, pp. 3099–3109, 2019.

[24] R. T. Q. Chen, "torchdiffeq," 2018. `https://github.com/rtqichen/torchdiffeq`.

[25] I. Jordan, P. A. Sokol, and I. M. Park, "Gated recurrent units viewed through the lens of continuous time dynamical systems," *arXiv:1906.01005*, 2019.

[26] L. S. Pontryagin, E. F. Mishchenko, V. G. Boltyanskii, and R. V. Gamkrelidze, "The mathematical theory of optimal processes," 1962.

[27] M. B. Giles and N. A. Pierce, "An Introduction to the Adjoint Approach to Design," *Flow, Turbulence and Combustion*, vol. 65, pp. 393–415, Dec 2000.

[28] W. W. Hager, "Runge-Kutta methods in optimal control and the transformed adjoint system," *Numerische Mathematik*, vol. 87, pp. 247–282, Dec 2000.

[29] A. Gholami, K. Keutzer, and G. Biros, "ANODE: Unconditionally Accurate Memory-Efficient Gradients for Neural ODEs," *arXiv:1902.10298*, 2019.

[30] T. Zhang, Z. Yao, A. Gholami, J. E. Gonzalez, K. Keutzer, M. W. Mahoney, and G. Biros, "ANODEV2: A Coupled Neural ODE Framework," in *Advances in Neural Information Processing Systems 32*, pp. 5151–5161, Curran Associates, Inc., 2019.

[31] A. Bagnall, H. A. Dau, J. Lines, M. Flynn, J. Large, A. Bostrom, P. Southam, and E. Keogh, "The uea multivariate time series classification archive," *arXiv:1811.00075*, 2018.

[32] M. Reyna, C. Josef, R. Jeter, S. Shashikumar, B. Moody, M. B. Westover, A. Sharma, S. Nemati, and G. Clifford, "Early Prediction of Sepsis from Clinical Data: The PhysioNet/Computing in Cardiology Challenge," *Critical Care Medicine*, vol. 48, no. 2, pp. 210–217, 2019.

[33] Goldberger, A. L. and Amaral L. A. N. and Glass, L. and Hausdorff, J. M. and Ivanov P. Ch. and Mark, R. G. and Mietus, J. E. and Moody, G. B. and Peng, C.-K. and Stanley, H. E., "PhysioBank, PhysioToolkit and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals," *Circulation*, vol. 23, pp. 215–220, 2003.

[34] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *arXiv:1804.03209*, 2020.

[35] J. Jia and A. R. Benson, "Neural Jump Stochastic Differential Equations," in *Advances in Neural Information Processing Systems 32*, pp. 9847–9858, Curran Associates, Inc., 2019.

[36] C. Cuchiero, W. Khosrawi, and J. Tiechmann, "A generative adversarial network approach to calibration of local stochastic volatility models," *arXiv:2005.02505*, 2020.

[37] B. Tzen and M. Raginsky, "Theoretical guarantees for sampling and inference in generative models with latent diffusions," *COLT*, 2019.

[38] R. Deng, B. Chang, M. Brubaker, G. Mori, and A. Lehrmann, "Modeling Continuous Stochastic Processes with Dynamic Normalizing Flows," *arXiv:2002.10516*, 2020.

[39] X. Liu, T. Xiao, S. Si, Q. Cao, S. Kumar, and C.-J. Hsieh, "Neural SDE: Stabilizing Neural ODE Networks with Stochastic Noise," *arXiv:1906.02355*, 2019.

[40] V. Oganesyan, A. Volokhova, and D. Vetrov, "Stochasticity in Neural ODEs: An Empirical Study," *arXiv:2002.09779*, 2020.

[41] N. Twomey, M. Kozłowski, and R. Santos-Rodríguez, "Neural ODEs with stochastic vector field mixtures," *arXiv:1905.09905*, 2019.

[42] X. Li, T.-K. L. Wong, R. T. Q. Chen, and D. K. Duvenaud, "Scalable Gradients and Variational Inference for Stochastic Differential Equations," *AISTATS*, 2020.

[43] B. Tzen and M. Raginsky, "Neural Stochastic Differential Equations: Deep Latent Gaussian Models in the Diffusion Limit," *arXiv:1905.09883*, 2019.

[44] L. Hodgkinson, C. van der Heide, F. Roosta, and M. Mahoney, "Stochastic Normalizing Flows," *arXiv:2002.09547*, 2020.

[45] I. Chevyrev and A. Kormilitzin, "A primer on the signature method in machine learning," *arXiv:1603.03788*, 2016.

[46] I. Perez Arribas, G. M. Goodwin, J. R. Geddes, T. Lyons, and K. E. A. Saunders, "A signature-based machine learning model for distinguishing bipolar disorder and borderline personality disorder," *Translational Psychiatry*, vol. 8, no. 1, p. 274, 2018.

[47] A. Fermanian, "Embedding and learning with signatures," *arXiv:1911.13211*, 2019.

[48] J. Morrill, A. Kormilitzin, A. Nevado-Holgado, S. Swaminathan, S. Howison, and T. Lyons, "The Signature-based Model for Early Detection of Sepsis from Electronic Health Records in the Intensive Care Unit," *International Conference in Computing in Cardiology*, 2019.

[49] J. Reizenstein, *Iterated-integral signatures in machine learning*. PhD thesis, University of Warwick, 2019. http://wrap.warwick.ac.uk/131162/.

[50] S. Liao, T. Lyons, W. Yang, and H. Ni, "Learning stochastic differential equations using RNN with log signature features," *arXiv:1908.08286*, 2019.

[51] F. J. Király and H. Oberhauser, "Kernels for sequentially ordered data," *Journal of Machine Learning Research*, 2019.

[52] C. Toth and H. Oberhauser, "Variational Gaussian Processes with Signature Covariances," *arXiv:1906.08215*, 2019.

[53] P. Kidger and T. Lyons, "Signatory: differentiable computations of the signature and logsignature transforms, on both CPU and GPU," *arXiv:2001.00706*, 2020.

[54] A. Quaglino, M. Gallieri, J. Masci, and J. Koutník, "Snode: Spectral discretization of neural odes for system identification," in *International Conference on Learning Representations*, 2020.

[55] C. Finlay, J.-H. Jacobsen, L. Nurbekyan, and A. Oberman, "How to train your neural ODE," *arXiv:2002.02798*, 2020.

[56] S. Massaroli, M. Poli, M. Bin, J. Park, A. Yamashita, and H. Asama, "Stable Neural flows," *arXiv:2003.08063*, 2020.

[57] S. Massaroli, M. Poli, J. Park, A. Yamashita, and H. Asama, "Dissecting Neural ODEs," *arXiv:2002.08071*, 2020.

[58] H. Yan, J. Du, V. Y. F. Tan, and J. Feng, "On Robustness of Neural Ordinary Differential Equations," *arXiv:1910.05513*, 2019.

[59] F. Castell and J. Gaines, "The ordinary differential equation approach to asymptotically efficient schemes for solution of stochastic differential equations," *Annales de l'Institut Henri Poincaré. Probabilités et Statistiques*, vol. 32, 1996.

[60] S. Malham and A. Wiese, "Stochastic Lie Group Integrators," *SIAM J. Sci. Comput.*, vol. 30, no. 2, pp. 597–617, 2007.

[61] L. G. Gyurkó, *Numerical approximations for stochastic differential equations*. PhD thesis, University of Oxford, 2008.

[62] A. Janssen, *Order book models, signatures and numerical approximations of rough differential equations*. PhD thesis, University of Oxford, 2011.

[63] Y. Boutaib, L. G. Gyurkó, T. Lyons, and D. Yang, "Dimension-free Euler estimates of rough differential equations," *Rev. Roumaine Math. Pures Appl.*, 2014.

[64] H. Boedihardjo, T. Lyons, and D. Yang, "Uniform factorial decay estimates for controlled differential equations," *Electronic Communications in Probability*, vol. 20, no. 94, 2015.

[65] J. Foster, *Numerical approximations for stochastic differential equations*. PhD thesis, University of Oxford, 2020.

[66] J. Foster, T. Lyons, and H. Oberhauser, "An optimal polynomial approximation of Brownian motion," *SIAM J. Numer. Anal.*, vol. 58, no. 3, pp. 1393–1421, 2020.

[67] J. M. Varah, "A Lower Bound for the Smallest Singular Value of a Matrix," *Linear Algebra and its Applications*, vol. 11, no. 1, pp. 3–5, 1975.

[68] A. Pinkus, "Approximation theory of the MLP model in neural networks," *Acta Numer.*, vol. 8, pp. 143–195, 1999.

[69] P. Kidger and T. Lyons, "Universal Approximation with Deep Narrow Networks," *arXiv:1905.08539*, 2019.

[70] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *ICLR*, 2015.

[71] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems 32*, pp. 8024–8035, Curran Associates, Inc., 2019.

Supplementary material

Appendix A discusses the technical considerations of schemes for constructing a path X from data. Appendix B proves universal approximation of the Neural CDE model, and is substantially more technical than the rest of this paper. Appendix C proves that the Neural CDE model subsumes alternative ODE models which depend directly and nonlinearly on the data. Appendix D gives the full details of every experiment, such as choice of optimiser, hyperparameter searches, and so on.

A Other schemes for constructing the path X

To evaluate the model as discussed in Section 3.2, X must be at least continuous and piecewise differentiable.

A.1 Differentiating with respect to the time points

However, there is a technical caveat in the specific case that derivatives with respect to the initial time t_0 are required, and that training is done with the adjoint method. In this case the derivative with respect to t_0 is computed using, and thus requires, derivatives of the vector field with respect to t.

To be precise, suppose we have a Neural CDE as before:

$$z_t = z_{t_0} + \int_{t_0}^{t} f_\theta(z_s)\mathrm{d}X_s \quad \text{for } t \in (t_0, t_n].$$

Let L be some (for simplicity scalar-valued) function of z_{t_n}, for example a loss. Consider

$$g_{\theta,X}(z, s) = f_\theta(z)\frac{\mathrm{d}X}{\mathrm{d}s}(s)$$

as before, and let

$$a_s = \frac{\mathrm{d}L}{\mathrm{d}z_s},$$

which is vector-valued, with size equal to the size of z_s, the number of hidden channels.

Then, applying [15, Equation 52] to our case:

$$\begin{aligned}
\frac{\mathrm{d}L}{\mathrm{d}t_0} &= \frac{\mathrm{d}L}{\mathrm{d}t_n} - \int_{t_n}^{t_0} a_s \cdot \frac{\partial g_{\theta,X}}{\partial s}(z_s, s)\mathrm{d}s \\
&= \frac{\mathrm{d}L}{\mathrm{d}t_n} - \int_{t_n}^{t_0} a_s \cdot f_\theta(z_s)\frac{\mathrm{d}^2 X}{\mathrm{d}s^2}(s)\mathrm{d}s,
\end{aligned} \tag{6}$$

where $\cdot$ represents the dot product.

In principle we may make sense of equation (6) when $\mathrm{d}^2 X/\mathrm{d}s^2$ is merely measure valued, but in practice most code is only set up to handle classical derivatives. If derivatives with respect to t_0 are desired, then practically speaking X must be at least twice differentiable.

A.2 Adaptive step size solvers

There is one further caveat that must be considered. Suppose X is twice differentiable, but that the second derivative is discontinuous. For example this would be accomplished by taking X to be a quadratic spline interpolation.

If seeking to solve equation (6) with an adaptive step-size solver, we found that the solver would take a long time to compute the backward pass, as it would have to slow down to resolve each jump in $\mathrm{d}^2 X/\mathrm{d}s^2$, and then speed back up in the intervals in-between.

A.3 Natural cubic splines

This is then the reason for our selection of natural cubic splines: by ensuring that X is twice continuously differentiable, the above issue is ameliorated, and adaptive step size solvers operate acceptably. Cubic splines give essentially the minimum regularity for the techniques discussed in this paper to work 'out of the box' in all cases.

Other than this smoothness, however, there is little that is special about natural cubic splines. Other possible options are for example Gaussian processes [10, 11] or kernel methods [12]. Furthermore, especially in the case of noisy data it need not be an interpolation scheme – approximation and curve-fitting schemes are valid too.

B Universal Approximation

The behaviour of controlled differential equations are typically described through the *signature transform* (also known as *path signature* or simply *signature*) [20] of the driving process X. We demonstrate here how a (Neural) CDE may be reduced to consideration of just the signature transform, in order to prove universal approximation.

The proof is essentially split into two parts. The first part is to prove universal approximation with respect to continuous paths, as is typically done for CDEs. The second (lengthier) part is to interpret what this means for the natural cubic splines that we use in this paper, so that we can get universal approximation with respect to the original data as well.

Definition B.1. Let $\tau, T \in \mathbb{R}$ with $\tau < T$ and let $v \in \mathbb{N}$. Let $\mathcal{V}^1([\tau, T]; \mathbb{R}^v)$ represent the space of continuous functions of bounded variation. Equip this space with the norm

$$\widehat{X} \mapsto \left\| \widehat{X} \right\|_{\mathcal{V}} = \left\| \widehat{X} \right\|_\infty + \left| \widehat{X} \right|_{BV}.$$

This is a somewhat unusual norm to use, as bounded variation seminorms are more closely aligned with L^1 norms than L^∞ norms.

Definition B.2. For any $\widehat{X} \in \mathcal{V}^1([\tau, T]; \mathbb{R}^v)$ let $X_t = (\widehat{X}_t, t) \in \mathcal{V}^1([\tau, T]; \mathbb{R}^{v+1})$. We choose to use the notation of 'removing the hat' to denote this time augmentation, for consistency with the main text which uses X for the time-augmented path.

Definition B.3. For any $N, v \in \mathbb{N}$, let $\kappa(N, v) = \sum_{i=0}^{N} (v+1)^i$.

Definition B.4 (Signature transform). For any $k \in \mathbb{N}$ and any $y \in \mathbb{R}^k$, let $M(y) \in \mathbb{R}^{k(v+1) \times (v+1)}$ be the matrix

$$M(y) = \begin{bmatrix} y^1 & 0 & 0 & \cdots & 0 \\ y^2 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ y^k & 0 & 0 & \cdots & 0 \\ 0 & y^1 & 0 & \cdots & 0 \\ 0 & \vdots & 0 & & \vdots \\ 0 & y^k & 0 & \cdots & 0 \\ & & & \ddots & \\ 0 & 0 & 0 & \cdots & y^1 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & y^k \end{bmatrix}$$

Fix $N \in \mathbb{N}$ and $X \in \mathcal{V}^1([\tau, T]; \mathbb{R}^{v+1})$. Let $y^{0,X,N} : [\tau, T] \to \mathbb{R}$ be constant with $y_t^{0,X,N} = 1$.

For all $i \in \{1, \ldots, N\}$, iteratively let $y^{i,X,N} : [\tau, T] \to \mathbb{R}^{(v+1)^i}$ be the value of the integral

$$y_t^{i,X,N} = y_\tau^{i,X,N} + \int_\tau^t M(y_s^{i-1,X,N}) \mathrm{d}X_s \quad \text{for } t \in (\tau, T],$$

with $y_\tau^{i,X,N} = 0 \in \mathbb{R}^{(v+1)^i}$.

Then we may stack these together:

$$y^{X,N} = (y^{0,X,N}, \ldots, y^{N,X,N}) \colon [\tau, T] \to \mathbb{R}^{\kappa(N,v)},$$

$$\widetilde{M}(y^{X,N}) = (0, M \circ y^{0,X,N}, \ldots, M \circ y^{N-1,X,N}) \colon [\tau, T] \to \mathbb{R}^{\kappa(N,v) \times v}.$$

Then $y^{X,N}$ is the unique solution to the CDE

$$y_t^{X,N} = y_\tau^{X,N} + \int_\tau^t \widetilde{M}(y^{X,N})_s \mathrm{d}X_s \quad \text{for } t \in (\tau, T],$$

with $y_\tau^{X,N} = (1, 0, \ldots, 0)$.

Then the *signature transform truncated to depth N* is defined as the map

$$\mathrm{Sig}^N \colon \mathcal{V}^1([\tau, T]; \mathbb{R}^{v+1}) \to \mathbb{R}^{\kappa(N,v)},$$

$$\mathrm{Sig}^N \colon X \mapsto y_T^{X,N}.$$

If this seems like a strange definition, then note that for any $a \in \mathbb{R}^k$ and any $b \in \mathbb{R}^v$ that $M(a)b$ is equal to a flattened vector corresponding to the outer product $a \otimes b$. As such, the signature CDE is instead typically written much more concisely as the exponential differential equation

$$y_t^{X,N} = y_\tau^{X,N} + \int_\tau^t y_s^{X,N} \otimes \mathrm{d}X_s \quad \text{for } t \in (\tau, T],$$

however we provide the above presentation for consistency with the rest of the text, which does not introduce $\otimes$.

Definition B.5. Let $\mathcal{V}_0^1([\tau, T]; \mathbb{R}^v) = \left\{ X \in \mathcal{V}^1([\tau, T]; \mathbb{R}^v) \,\middle|\, X_0 = 0 \right\}.$

With these definitions out of the way, we are ready to state the famous universal nonlinearity property of the signature transform. We think [22, Theorem 4.2] gives the most straightforward proof of this result. This essentially states that the signature gives a basis for the space of functions on compact path space.

Theorem B.6 (Universal nonlinearity). *Let $\tau, T \in \mathbb{R}$ with $\tau < T$ and let $v, u \in \mathbb{N}$.*

Let $K \subseteq \mathcal{V}_0^1([\tau, T]; \mathbb{R}^v)$ be compact. (Note the subscript zero.)

Let $\mathrm{Sig}^N \colon \mathcal{V}^1([\tau, T]; \mathbb{R}^{v+1}) \to \mathbb{R}^{\kappa(N,v)}$ denote the signature transform truncated to depth N.

Let

$$J^{N,u} = \left\{ \ell \colon \mathbb{R}^{\kappa(N,v)} \to \mathbb{R}^u \,\middle|\, \ell \text{ is linear} \right\}.$$

Then

$$\bigcup_{N \in \mathbb{N}} \left\{ \widehat{X} \mapsto \ell(\mathrm{Sig}^N(X)) \,\middle|\, \ell \in J^{N,u} \right\}$$

is dense in $C(K; \mathbb{R}^u)$.

With the universal nonlinearity property, we can now prove universal approximation of CDEs with respect to controlling paths X.

Theorem B.7 (Universal approximation with CDEs). *Let $\tau, T \in \mathbb{R}$ with $\tau < T$ and let $v, u \in \mathbb{N}$. For any $w \in \mathbb{N}$ let*

$$F^w = \left\{ f \colon \mathbb{R}^w \to \mathbb{R}^{w \times (v+1)} \,\middle|\, f \text{ is continuous} \right\},$$

$$L^{w,u} = \left\{ \ell \colon \mathbb{R}^w \to \mathbb{R}^u \,\middle|\, \ell \text{ is linear} \right\},$$

$$\xi^w = \left\{ \zeta \colon \mathbb{R}^{v+1} \to \mathbb{R}^w \,\middle|\, \zeta \text{ is continuous} \right\}.$$

For any $w \in \mathbb{N}$, any $f \in F^w$ and any $\zeta \in \xi^w$ and any $\widehat{X} \in \mathcal{V}^1([\tau, T]; \mathbb{R}^v)$, let $z^{f,\zeta,X} \colon [\tau, T] \to \mathbb{R}^w$ be the unique solution to the CDE

$$z_t^{f,\zeta,X} = z_\tau^{f,\zeta,X} + \int_\tau^t f(z_s^{f,\zeta,X}) \mathrm{d}X_s \quad \text{for } t \in (\tau, T],$$

with $z_\tau^{f,\zeta,X} = \zeta(X_\tau)$.

Let $K \subseteq \mathcal{V}^1([\tau, T]; \mathbb{R}^v)$ be compact.

Then

$$\bigcup_{w \in \mathbb{N}} \left\{ \widehat{X} \mapsto \ell(z_T^{f,\zeta,X}) \,\middle|\, f \in F^w, \ell \in L^{w,u}, \zeta \in \xi^w \right\}$$

is dense in $C(K; \mathbb{R}^u)$.

Proof. We begin by prepending a straight line segment to every element of K. For every $\widehat{X} \in K$, define $\widehat{X}^* \colon [\tau - 1, T] \to \mathbb{R}^v$ by

$$\widehat{X}_t^* = \begin{cases} (t - \tau + 1)\widehat{X}_\tau & t \in [\tau - 1, \tau), \\ \widehat{X}_t & t \in [\tau, T]. \end{cases}$$

Similarly define X^*, so that a hat means that time is *not* a channel, whilst a star means that an extra straight-line segment has been prepended to the path.

Now let $K^* = \left\{ \widehat{X}^* \,\middle|\, \widehat{X} \in K \right\}$. Then $K^* \subseteq \mathcal{V}_0^1([\tau - 1, T]; \mathbb{R}^v)$ and is also compact. Therefore by Theorem B.6,

$$\bigcup_{N \in \mathbb{N}} \left\{ \widehat{X}^* \mapsto \ell(\operatorname{Sig}^N(X^*)) \,\middle|\, \ell \in J^{N,u} \right\}$$

is dense in $C(K^*; \mathbb{R}^u)$.

So let $\alpha \in C(K; \mathbb{R}^u)$ and $\varepsilon > 0$. The map $\widehat{X} \mapsto \widehat{X}^*$ is a homeomorphism, so we may find $\beta \in C(K^*; \mathbb{R}^u)$ such that $\beta(\widehat{X}^*) = \alpha(\widehat{X})$ for all $\widehat{X} \in K$. Next, there exists some $N \in \mathbb{N}$ and $\ell \in J^{N,u}$ such that γ defined by $\gamma \colon \widehat{X}^* \mapsto \ell(\operatorname{Sig}^N(X^*))$ is ε-close to β.

By Definition B.4 there exists $f \in F^{\kappa(N,v)}$ so that $\operatorname{Sig}^N(X^*) = y_T^{X^*}$ for all $X^* \in K^*$, where y^{X^*} is the unique solution of the CDE

$$y_t^{X^*} = y_{\tau-1}^{X^*} + \int_{\tau-1}^t f(y_s^{X^*}) \mathrm{d}X_s^* \quad \text{for } t \in (\tau - 1, T],$$

with $y_{\tau-1}^{X^*} = (1, 0, \ldots, 0)$.

Now let $\zeta \in \xi$ be defined by $\zeta(X_\tau) = y_\tau^{X^*}$, which we note is well defined because the value of $y_t^{X^*}$ only depends on X_τ for $t \in [\tau - 1, \tau]$.

Now for any $X \in K$ let $z^X \colon [\tau, T] \to \mathbb{R}^w$ be the unique solution to the CDE

$$z_t^X = z_\tau^X + \int_\tau^t f(z_s^X) \mathrm{d}X_s \quad \text{for } t \in (\tau, T],$$

with $z_\tau^X = \zeta(X_\tau)$.

Then by uniqueness of solution, $z_t^X = y_t^{X^*}$ for $t \in [\tau, T]$, and so in particular $\operatorname{Sig}^N(X^*) = y_T^{X^*} = z_T^X$.

Finally it remains to note that $\ell \in J^{N,u} = L^{\kappa(N,v),u}$.

So let δ be defined by $\delta \colon \widehat{X} \mapsto \ell(z_T^X)$. Then δ is in the set which we were aiming to show density of (with $w = \kappa(N, v)$, $f \in F^w$, $\ell \in L^{w,u}$ and $\zeta \in \xi$ as chosen above), whilst for all $\widehat{X} \in K$,

$$\delta(\widehat{X}) = \ell(z_T^X) = \ell(\operatorname{Sig}^N(X^*)) = \gamma(\widehat{X}^*)$$

is ε-close to $\beta(\widehat{X}^*) = \alpha(\widehat{X})$. Thus density has been established. $\qquad\square$

Lemma B.8. *Let $K \subseteq C^2([\tau, T], \mathbb{R}^v)$ be uniformly bounded with uniformly bounded first and second derivatives. That is, there exists some $C > 0$ such that $\left\|\widehat{X}\right\|_\infty + \left\|\mathrm{d}\widehat{X}/\mathrm{d}t\right\|_\infty + \left\|\mathrm{d}^2\widehat{X}/\mathrm{d}t^2\right\|_\infty < C$ for all $\widehat{X} \in K$. Then $K \subseteq \mathcal{V}^1([\tau, T]; \mathbb{R}^v)$ and is relatively compact (that is, its closure is compact) with respect to $\|\cdot\|_\mathcal{V}$.*

Proof. K is bounded in $C^2([\tau, T], \mathbb{R}^v)$ so it is relatively compact in $C^1([\tau, T], \mathbb{R}^v)$.

Furthermore for any $\widehat{X} \in K$,

$$
\begin{aligned}
\left\|\widehat{X}\right\|_{\mathcal{V}} &= \left\|\widehat{X}\right\|_\infty + \left|\widehat{X}\right|_{BV} \\
&= \left\|\widehat{X}\right\|_\infty + \left\|\frac{\mathrm{d}\widehat{X}}{\mathrm{d}t}\right\|_1 \\
&\le \left\|\widehat{X}\right\|_\infty + \left\|\frac{\mathrm{d}\widehat{X}}{\mathrm{d}t}\right\|_\infty,
\end{aligned}
$$

and so the embedding $C^1([\tau, T], \mathbb{R}^v) \to \mathcal{V}^1([\tau, T]; \mathbb{R}^v)$ is continuous. Therefore K is also relatively compact in $\mathcal{V}^1([\tau, T]; \mathbb{R}^v)$. $\qquad\square$

Next we need to understand how a natural cubic spline is controlled by the size of its data. We establish the following crude bounds.

Lemma B.9. *Let $v \in \mathbb{N}$. Let $x_0, \ldots, x_n \in \mathbb{R}^v$. Let $t_0, \ldots, t_n \in \mathbb{R}$ be such that $t_0 < t_1 < \cdots < t_n$. Let $\widehat{X} \colon [t_0, t_n] \to \mathbb{R}^v$ be the natural cubic spline such that $\widehat{X}(t_i) = x_i$. Let $\tau_i = t_{i+1} - t_i$ for all i. Then there exists an absolute constant $C > 0$ such that*

$$
\left\|\widehat{X}\right\|_\infty + \left\|\mathrm{d}\widehat{X}/\mathrm{d}t\right\|_\infty + \left\|\mathrm{d}^2\widehat{X}/\mathrm{d}t^2\right\|_\infty < C \left\|\tau\right\|_\infty \left\|x\right\|_\infty (\min_i \tau_i)^{-2}(\left\|\tau\right\|_\infty + (\min_i \tau_i)^{-1}).
$$

Proof. Surprisingly, we could not find a reference for a fact of this type, but it follows essentially straightforwardly from the derivation of natural cubic splines.

Without loss of generality assume $v = 1$, as we are using the infinity norm over the dimensions v, and each cubic interpolation is performed separately for each dimension.

Let the i-th piece of $\widehat{X}$, which is a cubic on the interval $[t_i, t_{i+1}]$, be denoted Y_i. Without loss of generality, translate each Y_i to the origin so as to simplify the algebra, so that $Y_i \colon [0, \tau_i] \to \mathbb{R}$. Let $Y_i(t) = a_i + b_i t + c_i t^2 + d_i t^3$ for some coefficients a_i, b_i, c_i, d_i and $i \in \{0, \ldots, n-1\}$.

Letting $D_i = Y_i'(0)$ for $i \in \{0, \ldots, n-1\}$ and $D_n = Y_{n-1}(\tau_{n-1})$, the displacement and derivative conditions imposed at each knot are $Y_i(0) = x_i$, $Y_i(\tau_i) = x_{i+1}$, $Y_i'(0) = D_i$ and $Y_i'(\tau_i) = D_{i+1}$. This then implies that $a_i = x_i$, $b_i = D_i$,

$$
c_i = 3\tau_i^{-2}(x_{i+1} - x_i) - \tau_i^{-1}(D_{i+1} + 2D_i), \tag{7}
$$

$$
d_i = 2\tau_i^{-3}(x_i - xi + 1) + \tau_i^{-2}(D_{i+1} + D_i). \tag{8}
$$

Letting $\lesssim$ denote 'less than or equal up to some absolute constant', then these equations imply that

$$
\left\|\widehat{X}\right\|_\infty = \max_i \|Y_i\|_\infty \lesssim \max_i(|x_i| + \tau_i |D_i|) \le \|x\|_\infty + \|\tau\|_\infty \|D\|_\infty, \tag{9}
$$

$$
\left\|\frac{\mathrm{d}\widehat{X}}{\mathrm{d}t}\right\|_\infty = \max_i \|Y_i\|_\infty \lesssim \max_i(\tau_i^{-1} |x_i| + |D_i|) \le \|x\|_\infty (\min_i \tau_i)^{-1} + \|D\|_\infty, \tag{10}
$$

$$
\left\|\frac{\mathrm{d}^2\widehat{X}}{\mathrm{d}t^2}\right\|_\infty = \max_i \|Y_i\|_\infty \lesssim \max_i(\tau_i^{-2} |x_i| + \tau_i^{-1} |D_i|) \le \|x\|_\infty (\min_i \tau_i)^{-2} + \|D\|_\infty (\min_i \tau_i)^{-1}. \tag{11}
$$

Next, the second derivative condition at each knot is $Y_{i-1}''(\tau_{i-1}) = Y_i''(0)$ for $i \in \{1, \ldots, n-1\}$, and the natural condition is $Y_0''(0) = 0$ and $Y_{n-1}''(\tau_{n-1}) = 0$. With equations (7), (8) this gives

$$
\mathcal{T}D = k,
$$

where

$$\mathcal{T} = \begin{bmatrix} 2\tau_0^{-1} & \tau_0^{-1} \\ \tau_0^{-1} & 2(\tau_0^{-1} + \tau_1^{-1}) \\ & \tau_1^{-1} & 2(\tau_1^{-1} + \tau_2^{-1}) & \tau_2^{-1} \\ & & \ddots & \ddots & \ddots \\ & & & \tau_{n-2}^{-1} & 2(\tau_{n-2}^{-1} + \tau_{n-1}^{-1}) & \tau_{n-1}^{-1} \\ & & & & \tau_{n-1}^{-1} & 2\tau_{n-1}^{-1} \end{bmatrix},$$

$$D = \begin{bmatrix} D_0 \\ \vdots \\ D_n \end{bmatrix},$$

$$k = \begin{bmatrix} 3\tau_0^{-2}(x_1 - x_0) \\ 3\tau_1^{-2}(x_2 - x_1) + 3\tau_0^{-2}(x_1 - x_0) \\ \vdots \\ 3\tau_{n-1}^{-2}(x_n - x_{n-1}) + 3\tau_{n-2}^{-2}(x_{n-1} - x_{n-2}) \\ 3\tau_{n-1}^{-2}(x_n - x_{n-1}). \end{bmatrix}$$

Let $\left\| \mathcal{T}^{-1} \right\|_\infty$ denote the operator norm and $\|D\|_\infty$, $\|k\|_\infty$ denote the elementwise norm. Now $\mathcal{T}$ is diagonally dominant, so the Varah bound [67] and HM-AM inequality gives

$$\left\| \mathcal{T}^{-1} \right\|_\infty \leq (\min_i(\tau_i^{-1} + \tau_{i+1}^{-1}))^{-1} \lesssim \|\tau\|_\infty .$$

Thus

$$\|D\|_\infty \lesssim \|\tau\|_\infty \|k\|_\infty \lesssim \|\tau\|_\infty \|x\|_\infty (\min_i \tau_i)^{-2}.$$

Together with equations (9)–(11) this gives the result. $\qquad\square$

Definition B.10 (Space of time series)**.** Let $v \in \mathbb{N}$. and $\tau, T \in \mathbb{R}$ such that $\tau < T$. We define the space of time series in $[\tau, T]$ over $\mathbb{R}^v$ as

$$\mathcal{TS}_{[\tau,T]}(\mathbb{R}^v) = \{((t_0, x_0), \ldots, (t_n, x_n)) \mid n \in \mathbb{N}, t_i \in [\tau, T], x_n \in \mathbb{R}^v, t_0 = \tau, t_n = T, n \geq 2\} .$$

To our knowledge, there is no standard topology on time series. One option is to treat them as sequences, however it is not clear how best to treat sequences of different lengths, or how to incorporate timestamp information. Given that a time series is typically some collection of observations from some underlying process, we believe the natural approach is to treat them as subspaces of functions.

Definition B.11 (General topologies on time series)**.** Let $v \in \mathbb{N}$. and $\tau, T \in \mathbb{R}$ such that $\tau < T$. Let F denote some topological space of functions. Let $\iota\colon \mathcal{TS}_{[\tau,T]}(\mathbb{R}^v) \to F$ be some map. Then we may define a topology on $\mathcal{TS}_{[\tau,T]}(\mathbb{R}^v)$ as the weakest topology with respect to which ι is continuous.

Recall that we use subscripts to denote function evaulation.

Definition B.12 (Natural cubic spline topology)**.** Let $v \in \mathbb{N}$. and $\tau, T \in \mathbb{R}$ such that $\tau < T$. Let $F = C([\tau, T]; \mathbb{R}^v)$ equipped with the uniform norm. For all $\mathbf{x} = ((t_0, x_0), \ldots, (t_n, x_n)) \in \mathcal{TS}_{[\tau,T]}(\mathbb{R}^v)$, let $\widehat{\iota}\colon \mathcal{TS}_{[\tau,T]}(\mathbb{R}^v) \to F$ produce the natural cubic spline such that $\widehat{\iota}(\mathbf{x})_{t_i} = x_i$ with knots at $t_0, \ldots, t_n$. Then this defines a topology on $\mathcal{TS}_{[\tau,T]}(\mathbb{R}^v)$ as in the previous definition.

Remark B.13. In fact this defines a seminorm on $\mathcal{TS}_{[\tau,T]}(\mathbb{R}^v)$, by $\|\mathbf{x}\| = \|\widehat{\iota}(\mathbf{x})\|_\infty$. This is only a seminorm as for example $((0,0), (2,2))$ and $((0,0), (1,1), (2,2))$ have the same natural cubic spline interpolation. This can be worked around so as to instead produce a full norm, but it is a deliberate choice not to: we would often prefer that these time series be thought of as equal. (And if it they are not equal, then first augmenting with observational intensity as in the main paper should distinguish them.)

Theorem B.14 (Universal approximation with Neural CDEs via natural cubic splines). *Let $\tau, T \in \mathbb{R}$ with $\tau < T$ and let $v, u \in \mathbb{N}$. For any $w \in \mathbb{N}$ let*

$$F^w_{\mathcal{NN}} = \left\{ f \colon \mathbb{R}^w \to \mathbb{R}^{w \times (v+1)} \;\middle|\; f \text{ is a feedforward neural network} \right\},$$

$$L^{w,u} = \left\{ \ell \colon \mathbb{R}^w \to \mathbb{R}^u \mid \ell \text{ is linear} \right\},$$

$$\xi^w_{\mathcal{NN}} = \left\{ \zeta \colon \mathbb{R}^{v+1} \to \mathbb{R}^w \;\middle|\; \zeta \text{ is a feedforward neural network} \right\}.$$

Let $\widehat{\iota}$ denote the natural cubic spline interpolation as in the previous definition, and recall that 'removing the hat' is our notation for augmenting with time. For any $w \in \mathbb{N}$, any $f \in F^w$ and any $\zeta \in \xi^w_{\mathcal{NN}}$ and any $\mathbf{x} \in \mathcal{TS}_{[\tau,T]}(\mathbb{R}^v)$, let $z^{f,\zeta,\mathbf{x}} \colon [\tau, T] \to \mathbb{R}^w$ be the unique solution to the CDE

$$z_t^{f,\zeta,\mathbf{x}} = z_\tau^{f,\zeta,\mathbf{x}} + \int_\tau^t f(z_s^{f,\zeta,\mathbf{x}}) \mathrm{d}\iota(\mathbf{x})_s \quad \text{for } t \in (\tau, T],$$

with $z_\tau^{f,\zeta,\mathbf{x}} = \zeta(\iota(\mathbf{x})_\tau)$.

Let $K \subseteq \mathcal{TS}_{[\tau,T]}(\mathbb{R}^v)$ be such that there exists $C > 0$ such that

$$\|x\|_\infty \left(\min_i (t_{i+1} - t_i) \right)^{-3} < C \tag{12}$$

for every $\mathbf{x} = ((t_0, x_0), \ldots, (t_n, x_n)) \in K$. (With C independent of $\mathbf{x}$.)

Then

$$\bigcup_{w \in \mathbb{N}} \left\{ \mathbf{x} \mapsto \ell(z_T^{f,\zeta,\mathbf{x}}) \;\middle|\; f \in F^w_{\mathcal{NN}}, \ell \in L^{w,u}, \zeta \in \xi^w_{\mathcal{NN}} \right\}$$

is dense in $C(K; \mathbb{R}^u)$ with respect to the natural cubic spline topology on $\mathcal{TS}_{[\tau,T]}(\mathbb{R}^v)$.

Proof. Fix $\mathbf{x} = ((t_0, x_0), \ldots, (t_n, x_n)) \in K$. Let $\widehat{X} = \widehat{\iota}(\mathbf{x})$. Now $\|\tau\|_\infty \leq T - \tau$ is bounded so by Lemma B.9 and the assumption of equation (12), there exists a constant $C_1 > 0$ independent of $\mathbf{x}$ such that

$$\left\| \widehat{X} \right\|_\infty + \left\| \frac{\mathrm{d}\widehat{X}}{\mathrm{d}t} \right\|_\infty + \left\| \frac{\mathrm{d}^2 \widehat{X}}{\mathrm{d}t^2} \right\|_\infty < C_1.$$

Thus by Lemma B.8, $\widehat{\iota}(K)$ is relatively compact in $\mathcal{V}^1([\tau, T]; \mathbb{R}^v)$.

Let $K_1 = \overline{\widehat{\iota}(K)}$, where the overline denotes a closure. Now by Theorem B.7, and defining F^w and ξ^w as in the statement of that theorem,

$$\bigcup_{w \in \mathbb{N}} \left\{ \widehat{\iota}(\mathbf{x}) \mapsto \ell(z_T^{f,\zeta,\mathbf{x}}) \;\middle|\; f \in F^w, \ell \in L^{w,u}, \zeta \in \xi^w \right\}$$

is dense in $C(K_1, \mathbb{R}^u)$.

For any $f \in F^w$, any $\zeta \in \xi^w$, any $f_{\mathcal{NN}} \in F^w_{\mathcal{NN}}$ and any $\zeta_{\mathcal{NN}} \in \xi^w_{\mathcal{NN}}$, the terminal values $z_T^{f,\zeta,\mathbf{x}}$ and $z_T^{f_{\mathcal{NN}},\zeta_{\mathcal{NN}},\mathbf{x}}$ may be compared by standard estimates, for example as commonly used in the proof of Picard's theorem. Classical universal approximation results for neural networks [68, 69] then yield that

$$\bigcup_{w \in \mathbb{N}} \left\{ \widehat{\iota}(\mathbf{x}) \mapsto \ell(z_T^{f,\zeta,\mathbf{x}}) \;\middle|\; f \in F^w_{\mathcal{NN}}, \ell \in L^{w,u}, \zeta \in \xi^w_{\mathcal{NN}} \right\}$$

is dense in $C(K_1, \mathbb{R}^u)$.

By the definition of the natural cubic spline topology on $\mathcal{TS}_{[\tau,T]}(\mathbb{R}^v)$, then

$$\bigcup_{w \in \mathbb{N}} \left\{ \mathbf{x} \mapsto \ell(z_T^{f,\zeta,\mathbf{x}}) \;\middle|\; f \in F^w_{\mathcal{NN}}, \ell \in L^{w,u}, \zeta \in \xi^w_{\mathcal{NN}} \right\}$$

is dense in $C(K, \mathbb{R}^u)$. $\qquad\square$

C Comparison to alternative ODE models

Suppose if instead of equation (4), we replace $g_{\theta,X}(z,s)$ by $h_\theta(z, X_s)$ for some other vector field h_θ. This might seem more natural. Instead of having $g_{\theta,X}$ be linear in $\mathrm{d}X/\mathrm{d}s$, we take a h_θ that is potentially nonlinear in the control X_s.

Have we gained anything by doing so? It turns out no, and in fact we have lost something. The Neural CDE setup directly subsumes anything depending directly on X.

Theorem C.1. *Let $\tau, T \in \mathbb{R}$ with $\tau < T$, let $v, w \in \mathbb{N}$ with $v + 1 < w$. Let*

$$
\begin{aligned}
F &= \left\{ f \colon \mathbb{R}^w \to \mathbb{R}^{w \times (v+1)} \,\middle|\, f \text{ is continuous} \right\}, \\
H &= \left\{ h \colon \mathbb{R}^{w-v-1} \times \mathbb{R}^{v+1} \to \mathbb{R}^{w-v-1} \,\middle|\, h \text{ is continuous} \right\}, \\
\xi &= \left\{ \zeta \colon \mathbb{R}^{v+1} \to \mathbb{R}^w \,\middle|\, \zeta \text{ is continuous} \right\}, \\
\mathbb{X} &= \left\{ \widehat{X} \colon [\tau, T] \to \mathbb{R}^v \,\middle|\, \widehat{X} \text{ continuous and of bounded variation} \right\}.
\end{aligned}
$$

For any $\widehat{X} \in \mathbb{X}$, let $X_t = (\widehat{X}_t, t)$. Let $\pi \colon \mathbb{R}^w \to \mathbb{R}^{w-v-1}$ be the orthogonal projection onto the first $w - v - 1$ coordinates.

For any $f \in F$, any $\zeta \in \xi$, and any $\widehat{X} \in \mathbb{X}$, let $z^{f,\zeta,X} \colon [\tau, T] \to \mathbb{R}^w$ be the unique solution to

$$
z_t^{f,\zeta,X} = z_\tau^{f,\zeta,X} + \int_\tau^t f(z_s^{f,\zeta,X}) \mathrm{d}X_s \quad \text{for } t \in (\tau, T],
$$

with $z_\tau^{f,\zeta,X} = \zeta(X_\tau)$.

Similarly for any $h \in H$, any $\zeta \in \xi$, and any $\widehat{X} \in \mathbb{X}$, let $y^{f,X} \colon [\tau, T] \to \mathbb{R}^{w-v-1}$ be the unique solution to

$$
y_t^{h,\zeta,X} = y_\tau^{h,\zeta,X} + \int_\tau^t h(y_s^{h,\zeta,X}, X_s) \mathrm{d}s \quad \text{for } t \in (\tau, T],
$$

with $y_\tau^{h,\zeta,X} = \pi(\zeta(X_\tau))$.

Let $\mathcal{Y} = \left\{ \widehat{X} \mapsto y^{h,\zeta,X} \,\middle|\, h \in H, \zeta \in \xi \right\}$ and $\mathcal{Z} = \left\{ \widehat{X} \mapsto \pi \circ z^{f,\zeta,X} \,\middle|\, f \in F, \zeta \in \xi \right\}$.

Then $\mathcal{Y} \subsetneq \mathcal{Z}$.

In the above statement, then a practical choice of $f \in F$ or $h \in H$ will typically correspond to some trained neural network.

Note the inclusion of time via the augmentation $\widehat{X} \mapsto X$. Without it, then the reparameterisation invariance property of CDEs [18], [23, Proposition A.7] will restrict the possible functions that CDEs can represent. This hypothesis is not necessary for the $\mathcal{Y} \neq \mathcal{Z}$ part of the conclusion.

Note also how the CDE uses a larger state space of w, compared to $w - v - 1$ for the alternative ODE. The reason for this is that whilst f has no explicit nonlinear dependence on X, we may construct it to have such a dependence implicitly, by recording X into $v + 1$ of its w hidden channels, whereupon X is hidden state and may be treated nonlinearly. This hypothesis is also not necessary to demonstrate the $\mathcal{Y} \neq \mathcal{Z}$ part of the conclusion.

This theorem is essentially an algebraic statement, and is thus not making any analytic claims, for example on universal approximation.

Proof.

That $\mathcal{Y} \neq \mathcal{Z}$:   Let $z^{f,\zeta,\cdot} \in \mathcal{Z}$ for $\zeta \in \xi$ arbitrary and $f \in F$ constant and such that

$$f(z) = \left.\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}\right\} w$$

$$\underbrace{\qquad\qquad\qquad}_{v+1}$$

Then for any $\widehat{X} \in \mathbb{X}$, the corresponding CDE solution in $\mathcal{Z}$ is

$$z_t^{f,\varsigma,X} = z_\tau^{f,\varsigma,X} + \int_\tau^t f(z_s^{f,\varsigma,X})\mathrm{d}X_s,$$

and so the first component of its solution is

$$z_t^{f,\varsigma,X,1} = X_t^1 - X_\tau^1 + \varsigma^1(X_\tau),$$

whilst the other components are constant

$$z_t^{f,\varsigma,X,i} = \varsigma^i(X_\tau)$$

for $i \in \{2, \ldots, w\}$, where superscripts refer to components throughout.

Now suppose for contradiction that there exists $y^{h,\varsigma,\cdot} \in \mathcal{Y}$ for some $\Xi \in \xi$ and $h \in H$ such that $y^{h,\Xi,X} = \pi \circ z^{f,\varsigma,X}$ for all $\widehat{X} \in \mathbb{X}$. Now $y^{h,\Xi,X}$ must satisfy

$$y_t^{h,\Xi,X} = y_\tau^{h,\Xi,X} + \int_\tau^t h(y_s^{h,\Xi,X}, X_s)\mathrm{d}s,$$

and so

$$(X_t^1 - X_\tau^1 + \varsigma^1(X_\tau), 0, \ldots, 0) = \pi(\Xi(X_\tau)) + \int_\tau^t h((X_s^1 - X_\tau^1 + \varsigma^1(X_\tau), \varsigma^2(X_\tau), \ldots, \varsigma^w(X_\tau)), X_s)\mathrm{d}s.$$

Consider those X which are differentiable. Differentiating with respect to t now gives

$$\frac{\mathrm{d}X^1}{\mathrm{d}t}(t) = h^1((X_s^1 - X_\tau^1 + \varsigma^1(X_\tau), \varsigma^2(X_\tau), \ldots, \varsigma^w(X_\tau)), X_t). \tag{13}$$

That is, h^1 satisfies equation (13) for all differentiable X. This is clearly impossible: the right hand side is a function of t, X_t and X_τ only, which is insufficient to determine $\mathrm{d}X^1/\mathrm{d}t(t)$.

That $\mathcal{Y} \subseteq \mathcal{Z}$: Let $y^{h,\Xi,X} \in \mathcal{Y}$ for some $\Xi \in \xi$ and $h \in H$. Let $\sigma\colon \mathbb{R}^w \to \mathbb{R}^{v+1}$ be the orthogonal projection onto the last $v+1$ coordinates. Let $\varsigma \in \xi$ be such that $\pi \circ \varsigma = \pi \circ \Xi$ and $\sigma(\varsigma(X_\tau)) = X_\tau$. Then let $f \in F$ be defined by

$$f(z) = \left[\begin{array}{cccc|c} 0 & 0 & \cdots & 0 & h^1(\pi(z), \sigma(z)) \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & h^{w-v-1}(\pi(z), \sigma(z)) \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 \end{array}\right] \begin{array}{l} \left.\vphantom{\begin{array}{c}0\\\vdots\\0\end{array}}\right\} w-v-1 \\[2em] \left.\vphantom{\begin{array}{c}0\\0\\\vdots\\0\\1\end{array}}\right\} v+1 \end{array}$$

$$\underbrace{\qquad\qquad}_{v} \underbrace{\qquad}_{1}$$

21

Then for $t \in (\tau, T]$,

$$z_t^{f,\zeta,X} = \zeta(X_\tau) + \int_\tau^t f(z_s^{f,\zeta,X}) \mathrm{d}X_s$$

$$= \zeta(X_\tau) + \int_\tau^t \begin{bmatrix} 0 & 0 & \cdots & 0 & h^1(\pi(z_s^{f,\zeta,X}), \sigma(z_s^{f,\zeta,X})) \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & h^{w-v-1}(\pi(z_s^{f,\zeta,X}), \sigma(z_s^{f,\zeta,X})) \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathrm{d}\widehat{X}_s^1 \\ \vdots \\ \mathrm{d}\widehat{X}_s^v \\ \mathrm{d}s \end{bmatrix}$$

$$= \zeta(X_\tau) + \int_\tau^t \begin{bmatrix} h^1(\pi(z_s^{f,\zeta,X}), \sigma(z_s^{f,\zeta,X}))\mathrm{d}s \\ \vdots \\ h^{w-v-1}(\pi(z_s^{f,\zeta,X}), \sigma(z_s^{f,\zeta,X}))\mathrm{d}s \\ \mathrm{d}\widehat{X}_s^1 \\ \vdots \\ \mathrm{d}\widehat{X}_s^v \\ \mathrm{d}s \end{bmatrix}$$

$$= \zeta(X_\tau) + \int_\tau^t \begin{bmatrix} h(\pi(z_s^{f,\zeta,X}), \sigma(z_s^{f,\zeta,X}))\mathrm{d}s \\ \mathrm{d}X_s \end{bmatrix}.$$

Thus in particular

$$\sigma(z_t^{f,\zeta,X}) = \sigma(\zeta(X_\tau)) + \int_\tau^t \mathrm{d}X_s = \sigma(\zeta(X_\tau)) - X_\tau + X_t = X_t.$$

Thus

$$\pi(z_t^{f,\zeta,X}) = \pi(\zeta(X_\tau)) + \int_\tau^t h(\pi(z_s^{f,\zeta,X}), \sigma(z_s^{f,\zeta,X}))\mathrm{d}s = \pi(\Xi(X_\tau)) + \int_\tau^t h(\pi(z_s^{f,\zeta,X}), X_s)\mathrm{d}s.$$

Thus we see that $\pi(z^{f,\zeta,X})$ satisfies the same differential equation as $y^{h,\Xi,X}$. So by uniqueness of solution [20, Theorem 1.3], $y^{h,\Xi,X} = \pi(z^{f,\zeta,X}) \in \mathcal{Z}$. □

D Experimental details

D.1 General notes

Code Code to reproduce every experiment can be found at https://github.com/patrick-kidger/NeuralCDE.

Normalisation Every dataset was normalised so that each channel has mean zero and variance one.

Loss Every binary classification problem used binary cross-entropy loss applied to the sigmoid of the output of the model. Every multiclass classification problem used cross-entropy loss applied to the softmax of the output of the model.

Architectures For both the Neural CDE and ODE-RNN, the integrand f_θ was taken to be a feedforward neural network. A final linear layer was always used to map from the terminal hidden state to the output.

Activation functions For the Neural CDE model we used ReLU activation functions. Following the recommendations of [13], we used tanh activation functions for the ODE-RNN model, who remark that for the ODE-RNN model, tanh activations seem to make the model easier for the ODE

solver to resolve. Interestingly we did not observe this behaviour when trying tanh activations and `method='dopri5'` with the Neural CDE model, hence our choice of ReLU.

Optimiser   Every problem used the Adam [70] optimiser as implemented by PyTorch 1.3.1 [71]. Learning rate and batch size varied between experiments, see below. The learning rate was reduced if metrics failed to improve for a certain number of epochs, and training was terminated if metrics failed to improve for a certain (larger) number of epochs. The details of this varied by experiment, see the individual sections. Once training was finished, then the parameters were rolled back to the parameters which produced the best validation accuracy throughout the whole training procedure. The learning rate for the final linear layer (mapping from the hidden state of a model to the output) was typically taken to be much larger than the learning rate used elsewhere in the model; this is a standard trick that we found improved performance for all models.

Hyperparameter selection   In brief, hyperparameters were selected to optimise the ODE-RNN baseline, and equivalent hyperparameters used for the other models.

In more detail:

We began by selecting the learning rate. This was selected by starting at 0.001 and reducing it until good performance was achieved for a small ODE-RNN model with batch size 32.

After this, we increased the batch size until the selected model trained at what was in our judgement a reasonable speed. As is standard practice, we increased the learning rate proportionate to the increase in batch size.

Subsequently we selected model hyperparameters (number of hidden channels, width and depth of the vector field network) via a grid search to optimise the ODE-RNN baseline. A single run of each hyperparameter choice was performed. The equivalent hyperparameters were then used on the GRU-Δt, GRU-D, GRU-ODE baselines, and also our Neural CDE models, after being adjusted to produce roughly the same number of parameters for each model.

The grids searched over and the resulting hyperparameters are stated in the individual sections below.

Weight regularisation L^2 weight regularisation was applied to every parameter of the ODE-RNN, GRU-Δt and GRU-D models, and to every parameter of the vector fields for the Neural CDE and GRU-ODE models.

ODE Solvers   The ODE components of the ODE-RNN, GRU-ODE, and Neural CDE models were all computed using the fourth-order Runge-Kutta with 3/8 rule solver, as implemented by passing `method='rk4'` to the `odeint_adjoint` function of the `torchdiffeq` [24] package. The step size was taken to equal the minimum time difference between any two adjacent observations.

Adjoint backpropagation   The GRU-ODE, Neural CDE and the ODE component of the ODE-RNN are all trained via the adjoint backpropagation method [15], as implemented by `odeint_adjoint` function of the `torchdiffeq` package.

Computing infrastructure   All experiments were run on one of two computers; both used Ubuntu 18.04.4 LTS, were running PyTorch 1.3.1, and used version 0.0.1 of the `torchdiffeq` [24] package. One computer was equipped with a Xeon E5-2960 v4, two GeForce RTX 2080 Ti, and two Quadro GP100, whilst the other was equipped with a Xeon Silver 4104 and three GeForce RTX 2080 Ti.

D.2   CharacterTrajectories

The learning rate used was 0.001 and the batch size used was 32. If the validation loss stagnated for 10 epochs then the learning rate was divided by 10 and training resumed. If the training loss or training accuracy stagnated for 50 epochs then training was terminated. The maximum number of epochs allowed was 1000.

We combined the train/test split of the original dataset (which are of unusual proportion, being 50%/50%), and then took a 70%/15%/15% train/validation/test split.

The initial condition ζ_θ of the Neural CDE model was taken to be a learnt linear map from the first observation to the hidden state vector. (Recall that is an important part of the model, to avoid translation invariance.)

The hyperparameters were optimised (for just the ODE-RNN baseline as previously described) by performing most of a grid search over 16 or 32 hidden channels, 32, 48, 64, 128 hidden layer size, and 1, 2, 3 hidden layers. (The latter two hyperparameters corresponding to the vector fields of the ODE-RNN and Neural CDE models.) A few option combinations were not tested due to the poor performance of similar combinations. (For example every combination with hidden layer size of 128 demonstrated relatively poor performance.) The search was done on just the 30% missing data case, and the same hyperparameters were used for the 50% and 70% missing data cases.

The hyperparameters selected were 32 hidden channels for the Neural CDE and ODE-RNN models, and 47 hidden channels for the GRU-Δt, GRU-D and GRU-ODE models. The Neural CDE and ODE-RNN models both used a feedforward network for their vector fields, with 3 hidden layers each of width 32. The resulting parameter counts for each model were 8212 for the Neural CDE, 8436 for the ODE-RNN, 8386 for the GRU-D, 8292 for the GRU-Δt, and 8372 for the GRU-ODE.

D.3   PhysioNet sepsis prediction

The batch size used was 1024 and learning rate used was 0.0032, arrived at as previously described. If the training loss stagnated for 10 epochs then the learning rate was divided by 10 and training resumed. If the training loss or validation accuracy stagnated for 100 epochs then training was terminated. The maximum number of epochs allowed was 200. The learning rate for the final linear layer (a component of every model, mapping from the final hidden state to the output) used a learning rate that 100 times larger, so 0.32.

The original dataset does not come with an existing split, so we took our own 70%/15%/15% train/validation/test split.

As this problem featured static (not time-varying) features, we incorporated this information by allowing the initial condition of every model to depend on these. This was taken to be a single hidden layer feedforward network with ReLU activation functions and of width 256, which we did not attempt a hyperparameter search over.

As this dataset is partially observed, then for the ODE-RNN, GRU-Δt, GRU-D models, which require *something* to be passed at each time step, even if a value is missing, then we fill in missing values with natural cubic splines, for ease of comparison with the Neural CDE and ODE-RNN models. (We do not describe this as imputation as for the observational intensity case the observational mask is additionally passed to these models.) In particular this differs slightly from the usual implementation of GRU-D, which usually use a weighted average of the last observation and the mean. Splines accomplishes much the same thing, and help keep things consistent between the various models.

The hyperparameters were optimised (for just the ODE-RNN baseline as previously described) by performing most of a grid search over 64, 128, 256 hidden channels, 64, 128, 256 hidden layer size, and 1, 2, 3, 4 hidden layers. (The latter two hyperparameters corresponding to the vector fields of the ODE-RNN and Neural CDE models.)

The hyperparameters selected for the ODE-RNN model were 128 hidden channels, and a vector field given by a feedforward neural network with hidden layer size 128 and 4 hidden layers. In order to keep the number of parameters the same between each model, this was reduced to 49 hidden channels and hidden layer size 49 for the Neural CDE model, and increased to 187 hidden channels for the GRU-Δt, GRU-D and GRU-ODE models. When using observational intensity, the resulting parameter counts were 193541 for the Neural CDE, 194049 for the ODE-RNN, 195407 for the GRU-D, 195033 for the GRU-Δt, and 194541 for the GRU-ODE. When not using observational intensity, the resulting parameter counts were 109729 for the Neural CDE, 180097 for the ODE-RNN, 175260 for the GRU-D, 174886 for the GRU-Δt, and 174921 for the GRU-ODE. Note the dramatically reduced parameter count for the Neural CDE; this is because removing observational intensity reduces the number of channels, which affects the parameter count dramatically as discussed in Section 6.3.

D.4   Speech Commands

The batch size used was 1024 and the learning rate used was 0.0016, arrived at as previously described. If the training loss stagnated for 10 epochs then the learning rate was divided by 10 and

training resumed. If the training loss or validation accuracy stagnated for 100 epochs then training was terminated. The maximum number of epochs allowed was 200. The learning rate for the final linear layer (a component of every model, mapping from the final hidden state to the output) used a learning rate that 100 times larger, so 0.16.

Each time series from the dataset is univariate and of length 16000. We computed 20 Mel-frequency cepstral coefficients of the input as implemented by `torchaudio.transforms.MFCC`, with logarithmic scaling applied to the coefficients. The window for the short-time Fourier transform component was a Hann window of length 200, with hop length of 100, with 200 frequency bins. This was passed through 128 mel filterbanks and 20 mel coefficients extracted. This produced a time series of length 161 with 20 channels. We took a 70%/15%/15% train/validation/test split.

The hyperparameters were optimised (for just the ODE-RNN baseline as previously described) by performing most of a grid search over 32, 64, 128 hidden channels, 32, 64, 128 hidden layer size, and 1, 2, 3, 4 hidden layers. (The latter two hyperparameters corresponding to the vector fields of the ODE-RNN and Neural CDE models.)

The hyperparameters selected for the ODE-RNN model were 128 hidden channels, and a vector field given by a feedforward neural network with hidden layer size 64 and 4 hidden layers. In order to keep the number of parameters the same between each model, this was reduced to 90 hidden channels and hidden layer size 40 for the Neural CDE model, and increased to 160 hidden channels for the GRU-Δt, GRU-D and GRU-ODE models. The resulting parameter counts were 88940 for the Neural CDE model, 87946 for the ODE-RNN model, 89290 for the GRU-D model, 88970 for the GRU-dt model, and 89180 for the GRU-ODE model.