Log Signatures in Machine Learning

Shujian Liao

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
of
University College London.

Department of Mathematics
University College London

September 29, 2022

I, Shujian Liao, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Acknowledgements

First, I wish to express my deepest gratitude to my supervisor Hao Ni, without whom my Ph.D. and this thesis would not be possible. Hao is a great mentor both in life and academy, and provides me a good chance to learn things with her patience and enthusiastic support.

I am grateful to all my academic collaborators and colleagues; in particular, Jian Chen, Hang Lou, Marc Sabate-Vidales, Kevin Schlegel, Lukasz Szpruch, Jiajie Tao, Magnus Wiese, Baoren Xiao, and Weixin Yang. I would also like to thank my friends who have supported me during the tough Ph.D. journey and made it fruitful. They are Yanlong Fang, Yupeng Jiang, Zhuang Jiang, Xiaoshu Sun, Weiguan Wang, Yuting Xiao, Guang Yang. In addtion, I would like to thank my old friends: Yao Jing, Yangkun Wei, and Qinghao Zeng.

Last but not least, I owe my thanks to my parents for their consistent encouragement. I will be forever grateful for your infinite support throughout my life.

Abstract

Rough path theory, originated as a branch of stochastic analysis, is an emerging tool for analysing complex sequential data in machine learning with increasing attention. This is owing to the core mathematical object of rough path theory, i.e., the *signature/log-signature* of a path, which has analytical and algebraic properties. This thesis aims to develop a principled and effective model for time series data based on the log-signature method and the *recurrent neural network* (RNN). The proposed (generalized) *Logsig-RNN* model can be regarded as a generalization of the RNN model, which boosts the model performance of the RNN by reducing the time dimension and summarising the local structures of sequential data via the log-signature feature. This hybrid model serves as a generic neural network for a wide range of time series applications.

In this thesis, we construct the mathematical formulation for the (generalized) Logsig-RNN model, analyse its complexity and establish the universality. We validate the effectiveness of the proposed method for time series analysis in both supervised learning and generative tasks. In particular, for the *skeleton human action recognition* tasks, we demonstrates that by replacing the RNN module by the Logsig-RNN in state-of-the-art (SOTA) networks improves the accuracy, efficiency and robustness. In addition, our generator based on the Logsig-RNN model exhibits better performance in generating realistic-looking time series data than classical RNN generators and other baseline methods from the literature. Apart from

that, another contribution of our work is to construct a novel *Sig-WGAN* framework to address the efficiency issue and instability training of traditional generative adversarial networks for time series generation.

Impact Statement

The main motivation of the thesis lies in the growing interest in the applications of machine learning for time series modelling. Recurrent neural network (RNN), as one of the popular deep learning methods, has been empirically proven to excel in sequential data tasks. However, RNN and its variants suffer from the gradient vanishing/explosion issues and curse of dimensionality, and hence struggle to handle long-term temporal dependency of time series. To address these issues, we propose a novel and hybrid Logsig-RNN model by combining rough path theory with RNNs, leading to the performance gain and superior robustness.

This thesis is likely to aid a variety of fields. First of all, traditional stochastic models, such as diffusion processes, are largely parametric, limiting their ability to simulate complex and random phenomena in the real world. Our method could provide new insights, as it is a non-parametric model which has the universality and can be trained in a data-driven way. Besides, in the machine learning field, the developed method could provide a generic solution to any time series data related problems as an enhancement of the RNNs. For instance, our algorithm demonstrates performance boost in handwriting digits classification and skeleton human action recognition tasks. Further applications include financial market prediction, molecular dynamic data classification, and clinical anomaly detection, etc.. Furthermore, the proposed Logsig-RNN model can be an effective tool for generative models to simulate synthetic time series. Synthetic time series generation is a hot

topic with increasing attention, as it can be used to mitigate the data privacy risk and enable data sharing. Our work demonstrates that our method on the financial data and medical data achieves superior performance, indicating the method could be valuable in industries which are involved in the sensitive and confidential data or suffer from data scarcity.

Contents

List of Figures

List of Tables

Introduction

Stochastic differential equations (SDEs) are useful tools for modelling random phenomena of many physical, chemical and biological systems of interacting particles as well as financial derivative pricing and risk management ([1, 2, 3, 4]). The main objective of this thesis is to develop an effective model for time series learning using the *log-signature* in rough path theory, motivated by approximating the solutions to the SDEs. We substantially investigate its applications in supervised learning and data generation tasks.

Our approach is enlightened by exploiting the connection between the stochastic differential equations and architectures of neural networks, which has recently been a popular research area ([5, 6, 7, 8]). For example, neural ordinary differential equations [5] , as a continuous analogue of a ResNet, connect residual networks and discretized ODEs; Funahashi et al.[9], motivated by approximating the trajectory of dynamical system, introduced the continuous recurrent neural network (RNN). A typical continuous RNN has the form

$$\dot{Y}_t = -\frac{Y_t}{\tau} + A\sigma(BY_t) + I_t, \tag{1}$$

where I_t and Y_t are an input and output at time t respectively[1]. Rough Path Theory

[1] τ is a constant, A and B are matrices and σ is an activation function.

teaches us that is more robust to consider the differential equation of the type

$$dY_t = f(Y_t)dX_t, \tag{2}$$

and replace I as an input with its integral. We can rewrite (1) in this form by setting $X_t = (t, \int_{s=t_0}^{t} I_s ds)$ and $f(y,(t,x)) = -\frac{y}{\tau}t + A\sigma(By)t + x$. This allows the input to be of a broader type, and X needs not even be differentiable for the equation to be well defined. This reformulation provides a much broader class of mathematical models for functionals on streamed data, of which the continuous RNN is a special case.

Lyons [10] made sense of the solution to Equation (2) driven by a path rougher than semi-martingales, which can apply to paths driven by vector valued Brownian motion, diffusion processes, and many processes outside the SDE case. With the extension theorem [10], the control of the p-variation and the iterated integrals of X (i.e., the *signature* of X) up to degree $\lfloor p \rfloor$ is sufficient to control the solution of Equation (2). The universal limit theorem in [10] allows rough signals with p much larger than 1 and demonstrates that by viewing the driving signals as rough path, the differential equation (2) guarantees unique solution. While the signature provides coarse global description of X, the log-signature, as a more parsimonious transformation, carries exactly the same information as the signature; it is able to summarize and vectorize complex un-parameterized streams of multi-modal data effectively and locally with a low dimensional representation.

One area where the representation ability of the log-signature has been worked out in detail is with numerical analysis of SDEs. SDEs of the form (2) provide a general class of functionals on the path space. The most effective high order numerical approximation schemes for SDEs show that describing a path through the log-signature enables one to effectively approximate the solution to the equation and any linear functional of that solution globally over interval the path is defined on, without further dependence on the fine details of the *recurrent structure* of the

streamed data. It leads (in what is known as the log-ODE method) to produce a state-of-the-art discretization method of Inhomogenous Geometric Brownian Motion (IGBM)[11]. We exploit this understanding to propose a simple but surprisingly effective neural network module ((generalized) Logsig-RNN) by blending the Log-Signature (Sequence) Layer with the RNN type layer as an universal model for functionals on un-parameterized (and potentially complex) streamed data.

Our approach is a novel application of rough path theory in machine learning, which has been an emerging and active research area. The empirical applications of the rough paths theory primarily focused on the *signature* feature, which serves as an effective feature extraction, e.g. for online handwritten Chinese character/text recognition([12], [13]), action classification in videos [14], and financial data analysis ([15], [16]). In addition, those previous works mainly combine the signature with convolutional neural network or fully connected neural network. To our best knowledge, the proposed method is the first of its kind integrating the sequence of log signature with the recurrent type neural network. The log-signature has been used as a local feature descriptor for gesture [17] and action recognition [14]. These use cases are bespoke; in contrast, the proposed (generalized) Logsig-RNN is a general method for sequential data with outstanding performance in various machine learning tasks. Moreover, we theoretically justified the universality of our model and extend the work on the back-propagation algorithm of the log-signature transformation in [18] to *sequence* of log-signatures.
This thesis has the following major contributions:

- We propose the Log-Signature (Sequence) Layer as a transformation of sequential data, and design its backpropagation through time algorithm. It is highlighted that the Log-Signature Layer can be inserted between other neural network layers conveniently, not limited to the pre-defined feature extraction.

- We propose a novel and generic neural network model, i.e. (generalized)

Logsig-RNN model, by blending the Log-Signature Layer with RNN and establish the universality of the Logsig-RNN model for the approximation of SDEs solutions.

- It proposes Path Transformation Layers (PT)-Logsig-RNN model by inserting path transformation layers (e.g. Graph Neural Network, linear projection layers) in the front to efficiently and effectively exploit the spatial-temporal structure of the time series data. We show by experiments on both synthetic and empirical data of supervised learning tasks, especially on Skeleton Human Action Recognition tasks. The Logsig-RNN demonstrates superior accuracy, efficiency and robustness against traditional RNN model.

- It develops a novel generative framework Sig-WGAN and proposes an effective generator for the time series data, i.e. the generalized Logsig-RNN . The generated paths are overall closer to the real paths regarding test metrics (especially the correlation metric) than the paths generated by benchmarks including classical RNN. The generalized Logsig-RNN exhibits superior robustness in various tasks against the benchmarks.

The thesis is organized as follows:

Chapter 1 introduces the preliminary of rough path theory and neural networks. The former includes the mathematically principled technology of signatures and log-signatures as representations for streamed data, while the latter focuses on recurrent neural networks.

In Chapter 2, based on [19, 20], we discuss the mathematical formulation of the (generalized) Logsig-RNN model, which is motivated by the numerical approximation of the solutions to SDEs, and prove its universality. We first define the Log-Signature Layer which is a mapping that transforms potentially high-frequency streamed data to a sequence of log-signatures over sub-time interval. Then we apply

it on the recurrent type neural networks to produce sequential outputs. We also propose a generalized version of Logsig-RNN which is a sequence-to-sequence model and can generate time series data of arbitrary length. It is highlighted that the (generalized) Logsig-RNN can be inserted between other neural network layers conveniently, not limited to the pre-defined feature extraction. Given this flexibility, we can combine it with different modules when dealing with different tasks and investigate its advantages against conventional RNN layer.

Chapter 3, based on [19, 21], explores the application of Logsig-RNN in supervised learning. The illustrative examples are taken on synthetic SDEs data for regression and UCI pen-digit data[2] for classification. The challenging skeleton-based human action recognition (SHAR) tasks are investigated on the Chalearn 2013 [22] and NTU RGB+D 120 dataset [23]. As an enhancement of the RNN layer, the proposed Logsig-RNN module can reduce the time dimension, handle irregular time series and improve the time efficiency as well as the robustness against missing data and varying frame rates.

Chapter 4, which is written around [20], investigates synthesizing data using generalized Logsig-RNN. We establish high-fidelity time-series generative framework, the SigWGAN, by combining continuous time stochastic models with the newly proposed signature W_1 metric. The SigWGAN originates from the universal and principled mathematical features to characterize the measure induced by time series. It allows turning computationally challenging GAN min-max problem into supervised learning while generating high fidelity samples. We validate the generative ability and robustness of the generalized Logsig-RNN on synthetic data generated by popular quantitative risk models and medical eICU data[3].

[2] https://archive.ics.uci.edu/ml/datasets/Pen-Based+Recognition+of+Handwritten+Digits

[3] https://eicu-crd.mit.edu/

Chapter 1

Preliminary

1.1 Rough path theory

We introduce the signature and log-signature of a path in this section. Their efficient representations of unparametrized streamed data enables them to be competitive features extractor in the applications of data science [24, 25]. Throughout the thesis, let J be a compact interval, $E := \mathbb{R}^d$ and $F := \mathbb{R}^e$ be two real vector spaces and $X : J \to E$ be a continuous path.

1.1.1 Tensor algebra

We start by introducing the extended tensor algebra space, in which the signature and log-signature take values.

Definition 1.1 (Extended tensor algebra)**.** The extended tensor algebra or formal series of tensors over E, denoted by $T((E))$, is defined to be the space of the following sequence,

$$T((E)) = \{\mathbf{a} = (a_0, a_1, \dots, a_n, \dots) \mid a_n \in E^{\otimes n}\}.$$

It is equipped with two operations, i.e., an addition $+$ and a product $\otimes$, defined as follows. Let $\mathbf{a} = (a_i)_{i=0}^{\infty}$, $\mathbf{b} = (b_i)_{i=0}^{\infty} \in T((E))$, then

$$\mathbf{a}+\mathbf{b} = (a_i + b_i)_{i=0}^{\infty},$$
$$\mathbf{a}\otimes\mathbf{b} = (\sum_{k=0}^{i} a_i \otimes b_{i-k})_{i=0}^{\infty}.$$

The space $T((E))$ endowed with the two operations and the natural action of $\mathbb{R}$ by $\lambda\mathbf{a} = (\lambda a_i)_{i=0}^{\infty}$ is a real non-commutative unital algebra, with unit $\mathbf{1} = (1,0,0,\dots)$. The element $\mathbf{a}$ is invertible if and only if $a_0 \neq 0$, whose inverse is given by

$$\mathbf{a}^{-1} = \frac{1}{a_0}\sum_{n\geq 0}\left(1-\frac{\mathbf{a}}{a_0}\right)^{\otimes n}.$$

The extended tensor algebra $T((E))$ consists of all possible tensor series, including the ones with the infinite number of non-zero tensors. To distinguish with $T((E))$, we also introduce the tensor algebra, which is composed with all the finite tensor sequences and forms a subalgebra of $T((E))$.

Definition 1.2. The tensor algebra over E denoted by $T(E)$ is a subalgebra of $T((E))$, which is given by

$$T(E) := \bigoplus_{n=0}^{\infty} E^{\otimes n}.$$

In practice, we often work on the truncated tensor series. Therefore we define the truncated tensor algebra, which consists of all sequences of any finite length.

Definition 1.3. Let $n \geq 1$ be an integer and $B_n = \{(a_0, a_1, \dots,) \mid a_0 = \cdots = a_n = 0\}$. The truncated tensor algebra $T^{(n)}(E)$ over E of order n is defined as the quotient algebra

$$T^{(n)}(E) = T((E))/B_n.$$

For $n \geq 1$, we define the projection

$$\pi_n : T((E)) \to E^{\otimes n}$$
$$\mathbf{a} \to a_n.$$

The canonical projection is defined as the mapping

$$\Pi_n : T((E)) \to T^{(n)}(E)$$
$$\mathbf{a} \to (a_i)_{i=0}^n.$$

We proceed with introducing the admissible norm of tensor powers $E^{\otimes n}$.

Definition 1.4. We say that the tensor powers of E are endowed with an admissible norm $||\cdot||$, if the following conditions hold:

1. For each $n \geq 1$, the symmetric group $\mathscr{S}_n$ acts by isometry on $E^{\otimes n}$, i.e.

$$||\sigma v|| = ||v||, \ \forall v \in E^{\otimes n}, \ \forall \sigma \in \mathscr{S}_n$$

2. The tensor product has norm 1, i.e. $\forall n, m \geq 1$,

$$||v \otimes w|| \leq ||v||\,||w||, \ \forall v \in E^{\otimes n}, \ w \in E^{\otimes m}.$$

In this thesis, we use L_2 norm denoted as $||\cdot||_{L_2}$ for tensor powers. To define the norm, let $(e_1, \cdots, e_d)$ be a basis for d-dimensional space E, and $(e_1^*, \cdots, e_d^*)$ is a canonical basis for the dual space E^*. The elements $(e_I = e_{i_1} \otimes \cdots \otimes e_{i_n})_{I=(i_1,\ldots,i_n)\in\{1,\ldots,d\}^n}$ form a basis for $E^{\otimes n}$. More specifically,

$$\langle e_{i_1}^* \otimes \cdots \otimes e_{i_n}^*, e_{j_1} \otimes \cdots \otimes e_{j_n} \rangle = \delta_{i_1,j_1} \cdots \delta_{i_n,j_n},$$

where

$$\delta_{i,j} = \begin{cases} 1, & \text{if } i = j, \\ 0, & \text{if } i \neq j. \end{cases}$$

The linear mapping e_I^* acts on a tensor power $\mu \in E^{\otimes n}$ picks up the coefficient of the monomial e_I in μ. Actually the linear mapping e_I^* extends naturally to the linear mapping $(E^*)^{\otimes n} \to T((E))^*$ defined by

$$e_I^*(\mathbf{a}) = e_I^*(a_n), \quad \forall \mathbf{a} \in T((E)).$$

Definition 1.5. Let $\mu \in E^{\otimes n}$ be a tensor power. The L_2 norm of tensor powers is given by

$$||\mu||_{L_2} := \left(\sum_{I=(i_1,\ldots,i_n)} |e_I^*(\mu)|^2 \right)^{\frac{1}{2}}$$

where $i_1, \ldots, i_n \in \{1, \ldots, d\}$.

1.1.2 Path space

To define an appropriate class of paths used in our analysis in the thesis, we begin with the introduction of the p-variation to quantify the oscillation or the roughness of a path.

Definition 1.6 (p-Variation)**.** Let $p \geq 1$ be a real number. Let $X : J \to E$ be a continuous path. The p-variation of X on the interval J is defined by

$$||X||_{p,J} = \left[\sup_{\mathscr{D} \subset J} \sum_{j=0}^{r-1} \left| X_{t_{j+1}} - X_{t_j} \right|^p \right]^{\frac{1}{p}}, \tag{1.1}$$

where the supremum is taken over any finite time partition of J, i.e. $\mathscr{D} = (t_0, t_1, \cdots, t_r)$.

Let $V_p(J,E)$ denote the range of any continuous path mapping from J to E of finite p-variation.

Example 1.1. For any continuously differentiable path $X : J \to E$, the 1-variation of X is finite.

Example 1.2. [26] A fractional Brownian motion (fBM) with Hurst parameter H has sample paths of finite p-variation a.s. for $p > \frac{1}{H}$. The larger H is, the rougher fBM sample path is. For example, Brownian motion is a fBM with $H = 0.5$. It has finite $(2+\varepsilon)$-variation a.s $\forall \varepsilon > 0$, but it has infinite p-variation a.s.$\forall p \in [1,2]$.

Next, given two normed space K, V, we introduce the Lipschitz conditions for the function $f : K \to V$ to suffice for our purpose of defining controlled differential equations used in this thesis.

Definition 1.7 (Lipschitz norm)**.** The Lipschitz norm of a function $f : K \to V$, denoted by $||f||_{Lip,K}$, is defined as

$$||f||_{\mathrm{Lip},K} := \sup_{x \neq y, x, y \in K} \frac{||f(x) - f(y)||}{d(x,y)}, \tag{1.2}$$

where $d(x,y)$ is metric on K.

Definition 1.8 (Lipschitz map)**.** A map $f : K \to V$ between two normed spaces K, V is called $\gamma-$Lipschitz with $\gamma \geq 1$,

$$f \in Lip(\gamma),$$

if f is $\lfloor \gamma \rfloor$ times continuously differentiable and there exists a nonnegative constant M such that

$$|f|_{\circ\gamma} := \max_{|k| \leq \lfloor \gamma \rfloor} ||D^k f||_{\infty;K} + ||f||_{C^{\lfloor \gamma \rfloor, \{\gamma\}}} \leq M,$$

where $||\cdot||_{\infty;\cdot}$ denotes the infinity norm of a function $f : K \to V$ as follows

$$||f||_{\infty;K} := \sup_{x \in K} ||f(x)||.$$

and $||\cdot||_{C^{k,\alpha}}$ is the Hölder norm, where $\gamma = \lfloor \gamma \rfloor + \{\gamma\}$. $\lfloor \cdot \rfloor$ and $\{\cdot\}$ take the integer and fractional parts of a real number respectively.

We then define a control function that is used in controlling the p-variation of a path.

Definition 1.9 (Control function)**.** A control function on $J = [0,T]$ is a continuous non-negative function ω on $\Delta_T := \{(s,t) \in J^2 : 0 \leq s \leq t \leq T\}$ which is super-additive in the sense that

$$\omega(s,t) + \omega(t,u) \leq \omega(s,u), \ \ \forall s \leq t \leq u \in J$$

and for which $\omega(t,t) = 0$ for all $t \in J$.

Lemma 1.1. *[10] The control function ω on J is uniformly continuous.*

Remark 1.1. We say that the p-variation of X is controlled by ω if for all $(s,t) \in \Delta_T$,

$$||X||_{p,[s,t]} \leq \omega(s,t)^{\frac{1}{p}}.$$

1.1.3 The signature of a path

Definition 1.10 (The Signature of a path)**.** Let J denote a compact interval and $X : J \to E$ be a continuous path of p-variation such that the following integration makes sense. Let $I = (i_1, i_2, \cdots, i_n)$ be a multi-index of length n, where $i_j \in \{1, \cdots, d\}, \forall j \in \{1, \cdots, n\}$. Define the coordinate signature of the path X associate with the index I as follows:

$$X_J^I = \int_{\substack{u_1 < \cdots < u_k \\ u_1, \ldots, u_k \in J}} \cdots \int dX_{u_1}^{(i_1)} \otimes \cdots \otimes dX_{u_n}^{(i_n)}.$$

The signature of X is defined as follows:

$$S(X)_J = (1, \mathbf{X}_J^1, \ldots, \mathbf{X}_J^k, \ldots). \tag{1.3}$$

where $\mathbf{X}_J^k = (X_J^I)_{I=(i_1,\cdots,i_k)}, \forall k \geq 1$. Let $S_k(X)_J$ denote the truncated signature of X of degree k, i.e.

$$S_k(X)_J := \Pi_k(S(X)_J) = (1, \mathbf{X}_J^1, \ldots, \mathbf{X}_J^k). \tag{1.4}$$

The signature of a path is an element of a group called the space of *grouplike elements*, which is a subset of $\widetilde{T}((E))$ defined as

$$\widetilde{T}((E)) = \{\mathbf{a} \in T((E)) \mid a_0 = 1\}.$$

To characterise the grouplike elements, we need to define the shuffle product. We say that a permutation $\sigma \in \mathfrak{S}_{r+s}$ is a *shuffle* of $\{1,\ldots,r\}$ and $\{r+1,\ldots,r+s\}$ if $\sigma(1) < \cdots < \sigma(r)$ and $\sigma(r+1) < \cdots < \sigma(r+s)$. We write $\sigma \in \text{Shuffles}(r,s)$. Let $I = (i_1,\ldots,i_r)$ and $J = (j_1,\ldots,j_s)$ be two arbitrary indices, the shuffle product of e_I^* and e_J^* is

$$e_I^* \sqcup\!\sqcup e_J^* = \sum_{\sigma \in \text{Shuffles}(r,s)} e^*_{k_{\sigma^{-1}(1)},\ldots,k_{\sigma^{-1}(r+s)}}.$$

Definition 1.11 (Grouplike elements)**.** An element $\mathbf{a} \in \widetilde{T}((E))$ is said to be grouplike if given any $\mathbf{e}^*, \mathbf{f}^* \in T((E^*))$, the following equality holds

$$\mathbf{e}^*(\mathbf{a})\mathbf{f}^*(\mathbf{a}) = (\mathbf{e}^* \sqcup\!\sqcup \mathbf{f}^*)(\mathbf{a}).$$

The set of grouplike elements is denoted by $G^{(*)}(E)$.

We also define the space of projection of elements in $G^{(*)}(E)$ into $T^{(n)}(E)$ as follows

$$G^{(n)}(E) := \{\Pi_n(\mathbf{a}) | \forall \mathbf{a} \in G^{(*)}(E)\} \subset T^{(n)}(E).$$

Theorem 1.2 (Shuffle product property)**.** *[10] Let $I = (i_1,\ldots,i_r)$ and $J = (j_1,\ldots,j_s)$ be two arbitrary indices. For every path X of finite 1-variation, it holds that*

$$e_I^*(S(X))e_J^*(S(X)) = (e_I^* \sqcup\!\sqcup e_J^*)(S(X)).$$

The signatures take values in the space of grouplike elements due to the shuffle product property.

The multiplicative property asserts that the signature is a homomorphism. First we define the concatenation of two paths.

Definition 1.12. Let $X : [0,s] \to E$ and $Y : [s,t] \to E$ be two continuous paths. Their concatenation is the path denoted by $X * Y : [0,t] \to E$ defined by

$$(X * Y)_u = \begin{cases} X_u, & u \in [0,s], \\ Y_u - Y_s + X_s, & u \in [s,t]. \end{cases}$$

Theorem 1.3 (Chen's identity)**.** *Let $X : [0,s] \to E$ and $Y : [s,t] \to E$ be two continuous paths of bounded 1-variation. Then*

$$S(X * Y) = S(X) \otimes S(Y).$$

The multiplication property is useful when computing the signature of a piecewise linear path numerically.

The next property is the invariance under time reparameterization. Reparameterizing a path inside the interval does not change its signature.

Lemma 1.4 (Invariance under time parameterization)**.** *[10] Let $X \in V_1(J,E)$ and a path $\tilde{X} : J \to E$ is the time re-parameterization of X, i.e. there exists increasing function $r : J \to J$ such that*

$$\tilde{X}(t) = X(r(t)).$$

Then

$$S(X)_J = S(\tilde{X})_J. \tag{1.5}$$

In Figure 1.1, speed changes result in different time series representation but the same signature features. It means that signature features can reduce dimension

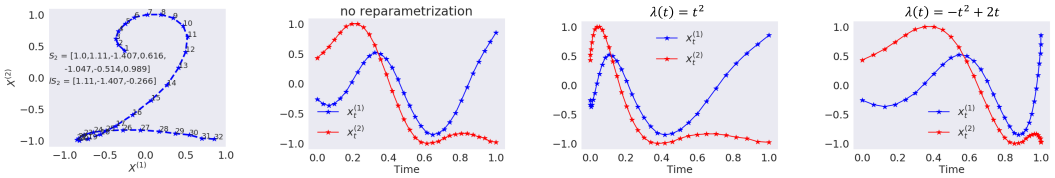


Figure 1.1: The top-left figure represents the trajectory of the digit 2, and the rest of figures plot the coordinates of the pen location via different speed respectively, which share the same signature and log signature given in the first subplot.

massively by removing the redundancy caused by the speed of traversing the path.

The uniqueness of signature states that the signature of a path is unique up to the tree-like equivalence. We first define the tree-like path.

Definition 1.13 (Tree-like path)**.** A path $X : J = [S,T] \to E$ is tree-like if there exists a continuous function $h : J \to [0,+\infty)$ such that $h(S) = h(T) = 0$ and such that, for all $s,t \in J$ with $s \leq t$,

$$||X_t - X_s|| \leq h(s) + h(t) - 2 \inf_{u \in [s,t]} h(u).$$

The function h is called a height function for the path X. Intuitively, a tree-like path is a trajectory in which there is a section where the path exactly retraces itself. The tree-like equivalence is defined as follows: we say that two paths X and Y are the same up to the tree-like equivalence if and only if the concatenation of X and the inverse of Y is tree-like.

Theorem 1.5 (Uniqueness of the signature)**.** *[27] Let $X \in V_p(J,E)$. Then $S(X)$ determines X up to the tree-like equivalence.*

Theorem 1.5 shows that the signature of the path can recover the path trajectory under a mild condition. The uniqueness of the signature is important, as it ensures itself to be a discriminative feature set of unparameterized streamed data.

Any functional on the path can be rewritten as a function on the signature based on the uniqueness of the signature. The signature of the path has the universality,

i.e. that any continuous functional on the signature can be well approximated by the linear functional on the signature (Theorem 1.6)[24].

Theorem 1.6 (Signature approximation theorem)**.** *Suppose $f : K \to \mathbb{R}$ is a continuous function, where K is a compact subset of* $\mathbf{S}(V_p(J,E))$[1]*. Then $\forall \varepsilon > 0$, there exists a linear functional $L \in T((E))^*$ such that*

$$||f - L||_{\infty;K} \leq \varepsilon. \tag{1.6}$$

Lastly, we state the decay rate of the signature of path with finite p-variation, which implies that the information of signatures decays exponentially as the degree increases.

Lemma 1.7 (Factorial decay of the signature)**.** *Let $X \in V_p(J,E)$. There exists a constant β depending only on p such that the k-th level signature $\mathbf{X}_J^k$ is controlled by the p-variation of the path in the following way*

$$||\mathbf{X}_J^k|| \leq \frac{||X||_{p,J}^k}{\beta k!}.$$

1.1.4 The log-signature of a path

After discussion of signature, we introduce the log -signature of a path in this subsection. First we define the logarithm map of an element in T((E)).

Definition 1.14 (Logarithm map)**.** Let $\mathbf{a} = (a_i)_{i=0}^{\infty} \in T((E))$. Then the logarithm map denoted by log is defined as follows:

$$\log(\mathbf{a}) = \log(a_0) + \sum_{n=1}^{\infty} \frac{-1}{n}(1 - \frac{\mathbf{a}}{a_0})^{\otimes n}, \forall \mathbf{a} \in T((E)). \tag{1.7}$$

Lemma 1.8. *The logarithm map is bijective on the domain $\{\boldsymbol{a} \in T((E)) | a_0 = 1\}$.*

[1] $\mathbf{S}(V_p(J,E))$ denotes the range of the signature of $X \in V_p(J,E)$.

Definition 1.15 (The log-signature of a path)**.** Similar as the signature, the log signature of path X by $\log(S(X))$ is the logarithm of the signature of the path X, denoted by $lS(X)$. Let $lS_k(X)$ denote the truncated log signature of a path X of degree k.

There is one-to-one correspondence between the signature and log signature. Define the exponential map of an element $\mathbf{a}$ in $\widetilde{T}((E))$ to be

$$\exp(\mathbf{a}) = \sum_{n=0}^{\infty} \frac{\mathbf{a}^{\otimes n}}{n!}.$$

Lemma 1.9. *The exponential and logarithm map are one-to-one and they are each other's inverse.*

While the log signature is a parsimonious representation of the signature, it retains lower dimension compared with the signature, which is of a good property when being applied in machine learning algorithm to help avoid the curse of dimensionality and improve the time efficiency.

Let us consider the linear subspace of $T((E))$ equipped with the Lie bracket operation $[\cdot,\cdot]$, defined as follows:

$$[a,b] = a \otimes b - b \otimes a.$$

If F_1 and F_2 are two linear subspaces of $T((E))$, let us denote by $[F_1,F_2]$ the linear span of all the elements of the form $[a,b]$, where $a \in F_1$ and $b \in F_2$. Consider the sequence $(L_n)_{n\geq 0}$ be the subspace of $T((E))$ defined recursively as follows:

$$L_0 = 0; \forall n \geq 1, L_n = [E, L_{n-1}]. \tag{1.8}$$

Definition 1.16. The space of Lie formal series over E, denoted as $\mathscr{L}((E))$ is de-

fined as the following subspace of $T((E))$:

$$\mathscr{L}((E)) = \{l = (l_0, \cdots, l_n, \cdots) | \forall n \geq 0, l_n \in L_n\}. \tag{1.9}$$

Theorem 1.10. *[10] For any path X of finite* 1*-variation , there exist* $\lambda_{i_1,\cdots,i_n}$ *such that the log-signature of X can be expressed in the following form*

$$lS(X) = \sum_{i=1}^{d} \lambda_i e_i + \sum_{\substack{n\geq 2 \\ e_{i_1},\cdots,e_{i_n} \\ \in\{1,\cdots,d\}}} \lambda_{i_1,,\cdots,i_n}[e_{i_1},[\cdots,[e_{n-1},e_n]]].$$

The above theorem shows that the dimension of the truncated log-signature is no greater than that of the truncated signature due to the linear dependence of $[e_{i_1},[e_{i_2}\cdots,[e_{n-1},e_n]]]$. For example, $[e_i,e_j] = -[e_j,e_i]$.

The analytic formula for the dimension of the truncated log signature can be found in the following theorem.

Theorem 1.11. *The dimension of the space of the truncated log signature of d-dimensional path of degree n over d letters is given by:*

$$\mathscr{DL}_n = \frac{1}{n}\sum_{d|n} \mu(d) q^{n|d}$$

where μ *is the Mobius function, which maps n to*

$$\begin{cases} 0, & \text{if } n \text{ has one or more repeated prime factors} \\ 1, & \text{if } n = 1 \\ (-1)^k & \text{if } n \text{ is the product of } k \text{ distinct prime numbers} \end{cases}$$

Example 1.3. In Figure 1.2, we compare the dimension of the signature and log signature of path of dimension up to $d = 5$ and truncated degree up to $M = 5$. The

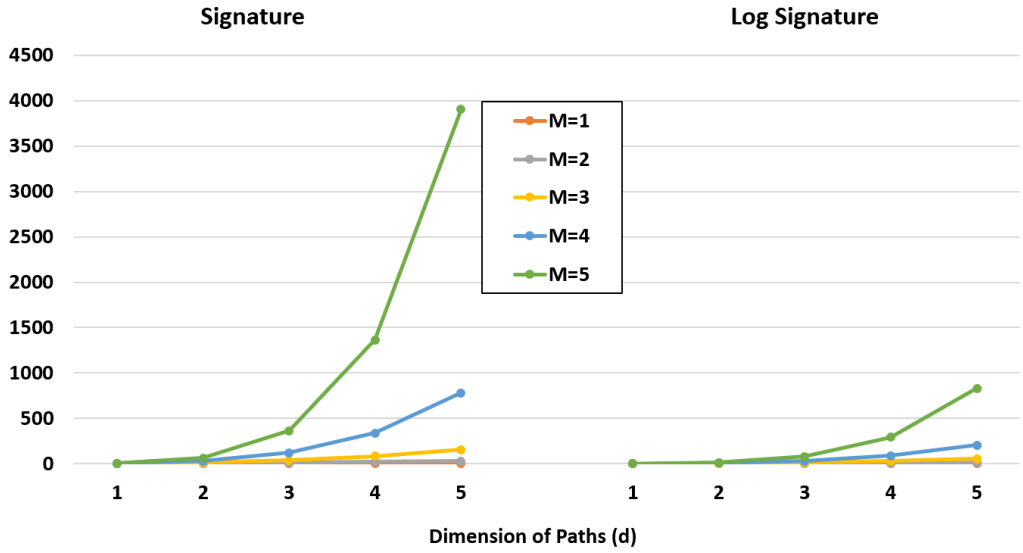


Figure 1.2: The dimensions of signatures and log-signatures comparison, where M is the truncated degree.

is noticeable dimension reduction of the log signature.

It is an immediate consequence of the bijection between the signature and log-signature, and the invariance of the signature (Lemma 1.4). Re-parameterizing a path does not change its log signature (Figure 1.1).

Like the signature, the log-signature has the uniqueness stated in the following theorem.

Theorem 1.12 (Uniqueness of the log-signature)**.** *Let $X \in V_p(J,E)$. Then $lS(X)$ determines X up to the tree-like equivalence defined in Definition 1.13.*

Theorem 1.12 shows that the signature of the path can recover the path trajectory under a mild condition.

Lemma 1.13. *A simple sufficient condition for the uniqueness of the log-signature of a path of finite length is that one component of X is monotone.*

Both the signature and log-signature take the functional view on discrete time series data, which allows a unified way to treat time series of variable length and missing data. For example, we chose one pen-digit data of length 53 and simulate 1000 samples of modified pen trajectories by dropping at most 16 points from it, to mimic the missing data of variable length case (See one example in Figure 1.3). Figure 1.4 shows that the mean absolute relative error (MARE) of the signature and log-signature of the missing data is no more than 6%. Besides, the MARE of the log-signature feature is less that of the signature feature which implies it is more robust against missing data, and it is of lower dimension compared with the signature feature, i.e. the log-signature has dimension 5 while the signature has dimension 14.

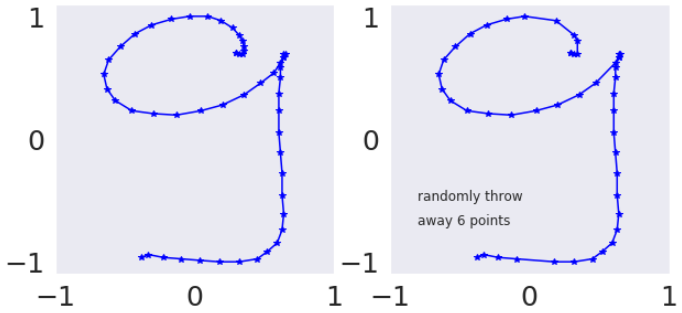


Figure 1.3: (Left) The chosen pen trajectory of digit 9. (Right) The simulated path by randomly dropping at most 16 points of the pen trajectory on the left.

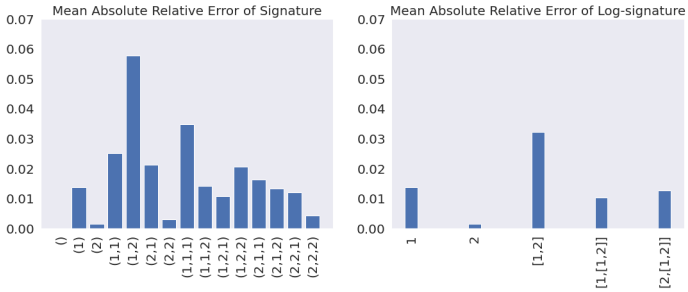


Figure 1.4: Signature and log-signature comparison for the missing data case.

1.1.5 Geometric rough paths

Definition 1.17 (p-variation distance)**.** Let $\Delta_T := \{(s,t) \in J^2 : 0 \le s \le t \le T\}$ and $p \ge 1$. Given $\mathbf{X}, \mathbf{Y} : \Delta_T \to G^{(p)}(E)$, we define the p-variation distance by

$$d_{p-var;[0,T]}(\mathbf{X},\mathbf{Y}) = \max_{k=1,\dots,\lfloor p \rfloor} \sup_{\mathscr{D}} \left(\sum_{t_i \in \mathscr{D}} ||\mathbf{X}^k_{t_{i-1},t_i} - \mathbf{Y}^k_{t_{i-1},t_i}||^{p/k} \right)^{1/p},$$

where the supremum is taken over any finite partition $\mathscr{D}$ of $[0,T]$.

Let $o = \mathbf{1}$ be the unit element in $G^{(p)}(E)$, then we note $d_{p-var;[0,T]}(\mathbf{X}, o) = ||\mathbf{X}||_{p-var}$, is the p-variation norm of $\mathbf{X}$.

Definition 1.18 (Geometric p-rough paths)**.** A geometric p-rough path is the limit of signatures of degree $\lfloor p \rfloor$ of bounded variation paths in the p-variation distance. The space of geometric p-rough paths is denoted by $G\Omega_p(E)$.

Theorem 1.14 (Extension theorem)**.** *[10] Let $\boldsymbol{X} \in G\Omega_p(E)$ in $[0,T]$ with finite p-variation controlled by a control ω. For any $n \ge \lfloor p \rfloor$, there exists a unique extension of $\mathbf{X}$ to a geometric n-rough path $(1, \mathbf{X}^1_J, \dots, \mathbf{X}^{\lfloor p \rfloor}_J, \dots, \mathbf{X}^n_J) \in G^{(n)}(E) \subset T^{(n)}(E)$ with finite p-variation controlled by ω, i.e., there exists some constant $\beta > 0$ depend only on p such that*

$$||\mathbf{X}^k_{s,t}|| \le \frac{\omega(s,t)^{\frac{k}{p}}}{\beta(\frac{k}{p})!}, \ \forall k \ge 1, \ \forall (s,t) \in \Delta_T.$$

1.2 Neural network

The neural network is one of the main streams in solving machine learning problems which are what we focus on in this thesis. In this section, we introduce the supervised learning, the basics of neural networks and generative adversarial network.

1.2.1 Supervised learning

The supervised learning is a type of machine learning tasks which aims to learn a function that maps from input to output based on given input-output pairs. Suppose the data set has N input-output pairs $\{(x_1, y_1), \ldots, (x_N, y_N)\}$ where the input $x \in \mathbb{R}^d$ and the label $y \in \mathbb{R}^e$. The learning process is to find a function $f : \mathbb{R}^d \to \mathbb{R}^e$ such that the following optimization problem is satisfied

$$f = \underset{f}{\operatorname{argmax}}\, g(y, f(x)),$$

where g is the scoring function which measures the prediction ability of the learned function. The scoring function, for instance, is the likelihood for classification tasks.

1.2.2 Introduction to neural networks

In recent years, neural networks have been very popular in machine learning, which gain wide attention. It is one way to construct the function with forecasting ability in supervised learning problems which has been successfully applied to many real world problems[28]. The simplest neural network (NN) consists of two different layers: one input layer where the data flows in and one output layer where the prediction $\hat{y} \in \mathbb{R}^e$ is made, which can be formulated as:

$$\hat{y} = \sigma(\mathbf{W}x + b), \forall x \in \mathbb{R}^d,$$

where $\mathbf{W} \in \mathbb{R}^{e \times d}$ is the weight matrix, $b \in \mathbb{R}^e$ is the bias vector and σ is called the **activation** function which can add non-linearity to the neural network. Let the set of parameters be denoted as Θ where $\Theta = \{W, b\}$. The process of training a neural network is indeed the following optimization problem

$$\underset{\Theta}{\operatorname{argmin}}\, \mathscr{L}(\Theta | \{x_i, y_i\}_{i=1}^N),$$

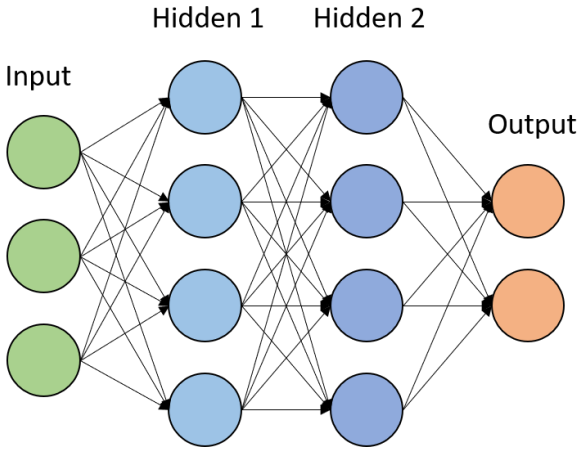


Figure 1.5: Neural Network with two hidden layers.

where $\mathscr{L}$ is called the **Loss** function which measures the difference between the ground-truth y_i and the prediction $\hat{y}_i$. We will discuss the choices of activation and loss functions in the latter part of this section, both of which play important roles in the design of neural networks.

To make the algorithm more complex, people increase the number of layers. The NN with L layers is a nonlinear map $h_L : \mathbb{R}^d \to \mathbb{R}^e$ which receives the input x and maps it to the output $\hat{y}$ such that $\hat{y} := h_L(x)$, where there are three different layers: input layer, hidden layer and output layer. Figure 1.5 shows a neural network with two hidden layers. In the first part of this section, we first introduce the basic feed forward neural network (FFNN). Since we aim at modelling sequential data, where the popular structure being used is recurrent neural network (RNN), we give the definition of RNN as well.

Feed Forward Neural Network Conventionally, the feed forward neural networks (FFNN) or deep neural network (DNN) is a NN with $L > 2$. Let $l \in \{1, \ldots, L-1\}$ be the index for hidden layers and n_l be the number of neurons, i.e. the number of elements in the output vector of the l^{th} layer. Layers in FFNN are defined in the following recursive way:

- Input layer: $h_0 : \mathbb{R}^d \to \mathbb{R}^d$,

$$h_0(x) = x, \forall x \in \mathbb{R}^d$$

- Hidden layer: $h_l : \mathbb{R}^{n_{l-1}} \to \mathbb{R}^{n_l}$

$$h_l = \sigma_l(\mathbf{W_l} h_{l-1} + b_l),$$

where $\mathbf{W_l} \in \mathbb{R}^{n_l \times n_{l-1}}$ is the weight matrix, $b_l \in \mathbb{R}^{n_l}$ is the bias term and σ_l is called **activation** function.

- Output layer: $h_L : \mathbb{R}^{n_{L-1}} \to \mathbb{R}^e$

$$h_L = \sigma_L(\mathbf{W_L} h_{L-1} + b_L),$$

where $\mathbf{W_L} \in \mathbb{R}^{e \times n_{L-1}}$ is the weight matrix, $b_L \in \mathbb{R}^e$ is the bias term and σ_L is activation function.

Thus when training the FFNN, the prediction is $\hat{y_i} = h_L(x)$ and the parameter set is $\Theta := \{W_l, b_l\}_{l=1}^L$.

Recurrent Neural Networks In this thesis, we mostly deal with time-dependent data, such as solutions of stochastic differential equations, human actions, hand-written characters, etc.. Given the time series type of data, it is natural to consider using the recurrent neural network (RNN) as part of the models. Unlike the feed forward neural network (FFNN), the data in RNN would flow backward from the output of hidden layer to the layer itself (Figure 1.6). The RNN is also composed with three types of layers, i.e. the input layer, the hidden layer and the output layer. RNN takes the sequential input data $(x_t)_{t=1}^T$, where $x_t \in \mathbb{R}^d$, and com-

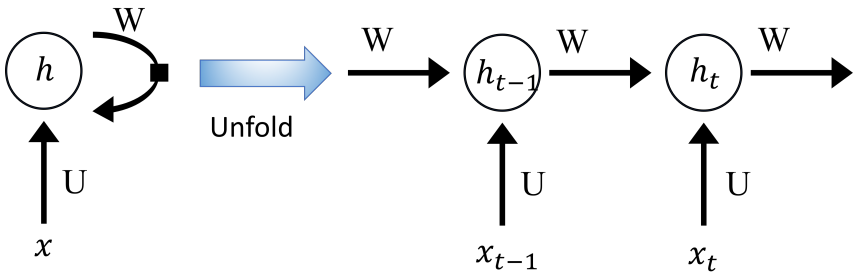


Figure 1.6: The recursive structure of the recurrent neural network.

pute the output $(o_t)_{t=1}^T$, where $o_t \in \mathbb{R}^e$ using Equation (1.10):

$$h_t = \sigma(Ux_t + Wh_{t-1}), o_t = q(Vh_t), \tag{1.10}$$

where $(h_t)_{t=1}^T$ is the hidden layer output with $h_t \in \mathbb{R}^h$ and $U \in \mathbb{R}^{h\times d}$, $W \in \mathbb{R}^{h\times h}$ and $V \in \mathbb{R}^{e\times h}$ are model parameters, and σ and q are two activation functions in the hidden layer and output layer respectively. We denote the RNN model as $R((x_t)_t)$. When training the RNN, the prediction is $\hat{y}_i := R((x_t)_t)$ and $\Theta := \{U, W, V\}$ is the parameter set.

Since neural network is a data-driven technique, there are many factors to consider when designing neural networks to deal with different learning tasks. The activation and loss functions play important role in designing neural networks. We introduce some options of activation and loss functions that would be used in our learning tasks.

Activation Function The choice of activation functions is always essential in neural network models design. The activation function σ could not only add non-linearity to the neural network, but also some constraints such as positivity such that the outputs could approximate the actual data well. Let $x = (x_1, \ldots, x_d) \in \mathbb{R}^d$. The common

options of scalar activation functions, which act element-wisely on the input, i.e.

$$\sigma((x_1,\cdots,x_d)) = (\sigma(x_1),\cdots,\sigma(x_d)),\ \ \forall x \in \mathbb{R}^d,$$

are:

- **Sigmoid function**: The sigmoid function is an approximation of the step function which is a 'S'-shaped curve, the form of which is

  $$\sigma(x) = \frac{1}{1+e^{-x}}.$$

  The sigmoid function is not zero-centered with range from 0 to 1.

- **Hyperbolic tangent function**: The hyperbolic tangent function is a generalization of the sigmoid function

  $$\sigma(x) = \frac{e^{2x}-1}{e^{2x}+1},\ \ \forall x \in \mathbb{R}^d.$$

  Its range is from -1 to 1, which is zero-centered.

- **ReLU**: The ReLU function is the most common activation function at present, which has the form

  $$\sigma(x) = \max(0,x),\ \ \forall x \in \mathbb{R}^d.$$

There are also vector valued activation function such as:

- **Softmax function**: The softmax function has the following form:

  $$\sigma(x) = (\frac{e^{x_1}}{e^{x_1}+\cdots+e^{x_d}},\ldots,\frac{e^{x_d}}{e^{x_1}+\cdots+e^{x_d}}),\ \ \forall x \in \mathbb{R}^d.$$

  Each element of the output vector is non-negative and the summation of all

elements is equal to 1. Thus it is often used to model the probability distribution as the output layer activation function.

Loss Function When training the proposed models, there are different choices of loss functions to fulfill different objectives. In our work, we deal with regression and classification problems. Suppose there are N samples in the dataset such that the dataset $\mathscr{D} = \{(x_i, y_i)\}_{i=1}^{N}$, where $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}^e$ for $i = 1, \ldots, N$. For the regression problem, y_i is real valued. For the classification problem, suppose there are K classes, then y_i is categorical vector, which is a 1-of-K representation, i.e. for a class 2 label $y_i = (0, 1, 0, \ldots, 0)$. The parameter set for FFNN is $\Theta := \{W_l, b_l\}_{l=1}^{L}$, while for RNN is $\Theta := \{U, V, W\}$. And the prediction of FFNN is $\hat{y}_i := h_L(x)$, while of RNN is $\hat{y}_i := R((x_t)_t)$.

In the regression problem, when optimizing the model, the loss function we use is mean squared error (MSE) defined as

$$\mathscr{L}(\Theta|\mathscr{D}) = \frac{1}{N}\sum_{i=1}^{N} |\hat{y}_i - y_i|^2. \tag{1.11}$$

In the multi-classification problem, let y_i be the actual label of sample i. Then the output $\hat{y}_i$ is a probability distribution describing the probability of each class the current sample belongs to. The commonly used loss function is cross-entropy (CE) defined as

$$\mathscr{L}(\Theta|\mathscr{D}) = -\sum_{i=1}^{N} y_i \log(\hat{y}_i). \tag{1.12}$$

When dealing with large and complicated data, the neural network models are always complex as well. They may include different types of neural networks into one.

1.2.3 Generative adversarial network

Generative adversarial network (GAN) introduced by [29] is a unsupervised learning technique on synthetic data generation using neural networks. The model contains two parts, a generator and a discriminator. And both can be represented by neural networks. The training process alternates between the generator and the discriminator, and aims to solve a minmax problem. To define it mathematically, let $(\Omega, \mathscr{F}, \mathbf{P})$ be a probability space, under which μ and ν are two distributions induced by $\mathscr{X}$-valued space. Let $\mathscr{Z}$ denote a latent space and $Z \in \mathscr{Z}$ denote a variable with the known distribution $\mu_Z \in \mathscr{P}(\mathscr{Z})$. Let $\mu \in \mathscr{P}(\mathscr{X})$ denote a target distribution of observed data. The GAN is composed of the generator G and the discriminator D. The generator $G_\theta : \mathscr{Z} \to \mathscr{X}$ is a parameterized map transporting the latent distribution μ_Z to the model distribution ν, i.e. the distribution induced by $G_{\theta_g}(Z)$, where $\theta_g \in \Theta^g$ is a parameter set. To discriminate between real and synthetic data, we also define the discriminator as $D_{\theta_d}(X)$ which maps from $\mathscr{X}$ to $\mathbb{R}$ with parameter $\theta_d \in \Theta^d$. $D(X)$ is the probability that x is from the real data rather than the generated data. Then we train the discriminator D to maximize the probability of assigning the correct labels to both training data and samples generated by G, and train G to minimize $\log(1 - D(G(Z)))$, which leads to the following minmax problem

$$\min_G \max_D \mathbb{E}_{X\sim\mu}[\log D(X)] + \mathbb{E}_{Z\sim\mu_Z}[\log(1 - D(G(Z)))]. \tag{1.13}$$

The min-max problem is solved by iterating gradient descent-ascent algorithms. In practice, for one step optimization of G, the discriminator D is optimitized for k steps such that it is maintained around the optimum as long as G improves slowly. The GAN proposed above has deficiencies such as the balance between training the discriminator and the generator needs to be carefully maintained, and collapse mode. Based on the GAN architecture, researchers [30] proposed model called Wasserstein GAN to tackle the above problems.

Chapter 2

Log-signature recurrent neural network (Logsig-RNN)

In this chapter, we propose a novel and generic network module, i.e. the so-called Logsig-RNN model by combing the log-signature with classcial RNNs. The proposed model, as an enhancement of the RNN model, exploits the mathematically principle representations of the log-signature to manage high-frequency data and hence reduce the time dimension of the RNNs to improve accuracy and robustness. The outline of this chapter is summarized as follows. In Section 2.1, we briefly introduce the background of the proposed Logsig-RNN model and describe how the numerical approximation of SDEs motivates the formulation of our methods. Then we state the advantages of our model and its related works. In Section 2.2, we mathematically define the Logsig-RNN network and prove its universality. In Section 2.3, we then extend it to a generalized sequence-to-sequence model and also prove the corresponding universality theorem.

2.1 Introduction

2.1.1 Background and motivation

Our work is designed to provide an effective non-parametric modelling of time series data in the machine learning field. Time series data is usually regarded as a sequence of observations sampled from a certain underlying process, and SDEs can be used as parametric modelling of that process. Most parametric SDE models are only vague approximations of the real world phenomena. They are usually formulated based on limited observations and statistical analysis of the time series. Model complexity is restricted due to calibration and computational concerns. With the development of machine learning techniques, deep learning methods, especially RNNs, are widely used in time series modelling. The RNN models are non-parametric and data-driven. They allow much freedom in the model complexity and achieve remarkable results in various fields such as natural language processing [31], financial forecasting [32], human action recognition [33, 34, 35] and time series generation [36, 37]. However, classical RNNs suffer from gradient vanishing/explosion and long temporal dependency issues when dealing with high-frequency data. To achieve the best of both worlds, we generalize the SDEs to the form (2) as a non-parametric model and integrate it with RNNs to develop the Logsig-RNN module. Then it enlightens us to investigate the connections between the numerical approximation of solutions to the SDEs and machine learning models. Specifically, the numerical approximation method reveals a strong relationship with rough path theory and RNNs, which motivates us to build up a hybrid neural network model leveraging the benefits of the two components.

To illustrate the main idea behind the proposed Logsig-RNN model, we first briefly recall the numerical approximation method for SDEs. Let $X := (X_t)_{t\in[0,T]}$ and $Y := (Y_t)_{t\in[0,T]}$ denote two stochastic processes respectively, which satisfy the

following differential equation

$$dY_t = f(Y_t)dX_t, Y_0 = y_0. \tag{2.1}$$

Let $\mathscr{D} := \{u_0, \ldots, u_N | u_0 = 0, u_N = T\}$ be a time partition of $[0,T]$. We approximate Y evaluated at the time partition $\mathscr{D}$ using the k-step Taylor approximation estimator, denoted by $(\hat{Y}_u)_{u\in\mathscr{D}}$ and defined in an inductive way, i.e., $\hat{Y}_{u_0} = y_0$, and $\forall i \in \{0, \cdots, N-1\}$,

$$\begin{aligned} \hat{Y}_{u_{k+1}} &= \hat{Y}_{u_k} + \sum_{m=1}^{M} f^{\circ m}(\hat{Y}_{t_i}) \int\limits_{u_k<s_1<\cdots<s_m<u_{k+1}} dX_{s_1} \otimes \cdots \otimes dX_{s_m}{}^{1}, \\ &=: g(\hat{Y}_{u_k}, lS_M(X_{u_k,u_{k+1}})), \end{aligned} \tag{2.2}$$

where $g : W \times \mathscr{L} \to W; g(y,l) = y + \sum_{k=1}^{M} f^{\circ m}(y)\exp(l)$.

Equation (2.2) shows that the estimator $\hat{Y}_{u_{k+1}}$ depends on the estimator at the previous step $(\hat{Y}_{u_k})$ with an additional dependency on the log-signature $(lS_M(X_{u_k,u_{k+1}}))$. This recursive structure of $\left(\hat{Y}_{u_k}\right)_{k=0}^{N}$ resembles that of the RNNs. Roughly speaking, $\left(\hat{Y}_{u_k}\right)_{k=0}^{N}$ plays a role similar to the hidden neurons of a RNN model with the log-signature as an input (see Figure 2.1).

Inspired by Equation (2.2), we propose a novel and non-parametric model, so-called the Logsig-RNN, which significantly generalize parametric and conventional SDE models to allow universality by parameterizing the vector field g via neural networks. The resulting model can be viewed as the combination of the Log-Signature and the RNN layer (see Figure 2.2 (Right) for the architecture of the Logsig-RNN model). On the one hand, the former one can effectively summarize the high frequency stream data locally, and hence reduce the time dimension of input stream data. On the other hand, the temporal dependence between the segments of stream

[1] $f^{\circ m}$ is given recursively by $f^{\circ 1} = f; f^{\circ k+1} = D(f^{\circ k})f$ and $D(f)$ denotes the derivative of the function f.

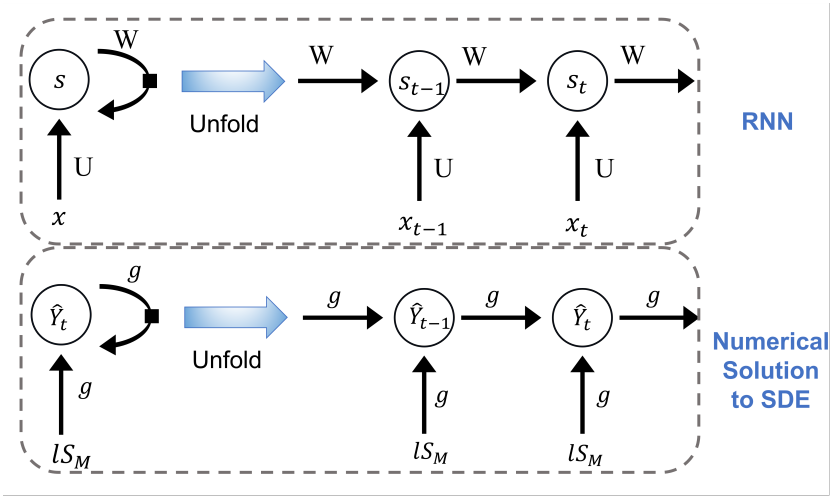


Figure 2.1: The shared recursive structure of numerical approximation of the solution $\hat{Y}_t$ and the RNN $\mathscr{R}_\sigma$.

data can be well captured by the RNN-type neural networks.

The proposed Logsig-RNN model is an enhancement of the RNN model, and can be used as a generic neural network module to be pluggable into neural network architectures just as RNN-type models. However, in comparison to the RNNs, our approach has advantages in dealing with time series of irregular sampling frequency and with missing data case. The RNNs are usually regarded as a discrete approximation to a certain process [9, 38], nevertheless, the discretization breaks down in this case as the samples of streamed data require varied and non-uniform timestamps. Our network, however, can provide an efficient and robust descriptor for such irregularly spaced sequential data thanks to the log-signature features. Moreover, our method is particularly effective for high-frequency data streams. The large time dimension of high frequency data usually leads to the curse of dimensionality and gradient vanishing/explosion issues of the RNNs. On the contrary, the Log-Signature Layer of our model can successfully alleviate these issues and reduce the time dimension of the input of RNNs by transforming the original data into a sequence of log signature features over a potentially much coarser time partition. To

enable the application of RNNs to high-frequency data, one has to downsample raw data, which may potentially result in information loss. In contrast to it, the Log-Signature Layer can avoid this issue by summarising high frequency information using the principled log-signature features (Figure 2.2).

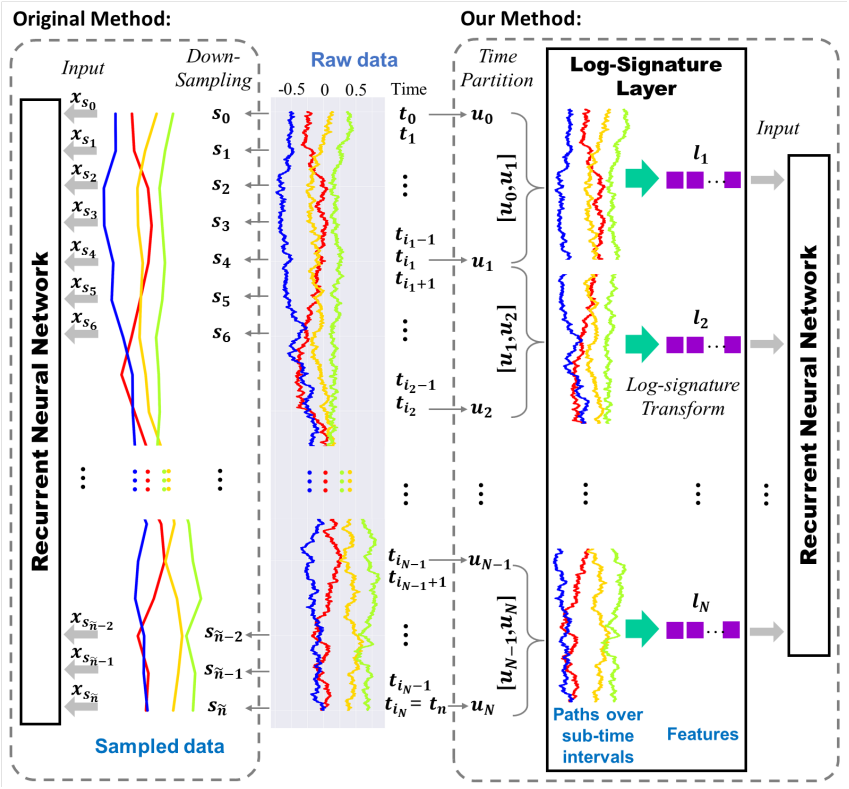


Figure 2.2: Comparison of Logsig-RNN and RNN.

The regular Logsig-RNN model takes sequential data as input and outputs a static variable. To apply it in broader applications, we can extend it to a sequence-to-sequence module called the generalized Logsig-RNN which can produce streamed data with any target length, and even be able to model continuous time series universally. We validate the effectiveness of our approach in two major types of machine learning tasks: (1) supervised learning problems where the input is sequential and output is static, for which we use the regular Logsig-RNN model; (2) generative tasks for time series, where we use the generalized model to produce sequential

outputs. Numerical results (Chapter 3, 4) demonstrate the Logsig-RNN leads to higher accuracy and efficiency compared with standard RNN-typed model.

2.1.2 Related works

Learning SDEs. SDEs of the form (2) are useful tools for modelling random phenomena and provide a general class of functionals on path space. Statistical inference for SDEs has a rich literature due to the importance of research outcomes and applications (see [39] for the survey and overview). On one side, some research focuses on the parameter estimation of (model-specific) stochastic processes; for example, in [40] the authors use statistical variational inference to estimate the solutions of SDEs; [41] estimates parameters for a general stochastic process by matching the expected signature of the solution process. In contrast, our approach is non-parametric and is used to learn the solution map without any assumption on the distribution of the stochastic process. On the other side, researchers develop neural networks based on SDEs to model unparametrized data. By considering controlled differential equations (CDEs) and combining them with Neural ODE techniques, [8] develops a memory-efficient method that can incorporate incoming information to deal with irregularly sampled or partially observed data. While Neural CDE has the same severe limitations on high-frequency data as RNNs, [42] is motivated by the log-ODE method and applies rough path theory to extend it to Neural rough differential equation (RDE), which represents long time series data by log-signature features. By comparison, as our work is free of implementing integration and numerically solving ODE systems, it is less time consuming in computation.

Time series modelling. In [24] Levin et al. firstly proposed the signature of a path as the basis functions for a functional on the un-parameterized path space and suggested the linear model on the signature feature (Sig-OLR) as the first non-parametric model for time series modelling. However, Sig-OLR has the limitation of inefficient global approximation due to the instability of the polynomial extrap-

olation. Despite the successful empirical applications of the signature feature sets ([12], [13], [14]), the theoretical question on which learning algorithms are most appropriate to be combined with the (log)-signature feature remains open. Our work is devoted to answering this question with both theoretical justification and promising numerical evidence.

Functional Data Analysis. Learning a functional on streamed data falls under the category of functional data analysis (FDA) [43], which models data using functions or functional parameters and analyses data providing information about curves, surfaces or anything else varying over a continuum. The representation theory of the functional on functions plays an important role in FDA study. Functional principal components analysis is one of the main techniques of FDA to represent the function data[44], which express the function data as the linear coefficients of the basis functions (usually without taking into account the response variable corresponding to the function input). In contrast to it, albeit taking the functional view of sequential data, our approach focuses on the representation of the path in terms of its effect (functional on the path, i.e. the solution of the controlled differential equation driven by the path).

2.2 Logsig-RNN

2.2.1 Model

In this section, we introduce the mathematical definition of the Logsig-RNN model. We first define the Log-Signature (Sequence) Layer, which transforms an input time series to a sequence of log-signatures of the input over a coarser time partition. Let $J := [0,T]$ be a compact interval. Consider a discrete d-dimensional time series $x^{\hat{\mathscr{D}}} = (x_{t_i})_{i=1}^{n}$ over the time interval J. We can first embed it into path space with bounded variation $V_1(J,\mathbb{R}^d)$ by linear interpolation and denote the continuous path by x. Let $\mathscr{D} = (u_k)_{k=0}^{N} \subset \hat{\mathscr{D}}$ be a coarser time partitions of J.

Definition 2.1 (Log-Signature (Sequence) Layer)**.** Given the time partition $\mathscr{D}$, a Log-Signature Layer of degree M is a mapping from $V_1(J,\mathbb{R}^d)$ to $\mathbb{R}^{d_{ls}\times N}$, which computes $(l_j)_{j=1}^N$ as an output for any x, where l_j is the truncated log-signature of x over the time interval $[u_{k-1},u_k]$ of degree M as follows:

$$l_k = lS_M(x)_{[u_{k-1},u_k]}, \tag{2.3}$$

where $k \in \{1,\cdots N\}$, lS_M is the truncated log-signature of level M defined in Definition 1.15 and d_{ls} is the dimension of the truncated log-signature.

Note that the Log-Signature Layer is a deterministic transformation and it does not have any trainable weights. In addition, the input dimension of the Log-Signature Layer is (d,n) and the output dimension is (N,d_{ls}) where $N \leq n$ and $d_{ls} \geq d$. It means that the Log-Signature Layer potentially shrinks the time dimension effectively by using the more informative spatial features of a higher dimension.

As illustrated in Section 2.1, the numerical approximation scheme (2.2) shares a common recursive structure with the RNN network, which inspires us to construct the following Logsig-RNN model by incorporating the Log-Signature Layer with RNNs.

Model 2.1 (Logsig-RNN Network). Given $\mathscr{D} := (u_k)_{k=0}^N$, a Logsig-RNN network computes a mapping $H := H_{M,\Theta}^{\mathscr{D}}$ from an input path $x \in V_1(J,\mathbb{R}^d)$ to an output defined as follows:

- Compute $(l_k)_{k=0}^{N-1}$ as the output of the Log-Signature Layer of degree M for an input x by Definition 2.1.
- The output layer is computed by $\mathscr{R}_\sigma((l_k)_{k=0}^{N-1}|\Theta)$, where $\mathscr{R}_\sigma$ is a RNN network with certain activation function σ.

We denote the family of all possible Logsig-RNN models (Model 2.1) by

$$\mathscr{H} := \bigcup_{\mathscr{D}} \bigcup_{M=1}^{\infty} \{H_M^{\mathscr{D}}\}. \tag{2.4}$$

Remark 2.1 (Link between RNN model and Logsig-RNN model). For $M = 1$, the Logsig-RNN network is reduced to the RNN model with $(x_{u_{k+1}} - x_{u_k})_{k=1}^N$ as an input. When $\mathscr{D}$ coincides with $\hat{\mathscr{D}}$, the Logsig-RNN Model is the RNN model with increments of the raw data input.

Remark 2.2. The sampling time partition of the raw data $\hat{\mathscr{D}}$ can potentially be much higher than $\mathscr{D}$ used in the Logsig-RNN model. The higher frequency of input data would not increase the dimension of the Log-Signature Layer, but it makes the computation of l_k more accurate.

2.2.2 Backpropogation

Let us consider the derivative of a scalar function F on $(l_k)_{k=1}^N$ with respect to the discrete input path $x^{\hat{\mathscr{D}}}$, given the derivatives of F with respect to $(l_k)_{k=1}^N$. By the Chain rule, it holds that

$$\frac{\partial F(l_1, \cdots, l_N)}{\partial x_{t_i}} = \sum_{k=1}^{N} \frac{\partial F(l_1, \cdots, l_N)}{\partial l_k} \frac{\partial l_k}{\partial x_{t_i}}. \tag{2.5}$$

where $k \in \{1, \cdots, N\}$ and $i \in \{0, 1, \cdots, n\}$.

If $t_i \notin [u_{k-1}, u_k]$, $\frac{\partial l_k}{\partial x_{t_i}} = 0$; otherwise $\frac{\partial l_k}{\partial x_{t_i}}$ is the derivative of the single log-signature l_k with respect to the path x_{u_{k-1},u_k}. The log signature $lS(x^{\hat{\mathscr{D}}})$ with respect to x_{t_i} is proved differentiable and the algorithm of computing the derivatives is given in [45], denoted by $\nabla_{x_{t_i}} lS(x^{\hat{\mathscr{D}}})$. This is a special case for our log-signature layer when $N = 1$. In general, for any $N \in \mathbb{Z}^+$, it holds that $\forall i \in \{0, 1, \cdots, n\}$ and $k \in \{1, \cdots, N\}$,

$$\frac{\partial l_k}{\partial x_{t_i}} = \mathbf{1}_{t_i \in [u_{k-1}, u_k]} \nabla_{x_{t_i}} lS(x_{u_{k-1},u_k}), \tag{2.6}$$

Thus the backpropogation algorithm of the Log-Signature Layer can be implemented using Equation (2.5) and (2.6). [2]

2.2.3 Complexity analysis

In this section, we investigate the computational complexity of the Logsig-RNN (Model 2.1) and compare it with the classical RNN. Consider the input to be a discrete d-dimensional time series $x = (x_{t_i})_{i=1}^{n}$ over the time interval J and the output $o_T \in \mathbb{R}^e$. Although the Logsig-RNN is combined of the Log-Signature Layer and a RNN, to highlight the ability of time dimension reduction of our model and simplify the analysis, we consider the Log-Signature Layer as the preprocessing layer for features extraction[3] and focus on the computational cost of the RNN layer. Suppose the Log-Signature Layer splits x into N segments and compute the truncated log-signature of degree M. It is noted that as preprocessing, the Log-Signature Layer is applied only once before the whole training process. The RNN layer dominates the computational complexity given large enough training epochs. For the Logsig-RNN, the input to the RNN is the log-signatures $(l_k)_{k=1}^{N}$ of dimension (N, d_{ls}). We consider the RNN:

$$h_k = \sigma(Ul_k + Wh_{k-1}), o_k = q(Vh_k),$$

where $(h_k)_{k=1}^{N}$ is the hidden layer output with $h_k \in \mathbb{R}^h$ and $U \in \mathbb{R}^{h \times d_{ls}}$, $W \in \mathbb{R}^{h \times h}$ and $V \in \mathbb{R}^{e \times h}$ are model parameters, and σ and q are two activation functions. We only compute the output from the hidden layer once at the terminal time T for the Logsig-RNN model. At each step, the complexities for computing Ul_k and Wh_{k-1} are hd_{ls} and h^2 respectively. The complexity of element-wisely applying

[2] In the signatory python package [46], the backpropagation part is embedded in pytorch, so that no function required when coding; In the iisignature python package [18], logsigbackprop(deriv, path, s, Method = None) returns the derivatives of some scalar function F with respect to path, given the derivatives of F with respect to logsig(path, s, methods). Our implementation of the back-propogation algorithm of the Log-Signature Layer uses logigbackprop() provided in iisignature.

[3] The computational complexities of signature/log-signature are discussed in Appendix A.

σ is counted as h. At the last step, we compute $q(Vh_k)$ which has the complexity $eh+e$. Thus, it leads to the complexity of the forward direction of RNN layer as follows

$$F_1 := F_1(d,e,h,N,M) = N(hd_{ls}+h^2+h)+eh+e.$$

As we only consider the output of the RNN at terminal time, the backward complexity is linear w.r.t to the length of the input as follows

$$B_1 := B_1(d,e,h,N,M) = N(eh^2+2eh+2h^2+hd_{ls}+h+d_{ls})+eh$$

Finally, the computation complexity of the Logsig-RNN is

$$E_1 := F_1+B_1 = N(eh^2+2eh+3h^2+2hd_{ls}+2h+d_{ls})+2eh+e. \tag{2.7}$$

Similarly, the forward complexity of the classical RNN is

$$F_2 := F_2(d,e,h,n) = n(hd+h^2+h)+eh+e.$$

The backward complexity of the classical RNN is

$$B_2 := B_2(d,e,h,n) = n(eh^2+2eh+2h^2+hd+h+d)+eh.$$

Then the computation complexity of the classical RNN is

$$E_2 := F_2+B_2 = n(eh^2+2eh+3h^2+2hd+2h+d)+2eh+e. \tag{2.8}$$

Next, we assume $n > N$ such that the Logsig-RNN model plays a role of time reduction, which decreases the original length n of input x to the length N of the output

of the Log-Signature Layer; with this assumption, we show that if the following additional condition is satisfied

$$\frac{d_{ls}}{d} \leq \frac{n}{N}, \tag{2.9}$$

the bound $E_1 \leq E_2$ holds such that the Logsig-RNN has an advantage on the computational complexity against the classical RNN. Then

$$E_2 \geq N(eh^2 + 2eh + 3h^2 + 2h) + n(2hd + d) + 2eh + e.$$

Thus, we see that it suffices to show

$$N(2hd_{ls} + d_{ls}) \leq n(2hd + d),$$

which implies (2.9). In practice, the inequality (2.9) is often fulfilled since we can choose small degree M such that d_{ls} is not too large, and small number of segments N such that the time dimension is reduced as much as possible.

2.2.4 Universality theorem

In this section, we prove the universality theorem of the Logsig-RNN (Model 2.1) to approximate the solution to any controlled differential equation at terminal time T under mild conditions. Let $E := \mathbb{R}^d$ and $F := \mathbb{R}^e$ be two real vector spaces. Let $L(E,F)$ denote the continuous linear mapping from E to F. Let $Y : J \to F$ be a solution of the given SDE of the form (2.1), i.e.

$$dY_t = f(Y_t)dX_t, Y_0 = y_0,$$

and $f : F \to L(E,F)$ is the vector field of SDE. We need the following assumption for the rest of the chapter:

Assumption 1.

(i) $\mathbf{X} = (1, \mathbf{X}^1, \ldots, \mathbf{X}^{\lfloor p \rfloor}) \in G^{(\lfloor p \rfloor)}(E)$ *is a geometric p-rough path controlled by* ω.

(ii) $f \in Lip(\gamma)$, *where* $\gamma > p$.

Theorem 2.1 (Universality of the Logsig-RNN Model)**.** *Under Assumption 1, for any* $\varepsilon > 0$, *there exist a constant* $\delta > 0$ *and parameters* Θ *such that for any* $M > \lfloor p \rfloor$, *there exists a model* $H \in \mathscr{H}$ *with* $\Delta\mathscr{D} \leq \delta$, *whose output* $o_T := \mathscr{R}_\sigma((l_k)_{k=1}^N | \Theta)$[4] *satisfies*

$$\sup_{\mathbf{X} \in K} ||Y_T - o_T|| \leq \varepsilon, \tag{2.10}$$

where K is any compact subset of $G^{(\lfloor p \rfloor)}(E)$.

To derive the error bound in (2.10), we note that the triangle inequality implies the following

$$||Y_T - o_T|| \leq \underbrace{||Y_T - \hat{Y}_T||}_{E_1} + \underbrace{||\hat{Y}_T - o_T||}_{E_2}. \tag{2.11}$$

where $\hat{Y}_T$ is the numerical approximated solution at time T given by (2.2). Thus, to prove the universality theorem 2.1, it suffices to establish the error bounds for E_1 and E_2 first. The next global approximation theorem states that the numerical solution $\hat{Y}_T$ can approximate Y_T sufficiently well, which implies E_1 can be controlled.

Theorem 2.2 (Global Approximation Theorem)**.** *Under Assumption 1, let* $\hat{Y}_{u_N}$ *be defined in* (2.2). *For any* $\varepsilon > 0$, *there exists* $\delta > 0$ *such that when* $\Delta\mathscr{D} \leq \delta$ *and* $M \geq \lfloor \gamma \rfloor$, *it satisfies that*

$$||Y_T - \hat{Y}_{u_N}|| \leq \varepsilon, \tag{2.12}$$

[4] l_k is the log-signature of $\mathbf{X}$ of the time interval $[u_{k-1}, u_k]$.

More specifically, it holds that

$$||Y_T - \hat{Y}_{u_N}|| \leq \bar{C} \left[\max_{u_i \in \mathscr{D}} \omega(u_{i-1}, u_i) \right]^{\frac{\lfloor \gamma \rfloor + 1}{p} - 1}, \tag{2.13}$$

where

$$\bar{C} = C\beta_1^{\lfloor \gamma \rfloor + 1 - p} ||f||_{\circ\gamma;J}^{\lfloor \gamma \rfloor + 1} C_2^p. \tag{2.14}$$

To prove Theorem 2.2, we introduce the following auxiliary theorem, which establishes an error bound of numerical approximated solution in terms of the p-variation of $\mathbf{X}$ as a classical result [47].

Theorem 2.3. *[47] Under Assumption 1, there exists $C := C(p, \gamma)$ such that*

$$||Y_T - \hat{Y}_{u_N}|| \leq C \sum_{k=1}^{N} |f|_{\circ\gamma;J}^{\lfloor \gamma \rfloor + 1} ||\mathbf{X}||_{p-var;[u_{k-1}, u_k]}^{\lfloor \gamma \rfloor + 1}. \tag{2.15}$$

Now we are ready to proceed with the proof of Theorem 2.2.

Proof of Theorem 2.2. According to Theorem 2.3, there exists $C := C(\rho, \gamma)$ such that

$$||Y_T - \hat{Y}_{u_N}|| \leq C \sum_{k=1}^{N} |f|_{\circ\gamma;J}^{\lfloor \gamma \rfloor + 1} ||\mathbf{X}||_{p-var;[u_{k-1}, u_k]}^{\lfloor \gamma \rfloor + 1}. \tag{2.16}$$

As $\mathbf{X}$ has finite p-variation controlled by ω, by Theorem 1.14, there exists some constants $\beta_1 := \beta_1(p, \gamma) > 0$ and $C_2 > 0$ such that for any $(s,t) \in \Delta_T$,

$$||\mathbf{X}||_{p-var;(s,t)} \leq \beta_1 \omega(s,t)^{\frac{1}{p}} \leq C_2. \tag{2.17}$$

The inequalities (2.16) and (2.17) implies that

$$||Y_T - \hat{Y}_{u_N}|| \leq C\beta_1^{\lfloor\gamma\rfloor+1}||f||_{\circ\gamma;J}^{\lfloor\gamma\rfloor+1}\sum_{i=1}^{N}\omega(u_{i-1},u_i)^{\frac{\lfloor\gamma\rfloor+1}{p}}$$

Then by the super-additivity of the control ω, we can derive the following inequality

$$\begin{aligned} ||Y_T - \hat{Y}_{u_N}|| &\leq C\beta_1^{\lfloor\gamma\rfloor+1}||f||_{\circ\gamma;J}^{\lfloor\gamma\rfloor+1}\omega(J)\left[\max_i \omega(u_{i-1},u_i)\right]^{\frac{\lfloor\gamma\rfloor+1}{p}-1} \\ &\leq \bar{C}\left[\max_i \omega(u_{i-1},u_i)\right]^{\frac{\lfloor\gamma\rfloor+1}{p}-1} \end{aligned} \tag{2.18}$$

where

$$\bar{C} = C\beta_1^{\lfloor\gamma\rfloor+1-p}||f||_{\circ\gamma;J}^{\lfloor\gamma\rfloor+1}C_2^p.$$

Thus by the uniform continuity of ω, for any $\varepsilon > 0$, there exists $\delta_2 > 0$, such that the right hand side of (2.18) can be smaller than ε uniformly over all the partitions with mesh size smaller than δ_2. □

Next, we control the error E_2 between the numerical approximated solution $\hat{Y}_T$ and the output o_T of the Logsig-RNN in the inequality (2.11). As shown in Figure 2.1, it is remarkable that there is a common recursive structure between the RNN $\mathscr{R}_\sigma$ and the one defined by the numerical Taylor approximation to solutions of SDEs $\hat{Y}_{u_N}$. Thus, we define a recursive function $G_{\tilde{f},k}$ applied to above two structures. Specifically, for any given function $\tilde{f}: \mathbb{R}^{d+e} \to \mathbb{R}^e$, define $G_{\tilde{f},k}: \mathbb{R}^{k\times d} \to \mathbb{R}^{k\times e}$ as follows:

$$G_{\tilde{f},k}: (x_1,\cdots,x_k) \mapsto (r_1,\cdots,r_k), \tag{2.19}$$

where $r_{t+1} = \tilde{f}(x_{t+1},r_t), \forall t \in \{1,\cdots,k-1\}$, and r_1 is the fixed initial value.

On the one hand, when $\tilde{f}(x,r) := A\sigma(\theta_1 x + \theta_2 r)$, where $A \in \mathbb{R}^{e\times m}$, $\theta_1 \in \mathbb{R}^{m\times d}$ and $\theta_2 \in \mathbb{R}^{m\times e}$, then $G_{\tilde{f},N}$ is the RNN equipped with the activation function σ, denoted by $\mathscr{R}(\cdot|\Theta)$; on the other hand, the numerical solution to SDE $\hat{Y}_{u_N}$ is $G_{g,N}$. Therefore controlling the error E_2 is equal to controlling the norm of the difference between $G_{A\sigma(\theta_1 x+\theta_2 r),N}$ and $G_{g,N}$, i.e.

$$\hat{Y}_{u_N} - \mathscr{R}_\sigma((l_k)_{k=1}^N) = \left(G_{g,N} - G_{A\sigma(\theta_1 x+\theta_2 r),N}\right)((l_k)_{k=1}^N). \tag{2.20}$$

Then we can prove the Theorem 2.1

Proof of Theorem 2.1. Theorem 2.2 implies that the error E_1 in the triangle inequality (2.11) can be arbitrarily small given large enough degree of the log-signature M and small enough $\Delta\mathscr{D}$.

Lemma B.3 demonstrates the continuity of the map $\tilde{f} \mapsto G_{\tilde{f},N}$, which leads to the following inequality given Equation (2.20)

$$\hat{Y}_{u_N} - \mathscr{R}_\sigma((l_k)_{k=1}^N) \leq \bar{C}_N ||g - \tilde{f}||_{\infty,K}$$

where $\bar{C}_N$ is defined in (B.11). Lemma B.2 ensures that for any $\varepsilon > 0$, there exist integer $m > 0$, $A \in \mathbb{R}^{e\times m}$, $\theta_1 \in \mathbb{R}^{m\times d}$ and $\theta_2 \in \mathbb{R}^{m\times e}$ such that $||g - \tilde{f}||_{\infty,K} \leq \varepsilon$, which means E_2 can be arbitrarily small. □

2.3 Generalized Logsig-RNN

2.3.1 Model

In this section, we build up the sequence-to-sequence generalized Logsig-RNN model, which is capable of producing streamed outputs of any target length. First we introduce two time partitions of the interval J. Assume $\mathscr{D} = (u_k)_{k=0}^{N_1}$ to be a time partition. Then we define another finer time discretization of J, denoted as $\mathscr{D}_Y = (t_j)_{j=0}^{N_2}$ of J, such that $\mathscr{D}_Y \supset \mathscr{D}$, $t_0 = u_0$ and $t_{N_2} = u_{N_1} = T$. The generalized

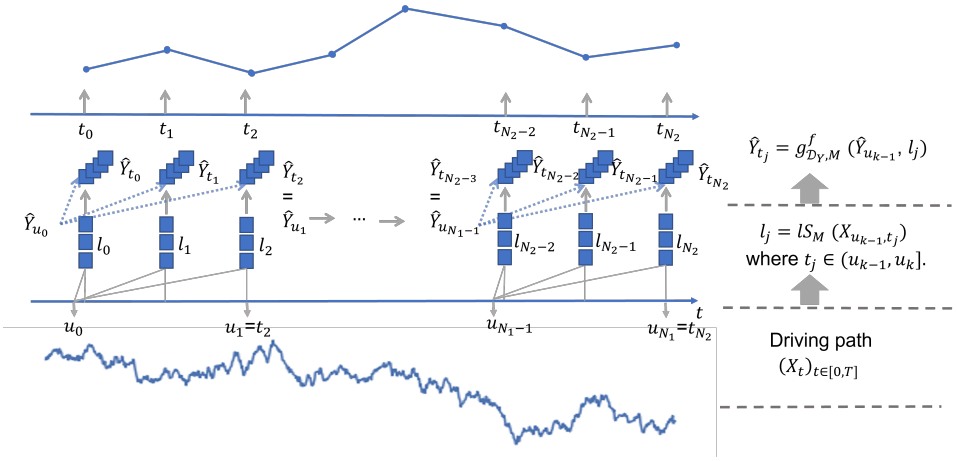


Figure 2.3: Visualization of the numerical method in Eqn. (2.23).

Logsig-RNN outputs data at the time steps of $\mathscr{D}_Y$. The idea is that when estimating the solution Y_{t_j} at $t_j \in (u_{k-1}, u_k]$, the local approximation scheme is based on the approximated value at u_{k-1} but not at t_{j-1} (Figure 2.3) in avoid of long-term dependency issue. In this way, the approximation error can still be controlled by the partition $\mathscr{D}$, and the output sequences can have arbitrary length. Similar as before, we need to define a generalized Log-Signature (Sequence) Layer, which transforms the input path $x \in V_1(J, \mathbb{R}^d)$ to a sequence of the log signature of x over the time partition $\mathscr{D}_Y$.

Definition 2.2 (Generalized Log-Signature (Sequence) Layer)**.** Given $\mathscr{D}$, $\mathscr{D}_Y$, a Log-Signature Layer of degree M is a mapping from $V_1(J, \mathbb{R}^d)$ to $\mathbb{R}^{d_{ls}\times N_2}$, which computes $(l_j)_{j=1}^{N_2}$ as an output for any x, where l_j is the truncated log signature of x over the time interval $[u_{k-1}, t_j]$ of degree M with $t_j \in (u_{k-1}, u_k]$ as follows:

$$l_j = lS_M(x)_{[u_{k-1},t_j]}, \tag{2.21}$$

where $j \in \{1, \cdots N_2\}$ and d_{ls} is the dimension of the truncated log-signature. At the

time steps belong to both $\mathscr{D}$ and $\mathscr{D}_Y$, we also define l^k as follows:

$$l^k = l_j, \text{ when } u_k = t_j \tag{2.22}$$

for $i \in \{1, \cdots N_1\}$ and $j \in \{1, \cdots N_2\}$.

Then similarly as the numerical scheme in (2.2), for any $t_j \in \mathscr{D}_Y$ find k such that $t_j \in (u_{k-1}, u_k]$, we apply the Taylor approximation of Y_{t_j} around the reference time point $t = u_{k-1} \in \mathscr{D}$ to obtain

$$\begin{aligned} \hat{Y}_0 &= y_0, \\ \hat{Y}_{t_j} &= \hat{Y}_{u_{k-1}} + \sum_{m=1}^{M} f^{\circ m}(\hat{Y}_{u_{k-1}}) \int_{u_{k-1}<s_1<\cdots<s_m<t_j} dX_{s_1} \otimes \cdots \otimes dX_{s_m} \\ &:= g(\hat{Y}_{u_{k-1}}, l_j). \end{aligned} \tag{2.23}$$

where M is the degree of log-signature and any model l_j is defined in Equation (2.21). The method is visualized in Figure 2.3. It is noted that to approximate the solution Y_{t_j}, the method approximates $\hat{Y}_{u_0}, \ldots, \hat{Y}_{u_{k-1}}, \hat{Y}_{t_j}$ in order. It implies that the above method estimate every Y_{t_j} in a recursive way the same as that of (2.2).

Thus, we can develop the generalized Logsig-RNN:

Model 2.2 (Generalized Logsig-RNN Network). Given $\mathscr{D} = (u_k)_{k=0}^{N_1} \subset \mathscr{D}_Y = (t_j)_{j=0}^{N_2}$, the generalized Logsig-RNN network $\bar{H} := \bar{H}_{M,\Theta}^{\mathscr{D},\mathscr{D}_Y}$ maps from an input path $x \in V_1(J, \mathbb{R}^d)$ to an output $(o_{t_j})_{j=1}^{N_2} \in \mathbb{R}^{N_2 \times e}$ such that

- For any $t_j \in (u_{k-1}, u_k]$, compute $(l_j)_{j=1}^{N_2}$ as the output of the Log-Signature Layer of degree M for an input x by Definition 2.2.

- The output layer is computed recursively as follows:

$$\begin{aligned} h_{t_j} &= \sigma_1(\theta_1 h_{u_{k-1}} + \theta_2 l_j); \\ o_{t_j} &= \sigma_2(\theta_3 h_{t_j}). \end{aligned} \tag{2.24}$$

  where σ_1, σ_2 are two activation functions and $\Theta = \{\theta_1, \theta_2, \theta_3\}$ is the learnable parameter set.

We denote the family of the generalized Logsig-RNN model defined in Model 2.2 to be

$$\overline{\mathscr{H}} := \bigcup_{\mathscr{D} \subset \mathscr{D}_Y} \bigcup_{M=1}^{\infty} \{\bar{H}_M^{\mathscr{D}, \mathscr{D}_Y}\}. \tag{2.25}$$

Remark 2.3. For $t_j \in (u_{k-1}, u_k]$, if we focus on the time steps $(u_0, \ldots, u_{k-1}, t_j)$, it is observed that the output sequence $(o_{u_0}, \ldots, o_{u_{k-1}}, o_{t_j})$ is derived recursively in the same way as the outputs of RNNs defined in (1.10). Thus, it implies $o_{t_j} = \mathscr{R}_\sigma((l^i)_{i=1}^{k-1}, l_j | \Theta)$, where $\mathscr{R}_\sigma$ is a RNN network with $\Theta = \{\theta_1, \theta_2, \theta_3\}$ and activation function σ.

2.3.2 Backpropogation

Let us consider the derivative of the scalar function F on $(l_j)_{j=1}^{N_2}$ with respect to path $x^{\hat{\mathscr{D}}}$, given the derivatives of F with respect to $(l_j)_{j=1}^{N_2}$. By the Chain rule, it holds that

$$\frac{\partial F(l_1, \cdots, l_{N_2})}{\partial x_{t_i}} = \sum_{k=1}^{N_2} \frac{\partial F(l_1, \cdots, l_{N_2})}{\partial l_j} \frac{\partial l_j}{\partial x_{t_i}}. \tag{2.26}$$

where $j \in \{1, \cdots, N_2\}$ and $i \in \{0, 1, \cdots, n\}$. If $t_i \notin [u_{k-1}, t_j]$, $\frac{\partial l_j}{\partial x_{t_i}} = 0$; otherwise $\frac{\partial l_j}{\partial x_{t_i}}$ is the derivative of the single log-signature l_j with respect to path x_{u_{k-1}, t_j} where $t_i \in \mathscr{D}_Y \cap [u_{k-1}, t_j]$. Then the computation of $\frac{\partial l_j}{\partial x_{t_i}}$ is the same as Equation (2.6).

2.3.3 Complexity analysis

In this section, we investigate the computational complexity of the generalized Logsig-RNN (Model 2.2) and compare it with the classical RNN. Consider the input to be a discrete d-dimensional time series $x = (x_{t_i})_{i=1}^{n}$ over the time interval J. Given two time partitions $\mathscr{D} = (u_k)_{k=0}^{N_1} \subset \mathscr{D}_Y = (t_j)_{j=0}^{N_2}$ for J, the generalized Logsig-RNN splits the input data into N_1 segments with $N_1 < n$ and outputs $(o_{t_j})_{j=1}^{N_2} \in \mathbb{R}^{N_2 \times e}$. Similar as in Section 2.2.3, we consider the generalized Log-Signature Layer as the preprocessing layer. The input of the RNN layer is the sequence of log-signatures $(l_j)_{j=1}^{N_2}$ of dimension (N_2, d_{ls}). At each step, the complexities for computing $\sigma(Ul_k + Wh_{k-1})$ and $q(Vh_k)$ are $hd_{ls} + h^2 + h$ and $eh + e$ respectively, where σ and q are applied element-wisely. Thus, the forward complexity of the RNN layer is

$$F_1 := N_2(hd_{ls} + h^2 + h + eh + e).$$

To compute the complexity of the backpropagation of the RNN layer, we assume that the partition $\mathscr{D}$ evenly splits the partition $\mathscr{D}_Y$ and $N_2 \equiv 0 \pmod{N_1}$, i.e. $\frac{N_2}{N_1}$ is an integer. Then the backward complexity of the RNN layer is

$$\begin{aligned} B_1 &:= \frac{N_2}{N_1}\left(\frac{N_1(N_1+1)}{2}(eh^2 + 2eh + 2h^2 + hd_{ls} + h + d_{ls}) + N_1 eh\right) \\ &= \frac{N_2(N_1+1)}{2}(eh^2 + 2eh + 2h^2 + hd_{ls} + h + d_{ls}) + N_2 eh. \end{aligned}$$

We see the total complexity of the generalized Logsig-RNN is

$$\begin{aligned} E_1 = F_1 + B_1 &= \frac{N_2(N_1+1)}{2}(eh^2 + 2eh + 2h^2 + hd_{ls} + h + d_{ls}) \\ &+ N_2(hd_{ls} + h^2 + h + 2eh + e). \end{aligned} \tag{2.27}$$

For the classical RNN, the forward complexity is

$$F_2 = n(hd + h^2 + h + eh + e).$$

The backward complexity of the classical RNN is

$$B_2 = \frac{n(n+1)}{2}(eh^2 + 2eh + 2h^2 + hd + h + d) + neh.$$

Then the computation complexity of the classical RNN is

$$\begin{aligned} E_2 := F_2 + B_2 &= \frac{n(n+1)}{2}(eh^2 + 2eh + 2h^2 + hd + h + d) \\ &+ n(hd + h^2 + h + 2eh + e). \end{aligned} \tag{2.28}$$

In the following, we assume $n \geq N_2$[5] and we show the step-by-step derivation to prove that if the following additional condition is satisfied

$$\frac{d_{ls}}{d} \leq \frac{n(n+1)(h+1) + 2nh}{N_2(N_1+1)(h+1) + 2N_2h}, \tag{2.29}$$

the bound $E_1 \leq E_2$ holds. Then

$$\begin{aligned} E_2 \geq \frac{N_2(N_1+1)}{2}(eh^2 + 2eh + 2h^2 + h) + \frac{n(n+1)}{2}(hd + d) \\ + N_2(h^2 + h + 2eh + e) + nhd. \end{aligned}$$

It suffices to show that

$$\frac{N_2(N_1+1)}{2}(hd_{ls} + d_{ls}) + N_2hd_{ls} \leq \frac{n(n+1)}{2}(hd + d) + nhd,$$

which implies (2.29). As $n > N_1$ and $n \geq N_2$, the bound $E_1 \leq E2$ can hold easily

[5] If $n > N_2$, we can down-sample the output of the RNN to match the target length of the time series

when we choose the log-signature degree M and number of segments N_1 to satisfy the inequality (2.29).

2.3.4 Universality theorem

In this section, we establish the universality theorem of the generalized Logsig-RNN model to approximate a solution to any controlled differential equation of the form (2.1) defined on the compact interval J under mild conditions. First given the partition $\mathscr{D}_Y = (t_j)_{j=0}^{N_2}$, we define the whole path of the output of the generalized Logsig-RNN Model 2.2 for any $t \in [0,T]$ by

$$o_t = o_{t_j}, \ \ \forall t \in [t_j, t_{j+1}). \tag{2.30}$$

Similarly, we give the definition of $\hat{Y}$ of the numerical approximation scheme (2.23) for any $t \in [0,T]$ as follows

$$\hat{Y}_t = \hat{Y}_{t_j}, \ \ t \in [t_j, t_{j+1}). \tag{2.31}$$

The universality theorem is as follows:

Theorem 2.4 (Universality of Generalized Logsig-RNN Model)**.** *Under Assumption 1, for any $\varepsilon > 0$, there exist a constant δ and parameters Θ such that for $M > \lfloor p \rfloor$, there exists a model $\bar{H} \in \overline{\mathscr{H}}$ with $\Delta\mathscr{D} \leq \delta$, its outputs o defined in Eqn.* (2.30) *satisfies*

$$\sup_{\mathbf{X} \in K} ||Y - o||_{\infty;J} \leq \varepsilon, \tag{2.32}$$

where K is any compact subset of $G^{(\lfloor p \rfloor)}(E)$.

The main idea of proof is similar to that of Theorem 2.1. By triangle inequality

we have

$$||Y - o||_{\infty;J} \leq \underbrace{||Y - \hat{Y}||_{\infty;J}}_{E_3} + \underbrace{||\hat{Y} - o||_{\infty;J}}_{E_4}. \tag{2.33}$$

In Theorem 2.2, we state that $\hat{Y}$ can approximate Y sufficiently well at terminal time T. In fact, we can prove the following generalised theorem that the error between $\hat{Y}$ and Y can be uniformly bounded for any $t \in J$, which implies E_3 can be controlled.

Theorem 2.5 (Global Approximation Theorem)**.** *Under Assumption 1, let* $(\hat{Y}_t)_{t\in J}$ *be defined in* (2.23)*. For any* $\varepsilon > 0$*, there exists* $\delta > 0$ *such that when* $\Delta\mathscr{D} \leq \delta$ *and* $M \geq \lfloor \gamma \rfloor$*, it satisfies that*

$$||Y - \hat{Y}||_{\infty;J} \leq \varepsilon, \tag{2.34}$$

More specifically, there exists positive constants C_1 *and* $\bar{C}$ *defined in* (2.42) *and* (2.14) *respectively such that*

$$||Y - \hat{Y}||_{\infty;J} \leq C_1 \phi_p \left(\max_{u_i \in \mathscr{D}_Y} \omega(u_{i-1}, u_i)^{\frac{1}{p}} \right) + \bar{C} \left[\max_{u_i \in \mathscr{D}_Y} \omega(u_{i-1}, u_i) \right]^{\frac{\lfloor \gamma \rfloor + 1}{p} - 1}. \tag{2.35}$$

To prove Theorem 2.5, we need the next lemma, which states the uniform bound for the approximation error between the numerical solution $\hat{Y}$ and Y at discrete time partition $t_j \in \mathscr{D}_Y$.

Lemma 2.6. *Under Assumption 1, let* $(\hat{Y}_{t_j}^{\mathscr{D}_Y, M})_{j=0}^{N_2}$ *be defined in* (2.2)*. For any* $\varepsilon > 0$*, there exists* $\delta_1 > 0$ *such that when* $\Delta\mathscr{D} \leq \delta_1$ *and* $M \geq \lfloor \gamma \rfloor$*, then* $\hat{Y}_{t_j}$ *satisfies that*

$$\sup_{t_j \in \mathscr{D}_Y} ||Y_{t_j} - \hat{Y}_{t_j}|| \leq \varepsilon, \tag{2.36}$$

More specifically, there exists a positive constant $\bar{C}$ *defined in* (2.14) *such that*

$$\sup_{t_j \in \mathscr{D}_Y} ||Y_{t_j} - \hat{Y}_{t_j}|| \leq \bar{C} \left[\max_{u_i \in \mathscr{D}_Y} \omega(u_{i-1}, u_i) \right]^{\frac{|\gamma|+1}{p}-1}. \tag{2.37}$$

The proof of Lemma 2.6 can be found in the Appendix B. Now we are ready to prove the Theorem 2.5.

Proof of Theorem 2.5. For any $t \in J$, there exist $j \in \{1, \ldots, N_2\}$ such that $t \in [t_j, t_{j+1})$. Then by triangle inequality, we have

$$||Y_t - \hat{Y}_t|| \leq \underbrace{||Y_t - Y_{t_j}||}_{E_5} + \underbrace{||Y_{t_j} - \hat{Y}_{t_j}||}_{E_6} + \underbrace{||\hat{Y}_{t_j} - \hat{Y}_t||}_{E_7}. \tag{2.38}$$

As $\mathbf{X}$ has finite p-variation controlled by ω, there exists some constant $\beta_1 := \beta_1(p, \gamma)$ such that for any $(s,t) \in \Delta_T$,

$$||\mathbf{X}||_{p-var;[s,t]} \leq \beta_1 \omega(s,t)^{\frac{1}{p}}. \tag{2.39}$$

Then by the uniform continuity of ω, for any $\varepsilon > 0$, there exits $\delta_1 > 0$ such that for any $(s,t) \in \Delta_T$ with $|s-t| < \delta_1$, the following inequality holds

$$||\mathbf{X}||_{p-var;[s,t]} \leq \beta_1 \max_{|u-v|<\delta_1} \omega(u,v)^{\frac{1}{p}} \leq \varepsilon. \tag{2.40}$$

By Lemma B.1, for any $\varepsilon > 0$, there exists δ_1 such that for any $|s-t| < \delta_1$,

$$||Y_t - Y_s|| \leq C_1 \phi_p \left(\max_{|u-v|<\delta_1} \omega(u,v)^{\frac{1}{p}} \right) < \varepsilon. \tag{2.41}$$

where constant C_1 is defined as

$$C_1 := C \left(|f|_{\circ\gamma-1} \vee |f|^p_{\circ\gamma-1} \right) \beta_1. \tag{2.42}$$

As per the above, E_5 can be arbitrarily small given $\Delta\mathscr{D}$ sufficiently small. According to Lemma 2.6, for any $\varepsilon > 0$, there exists δ_2 such that $E_6 \leq \varepsilon$ given $\Delta\mathscr{D} < \delta_2$ and truncation degree of the log-signature M sufficiently large. Finally, as $E_7 = 0$, we can obtain the required results given $\delta = \min(\delta_1, \delta_2)$. □

The error E_4 between $\hat{Y}$ and o in (2.33) is also similar to E_2 in the inequality (2.11) of Theorem 2.1. By Remark 2.3, it is noted that for $t_j \in (u_{k-1}, u_k]$, the output o_{t_j} is indeed the ouput of the function $\mathscr{R}_\sigma((l^i)_{i=1}^{k-1}, l_j|\Theta)$, where $\mathscr{R}_\sigma(\cdot|\Theta)$ is a RNN network. Thus, given the definition of the recursive function $G_{\tilde{f},k}$ in (2.19), the control of E_4 is equal to control

$$\hat{Y}_{t_j} - \mathscr{R}_\sigma((l^i)_{i=1}^{k-1}, l_j) = \left(G_{g,k} - G_{A\sigma(\theta_1 x + \theta_2 r),k}\right)((l_i)_{i=1}^{k-1}, l_j), \tag{2.43}$$

for all $j \in \{1, \ldots, N_2\}$. for all $j \in \{1, \ldots, N_2\}$.

Finally, we arrive at the stage of proving Theorem 2.4.

Proof of Theorem 2.4. Theorem 2.5 implies that the error E_1 in the triangle inequality (2.11) can be arbitrarily small given large enough degree of the log-signature M and small enough $\Delta\mathscr{D}$.

Lemma B.3 demonstrates the continuity of the map $\tilde{f} \mapsto G_{\tilde{f},k}$, which leads to the following inequality

$$\sup_j \left\{\hat{Y}_{t_j} - \mathscr{R}_\sigma((l^i)_{i=1}^{k-1}, l_j)\right\} \leq \sup_k \{\bar{C}_k\} ||g - \tilde{f}||_{\infty,K} \leq C_3 ||g - \tilde{f}||_{\infty,K}$$

where $C_3 := C_3(N_2, K)$. As Lemma B.2 ensures that the shallow neural network can approximate any continuous function uniformly well, we are able to show that E_4 can be arbitrarily small by the definition of piecewise-constant interpolation. □

Chapter 3

Logsig-RNN in supervised learning

3.1 Introduction

In this chapter, we investigate the performance of the Logsig-RNN (Model 2.1) on several supervised learning tasks to demonstrate its effectiveness and usefulness in time series data modelling. Supervised learning are mainly divided into regression and classification depending on the output type and widely studied in machine learning research [48]. To validate our model on a diverse set of supervised learning problems, we consider the regression task of learning the SDE solution from synthetic data, and two classification tasks of recognising handwritten digits and human action from empirical data respectively. We provide an extensive benchmark experiments of our model against conventional RNN model and Sig-OLR [24] on the two illustrative examples, i.e. synthetic SDEs data and handwritten digits data. Numerical results confirm that that our model is consistently more efficient and robust than the baseline methods.

The most challenging one among the above proposed tasks is human action recognition (HAR). HAR is an important and difficult task in computer vision with a wide range of applications in human-computer interfaces and communications. In particular, skeleton-based HAR (SHAR) that involves skeleton representation of human bodies instead of raw RGB videos, has gained increasing popularity of

research and development due to low-cost motion sensing devices, e.g. Microsoft Kinect, and reliable pose estimation methods. Compared with RGB representation, skeleton-based methods are robust to illumination changes, free of environmental noises and have benefits of data privacy and security.

Although vast literature is devoted to SHAR [49, 50, 51], the challenge remains open due to two main issues: (1) how to extract discriminative representations for the high dimensional spatial structure of skeletons; (2) how to model the temporal dynamics of motion. With the increasing development and impressive performance of deep learning models e.g. Recurrent Neural Networks (RNN) [33, 34, 35, 52, 53], Convolutional Neural Networks (CNN) [54, 55, 56, 57, 58, 59], and Graph Convolutional Networks (GCN) [60, 61, 62], data-driven deep features have gained increasing attention in SHAR [51]. However, these methods are often data greedy and computationally expensive, and not well adapted to data of different sizes/lengths. For example, when the lengths of data sequences are long and diverse, LSTMs either suffer from tremendous training cost with heuristic padding or are forced to down-sample/re-sample the data, which potentially misses the microscopic information.

To address the above two difficulties, we propose the Path Transformation Layers (PTs) and Logsig-RNN(Model 2.1) respectively, leading to a hybrid model for the SHAR tasks to achieve the comparable state-of-the-art results. On the one hand, the spatial structure in SHAR methods is commonly modelled using coordinates of joints [63, 64, 65], using body parts to model the articulated system [66, 67, 68] or by hybrid methods using information from both joints and body parts [69, 57]. Inspired by [57] and [70], we propose the Path Transformation Layers (PTs), which include an Embedding Layer (EL) to reduce the spatial dimension of pure joint information and a Graph Convolutional Layer (GCN) to learn to implicitly capture the discriminative joints and body parts. On the one other hand, the Log-Signature

Layer (definition 2.1) provides a parsimonious summary of the extracted spatial features locally and hence significantly reduce the time dimension of the RNN. The log-signature transformation is effective to handle time series with variable length without the use of padding and provide robustness to missing data. This enables the following RNN layer to learn more expressive deep features, leading to a systematic method to treat the complex time series data in SHAR.

This chapter is organised as follows. Section 3.2 demonstrates the superior performance of Logsig-RNN against RNNs on the two illustrative examples in both accuracy and robustness. Section 3.3 first explains the architecture of the model PT-Logsig-RNN customized for learning SHAR datasets. Then we show that replacing RNNs with Logsig-RNN in the SOTA networks consistently improves the accuracy and robustness on both Chalearn 2013 and NTU RGB+D 120 dataset. We also illustrates the efficiency of Logsig-RNN against RNNs and discrete consine transformation (DCT). Section 3.4 concludes.

3.2 Illustrative examples

In this section, we demonstrate the performance of the Logsig-RNN algorithm on the synthetic SDE data and handwritten pen-digit data in terms of the accuracy, efficiency and robustness.[1]

3.2.1 Synthetic data

As an example of high frequency data, we simulate the solution Y_T to the SDE of Example 3.1 using Milstein's method with the time step $\frac{T}{50000}$ for $T = 10$. An input path is the discretized Brownian motion $W_{\hat{\mathscr{D}}}$, where $\hat{\mathscr{D}} = D_{50001}$[2]. We simulate 2000 samples of $(X_{\hat{\mathscr{D}}}, Y_T)$, which is split to 80% for the training and the rest for the testing. Here we benchmark our approach with (1) RNN_0: the conventional RNN

[1] We implement all algorithms in Pytorch. It runs on a computer equipped with GeForce RTX 2080 Ti and GeForce Quadro RTX 8000 GPU.

[2] D_n denotes an equally spaced partition of $[0, T]$ of n steps.

model, (2) Sig-OLR: the linear regression model on the signature, (3) Sig-RNN: the RNN model with the signature sequence.

Example 3.1. Suppose Y_t satisfies the following SDE:

$$dY_t = (-\pi Y_t + \sin(\pi t))dX_t^{(1)} + Y_t dX_t^{(2)}, Y_0 = 0, \tag{3.1}$$

where $X_t = (X_t^{(1)}, X_t^{(2)}) = (t, W_t)$, W_t is a 1-d Brownian motion, and the integral is in the Stratonovich sense.

Data	**Methods**	**Fea. dim.**[3]	**Error**($\times 10^{-6}$)	**Train time(s)**
High Frequency (50k steps)	RNN_0	(50k, 1)	–	–
	Sig-OLR	62	2.25	**178**
	Sig-RNN	(4,14)	2.40	360
	Logsig-RNN	**(4,8)**	**2.14**	529
Down-sampling (1k steps)	RNN_0	(1k,1)	7.79	50930
	Sig-OLR	62	3.69	**9**
	Sig-RNN	(4,14)	2.55	177
	Logsig-RNN	**(4,8)**	**2.16**	343
Missing Data (drop 5% from 1k)	RNN_0	(1k,1)	16.40	47114
	Sig-OLR	62	3.75	**9**
	Sig-RNN	(4,14)	3.05	182
	Logsig-RNN	**(4,8)**	**2.91**	372

Table 3.1: Comparison of methods on the SDEs data.

As shown in Table 3.1, we apply the above four methods for three kinds of inputs (1) $X_{\hat{\mathscr{D}}}$ (high frequency); (2) down-sampling $X_{\hat{\mathscr{D}}}$ to $1k$ time steps (down-sampling); (3) randomly throw away 5% points of $1k$ down sampled data (missing data). We compare the accuracy and training time of the Logsig-RNN algorithm. The training time is the first time of the loss function of the model to reach the error tolerance level $2 * 10^{-6}$ before 25k epochs in the train set and the MSE is chosen as performance metric. First of all, Table 3.1 shows that the Logsig-RNN achieves the best accuracy for all three cases among all the methods. In particular, it is the

[3]For features of two dimensions, the first one is for temporal and the second is for spatial.

most robust to missing data. Moreover, it reduces the time dimension of RNN from $50k/1k$ to 4, and thus significantly save the training time from $50930s$ to $343s$.

3.2.2 Pen-digit data

In this subsection, we apply the Logsig-RNN algorithm on the UCI sequential pen-digit data[4]. Because the average length of the data is 40 and the minimum length is 9, we did parameter tuning for the number of segments in the set of $\{2, 4, 8\}$. In Table 3.2, the Logsig-RNN with $M = 4$ and $N = 4$ achieves the accuracy 97.88% in the testing data compared with 95.80% of RNN_0. In addition, the training time of the Logsig-RNN takes 3% of the training time of RNN_0.

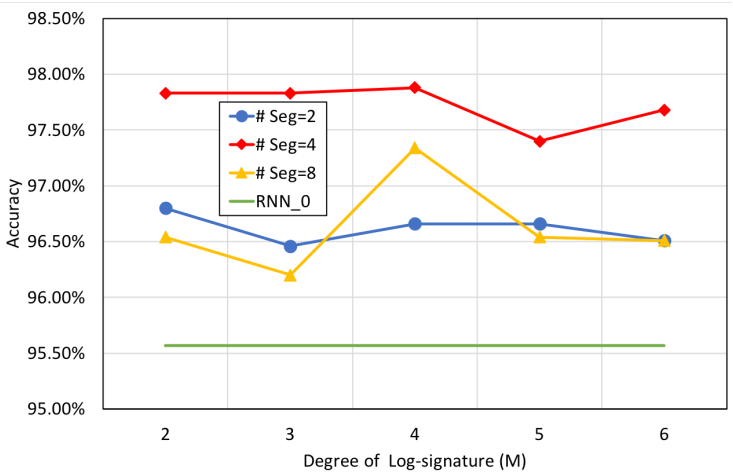


Figure 3.1: The accuracy comparison of Logsig-RNN in the testing set.

Robustness to missing data and change of sampling frequency To mimic the missing data case, we randomly throw a certain portion of points for each sample, and evaluated the trained models of Logsig-RNN and RNN_0 to the new testing data. Table 3.2 shows that our proposed method outperforms the other methods significantly for the missing data case. Figure 3.2 shows the robustness of trained Logsig-RNN model for the down-sampled test data. Here the test data is down-

[4] https://archive.ics.uci.edu/ml/datasets/Pen-Based+Recognition+of+Handwritten+Digits

sampled to that of length 8 provided in UCI data. For $M = 4$, the accuracy of testing data is above 80%, which is about 4 times of that of the baseline methods.

r \ M	2	3	4	5	6	RNN_0
0%	97.83%	97.83%	97.88%	97.40%	97.68%	95.80%
10%	97.63%	97.77%	97.14%	96.88%	97.60%	40.91%
20%	96.74%	97.06%	96.68%	95.85%	97.06%	37.28%
30%	95.99%	95.40%	95.17%	95.03%	95.77%	32.65%

Table 3.2: The accuracy of the modified testing set using different missing data rate (r). Here $N = 4$.

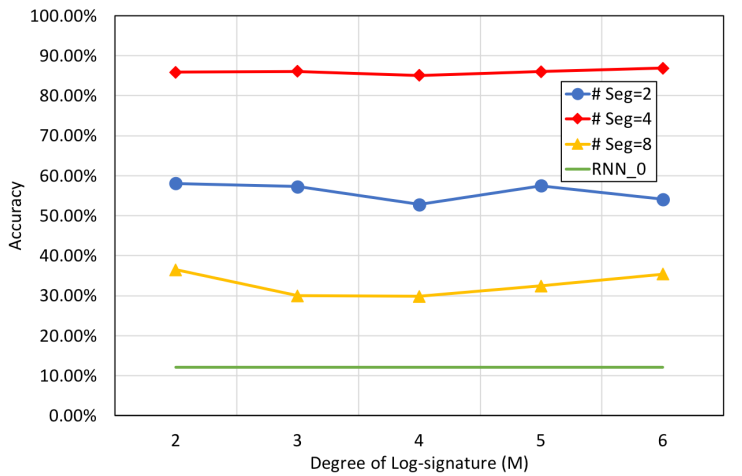


Figure 3.2: Validation of the trained models on the down-sampled dataset. The accuracy of RNN_0 is below 13.5%

3.3 Skeleton human action recognition

In this section, we propose a simple, compact and efficient PT-Logsig-RNN Network for SHAR, which is composed of (1) path transformation layers, (2) the Logsig-RNN (Model 2.1) and (3) a fully connected layer. The overall PT-Logsig-RNN model is depicted in Figure 3.3. We start by propose useful path transformation layers to further improve the performance of the Logsig-RNN module in SHAR tasks. Then, with quantitative analysis on Chalearn2013 gesture dataset and NTU

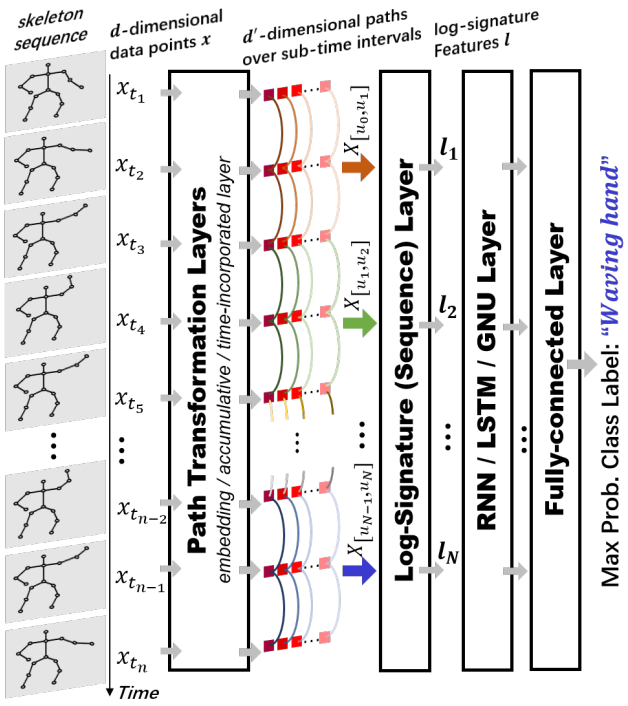


Figure 3.3: Architecture of PT-Logsig-RNN Model. It consists with the first Path Transformation Layers, the Log-Signature (Sequence) Layer, the RNN-type layer and the last fully connected layer. It is used for both action and gesture recognition in our experimental section.

RGB+D 120 action dataset, we validated the efficiency and robustness of Logsig-RNN and the effects of the Path Transformation layers.

3.3.1 PT-Logsig-RNN network

To more efficiently and effectively exploit the spatio-temporal structure of the path, we further investigate the use of two main path transformation layers (i.e. Embedding Layer and Graph Convolutional Layer) in conjunction with the Log-Signature Layer.

A skeleton sequence X can be represented as a $n \times F \times D$ tensor (landmark sequence) and a $F \times F$ matrix $\mathscr{A}$ (bone information), where n is the number of frames in the sequence, F is the number of joints in the skeleton, D is the coordinate dimension and $\mathscr{A}$ is the adjacency matrix to denote whether two joints have a bone

connection or not.

Embedding Layer (EL) In the literature, many models only use landmark data without explicit bone information. One can view a skeleton sequence as a single path of high dimension(d) (e.g. a skeleton of 25 3D joints has $d = F \cdot D = 75$). Since the dimension of the truncated log-signature grows fast w.r.t. d, we add a linear Embedding Layer before the Log-Signature Layer to reduce the spatial dimension and avoid this issue. Motivated by [57], we first apply a linear convolution with kernel dimension 1 along the time and joint dimensions to learn a joint level representation. Then we apply full convolution on the second and third coordinates to learn the interaction between different joints for an implicit representation of skeleton data. The output tensor of EL has the shape $n \times d_{el}$, where d_{el} is a hyperparameter to control spatial dimension reduction. One can view the embedding layer as a learnable path transformation that can help to increase the expressivity of the (log)-signature.

In practice, the Embedding Layer is more effective when subsequently adding the Time-Incorporated Layer (TL) and the Accumulative Layer (AL) which applies time augmentation and cumulative sum functions respectively. The details of time augmentation and cumulative sum can be found in Appendix A. For simplicity we will use EL to denote the Embedding Layer composed with TL and AL in the below numerical experiments.

Graph Convolutional Layer (GCN) Recently, graph-based neural networks have been introduced and achieved SOTA accuracy in several SHAR tasks due to their ability to extract spatial information by incorporating additional bone information using graphs. We demonstrate how a GCN and the Logsig-RNN can be combined to form the GCN-Logsig-RNN to model spatio-temporal information.

First, we define the GCN layer on the skeleton sequence. Let G_θ denote a graph convolutional operator $F \times D \to F \times \tilde{D}$ associated with $\mathscr{A}$ by the following

mapping

$$G_\theta(x) := (\Gamma^{-\frac{1}{2}}(\mathscr{A}+I)\Gamma^{-\frac{1}{2}})x\theta,$$

where θ is a $D \times \tilde{D}$ matrix, $\Gamma^{ii} = \sum_j(\mathscr{A}^{ij} + I^{ij})$, and I is the identity matrix. Then we extend G_θ to the skeleton sequence by applying G_θ to each frame X_t, i.e. $G_\theta : X = (X_t)_{t=1}^n \mapsto (G_\theta(X_t))_{t=1}^n$ to obtain an output as a sequence of graphs of time dimension n with the adjacency matrix $\mathscr{A}$.

Next we propose the below *GCN-Logsig-RNN* to combine GCN with the Logsig-RNN. Let $\hat{X}_t^{(i)} \in \mathbb{R}^{\tilde{D}}$ denote the features of the i^{th} joint of the GCN output $G_\theta(X_t)$ at time t. For each i^{th} joint, $\hat{X}^{(i)} = (\hat{X}_t^{(i)})_{t=1}^n$ is a $\tilde{D}$-dimensional path. We apply the Logsig-RNN to $\hat{X}^{(i)}$ as the feature sequence of each i^{th} joint, and hence obtain a sequence of graphs whose feature dimension is equal to the log-signature dimension and whose time dimension is the number of segments in Logsig-RNN. This in particular also allows for the module to be stacked. To further efficiently and effectively exploit the path, we further propose the transformation layers accompanied with Log-Signature Layer which could help extract spatial information or avoid curse of dimensionality. The overall model, namely PT-Logsig-RNN, is shown in Figure 3.3. The implementation details of SHAR datasets are given in Appendix C.

3.3.2 Gesture recognition

The Chalearn 2013 dataset [22] is a public available dataset for gesture recognition, which contains 20 Italian gestures performed by 27 subjects. It contains 13,858 samples with the average number of frames equals to 39. It provides Kinect data, which contains RGB, depth, foreground segmentation and skeletons. Here, we only use skeleton data (20 3D joints) for the gesture recognition.

State-of-the-art performance:We apply the EL-Logsig-LSTM model to

Chalearn2013 and achieve state-of-the-art (SOTA) classification accuracy shown in Table 3.3 of the 5-fold cross validation results. The EL-Logsig-LSTM ($M = 2, N = 4$) with data augmentation achieves performance comparable to the SOTA [71]. With augmentation, the PT-Logsig-RNN significantly outperform others to achieve the state-of-the-art result according to 5-folds cross validation results. Figure 3.4 displays when number of segments (N) is 4, the testing accuracy is the highest, and when it is larger than 16, there is a significant decrease.

Methods	**Accuracy(%)**	**Data Aug.**
Deep LSTM [52]	87.10	–
Two-stream LSTM [53]	91.70	$\surd$
ST-LSTM + Trust Gate [72]	92.00	$\surd$
3s_net_TTM [73]	92.08	$\surd$
Shift-GCN [74]	90.86	$\surd$
Multi-path CNN[75]	93.13	$\surd$
TS-DPM[71]	**94.26**	$\surd$
$LSTM_0$	90.92	$\times$
$LSTM_0$ (+data aug.)	91.18	$\surd$
EL-Logsig-LSTM	91.77 ± 0.34	$\times$
EL-Logsig-LSTM(+data aug.)	$\mathbf{92.94 \pm 0.21}$	$\surd$
GCN-Logsig-LSTM	91.92 ± 0.28	$\times$
GCN-Logsig-LSTM(+data aug.)	92.86 ± 0.23	$\surd$

Table 3.3: Comparison of the accuracy ($\pm$ standard deviation) for different methods on the Chalearn 2013 data.

Investigation of path transformation layers: To validate the effects of EL, we compare the test accuracy and number of trainable weights in our network with and without EL on Chalearn 2013 data. Table 3.4 shows that the addition of EL increases the accuracy by 1.87 percentage points (*pp*) while reducing the number of trainable weights by over 60%. Let D_{el} denote the spatial dimension of the output of EL. We can see that even introducing EL without a reduction in dimensionality, i.e. setting D_{el} to the original spatial dimension of 60, improves the test accuracy. Decreasing the dimensionality can lead to further improvements, with the best results in our experiments at $D_{el} = 30$ with a test accuracy of 93.38%. A further decrease of D_{el} leads to the performance deteriorating. The high accuracy of our model using

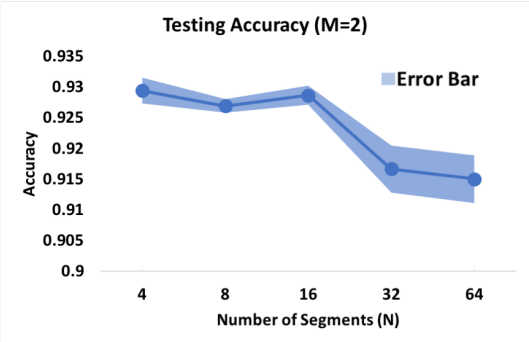


Figure 3.4: The sensitivity analysis of EL-Logsig-LSTM model w.r.t. the number of segments on Chalearn 2013 data.

EL to reduce the original spatial dimension from 60 to $D_{el} = 30$ suggests that EL can learn implicit and effective spatial representations for the motion sequences. AL and TL contribute a 0.86 *pp* gain in test accuracy to the EL-Logsig-LSTM model.

Methods	D_{el}	Accuracy(%)	# Trainable weights
With EL	10	91.09	120,594
	20	92.92	213,574
	30	**93.38**	357,954
	40	93.10	553,734
	50	93.33	800,914
	60	93.30	1,099,494
W/O EL	-	91.51	985,458

Table 3.4: The effect of EL on the testing accuracy. D_{el} is the spatial dimension of EL output.

Investigation of different segment numbers in Logsig-LSTM: Figure 3.4 shows that for the EL-Logsig-LSTM, increasing the number of segments (N) up to 16, the test accuracy stays stable, and increasing N further worsens the model performance. The optimal number of segments N is 4. As the average number of frames of the dataset is 39, if N is larger than 16, the information contained in each segment of many samples would be a straight line. Since the log-signature of a straight line is just the difference of the start and end point, applying the Logsig-LSTM would be the same as applying the LSTM, which explains the decreasing in

accuracy of the figure.

Robustness analysis: Regarding to the robustness to missing data, we randomly set a certain percentage of frames (r) by all-zeros for each sample in the validation set, and evaluate the trained models of our method and RNN_0 to the new validation data. Table 3.5 shows that EL-Logsig-LSTM model with $M = 2$ consistently beats the baseline RNN for different r, which validates the robustness of our method comparing with the benchmark.

r \ M	2	3	4	LSTM
0%	92.21	89.66	70.71	90.92
10%	91.32	88.58	69.11	81.77
20%	90.33	86.60	67.89	68.22
30%	87.68	81.74%	63.24	50.35
50%	74.40	57.13	41.07	21.78

Table 3.5: The accuracy (%) of the testing set with missing data with different dropping ratio (r) on Chalearn 2013. Here $N = 4$.

3.3.3 Action recognition

NTU RGB+D 120 [23] is a large-scale benchmark dataset for 3D action recognition, which consists of $114,480$ RGB+D video samples that are captured from 106 distinct human subjects for 120 action classes.

In this subsection, we apply the EL-Logsig-LSTM, GCN-Logsig-LSTM and a stacked two-layer GCN-Logsig-LSTM(GCN-Logsig-LSTM^2) to demonstrate that the Logsig-RNN can be conveniently plugged into different neural networks and achieve competitive performance.

Figure 3.5 shows the testing accuracy increases with respect to number of segments (N) up to 64 and then decreases on both cross-setup and cross-subject tasks. Different from the Chalearn 2013 dataset, the NTU RGB+D 120 is a large and complex one. Methods in the literature always have severe over-fitting issue, which have multi-layer architecture with large amount of parameters such that the accuracy can

be improved. It is the possible reason that when $N = 64$, which is close to the average number of frames (i.e. 79), such that our model achieves best results.

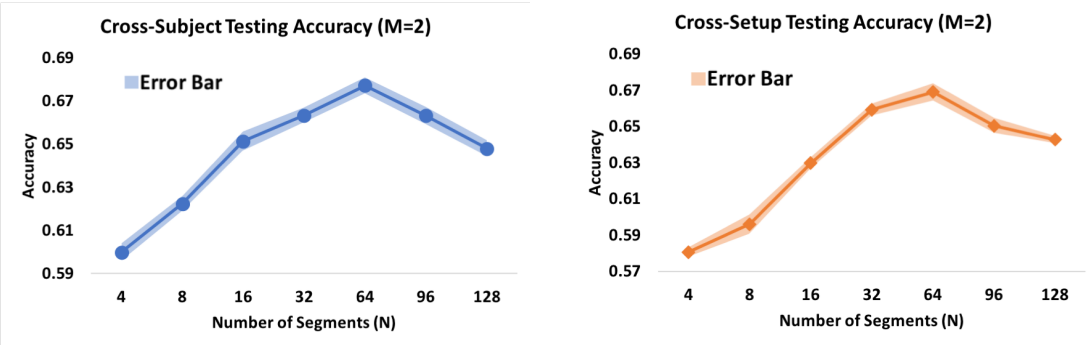


Figure 3.5: The sensitivity analysis of Logsig-RNN model w.r.t the number of segments on NTU+B 120. datast.

Among non-GCN models, for X-Subject protocol, our EL-Logsig-LSTM model outperforms other methods, while it is competitive with [54] and [59] for X-Setup. The latter leverages the informative pose estimation maps as additional clues. Table 3.6 shows the ablation study of EL-Logsig-LSTM For the X-Subject task, adding EL layer results in a 0.7 *pp* gain over the baseline and the Logsig layer further gives a 5.9 *pp* gain. As the 5-fold cross validation results shown in Table 3.7, we subsequently add the Path Transformation Layers (PT) and the Log-signature layer (Logsig) to the baseline LSTM to validate the performance of each model. For X-Subject task, adding PT Layer results in a 0.7 percentage points (*pp*) gain over the baseline and the Logsig layer further gives a 5.9 *pp* gain. For X-Subject protocol, our method outperforms other methods. For X-Setup, our method is competitive with [59]. The latter leverages the informative pose estimation maps as additional clues. Notice that our PT-Logsig-LSTM is flexible enough to allow incorporating other advanced techniques (e.g. data augmentation and attention module) or combining multimodal clues (e.g. pose confidence score) to achieve further improvement.

When changing the EL to GCN in EL-Logsig-LSTM, we improved the accu-

Methods	X-Subject(%)	X-Setup(%)
ST LSTM[34]	55.7	57.9
FSNet[58]	59.9	62.4
TS Attention LSTM[35]	61.2	63.3
Pose Evolution Map[59]	64.6	**66.9**
Skelemotion[54]	**67.7**	**66.9**
LSTM (baseline)	60.9 ± 0.47	57.6 ± 0.58
EL-LSTM	61.6 ± 0.32	60.0 ± 0.35
EL-Logsig-LSTM	$\mathbf{67.7 \pm 0.38}$	$\mathbf{66.9 \pm 0.47}$

Table 3.6: Comparison of accuracy ($\pm$ standard deviation) among non-GCN models on the NTU RGB+D 120.

Methods	X-Subject(%)	X-Setup(%)
RA-GCN[76]	81.1	82.7
4s Shift-GCN[74]	85.9	87.6
MS-G3D Net[70]	86.9	**88.4**
PA-Res-GCN[77]	**87.3**	88.3
(GCN-LSTM)	69.4 ± 0.46	71.4 ± 0.30
$(\text{GCN-LSTM})^2$	72.1 ± 0.53	74.9 ± 0.27
GCN-Logsig-LSTM	70.9 ± 0.22	72.4 ± 0.33
$\mathbf{(GCN\text{- }Logsig\text{-}LSTM)^2}$	$\mathbf{75.8 \pm 0.35}$	$\mathbf{78.0 \pm 0.46}$

Table 3.7: Comparison of accuracy ($\pm$ standard deviation) among GCN models on the NTU RGB+D 120.

racy by 3.2 *pp* and 5.5 *pp* for X-Subject and X-Setup tasks respectively. By stacking two layers of the GCN-Logsig-LSTM, we further improve the accuracy by 4.9 *pp* and 5.6 *pp*. The SOTA GCN models ([70, 77]) have achieved superior accuracy, which is about 11 *pp* higher than our best model. This may result from the use of multiple input streams (e.g. joint, bones and velocity) and more complex network architecture (e.g. attention modules and residual networks). Notice that our EL-Logsig-LSTM is flexible enough to allow incorporating other advanced techniques or combining multimodal clues to achieve further improvement.

To test the robustness of each method in handling missing data and varying frame rate, we construct new test data by randomly discarding/repeating a certain percentage (r) of frames from each test sample, and evaluate the trained models on the new test data. Table 3.5 shows that the proposed EL-Logsig-LSTM exhibit only

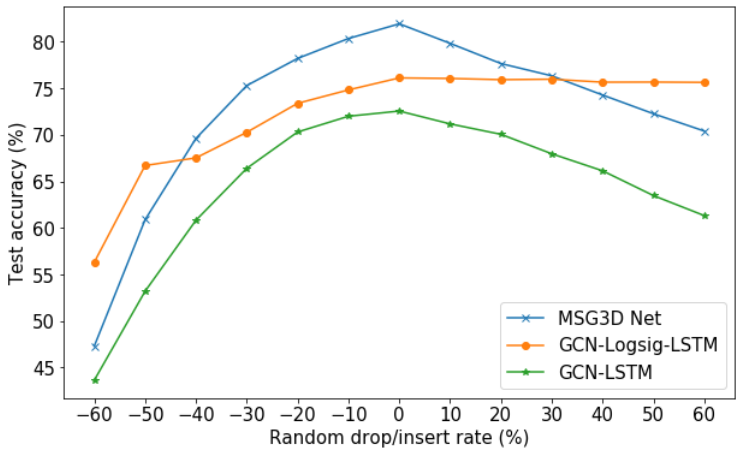


Figure 3.6: The robustness test of random dropping/inserting frames to the NTU RGB+D 120 data.

very small drops in accuracy on Chalearn2013 as r increases while the accuracy of the baseline drops significantly. We start to see a more significant drop in accuracy in our models only as we reach a drop rate of 50%. Figure 3.6 shows that the same is true for the proposed GCN-Logsig-LSTM model on the NTU data. Compared with GCN-LSTM (baseline), it is clearly more robust. And compared with the SOTA model MSG3D Net [70], at a drop rate of 50% or more and a increase rate of 30% or more, our method even outperforms it which has a 10 *pp* higher accuracy than our model at $r = 0$. This demonstrates that both EL-Logsig-LSTM and GCN-Logsig-LSTM are significantly more robust to missing data than previous models in many cases.

3.3.4 Efficiency analysis

To demonstrate that the log-signature can help reduce the computational cost of backpropagating through many timesteps associated with RNN-type models we compare the training time and accuracy of a standard single LSTM block with a Logsig-LSTM using the same LSTM component on the ChaLearn dataset. To evaluate the efficiency as the length of the input sequence grows we linearly interpolate

between frames to generate longer input sequences. We can see in the results in Figure 3.7 that, as the length of the input sequence grows, the time to train the Logsig-LSTM grows much slower than that of the standard LSTM. Moreover, the Logsig-LSTM retains its accuracy while the accuracy of the LSTM drops significantly as the input length increases. This shows that the addition of the log-signature helps with capturing long-range dependencies in the data by efficiently summarizing local time intervals and thus reducing the number of timesteps in the LSTM.

We also compare the performance of the log signature and the discrete cosine transformation (DCT), which was used in [78] for reduction of the temporal dimension. Both transformations can be computed as a pre-processing step. As can be seen in Figure 3.7 in this case the log-signature leads to slightly longer training time than DCT due to a larger spatial dimension, but achieves a considerably higher accuracy. If the transformation is computed at training time, as is essential if it is not used as the first layer of the model as in this work, we can see in Figure 3.7 that the cost of DCT is comparable to the log-signature.

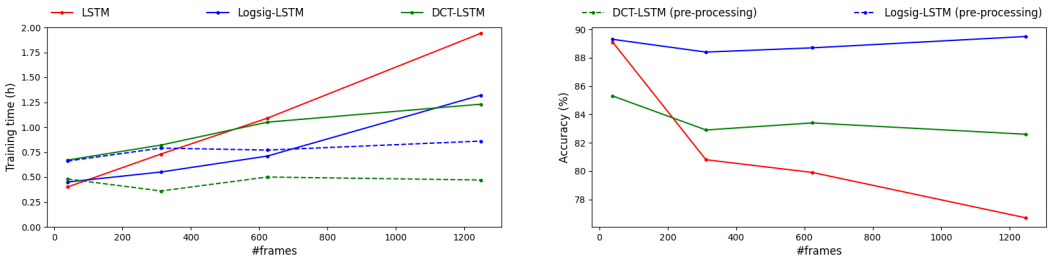


Figure 3.7: Comparison of training time and accuracy of standard LSTM and Logsig-LSTM. With increasing length of the input sequence the training time of the Logsig-LSTM model grows slower than that of the LSTM, without a drop in accuracy. DCT achieves a lower accuracy at comparable training time.

3.4 Conclusion and future work

The Logsig-RNN model, inspired from the numerical approximation theory of SDEs, provides an accurate, efficient and robust algorithm to learn any continu-

ous functional on streamed data. Numerical results show that it improves the performance of LSTM significantly on both synthetic data and empirical data. We propose an efficient and compact end-to-end PT-Logsig-RNN network for SHAR tasks, providing a consistent performance boost of the SOTA models by replacing the RNN with the Logsig-RNN. As an enhancement of the RNN layer, the proposed Logsig-RNN module can reduce the time dimension, handle irregular time series and improve the robustness against missing data and varying frame rates. In particular, EL-Logsig-RNN achieves SOTA accuracy on Chalearn2013 for gesture recognition. For large-scale action data, the GCN-Logsig-RNN based models significantly improve the performance of EL-Logsig-RNN. Our model shows better robustness in handling varying frame rates. It merits further research to improve the combination with GCN-based models to further improve the accuracy while maintaining robustness. In situations where the input is time dependent, applying the algorithm is straightforward where we need to take care of the spatial dimension by applying embedding layers. When the input is time independent, such as in Graph Neural Networks [79], by regarding the trajectory starting from a certain node in the graph as a path, we can apply Log-Signature Layer on it as well. The difficulty comes from how we define the path, where we need to incorporate domain knowledge of the data such as molecule structures, human skeletons,etc., which we will focus on in our future works.

Chapter 4

Logsig-RNN in generative tasks

4.1 Introduction

In this Chapter, we investigate the ability of the generalized Logsig-RNN (Model 2.2) in the challenging data generation tasks. The main obstacles of synthesizing time series data mainly accrue from the complex dynamics of sequential data. Generative adversarial networks (GANs) have recently been applied in generating realistic-looking time series data, by simply deploying RNNs as the generators [36, 37, 80, 81, 82]. For instance, in [36], Recurrent GAN (RGAN) and Recurrent Conditional GAN (RCGAN), which adopt RNN in both generator and discriminator, are proposed to produce realistic medical data to help resolve the privacy concerns. However, classical RNNs are less adapted to generating irregular sampled and high-frequency data as we discussed in Section 2.1. To overcome these deficiencies, we propose to replace the commonly used RNNs by our generalized Logsig-RNN in order to generate high-fidelity time series data. In contrast to classical RNNs, the generalized Logsig-RNN, in its sequence-to-sequence formulation, can generate time series of arbitrary length at uneven-spaced time steps. As Remark 2.3 illustrates, the output data at each time step of our model does not depend on the output at the previous time step, but depends on the output at the time step of a much coarser time partition. The novel structure avoids the long

temporal dependency issue of classical RNNs. Thus it can handle inputs of long length and efficiently produce high-frequency data. Apart from RNNs, we use Neural RDE (NRDE) [42] as another baseline method. NRDE is a model based on the Neural CDE [8] framework. Motivated by the log-ODE method, NRDE is developed to handle long length input. By applying the adjoint method, NRDE is memory-efficient in generating high-frequency output, but it is less time-efficient compared with the generalized Logsig-RNN model.

Besides innovating the generator based on the Logsig-RNN, we develop a novel generative framework to enhance the ability of traditional GAN in capturing the temporal dependence of the joint probability distributions of time series data. TimeGAN [37] illustrates improvements by adding supervised loss to the optimization of the generator and by minimising recovery loss in a embedding space of the time series data. COT-GAN [82] proposes to replace standard Jensen–Shannon (JS) divergence by Sinkkorn divergence to overcome the deficiencies of optimizing traditional GAN. In fact, the min-max objective function of classical GANs make them notoriously difficult to tune. [83] leverages the Wasserstein distance, proposing an alternative, i.e. Wasserstein GAN (WGAN), to overcome issues such as the locally saturated JS divergence which leads to 0 gradient for well-trained discriminator, and collapse mode of classical GAN. In this work, we contribute by changing the min-max formulation of WGANs to a supervised learning problem. The proposed method, called Sig-Wasserstein GAN (Sig-WGAN), leverages the representation ability of the expected signature features. It enables Sig-WGAN to efficiently discriminate real and synthesised data, especially for high frequency and irregular sampled time series.

This chapter is organised as follows. Section 4.2 explains the details of WGAN, provides a mathematical definition of the Sig-Wasserstein GAN and states the theoretical results. In Section 4.3, we train the generalized Logsig-RNN as the

generator with both Sig-WGAN and WGAN on synthetic and empirical datasets. The generalized Logsig-RNN demonstrates superior performance on the evaluation metrics of correlations and marginal distributions, and is more rubust and efficient than LSTM and NRDE. We also show our method outperforms existing methods in the literature on the eICU dataset. The readers are referred to additional numerical results such as on stock data in [20] for the superior performance of Sig-WGAN over WGAN.

4.2 Method

We aim to illustrate the generative ability of the proposed generalized Logsig-RNN model under both the Wasserstein and Sig-Wasserstein scenarios. In this section, we first specify the path space used in our methods. Then we review the formulation of WGAN. Finally, based on the Wasserstein distance, we develop the Sig-Wasserstein distance and construct Sig-WGAN mathematically.

Let $(\Omega, \mathscr{F}, \mathbf{P})$ be a probability space, under which μ and ν are two distributions induced by a $\mathscr{X}$-valued stochastic process. $\mathscr{P}(\Omega)$ denotes the set of distributions supported on Ω. Given D independent trajectories $((x_{t_i}^j))_{j=1}^D$ sampled from $\nu \in \mathscr{P}(\mathscr{X})$ or given one long trajectory $(x_{jt,(j+1)t})_{j=0}^D$ of stationary data the aim of generative modelling is to learn a model capable of producing high fidelity data sampled from $\nu \in \mathscr{P}(\mathscr{X})$, without explicitly modelling the target distribution.

Let $\Omega_0^1(J, \mathbb{R}^d)$ be the space of continuous time augmented paths, i.e. $\Omega_0^1(J, \mathbb{R}^d) = \{t \to (t, x_t) \mid x_t \in C_0^1(J, \mathbb{R}^d)\}$. For the discrete time series $(x_{t_i})_{i=1}^n \in \mathscr{X} \subseteq \mathbb{R}^{d \times n}$, we assume that $(x_{t_i})_{i=1}^n$ is distributed according to some unknown target distribution $\nu \in \mathscr{P}(\mathscr{X})$. Unlike the majority of the work in the literature, we don't assume that the data points are equidistant from each other. The time augmentation for discrete time series is defined as a map ϕ: for all $(x_{t_i})_{i=1}^n$,

$$\phi((x_{t_i})_{i=1}^n) = ((t_1, x_{t_1}), \ldots, (t_n, x_{t_n})).$$

Apart from the time augmentation, there are other feasible transformation methods (listed in Appendix A) on the discrete time series as long as the transformation secures the uniqueness of the signature.

Remark 4.1. Theorem 1.5 states that the signature determines a path uniquely up to tree-like equivalence. With the time augmentation, the parametrization variance of the signature is removed [24] and the signature of the path in $\Omega_0^1(J,\mathbb{R}^d)$ is completely unique.

4.2.1 Wasserstein generative adversarial network

We start with introducing the Wasserstein distance, which measures the cost of transporting one probability distribution to another. The Wasserstein distance is a distance function between two measures μ and ν on the metric space (Ω,d) defined as follows:

Definition 4.1 (p-th Wasserstein distance)**.** Let $P_p(\Omega)$ be the set of measures on Ω with finite p-th moment, that is there exists x_0 in Ω such that for any $\mu \in P_p(\Omega)$,

$$\int_\Omega d(x,x_0)^p d\mu(x) < \infty.$$

The p-th Wasserstein distance between two probability measures $\mu,\nu \in P_p(\Omega)$ is

$$W_p(\mu,\nu) = \left(\inf_{\gamma\in\Gamma(\mu,\nu)} \int_{\Omega\times\Omega} d(x,y)^p d\gamma(\mu,\nu)\right)^{\frac{1}{p}} \tag{4.1}$$

where $\Gamma(\mu,\nu)$ is the set of all measures on $\Omega\times\Omega$ with marginals μ and ν on the first and second factors respectively.

Intuitively, the joint distribution $\gamma(\mu,\nu)$ describes how much "mass" need to be transported from x to y such that the distribution μ is transformed into ν.

In WGAN, the W_1 distance is used, because it has so-called Kantorovich-Rubinstein dual representation which provides a computationally efficient way of

obtaining the W_1 distance between two measures μ and ν, denoted by $W_1(\mu,\nu)$ as follows:

$$W_1(\mu,\nu) = \sup_{\|f\|_{\text{Lip}}\leq 1} \mathbb{E}_{X\sim\mu}[f(X)] - \mathbb{E}_{X\sim\nu}[f(X)], \tag{4.2}$$

where the supremum is over all the 1-Lipschitz functions $f : E \to \mathbb{R}$ with its Lipschitz norm smaller than 1.

In the context of generative modelling, let $\mathscr{Z}$ denote a latent space, and $Z \in \mathscr{Z}$ denote a variable with the known distribution $\mu_Z \in \mathscr{P}(\mathscr{Z})$. Let $\mu \in \mathscr{P}(\mathscr{X})$ denote a target distribution of observed data. The aim of Wasserstein Generative Adversarial Network (WGAN) is to train a model that induces a distribution ν, so that $W_1(\mu,\nu)$ is small. Similar to classical GANs, the WGAN is composed of the generator G and the discriminator D. The generator $G_\theta : \mathscr{Z} \to \mathscr{X}$ is a parameterized map transporting the latent distribution μ_Z to the model distribution ν, i.e. the distribution induced by $G_\theta(Z)$, where $\theta \in \Theta^g$ and Θ^g is the parameter set. To discriminate between real and synthetic samples, one parameterises the test function f in the definition of the W_1 metric (Eqn. (4.2)) by a network f_η with the parameter set $\eta \in \Theta^d$. Training the generator entails solving a min-max problem. Indeed, to find optimal $(\theta^\star,\eta^\star)$ one needs to solve

$$\min_\theta \max_{\|f_\eta\|_{\text{Lip}}\leq 1} \mathbb{E}[f_\eta(X)] - \mathbb{E}[f_\eta(G_\theta(Z))].$$

In practice, the min-max problem is solved by iterating gradient descent-ascent algorithms. Its convergence can be studied using tools from game theory [84, 85]. It is well known that first order methods to solve the min-max problem might not converge in practice, even in the convex-concave case [86, 87, 88]. Consequently the adversarial training is notoriously difficult to tune, [89, 84], and generalisation error is very sensitive to the choice of discriminator and hype-parameters as was

demonstrated in a large scale study [90].

4.2.2 Sig-Wasserstein generative adversarial network

To surmount the inefficiency of training a WGAN for time series generation, we design a new metric, i.e. Signature Wasserstein-1 (Sig-W_1) metric, based on the desirable properties of signature features. The first is the characteristic property of the expected signature [91]. The second is the universality of the signature to approximate any continuous functions on the signature space. The former ensures the law of the time series can be efficiently captured, and the latter helps to change the optimization problem and derive the analytical formula for the solution.

4.2.2.1 Expected signature of a stochastic process

Let us consider a E-valued stochastic process X under the probability space. Assume that the signature of X is well defined almost surely, and $S(X)$ has finite expectation. We call $\mathbb{E}[S(X)]$ the expected signature of X. An immediate consequence of Proposition 6.1 in [91] on the uniqueness of the expected signature is summarized in the below theorem:

Theorem 4.1. *Let X and Y be two $\Omega(J,E)$-valued random variables. If $\mathbb{E}[S(X)] = \mathbb{E}[S(Y)]$ and $\mathbb{E}[S(X)]$ has infinite radius of convergence, then $X = Y$ in the distribution sense.*

Intuitively, under some regularity condition, the expected signatures serves as the moment generating function, which can characterize the law induced by a stochastic process. For example, the expected Stratonovich signature of Brownian motion determines the law of the Brownian motion in [92].

4.2.2.2 Signature Wasserstein-1 (Sig-W_1) metric

We propose a new Sig-W_1 metric on the measures on the path space $\Omega_0^1(J,\mathbb{R}^d)$ by combining the signature features and the W_1 metric to achieve better computation efficiency. Let μ and ν be two measures on the path space $\Omega_0^1(J,\mathbb{R}^d)$. When using

W_1 with discrete time series, one needs to fix the time dimension of the time series a priori. The signature, as a universal and principled feature, encodes the temporal information regardless of the sampling frequency and variable length. It motivates us to consider using the W_1 metric on the signature space to define a distance between the measure induced by two measures on the path space μ and ν, i.e.

$$W_1^{Sig}(\mu,\nu) = \sup_{\|f\|_{\text{Lip}}\leq 1} \mathbb{E}_{X\sim\mu}[f(S(X))] - \mathbb{E}_{X\sim\nu}[f(S(X))] \tag{4.3}$$

While the use of signature features in W_1^{Sig} significantly reduces the dimension of the time series, practical challenges of solving the min-max problem remain. By working with the signature features, one can reduce the computation of the W_1^{Sig} distance over the class of Lipschitz functionals to the linear functionals on the signature space thanks to the universality property of the signature (Theorem 1.6). Let K be a compact subset of $\Omega_0^1(J,E)$ and μ a measure defined on K. For any X sampled from μ, the expected signature of X under the measure μ is well defined and determines the measure on K uniquely. By the definition of $W_1(\mu,\nu)$, there exists a sequence of $f_n : K \to \mathbb{R}$ with bounded Lipschitz norm to attain the supremum $W_1(\mu,\nu)$. By the universality of the signature, it implies that $\forall\varepsilon > 0$, for each f_n, there exists a linear functional $L_n : T((E)) \to \mathbb{R}$ to approximate f_n uniformly, i.e.

$$\left| \int_K f_n(x)\mu(dx) - f_n(x)\nu(dx) - \left(\int_K L_n(S(x))\mu(dx) - L_n(S(x))\nu(dx) \right) \right| \leq 2\varepsilon$$

This leads to the Signature Wasserstein-1 metric by restricting the admissible set of f to be linear functionals $L : T((E)) \to \mathbb{R}$ as follows

Definition 4.2 (Sig-W_1 metric)**.** For two measures μ, ν with compact support K on the signature space $\mathscr{S}(\Omega_0^1(J,\mathbb{R}^d))$, the Signature Wasserstein-1 metric between μ

and ν is defined by

$$\text{Sig-W}_1(\mu,\nu) := \sup_{L \text{ is linear, } \|L\|_{\text{Lip}} \leq 1} \mathbb{E}_{\mathbf{X}\sim\mu}[L(S(X))] - \mathbb{E}_{\mathbf{X}\sim\nu}[L(S(X))]. \tag{4.4}$$

Hence, by using Sig-W_1 we changed the nonlinear optimisation task to a linear problem with constraints.

In practice, one needs to truncate the infinite dimensional signature to a finite degree for numerical computation of W_1^{Sig} and $\text{Sig-}W_1(\mu,\nu)$. The factorial decay of the signature enables us to approximate the signature in Eqn. (4.4) by its truncated signature up to degree M for a sufficiently large M, which is given in the following lemma:

Lemma 4.2. *Let $L : T((E)) \to \mathbb{R}$ be a bounded linear functional, and K be a compact set of the range of the signature of a path in $V_1(J,E)$. For any $\varepsilon > 0$, there exists an integer $M > 0$,*

$$\sup_{x\in K} |L(x) - L(\Pi_M(x))| \leq \varepsilon. \tag{4.5}$$

Proof. By the factorial decay of the signature (Lemma 1.7), for any $x \in K$, there exists $l \in V_1(J,E)$,

$$|x - \Pi_M(x)| \leq \sum_{m\geq M} |\pi_m(x)| \leq \sum_{m\geq M} \frac{||l||_{1-var}^m}{m!} \leq \frac{||l||_{1-var}^{M+1}}{(M+1)!}. \tag{4.6}$$

As K is a compact set, therefore $\tilde{l} := \sup_{S(l)\in K} ||l||_{1-var}$ is bounded. It follows that

$$\lim_{M\to\infty} \frac{\tilde{l}^{M+1}}{(M+1)!} = 0,$$

which concludes the proof. □

Therefore we propose the definition

Definition 4.3 (Sig-W_1 metric of degree M)**.** The truncated Sig-$W_1(\mu,\nu)$ metric up to a degree M is given as follows:

$$\text{Sig-W}_1^{(M)}(\mu,\nu) := \sup_{L \text{ is linear, } \|L\|_{\text{Lip}} \leq 1} L(\mathbb{E}_{\mathbf{X}\sim\mu}[S_M(X)] - \mathbb{E}_{\mathbf{X}\sim\nu}[S_M(X)]). \tag{4.7}$$

To derive the analytical formula for the Sig-W_1 metric, we need the following lemma, which enables us to simplify the Lip-norm[1] of linear functionals to the L_2 norm.

Lemma 4.3. *Let $a \in T((E))$ be an element of the extended tensor algebra. For any linear functional $L \in T((E))^*$, we have*

$$\sup_{||a||_{L_2}=1} |La| = ||L||_{L_2}. \tag{4.8}$$

Similarly, it holds that

$$\sup_{||L||_{L_2}\leq 1} |La| = ||a||_{L_2}. \tag{4.9}$$

Proof. When $L = 0$, it is trivial as $|La| = ||L||_{L_2} = 0$. When $L \neq 0$, we use the Lagrange multiplier to prove the lemma. To solve the optimization problem, the Lagrange function is

$$\mathscr{L}(a,\lambda) = |La| + \lambda(||a||_{L_2} - 1). \tag{4.10}$$

We can write $a = \sum_I a_I e_I$ and $L = \sum_I l_I e_I^*$, where $(a_I)_I$ and $(l_I)_I$ are real values, $(e_I)_I$ and $(e_I^*)_I$ are the basis of $T((E))$ and $T((E))^*$ respectively. Then we solve the

[1] We let the metric $d = ||x-y||_{L_2}$ in the Lip-norm for any $x, y \in T((E))$.

following equations

$$\frac{\partial \mathscr{L}}{\partial a_I} = \operatorname{sgn}(a_I l_I) l_I + 2\lambda a_I = 0, \tag{4.11}$$

$$\frac{\partial \mathscr{L}}{\partial \lambda} = \sum_I a_I^2 = 1. \tag{4.12}$$

The optimal solution is

$$a_I^* = \frac{\operatorname{sgn}(l_I)|l_I|}{(\sum_I l_I^2)^{\frac{1}{2}}} \tag{4.13}$$

$$\lambda^* = \frac{(\sum_I l_I^2)^{\frac{1}{2}}}{2}. \tag{4.14}$$

Thus, it leads to the optimal value

$$\sup_{||a||_{L_2}=1} |La| = |La^*| = |\sum_I l_I a_I^*| = \frac{\sum_I |l_I|^2}{(\sum_I l_I^2)^{\frac{1}{2}}} = ||L||_{L_2}. \tag{4.15}$$

To prove the Equation (4.9), we only consider the case when $a \neq 0$ as for $a = 0$ it is trivial. Similarly, when $||L||_{L_2} = 1$, by the Lagrange multiplier, the optimal value for l_I is

$$l_I^* = \frac{\operatorname{sgn}(a_I)|a_I|}{(\sum_I a_I^2)^{\frac{1}{2}}} \tag{4.16}$$

such that $\sup_{||L||_{L_2}=1} |La| = ||a||_{L_2}$. By Hölder's inequality,

$$\sup_{||L||_{L_2}\leq 1} |La| \leq \sup_{||L||_{L_2}\leq 1} ||L||_{L_2} ||a||_{L_2} \leq ||a||_{L_2}, \tag{4.17}$$

which implies that $\sup_{||L||\leq L_2=1} |La| = ||a||_{L_2}$. □

By exploiting the linearity of the linear functional L, we can obtain the Lip

norm is equal to L_2 norm for L by Lemma 4.3 as follows

$$||L||_{Lip} = \sup_{x\neq y, x,y\in T((E))} \frac{|Lx-Ly|}{||x-y||_{L_2}} = \sup_{||a||_{L_2}=1} |La| = ||L||_{L_2}. \tag{4.18}$$

Then we can achieve the following theorem for the analytical formula of the Sig-W_1 metric.

Theorem 4.4. *For two measures μ, ν with a compact support K on the signature space $\mathbf{S}(\Omega_0^1(J,\mathbb{R}^d))$, the analytical formula of Sig-W_1 metric between μ and ν is given as follows*

$$\textit{Sig-}W_1(\mu,\nu) = ||\mathbb{E}_\mu[S(X)] - \mathbb{E}_\nu[S(X)]||_{L_2}. \tag{4.19}$$

Moreover, Sig-W_1 metric of degree M is given by

$$\textit{Sig-}W_1^{(M)}(\mu,\nu) = ||\mathbb{E}_\mu[S_M(X)] - \mathbb{E}_\nu[S_M(X)]||_{L_2}. \tag{4.20}$$

Proof. Let $a := \mathbb{E}_\mu[S(X)] - \mathbb{E}_\nu[S(X)]$ and $a = (a_I)_I$. Then by Lemma 4.3 , one can derive the analytic formula of Sig-W_1 metric as follows:

$$\text{Sig-}W_1(\mu,\nu) = \sup_{||L||_{Lip}\leq 1} L(\mathbb{E}_\mu[S(X)] - \mathbb{E}_\nu[S(X)]) = \sup_{||L||_{L_2}\leq 1} L(a) = ||a||_{L_2}$$

where $L = \sum_I l_I e_I^*$. □

With the optimal solution (4.19), we can efficiently compute the metric without implementing the optimization procedure. In [93], if one chooses the truncated signature up to degree M as the feature map, then the corresponding Maximum Mean Discrepancy (Sig-MMD) is the square of Sig-$W^{(M)}(\mu,\nu)$.

The following toy example illustrates the relationship between the Sig-W_1 distance and the W_1 distance between two path distributions.

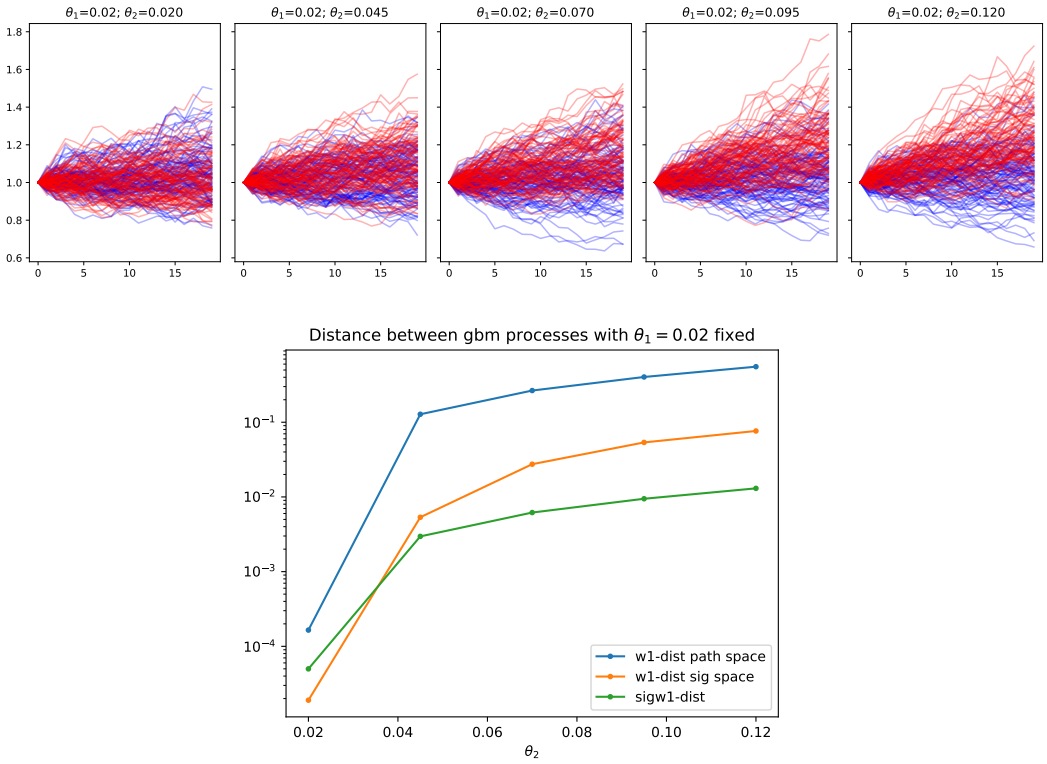


Figure 4.1: The top row displays blue and red samples from $\mathbf{X}, \hat{\mathbf{X}}$ respectively for fixed θ_1, and different values of θ_2

Example 4.1. Let $X = (X_t)_{t\in[0,T]}, \hat{X} = (\hat{X}_t)_{t\in[0,T]}$ be two 1-dimensional GBMs given by,

$$dX_t = \theta_1 X_t dt + \sigma X_t dW_t,\ X_0 = 1;$$
$$d\hat{X}_t = \theta_2 \hat{X}_t dt + \sigma \hat{X}_t d\hat{W}_t,\ \hat{X}_0 = 1,$$

with the same volatility but with possibly different drifts θ_1, θ_2. Let μ, ν be the laws of $X, \hat{X}$. We fix $\sigma = 0.1, \theta_1 = 0.02$, and for $\theta_2 = 0.02 + j0.025, j = 0, \ldots, 4$. When θ_2 is increasing, the discrepancy between X and $\hat{X}$ is increasing. We calculate three distances, i.e. W_1, W_1^{Sig} and Sig-W_1 as shown in Figure 4.1 to quantify the distance between X and $\hat{X}$ for different θ_2, which all increase when enlarging θ_2 as expected. Since Sig-$W_1^{(M)}(\mu, \nu)$ admits an analytic solution, it is cheaper to calculate than

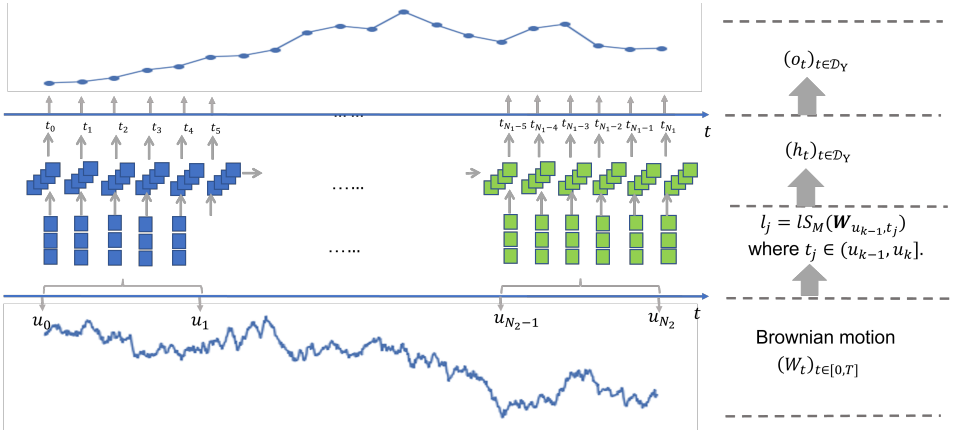


Figure 4.2: The generalized Logsig-RNN as generator in Sig-WGAN.

$W_1(\mu,\nu)$ and $W_1^{Sig}(\mu,\nu)$, where one needs to parametrise f by a neural network and optimise its weights. We observe in Figure 4.1 how these three values increase with similar rate as θ_2 increases.

4.2.3 Generator

In GAN variants, generators are fed with random noise. Suppose the target time series is of length N and dimension e, i.e. in $\mathbb{R}^{e\times N}$. Fix a probability space $(\Omega,\mathscr{F},\mathbf{P})$, under which $W=(W_t)_{t\in[0,T]}$ be a d-dimensional Brownian motion. Let $\mathbf{W}=(\mathbf{W}_t)_{t\in[0,T]}$ denote the time-augmented Brownian motion, where $\mathbf{W}_t=(t,W_t)$ for ease of the notation. For this chapter, we propose the generalized Logsig-RNN (Figure 4.2) generator. Given two time partitions $\mathscr{D}=(u_k)_{k=0}^n \subset \mathscr{D}_Y=(t_j)_{j=0}^N$ of the interval J, we consider the generalized Logsig-RNN network $\bar{H}:=\bar{H}_{M,\Theta}^{\mathscr{D},\mathscr{D}_Y}$ as defined in Model 2.2 and use it as the generator mapping from a d-dimensional Brownian motion $\mathbf{W}$ to a path in $\mathbb{R}^{e\times N}$.

4.3 Numerical results

To validate the performance of the generalized Logsig-RNN trained with both Sig-WGAN and WGAN, we consider three datasets: (1) synthetic data generated by multi-dimensional Geometric Brownian motion (GBM); (2) synthetic data gener-

ated by the rough volatility model; (3) eICU data. The former two datasets are representatives of commonly-used Markovian and non-Markovian model for an underlying price process. For each dataset, we benchmark the generalized Logsig-RNN generators of continuous type against the LSTM and NRDE.

To assess the quality of the generated data, we give precise definitions of the test metrics in the following. More specifically, we let $(X_{t_i})_{i=0}^N$ and $(\hat{X}_{t_i})_{i=0}^N$ to be $\mathbb{R}^{N\times d}$-valued random variables of the real and generated paths respectively. Except for the Sig-W_1 metric, we consider three main criteria: (a) the marginal distribution of time series; (b) the temporal dependence; (c) the usefulness [37] - generated data should be as useful as the real data when used for the same predictive purposes (i.e. train-on-synthetic, test-on-real (TSTR)).

- **Marginal distribution metric**. Following [94], at every time step $i \in \{1,\dots,N\}$, for each feature dimension $j \in \{1,\dots,d\}$, we compute two empirical density functions based on the histograms of the real data and synthetic data respectively denoted by $\hat{df}_r^{i,j}$ and $\hat{df}_G^{i,j}$. We take the absolute difference of those two epdfs as the metric on marginal distribution averaged over feature dimensions and time steps, i.e.,

$$\frac{1}{Nd}\sum_{i=1}^{N}\sum_{j=1}^{d}|\hat{df}_r^{i,j}-\hat{df}_G^{i,j}|.$$

- **Correlation metric**. To quantify the fitting of the spatial and temporal dependence, we consider how close the correlation of $X_t^{(i)}$ and $X_s^{(j)}$ for any feature coordinate i and j, and any time s and t. The correlation metric of X and $\hat{X}$ is defined as

$$\mathrm{cor}(X,\hat{X}) = \sum_{p,q=1}^{N}\sum_{i,j=1}^{d}\left|\rho(X_{t_p}^{(i)},X_{t_q}^{(j)})-\rho(\hat{X}_{t_p}^{(i)},\hat{X}_{t_q}^{(j)})\right|,$$

where $\rho(X,Y)$ denotes the correlation of two real-valued random variables X and Y.

- **TRTR/TSTR**. We establish classification tasks for the dataset. We train a random forest classifier on the generated data and then test the model on the out-of-sample data. We compare the testing accuracy with that of the model trained on the real data. This is used on the eICU data, for which we can form classification problems.

Moreover, to examine the robustness of each method, we design three experiments on the synthetic data: first, we vary the target length of output to test the robustness in generating the long time series; by feeding the models with various length of inputs, we investigate their ability in handling high-frequency time series; lastly, we test the predictive performance of the generative models, which are trained on the dataset of one time frequency and are tested on another time frequency. The better fitting performance on the test time frequency shows the robustness and generalization of the trained generative model against variation in sampling time.

4.3.1 Multi-dimensional geometric Brownian motion (GBM)

As a motivating example, we consider a d-dimensional GBM $X = (X_t)_{t\in[0,T]}$ satisfying the following SDE:

$$dX_t = \mu X_t dt + \sigma X_t d\tilde{W}_t, \tag{4.21}$$

where $X_t \in \mathbb{R}^d$, $\mu \in \mathbb{R}^d$, σ is a $d \times d$ diagonal matrix, $\tilde{W}$ is a d-dimensional correlated Brownian motions with correlation matrix $(\rho_{ij})_{i,j\in\{1,\cdots,d\}}$. In this example, we set $\mu^i = 0.1$, $\sigma^{i,i} = 0.2$, $\forall i \in \{1,...,d\}$ and $\rho_{ij} = 0.5$, for $i \neq j$. We use the equally spaced time partition of the time interval $[0,1]$ with step size $\Delta t = 10^{-3}$, and simulate 4000 samples of the solution X to the SDE (4.21) using the analytic

formula:

$$X_{t_n}^{(i)} = X_{t_{n-1}}^{(i)} \exp[(\mu_i - \frac{1}{2}\sigma_{i,i}^2)\Delta t + \sigma(\tilde{W}_{t_n}^i - \tilde{W}_{t_{n-1}}^i)], \tag{4.22}$$

where $t_n = n\Delta t$ and $i \in \{1, \cdots, d\}$.

For a fair comparison, we train the Wasserstein GAN and the Sig-WGAN for 2 500 iterations of the generator. The Sig-WGAN provides an explicit form for the Sig-W_1 distance between two path distributions, which implies that the overall number of gradient steps is 2 500. However, training the Wasserstein GAN involves training the generator and the discriminator; in our settings, for every gradient step of the generator, we take three gradient steps on the discriminator, thus the overall number of gradient steps is 10 000.

Sensitivity analysis in terms of the length of real data. In Table 4.1, we compare the performance of different generators under varying time steps of the real data. It highlights that the performance of the LogsigRNN generator is much more robust than the LSTM in terms of the high frequency sampling of the data in all evaluation metrics. The LSTM works well when N is small (e.g. 10 or 20) with both WGAN and Sig-WGAN. However, when increasing the number of time step N, it gets more challenging for the LSTM to perform well in all the evaluation metrics, especially for the correlation metric which describes the spatial-temporal dependency of the time series. In contrast, combined with the Sig-WGAN, the Logsig-RNN generator maintains in narrow range of $[0.045, 0.050]$ and $[0.919, 1.032]$ for the correlation metric and marginal distribution for each $N \in \{10, 20, 50, 100\}$ respectively. For $N = 100$, when using the SigWGAN, the test metrics of the LSTM generator is about four times of that of the Logsig-generator. Compared with the NRDE, the Logsig-RNN outcompetes it in most cases, especially for the correlation metric when trained with Sig-WGAN. Regarding the training time, Logsig-RNN is consistently faster than the NRDE, which reflects the fact that solving the ODE systems

for NRDE is time consuming. For a fixed generator, we observe the consistent performance improvement of using Sig-W_1 over the W_1 metric as the discriminator.

	W_1				Sig-W_1			
N	10	20	50	100	10	20	50	100
SigW_1 Metric (1e-1)								
LSTM	**0.140**	**0.125**	0.677	0.651	0.096	0.064	**0.020**	0.196
LogsigRNN	0.264	0.128	**0.153**	**0.160**	**0.047**	0.040	0.042	**0.037**
NRDE	0.186	0.133	0.154	0.184	0.055	**0.031**	0.034	0.046
Correlation Metrics (1e-2)								
LSTM	**1.057**	**0.862**	5.512	2.658	**0.478**	**0.852**	0.969	3.391
LogsigRNN	2.125	1.811	**1.658**	**2.219**	0.962	1.032	**0.967**	**0.919**
NRDE	2.265	1.749	1.832	2.470	1.148	1.896	1.314	1.454
Marginal Distribution Metric (1e-1)								
LSTM	**0.052**	0.065	0.132	0.082	**0.026**	**0.043**	**0.036**	0.140
LogsigRNN	0.122	**0.059**	**0.104**	**0.067**	0.050	0.047	0.050	0.045
NRDE	0.106	0.072	0.117	0.074	0.089	0.074	0.048	**0.041**
Training time (seconds)								
LSTM	104	194	561	1075	98	217	476	908
LogsigRNN	324	606	1379	2586	187	334	792	1451
NRDE	831	1109	2013	3369	357	484	942	1755

Table 4.1: Evaluation for 3-dimensional GBM with various numbers of timesteps $N \in \{10, 20, 50, 100\}$.

Sensitivity analysis in terms of the length of inputs. In Table 4.2, we compare the performance of different generators under varying length of inputs to investigate their ability in dealing with high-frequency data. We generate outputs of 50 steps with the length of inputs in $\{500, 1000, 2000, 5000\}$. In contrast to the LSTM, the generalized Logsig-RNN performs consistently better on the Sig-W_1 and correlation metrics when trained with WGAN, and they both achieve comparable results in other cases. The training time of the Logsig-RNN increases much more mildly than that of LSTM. Specifically, with the WGAN, the training time of the LSTM increases by 363% from 500 to 5000 steps of inputs, while it increases by 188% for the Logsig-RNN; with the Sig-WGAN, the training time of the LSTM increases by 223%, while it only increases by 73% for the Logsig-RNN. It validates the advantage of the Logsig-RNN against classical RNNs in efficiently handling long length

input data. When compared with the NRDE, the Logsig-RNN works especially better on the correlation metrics, which is consistent with results in Table 4.1; it implies that the Logsig-RNN is more powerful than NRDE in capturing the spatial-temporal dependency of the time series. Regarding the training efficiency, although the NRDE increases mildly given longer inputs, the Logsig-RNN is always faster in all cases.

	W_1				Sig-W_1			
Noise Length	500	1000	2000	5000	500	1000	2000	5000
SigW$_1$ Metric (1e-1)								
LSTM	0.629	0.434	0.548	0.570	0.037	0.041	0.045	0.107
LogsigRNN	0.194	**0.153**	**0.129**	0.161	**0.034**	0.042	0.057	**0.055**
NRDE	**0.146**	0.154	0.146	**0.108**	0.066	**0.034**	**0.042**	0.209
Correlation Metrics (1e-2)								
LSTM	2.924	2.712	1.890	1.779	0.887	**0.706**	**0.911**	0.977
LogsigRNN	**1.025**	**1.658**	1.490	**0.959**	**0.695**	0.967	0.940	**0.864**
NRDE	1.490	1.832	**1.436**	1.662	1.255	1.314	0.982	3.732
Marginal Distribution Metric (1e-1)								
LSTM	**0.092**	**0.088**	0.084	0.093	**0.046**	0.051	**0.038**	**0.045**
LogsigRNN	0.097	0.104	0.105	0.086	0.060	0.050	0.044	0.059
NRDE	0.126	0.117	**0.073**	**0.075**	0.053	**0.048**	0.048	0.734
Training time (seconds)								
LSTM	618	930	1205	2861	645	806	1175	2084
LogsigRNN	1128	1379	1799	3253	764	792	913	1324
NRDE	1561	2013	2845	4660	826	942	1115	1675

Table 4.2: Evaluation for 3-dimensional GBM of 50 timesteps with various length of input noise in $\{500, 1000, 2000, 5000\}$.

4.3.2 Rough volatility model

We consider a rough stochastic volatility model for an asset price process S_t, which satisfies the below SDE:

$$\begin{aligned} dS_t &= \sqrt{v_t} S_t dZ_t \\ v_t &:= \xi(t) \exp\left(\eta W_t^H - \frac{1}{2}\eta^2 t^{2H}\right) \end{aligned} \tag{4.23}$$

where $\xi(t)$ denotes the forward variance and W_t^H denotes the fBM given by

$$W_t^H := \int_0^t K(t-s)dW_s, \quad K(r) := \sqrt{2H}r^{H-1/2}$$

where Z_t, W_t are (possibly correlated) Brownian motions. In our experiments, the synthetic dataset is sampled from (4.23) with $t \in [0,1], H = 0.25, \xi(t) = 0.25, \eta = 0.5, \rho = 0$ where $\rho dt = d\langle W, Z\rangle_t$. Each sampled path is a stream of data of 20 points sampled uniformly between $[0,1]$.

For the training details, we train the generators to learn

- the log price distribution,
- the log price and the log volatility joint distributions

using WGAN and Sig-WGAN. In both learning algorithms we scale the log-price paths by 2 and the log-volatiliy by 0.5 so that they have similar variance. In addition, in the Sig-WGAN we augment the paths by adding time dimension and visibility transformation before computing the expected signature up to degree 4. We train the generators for 2500 steps on both Sig-WGAN and WGAN.

Table 4.3 shows the results of the evaluation metrics of the trained generators. The generalized Logsig-RNN consistently beats the other two generators in all evaluation metrics when trained with Sig-WGAN. Although LSTM with WGAN gains a small edge for generating (S_t, v_t), Logsig-RNN with WGAN outcompetes it by 53% and 51% on correlation and marginal distribution metric respectively. Compared with the NRDE, the Logsig-RNN outperforms on almost all cases except being slightly defeated on the marginal distribution metric for generating the 2-dim time series with WGAN. It implies the Logsig-RNN can generate overall higher-quality data than the benchmarks. Regarding the training time, the LSTM is the fastest, which is because the input length (20) is much shorter than that of the other

	W_1		Sig-W_1	
Data $(X_t)_t$	(S_t)	(S_t, v_t)	(S_t)	(S_t, v_t)
SigW$_1$ Metric				
LSTM	0.29	**0.42**	**0.07**	0.17
LogsigRNN	**0.20**	0.49	0.11	**0.15**
NRDE	0.24	0.58	0.19	0.46
Correlation Metric (1e-3)				
LSTM	4.76	**5.21**	1.58	8.23
LogsigRNN	**2.19**	5.59	**1.09**	**1.83**
NRDE	3.25	7.44	1.96	5.05
Marginal Distribution Metric (1e-2)				
LSTM	2.37	**1.95**	1.62	4.83
LogsigRNN	**1.17**	3.74	**1.38**	**2.97**
NRDE	3.76	3.15	2.33	5.42
Training time(seconds)				
LSTM	158	205	164	256
LogsigRNN	814	845	352	452
NRDE	1520	1613	557	611

Table 4.3: The test metrics of the trained models on a one dimensional price data $(S_t)_{t\in[0,T]}$ and two dimensional price and volatility data $(S_t, v_t)_{t\in[0,T]}$ respectively.

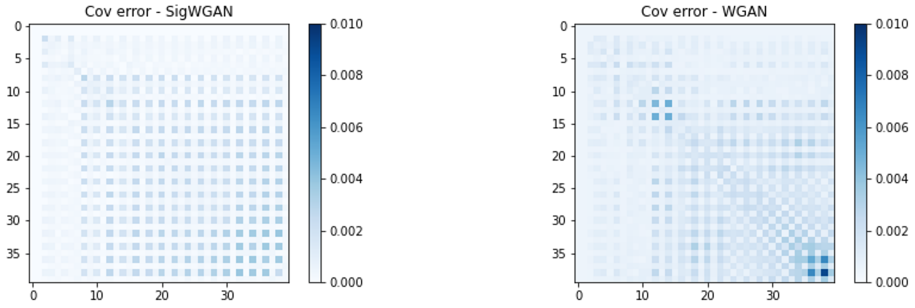
two (1000). The Logsig-RNN is more efficient than the NRDE in all cases, which is consistent with those of other experiments.

Correlation metric comparison. The error plot of the covariance matrix

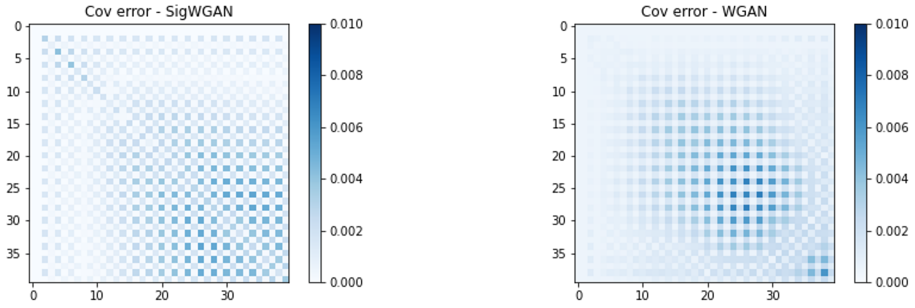
$$\left(\mathrm{cov}(X_s^{(i)}, X_t^{(j)})\right)_{i,j\in\{1,2\},s,t\in\{1,\cdots,N\}}$$

in Figure 4.3 clearly shows that the combination of the Logsig-RNN and SigW$_1$ metric are able to learn the temporal and spatial dependency of the rough volatility data best. When trained with WGAN, Logsig-RNN has similar performance as LSTM and NRDE. It highlights that when combined with the efficient Sig-WGAN framework, Logsig-RNN can generate data with realistic-looking spatial-temporal coherence.

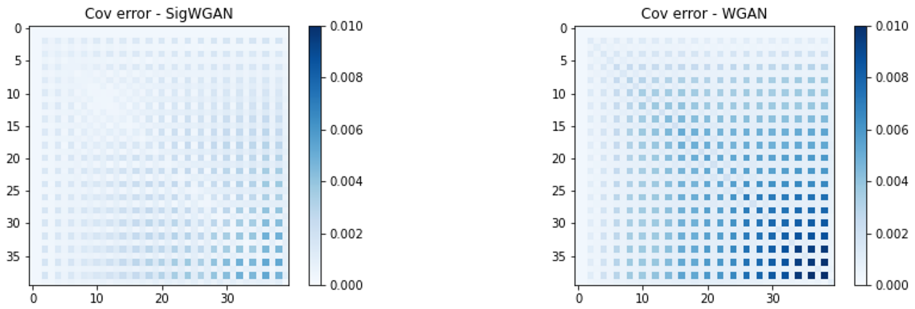
Robustness to the variable sampling frequency. We test the robustness of the trained generator by generating data streams with a lower/higher frequency than the one used for training and we test them on synthetic data generated from (4.23)



(a) LogsigRNN



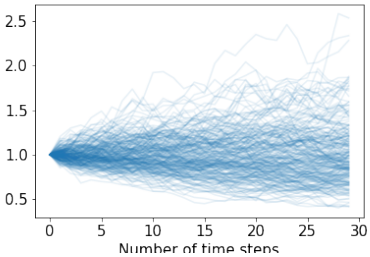
(b) LSTM



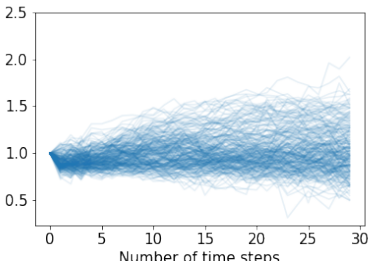
(c) NRDE

Figure 4.3: The three figures are the covariance error plots of the Logsig-RNN/LSTM/NRDE generators for each dimension (S_t, v_t) and each timestep. From Left to Right, each heatmap displays the covariance error of the Sig-WGAN and the WGAN respectively.

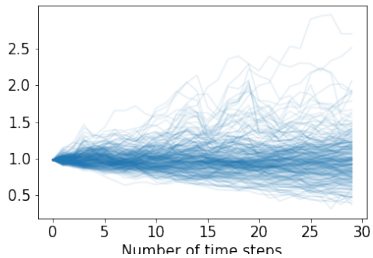
with the same new frequency. Namely, we generate data streams with length in $\{5, 10, 30, 40\}$ on $[0, 1]$, whilst training was performed on data streams of 20 points on $[0, 1]$. Table 4.4 displays that the Logsig-RNN generator outperforms LSTM and



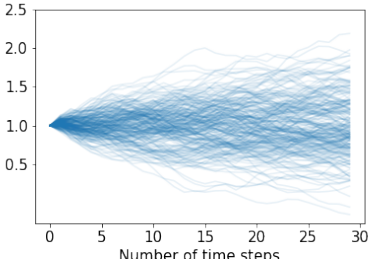
(a) Real paths of new frequency



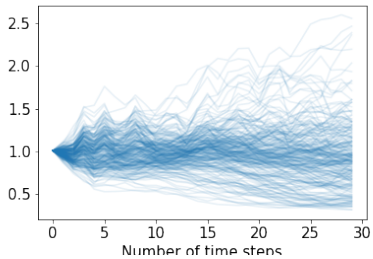
(b) Logsig-RNN+Sig-WGAN



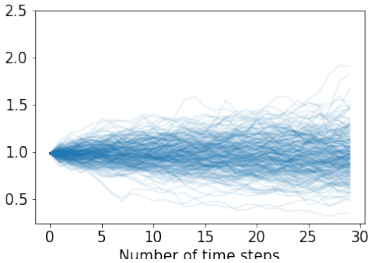
(c) Logsig-RNN+WGAN



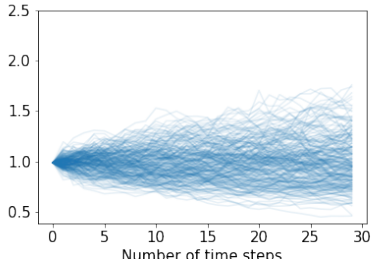
(d) LSTM+Sig-WGAN



(e) LSTM+WGAN



(f) NRDE+Sig-WGAN



(g) NRDE+WGAN

Figure 4.4: Comparison of the generated paths under new frequency (i.e. 30 time steps). (a) The real paths with new frequency; (b-c) generated path by the generalized Logsig-RNN; (d-e) generated path by the LSTM; (f-g) generated path by the NRDE.

	W_1				Sig-W_1			
Test Length	5	10	30	40	5	10	30	40
SigW$_1$ Metric								
LSTM	0.793	0.808	0.438	0.418	0.389	0.379	0.421	0.472
LogsigRNN	**0.656**	**0.614**	**0.420**	**0.417**	**0.310**	**0.340**	**0.155**	**0.414**
NRDE	0.937	0.621	0.727	0.502	1.621	0.590	0.491	0.506
Correlation Metrics (1e-3)								
LSTM	22.43	19.89	15.97	17.69	22.14	19.28	5.313	**7.605**
LogsigRNN	**13.88**	15.07	**12.32**	**16.72**	**15.82**	**15.99**	**2.130**	8.401
NRDE	40.70	**12.25**	13.92	20.08	118.8	18.02	3.756	11.35
Marginal Distribution Metric								
LSTM	**0.227**	**0.213**	0.390	**0.328**	0.529	0.436	0.334	0.453
LogsigRNN	0.273	0.232	**0.315**	0.407	0.319	0.320	**0.128**	**0.412**
NRDE	0.296	0.245	0.326	0.498	**0.317**	**0.293**	0.187	0.452

Table 4.4: The test metrics of the trained models on two dimensional price and volatility data $(S_t, v_t)_{t\in[0,1]}$. Models are trained on data streams sampled every $\frac{1}{20}$ units of time, and evaluated on data streams sampled every $\frac{1}{N}$ units of time, where $N \in \{5, 10, 30, 40\}$.

NRDE on the Sig-W_1 metric and correlation metric consistently and have comparable results on the marginal distribution metric. It validates the trained Logsig-RNN generator is more robust than LSTM and NRDE regarding producing data with varying frequencies. It is also verified qualitatively and visually by comparing the sample trajectories of the real paths (30 time steps) and the synthetic path generated by the different generators trained with Sig-WGAN and WGAN in Figure 4.4. The paths generalized Logsig-RNN are visually closer to the real paths of new frequency.

4.3.3 eICU data

The eICU[2] data is a medical type data monitoring patients' biomedical information during their stay in ICU. This dataset was collected by the critical care telehealth program provided by Philips, and it contains 200,000 patients from 208 care units across the US. We conduct experiments on the four most frequently sampled vital signs by the bedside monitor: oxygen saturation measured by pulse oximeter

[2]https://eicu-crd.mit.edu/

(SaO2), heart rate (HR), respiratory rate (RR) and mean arterial pressure (MAP). We implement the same data preprocessing as that in [36]. We consider the sample frequency of every fifteen minutes and take the median value of each window. For the temporal length of data, we focus on the first four hours of patients staying, which leads to sequential data of 16 steps. To avoid the missing data issue, we drop out the patients with missing values. After preprocessing the data this way, we end up with a dataset of 17,693 patients.

The ICU data is visually complex and challenging for nonprofessionals to capture useful patterns from it. Thus, apart from the comparison of test metrics in Table 4.5, we evaluate the model performance by TSTR method the same as [36]. To set up the supervised task for evaluating the TSTR method, [36] defines thresholds for vital signs and generates binary labels of whether or not that sign would exceed the threshold in the next hour of the patient's stay. As we use the first four hours as the observed data, the hour between four and five is where the excess of thresholds is checked. The thresholds are shown in Table 4.6. We evaluate different methods by training a random forest classifier[3] on their generated data and comparing the values of the area under the ROC curve (AUROC) and area under the precision-recall curve (AUPRC).

	W_1	Sig-W_1
SigW_1 Metric (1e-1)		
LSTM	5.52	5.38
LogsigRNN	**5.37**	**5.24**
NRDE	5.45	5.41
Correlation Metric (1e-2)		
LSTM	**1.13**	1.23
LogsigRNN	1.30	**1.03**
NRDE	1.19	1.73

	W_1	Sig-W_1
Marginal Distribution Metric		
LSTM	0.19	0.33
LogsigRNN	0.17	**0.19**
NRDE	**0.16**	0.22
Training Time (seconds)		
LSTM	**73**	**68**
LogsigRNN	202	107
NRDE	292	138

Table 4.5: eICU data generation under W_1 and SigW_1 metrics.

In the implementations, for Sig-WGAN, we augment the paths by visibility

[3]The setting of the classifier follows [36], which is given in `https://github.com/ratschlab/RGAN/`.

transformation when computing the expected signature up to level 4. For both WGAN and Sig-WGAN, we train the generator for 2000 epochs. In Table 4.5, we compare the performance of trained generators under evaluation metrics. Logsig-RNN outperforms the other two on Sig-W_1 metric. For the marginal distribution metric, Logsig-RNN has comparable results with NRDE, and they both outperform LSTM. They have similar results on correlation metric. Regarding the time efficiency, the Logsig-RNN can be 31% faster than NRDE when with WGAN and 22% faster when with Sig-WGAN. It is consistent with the previous results on synthetic data and highlights that the Logsig-RNN is more efficient than the NRDE method.

		SpO2 $<$ 95	HR $<$ 70	RR $>$ 20
AUROC	real	0.95	0.99	0.96
	RGAN	0.83	0.93	**0.84**
	TimeGAN	0.80	0.94	0.79
	Logsig-RNN+Sig-W_1	0.82	**0.95**	0.82
	LSTM-RNN+Sig-W_1	**0.85**	**0.95**	0.81
	NRDE+Sig-W_1	0.84	0.91	0.81
	Logsig-RNN+W_1	0.80	0.92	0.75
	LSTM-RNN+W_1	0.79	0.94	0.66
	NRDE+W_1	0.77	**0.95**	0.72
AUPRC	real	0.90	0.98	0.89
	RGAN	0.62	0.89	0.50
	TimeGAN	0.63	0.90	0.61
	Logsig-RNN+Sig-W_1	0.63	**0.91**	**0.72**
	LSTM-RNN+Sig-W_1	**0.66**	0.88	0.65
	NRDE+Sig-W_1	0.55	0.87	0.70
	Logsig-RNN+W_1	0.56	0.86	0.61
	LSTM-RNN+W_1	0.49	**0.91**	0.41
	NRDE+W_1	0.41	0.90	0.61

Table 4.6: Performance of random forest classifier for eICU tasks when trained with real data and when trained with synthetic data.

Table 4.6 illustrates the performance of different methods in the TSTR tasks. The first rows of both AUROC and AUPRC shows the results of random forest classifier on the real data. It shows that the Logsig-RNN generator, when trained with the Sig-WGAN, outperforms the RGAN [36] and TimeGAN [37] on most tasks except for the AUROC of 'RR$>$20' column. Although the RGAN outperforms the

Logsig-RNN by 2.5% for AUROC, the Logsig-RNN achieves 22% higher than the RGAN for AUPRC. It implies that for the case of 'RR>20' case, the Logsig-RNN has comparable performance on generating imbalanced data and can generate more realistic positive data given the same training data. For the comparison with the LSTM and NRDE, the Logsig-RNN has overall better performance. In particular, when trained with WGAN, the AUROC of Logsig-RNN is 14% higher than LSTM for 'RR>20', the AUPRC of Logsig-RNN is 20% higher than LSTM for 'RR>20' and 15% higher than NRDE for 'SpO2<95'. We concludes that the Logsig-RNN can generate higher-fidelity data than other generators, and when combined with the Sig-WGAN, it can achieve better performance than other methods.

4.4 Conclusions

In this chapter, we evaluated the generative ability of the generalized Logsig-RNN (Model 2.2) for time series data. Leveraging the characteristic property of the expected signature, we established Sig-WGAN to improve the optimization efficiency of the classical WGAN. With both Sig-WGAN and WGAN, we compared our approach with baseline methods, i.e. the LSTM and NRDE. On synthetic datasets, the generalized Logsig-RNN demonstrates higher robustness than the other two on the tasks of generating long outputs and generating data with different length from the training data. Our method is more time efficient than LSTM when dealing with high-frequency data, and is consistently faster than NRDE on all tasks. The generalized Logsig-RNN demonstrates better performance than the LSTM and NRDE on the evaluation metrics, especially in capturing the spatial-temporal dependency of the time series data. On the eICU dataset, we build up classification problems and evaluated different methods using TSTR technique. The generalized Logsig-RNN with Sig-WGAN outcompetes the state-of-the-art benchmarks in generating realistic-looking time series data.

Chapter 5

Discussion

This thesis proposes a novel and generic module, i.e. the Logsig-RNN, for time series data in the machine learning context. Enlightened by the non-parametric modelling of SDEs and the time series learning ability of classical RNNs, we developed a data-driven model that is capable of managing high-frequency, irregular sampled or re-sampled data. As an enhancement to classical RNNs, the Logsig-RNN exploits the mathematically principle representations of the log-signature to extract features efficiently from high-frequency data, and hence reduce the dimension of RNNs to improve model performance such as accuracy and robustness.

We highlight the advantages of our approach in two major machine learning tasks, i.e. the supervised learning and the time series generation. Our results achieves higher accuracy and robustness against the LSTM in SHAR tasks. Combined with path transformation layers, the PT-Logsig-RNN achieves second to the best result in Chalearn 2013 dataset, while the SOTA results is achieved two years after our publication. On the generation side, our results show that the Logsig-RNN generator can produce more realistic-looking data in a efficient and robust way compared with the LSTM and NRDE benchmarks. For the empirical eICU data, the Logsig-RNN trained with the Sig-WGAN discriminator outcompetes the methods in the literature.

5.1 Limitations and future work

The major drawback of our approach is from the dimensionality of the algebraic space of the signature/log-signature features, which maps the time series to tensor space/free Lie algebra. While the Logsig-RNN reduces the time dimension of high-frequency data, it increases the spatial dimension. For example, in the SHAR cases, given a dataset with 25 joints with 6 channels in each joint, i.e. 150-dim spatially, the degree 4 log-signature would project it to space of 120 million dimensions, which leaves the dimensionality issue unsolved. As illustrated in Chapter 3, we proposed Path Transformation layers to handle it. However, this would depend on the performance of the chosen layers. In the NTU RGB+D 120 dataset, we used idea from GCN-based SOTA models [70, 77] such that the GCN-Logsig-LSTM has 8% improvement from the EL-Logsig-LSTM, which used plain CNN layers. In the future work, it merits further research to improve the combination with GCN-based models to enhance the accuracy while maintaining robustness. Apart from this, it worths investigation of the application of the log-signature in the Graph Neural Network [79] field, where the signature/log-signature serves as a natural features extractor by regarding the trajectory starting from a certain node in the graph as a path. However, it might not be straightforward to define the path, where we need to incorporate domain knowledge of the data such as molecular, traffic map, etc..

On the generative tasks, current work emphasises the combination of the log-signature method with the GAN framework. Recently, the diffusion-based generative models [95, 96] gain popularity in this area given its success such as in image, audio synthesis. By slowly adding noise at each diffusion step of a Markov chain to the data, the model learns the reverse diffusion process to establish the data from the noise. Authors of [97] extend the idea to continuous time modelling, where one can apply the rough path theory in the future work to enhance the performance of the score-based diffusion models.

Appendix A

Auxiliary Properties of Rough Path Theory

Computations of signature and log-signature. To use signature methods in real application, we need to compute it numerically. We explain how to compute the truncated signature of a piecewise linear path. Let us start with a d-dimensional linear path.

Lemma A.1. *Let $X : [S,T] \to E$ be a linear path. Then*

$$S_n(X) = \frac{(X_T - X_S)^{\otimes n}}{n!}. \tag{A.1}$$

Equivalently speaking, for any multi-index $I = (i_1, \cdots, i_n)$,

$$S^I = \frac{\prod_{j=1}^n (X_T^{(i_j)} - X_S^{(i_j)})}{n!} \tag{A.2}$$

Chen's identity is a useful tool to enable compute the signature of the piecewise linear path numerically.

Lemma A.2. *Let X be a E-valued piecewise linear path, i.e.X is the concatenation of a finite number of linear paths, and in other words there exists a positive integer*

L and linear paths $X_1, X_2, ..., X_L$ *such that* $X = X_1 * X_2 * \cdots * X_L$*. Then*

$$S(X) = \otimes_{i=1}^{L} \exp(X_i). \tag{A.3}$$

Lemma A.2 implies that we may compute several exponentials and $\otimes$ to derive the signature of a path. According to [46], the computational complexity can be optimised by the fused multiply-exponentiate for computing the multiplication between $B \in T^{(M)}(E)$ and $\exp(x)$ with $x := (x^1, \ldots, x^d) \in \mathbb{R}^d$, i.e.

$$B \otimes \exp(x) = \left(\sum_{i=0}^{k} B_i \otimes \frac{x^{\otimes(k-i)}}{(k-i)!} \right)_{k=1}^{M}.$$

The fusing is to expand the level k-th term as follows

$$\sum_{i=0}^{k} B_i \otimes \frac{x^{\otimes(k-i)}}{(k-i)!} = \left(\cdots \left(\left(\frac{x}{k} + B_1 \right) \otimes \frac{x}{k-1} + B_2 \right) \otimes \frac{x}{k-2} + \cdots \right) \otimes x + B_k. \tag{A.4}$$

Lemma A.3 ([46])**.** *Let* X *be a* $\mathbb{R}^d$*-valued piecewise linear path that can be written as a concatenation of* L *linear paths* $X_1, X_2, ..., X_L$*. Then the complexity of computing the truncated signature of* X *of degree* M *is*

$$\sum_{k=2}^{M} \left(d + \binom{d+k-1}{k} \right) + (L-1) \left(d(M-1) + \sum_{k=1}^{M} \sum_{i=2}^{k} d^i \right). \tag{A.5}$$

Next, we state the complexity of the backpropagation of the signature of a path by the following lemma.

Lemma A.4. *Let* X *be a* $\mathbb{R}^d$*-valued piecewise linear path that can be written as a concatenation of* L *linear paths* $X_1, X_2, ..., X_L$*. The complexity of the backpropaga-*

tion of the truncated signature of X of degree M is

$$(L-1)d\left(d(M-1)+4\sum_{k=1}^{M}\sum_{i=2}^{k}d^i\right). \tag{A.6}$$

Proof. For simplicity, we denote

$$S_i := \left(\left(\left(\frac{x}{k}+B_1\right)\otimes\frac{x}{k-1}+B_2\right)\otimes\frac{x}{k-2}+\cdots\right)\otimes\frac{x}{k-i+1}+B_i,$$

for $i\in\{1,\ldots,k\}$. Denote $\frac{\mathbf{1}}{\mathbf{k}}_j=(0,\ldots,\frac{1}{k},\ldots,0)$ for 1 has index $j\in\{1,\ldots,d\}$. We can derive the derivative of S_k w.r.t x^i in the style of (A.4) as follows

$$\frac{\partial S_k}{\partial x^i}=S_{k-1}\otimes\mathbf{1}_i+\left(S_{k-2}\otimes\frac{\mathbf{1}}{\mathbf{2}}_i+\left(\cdots\left(S_1\otimes\frac{\mathbf{1}}{\mathbf{k-1}}_i+(\frac{\mathbf{1}}{\mathbf{k}}_i)\otimes\frac{x}{k-1}\right)\cdots\right)\otimes\frac{x}{2}\right)\otimes x$$

It leads to the complexity of the backpropagation of the signature

$$(L-1)d\left(d(M-1)+4\sum_{k=1}^{M}\sum_{i=2}^{k}d^i\right).$$

□

Different from the computation of signature in the Lie group, the log signature lives in the Lie algebra. Let's start with a linear path. The log signature of a linear path X_J is nothing else, but the increment of the path $X_T - X_S$.

There are two ways of computing the log-signature of a path [98]. The first is to use the Baker-Cambpell-Hausdorff (BCH) formula directly. BCH formula gives a general method to compute the log-signature of the concatenation of two paths, which uses the multiplicativity of the signature and the free Lie algebra. It provides a way to compute the log-signature of the piecewise linear path by induction.

Theorem A.5. *[99] Let A be the set of alphabet, A^* be the set of words. For any*

$S_1, S_2 \in \mathscr{L}((E))$

$$Z = log(e^{S_1}e^{S_2}) = \sum_{\substack{n\geq 1 \\ p_1,\ldots,p_n\geq 0 \\ q_1,\ldots.q_n\geq 0 \\ p_i+q_i>0}} \frac{(-1)^{n+1}}{n} \frac{1}{p_1!q_1!\ldots p_n!q_n!} r(S_1^{p_1}S_2^{q_1}\ldots S_1^{p_n}S_2^{q_n}) \quad \text{(A.7)}$$

where $r : A^* \to A^*$ *is the right-Lie-bracketing operator, such that for any word* $w = a_1\ldots a_n$

$$r(w) = [a_1, \ldots, [a_{n-1}, a_n]\ldots].$$

This version of BCH is sometimes called the Dynkin's formula.

The second way is to compute the log-signature from the signature, which is simpler and is the choice of the *signatory* package [46] that we use for our experiments. The first step is to apply the logarithm map (1.7) to the signature and obtain the log-signature in tensor space. The next step is to project it into a basis of the free Lie algebra. A typical choice is the *Lyndon* basis [98]. The author of [46] found a more computational efficient basis such that the projection to one basis element only requires extracting the coefficient of that Lyndon word from the log-signature in tensor space. We can derive the computational complexity of log-signature as follows

Lemma A.6. *Let* X *be a* $\mathbb{R}^d$*-valued piecewise linear path that can be written as a concatenation of* L *linear paths* $X_1, X_2, \ldots, X_L$*. Then the complexity of computing the truncated log-signature of* X *of degree* M *is*

$$\sum_{k=2}^{M}\left(d + \binom{d+k-1}{k}\right) + (L-1)\left(d(M-1) + \sum_{k=1}^{M}\sum_{i=2}^{k} d^i\right) + M\sum_{k=1}^{M} kd^k. \quad \text{(A.8)}$$

Proof. The computational complexity of the log-signature of a path only needs to

include additional multiplications of the logarithm map, i.e. for $B \in T^{(M)}(E)$,

$$\log(B) = \sum_{k=1}^{M} \frac{-1}{k}(1-B)^{\otimes k}. \tag{A.9}$$

Thus, we need to count the multiplications of computing a single $\otimes$ and according to [46], it requires complexity of

$$\sum_{k=1}^{M}(k-1)d^k.$$

Finally, the multiplications with $\frac{-1}{k}$ count

$$\sum_{k=1}^{M} d^k.$$

After summing up with the complexity of signature, we achieve the results in (A.8).

□

Path augmentations. In order to derive different information in the signature features of time series data, we introduce classical transformation methods that return us new series. Let $\mathbf{x} := (x_{t_i})_{i=1}^{n} \in \mathbb{R}^{d\times n}$ be the discrete time series

- **Cumulative sum:** The cumulative sum is a map $\phi : \mathbb{R}^{d\times n} \to \mathbb{R}^{d\times n}$ defined by

$$\phi(\mathbf{x}) = (x_{t_1}, \ldots, \sum_{i=1}^{j} x_{t_i}, \ldots, \sum_{i=1}^{n} x_{t_i}).$$

  The advantage of cumulative sum with signature is to extract the quadratic variation and other higher order statistics of the input path $\mathbf{x}$ effectively [100].

- **Lead-lag transformation:** We define the lead-lag transformation by the

function $\phi : \mathbb{R}^{d\times n} \to \mathbb{R}^{2d\times 2n-1}$ as follows

$$\phi(\mathbf{x}) = ((x_{t_1}, x_{t_1}), (x_{t_2}, x_{t_1}), (x_{t_2}, x_{t_2}), (x_{t_3}, x_{t_2}) \ldots, (x_{t_n}, x_{t_n}))$$

This transformation help capture the quadratic variation of the time series [101].

- **Visibility transformation:** The initial-position-incorporated visibility transformation (I-visibility transformation) [102] is defined as a map $\phi : \mathbb{R}^{d\times n} \to \mathbb{R}^{d+1\times n+2}$. Let $\bar{\mathbf{x}} = \phi(\mathbf{x})$, then we have

$$\bar{\mathbf{x}}_{t_1} = \mathbf{0}, \quad \bar{\mathbf{x}}_{t_2} = (x^1_{t_1}, \ldots, x^d_{t_1}, 0)^T$$

and

$$\bar{\mathbf{x}}_{t_{i+2}} = (x^1_{t_i}, \ldots, x^d_{t_i}, 1)^T, \quad \forall i \in \{0, \ldots, n\}.$$

The transformation adds time translation sensitivity and encodes the effect of absolute position to the signature features.

Along with the time augmentation, all these transformations can be combined together when dealing with time series data to extract more informative features. For example, with the cumulative sum and lead-lag embedding together, the signature features provide information of moments of the data.

Appendix B

Proof of Chapter 2

In this section, we prove some necessary lemmas for the universality theorems. The first lemma states the error between the numerical approximated solution $\hat{Y}$ and the solution to SDE Y at discrete time steps $t_j \in \mathscr{D}_Y$ can be uniformly bounded.

Lemma 2.6. *Under Assumption 1, let* $(\hat{Y}_{t_j}^{\mathscr{D}_Y,M})_{j=0}^{N_2}$ *be defined in* (2.2)*. For any* $\varepsilon > 0$*, there exists* $\delta_1 > 0$ *such that when* $\Delta\mathscr{D} \leq \delta_1$ *and* $M \geq \lfloor \gamma \rfloor$*, then* $\hat{Y}_{t_j}$ *satisfies that*

$$\sup_{t_j \in \mathscr{D}_Y} ||Y_{t_j} - \hat{Y}_{t_j}|| \leq \varepsilon, \tag{B.1}$$

More specifically, there exists $\bar{C}$ *defined in* (2.14) *such that*

$$\sup_{t_j \in \mathscr{D}_Y} ||Y_{t_j} - \hat{Y}_{t_j}|| \leq \bar{C} \left[\max_{u_i \in \mathscr{D}_Y} \omega(u_{i-1}, u_i) \right]^{\frac{\lfloor \gamma \rfloor + 1}{p} - 1}. \tag{B.2}$$

Proof. According to Theorem 2.3, for $t_j \in (u_{k-1}, u_k]$, there exists $C := C(p, \gamma)$ such

that

$$\begin{aligned} ||Y_{t_j} - \hat{Y}_{t_j}|| &\leq C \sum_{i=1}^{k-1} |f|_{\circ\gamma;J}^{\lfloor\gamma\rfloor+1} ||\mathbf{X}||_{p-var;[u_{i-1},u_i]}^{\lfloor\gamma\rfloor+1} \\ &+ C|f|_{\circ\gamma;J}^{\lfloor\gamma\rfloor+1} ||\mathbf{X}||_{p-var;[u_{k-1},t_j]}^{\lfloor\gamma\rfloor+1} \\ &\leq C \sum_{i=1}^{k} |f|_{\circ\gamma;J}^{\lfloor\gamma\rfloor+1} ||\mathbf{X}||_{p-var;[u_{i-1},u_i]}^{\lfloor\gamma\rfloor+1}. \end{aligned} \tag{B.3}$$

And there exist constants $\beta_1, C_2 > 0$ such that for any $(s,t) \in \Delta_T$,

$$||\mathbf{X}||_{p-var;(s,t)} \leq \beta_1 \omega(s,t)^{\frac{1}{p}} \leq C_2. \tag{B.4}$$

Then Equation (B.3) and the super-additivity of the control ω implies that for any $j \in \{0,\ldots,N_2\}$ and $t_j \in (u_{k-1}, u_k]$,

$$||Y_{t_j} - \hat{Y}_{t_j}|| \leq C\beta_1^{\lfloor\gamma\rfloor+1} ||f||_{\circ\gamma;J}^{|\gamma|+1} \sum_{i=1}^{k} \omega(u_{i-1},u_i)^{\frac{\lfloor\gamma\rfloor+1}{p}} \tag{B.5}$$

$$\leq C\beta_1^{\lfloor\gamma\rfloor+1} ||f||_{\circ\gamma;J}^{|\gamma|+1} \omega(0,u_k) \left[\max_i \omega(u_{i-1},u_i)\right]^{\frac{|\gamma|+1}{p}-1} \tag{B.6}$$

$$\leq C\beta_1^{\lfloor\gamma\rfloor+1} ||f||_{\circ\gamma;J}^{|\gamma|+1} \omega(J) \left[\max_i \omega(u_{i-1},u_i)\right]^{\frac{|\gamma|+1}{p}-1} \tag{B.7}$$

$$\leq \bar{C} \left[\max_i \omega(u_{i-1},u_i)\right]^{\frac{\lfloor\gamma\rfloor+1}{p}-1} \tag{B.8}$$

where

$$\bar{C} = C\beta_1^{\lfloor\gamma\rfloor+1-p} ||f||_{\circ\gamma;J}^{|\gamma|+1} C_2^p. \tag{B.9}$$

Thus by the uniform continuity of ω, for any $\varepsilon > 0$, there exists $\delta_2 > 0$, such that the right hand side of (B.8) can be smaller than ε uniformly over all the partitions with mesh size smaller than δ_2. □

For the global approximation theorem 2.5, we need the following lemma. According to the existence theorem of solutions to (2.1) in [47], we have

Lemma B.1. *[47] Under the Assumption 1, there exists constant $C := C(p,\gamma)$ such that for any $s < t$ in $[0,T]$, the following holds*

$$||Y_t - Y_s|| \leq C\phi_p\left(|f|_{\circ\gamma-1}||\boldsymbol{X}||_{p-var;[s,t]}\right), \tag{B.10}$$

where the function $\phi_p(x) = (x \vee x^p)$.

Next, we introduce the auxiliary lemmas that are necessary for the control of the error between the numerical approximated solutions and the output of the (generalized) Logsig-RNN. The following lemma on the universality of shallow neural network was proved by Funahshi [103].

Lemma B.2. *Let $\sigma(x)$ be a non-constant, increasing, and bounded continuous function on $\mathbb{R}$. Let K be any compact subset of $\mathbb{R}^d$, and $f : K \to \mathbb{R}^e$ be a continuous function mapping. Then for an arbitrary $\varepsilon > 0$, there exists an integer $m > 0$, an $e \times m$ real matrix A, an $m \times d$ real matrix B and an m dimensional vector θ such that*

$$\max_{x\in K} |f(x) - A\sigma(Bx+\theta)| < \varepsilon,$$

holds where $\sigma : \mathbb{R}^m \mapsto \mathbb{R}^m$ is a continuous mapping defined by

$$\sigma((u_1,\cdots,u_m)) = (\sigma(u_1),\cdots,\sigma(u_m)).$$

Lemma B.3. *Let K be any compact subset of $\mathbb{R}^d$. Let f and $\tilde{f}$ be two continuously differentiable functions on $\mathbb{R}^{d+e}$. The function $G_{f,k}$ is defined in* (2.19)*. Then it*

follows that

$$||G_{f,k} - G_{\tilde{f},k}||_{\infty,K} < \bar{C}_k ||f - \tilde{f}||_{\infty,K},$$

where $\bar{C}_k$ *is a constant depending on the* Df *and k, i.e.*

$$\bar{C}_k = \begin{cases} \frac{C_{1,k}^k - 1}{C_{1,k} - 1}, & \text{if } C_{1,k} \neq 1; \\ k, & \text{if } C_{1,k} = 1. \end{cases} \tag{B.11}$$

$$C_{1,k} := \sup_{(x_1,\cdots,x_k)\in K} \max_{i=1}^{k} ||D_r f(x_i, r_i)||.$$

Proof. As f and $\tilde{f}$ are continuous functions and K is compact, then the image of $G_{f,k}$ and $G_{\tilde{f},k}$ for any $(x_1,\cdots,x_k) \in K$ are compact. Let $(h_i)_{i=1}^k$ and $(\tilde{h}_i)_{i=1}^k$ denote $G_{f,k}$ and $G_{\tilde{f},k}$ evaluated at $(x_1, x_2, \cdots, x_k)$ respectively. Then we have

$$h_{i+1} = f(x_{i+1}, h_i) \text{ and } \tilde{h}_{i+1} = \tilde{f}(x_{i+1}, \tilde{h}_i).$$

Then it follows that

$$\begin{aligned} & ||h_{i+1} - \tilde{h}_{i+1}|| = ||f(x_{i+1}, h_i) - \tilde{f}(x_{i+1}, \tilde{h}_i)|| \\ \leq\ & ||f(x_{i+1}, h_i) - f(x_{i+1}, \tilde{h}_i)|| + \\ & ||f(x_{i+1}, \tilde{h}_i) - \tilde{f}(x_{i+1}, \tilde{h}_i)|| \\ \leq\ & ||f(x_{i+1}, \tilde{h}_i) - \tilde{f}(x_{i+1}, \tilde{h}_i)|| + \\ & \sup_{x\in K} ||D_h f(x,h)||||h_i - \tilde{h}_i||, \end{aligned}$$

where the supremum of $\sup_{x\in K} ||D_h f(x,h)||$ is taken over x only, which shows the recursive relation of $||h_i - \tilde{h}_i||$.

It is easy to check that if $a_{i+1} \leq C_0 + C_{1,k} a_i$ with $a_0 = 0$, it implies that

$$a_i \leq \begin{cases} \frac{C_{1,k}^i - 1}{C_{1,k} - 1} C_0, \text{ if } C_{1,k} \neq 1 \\ iC_0, \text{ if } C_{1,k} = 1. \end{cases}$$

Therefore using the above inequality when $a_i = ||h_i - \tilde{h}_i||$, $C_0 = \max_{x \in K} ||f - \tilde{f}||$, it follows that

$$||h_i - \tilde{h}_i|| \leq \bar{C}_k ||f - \tilde{f}||_{\infty,K},$$

and so does

$$||G_{f,k} - G_{\tilde{f},k}||_{\infty,K} \leq \bar{C}_k ||f - \tilde{f}||_{\infty,K},$$

where $\bar{C}_k$ is defined by Equation (B.11). □

Appendix C

Implementation Details of PT-Logsig-RNN

In this section, we provide the implementation details of the PT-Logsig-RNN approach on the SHAR datasets.

Gesture Recognition The skeletons are pre-processed by first subtracting the joint center, which is the average position of all joints for each sample. Then for each sample we normalize the sequence of joint positions within $[-1,1]$. For the EL-Logsig-LSTM, the path transform layers are composed of two Conv2D layers followed by a Conv1D layer [57] with kernel size $(1,32)$, $(3,16)$ and 30 respectively, a Time-incorporated layer and an Accumulative Layer followed by the Log-Signature Layer with 4 segments and degree 2 log-signatures. For the GCN-Logsig-LSTM, we apply it twice where each block has one GCN layer and one Logsig-RNN layer. The two GCN layers have 192 and 384 features respectively, and the two Log-Signature Layers have segments equal to 16 and 4 respectively and both use degree 2 log-signatures. Three methods of data augmentation are used in the experiments. The first one is to rotate the coordinates along x,y,z axis by the angles in range of $[-\pi/36,\pi/36]$, $[-\pi/18,\pi/18]$ and $[-\pi/36,\pi/36]$ respectively. The second one is randomly shifting the frame temporally in range of $[-5,5]$. The

last one is to add a Gaussian noise with standard deviation 0.001 to joints coordinates at each frame.

Action Recognition For both the EL-Logsig-LSTM and the GCN-Logsig-LSTM, we consider variable length input, for which we pad each sequence to the maximum length of the batch. For the EL-Logsig-LSTM, the path transform layers are mainly composed with two Conv2D layers followed by one Conv1D layer with kernel size equal to $(1,32)$, $(5,16)$ and 40 respectively. The input of the LSTM layer is the concatenation of the starting frame of each segment of Conv1D output and the output of Log-Signature Layer with 64 segments and degree 2 log-signature. For the GCN-Logsig-LSTM, we also apply it twice where each block has one GCN layer and one Logsig-RNN layer. The two GCN layers have 96 and 480 features respectively, and the two Log-Signature Layers have segments equal to 80 and 40 respectively and both use degree 2 log-signatures.

Bibliography

[1] Crispin W Gardiner et al. *Handbook of stochastic methods*, volume 3. Springer Berlin, 1985.

[2] Fischer Black and Myron Scholes. The pricing of options and corporate liabilities. *Journal of political economy,*, 81(3):637–654, 1973.

[3] Robert C Merton et al. Theory of rational option pricing. *Theory of Valuation,*, pages 229–288, 1973.

[4] John C Cox and Stephen A Ross. The valuation of options for alternative stochastic processes. *Journal of financial economics,*, 3(1-2):145–166, 1976.

[5] Weinan E. A proposal on machine learning via dynamical systems. *Comm. in Math. and Stat.,*, 5(1):1–11, 2017.

[6] Yiping Lu, Aoxiao Zhong, Quanzheng Li, and Bin Dong. Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. *arXiv preprint arXiv:1710.10121*, 2017.

[7] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[8] Patrick Kidger, James Morrill, James Foster, and Terry Lyons. Neural controlled differential equations for irregular time series. In *Advances in Neural*

Information Processing Systems, volume 33, pages 6696–6707. Curran Associates, Inc., 2020.

[9] Ken-ichi Funahashi and Yuichi Nakamura. Approximation of dynamical systems by continuous time recurrent neural networks. *Neural networks,*, 6(6):801–806, 1993.

[10] Terry J Lyons, Michael Caruana, and Thierry Lévy. *Differential equations driven by rough paths*. Springer, 2007.

[11] James Foster, Terry Lyons, and Harald Oberhauser. An optimal polynomial approximation of brownian motion. *arXiv preprint arXiv:1904.06998*, 2019.

[12] Benjamin Graham. Sparse arrays of signatures for online character recognition. *arXiv preprint arXiv:1308.0371*, 2013.

[13] Zecheng Xie, Zenghui Sun, Lianwen Jin, Hao Ni, and Terry Lyons. Learning spatial-semantic context with fully convolutional recurrent network for online handwritten chinese text recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(8):1903–1917, 2018.

[14] Weixin Yang, Terry Lyons, Hao Ni, Cordelia Schmid, and Lianwen Jin. Developing the path signature methodology and its application to landmark-based human action recognition. In *Stochastic Analysis, Filtering, and Stochastic Optimization: A Commemorative Volume to Honor Mark H. A. Davis's Contributions*, pages 431–464, Cham, 2022. Springer International Publishing.

[15] Lajos Gergely Gyurkó, Terry Lyons, Mark Kontkowski, and Jonathan Field. Extracting information from the signature of a financial data stream. *arXiv preprint arXiv:1307.7244*, 2013.

[16] Terry Lyons, Hao Ni, and Harald Oberhauser. A feature set for streams and an application to high-frequency financial tick data. In *ACM International Conference on Big Data Science and Computing*, page 5, 2014.

[17] Chenyang Li, Xin Zhang, and Lianwen Jin. Lpsnet: A novel log path signature feature based hand gesture recognition framework. In *2017 IEEE International Conference on Computer Vision Workshops*, pages 631–639, 2017.

[18] Jeremy Reizenstein and Benjamin Graham. The iisignature library: efficient calculation of iterated-integral signatures and log signatures. *arXiv preprint arXiv:1802.08252*, 2018.

[19] Shujian Liao, Terry Lyons, Weixin Yang, and Hao Ni. Learning stochastic differential equations using rnn with log signature features. *arXiv preprint arXiv:1908.08286*, 2019.

[20] Hao Ni, Lukasz Szpruch, Marc Sabate-Vidales, Baoren Xiao, Magnus Wiese, and Shujian Liao. Sig-wasserstein gans for time series generation. In *2nd ACM International Conference on AI in Finance*, 2021.

[21] Shujian Liao, Terry Lyons, Weixin Yang, Kevin Schlegel, and Hao Ni. Logsig-rnn: a novel network for robust and efficient skeleton-based action recognition. In *British Machine Vision Conference*, 2021.

[22] Sergio Escalera, Jordi Gonzàlez, Xavier Baró, Miguel Reyes, Oscar Lopes, Isabelle Guyon, Vassilis Athitsos, and Hugo Jair Escalante. Multi-modal gesture recognition challenge 2013: dataset and results. In *Proceedings of the 15th ACM on International Conference on Multimodal Interaction*, 2013.

[23] Jun Liu, Amir Shahroudy, Mauricio Perez, Gang Wang, Ling-Yu Duan, and Alex C. Kot. Ntu rgb+d 120: A large-scale benchmark for 3d human ac-

tivity understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[24] Daniel Levin, Terry Lyons, and Hao Ni. Learning from the past, predicting the statistics for the future, learning an evolving system. *arXiv preprint arXiv:1309.0260*, 2013.

[25] Patrick Kidger, Patric Bonnier, Imanol Perez Arribas, Cristopher Salvi, and Terry Lyons. Deep signature transforms. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[26] Ton Dieker. Simulation of fractional brownian motion. *Masters Thesis, Department of Mathematical Sciences, University of Twente*, 2004.

[27] B.M. Hambly and Terry Lyons. Uniqueness for the signature of a path of bounded variation and the reduced path group. *Annals of Mathematics,*, 171(1):109–167, 2010.

[28] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85 – 117, 2015.

[29] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[30] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[31] A. Graves, A. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649, 2013.

[32] Chang Sim Vui, Gan Kim Soon, Chin Kim On, Rayner Alfred, and Patricia Anthony. A review of stock market prediction with artificial neural network (ann). In *2013 IEEE International Conference on Control System, Computing and Engineering*, pages 477–482, 2013.

[33] Guy Lev, Gil Sadeh, Benjamin Klein, and Lior Wolf. Rnn fisher vectors for action recognition and image annotation. In *European Conference on Computer Vision*, pages 833–850, 2016.

[34] Jun Liu, Amir Shahroudy, Dong Xu, and Gang Wang. Spatio-temporal lstm with trust gates for 3d human action recognition. In *European Conference on Computer Vision*, pages 816–833, 2016.

[35] Jun Liu, Guanghui Wang, Ling yu Duan, Kamila Abdiyeva, and Alex ChiChung Kot. Skeleton-based human action recognition with global context-aware attention lstm networks. *IEEE Transactions on Image Processing*, 27:1586–1599, 2018.

[36] Cristóbal Esteban, Stephanie L. Hyland, and Gunnar Rätsch. Real-valued (medical) time series generation with recurrent conditional gans. *arXiv preprint arXiv:1706.02633*, 2017.

[37] Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. Time-series generative adversarial networks. In *Advances in Neural Information Processing Systems*, pages 5509–5519, 2019.

[38] Coryn Bailer-jones, David Mackay, and Philip Withers. A recurrent neural network for modelling dynamical systems. *Network: Computation in Neural Systems*, 9, 08 2002.

[39] Jaya PN Bishwal. *Parameter estimation in stochastic differential equations*. Springer, 2007.

[40] Tom Ryder, Andrew Golightly, A. Stephen McGough, and Dennis Prangle. Black-box variational inference for stochastic differential equations. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 4423–4432. PMLR, 2018.

[41] Anastasia Papavasiliou, Christophe Ladroue, et al. Parameter estimation for rough differential equations. *The Annals of Statistics,*, 39(4):2047–2073, 2011.

[42] James Morrill, Cristopher Salvi, Patrick Kidger, and James Foster. Neural rough differential equations for long time series. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pages 7829–7838. PMLR, 2021.

[43] Hans-Georg Müller, Rituparna Sen, and Ulrich Stadtmüller. Functional data analysis for volatility. *Journal of Econometrics,*, 165(2):233–245, 2011.

[44] Bernard W Silverman et al. Smoothed functional principal components analysis by choice of norm. *The Annals of Statistics,*, 24(1):1–24, 1996.

[45] Reizenstein Jeremy. *Iterated-Integral Signatures in Machine Learning*. PhD thesis, 2019.

[46] Patrick Kidger and Terry Lyons. Signatory: differentiable computations of the signature and logsignature transforms, on both CPU and GPU. In

International Conference on Learning Representations, 2021. `https://github.com/patrick-kidger/signatory`.

[47] P.K. Friz and N.B. Victoir. *Multidimensional Stochastic Processes as Rough Paths: Theory and Applications*. Cambridge Studies in Advanced Mathematics. 2010.

[48] Rich Caruana and Alexandru Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd International Conference on Machine Learning*, page 161–168, 2006.

[49] Liliana Lo Presti and Marco La Cascia. 3d skeleton-based human action classification. *Pattern Recognition*, 53(C):130–147, 2016.

[50] Lei Wang, Du Q Huynh, and Piotr Koniusz. A comparative review of recent kinect-based action recognition algorithms. *IEEE Transactions on Image Processing*, 29:15–28, 2019.

[51] Bin Ren, Mengyuan Liu, Runwei Ding, and Hong Liu. A survey on 3d skeleton-based action recognition using learning method. *arXiv preprint arXiv:2002.05907*, 2020.

[52] Amir Shahroudy, Jun Liu, Tian-Tsong Ng, and Gang Wang. Ntu rgb+d: A large scale dataset for 3d human activity analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2016.

[53] Hongsong Wang and Liang Wang. Modeling temporal dynamics and spatial configurations of actions using two-stream recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3633–3642, 2017.

[54] Carlos Caetano, Jessica Sena, François Brémond, Jefersson A Dos Santos, and William Robson Schwartz. Skelemotion: A new representation of skeleton joint sequences based on motion information for 3d action recognition. In *2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–8. IEEE, 2019.

[55] Guilhem Chéron, Ivan Laptev, and Cordelia Schmid. P-cnn: Pose-based cnn features for action recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 3218–3226, 2015.

[56] Qiuhong Ke, Mohammed Bennamoun, Senjian An, Ferdous Sohel, and Farid Boussaid. Learning clip representations for skeleton-based 3d action recognition. *IEEE Transactions on Image Processing*, 27(6):2842–2855, 2018.

[57] Chao Li, Qiaoyong Zhong, Di Xie, and Shiliang Pu. Co-occurrence feature learning from skeleton data for action recognition and detection with hierarchical aggregation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 786–792, 2018.

[58] Jun Liu, Amir Shahroudy, Gang Wang, Ling-Yu Duan, and Alex C. Kot. Skeleton-based online action prediction using scale selection network. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(6):1453–1467, 2020.

[59] Mengyuan Liu and Junsong Yuan. Recognizing human actions as the evolution of pose estimation maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1159–1168, 2018.

[60] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Proceedings of*

the Thirty-Second AAAI Conference on Artificial Intelligence,, pages 7444–7452. AAAI Press, 2018.

[61] Maosen Li, Siheng Chen, Xu Chen, Ya Zhang, Yanfeng Wang, and Qi Tian. Actional-structural graph convolutional networks for skeleton-based action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3595–3603, 2019.

[62] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. Skeleton-based action recognition with directed graph neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7912–7921, 2019.

[63] Jiayi Fan, Zhengjun Zha, and Xinmei Tian. Action recognition with novel high-level pose features. In *2016 IEEE International Conference on Multimedia & Expo Workshops*, pages 1–6. IEEE, 2016.

[64] Hueihan Jhuang, Juergen Gall, Silvia Zuffi, Cordelia Schmid, and Michael J Black. Towards understanding action recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 3192–3199, 2013.

[65] Xiaodong Yang and YingLi Tian. Effective 3d action recognition using eigenjoints. *Journal of Visual Communication and Image Representation*, 25(1):2–11, 2014.

[66] Georgios Evangelidis, Gurkirt Singh, and Radu Horaud. Skeletal quads: Human action recognition using joint quadruples. In *2014 22nd International Conference on Pattern Recognition*, pages 4513–4518. IEEE, 2014.

[67] Meng Li and Howard Leung. Multiview skeletal interaction recognition using active joint interaction graph. *IEEE Transactions on Multimedia*, 18(11):2293–2302, 2016.

[68] Amir Shahroudy, Tian-Tsong Ng, Qingxiong Yang, and Gang Wang. Multimodal multipart learning for action recognition in depth videos. *IEEE transactions on pattern analysis and machine intelligence*, 38(10):2123–2129, 2015.

[69] Qiuhong Ke, Senjian An, Mohammed Bennamoun, Ferdous Sohel, and Farid Boussaid. Skeletonnet: Mining deep part features for 3-d action recognition. *IEEE signal processing letters*, 24(6):731–735, 2017.

[70] Ziyu Liu, Hongwen Zhang, Zhenghao Chen, Zhiyong Wang, and Wanli Ouyang. Disentangling and unifying graph convolutions for skeleton-based action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[71] Honghui Lin, Jiale Cheng, Yu Li, and Xin Zhang. Temporal-spatial deformable pose network for skeleton-based gesture recognition. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 2324–2328, 2021.

[72] Jun Liu, Amir Shahroudy, Dong Xu, Alex C. Kot, and Gang Wang. Skeleton-based action recognition using spatio-temporal lstm network with trust gates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):3007–3021, Dec 2018.

[73] Chenyang Li, Xin Zhang, Lufan Liao, Lianwen Jin, and Weixin Yang. Skeleton-based gesture recognition using several fully connected layers with path signature features and temporal transformer module. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8585–8593, 2019.

[74] Ke Cheng, Yifan Zhang, Xiangyu He, Weihan Chen, Jian Cheng, and Hanqing Lu. Skeleton-based action recognition with shift graph convolutional

network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2020.

[75] Lufan Liao, Xin Zhang, and Chenyang Li. Multi-path convolutional neural network based on rectangular kernel with path signature features for gesture recognition. In *2019 IEEE Visual Communications and Image Processing (VCIP)*, pages 1–4. IEEE, 2019.

[76] Yi-Fan Song, Zhang Zhang, Caifeng Shan, and Liang Wang. Richly activated graph convolutional network for robust skeleton-based action recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(5):1915–1925, 2021.

[77] Yi-Fan Song, Zhang Zhang, Caifeng Shan, and Liang Wang. Stronger, faster and more explainable: A graph convolutional baseline for skeleton-based action recognition. In *Proceedings of the 28th ACM International Conference on Multimedia*, page 1625–1633, 2020.

[78] Wei Mao, Miaomiao Liu, Mathieu Salzmann, and Hongdong Li. Learning trajectory dependencies for human motion prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9489–9497, 2019.

[79] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2021.

[80] Shota Haradal, Hideaki Hayashi, and Seiichi Uchida. Biosignal data augmentation based on generative adversarial networks. In *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 368–371, 2018.

[81] O. Mogren. C-rnn-gan: A continuous recurrent neural network with adversarial training. In *Constructive Machine Learning Workshop (CML) at NIPS*, 2016.

[82] Tianlin Xu, Li Kevin Wenliang, Michael Munn, and Beatrice Acciaio. Cot-gan: Generating sequential data via causal optimal transport. In *Advances in Neural Information Processing Systems*, volume 33, pages 8798–8809. Curran Associates, Inc., 2020.

[83] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223. PMLR, 2017.

[84] Eric V Mazumdar, Michael I Jordan, and S Shankar Sastry. On finding local nash equilibria (and only local nash equilibria) in zero-sum games. *arXiv preprint arXiv:1901.00838*, 2019.

[85] Tianyi Lin, Chi Jin, and Michael Jordan. On gradient descent ascent for nonconvex-concave minimax problems. In *International Conference on Machine Learning*, pages 6083–6093. PMLR, 2020.

[86] Constantinos Daskalakis and Ioannis Panageas. The limit points of (optimistic) gradient descent in min-max optimization. *arXiv preprint arXiv:1807.03907*, 2018.

[87] Constantinos Daskalakis, Andrew Ilyas, Vasilis Syrgkanis, and Haoyang Zeng. Training gans with optimism. *arXiv preprint arXiv:1711.00141*, 2017.

[88] Panayotis Mertikopoulos, Christos Papadimitriou, and Georgios Piliouras. Cycles in adversarial regularized learning. In *Proceedings of the Twenty-*

Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, pages 2703–2717. SIAM, 2018.

[89] Farzan Farnia and Asuman Ozdaglar. Gans may have no nash equilibria. *arXiv preprint arXiv:2002.09124*, 2020.

[90] Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. Are gans created equal? a large-scale study. *arXiv preprint arXiv:1711.10337*, 2017.

[91] Ilya Chevyrev and Terry Lyons. Characteristic functions of measures on geometric rough paths. *The Annals of Probability*, 44(6):4049 – 4082, 2016.

[92] Terry Lyons and Hao Ni. Expected signature of Brownian motion up to the first exit time from a bounded domain. *The Annals of Probability*, 43(5):2729 – 2762, 2015.

[93] Ilya Chevyrev and Harald Oberhauser. Signature moments to characterize laws of stochastic processes. *arXiv preprint arXiv:1810.10971*, 2018.

[94] Magnus Wiese, Lianjun Bai, Ben Wood, and Hans Buehler. Deep hedging: Learning to simulate equity option markets. *SSRN Electronic Journal*, 2019.

[95] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2256–2265, Lille, France, 07–09 Jul 2015. PMLR.

[96] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and

H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020.

[97] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

[98] Reizenstein Jeremy and Graham Benjamin. The iisignature library: efficient calculation of iterated-integral signatures and log signatures. *arXiv preprint arXiv:1802.08252.*, 2018.

[99] Christophe Reutenauer. Free lie algebras. In *Handbook of algebra*, volume 3, pages 887–903. Elsevier, 2003.

[100] Hao Ni. A multi-dimensional stream and its signature representation. *arXiv preprint arXiv:1509.03346*, 2015.

[101] Ilya Chevyrev and Andrey Kormilitzin. A primer on the signature method in machine learning. *arXiv preprint arXiv:1603.03788*, 2016.

[102] Yue Wu, Hao Ni, Terence J. Lyons, and Robin L. Hudson. Signature features with the visibility transformation. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 4665–4672, 2021.

[103] Ken-Ichi Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural Networks*, 2(3):183–192, 1989.