Scalable Machine Learning Algorithms using Path Signatures

Csaba Tóth

Corpus Christi College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy in Mathematics

Michaelmas 2024

Acknowledgements

First and foremost, I would like to thank my supervisor Prof. Harald Oberhauser for his ongoing support and guidance throughout my DPhil journey. I am deeply grateful to Harald for the thought-provoking conversations we had and for always supporting me in exploring my ideas. I would like to extend my gratitude to Prof. Terry Lyons and the DataSig team, who welcomed me as one of their own, and provided me with opportunities to present my work at various workshops and conferences. I would also like to thank the Mathematical Institute of the University of Oxford for generously providing me with a scholarship that allowed me to undertake this journey, and for providing the computational resources required for my applied projects.

I am grateful to my collaborators and colleagues for the many discussions we had: Patric Bonnier, Darrick Lee, Zoltán Szabó, Masaki Adachi, Patrick Kidger, Cristopher Salvi, Maud Lemercier, Christina Zou, Alexander Schell; thank you for your help and collegiality. I am also grateful to all my friends whether located in the UK or Hungary for supporting me in all ways and standing by me potentially despite the physical distance between us. Finally, I would like to thank the infinite support of my family: my sister Réka and my mother Éva; without your love and support I would not be where or who I am today. I dedicate this thesis to you.

Declarations

This thesis is submitted to the University of Oxford in support of my application for the degree of Doctor of Philosophy. It has been composed by myself under the supervision of Prof. Harald Oberhauser. I confirm that this thesis has not been submitted for another university degree. Parts of this thesis have appeared as published papers some of which were written with collaborators:

1. The work in Chapter 2 adapts the paper [266] written by me under the supervision of Harald Oberhauser published at the *International Conference on Machine Learning* in 2020.

2. The work in Chapter 3 adapts the paper [263], that was written in collaboration with Patric Bonnier under the supervision of Harald Oberhauser published at the *International Conference on Learning Representations* in 2021. In particular, I wrote the paper besides the theoretical results appearing originally in Appendices A, B and C, which were written by Patric, and due to space limitations these were omitted here and deferred to the article.

3. Chapter 4 adapts the paper [265] written in collaboration with Darrick Lee with frequent discussions with Celia Hacker and Harald Oberhauser published at the *Advances in Neural Information Processing Systems* in 2022. My contribution was the idea of using expected signatures to describe random walks on graphs, deriving the governing equations both in the tensor-valued and low-rank case, implementing and running experiments, while Darrick formalized the theory and the connection to hypo-elliptic diffusions. The main text was written with equal contribution, while the parts of the appendix written by Darrick consisting of a background and certain theoretical proofs were deferred to the article.

4. Chapter 5 adapts the paper [267] published in *SIAM Mathematics of Data Science* in 2025 written by me with frequent discussions with Zoltán Szabó and Harald Oberhauser.

5. Chapter 6 adapts [262] which was published at *Artificial Intelligence and Statistics* in 2025 written by me with frequent discussions with Masaki Adachi and Harald Oberhauser.

Abstract

In this thesis, we consider the integration of path signatures–a mathematical object rooted in rough path theory–with scalable machine learning algorithms to address challenges in sequential and structured data modelling. The key topics considered include:

- **Path Signatures**: Introduced as a hierarchical and theoretically robust feature representation for sequential data, path signatures faithfully capture dynamics while offering properties like invariance to reparameterization and tree-like equivalence. Challenges like computational overheads are addressed through novel algorithms throughout the thesis.
- **Gaussian Processes**: We demonstrate embedding signature kernels into Gaussian process models, offering an expressive probabilistic modelling approach for sequential data while tackling computational barriers via sparse variational inference techniques. This approach enhances performance on probabilistic time series classification tasks.
- **`Seq2Tens` Framework**: Combines signature features with deep learning, using iterations of low-rank layers to mitigate computational costs while retaining expressiveness. Applications include time series classification, mortality prediction, generative modelling.
- **Graph Representation**: Extends path signatures to graph data and connects them to hypo-elliptic diffusions, combined with low-rank techniques offering scalable architectures for capturing global and local graph structures, and outperforming conventional graph neural networks on tasks requiring long-range reasoning.
- **Random Fourier Signature Features**: Introduces scalable random feature-based approximations for signature kernels with supporting theoretical results, overcoming computational limitations for large datasets while retaining state-of-the-art performance.
- **Recurrent Sparse Spectrum Signature Gaussian Processes**: Combines Random Fourier Signature Features with Gaussian Processes and a forgetting mechanism for adaptive context focus in time series forecasting, bridging short- and long-term dependencies.

The topics are divided into separate, self-contained chapters that can be read independently.

Contents

Introduction **1**

Unifying theme 1

Outline 2

1 Path Signatures **7**

1.1 Tensors, tensor algebras, and their dual 7

1.1.1 Tensor products 7

1.1.2 Tensor algebras 9

1.1.3 Duals of tensors 10

1.1.4 Motivating example: polynomial features 12

1.2 Path signatures 13

1.2.1 Definition 13

1.2.2 Properties 15

1.2.3 Examples 17

1.2.4 Signature kernels 17

1.2.5 Rough paths 19

1.2.6 Discrete-time setting 20

1.3 Algorithms 23

1.3.1 Notation for algorithms 23

1.3.2 Algorithms for path signatures 24

2 Gaussian Processes with Signature Covariances **26**

2.1 Introduction 26

2.2 Background and Notation 27

2.2.1 The feature space view 27

2.2.2 Polynomial features 28

2.3 From signature features to covariances 29

2.3.1 Automatic parameterization invariance determination 29

2.3.2 Signature covariances . . . 30
2.3.3 Regularity of the resulting Gaussian process . . . 31
2.4 Sparse variational inducing tensors . . . 32
2.4.1 Variational inference . . . 32
2.4.2 Inter-domain low-rank tensors . . . 33
2.4.3 Algorithms. . . . 34
2.5 Experiments . . . 36
2.6 Conclusion . . . 41
2.A Continuity of GPs with Signature Covariances . . . 42
2.B Benchmark results . . . 45

3 Seq2Tens 47
3.1 Introduction . . . 47
3.2 Capturing order by non-commutative multiplication . . . 48
3.2.1 Seq2Tens in a nutshell . . . 48
3.2.2 Construction . . . 49
3.3 Approximation by low-rank linear functionals . . . 51
3.4 Building neural networks with LS2T layers . . . 54
3.5 Experiments . . . 55
3.5.1 Multivariate time series classification . . . 55
3.5.2 Mortality prediction . . . 58
3.5.3 Generating sequential data . . . 59
3.6 Related work and Summary . . . 60
3.A Computational details . . . 61
3.A.1 Recursive computations . . . 61
3.A.2 Algorithms . . . 63
3.A.3 Complexity analysis . . . 63
3.B Further results . . . 65

4 Capturing Graphs by Hypo-Elliptic Diffusions 66
4.1 Introduction . . . 66
4.2 Sequence Features by Non-Commutative Multiplication. . . . 68
4.3 Hypo-Elliptic Diffusions . . . 71
4.4 Efficient Algorithms for Deep Learning . . . 75
4.5 Experiments . . . 77
4.6 Conclusion . . . 80
4.A Further Details on Hypo-elliptic Graph Diffusion . . . 80

4.B Details on the low-rank Algorithm. 85
4.C Variations and Hyperparameters of Hypo-Elliptic Diffusion 87
4.D Experiments 90
4.D.1 Model details. 90
4.E Details on experiments. 91
4.E.1 Ablation Studies 91
4.E.2 Further Experiments 92
4.E.3 Computation Time 94
4.E.4 Parameter Count 95
4.E.5 Summary of Dataset Parameters 95

5 Random Fourier Signature Features **96**
5.1 Introduction 96
5.2 Background 100
5.2.1 Notation 100
5.2.2 Signature Features for Sequential Data 100
5.2.3 Random Fourier Features 103
5.3 Methodology 104
5.3.1 Random Fourier Signature Features 105
5.3.2 Dimensionality Reduction: Diagonal Projection 107
5.3.3 Dimensionality Reduction: Tensor Random Projection 110
5.3.4 Numerical evaluation 113
5.4 Experiments 114
5.4.1 Time series classification 114
5.4.2 Methods 114
5.4.3 Hyperparameter selection 115
5.4.4 Datasets 116
5.4.5 Results 117
5.5 Conclusion 119
5.A Concentration of measure 120
5.A.1 Classic inequalities 120
5.A.2 Subexponential concentration 120
5.A.3 α-exponential concentration 122
5.A.4 Hypercontractivity 126
5.B Random Fourier Features 127
5.C Bounds on Signature Kernels 130

5.C.1 Distance bounds in the RKHS 130
5.C.2 Proofs of main concentration results 137
5.D Algorithms 150

6 Recurrent Sparse Spectrum Signature Gaussian processes 151
6.1 Introduction 151
6.1.1 Related work 152
6.2 Background 152
6.2.1 Signature Features and Kernels 152
6.2.2 Random Fourier Features 154
6.3 Recurrent Sparse Spectrum Signature Gaussian processes 155
6.3.1 Revisiting Random Fourier Signature Features 155
6.3.2 A forgetting mechanism for signatures 157
6.3.3 Recurrent Sparse Spectrum Signature Gaussian processes 158
6.4 Experiments 161
6.4.1 Synthetic Dataset 161
6.4.2 Real-World Datasets 161
6.5 Conclusion and Limitations 164
6.A Model details 164
6.A.1 Fractional Differencing 164
6.A.2 Algorithms 165
6.A.3 Treatment of Random Parameters 166
6.A.4 Fixing Pathologies in the ELBO 168
6.A.5 Scalability 169

Conclusion 170
Summary 170
Outlook 172

Bibliography 175

Introduction

Unifying theme

The unifying theme of this thesis centers on the integration of path signatures, a mathematical framework rooted in rough path theory, with scalable machine learning algorithms to address challenges in sequential and structured data modelling. Sequential data, such as time series, DNA sequences, or video streams, represent a cornerstone of modern data analysis due to its prevalence across various fields, including finance, healthcare, and natural language processing. The core challenge lies in extracting meaningful representations from such data while preserving its intrinsic sequential structure and ensuring computational efficiency.

A prominent approach for exploiting sequential structure is through handcrafted feature engineering techniques, which find representations of the data so that well-established models become applicable. The dataset is then typically represented in a finite-dimensional Euclidean space. This approach is problematic in the sense that the feature extraction method often has to be tailored to the problem at hand, and some loss of information is inevitably incurred. In recent years, the signature method originating from rough path theory has become a popular method for extracting features of streamed data, that describe sequences and paths using a hierarchy of tensors with increasing degrees. Path signatures provide a robust theoretical foundation for representing sequential data as a hierarchy of tensor features, faithfully capturing the dynamics of the data. On the one hand, this has been remarkably successful due to the powerful mathematical machinery that underlies signatures allowing to provide strong theoretical guarantees. Their mathematical elegance offers unique benefits, such as invariance to reparameterization and universality for approximating continuous functions. On the other hand, this representation invokes a considerable computational burden due to the combinatorially increasing dimensionality of tensors of increasing degrees, as such their practical application in machine learning has been limited by significant computational challenges, e.g. the polynomial growth in feature space dimension leading to bottlenecks in handling high-dimensional data.

This thesis takes a multifaceted approach to address these challenges by developing novel algorithms that retain the theoretical strengths of path signatures while significantly improving their computational scalability. Central to this work is the integration of path signatures with popular machine learning methodologies, including Gaussian processes, deep learning approaches, and kernel methods. Additionally, the thesis extends the application of path signatures beyond sequential data to the domain of graphs, showcasing their versatility and adaptability. Overall, this approach proved to be a fruitful one often leading to outperforming state-of-the-art methods in their genre on tasks such as time series classification, time series forecasting, mortality prediction in healthcare, generative modelling, and graph-based learning. This unifying theme underscores the thesis's contribution to bridging the gap between the theoretical elegance of path signatures and their practical deployment in real-world machine learning applications.

Outline

The thesis is organized into six chapters that serve as standalone blocks readable independently, each complete with their own set of introductions, conclusions, and appendices.

Chapter 1. This introductory chapter provides an accessible introduction to path signatures, a mathematical object rooted in rough path theory, that serves as a powerful feature representation for sequential data. This chapter lays the foundation for understanding the algebraic, analytic, and computational properties of path signatures that make them suitable for machine learning applications. The chapter begins by defining tensor products and constructing tensor algebras, which form the hierarchical feature space where path signatures reside. Tensor algebras, with their associative and noncommutative structure, allow for the representation of data as sequences of tensors, while the dual space and operations like the shuffle product further enhance their algebraic utility. Path signatures are then presented as sequences of iterated integrals capturing the geometry of paths. For bounded variation paths, these signatures encode the interactions of path coordinates, with practical computation simplified using tools like Chen's identity. Although the signature admits extensions to the unbounded variation setting, our applications lie in the discrete domain, and we defer to [182] for a review of rough path theory, one of the main themes of which is exactly that how highly oscillatory driving signals affect systems of differential equations, and how the signature can be extended to this setting.

The chapter emphasizes the key properties of path signatures: their universality as an algebra of functions that approximate continuous mappings, reparameterization invariance for

robustness to changes in time sampling, and the ability to algebraically manipulate path concatenation and reversal. Motivating examples illustrate how path signatures generalize polynomial feature maps for sequential data, providing a versatile and robust framework for sequence modelling. Challenges like the combinatorial growth of tensor dimensions are acknowledged, which is addressed via various scalable solutions later in the thesis, such as low-rank approximations and random projections. By establishing the theoretical and practical foundations of path signatures, Chapter 1 sets the stage for their integration into machine learning models.

Chapter 2. Gaussian Processes (GPs) are a cornerstone of Bayesian machine learning, providing a flexible, non-parametric framework for modelling uncertainty in data. However, their application to sequential data, such as time series or trajectories, is often limited by the difficulty of constructing covariance functions that adequately capture temporal dependencies and patterns. Path signatures offer a powerful solution to this challenge. They provide a hierarchical and structured encoding of sequence information through iterated integrals, making them an ideal tool for defining expressive covariance functions in GPs. This work introduces a novel approach to embedding signature kernels within the GP framework. Thus, we are concerned with learning from sequential data in a Bayesian manner by constructing Gaussian processes with the inner product of signatures as covariance functions, continuing on the premise of [146], where a kernel trick was introduced for the inner product of signature features.

The chapter demonstrates the potential of integrating path signature features into GPs, offering a mathematically grounded and practical approach to Bayesian modelling of sequential data. By constructing covariance functions based on the inner product of signatures, the method captures the temporal and structural dependencies inherent in sequential data. The introduction of sparse variational inference further addresses the computational challenges, ensuring scalability even for large datasets of time series data. Empirical evaluations highlight the effectiveness of the approach, showcasing superior performance on time series classification tasks compared to conventional sequence kernels and comparable performance to state-of-the-art frequentist methods. The results underscore the versatility of signature-based GPs, providing a robust framework for combining theoretical rigor with practical applicability in Bayesian sequential data modelling. This integration not only enriches the representational capacity of GPs, but also opens up new venues for applying Bayesian methods to sequential data problems.

Chapter 3. Deep learning and Gaussian processes have emerged as powerful tools for modelling sequential data, yet both face limitations when processing complex, high-volume or high-dimensional streams. Signature features provide an effective representation of sequences but

can be computationally expensive when incorporated into parametric models like neural networks as in [140]. We introduce Seq2Tens, a novel framework that bridges this gap by leveraging the expressive power of signature-based features while introducing low-rank linear functionals to mitigate computational challenges. This approach allows the representation of sequential data through layers of low-rank transformations, offering a computationally efficient mechanism to extract global sequence information. By stacking multiple such layers, Seq2Tens achieves deep learning-style expressiveness while preserving interpretability and scalability. This chapter presents both the theoretical foundation of Seq2Tens and its practical utility, extending the versatility of signature-based features into new domains of deep learning.

The chapter establishes Seq2Tens as an advancement in the scalable deployment of signature-based features for sequential data modelling. The framework introduces low-rank constraints on linear functionals of signatures, enabling efficient computations directly in the dual space. This idea is further enhanced by the concept of deep layers, which stack multiple low-rank transformations to recover lost expressiveness due to the low-rank constraint in the style of deep learning. Experimental results validate the effectiveness of Seq2Tens across a range of tasks, including time series classification, mortality prediction, and generative modelling, where it consistently outperforms state-of-the-art neural network baselines. The experiments demonstrate how Seq2Tens not only improves computational efficiency but also preserves the expressiveness of signature features, making it a practical and robust solution for modern sequential data challenges. This work paves the way for broader adoption of signature-based methods in deep learning while addressing scalability concerns.

Chapter 4. Graphs provide a natural representation for structured data, appearing in diverse fields such as social networks, molecular chemistry, and computer vision. A major challenge in graph learning is to design representations that capture not only local neighborhood structures but also long-range dependencies and global characteristics. Building upon the foundational ideas of path signatures and their algebraic properties, The chapter extends their application to graph domains by leveraging hypo-elliptic diffusion processes. This novel approach introduces the hypo-elliptic graph Laplacian, a tensor-valued operator that encodes graph substructures through the evolution of random walks and diffusion dynamics. By integrating this operator into neural network architectures, the chapter proposes a scalable framework for capturing complex interactions in graph-structured data. This innovation bridges the gap between local aggregation and long-range structural encoding in graph neural networks, offering a theoretically grounded and computationally efficient solution for graph learning tasks.

This work can be treated as a follow-up to Chapter 3, where we generalized the signature method using algebraic techniques, and devised scalable neural network architectures for

streamed data. We present a significant extension of signature-based methods from sequential data to graph domains, addressing a critical gap in the ability to model long-range interactions and hierarchical structures in graphs by classical graph neural networks. The introduction of the hypo-elliptic graph Laplacian provides a mathematically robust mechanism to encode diffusion dynamics and random walk distributions, enabling the representation of rich graph features. The chapter further adapts low-rank algorithms from the previous chapter to design scalable neural architectures for graph learning, ensuring computational efficiency without sacrificing expressiveness. Empirical evaluations demonstrate state-of-the-art performance on challenging graph tasks compared to classic graph neural networks, such as node classification and graph-level predictions, where traditional methods struggle to capture long-range dependencies. By combining algebraic insights with practical implementation, this work lays the groundwork for future exploration of graph-specific adaptations of signature-based methodologies.

Chapter 5. This chapter follows the premise of increased interest in the signature kernel. Signature kernels have emerged as a powerful tool for modelling sequential data, offering theoretical guarantees such as universality and invariance to reparameterization. Further, the benefits of kernelization allow us to compose the signature map with a static kernel enhancing expressiveness and allowing us to circumvent the curse of dimensionality incurred by high-degree tensors. However, their practical application faces significant computational barriers, particularly the quadratic complexity in sequence length and sample size, which limits scalability. We address this limitation by introducing Random Fourier Signature Features, a novel approximation method that combines the algebraic richness of path signatures with the efficiency of random feature maps. By constructing low-dimensional embeddings that approximate the signature kernel, this approach dramatically reduces the computational overhead while preserving key properties of the kernel with high probability. By refining this method with dimensionality reduction techniques, we end up with even more scalable variants capable of handling millions of sequences. This innovation extends the applicability of signature kernels to large-scale machine learning tasks, offering a practical solution for computationally intensive domains.

Thus, we make a contribution to signature kernels by overcoming their scalability challenges, introducing a random feature-based approximation that is both efficient and theoretically robust. The Random Fourier Signature Features approach leverages probabilistic concentration arguments to ensure accurate kernel approximations with high probability. Experimental evaluations demonstrate that this approach achieves performance on par with full-rank signature kernels on moderate datasets while significantly outperforming other random feature methods on large-scale datasets. This chapter not only addresses the computational bottlenecks of signature kernels but also establishes Random Fourier Signature Features as a scalable and versatile

tool for modern machine learning, enabling the practical application of signature methods in domains requiring high-throughput data processing.

While working on this project, a general `Scikit-Learn` compatible Python package has also been produced for all kinds of signature kernel computations on `GPU` called `KSig` available at `https://github.com/tgcsaba/ksig`, and a preprint that serves as a user guide and documentation to it [264]. This paper is not included as part of this thesis due to space limitations.

Chapter 6. Time series forecasting is a crucial task in fields ranging from finance and energy to healthcare and logistics, requiring models that can capture both short-term dynamics and long-term dependencies. Traditional Gaussian processes (`GPs`) provide a powerful probabilistic framework but struggle with scalability and context adaptability for long time series data. We introduce Recurrent Sparse Spectrum Signature Gaussian Processes (`RS`^{3}`GP`), a novel approach that combines the efficiency of low-rank signature features with variational inference in Gaussian processes. `RS`^{3}`GP` incorporates a principled forgetting mechanism using a novel decay parameterization, adjusting the context length to focus on more recent time steps. This innovation enables `RS`^{3}`GP` to overcome the challenges of traditional `GPs` and signature-based models, offering a scalable and interpretable solution for time series forecasting across diverse problems.

We present `RS`^{3}`GP` as a novel approach that bridges the gap between theoretical rigour and practical applicability in time series forecasting. By leveraging the expressive power of Random Fourier Signature Features and introducing a decay-based forgetting mechanism, `RS`^{3}`GP` dynamically adjusts its focus on relevant data segments, making it particularly effective in scenarios with both short-term and long-term dependencies. The model is computationally efficient, processing long sequences in real time through efficient parallelized computations while maintaining a scalable memory footprint. Experimental evaluations highlight its superior performance over traditional `GPs` and competitive results against deep learning models on synthetic and real-world datasets. This contribution establishes `RS`^{3}`GP` as a competitive approach with the state-of-the-art for scalable, probabilistic time series forecasting. Overall, the proposed methodology opens new directions in advancing scalable and interpretable models in time series forecasting.

Chapter 1

Path Signatures

1.1 Tensors, tensor algebras, and their dual

First, we construct the space that path signatures live in called the tensor algebra.

1.1.1 Tensor products

Here we provide a brief overview of tensor products of vector spaces, which we will use to construct our feature space called the tensor algebra .

Preliminary example. We start with some preliminary examples about tensors.

Example 1.1 (Tensors on $\mathbb{R}^d$)**.** *If $\mathbf{x} = (x_1, \dots, x_d) \in \mathbb{R}^d$ and $\mathbf{y} = (y_1, \dots, y_e) \in \mathbb{R}^e$ are two vectors, then their tensor product $\mathbf{x} \otimes \mathbf{y}$ is defined as the $(d \times e)$-matrix, or degree-2 tensor, with entries $(\mathbf{x} \otimes \mathbf{y})_{i,j} = x_i y_j$. This is also commonly called the outer product of the two vectors. The space $\mathbb{R}^d \otimes \mathbb{R}^e$ is defined as the linear span of all degree-2 tensors $\mathbf{x} \otimes \mathbf{y}$ for $\mathbf{x} \in \mathbb{R}^d, \mathbf{y} \in \mathbb{R}^e$. If $\mathbf{z} \in \mathbb{R}^f$ is another vector, then one may form a degree-3 tensor $\mathbf{x} \otimes \mathbf{y} \otimes \mathbf{z}$ with shape $(d \times e \times f)$ defined to have entries $(\mathbf{x} \otimes \mathbf{y} \otimes \mathbf{z})_{i,j,k} = x_i y_j z_k$. The space $\mathbb{R}^d \otimes \mathbb{R}^e \otimes \mathbb{R}^f$ is analogously defined as the linear span of all degree-3 tensors $\mathbf{x} \otimes \mathbf{y} \otimes \mathbf{z}$ for $\mathbf{x} \in \mathbb{R}^d, \mathbf{y} \in \mathbb{R}^e, \mathbf{z} \in \mathbb{R}^f$, and so forth for higher orders.*

Abstract tensor product. The abstract definition of the tensor product, which holds generally for infinite-dimensional vector spaces is given below.

Definition 1.2 (Abstract tensor product)**.** *If U and V are (not necessarily finite-dimensional) vector spaces, their tensor product $U \otimes V$ is a vector space together with a bilinear map $\otimes : U \times V \to U \otimes W$, such that any bilinear map $B : U \times V \to W$ factors through $U \otimes V$, i.e. there exists a unique*

linear map $\tilde{B} : U \otimes V \to W$, which satisfies $B = \tilde{B} \circ \otimes$. Let $\mathcal{B}_U$ and $\mathcal{B}_V$ respectively be bases of U and V. Then, $U \times V$ is unique up to isomorphism, and its basis is formed by the tensor product of the bases, i.e. $\mathcal{B}_{U\otimes V} = \{\mathbf{u} \otimes \mathbf{v} \in U \otimes V \,:\, \mathbf{u} \in \mathcal{B}_U, \mathbf{v} \in \mathcal{B}_V\}$.

Example 1.3. *Similarly to the example about $\mathbb{R}^d$ above, a coordinate-wise construction of the tensor product is given by identifying $U \otimes V$ with the set of functions from $\mathcal{B}_U \times \mathcal{B}_V$ to $\mathbb{R}$ that are nonzero only on finitely many elements, which is a vector space by pointwise operations on functions. Specifically, for $\mathbf{u} \in \mathcal{B}_U$ and $\mathbf{v} \in \mathcal{B}_V$, we may identify $\mathbf{u}\otimes\mathbf{v}$ with the function that takes 1 on $(\mathbf{u}, \mathbf{v})$ and 0 on every other element of $\mathcal{B}_U \times \mathcal{B}_V$. Then, we can extend this to $U \otimes V$ by asserting the bilinearity of $\otimes$, so that if we denote $\mathbf{x} = \sum_{\mathbf{u}\in\mathcal{B}_U} \alpha_{\mathbf{u}}\mathbf{u}$ and $\mathbf{y} = \sum_{\mathbf{v}\in\mathcal{B}_V} \beta_{\mathbf{v}}\mathbf{v}$, we have that*

$$\mathbf{x} \otimes \mathbf{y} = \left(\sum_{\mathbf{u}\in\mathcal{B}_U} \alpha_{\mathbf{u}}\mathbf{u}\right) \otimes \left(\sum_{\mathbf{v}\in\mathcal{B}_V} \beta_{\mathbf{v}}\mathbf{v}\right) = \sum_{\mathbf{u}\in\mathcal{B}_U}\sum_{\mathbf{v}\in\mathcal{B}_V} \alpha_{\mathbf{u}}\beta_{\mathbf{v}}\mathbf{u} \otimes \mathbf{v},$$

where finitely many coefficients are nonzero in $\{\alpha_{\mathbf{u}} \in \mathbb{R} \,:\, \mathbf{u} \in \mathcal{B}_U\}$ and $\{\beta_{\mathbf{v}} \in \mathbb{R} \,:\, \mathbf{v} \in \mathcal{B}_V\}$.

We refer the reader to [223, 299, 155] for further details about abstract tensors.

Tensor products of Hilbert spaces. It is often that one comes across the definition of the tensor product formulated in the category of vector spaces as an abstract vector space such that any bilinear map factors through it as discussed in the previous paragraph. This can then be endowed with an inner product and completed to get a tensor product in the category of Hilbert spaces. However, there also exist explicit models of tensor products of Hilbert spaces. Hence, we introduce a concrete model of the Hilbert space tensor product originally proposed by [194].

Definition 1.4. *Let $\mathcal{H}_1, \ldots, \mathcal{H}_m$ be Hilbert spaces. To each element $(h_1, \ldots, h_m) \in \mathcal{H}_1 \times \cdots \mathcal{H}_m$, associate the multi-linear operator $h_1 \otimes \cdots \otimes h_m : \mathcal{H}_1 \times \cdots \times \mathcal{H}_m \to \mathbb{R}$ defined for each $(f_1, \ldots, f_m) \in \mathcal{H}_1 \times \cdots \times \mathcal{H}_m$ by*

$$(h_1 \otimes \cdots \otimes h_m)(f_1, \ldots, f_m) = \langle h_1, f_1\rangle_{\mathcal{H}_1} \cdots \langle h_m, f_m\rangle_{\mathcal{H}_m}.$$

Take the linear span of all such multi-linear operators to build the space

$$\mathcal{H}_1 \otimes' \cdots \otimes' \mathcal{H}_m = \operatorname{span}\{h_1 \otimes \cdots \otimes h_m \,:\, h_1 \in \mathcal{H}_1, \ldots, h_m \in \mathcal{H}_m\},$$

and endow $\mathcal{H}_1 \otimes' \cdots \otimes' \mathcal{H}_m$ with an inner product via

$$\langle h_1 \otimes \cdots h_m, f_1 \otimes \cdots \otimes f_m\rangle_{\mathcal{H}_1\otimes'\cdots\otimes'\mathcal{H}_m} = \langle h_1, f_1\rangle_{\mathcal{H}_1} \cdots \langle h_m, f_m\rangle_{\mathcal{H}_m} \tag{1.1}$$

for all $h_1, f_1 \in \mathcal{H}_1, \ldots, h_m, f_m \in \mathcal{H}_m$*, and extend by linearity to* $\mathcal{H}_1 \otimes' \cdots \otimes' \mathcal{H}_m$*. Taking the topological completion of this space under this inner product gives a Hilbert space denoted by* $\mathcal{H}_1 \otimes \cdots \otimes \mathcal{H}_m$ *called the tensor product of the Hilbert spaces* $\mathcal{H}_1, \ldots, \mathcal{H}_m$*.*

1.1.2 Tensor algebras

Now that we have discussed how to take tensor products of vector (Hilbert) spaces, we show how to embed a linear space V into a much bigger linear space $\mathsf{T}((V))$, which is also an associative algebra[1] using a so-called free construction. Since the tensor product is associative, we can unambiguously take tensor powers of the vector space V, and we denote $V^{\otimes m} = V \otimes \cdots \otimes V$. Note that if $\mathcal{B}_V$ is a basis of V, then the basis of $V^{\otimes m}$ is given by all elements of the form

$$\mathbf{v}_1 \otimes \cdots \otimes \mathbf{v}_m \quad \text{for} \quad \mathbf{v}_1, \ldots, \mathbf{v}_m \in \mathcal{B}_V.$$

Definition 1.5 (Extended tensor algebra)**.** *We define the extended tensor algebra as formal series of tensors over a vector space* V *indexed by their degree* $m \in \mathbb{N}$*,*

$$\mathsf{T}((V)) = \prod_{m \geq 0} V^{\otimes m} = \left\{ (t_0, \mathbf{t}_1, \mathbf{t}_2, \ldots) : \mathbf{t}_m \in V^{\otimes m} \text{ for } m \in \mathbb{N} \right\},$$

where by convention $V^{\otimes 0} = \mathbb{R}$ *is a scalar.* $\mathsf{T}((V))$ *is again a vector space with vector addition and scalar multiplication defined coordinate-wise as*

$$\mathbf{s} + \mathbf{t} = (\mathbf{s}_m + \mathbf{t}_m)_{m \geq 0}, \quad \lambda \mathbf{s} = (\lambda \mathbf{s}_m)_{m \geq 0}.$$

V *is a linear subspace of* $\mathsf{T}((V))$ *by identifying* $\mathbf{v} \in V$ *as* $(0, \mathbf{v}, \mathbf{0}, \mathbf{0}, \ldots) \in \mathsf{T}((V))$*. Further,* $\mathsf{T}((V))$ *is also an associative unital algebra with unit* $\mathbf{1} = (1, \mathbf{0}, \ldots) \in \mathsf{T}((V))$*. It is endowed with a (noncommutative*[2] *when* $\dim V > 1$*) product defined for* $\mathbf{s}, \mathbf{t} \in \mathsf{T}((V))$

$$\mathbf{s} \cdot \mathbf{t} = \left(\sum_{i=0}^{m} \mathbf{s}_i \otimes \mathbf{t}_{m-i} \right)_{m \geq 0} \in \mathsf{T}((V)). \tag{1.2}$$

An element $\mathbf{t} = (t_0, \mathbf{t}_1, \ldots) \in \mathsf{T}((V))$ *is invertible if and only if* $t_0 \neq 0$*, with inverse*

$$\mathbf{t}^{-1} = \frac{1}{t_0} \sum_{n \geq 0} \left(\mathbf{1} - \frac{\mathbf{t}}{t_0} \right)^n$$

[1] An algebra A is a vector space A, where one can multiply elements together, i.e. $\mathbf{ab} \in A$ for $\mathbf{a}, \mathbf{b} \in A$.

[2] Noncommutative means that $\mathbf{ab} \neq \mathbf{ba}$ in general for elements $\mathbf{a}, \mathbf{b} \in V$ of the algebra.

Definition 1.6 (Tensor algebra)**.** *Another commonly encountered formulation of the tensor algebra consists of sequences of tensors with only finitely many non-zero coefficients, which we refer to simply as the tensor algebra. It is defined as*

$$\mathsf{T}(V) = \bigoplus_{m\geq 0} V^{\otimes m} = \left\{(t_0, \mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_M, \mathbf{0}, \mathbf{0}, \ldots) : \mathbf{t}_m \in V^{\otimes m} \text{ for } m \in \mathbb{N}, M \in \mathbb{N}\right\}.$$

All algebraic operations carry over from $\mathsf{T}((V))$ *to* $\mathsf{T}(V)$*. Hence, we may equivalently define* $\mathsf{T}(V)$ *as the subalgebra of* $\mathsf{T}((V))$ *where all but finitely many projections are zero.*

For example, take $V = \mathbb{R}^d$, then the degree-0 term is a scalar, the degree-1 term is a vector, the degree-2 term is matrix, the degree-3 term is a $d \times d \times d$ array, and so forth. This process of turning V into an algebra $\mathsf{T}(V)$ or $\mathsf{T}((V))$ is a free construction; informally this means that this is the minimal structure that turns V into an algebra; for more details about tensor algebras, see [299, 155, 217].

Definition 1.7 (Tensor algebra over a Hilbert space)**.** *Now, let us consider the case when* $V = \mathcal{H}$ *is a Hilbert space. We now define for* $\mathbf{s}, \mathbf{t} \in \bigoplus_{m\geq 0} \mathcal{H}^{\otimes m}$ *their inner product as*

$$\langle \mathbf{s}, \mathbf{t} \rangle_{\bigoplus_{m\geq 0} \mathcal{H}^{\otimes m}} = \sum_{m\geq 0} \langle \mathbf{s}_m, \mathbf{t}_m \rangle_{\mathcal{H}^{\otimes m}},$$

where the inner product $\langle \mathbf{s}_m, \mathbf{t}_m \rangle_{\mathcal{H}^{\otimes m}}$ *on* $\mathcal{H}^{\otimes m}$ *is as in* (1.1)*. Finally, the topological completion of* $\bigoplus_{m\geq 0} \mathcal{H}^{\otimes m}$ *in this inner product gives a Hilbert space* $\mathsf{T}(\mathcal{H})$*, which we call the tensor algebra over the Hilbert space* $\mathcal{H}$*, which can be equivalently defined as*

$$\mathsf{T}(\mathcal{H}) = \{\mathbf{t} = (t_0, \mathbf{t}_1, \mathbf{t}_2, \ldots) : \mathbf{t}_m \in \mathcal{H}^{\otimes m}, \langle \mathbf{t}, \mathbf{t} \rangle_{\mathsf{T}(\mathcal{H})} < \infty\}.$$

1.1.3 Duals of tensors

In this section, we study linear functionals acting on tensor algebras over vector spaces following the exposition in [182]. Recall that the algebraic dual $V^\star$ for a vector space V denotes the set of all linear functions from V to $\mathbb{R}$, i.e. $L(V, \mathbb{R})$. When V is finite-dimensional, V and $V^\star$ are isomorphic in a basis dependent way, and in particular $\dim V = \dim V^\star$. Let $\{\mathbf{e}_1, \ldots, \mathbf{e}_d\}$ be a basis of V. This induces a dual basis $\{\mathbf{e}_1^\star, \ldots \mathbf{e}_d^\star\}$ of $V^\star$ uniquely defined by

$$\left\langle \mathbf{e}_i^\star, \mathbf{e}_j \right\rangle = \delta_{i,j} = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$

Similarly, we know that the basis of $V^{\otimes m}$ is given by

$$\left\{\mathbf{e}_{i_1} \otimes \cdots \otimes \mathbf{e}_{i_m} \,:\, i_1, \ldots, i_m \in [d]\right\},$$

and the basis of $(V^\star)^{\otimes m}$ is given by the elements

$$\left\{\mathbf{e}^\star_{i_1} \otimes \cdots \otimes \mathbf{e}^\star_{i_m} \,:\, i_1, \ldots, i_m \in [d]\right\}.$$

Now, note that $\left(V^{\otimes m}\right)^\star \cong (V^\star)^{\otimes m}$, where the isomorphism is given by identifying the basis of $\left(V^{\otimes m}\right)^\star$ with the basis of $(V^\star)^{\otimes m}$ such that

$$\mathbf{e}^\star_{i_1} \otimes \cdots \otimes \mathbf{e}^\star_{i_m} \mapsto \left(\mathbf{e}_{i_1} \otimes \cdots \otimes \mathbf{e}_{i_m}\right)^\star,$$

and this isomorphism is natural, that is, independent of the choice of basis for V, i.e.

$$\left\langle \mathbf{e}^\star_{i_1} \otimes \cdots \otimes \mathbf{e}^\star_{i_m}, \mathbf{e}_{j_1} \otimes \cdots \otimes \mathbf{e}_{j_m} \right\rangle = \delta_{i_1,j_1} \cdots \delta_{i_m,j_m}.$$

The linear action of $(E^\star)^{\otimes m}$ on $E^{\otimes m}$ extends naturally to a linear mapping $(E^\star)^{\otimes m} \to \mathsf{T}((V))^\star$ defined for a $\mathbf{t} = (t_0, \mathbf{t}_1, \ldots) \in \mathsf{T}((V))$ by

$$\left\langle \mathbf{e}^\star_{i_1} \otimes \cdots \otimes \mathbf{e}^\star_{i_m}, \mathbf{t} \right\rangle = \left\langle \mathbf{e}^\star_{i_1} \otimes \cdots \otimes \mathbf{e}^\star_{i_m}, \mathbf{t}_m \right\rangle.$$

In essence, if we think of $\mathbf{t}$ as a non-commuting power series in the letters $\{i_1, \ldots i_d\}$, then this action picks up the coefficient of the word $I = (i_1, \ldots i_m) \in [d]^m$. Analogously, by taking linear combinations of such linear functionals, the action of $\mathsf{T}(V^\star)$ can be extended to a linear mapping $\mathsf{T}(V^\star) \to \mathsf{T}((V))^\star$, and with this identification, $\mathsf{T}(V^\star)$ can be identified as a strict subspace of $\mathsf{T}((V))^\star$. It is important that $\mathsf{T}(V^\star)$ contains sequences of tensors with finitely many nonzero coefficients, since this allows to consider its action on $\mathsf{T}((V))$ without any issues of convergence.

Next, we introduce a product on $\mathsf{T}(V^\star)$ which turns it into a commutative algebra.

Definition 1.8 (Shuffle product [179])**.** *Let $\left\{\mathbf{e}^\star_1, \ldots, \mathbf{e}^\star_d\right\}$ be a basis of $V^\star$, and let $I = (i_1, \ldots, i_n) \in [d]^n$, $J = (i_{n+1}, \ldots, i_{n+m}) \in [d]^m$ be arbitrary multi-indices. The shuffle product of two elementary tensors $\mathbf{e}^\star_I \in (V^\star)^{\otimes n}$ and $\mathbf{e}^\star_J \in (V^\star)^{\otimes m}$ is defined as*

$$\mathbf{e}^\star_I \sqcup\!\sqcup \mathbf{e}^\star_J \sum_{\sigma \in \mathrm{Sh}(n,m)} \mathbf{e}^\star_{i_{\sigma^{-1}(1)}} \otimes \cdots \otimes \mathbf{e}^\star_{i_{\sigma^{-1}(n+m)}},$$

where the summation is taken over all (n,m)-shuffles, that is over the subset of all permutations of $\{1,2,\dots,n+m\}$, that satisfy $\sigma(1)<\cdots<\sigma(n)$ and $\sigma(n+1)<\cdots<\sigma(n+m)$. It can be checked that this construction does not depend on the choice of basis, and turns $\mathsf{T}(V^\star)$ into a commutative algebra by considering its linear extension.

1.1.4 Motivating example: polynomial features

Finally, before introducing path signatures, we consider a motivating example in the spirit of statistical learning, and recall how polynomials in the coordinates of a vector $\mathbf{x}=(x_1,\dots,x_d)\in\mathbb{R}^d$ can be used to approximate continuous functions on compact sets, a property known as universality in the machine learning literature.

Definition 1.9 (Tensor exponential)**.** *The tensor exponential is defined as*

$$\exp_\otimes : V\to\mathsf{T}((V)),\quad \exp_\otimes(\mathbf{x})=\left(1,\mathbf{x},\frac{\mathbf{x}^{\otimes 2}}{2!},\dots,\frac{\mathbf{x}^{\otimes m}}{m!},\dots\right).$$

That is, the map $\exp_\otimes$ takes a vector $\mathbf{x}\in\mathbb{R}^d$ and maps it to the infinite sequence of tensors, which has as its degree-m term $\exp_{\otimes_m}(\mathbf{x})=\mathbf{x}^{\otimes m}/m!$.

Let $V=\mathbb{R}^d$. We can build a functions $f:V\to\mathbb{R}$ by taking linear functionals of $\exp_\otimes$, that is for $\ell\in\mathsf{T}(V^\star)$, we may consider the composition

$$f:\mathbf{x}\mapsto\ell\odot\exp_\otimes(\mathbf{x})=\left\langle\ell,\exp_\otimes(\mathbf{x})\right\rangle.$$

Hence, any choice of $\ell\in\mathsf{T}(V^\star)$ induces a function $f:V\to\mathbb{R}$, which is a linear function in terms of $\exp_\otimes$. It might be helpful to spell out the definition of f in coordinates. By the definition of $\ell=(\ell_0,\ell_1,\dots,\ell_M,\mathbf{0},\dots)\in\mathsf{T}(V^\star)$, we have for some $M\in\mathbb{N}$

$$\left\langle\ell,\exp_\otimes(x)\right\rangle=\sum_{m=0}^{M}\frac{1}{m!}\left\langle\ell_m,\mathbf{x}^{\otimes m}\right\rangle.$$

Spelled out in coordinates, $\mathbf{x}=(x_1,\dots,x_d)$ and $\ell_m=\sum_{I\in[d]^m}a_I\mathbf{e}_I^\star$, this reads as

$$\left\langle\ell_m,\mathbf{x}^{\otimes m}\right\rangle=\sum_{i_1,\dots,i_m\in[d]}a_{i_1,\dots,i_m}\mathbf{e}^\star_{i_1,\dots,i_m}(\mathbf{x})=\sum_{i_1,\dots,i_m\in[d]}a_{i_1,\dots,i_m}x_{i_1}\cdots x_{i_m}.$$

Thus putting this together, formulated in coordinates f is written as

$$f(\mathbf{x})=a_0+a_1x_1+\cdots a_dx_d$$

$$
\begin{aligned}
&+ \frac{1}{2!}\left(a_{1,1}x_1^2 + a_{1,2}x_1x_2 + \cdots + a_{d,d}x_d^2\right)\\
&+ \vdots\\
&+ \frac{1}{M!}\left(a_{1,\ldots,1}x_1^M + \cdots + a_{d,\ldots,d}x_d^M\right)
\end{aligned}
$$

which is a polynomial of degree-m. The Weierstrass approximation theorem guarantees that continuous functions can be uniformly approximated over compact sets in terms of polynomials. This property makes polynomials a classical choice for machine learning, making it possible to learn from vector-valued data. Due to this approximation property, fit datasets well, but they can exhibit arbitrarily large oscillations between the interpolation points, and they are often replaced by other nonlinearities [216].

1.2 Path signatures

In this part, let $(V, \|\cdot\|_V)$ be a finite-dimensional Banach space. In the following, we define the signature of a path and shed light on its properties, which make it a viable feature map for learning from streamed data. Path signatures can be seen as a natural generalization of the tensor exponential map, but instead of mapping a point in V to a sequence of tensors, its domain is the space of paths. It generalizes many of the nice properties of polynomial features such as universality and simultaneously gives the option to ignore time-parameterization without an explicit search over all possible time changes (like in dynamic time warping approaches).

1.2.1 Definition

Definition 1.10. *A continuous path $\mathbf{x} : [0, T] \to V$ is of bounded variation if*

$$
\|\mathbf{x}\|_{1\text{-var}} = \sup_{D \subset [0,T]} \sum_{t_i \in D} \left\|\mathbf{x}_{t_i} - \mathbf{x}_{t_{i-1}}\right\|_V < \infty,
$$

where $D = \{0 = t_0 < t_1 < \cdots < t_n = T \,:\, n \in \mathbb{N}\}$ runs over all finite partitions of $[0, T]$.

We denote the set of bounded variation paths on some arbitrary time horizon by

$$
\mathsf{Paths}(V) = \{\mathbf{x} \in C([0, T], V) : T \in \mathbb{R}_+, \mathbf{x}_0 = 0, \|\mathbf{x}\|_{1\text{-var}} < \infty\},
$$

To ease the notation, let us define the continuous m-simplex:

$$
\Delta_m([0, T]) = \{(t_1, \ldots, t_m) \in \mathbb{R}^m \,:\, 0 < t_1 < \cdots < t_m < T\},
$$

and for a path $(\mathbf{x}_t)_{t\in[0,T]} \in \mathsf{Paths}(V)$ denote its time horizon by $\mathsf{T}_{\mathbf{x}} = T$.

Definition 1.11. *A classic way to obtain a structured and hierarchical description of a path* $\mathbf{x} \in \mathsf{Paths}(V)$ *is by computing a sequence of iterated integrals called the path signature given as tensors of increasing degrees such that the degree-m object is*

$$\mathsf{S}_m(\mathbf{x}) = \int_{\mathbf{t}\in\Delta_m([0,\mathsf{T}_x])} \mathrm{d}\mathbf{x}_{t_1} \otimes \cdots \otimes \mathrm{d}\mathbf{x}_{t_m} = \int_{\mathbf{t}\in\Delta_m([0,\mathsf{T}_x])} \dot{\mathbf{x}}_{t_1} \otimes \cdots \otimes \dot{\mathbf{x}}_{t_m}\, \mathrm{d}t_1 \cdots \mathrm{d}t_m, \quad (1.3)$$

where the integrals are defined in the Riemann–Stieltjes sense. Formally, we refer to the map that takes a path to its iterated integrals as the path signature map, and it is defined as

$$\mathsf{S} : \mathsf{Paths}(V) \to \mathsf{T}((V)), \quad \mathsf{S}(\mathbf{x}) = (1, \mathsf{S}_1(\mathbf{x}), \mathsf{S}_2(\mathbf{x}), \ldots).$$

For a path $(\mathbf{x}_t)_t = (x_t^1, \ldots, x_t^d)_t \in \mathsf{Paths}(V)$ that evolves in V, one can spell this out in coordinates: the m-th signature level $\mathsf{S}_m(\mathbf{x}) \in V^{\otimes m}$ is the tensor that has as its $(i_1, \ldots, i_m) \in [d]^m$–th coordinate entry the real number computed by the Riemann–Stieltjes integral

$$\mathsf{S}_m(\mathbf{x})_{i_1,\ldots,i_m} = \int_{\mathbf{t}\in\Delta_m([0,\mathsf{T}_{\mathbf{x}}])} \mathrm{d}x_{t_1}^{i_1} \cdots \mathrm{d}x_{t_m}^{i_m} = \int_{\mathbf{t}\in\Delta_m([0,\mathsf{T}_{\mathbf{x}}])} \dot{x}_{t_1}^{i_1} \cdots \dot{x}_{t_m}^{i_m}\, \mathrm{d}t_1 \cdots \mathrm{d}t_m$$

where $\dot{x}_t^i = \frac{\mathrm{d}x_t^i}{\mathrm{d}t}$ denotes the time derivative.

Proposition 1.12 (Factorial decay [182])**.** *Let* $\mathbf{x} \in \mathsf{Paths}(V)$ *be a bounded variation path. We have for the level-m path signature feature for* $m \in \mathbb{N}$ *that*

$$\|\mathsf{S}_m(\mathbf{x})\|_{V^{\otimes m}} \leq \frac{\|\mathbf{x}\|_{1\text{-var}}^m}{m!}.$$

Next, we show that there is a precise connection between the algebra structure of $\mathsf{T}((\mathbb{R}^d))$ and operations on paths. Hence, we first define the concatenation operation on paths, and then state how this affects the signature.

Definition 1.13 (Path concatenation)**.** *Let* $\mathbf{x}, \mathbf{y} \in \mathsf{Paths}(V)$ *such that* $\mathbf{x} : [0, S] \to V$ *and* $\mathbf{y} : [0, T] \to V$*. Their concatenation denoted by* $\mathbf{x} \star \mathbf{y}$ *is defined for* $t \in [0, S+T]$ *as*

$$(\mathbf{x} \star \mathbf{y})_t = \begin{cases} \mathbf{x}_t & \text{if } t \in [0, S] \\ \mathbf{x}_S - \mathbf{y}_0 + \mathbf{y}_{t-S} & \text{if } t \in [S, S+T]. \end{cases}$$

Proposition 1.14 (Chen identity [182]). *Let* $\mathbf{x}, \mathbf{y} \in \mathsf{Paths}(V)$. *Then, we have that*

$$\mathsf{S}(\mathbf{x} \star \mathbf{y}) = \mathsf{S}(\mathbf{x}) \cdot \mathsf{S}(\mathbf{y}).$$

Similarly, path reversal can also be represented in terms of the signature.

Proposition 1.15 (Path reversal [182]). *Let* $\mathbf{x} \in \mathsf{Paths}(V)$ *defined on* $[0, T]$ *taking values in* V. *Denote by* $(\overleftarrow{\mathbf{x}}_t)_{t\in[0,T]} = (\mathbf{x}_{T-t})_{t\in[0,T]}$. *Then, we have that*

$$\mathsf{S}(\overleftarrow{\mathbf{x}}) = \mathsf{S}^{-1}(\mathbf{x}).$$

In particular, the range of $\mathsf{S} : \mathsf{Paths}(V) \to \mathsf{T}((V))$ *is a group.*

1.2.2 Properties

Invariance under tree-like equivalence. A classic result going back to Chen [40] shows that the map $\mathbf{x} \mapsto \mathsf{S}(\mathbf{x})$ is injective up to tree-like equivalence. Loosely speaking, tree-like equivalence is from a purely analytic point of view more natural to work with than reparameterization since tree-like equivalence between paths is analogous to Lebesgue almost sure equivalence between sets. Howevever, from a practical point of view, the difference between paths that are tree-like equivalent and paths that differ by a reparameterization is negligible and we invite the reader to use them as synonyms throughout. Nevertheless, we give the precise definition below and refer the interested reader to [110] for a detailed discussion.

Definition 1.16. *A bounded variation path* $\mathbf{x} \in \mathsf{Paths}(V)$ *is* tree-like *if there exists a continuous function* $h : [0, T] \to [0, \infty)$ *such that* $h(0) = h(T) = 0$ *and for all* $s < t$

$$\|\mathbf{x}_t - \mathbf{x}(s)\|_V \leq h(s) + h(t) - 2 \inf_{u\in[s,t]} h(u).$$

Proposition 1.17. *Let* $\mathbf{x}, \mathbf{y} \in \mathsf{Paths}(V)$ *be two paths of bounded variation. Then*

$$\mathsf{S}(\mathbf{x}) = \mathsf{S}(\mathbf{y})$$

if and only if $\mathbf{x} \star \overleftarrow{\mathbf{y}}$ *is tree-like where* $\star$ *denotes path concatenation and* $\overleftarrow{\mathbf{y}}$ *denotes path reversal.*

In particular for $\mathbf{x}, \mathbf{y} \in \mathsf{Paths}(V)$ and $\ell \in \mathsf{T}(V^\star)$, for any functions of the form $f(\mathbf{x}) = \langle \ell, \mathsf{S}(\mathbf{x})\rangle$, $f(\mathbf{x}) = f(\mathbf{y})$ if and only if $\mathbf{x}$ and $\mathbf{y}$ differ by parameterization (strictly speaking, by a tree-like equivalence). This ability to factor out time-invariance can be very powerful since

the space of all possible parameterizations is huge and we never make an explicit search over possible time changes like in the DTW distance.

However, we are often interested in time parameterization, and the path signature can be made sensitive to it by adding an extra coordinate to the path $\mathbf{x} \in \mathsf{Paths}(V)$ before computing the signature features of this enhanced path $\hat{\mathbf{x}}_t = (t, \mathbf{x}_t) \in \mathbb{R} \oplus V$. In other words, we make the parameterization part of the trajectory. The augmented path $\hat{\mathbf{x}}$ is never tree-like since the first coordinate is strictly increasing. Hence, the map $\mathsf{Paths}(V) \ni \mathbf{x} \mapsto \mathsf{S}(\hat{\mathbf{x}}) \in \mathsf{T}((\mathbb{R} \oplus V))$ is injective.

Universality. One of the most attractive properties of the monomial feature map $\mathbf{x} \to \exp_\otimes(\mathbf{x})$ for vectors $\mathbf{x} \in V$, is that any continuous function $f : V \to \mathbb{R}$ can be uniformly approximated on compact sets as a linear functional of $\exp_\otimes$, that is $f(\mathbf{x}) \approx \langle \ell, \exp_\otimes(\mathbf{x}) \rangle$ for some $\ell \in \mathsf{T}(V^\star)$. The reason is that linear combinations of monomials (polynomials) form an algebra and the Stone–Weierstrass theorem applies. Such approximation properties of feature maps are usually referred to as "universality" in the ML literature. One of the most attractive properties of the signature feature map $\mathbf{x} \mapsto \mathsf{S}(\mathbf{x})$ for paths $\mathbf{x} \in \mathsf{Paths}(V)$ is that a universality result holds. A central result in this direction is the shuffle identity for path signatures, which we state below.

Theorem 1.18 (Shuffle identity [182])**.** *Let* $\mathbf{x} \in \mathsf{Paths}(V)$*, and* $\ell_1, \ell_2 \in \mathsf{T}(V^\star)$*. Then,*

$$\langle \ell_1, \mathsf{S}(\mathbf{x}) \rangle \langle \ell_2, S(\mathbf{x}) \rangle = \langle \ell_1 \sqcup\!\sqcup \ell_2, \mathsf{S}(\mathbf{x}) \rangle$$

In particular, the range of the path signature forms an algebra of real-valued functons. This means that the Stone-Weierstrass theorem applies, and linear functionals acting on the signature allow to approximate continuous functions on paths.

Proposition 1.19. *For every continuous* $f : \mathsf{Paths}(V) \to V$*,* $\mathcal{K} \subset \mathsf{Paths}(V)$ *compact,* $\epsilon > 0$ *there exists a* $\ell \in \mathsf{T}(V^\star)$ *such that*

$$\sup_{\mathbf{x} \in \mathcal{K}} |f(\mathbf{x}) - \langle \ell, \mathsf{S}(\hat{\mathbf{x}}) \rangle| < \epsilon. \tag{1.4}$$

The analogous result holds for paths $\mathbf{x} \in \mathsf{Paths}(V)$ *without a time coordinate by replacing the domain of paths by equivalence classes of paths under tree-like equivalence. For a proof and many extensions, see [44].*

Informally, iterated integrals of paths can be seen as a generalization of classical monomials and from this perspective, the approximation (1.4) can be regarded as the extension of classic polynomial regression to path-valued data. Thus at least informally it is not surprising, that

vanilla signature features suffer from similar drawbacks as classic monomial features; for example, if classic monomials are replaced by other nonlinearities this often drastically improves the approximations; see Section 1.2.4 for kernelized methods.

1.2.3 Examples

Linear paths. Consider the path $\mathbf{x} : [0,1] \to V$ that just runs along a straight line $\mathbf{x} : t \mapsto t\mathbf{v}$, where $\mathbf{v} \in \mathbb{R}^d$ is a given vector. Plugging this into the definition of the path signature (1.3), we get by a direct calculation that

$$\mathsf{S}_m(\mathbf{x}) = \frac{\mathbf{v}^{\otimes m}}{m!} \in V^{\otimes m}, \quad \mathsf{S}(\mathbf{x}) = \exp_\otimes(\mathbf{v}) \in \mathsf{T}((V))$$

We see that for this special case the that $\mathbf{x}$ is determined by its increment $\mathbf{x}_1 - \mathbf{x}_0 = \mathbf{v}$, the path signature $\mathsf{S}(\mathbf{x})$ equals the monomial feature map $\exp_\otimes(\mathbf{v})$ of the total increment $\mathbf{v} = \mathbf{x}_1 - \mathbf{x}_0$. This shows how in this special case the signature reduces to monomial features, and this is one of the many reasons why signatures are regarded as "polynomials of paths".

Piecewise linear paths. In general, these integrals need to be computed by standard integration techniques, but for a piecewise linear path $\mathbf{x} : [0,T] \to V$, that is partitioned into L disjoint intervals, $[0,T] = \bigsqcup_{i=1}^{L} [t_{i-1}, t_i]$, and $\mathbf{x}$ is piecewise linear on each of these pieces with increments $\mathbf{x}_{t_i} - \mathbf{x}_{t_{i-1}} = \mathbf{v}_i$ for some $\mathbf{v}_i \in V$ for $i \in [L]$, these iterated integrals can be computed by Chen's identity (Proposition 1.14) via

$$\mathsf{S}(\mathbf{x}) = \mathsf{S}(\mathbf{x}|_{[t_0,t_1]}) \cdot \mathsf{S}(\mathbf{x}|_{[t_1,t_2]}) \cdots \mathsf{S}(\mathbf{x}|_{[t_{L-1},t_L]}) = \exp_\otimes(\mathbf{v}_1) \cdot \exp_\otimes(\mathbf{v}_2) \cdots \exp_\otimes(\mathbf{v}_L), \quad (1.5)$$

where $\mathbf{x}|_{[t_i,t_{i+1}]}$ denotes the restriction of $\mathbf{x}$ to $[t_i, t_{i+1}]$. Performing the algebra multiplication from (1.2), we get that the degree-m term reduces to an iterated sum of the form

$$\mathsf{S}_m(\mathbf{x}) = \sum_{1 \le i_1 \le \cdots \le i_m \le L} \frac{1}{\mathbf{i}!} \mathbf{v}_{i_1} \otimes \cdots \otimes \mathbf{v}_{i_m} \in V^{\otimes m}, \quad (1.6)$$

where the coefficient $\mathbf{i}!$ is such that if there are $k \in [m]$ distinct entries among $i_1, \ldots, i_m$ with number of repetitions $p_1, \ldots, p_k$, then $\mathbf{i}! = p_1! \cdots p_k!$.

1.2.4 Signature kernels

Kernel trick for path signatures. Signature kernels provide a means to compare paths through their signatures by defining inner products in the tensor algebra. Given their structure,

they offer a powerful similarity measure for sequences of potentially different lengths, making them particularly suitable for sequential data analysis. The signature kernel inherently captures the algebraic and geometric structure of the paths by inheriting the powerful properties of path signatures, such as invariance to reparameterization and universality, see [33, Thm. 2.2.4] for a proof that the signature kernel is universal in the kernel learning sense.

Next, we will consider the setting when $\mathcal{H}$ is a Hilbert space, and our bounded variation paths evolve in this space, that is, $\mathsf{Paths}(\mathcal{H})$. Then, the signature kernel $\mathsf{k}_{\mathsf{S}} : \mathsf{Paths}(\mathcal{H}) \times \mathsf{Paths}(\mathcal{H}) \to \mathbb{R}$ is defined for paths $\mathbf{x}, \mathbf{y} \in \mathsf{Paths}(\mathcal{H})$ by taking the inner product of signatures

$$\mathsf{k}_{\mathsf{S}}(\mathbf{x}, \mathbf{y}) = \langle \mathsf{S}(\mathbf{x}), \mathsf{S}(\mathbf{y}) \rangle_{\mathsf{T}(\mathcal{H})} = \sum_{m=0}^{\infty} \langle \mathsf{S}_m(x), \mathsf{S}_m(y) \rangle_{\mathcal{H}^{\otimes m}}$$

where $\langle \cdot, \cdot \rangle_{\mathsf{T}(\mathcal{H})}$ denotes the inner product in the tensor algebra $\mathsf{T}(\mathcal{H})$ over the Hilbert space $\mathsf{T}(\mathcal{H})$, which itself is indeed a Hilbert space as discussed in Section 1.1.2. In practice, one often introduces a truncation factor $M \in \mathbb{N} \cup \{\infty\}$, and considers

$$\mathsf{k}_{\mathsf{S}_{\leq M}}(x, y) = \sum_{m=0}^{M} \langle \mathsf{S}_m(x), \mathsf{S}_m(y) \rangle_{\mathcal{H}^{\otimes m}},$$

and with some abuse of notation we may omit the truncation superscript, that is, when talking about the signature kernel k_{S}, we may mean the truncated version.

First, [146] developed an algorithm for finite M, and later [228] formulated a PDE to solve for infinite M. The kernel trick of [146] is written on the level-m for $\mathbf{x}, \mathbf{y} \in \mathsf{Paths}(\mathcal{H})$ as

$$\mathsf{k}_{\mathsf{S}_m}(\mathbf{x}, \mathbf{y}) = \int_{\substack{\mathbf{s} \in \Delta_m([0, \mathsf{T}_{\mathbf{x}}]) \\ \mathbf{t} \in \Delta_m([0, \mathsf{T}_{\mathbf{y}}])}} \left\langle \mathrm{d}\mathbf{x}_{s_1}, \mathrm{d}\mathbf{y}_{t_1} \right\rangle_{\mathcal{H}} \cdots \left\langle \mathrm{d}\mathbf{x}_{s_m}, \mathrm{d}\mathbf{y}_{t_m} \right\rangle_{\mathcal{H}}. \tag{1.7}$$

This allows to compute k_{S} solely by inner product evaluation on $\mathcal{H}$.

Lifting paths by static kernels. Since the kernel trick introduced above allows to compute the inner product of path signatures only by inner product evaluation, it is often beneficial to preprocess paths by lifting them to a higher-dimensional space, which can enhance the richness of the information captured by the signature. One such tool for this is the method of reproducing kernel Hilbert spaces (RKHS), which has become wide-spread in machine learning, since it allows for nonlinear learning in infinite-dimensional spaces using the kernel trick [21].

Now let $\mathcal{X}$ be a set, which will be our data space where our paths take values. Further, let $\mathsf{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a so-called static kernel on $\mathcal{X}$ with RKHS $\mathcal{H}_{\mathsf{k}}$. Let us denote for an $\mathbf{x} \in \mathcal{X}$, $\mathsf{k}_{\mathbf{x}} = \mathsf{k}(\mathbf{x}, \cdot) \in \mathcal{H}_{\mathsf{k}}$ the reproducing kernel lift of $\mathbf{x}$. Analogously, for a path $(\mathbf{x}_t)_{t \in [0,T]} \in \mathsf{Paths}(\mathbf{x})$

denote by $\mathsf{k}_{\mathbf{x}} = (\mathsf{k}_{\mathbf{x}_t})_{t\in[0,T]} \in \mathsf{Paths}(\mathcal{H}_{\mathsf{k}})$ the path $\mathbf{x}$ lifted into the RKHS of k, $\mathcal{H}_{\mathsf{k}}$. Then, (1.7) can formally be written for the lifts of $\mathbf{x}, \mathbf{y} \in \mathsf{Paths}(\mathcal{X})$ as

$$\begin{aligned}\mathsf{k}_{\mathsf{S}_m}(\mathsf{k}_{\mathbf{x}}, \mathsf{k}_{\mathbf{y}}) &= \int_{\substack{\Delta_m([0,\mathsf{T}_{\mathbf{x}}])\\ \Delta_m([0,\mathsf{T}_{\mathbf{y}}])}} \left\langle \mathrm{d}\mathsf{k}_{\mathbf{x}_{s_1}}, \mathrm{d}\mathsf{k}_{\mathbf{y}_{t_1}} \right\rangle_{\mathcal{H}_{\mathsf{k}}} \cdots \left\langle \mathrm{d}\mathsf{k}_{\mathbf{x}_{s_m}}, \mathrm{d}\mathsf{k}_{\mathbf{y}_{t_m}} \right\rangle_{\mathcal{H}_{\mathsf{k}}} \\ &= \int_{\substack{\Delta_m([0,\mathsf{T}_{\mathbf{x}}])\\ \Delta_m([0,\mathsf{T}_{\mathbf{y}}])}} \mathrm{d}\kappa(s_1, t_1), \cdots \mathrm{d}\kappa(s_m, t_m),\end{aligned}$$

where $\kappa([s,t]\times[u,v]) = \mathsf{k}(\mathbf{x}_t, \mathbf{y}_v) - \mathsf{k}(\mathbf{x}_t, \mathbf{y}_u) - \mathsf{k}(\mathbf{x}_s, \mathbf{y}_v) + \mathsf{k}(\mathbf{x}_s, \mathbf{y}_u)$ is a signed Borel measure on $[0, \mathsf{T}_{\mathbf{x}}]\times[0, \mathsf{T}_{\mathbf{y}}]$ for fixed $\mathbf{x}, \mathbf{y} \in \mathsf{Paths}(\mathcal{X})$, see [146, Thm. 2] for a proof. This allows to compute the signature kernel for paths lifted into reproducing kernel Hilbert spaces, that significantly enhances the expressiveness of the signature kernel. Note that $\mathcal{H}_k$ is a genuine infinite-dimensional space for non-degenerate kernels k, for which there are various choices, see [216, Ch. 4], e.g. the RBF or Matérn family of kernels. Abstractly, we can think of the signature kernel as taking a static kernel k on the state space $\mathcal{X}$ and turning it into a kernel k_{S} for paths evolving in that state space. In the following, we will treat the static kernel k as a hyperparameter of k_{S}, and we do not make the composition explicit, i.e. for $\mathbf{x}, \mathbf{y} \in \mathsf{Paths}(\mathcal{X})$, we will write $\mathsf{k}_{\mathsf{S}}(\mathbf{x}, \mathbf{y})$ and implicitly mean that $\mathbf{x}$ and $\mathbf{y}$ are first lifted into an RKHS before taking signatures. The standard formulation of signatures when $\mathcal{X} = \mathcal{E}$ is a Euclidean space can be recovered by using the linear kernel $\mathsf{k}(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y}\rangle_{\mathcal{E}}$ as the static kernel lift for $\mathbf{x}, \mathbf{y} \in \mathcal{E}$.

1.2.5 Rough paths

So far we assumed that our input domain consists of paths of bounded variation paths, but in the real-world the evolution of quantities is often subject to noise, e.g. a classical model in physics and engineering is $\mathbf{x}_t = \mathbf{a}_t + B_t$, where $\mathbf{a}$ is a bounded variation path, but B is a Brownian sample path. Since Brownian sample paths are not of bounded variation, $\mathbf{x}$ is not of bounded variation. However, the iterated integrals of $\mathbf{x}$ can still be defined as above, but one has to replace the iterated Riemann–Stieltjes integrals by Ito–Stratonovich integrals in the definition of $\mathsf{S}(\mathbf{x})$. Even rougher trajectories such as fractional Brownian motion and non-Markovian processes can be handled that way with so-called rough path integrals. Rough path theory provides a systematic study that comes with a rich toolbox combining analytic and algebraic estimates, rich enough to cover the trajectories of large classes of stochastic processes; see [182, 178, 87].

1.2.6 Discrete-time setting

In practice, we often do not have access to paths, but are only given discrete-time observations that lie in some state space $\mathcal{X}$, that is, we work with the space of finite length sequences

$$\mathsf{Seq}(\mathcal{X}) = \{\mathbf{x} = (\mathbf{x}_0, \dots, \mathbf{x}_L) : \mathbf{x}_0, \dots, \mathbf{x}_L \in \mathcal{X}, \mathbf{x}_0 = \mathbf{0}, L \in \mathbb{N}\},$$

where we denote the "effective" length of a sequence $\mathbf{x} = (\mathbf{x}_0, \dots, \mathbf{x}_L) \in \mathsf{Seq}(\mathcal{X})$ by $\mathsf{L}_\mathbf{x} = L$. Notice that for sequences we use 0-based indexing, that does not count to the length. We make this choice because signatures use sequence increments, which reduces length by 1-step.

Below, we will assume that $\mathcal{X}$ is either a topological vector space (i.e. $\mathbb{R}^d$), or that as a preprocessing step our input sequence $\mathbf{x}$ has been lifted to one using a static feature map $\varphi : \mathcal{X} \to V$, i.e. consider $\varphi(\mathbf{x})$ substituted in place of $\mathbf{x}$ which we do not make explicit here.

Discretized signature features. It is not obvious how to extend path signatures to the discrete-time setting, since they are defined in continuous-time via iterated integrals, and we discuss multiple choices below. First, we consider the so-called "canonical" choice, which proceeds by identifying $\mathbf{x}$ with a continuous time path by its linear interpolation for any choice of time parameterization, and then computing the signature of this path.

To ease the notational burden, we define the following discrete-time m-simplices:

$$\Delta_m(L) = \left\{(i_1, \dots, i_m) \in \mathbb{Z}_+^m : 1 \le i_1 < \cdots < i_m \le L\right\},$$

and its closed companion

$$\bar{\Delta}_m(L) = \left\{(i_1, \dots, i_m) \in \mathbb{Z}_+^m : 1 \le i_1 \le \cdots \le i_m \le L\right\}.$$

Then, under this choice of path embedding (linear interpolation), we can compute the level–m signature of $\mathbf{x} \in \mathsf{Seq}(\mathcal{X})$ using the relation (1.6) by

$$\mathsf{S}_m(\mathbf{x}) = \sum_{\mathbf{i} \in \bar{\Delta}_m(\mathsf{L}_\mathbf{x})} \frac{1}{\mathbf{i}!} \delta\mathbf{x}_{i_1} \otimes \cdots \otimes \delta\mathbf{x}_{i_m}, \tag{1.8}$$

where $\delta\mathbf{x}_i = \mathbf{x}_i - \mathbf{x}_{i-1}$ refers to first-order (backwards) sequence differencing.

Before discussing (1.8) in detail, let us first generalize this formula. Thus, next we consider other choices for approximating iterated path integrals using discrete-time data. We introduce

the order-p approximation to the path signature considered in [146, App. B]. The order-p level-m signature feature $\mathsf{S}^{(p)} : \mathsf{Seq}(\mathcal{X}) \to \mathsf{T}(\mathcal{X})$ of a sequence $\mathbf{x} \in \mathsf{Seq}(\mathcal{X})$ is defined as

$$\mathsf{S}_m^{(p)}(\mathbf{x}) = \sum_{\mathbf{i} \in \bar{\Delta}_m(\mathsf{L}_\mathbf{x})} \frac{1}{\mathbf{i}!^{(p)}} \delta\mathbf{x}_{i_1} \otimes \cdots \otimes \delta\mathbf{x}_{i_m}, \tag{1.9}$$

where $\mathbf{i}!^{(p)}$ is such that if there are k distinct indices in $\mathbf{i}$ with number of repetitions $p_1, \ldots, p_k$ respectively, then it is defined as

$$\mathbf{i}!^{(p)} = \begin{cases} p_1! \cdots p_k! & \text{for } p_1, \ldots, p_k \leq p \\ 0 & \text{otherwise.} \end{cases}$$

Now, we notice that (1.9) is a generalization of (1.8), since the latter can be recovered by choosing $p = m$. As mentioned before, we refer to p as the order hyperparameter of the discretized signature features. Notice that (1.9) summarizes the sequence $\mathbf{x}$ by a summation over its possibly contiguous length-m subsequences with at most p-repetitions for each sequence index, such that each term is normalized by a factor depending on the number of repetitions for each unique index. In particular, we will often focus on the $p = 1$ case, which is often referred to as a non-geometric approximation since there does not exist a continuous-time path such that $\mathsf{S}^{(1)}(\mathbf{x})$ is its path signature. Nevertheless, a key observation of this thesis is that the order-1 case often works well or even better than higher orders, while preserving a considerable amount of computation time and memory. This is explained in [263], which is an extension of Chapter 3, where my collaborator Patric proved that the $p = 1$ case is universal on the space of sequences given the existence a time coordinate which encodes the position within the sequence.

In particular, the $p = 1$ case can be written for a sequence $\mathbf{x} \in \mathsf{Seq}(\mathcal{X})$ as

$$\mathsf{S}_m^{(1)}(\mathbf{x}) = \sum_{\mathbf{i} \in \Delta_m(\mathsf{L}_\mathbf{x})} \delta\mathbf{x}_{i_1} \otimes \cdots \otimes \delta\mathbf{x}_{i_m},$$

i.e. notice that the summation is taken exclusively over non-contiguous subsequences, and as such, can be treated as a generalization of k-mer features appearing in the context of string kernels, see [146, Sec. 5] for a connection to such classic kernels.

We remark that different choices of p correspond to different algebraic embeddings, i.e. replacing the tensor exponential in (1.5) with a different algebra embedding. This is one of the main themes of Chapter 3, and different choices lead to beside mildly different computational complexities and performances, to interesting algebraic questions, see [68, 69] for a discussion.

In the following, we may omit the order superscript when talking about signature features and kernels in the discrete-time setting, and simply treat it as a hyperparameter setting.

Discretized signature kernels. The signature is a powerful feature set for nonlinear regression on streamed data. A computational bottleneck associated with it is the dimensionality of $\mathsf{T}(\mathcal{X})$. As we are dealing with tensors, for $\mathcal{X}$ finite-dimensional $\mathsf{S}_m(\mathbf{x})$ is a tensor of degree-m which has $(\dim \mathcal{X})^m$ coordinates that need to be computed. This can quickly become computationally expensive. For infinite-dimensional $\mathcal{X}$, it is infeasible to directly compute S.

The work of [146] takes the setting when $\mathcal{X} = \mathcal{H}$ is a Hilbert space, and shows a kernel trick that allows to compute signature kernels for discrete data up to finite truncation $M \in \mathbb{Z}_+$ using dynamic programming, even when $\mathcal{H}_\mathsf{k}$ is infinite-dimensional. Subsequently, [228] proposed a PDE-based algorithm to approximate the untruncated signature kernel, which was further extended in [32], and we refer to [163] for a recent overview of signature kernels.

Here, we focus on discrete-time, and continue the discussion of signature kernels. Our starting point is the generalized order-p approach of [146], that results in the features (1.9).

The signature kernel is a powerful formalism that allows to transform any static kernel on $\mathcal{X}$ into a kernel for sequences that evolve in $\mathcal{X}$. Let $\mathsf{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a static kernel with RKHS $\mathcal{H}_\mathsf{k}$. Now, let $\mathbf{x}, \mathbf{y} \in \mathsf{Seq}(\mathcal{X})$, and consider the discretized signature kernel of order-p and level-m $\mathsf{k}^{(p)}_{\mathsf{S}_m} : \mathsf{Seq}(\mathcal{X}) \times \mathsf{Seq}(\mathcal{X}) \to \mathbb{R}$ given as the inner product of order-p (lifted) signatures from (1.9)

$$
\begin{aligned}
\mathsf{k}^{(p)}_{\mathsf{S}_m}(\mathbf{x}, \mathbf{y}) &= \left\langle \mathsf{S}^{(p)}_m(\mathsf{k}_\mathbf{x}), \mathsf{S}^{(p)}_m(\mathsf{k}_\mathbf{x}) \right\rangle_{\mathcal{H}_\mathsf{k}^{\otimes m}} \\
&= \left\langle \sum_{\mathbf{i} \in \bar{\Delta}_m(\mathsf{L}_\mathbf{x})} \frac{1}{\mathbf{i}!^{(p)}} \delta \mathsf{k}_{\mathbf{x}_{i_1}} \otimes \cdots \otimes \delta \mathsf{k}_{\mathbf{x}_{i_m}}, \sum_{\mathbf{j} \in \bar{\Delta}_m(\mathsf{L}_\mathbf{y})} \frac{1}{\mathbf{j}!^{(p)}} \delta \mathsf{k}_{\mathbf{y}_{j_1}} \otimes \cdots \otimes \delta \mathsf{k}_{\mathbf{y}_{j_m}} \right\rangle \\
&= \sum_{\mathbf{i} \in \bar{\Delta}_m(\mathsf{L}_\mathbf{x})} \sum_{\mathbf{j} \in \bar{\Delta}_m(\mathsf{L}_\mathbf{y})} \frac{1}{\mathbf{i}!^{(p)} \mathbf{j}!^{(p)}} \left\langle \delta \mathsf{k}_{\mathbf{x}_{i_1}}, \delta \mathsf{k}_{\mathbf{x}_{j_1}} \right\rangle_{\mathcal{H}_\mathsf{k}} \cdots \left\langle \delta \mathsf{k}_{\mathbf{x}_{i_m}}, \delta \mathsf{k}_{\mathbf{y}_{j_m}} \right\rangle_{\mathcal{H}_\mathsf{k}} \\
&= \sum_{\substack{\mathbf{i} \in \bar{\Delta}_m(\mathsf{L}_\mathbf{x}) \\ \mathbf{j} \in \bar{\Delta}_m(\mathsf{L}_\mathbf{y})}} \frac{1}{\mathbf{i}!^{(p)} \mathbf{j}!^{(p)}} \delta_{i_1, j_1} \mathsf{k}(\mathbf{x}_{i_1}, \mathbf{y}_{j_1}) \cdots \delta_{i_m, j_m} \mathsf{k}(\mathbf{x}_{i_m}, \mathbf{y}_{j_m}),
\end{aligned} \tag{1.10}
$$

where $\delta_{i,j}\mathsf{k}(\mathbf{x}_i, \mathbf{y}_j) = \mathsf{k}(\mathbf{x}_i, \mathbf{y}_j) - \mathsf{k}(\mathbf{x}_i, \mathbf{y}_{j-1}) - \mathsf{k}(\mathbf{x}_{i-1}, \mathbf{y}_j) + \mathsf{k}(\mathbf{x}_{i-1}, \mathbf{y}_{j-1})$ denotes a second-order (backwards) cross-differencing operator. The key insight by [146] is equation (1.10), i.e. that $\mathsf{k}_{\mathsf{S}_m}$ can be computed without computing S_m itself by a kernel trick using only kernel evaluations.

The hyperparameters are: (1) the choice of the static kernel k, for which there is a wide range of options, e.g. for $\mathcal{X} = \mathbb{R}^d$ the RBF, exponential, polynomial or Matérn family of kernels; (2) any hyperparameters that k comes with, such as the bandwidth or polynomial degree; (3) truncation level-M; (4) the choice of algebra embedding, i.e. the order-p; (5) and the choice

of kernel normalization that scales each level $\mathsf{k}_{\mathsf{S}_m}$ appropriately. It also comes with theoretical guarantees such as analytic estimates when sequences converge to paths [146], its maximum mean discrepancy (MMD) [107] metrizes classic topologies for stochastic processes, and can lead to robust statistics in the classic statistical sense (B-robustness); see [44] for details.

Although (1.10) looks expensive to compute, [146] applies dynamic programming to efficiently compute k_{S} for finite M using a recursive algorithm; an alternative algorithm is the above mentioned approach of approximating the (untruncated) signature kernel k_{S} using PDE-discretization. Importantly, (1.10) avoids computing tensors, and only depends on the entry-wise evaluations of the static kernel $\mathsf{k}(\mathbf{x}_i, \mathbf{y}_j)$. Indeed, this leads to a computational cost of $O(\mathsf{L}_{\mathbf{x}}\mathsf{L}_{\mathbf{y}})$ with respect to length, that is feasible to compute for sequences evolving in high-dimensional state-spaces, but only with moderate sequence length. Note that the same bottleneck applies to PDE-based approaches. In part, the aim of Chapter 5 is to alleviate this quadratic cost in sequence length, while approximately enjoying the modelling capability of working within an infinite-dimensional RKHS.

1.3 Algorithms

1.3.1 Notation for algorithms

We define notation based on [146] for describing vectorized computations. We use 1-based indexing for arrays. Note that elements outside the bounds of an array are treated as zeros. Let A and B be k-fold arrays of size $(n_1 \times \cdots \times n_k)$, indexed by $i_j \in \{1, \ldots, n_j\}$ for $j \in \{1, \ldots, k\}$.

We define the following operations:

(i) The cumulative sum along axis j as:

$$A[:, \ldots, :, \boxplus, :, \ldots, :][i_1, \ldots, i_{j-1}, i_j, i_{j+1}, \ldots i_k] = \sum_{\kappa=1}^{i_j} A[i_1, \ldots, i_{j-1}, \kappa, i_{j+1}, \ldots, i_k].$$

(ii) The slice-wise sum along axis j as:

$$A[:, \ldots, :, \Sigma, :, \ldots, :][i_1, \ldots, i_{j-1}, i_{j+1}, \ldots, i_k] = \sum_{\kappa=1}^{n_j} A[i_1, \ldots, i_{j-1}, \kappa, i_{j+1}, \ldots i_k].$$

(iii) The shift along axis j by $+m$ for $m \in \mathbb{Z}$ as:

$$A[:, \ldots, :, +m, :, \ldots, :][i_1, \ldots, i_j, \ldots, i_k] = A[i_1, \ldots, i_j - m, \ldots i_k]$$

(iv) The Hadamard product of arrays A and B:

$$(A \odot B)[i_1, \ldots, i_k] = A[i_1, \ldots, i_k] \cdot B[i_1, \ldots, i_k].$$

(v) Now, if A has shape $(n_1 \times \cdots \times n_j \times \cdots \times n_k)$ and B has shape $(n_1 \times \cdots \times n'_j \times \cdots \times n_k)$, then their Kronecker product along axis j is defined for $i_j \in [n_j]$ and $i'_j \in [n'_j]$ as

$$(A \boxtimes_j B)[i_1, \ldots, (i_j - 1)n'_j + i'_j, \ldots, i_k] = A[i_1, \cdots, i_j, \cdots i_k]B[i_1, \ldots, i'_j, \ldots, i_k].$$

1.3.2 Algorithms for path signatures

In this section, we provide adapted algorithms for computing signature features and kernels.

First, we provide an algorithm for computing the order-p signature features given in (1.9) in Algorithm 1. This algorithm has not appeared explicitly in previous work, and is a combination of Algorithms 5 and 6 in [146]. The output of the algorithm are flattened signature tensors of dimension $d, d^2, \ldots, d^M$. Upon investigation, the time complexity of the algorithm is $O(MNLpd^M)$ and requires a space of $O(p^2NLD + Nd^M)$. This is clearly inefficient due to the polynomail complexity in d and exponential in M, and in the later Chapters 3 and 5 we discuss how this complexity can be reduced via various low-rank and random projection techniques.

Next, we adapt the dual algorithm for computing the order-p signature kernel (1.10) from [146, Alg. 6] under Algorithm 2. The output of the algorithm consists of signature kernel matrices per level $K_1, \ldots, K_m \in \mathbb{R}^{N \times N}$. The time complexity of the algorithm is $O(N^2L^2d + MN^2L^2p^2)$, while the space is $O(p^2N^2L^2)$, where d is the complexity of static kernel evaluation. Due to joint quadratic complexity in N^2 and L^2, this is infeasible for high-volume (N large) and long time series (L large). In Chapter 2, we combine the signature kernel with Gaussian processes and variational inference techniques reducing the complexity to linear in N, allowing to scale to large datasets, although the length still remains a bottleneck. Later on, in Chapter 5, we introduce scalable random approximations to the signature kernel, which alleviate the quadratic complexity in both N and L, while also scaling favourably in the "feature size" i.e. in the dimensionality of the random feature map that represents the random approximate kernel.

Algorithm 1 Computing order-p signature features (1.9).

Input: Time series $X = (\mathbf{x}_1, \dots, \mathbf{x}_N) \subset \mathsf{Seq}(\mathbb{R}^d)$, truncation-$M \in \mathbb{Z}_+$, order-$p \in \mathbb{Z}_+$
Optional: Add time-parameterization $\mathbf{x}_i \leftarrow (\mathbf{x}_{i,t}, t/\mathsf{L}_{\mathbf{x}_i})_{t=1}^{\mathsf{L}_{\mathbf{x}_i}}$ for all $i \in [N]$
Tabulate to uniform length $L = \max_{j\in[N]} \mathsf{L}_{\mathbf{x}_j}$ by $\mathbf{x}_i \leftarrow (\mathbf{x}_{i,1}, \dots, \mathbf{x}_{i,\mathsf{L}_{\mathbf{x}_i}}, \dots, \mathbf{x}_{i,\mathsf{L}_{\mathbf{x}_i}})$ for all $i \in [N]$
Initialize an array U with shape $[N, L, d]$
Fill U with values $U[i, l, :] = \mathbf{x}_{i,l} - \mathbf{x}_{i,l-1}$ for all $i \in [N]$ and $l \in [L]$
Accumulate into level-1 features $P_1 \leftarrow U[:, \Sigma, :]$
Initialize an array R of shape $[1, N, L, d]$
Assign values $R[1, :, :, :] \leftarrow U$
for $m = 2$ **to** M **do**
    Let $q \leftarrow \min(p, m)$
    Collapse into expanding window features $P' \leftarrow R[\Sigma, :, \boxplus, :]$
    Update with next increment $P' \leftarrow P'[:, +1, :] \boxtimes_{-1} U$
    Initialize new array R' of shape $[q, N, L, d]$
    Assign values $R'[1, :, :, :] \leftarrow P'$
    for $r = 2$ to q **do**
        Update with current increment $R'[r, :, :, :] \leftarrow \frac{1}{r} R[r-1, :, :, :] \boxtimes_{-1} U$
    end for
    Aggregate into level-m features $P_m \leftarrow R'[\Sigma, :, \Sigma, :]$
    Roll array $R \leftarrow R'$
end for
Output: Arrays of signature features per level $P_1 \in \mathbb{R}^d, \dots, P_M$.

Algorithm 2 Computing order-p signature kernels (1.10).

Input: Sequences $\mathbf{X} = (\mathbf{x}_i)_{i=1,\dots,N} \subset \mathsf{Seq}(\mathcal{X})$, truncation-$M \in \mathbb{Z}_+$, order-$p \in \mathbb{Z}_+$, static kernel $\mathsf{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$
Optional: Add time-parameterization $\mathbf{x}_i \leftarrow (\mathbf{x}_{i,t}, t/\mathsf{L}_{\mathbf{x}_i})_{t=1}^{\mathsf{L}_{\mathbf{x}_i}}$ for all $i \in [N]$
Tabulate to uniform length $L = \max_{j\in[N]} \mathsf{L}_{\mathbf{x}_j}$ by $\mathbf{x}_i \leftarrow (\mathbf{x}_{i,1}, \dots, \mathbf{x}_{i,\mathsf{L}_{\mathbf{x}_i}}, \dots, \mathbf{x}_{i,\mathsf{L}_{\mathbf{x}_i}})$ for all $i \in [N]$
Initialize an array K with shape $[N, N, L, L]$
Compute kernel increments $K[i, j, k, l] = \delta_{k,l} \mathsf{k}(\mathbf{x}_{i,k}, \mathbf{x}_{j,k})$ for $i, j \in [N]$ and $k, l \in [L]$
Collapse into level-1 kernel entries $K_1 \leftarrow K[:, :, \Sigma, \Sigma]$
Initialize an array R of shape $[1, 1, N, N, L, L]$
Assign values $R[1, 1, :, :, :, :] \leftarrow K$
for $m = 2$ **to** M **do**
    Let $q \leftarrow \min(p, m)$
    $K' \leftarrow R[\Sigma, \Sigma, :, :, \boxplus + 1, \boxplus + 1] \odot K$
    Initialize new array R' of shape $[q, q, N, N, L, L]$
    Assign values $R'[1, 1, :, :, :, :] \leftarrow K'$
    for $r = 2$ **to** q **do**
        $R'[r, 1, :, :, :, :] \leftarrow \frac{1}{r} R[r-1, \Sigma, :, :, :, \boxplus + 1] \odot K$
        $R'[1, r, :, :, :, :] \leftarrow \frac{1}{r} R[\Sigma, r-1, :, :, \boxplus + 1, :] \odot K$
        for $s = 2$ **to** q **do**
            $R'[r, s, :, :, :, :] \leftarrow \frac{1}{rs} R[r-1, s-1, :, :, :, :] \odot K$
        end for
    end for
    Collapse into level-m kernel entries $K_m \leftarrow R'[\Sigma, \Sigma, :, :, \Sigma, \Sigma]$
    Roll array $R \leftarrow R'$
end for
Output: Signature kernel matrices per level $K_1, \dots, K_M$.

Chapter 2

Gaussian Processes with Signature Covariances

2.1 Introduction

The evolution of some state variable, parameter or object gives naturally gives rise to sequential data, which is defined by having a notion of order on the incoming information. The ordering relation, or index set does not have to represent physical time, but for simplicity we will call it as such. When the index-set is discrete, we refer to the data as a sequence, and when continuous as a path. For example, besides time series (TS), sources of sequential data are text [205], DNA [114], or even topological data analysis [43]. This ubiquity of sequential data has received special attention by the machine learning community in recent years.

In this chapter, we are focused on combining signature kernels with these approaches:

Bayesian approaches. Often not only point predictions, but estimates of the associated uncertainties are required [94]. Gaussian processes (GP) [216] provide flexible priors over functions of the data in nonparametric Bayesian models. In the context of sequential data, two prominent ways to use GPs are: (1) using as covariance functions kernels specifically designed for sequences [176, 58, 59, 5], (2) modelling the evolution in a latent space, that emits the observations, as a discrete dynamical system with a GP prior on the transition function, a model called the Gaussian Process State Space Model (GPSSM) [85, 84, 187, 78, 70, 127]. These two approaches are not mutually exclusive; if one models the latent system as a higher order Markov process, then sequence kernels can incorporate the effect of past states.

Deep learning approaches. Deep learning approaches, such as the celebrated LSTM network [120], other forms of RNNs [45], convolutional networks, and transformers [275] have successfully been applied to a variety of tasks involving sequential data [254, 201, 285]. Deep learning models can approximate any continuous function, but the cost is a large number of parameters, high variance and poor interpretability. This leaves the door open for alternative approaches not only as competitors, but as complementary building blocks in larger models.

Contribution. In principle, one can just use the signature kernel and algorithms as covariance function to define a GP for sequential data. However, the computational complexity becomes quickly prohibitive, which ultimately does not lead to scalable performance on many time series benchmarks. We therefore develop a different approach to signature covariances that builds on two recent advances in GP inference: variational inference [259, 115, 186] and inter-domain inducing points [157] to alleviate the computational burden. In particular, we show that one can use low-rank tensors as inter-domain inducing points by optimizing a variational bound. Moreover, we use this GP as a building block in combination with RNNs to build models that combine the strenghts of signatures, Bayesian models, and deep learning architectures. This results in scalable inference algorithms and we use this to benchmark on standard TS datasets (i) against popular non-Bayesian time series classifiers purely in terms of accuracy, (ii) against alternative Bayesian models by comparing the calibration of uncertainties for predictions. Code and benchmarks are publically available at `http://github.com/tgcsaba/GPSig`.

2.2 Background and Notation

Given data $(\mathbf{X}, Y)$ consisting of $n_{\mathbf{X}}$ inputs $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_{n_{\mathbf{X}}}) \subset \mathcal{X}$ with labels $Y = (y_1, \ldots, y_{n_{\mathbf{X}}}) \subset \mathbb{R}$, the Bayesian approach is to put a prior on a set of functions $\{f : f : \mathcal{X} \to \mathbb{R}\}$, update this prior by conditioning on $(\mathbf{X}, Y)$, and then use the resulting posterior to make inference about the label $y_\star$ of an unseen point $\mathbf{x}_\star$. When this is done with Gaussians, the central object is a GP $f = (f_{\mathbf{x}})_{\mathbf{x} \in \mathcal{X}}$ which is specified by the mean and covariance functions. Below we recall how covariances can be constructed from polynomial features, which we then extend to signatures.

2.2.1 The feature space view

Given a map $\varphi : \mathcal{X} \hookrightarrow V$ that injects $\mathcal{X}$ into a Hilbert space $\mathcal{H}$, a natural way to put a prior on a function class $\mathcal{X} \to \mathbb{R}$ is to consider linear functions of φ as model, that is $f_{\mathbf{x}} = \langle \ell, \varphi(\mathbf{x}) \rangle$ to model $f(\mathbf{x}_i) \approx y_i$ for some "weights" $\ell \in \mathcal{H}$. Uncertainty about f is then specified by uncertainty about ℓ (and the hyperparameters of φ). We refer to φ as a feature map and to $\mathcal{H}$ as feature space.

The advantage of the nonparametric approach is that it allows one to avoid explicitly computing features and weights, and to perform inference directly on the function f. Throughout we assume that $f = (f_{\mathbf{x}})_{\mathbf{x}\in\mathcal{X}}$ is a centered GP. Then, predictions about unseen points can then be made by Bayesian inference using Gaussian conditioning. If the task is classification where the labels Y are discrete, such an approach can be still applied by using a GP $f = (f_{\mathbf{x}})_{\mathbf{x}\in\mathcal{X}}$ as nuiscance function to put a prior on the class membership probability by specifying $p(y = 1|\mathbf{x}) = \sigma(f_{\mathbf{x}})$, where σ is for example a sigmoid.

2.2.2 Polynomial features

First, we review the classical case of how a Gaussian process can be constructed from polynomial features, which we will later on generalize to signatures.

Now, let $\mathcal{X} \subseteq \mathbb{R}^d$ and define the monomial feature map

$$\varphi(\mathbf{x}) = (1, \mathbf{x}, \mathbf{x}^{\otimes 2}, \mathbf{x}^{\otimes 3}, \dots, \mathbf{x}^{\otimes M}) \tag{2.1}$$

where $\mathbf{x}^{\otimes m} \in \left(\mathbb{R}^d\right)^{\otimes m}$ is a tensor. We refer to Section 1.1 about tensors. If we set $f_{\mathbf{x}} = \langle \ell, \varphi(\mathbf{x})\rangle$ and put a centered Gaussian prior on $\ell = (\ell_1, \dots, \ell_M)$, then it follows from Section 1.1 that

$$\begin{aligned}
\mathbb{E}\left[f_{\mathbf{x}} f_{\mathbf{y}}\right] &= 1 + \sum_{m=1}^{M} \mathbb{E}\left[\left\langle \ell \otimes \ell, \mathbf{x}^{\otimes m} \otimes \mathbf{y}^{\otimes m}\right\rangle\right] \\
&= 1 + \sum_{m=1}^{M} \left\langle \mathbb{E}\left[\ell \otimes \ell\right], \mathbf{x}^{\otimes m} \otimes \mathbf{y}^{\otimes m}\right\rangle \\
&= 1 + \sum_{m=1}^{M} \langle \Sigma_m^2, \mathbf{x}^{\otimes m} \otimes \mathbf{y}^{\otimes m}\rangle
\end{aligned}$$

where $\Sigma_m^2 = \mathbb{E}[\ell_m \otimes \ell_m] \in \left(\mathbb{R}^d\right)^{\otimes(2m)}$. Taking Σ_m^2 to be an isotropic "diagonal" tensor $\Sigma_m^2 = \sigma_m^2 \sum_{i_1,\dots,i_m=1}^{m} (e_{i_1} \otimes \cdots \otimes e_{i_m})^{\otimes 2}$ recovers the polynomial kernel,

$$\mathbb{E}[f_{\mathbf{x}} f_{\mathbf{y}}] = \sum_{m=0}^{M} \sigma_m^2 \langle \mathbf{x}, \mathbf{y}\rangle^m. \tag{2.2}$$

Many other variations exist, for example other classes of polynomials, such as Hermite polynomials (the eigenfunctions of the classic RBF kernel), can increase the effectiveness, since they allow to make the associated feature expansion infinite dimensional. However, what makes any such class of polynomials a sensible choice for φ is that by the Stone–Weierstrass theorem, any

continuous compactly supported function $\mathcal{X} \to \mathbb{R}$ can be arbitrary well approximated as linear functions of $\varphi(\mathbf{x})$. This approximation property is often called universality [189, 245].

2.3 From signature features to covariances

The signature feature map S can be seen as a generalization of the polynomial feature map φ as defined in (2.1) from the domain $\mathcal{X} \subseteq \mathbb{R}^d$ of vectors to the domain of bounded variation paths.

Recall from Section 1.2 that it is defined for a $(\mathbf{x}_t)_{t\in[0,T]} \in \mathsf{Paths}(\mathcal{X})$ as

$$\mathsf{S}(\mathbf{x}) = \left(\int_{\mathbf{t}\in\Delta_m([0,T])} \mathrm{d}\mathbf{x}_{t_1} \otimes \cdots \otimes \mathrm{d}\mathbf{x}_{t_m} \right)_m .$$

2.3.1 Automatic parameterization invariance determination

A classic empirical finding that led to dynamic time warping (DTW) is that functions of sequences are often invariant to a certain degree of time parameterization: for example, different speakers pronounce words at different speeds. However, sometimes the parameterization matters, e.g. for financial data. Thus we do not only care about

$$\mathcal{H}_{\mathsf{Paths}} = \{f : \mathsf{Paths}(\mathcal{X}) \to \mathbb{R} : \text{continuous and compactly supported}\} \tag{2.3}$$

but also about the subset of it that consist of parameterization invariant functions. To make this precise, we call $(\mathbf{x}_s)_{s\in[0,S]}$ a reparameterization of $(\mathbf{y}_t)_{t\in[0,T]}$ if there exists a a smooth increasing function $\rho : [0,S] \to [0,T]$ (the "time change") such that $\mathbf{x}_s = y_{\rho(s)}$ for all $s \in [0,S]$.

Often the function we want to learn is invariant to some but not extreme reparameterization, so we need a more nuanced way to quantify parameterization (in)variance. Hence, what we really want is a hyperparameter $\tau \geq 0$ that signifies the degree of parameterization invariance: for $\tau = 0$ all the prior mass should concentrate on the "extreme case" that is the subset of (2.3) consisting parameterization invariant functions; and as τ gets increased the probability mass should spread out to parameterization sensitive functions. This would allow to infer the degree of parameterization invariance by automatic relevance determination (ARD).

To accomplish this, we augment our paths with a hyperparameter $\tau \geq 0$ such that

$$(\mathbf{x}_t^{(\tau)})_{t\in[0,T]} = (\tau \cdot t, \mathbf{x}_t))_{t\in[0,T]} \in \mathsf{Paths}(\hat{\mathcal{X}}),$$

where $(\mathbf{x}_t)_{t\in[0,T]} \in \mathsf{Paths}(\mathcal{X})$ and $\hat{\mathcal{X}} \subseteq \mathbb{R}^{1+d}$. This simply makes the parameterization part of the trajectory by adding an extra coordinate, and allows the magnitude of τ to scale the effect

	VECTORS	PATHS	SEQUENCES
DOMAIN	$\mathcal{X}$	$\mathsf{Paths}(\mathcal{X})$	$\mathsf{Seq}(\mathcal{X})$
FEATURES	$\varphi(\mathbf{x}) = \left(\mathbf{x}^{\otimes m}\right)_m$	$\mathsf{S}(\mathbf{x}) = \left(\int_{\mathbf{t}\in\Delta_m([0,\mathsf{T}_\mathbf{x}])} \mathrm{d}\mathbf{x}_{t_1} \otimes \cdots \otimes \mathrm{d}\mathbf{x}_{t_m}\right)_m$	$\mathsf{S}^{(1)}(\mathbf{x}) = \left(\sum_{\mathbf{i}\in\Delta_m(\mathsf{L}_\mathbf{x})} \delta\mathbf{x}_{i_1} \otimes \cdots \delta\mathbf{x}_{i_m}\right)_m$
FEATURE SPACE	$\prod_m \left(\mathbb{R}^d\right)^{\otimes m}$	$\prod_m \left(\mathbb{R}^d\right)^{\otimes m}$	$\prod_m \left(\mathbb{R}^d\right)^{\otimes m}$
FUNCTIONS	$f : \mathbb{R}^d \to \mathbb{R}$	$f : \mathsf{Paths}(\mathcal{X}) \to \mathbb{R}$	$f : \mathsf{Seq}(\mathcal{X}) \to \mathbb{R}$
COVARIANCE	$\sum_m \sigma_m^2 \langle \mathbf{x}, \mathbf{y}\rangle^m$	$\sum_m \sigma_m^2 \int_{\substack{\mathbf{s}\in\Delta_m([0,\mathsf{T}_\mathbf{x}]) \\ \mathbf{t}\in\Delta_m([0,\mathsf{T}_\mathbf{y}])}} \prod_{k=1}^m \left\langle d\mathbf{x}_{s_m}, d\mathbf{y}_{t_m}\right\rangle$	$\sum_m \sigma_m^2 \sum_{\substack{\mathbf{i}\in\Delta_m(\mathsf{L}_\mathbf{x}) \\ \mathbf{j}\in\Delta_m(\mathsf{L}_\mathbf{y})}} \prod_{k=1}^m \left\langle \delta\mathbf{x}_{i_k}, \delta\mathbf{y}_{j_k}\right\rangle$

Table 2.1: Comparison of polynomial and signature features

of parameterization sensitivity. Since signatures distinguish different trajectories (but not the speed at which we run through them), it follows that for $\tau > 0$ and $\mathbf{x}, \mathbf{y} \in \mathsf{Paths}(\mathcal{X})$

$$\mathsf{S}(\mathbf{x}^{(\tau)}) = \Phi(\mathbf{y}^{(\tau)}) \text{ if and only if } \mathbf{x} = \mathbf{y},$$

while for $\tau = 0$ we have,

$$\mathsf{S}(\mathbf{x}^{(0)}) = \mathsf{S}(\mathbf{y}^{(0)}) \text{ if and only if } \mathbf{x} \sim \mathbf{y}$$

since the extra coordinate is "switched off". Here, $\sim$ denotes tree-like equivalence, but we invite the reader to read $\mathbf{x} \sim \mathbf{y}$ as saying that $\mathbf{x}$ is a reparameterization of $\mathbf{y}$. This is strictly speaking not true and the precise mathematical statement is given in Section 1.2.2, but note that for real-world data, tree-like equivalence is synonymous with reparameterization.

In the rest of the chapter, we omit the dependence on τ from the notation, but it should be noted that we are still working with the parameterization augmented path $\mathbf{x}^{(\tau)}$ in place of $\mathbf{x}$.

2.3.2 Signature covariances

In this section, we focus on deriving the signature covariances with paths lifted by the linear kernel for simplicity, but the construction extends to other variations, e.g. the RBF kernel, by replacing the coordinate-wise expansion of paths with the eigenfunctions of the kernel, which forms an orthonormal basis of the RKHS. The formulation gets more technical since in this case the linear functional ℓ becomes infinite dimensional, and equation (2.4) becomes the Karhunen-Loeve expansion for GPs [3]. Following the feature space view, we now argue in complete analogy to the case of the classical polynomial feature map φ for $\mathcal{X} = \mathbb{R}^d$, and define a centered GP $f = (f_\mathbf{x})_{\mathbf{x}\in\mathsf{Paths}(\mathcal{X})}$ by putting a centered Gaussian prior on $\ell = (\ell_1, \ldots, \ell_M)$ and setting

$$f_\mathbf{x} = \langle \ell, \mathsf{S}(\mathbf{x})\rangle \quad \text{for } \mathbf{x} \in \mathsf{Paths}(\mathcal{X}). \tag{2.4}$$

The GP is hence fully specified by its covariance function

$$\mathsf{k}_{\mathsf{S}}(\mathbf{x},\mathbf{y}) = \sum_{m=0}^{M} \left\langle \Sigma_m^2, \int_{\mathbf{s}\in\Delta_m([0,\mathsf{T}_\mathbf{x}])} \mathrm{d}\mathbf{x}_{s_1} \otimes \cdots \otimes \mathrm{d}\mathbf{x}_{s_m} \otimes \int_{\mathbf{t}\in\Delta_m([0,T])} \mathrm{d}\mathbf{y}_{t_1} \otimes \cdots \otimes \mathrm{d}\mathbf{y}_{t_m} \right\rangle_m \tag{2.5}$$

that has $(\tau, M, \Sigma_1^2, \ldots, \Sigma_M^2)$ as hyperparameters. In particular, choosing an isotropic covariance structure for Σ_m^2 completely analogously to how it is done above equation (2.2) gives that

$$\mathsf{k}_{\mathsf{S}}(\mathbf{x},\mathbf{y}) = \sum_{m=0}^{M} \sigma_m^2 \int_{\substack{\mathbf{s}\in\Delta_m([0,\mathsf{T}_\mathbf{x}])\\ \mathbf{t}\in\Delta_m([0,\mathsf{T}_\mathbf{y}])}} \left\langle \mathrm{d}\mathbf{x}_{s_1}, \mathrm{d}\mathbf{y}_{t_1} \right\rangle \cdots \left\langle \mathrm{d}\mathbf{x}_{s_m}, \mathrm{d}\mathbf{y}_{t_m} \right\rangle, \tag{2.6}$$

Finally, as we are given data not as paths, but as discrete-time observations $\mathbf{x},\mathbf{y} \in \mathsf{Seq}(\mathcal{X})$, in practice we use the order-1 discretized signature kernel (see Section 1.2.6) given as

$$\mathsf{k}_{\mathsf{S}}^{(1)}(\mathbf{x},\mathbf{y}) = \sum_{m=0}^{M} \sigma_m^2 \sum_{\substack{\mathbf{i}\in\Delta_m(\mathsf{L}_\mathbf{x})\\ \mathbf{j}\in\Delta_m(\mathsf{L}_\mathbf{y})}} \left\langle \delta\mathbf{x}_{i_1}, \delta\mathbf{y}_{j_1} \right\rangle \cdots \left\langle \delta\mathbf{x}_{i_m}, \delta\mathbf{y}_{j_m} \right\rangle, \tag{2.7}$$

and in forth going we suppress the dependence on the order-$p = 1$ hyperparameter.

2.3.3 Regularity of the resulting Gaussian process

Next, we want to find out the regularity properties of the GP with covariance (2.6). However, the index set $\mathsf{Paths}(\mathcal{X})$ is a very large space so some care is needed. In Appendix 2.A, we compute covering numbers that yield explicit bounds on the modulus of continuity in terms of $\|\mathbf{x}\|_{1\text{-var}}$.

Theorem 2.1. *Let $L > 0$ and $\mathsf{Paths}_L(\mathcal{X}) = \{\mathbf{x} \in \mathsf{Paths}(\mathcal{X}) : \|\mathbf{x}\|_{1\text{-var}} \leq L\}$. There exists a centered* GP *$f = (f_\mathbf{x})_{\mathbf{x}\in\mathsf{Paths}_L(\mathcal{X})}$ with $\mathsf{k}_{\mathsf{S}}(\mathbf{x},\mathbf{y})$ as defined in* (2.6) *as covariance function and that has continuous sample paths $\mathbf{x} \mapsto f_\mathbf{x}$. Further, an explicit bound on its modulus of continuity in terms of L is given in equation* (2.12).

We now have a well-defined GP for Bayesian inference for sequences at hand that inherits many of the attractive properties of signature features. To turn this into useful models for large TS benchmarks we develop efficient inference algorithms in the next section.

2.4 Sparse variational inducing tensors

To reiterate, we are given data $(\mathbf{X}, Y)$ consisting of $n_{\mathbf{X}}$ sequences $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_{n_{\mathbf{X}}}) \subset \mathsf{Seq}(\mathcal{X})$ of maximal length $\mathsf{L}_{\mathbf{X}} = \max_{\mathbf{x}\in\mathbf{X}} \mathsf{L}_{\mathbf{x}}$ that evolve in $\mathbb{R}^d$ with labels $Y = (y_1, \dots, y_{n_{\mathbf{X}}})$, and the task is to predict labels $y_\star$ of unseen points $\mathbf{x}_\star$. For sequences, the sample size $n_{\mathbf{X}}$ and associated covariance matrix inversion is not the only compational bottleneck but also the maximal length $\mathsf{L}_{\mathbf{X}}$, and the dimension d of the state space matter: $n_{\mathbf{X}}$, $\mathsf{L}_{\mathbf{X}}$ and d can be simultaneously large.

In this section, we introduce a sparse variational inference scheme to approximate the posterior, that locates the inducing points in a space other than the data-domain; this approach is usually called *inter-domain* sparse variational inference [157, 186]. This allows for more efficient data-representation and faster inference. Key to our approach is that signature features take values in a well-understood subset of the feature space $\mathsf{T}(\mathbb{R}^d)$. This allows as us to augment the index set with structured tensors, and locate inducing points in this larger set.

2.4.1 Variational inference

As is well-known, inference for GPs scales as $O(n_{\mathbf{X}}^3)$, see Section 3.3. in [216]. This first led to sparse models, [208], that select a subset $\mathbf{Z} = \{\mathbf{z}_1, \dots, \mathbf{z}_{n_{\mathbf{Z}}}\}$ of $\mathbf{X}$ consisting of $n_{\mathbf{Z}} \ll n_{\mathbf{X}}$ points, and subsequently to pseudo-inputs, [241], that select points $\mathbf{Z}$ that are not necessarily in $\mathbf{X}$. This was a big step towards complexity reduction, but pseudo-inputs are prone to overfitting, [185]. A different idea is to treat $\mathbf{Z}$ as parameters of a variational approximation [259] and not as model parameters; that is the points $\mathbf{Z}$ are choosen simultaneously with the hyperparameters of the GP by maximising a lower bound on the log-marginal likelihood $\log p(Y)$, the so-called evidence lower bound (ELBO), given as

$$\log p(Y) \geq \mathbb{E}_{q(f_{\mathbf{x}})}[\log p(Y|f_{\mathbf{x}})] - \mathsf{D}_{\mathsf{KL}}\left[q(f_{\mathbf{Z}}) \,\|\, p(f_{\mathbf{Z}})\right], \tag{2.8}$$

where $f_{\mathbf{x}}$ and $f_{\mathbf{Z}}$ denotes the GP evaluated at the data-points and the inducing locations. Typically, $q(f_{\mathbf{Z}})$ is given a free-form multivariate Gaussian to be learnt from the data, and then extended to other indices of the GP by *prior conditional matching*, i.e. $q(f_{\mathbf{x}}|f_{\mathbf{Z}}) = p(f_{\mathbf{x}}|f_{\mathbf{Z}})$. Initially applied to regression, this was extended to classification [34, 115]. Among its advantages are that it gives a nonparametric approximation to the true posterior, adding inducing points only improves the approximation, and any optimization method can be used to maximize the ELBO, most importantly, stochastic optimization; see [117, 16, 29].

Inter-domain approaches. Another idea is to go beyond the original index set and place inducing points $\mathbf{Z}$ in a different space $\mathcal{X}'$, that is, given a centered GP $g = (g_{\mathbf{x}})_{\mathbf{x}\in\mathcal{X}}$ one augments

the original index set $\mathcal{X}$ by a set $\mathcal{X}'$ to define a new GP $(g_{\mathbf{x}})_{\mathbf{x}\in\mathcal{X}\cup\mathcal{X}'}$ and then locates the inducing points in this bigger model. This was suggested in [157] in the context of integral transforms, which was extended in [116], and studied in more generality in [186]. In general, it is not obvious how to find a useful augmentation set $\mathcal{X}'$ and define the covariance enlarged to $\mathcal{X}\cup\mathcal{X}'$.

2.4.2 Inter-domain low-rank tensors

A feature space augmentation. Given any GP with a covariance function $\mathsf{k}(\mathbf{x},\mathbf{y}) = \langle\Phi(\mathbf{x}),\Phi(\mathbf{y})\rangle$ where Φ is explicitly known[1], we propose that a natural augmentation candidate is the "feature space" $\mathcal{X}' = \operatorname{span}\{\Phi(\mathbf{x}) : \mathbf{x}\in\mathcal{X}\}$ itself. The covariance function k of g can be simply extended to $\mathcal{X}\cup\mathcal{X}'$ by linearity,

$$\mathsf{k}(\mathbf{x},\mathbf{z}) = \mathsf{k}(\mathbf{z},\mathbf{x}) = \alpha\mathsf{k}(\mathbf{x},\mathbf{x}') + \beta\mathsf{k}(\mathbf{x},\mathbf{x}'') \tag{2.9}$$

for $\mathbf{x}\in\mathcal{X}$, $\mathbf{z} = \alpha\Phi(\mathbf{x}') + \beta\Phi(\mathbf{x}'') \in \mathcal{X}'$, $\alpha,\beta\in\mathbb{R}$; analogous for $\mathsf{k}(\mathbf{z},\mathbf{z}')$ with $\mathbf{z},\mathbf{z}'\in\mathcal{X}'$. For our GP,

$$\mathcal{X}' = \operatorname{span}\{\Phi_\tau(\mathbf{x}) : \mathbf{x}\in\mathsf{Paths}(\mathcal{X})\} \subset \mathsf{T}(\mathbb{R}^d).$$

We can thus extend our signature covariance (2.5) to $\mathsf{Seq}(\mathcal{X})\cup\mathsf{T}(\mathbb{R}^d)$ by (2.9). This provides a flexible class of inducing point locations $\mathbf{Z}$ by optimizing over elements of the tensor algebra $\mathbf{Z}\subset\mathsf{T}(\mathbb{R}^d)$. We coin these inducing point locations as *inducing tensors*.

Consistency of augmentation. A subtle point about augmenting the index set is that maximizing the ELBO in (2.8) is not necessarily equivalent anymore to minimizing a rigorously defined KL divergence between the true posterior process and its approximation over the unaugmented index set. In [186], a sufficient condition given for this to hold is that the prior GP evaluated at the newly added indices is deterministic conditioned on the original GP. In the case of (2.9), this is easily seen to be true, since the augmented indices arise as linear combinations of (the lifts of) elements in the original index set. Therefore, the corresponding GP evaluations arise as linear combinations of evaluations of the original process by the fact that the (pre-)feature space $\mathcal{X}'$ is a *representation* of the *Hilbert space generated by the process* [21, Sec. 3].

[1] Mercer's Theorem guarantees the existence of Φ, but not in a sufficiently explicit form.

Representation of inducing tensors. We define our inducing tensors as in the low-rank format, specifically as rank-1 tensors such each inducing point is represented as

$$\mathbf{z} = (\mathbf{z}_m)_{m=0,\dots,M} \in \prod_{m=0}^{M} (\mathbb{R}^d)^{\otimes m},$$

where $z_0 \in \mathbb{R}$ and $\mathbf{z}_m = \mathbf{v}_{m,1} \otimes \mathbf{v}_{m,2} \otimes \cdots \otimes \mathbf{v}_{m,m}$ for $m \geq 1$. We remark that this construction does not generally give tensors that can be signatures of paths. However, they can be represented as linear combinations of signatures, since path signatures span the whole tensor algebra, hence the previous argument about the augmentation carries over. Also, informally, what gives the data-efficiency of inducing tensors is exactly that they are not represented in a basis of signatures.

By linearity of integration and the inner product, the inducing point covariance equals

$$\mathbb{E}[f_{\mathbf{z}} f_{\mathbf{z}'}] = \sum_{m=0}^{M} \sigma_m^2 \langle \mathbf{z}_m, \mathbf{z}'_m \rangle = \sum_{m=0}^{M} \sigma_m^2 \prod_{k=1}^{m} \langle \mathbf{v}_{m,k}, \mathbf{v}'_{m,k} \rangle,$$

and the cross-covariance between paths and inducing tensors equals

$$\mathbb{E}[f_{\mathbf{x}} f_{\mathbf{z}}] = \sum_{m=0}^{M} \sigma_m^2 \langle \mathsf{S}_m(\mathbf{x}), z_m \rangle = \sum_{m=0}^{M} \int_{\mathbf{t} \in \Delta_m([0,\mathsf{T}_\mathbf{x}])} \langle \mathrm{d}\mathbf{x}_{t_1}, \mathbf{v}_{m,1} \rangle \cdots \langle \mathrm{d}\mathbf{x}_{t_m}, \mathbf{v}_{m,m} \rangle,$$

while when working with discrete-time data, using the order-1 signature (2.7) it equals

$$\mathbb{E}[f_{\mathbf{x}} f_{\mathbf{z}}] = \sum_{\mathbf{i} \in \Delta_m(\mathsf{L}_\mathbf{x})} \langle \delta \mathbf{x}_{i_1}, \mathbf{v}_{m,1} \rangle \cdots \langle \delta \mathbf{x}_{i_m}, \mathbf{v}_{m,m} \rangle. \tag{2.10}$$

Below we use the approximation (2.10) since it makes the recursive algorithms simpler but note that a simple modification exactly computes (2.10) for a marginal computational overhead.

2.4.3 Algorithms.

We need to compute the three covariance matrices: (1) $K_{\mathbf{ZZ}}$ of inducing tensors $\mathbf{Z}$ and inducing tensors $\mathbf{Z}$, (2) $K_{\mathbf{ZX}}$ of inducing tensors $\mathbf{Z}$ and sequences $\mathbf{X}$, (3) $K_{\mathbf{XX}}$ of sequences $\mathbf{X}$ and sequences $\mathbf{X}$. Using the above tensor representations allows to give vectorized algorithms for (1) and (2) in Algorithms 3 and 4, respectively. For (3) we use a variation of Algorithm 2 which can be found in [266, App. D.2]. We use notation defined in Section 1.3.1. For $v, v' \in V$, d denotes the time to compute $\langle v, v' \rangle$, c the memory requirement of v.

Algorithm 3 Computing the inducing covariances $K_{\mathbf{ZZ}}$

Input: Tensors $\mathbf{Z} = (\mathbf{z}_i)_{i=1,\dots,n_\mathbf{Z}} \subset \prod_{n=0}^{m} V^{\otimes n}$, scalars $(\sigma_0^2, \sigma_1^2, \dots, \sigma_m^2)$, depth $m \in \mathbb{N}$
Compute $K[i,j,n,k] \leftarrow \langle v_{n,k}^i, v_{n,k}^j \rangle$ for $i,j \in [n_\mathbf{Z}]$, $n \in [m]$ and $k \in [n]$
Initialize $R[i,j] \leftarrow \sigma_0^2$ for $i,j \in [n_\mathbf{Z}]$
for $n = 1$ **to** m **do**
    Assign $A \leftarrow K[:,:,n,1]$
    for $k = 2$ **to** n **do**
        Iterate $A \leftarrow K[:,:,n,k] \odot A$
    end for
    Update $R \leftarrow R + \sigma_n^2 \cdot A$
end for
Output: Matrix of inducing covariances $K_{\mathbf{ZZ}} \leftarrow R$

Algorithm 4 Computing the cross-covariances $K_{\mathbf{ZX}}$

Input: Tensors $\mathbf{Z} = (\mathbf{z}_i)_{i=1,\dots,n_\mathbf{Z}} \subset \prod_{n=0}^{m} V^{\otimes n}$,
sequences $\mathbf{X} = (\mathbf{x}_i)_{i=1,\dots,n_\mathbf{X}} \subset \mathsf{Seq}(\mathcal{X})$, scalars $(\sigma_0^2, \sigma_1^2, \dots, \sigma_m^2)$, depth $M \in \mathbb{N}$
Compute $K[i,j,l,m,k] \leftarrow \langle \mathbf{v}_{m,k}^i, \delta\mathbf{x}_{j,t_l} \rangle$ for $i \in [n_\mathbf{Z}]$, $j \in [n_\mathbf{X}]$, $l \in [L_\mathbf{X}]$, $m \in [M]$ and $k \in [m]$
Initialize $R[i,j] \leftarrow \sigma_0^2$ for $i \in [n_\mathbf{Z}]$, $j \in [n_\mathbf{X}]$
for $m = 1$ **to** M **do**
    Assign $A \leftarrow K[:,:,n,1]$
    for $k = 2$ **to** m **do**
        Iterate $A \leftarrow K[:,:,:,m,k] \odot A[:,:,\boxplus + 1]$
    end for
    Update $R \leftarrow R + \sigma_m^2 \cdot A[:,:,\Sigma]$
end for
Output: Matrix of cross-covariances $K_{\mathbf{ZX}} \leftarrow R$

Proposition 2.2. *Algorithm 3 computes the covariance matrix $K_{\mathbf{ZZ}}$ of $n_\mathbf{Z}$ inducing points in $O(M^2 \cdot n_\mathbf{Z}^2 \cdot d)$ steps. Algorithm 4 computes the cross-covariance matrix $K_{\mathbf{ZX}}$ in $O(M^2 \cdot n_\mathbf{X} \cdot n_\mathbf{Z} \cdot L_\mathbf{X} \cdot d)$ steps. Additionally to storing the inducing tensors $\mathbf{Z}$, Algorithm 3 requires $O(M^2 \cdot n_\mathbf{Z}^2)$ memory, Algorithm 4 requires $O(M^2 \cdot n_\mathbf{X} \cdot n_\mathbf{Z} \cdot L_\mathbf{X})$ memory.*

Proposition 2.2 follows details the scalability and we emphasize: (i) Both algorithms are linear in the maximal sequence length $L_\mathbf{X}$. (ii) M is a hyperparameter, and in all our experiments we learnt from the data $M \leq 5$, thus the quadratic complexity in M is negligible. (iii) The memory cost of inducing tensors $\mathbf{Z}$ is much less than for the data $\mathbf{X}$, which is stored in $O(n_\mathbf{X} \cdot L_\mathbf{X} \cdot d)$ memory, which is important because the inducing tensors are variational parameters, and not amenable to subsampling, while the learning inputs can be subsampled as noted by [117]. Especially for GPUs memory cost is decisive and such savings are very important.

The computation of $K_{\mathbf{XX}}$ detailed in Appendix [266, App. D.2] has time complexity $O((M + d) \cdot n_\mathbf{X}^2 \cdot {L_\mathbf{X}}^2)$ and memory of $O(d \cdot n_\mathbf{X} \cdot L_\mathbf{X} + n_\mathbf{X}^2 \cdot {L_\mathbf{X}}^2)$. However, given a factorizing likelihood, one only requires $K_\mathbf{X} = [k(\mathbf{x}, \mathbf{x})]_{\mathbf{x} \in \mathbf{X}}$, which eliminates the quadratic cost in $n_\mathbf{X}$. It turns out that this is enough to train on GPUs with reasonable minibatch sizes (e.g. $n_\mathbf{X} = 50$) on several real world datasets. Finally, note that the ELBO (2.8) requires an additional matrix inversion and multiplication in $O(n_\mathbf{Z}^2 \cdot n_\mathbf{X} + n_\mathbf{Z}^3)$ time, which is not significant in our case.

2.5 Experiments

TS classification. Using GPFlow [64], Keras [46], we implemented three models: GPSig, GPSigLSTM, and GPSigGRU. All three use the signature covariance with the low-rank inducing tensors of Section 2.4. Note that in place of the signature kernel with the Euclidean lift (2.7), we precompose the signature kernel with the RBF kernel as detailed in Section 1.2.4 and 1.2.6, so that it has greater flexibility. All computations carry on mutatis mutandis by replacing inner product evaluations with the RBF kernel. Regarding the implemented models, GPSig is a plain vanilla variational GP classifier. Previous applications of neural nets to covariance constructions, in particular [290, 5], inspired GPSigLSTM and GPSigGRU that include an RNN as a sequence-to-sequence transformation with h hidden units; see Figures 2.2 and 2.3 where $\hat{\mathbf{x}}$ denotes augmentation with lags and $\mathsf{k}_{\hat{\mathbf{x}}}$ a static kernel lift. We benchmarked these GP models on 16 multivariate TS classification datasets, a collection introduced in [17] that was a semi-standard archive at the time of working on this project in TS classification, e.g. see the citations in [266, App. E.4] for papers that use these datasets. The same datasets are also used in [128] to compare several deep learning architectures for TSC.

As Bayesian baselines we used three GP models: (i) GPLSTM and (ii) GPGRU consist of an LSTM and a GRU network with an RBF kernel on top, in which case the RNNs are used as a sequence-to-vector transformation from $\mathsf{Seq}(\mathbb{R}^d)$ to $\mathbb{R}^h$; (iii) GPKConv1D uses the convolutional kernel introduced in [273] in 1-dimension (time). Throughout we used sparse variational inference: for GPSigLSTM, GPSigGRU, GPSig, the inducing tensors detailed in Section 2.4 are used; for GPLSTM and GPGRU the inducing points are located in the output space of the RNN layer, $\mathbb{R}^h$; for GPKConv1D, the inducing patches of [273] are used.

We used $n_{\mathbf{Z}} = 500$ for all models; further all use a static kernel in one form or another, which we fixed to be the RBF kernel. The signature kernel was truncated[2] at $M = 4$, and for GPSig $p = 1$ lags were used; the GPSigRNNs did not use lags, as the sequence of hidden states already incorporate lagged information about past observations. The window size in GPKConv1D was set to $w = 10$. The RNN-architectures were selected independently for all models by grid-search among 6 variants, that is, the number of hidden units from $\{8, 32, 128\}$ and with or without dropout. For training, early stopping was used with $n = 500$ epochs patience; a learning rate of $\alpha = 1 \times 10^{-3}$; a minibatch size of 50; as optimizer Adam [143] and Nadam [72] were employed. Implementations, datasets, and the training and grid-search methodology are respectively detailed in Appendices E.1, E.2, and E.3 in [266].

[2]For these experiments, the $M = 4$ value seemed to give an optimal trade-off between computational complexity and expressiveness of the kernel, see [266, App. E.1] for more details.

Table 2.2: Average ranks of GPs with the 1st and 2nd best in **bold** and *italicized* for each row.

	GPSigLSTM	GPSigGRU	GPSig	GPLSTM	GPGRU	GPKConv1D
MEAN RANK (NLPP, $n_X < 300$)	*2.80*	2.90	**2.20**	4.70	4.00	4.40
MEAN RANK (ACC., $n_X < 300$)	*3.00*	3.10	**2.80**	4.25	4.25	3.60
MEAN RANK (NLPP, $n_X \geq 300$)	**2.33**	3.33	*2.83*	4.83	4.33	3.33
MEAN RANK (ACC., $n_X \geq 300$)	**2.17**	3.50	*3.00*	4.17	4.33	3.83
MEAN RANK (NLPP, ALL)	*2.63*	3.06	**2.44**	4.75	4.13	4.00
MEAN RANK (ACC., ALL)	**2.69**	3.25	*2.88*	4.22	4.28	3.69

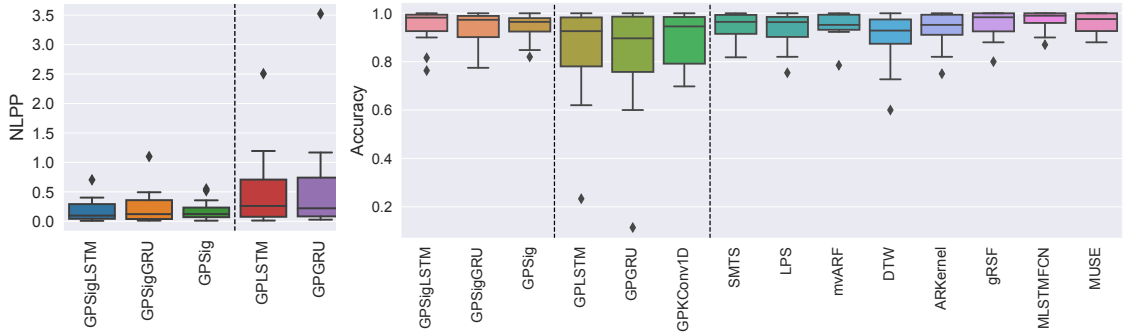

Figure 2.1: Box-plots of negative log-predictive probabilities (left) and classification accuracies (right) on 16 TSC datasets.

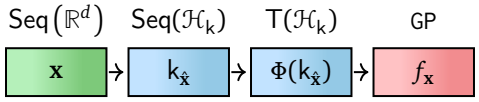

Figure 2.2: The GPSig model

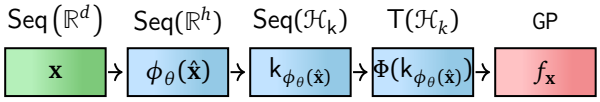

Figure 2.3: The GPSigRNN model

Discussion of results. For GPs, we report accuracies and negative log-predictive probabilities (NLPP), the latter take not only accuracies, but the calibration of probabilities into account as well. Table 2.2 shows the average ranks among the GPs. The full table of NLPPs and accuracies with mean and standard deviation over 5 model trains are reported in Appendix 2.B in Table 2.4 and Table 2.5. As non-Bayesian baselines, we report accuracies of eight frequentist TS classifiers in Table 2.6. On Figure 2.1, we visualize the box-plot distributions of (i) negative log-predictive probabilities of GPs, (ii) accuracies of both GPs and frequentist methods.

The signature models perform consistently the best in terms of average rankings of both NLPP and accuracy among the GPs. Particularly, they achieve stronger mean performance and a smaller variance across datasets. To explain this, inspecting the results in Tables 2.4, 2.5, we observe that all other GP baselines perform very poorly on some datasets, while the signature based models perform at least moderately well on all datasets. We believe this ties in to

Table 2.3: Average ranks of GPs in terms of negative log-predictive probabilities (NLPP) across data domains with the 1st and 2nd best in **bold** and *italicized* for each row.

DOMAIN	GPSigLSTM	GPSigGRU	GPSig	GPLSTM	GPGRU	GPKConv1D
HANDWRITING	*2.75*	4.00	**1.75**	3.75	5.50	3.25
MOTION	**1.92**	2.42	*2.25*	5.17	4.08	5.17
SENSOR	*2.75*	4.25	**2.38**	5.13	3.50	3.00
SPEECH	4.00	**1.50**	5.00	4.50	*2.50*	3.50

the universality property of signatures, see Section 1.2.2. The convolutional GP, GPKConv1D, which also has a very small parameter set, performed rather competitively with the deep kernel baselines, even on larger datasets. Comparison among variants of GPSig can be summarized as follows: for smaller datasets ($n_{\mathbf{X}} < 300$), GPSig outperforms other variants as it has a very small parameter set; for larger datasets ($n_{\mathbf{X}} \geq 300$), GPSigLSTM performs best which conforms with the intuition that RNNs suffer from small sample sizes. A related observation is that GPLSTM and GPGRU perform about on par, while GPSigLSTM does much better than GPSigGRU, which suggests that the signature makes explicit use of the additional complexity of the LSTM network.

Compared only in terms of accuracy, GPSig competes with frequentist classifiers: it outperforms the usual DTW baseline and competes with state-of-the art classifiers such as MUSE and MLSTMFCN. Purely based on accuracy, these win overall, but the difference is usually small, hence the extra Bayesian advantages come at a small cost. Obviously TS classification is a vast field and many other models could be considered; e.g. we did not use recurrent GPs or GPSSMs since (1) they have so-far not been used for TS classification, possibly because there is no sequential nature in the output space, (2) we did not find a GPFlow implementation that would allow to use sequence kernels in the GP transition function. (Implementation of [127] does currently not allow taking subsequences of past states in the transition function. An implementation would require much further work, but an interesting project would be to combine our models.)

Domain analysis. Next, we provide a breakdown of the performance of the various GP models across data domains. The datasets can be categorized into 4 different domains: handwriting, motion, sensor, and speech. The average ranks of NLPPs of each GP model for each data domain is provided in Table 2.3. We observe the following points: (1) on handwriting datasets, the vanilla GPSig model provides the best performance with GPSigLSTM in second and GPKConv1D in third while GPSigGRU underperforms; (2) on motion datasets, the additional flexibility of the preprocessing LSTM layer shines and GPSigLSTM takes first place while GPSig second and

GPSigGRU third; (3) on sensor datasets again GPSig performs the best and GPSigLSTM the second best while GPSigGRU severely underperforms; (4) somewhat surprisingly, on speech datasets the GRU architecture provides the best performance, and GPSigGRU takes first place while GPGRU second and GPKConv1D third while the other models underperform. Overall, we conclude that the vanilla GPSig model often provides sufficient performance, while augmenting it with a preprocessing LSTM layer can often provide further improvements. However, in some cases, i.e. as seen on speech data here, the inductive bias of the GRU layer is needed, and the signature kernel can only perform well when the data has been first fed through one such layer.

Inducing tensors vs inducing sequences. Our results rely on the inter-domain approach using tensors to locate inducing points from Section 2.4. An alternative is to use sequences for the inducing points, $\mathbf{Z} \subset \mathsf{Seq}(\mathcal{X})$, and controlling their maximal length $\mathsf{L}_{\mathbf{Z}} = \max_{\mathbf{z}\in\mathbf{Z}} \mathsf{L}_{\mathbf{z}}$ to be of order m, i.e. $\mathsf{L}_{\mathbf{Z}} \sim m$. We coin this approach *inducing sequences*. Intuitively, one expects the inducing tensors to be more efficient than inducing sequences, since they make full use of the structure of the signature feature space/covariance. To test this intuition, we compared the performance of the inducing tensors and inducing sequences subject to both having the same computational complexitiy. For this experiment, we took the AUSLAN dataset [73], which consists of $n_c = 95$ classes for $n_{\mathbf{X}} = 1140$ training examples. This is a challenging dataset as the inducing variables need to characterize the abundance of classification boundaries.

We used GPSig with the same settings as in the previous experiments. The hyperparameters of the kernel were a-priori learnt with $n_{\mathbf{Z}} = 500$ inducing tensors, and we purely investigated how the quality of the approximation changes for both approaches by varying the number of inducing points $n_{\mathbf{Z}}$. For each number of inducing variables, both approaches were trained independently 5 times for 300 epochs with random initialization of the inducing variables, for details [266, App. E.3]. We plot on Figure 2.4 three metrics: (1) the achieved ELBO, (2) the achieved accuracy, and (3) NLPP on the testing set. At $n_{\mathbf{Z}} = 500$ both approaches are close to saturation, but the inducing tensors consistently perform better. We remark that in practice, an important aspect is also how well the kernel hyperparameters can be recovered, that we did not consider here, and is a tricky question for sparse variational inference in general [16]. Although, intuition suggests that the closer the model to saturation is with respect to the inducing points, the more consistent should the optimization be with un-sparsified variational inference.

Visualizing inducing tensors. To gain more intuition, we visualized the feature space for one of the trained models on AUSLAN with $n_{\mathbf{Z}} = 500$ inducing tensors. We used UMAP [188] with the semi-metric $(\mathbf{x}, \mathbf{y}) \mapsto \sqrt{\mathsf{k}(\mathbf{x},\mathbf{x}) + \mathsf{k}(\mathbf{y},\mathbf{y}) - 2\mathsf{k}(\mathbf{x},\mathbf{y})}$ for $\mathbf{x}, \mathbf{y} \in \mathcal{X} \cup \mathcal{X}'$, see Figure 2.5. There are two imminent observations: (i) in the point cloud corresponding to the data, the

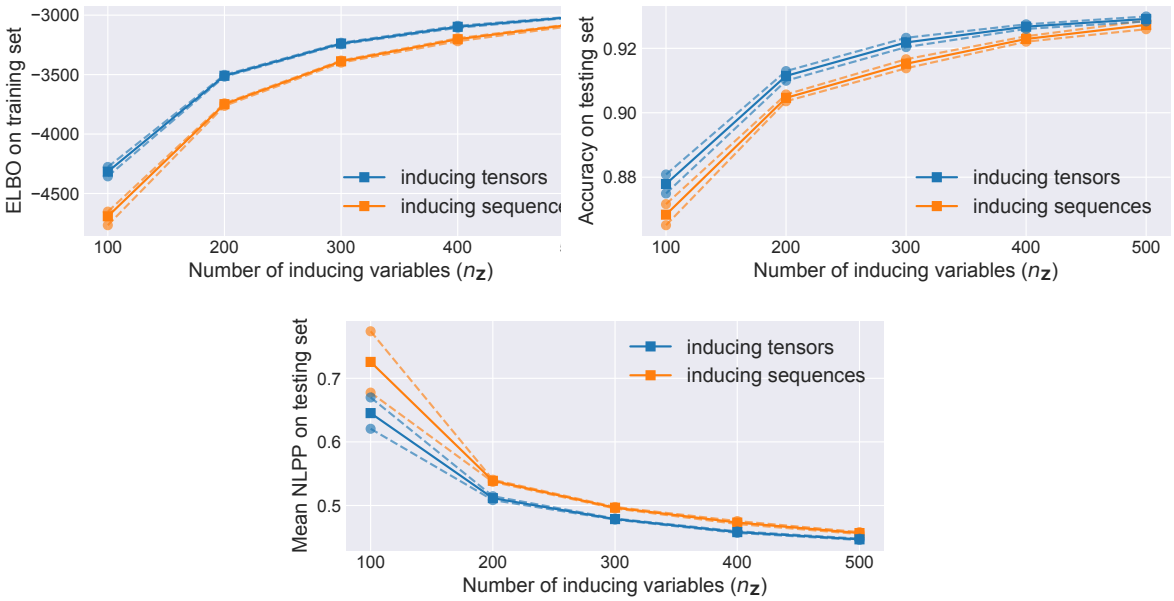

Figure 2.4: Achieved ELBO (top), accuracy (middle), mean NLPP (bottom) after 300 epochs of training the variational parameters with random initialization and frozen pre-learnt kernel hyperparameters. Solid is the mean over 5 independent runs, dashed is the 1-std region.

classes hardly look linearly separable; (ii) the tensors, however, seem to live on a completely separate subspace than the data. The algorithm achieves 92% accuracy on this set, therefore, point (i) is likely due to information being lost in the projection. However, point (ii) challenges the intuition about classical sparse variational inference, that the inducing points are located mixed-in with the data-points, concentrating close to the classification boundaries [115]. In general, the mechanism of how inter-domain inducing points represent the information in the data seems to be more complicated than classically.

To explain point (ii), we remark that this phenomenon is not surprising at all: signature features live in a manifold that is embedded in the linear tensor space $\mathsf{T}(\mathcal{H}_k)$. In general, sparse tensors of the form (2.4.2) will not be signatures of paths. We believe variational inference works because of an interplay of two factors: Firstly, signatures of finite sequences can be written as finite linear combinations of such sparse tensors. Secondly, the prior conditional term used to define $q(f_{\mathbf{x}}|f_{\mathbf{Z}})) = p(f_{\mathbf{x}}|f_{\mathbf{Z}})$. The feature space is congruent to the prior GP [21], which means that for $\mathbf{x} \in \mathsf{Seq}(\mathcal{X})$, the value of $f_{\mathbf{x}}$ given $f_{\mathbf{Z}}$ is not only almost deterministic when $\mathbf{x}$ is close to any of $\mathbf{z} \in \mathbf{Z}$, but when it is close to any linear combinations of elements in $\mathbf{Z}$. By the first remark this can always be achieved given a large enough $n_{\mathbf{Z}}$. To sum up, the inducing tensors do not represent signature features individually, but form atomic building blocks such that their linear combinations induce the actual variational posterior at the data-examples.

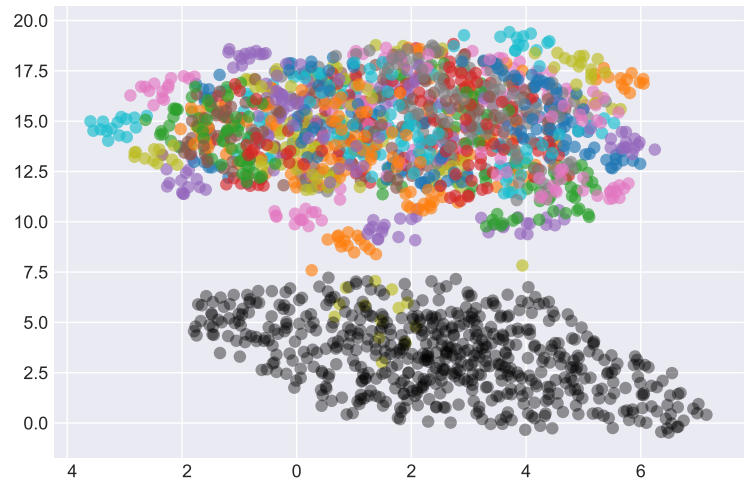

Figure 2.5: A UMAP visualization of the features corresponding to data-points (coloured), and inducing tensors (black) in the feature space on the AUSLAN dataset.

2.6 Conclusion

We used a classical object from stochastic analysis – signatures – to define a GP for sequential data. The GP inherits many of the theoretical guarantees that are known for signature features such as universality and parameterization invariance. To make it scalable, we develop "inducing tensors" that exploit the structure of the feature space, inter-domain inducing points, and variational inference. Applied in a plain vanilla variational framework, this yields a classifier, GPSig, that is not only competitive in terms of NLPP with other GP models, but also with state-of-the-art frequentist TS classifiers in terms of accuracy alone. As one of our reviewers remarked, several datasets we consider have a strong signal-to-noise ratio, which makes it worthwhile to point out that even for such datasets, the alternative GP baselines suffer on at least some of them, while the proposed models are consistently able to learn on all datasets. This observation ties in to the universality property, and it suggests that GPs with signatures can be a good starting point when building Bayesian models on time series datasets.

We also demonstrate that signatures can be used as a building block in deep kernels to build larger GP models that leverage the benefits of both, RNNs and signatures. Interestingly, we find that the vanilla GPSig model outperforms the GPSigRNNs for smaller datasets, conforming to the intuition that smaller sample sizes are detrimental for recurrent neural nets. To really get the best of both worlds, one could insert an additional model selection step, that specifies whether a parametric transformation is layer used before feeding the input into the kernel or not. Alternatively, it could also be possible to increase the flexibility of the sequential GP model while staying within a purely nonparametric framework using deep GPs [61] by e.g. applying a GP layer as observation-wise state-space embedding before the kernel computation. The inference framework of [226] for deep GPs could also come in handy when moving to datasets

with lower signal-to-noise ratios, which can require GP models capable of handling not only epistemic (reducible) uncertainty, but aleatoric (irreducible) uncertainty in the data. It would also be interesting to see if such sequence kernels can be used to improve recurrent GP models [187, 127] by incorporating sequential information into the GP transition function, that could potentially allow for a more efficient latent state representation.

2.A Continuity of GPs with Signature Covariances

We specified a covariance function k on the set $\mathsf{Paths}(\mathcal{X})$ as inner product of the signature map. This guarantees that $(\mathbf{x},\mathbf{y}) \mapsto \mathsf{k}(\mathbf{x},\mathbf{y}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{y})\rangle$ is a positive definite function, and from the general theory of stochastic processes by the Kolmogorov extension theorem [75], the existence of a centered GP $(f_{\mathbf{x}})_{\mathbf{x}\in\mathsf{Paths}(\mathcal{X})}$ such that $\mathbb{E}[f_{\mathbf{x}} f_{\mathbf{y}}] = \mathsf{k}(\mathbf{x},\mathbf{y})$. However, this does not guarantee that the sample paths $\mathbf{x} \mapsto f_{\mathbf{x}}$ are continuous. Seminal work of Dudley [74] showed that such regularity estimates can be derived by bounding the growth of the covering number of the index set of the GP f under the semi-metric

$$d_{\mathsf{k}}(\mathbf{x},\mathbf{y}) = \sqrt{\mathbb{E}[|f_{\mathbf{x}} - f_{\mathbf{y}}|^2]} = \sqrt{\mathsf{k}(\mathbf{x},\mathbf{x}) - 2\mathsf{k}(\mathbf{x},\mathbf{y}) + \mathsf{k}(\mathbf{y},\mathbf{y})}.$$

Already when when the index set is finite dimensional "nice" covariance functions can lead to discontinuous GPs, see e.g. Section 1.4. in [3]. Our GP has as index set the space of bounded variation paths $\mathsf{Paths}(\mathcal{X})$, which is infinite-dimensional so some caution is needed. However, as we show below we can cover this space by lattice paths and derive covering number estimates.

Theorem 2.3. *For $V > 0$ and $\epsilon > 0$ denote with $N(\epsilon, V)$ the covering number of the set*

$$\mathsf{Paths}_L(\mathcal{X}) = \{\mathbf{x} \in \mathsf{Paths}(\mathcal{X}) : \|\mathbf{x}\|_{1\text{-var}} \le V\}$$

of bounded variation paths of length less or equal than V under the $d_{\mathsf{k_s}}$ pseudo-metric. Then

$$\log_2 N(\epsilon, L) \le 2(d+1)L\frac{\sqrt{M}}{\epsilon}$$

Proof. By definition of the metric $d_{\mathsf{k_s}}$

$$\begin{aligned} d_{\mathsf{k_s}}(x,y) &\equiv \sqrt{\langle \Phi(\mathbf{x}) - \Phi(\mathbf{y}), \Phi(\mathbf{x}) - \Phi(\mathbf{y})\rangle} \\ &= \|\Phi(\mathbf{x}) - \Phi(\mathbf{y})\|. \end{aligned}$$

By definition Φ and of the norm $\|\cdot\|$ on $\prod_{m=0}^{M}\left(\mathbb{R}^d\right)^{\otimes m}$ this reads

$$\begin{aligned} d_{\mathsf{k_S}}^2(\mathbf{x},\mathbf{y}) &= \sum_{m=1}^{M} \| \int d\mathbf{x}^{\otimes m} - \int d\mathbf{y}^{\otimes m} \|^2 \\ &\leq M \max_{m=1,\dots,M} \Delta_m^2(\mathbf{x},\mathbf{y}) \end{aligned}$$

where we denote $\Delta_m(\mathbf{x},\mathbf{y}) = \| \int d\mathbf{x}^{\otimes m} - \int d\mathbf{y}^{\otimes m} \|$. Let $\mathsf{Lattice}_{s,L}(\mathcal{X}) \subset \mathsf{Paths}_L(\mathcal{X})$ be the set of lattice paths starting at $0 \in \mathbb{R}^d$ that take steps of size s and that are of total length at most L. By the results in Section 4 of [181], for every $\mathbf{x} \in \mathsf{Paths}_L(\mathcal{X})$ and every $n \geq 1$ there exists a $\mathbf{y} \in \mathsf{Lattice}_{L2^{-n},L}(\mathcal{X})$ such that for every $m \geq 1$,

$$\Delta_m(\mathbf{x},\mathbf{y}) \leq \frac{d}{2^{n-1}} \frac{4L^{m-1}}{(m-1)!}. \tag{2.11}$$

Since $\frac{L^{m-1}}{(m-1)!} \leq e^L - 1$ we can apply (2.11) with $n = n(\epsilon) = 1 - \log_2 \frac{\epsilon}{d\sqrt{M}4(e^L-1)}$ to get $\Delta_m(\mathbf{x},\mathbf{y}) \leq \epsilon$. Hence, there exists a lattice path $\mathbf{y} \in \mathsf{Lattice}_{L2^{-n(\epsilon)},L}(\mathcal{X})$ such that

$$d_{\mathsf{k_S}}(\mathbf{x},\mathbf{y}) \leq \epsilon.$$

Further, the set $\mathsf{Paths}(\mathcal{X})$ is finite and we can bound it by

$$\begin{aligned} |\mathsf{Lattice}_{L2^{-n(\epsilon)},L}(\mathcal{X})| &\leq (2^d+1)^{L2^{n(\epsilon)}} \leq 2^{(d+1)L2^{n(\epsilon)}} \\ &= 2^{2(d+1)L2^{n(\epsilon)-1}} = 2^{2(d+1)L\frac{\sqrt{M}}{\epsilon}} \end{aligned}$$

where the first inequality follows since a lattice path has at every step 2^d directions to choose from and in addition can choose not to make a step. The last equality follows from the definition of $n(\epsilon)$. Since $\mathbf{x} \in \mathsf{Paths}_L(\mathcal{X})$ was chosen arbitrary it follows that $\mathsf{Paths}_L(\mathcal{X})$ can be covered by $2^{2(d+1)L\frac{\sqrt{M}}{\epsilon}}$ balls of radius ϵ centered at lattice paths. □

Theorem 2.3 combined with Dudley's celebrated entropy estimates gives regularity results for samples of our GP. In fact, this even yields a modulus of continuity for our GP.

Theorem 2.4. *There exists a centered* GP $(f_{\mathbf{x}})_{\mathbf{x}\in\mathcal{X}_{paths}^{L}}$ *that has a covariance* $\mathbb{E}[f_{\mathbf{x}}f_{\mathbf{y}}]$ *the signature covariance function* $k(\mathbf{x},\mathbf{y}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle$. *Moreover, if we denote its modulus of continuity on*

$\mathsf{Paths}_L(\mathcal{X})$ *with*

$$\omega(\delta) = \sup_{\substack{\mathbf{x},\mathbf{y}\in\mathsf{Paths}_L(\mathcal{X}) \\ d_{\mathsf{kS}}(\mathbf{x},\mathbf{y})<\delta}} |f_{\mathbf{x}} - f_{\mathbf{y}}|$$

then it holds with probability one that

$$\limsup_{\delta\to 0} \frac{\omega(\delta)}{\sqrt{\delta}4\sqrt{(d+1)L\sqrt{M}} + c\delta\sqrt{\ln\ln\frac{1}{\delta}}} \leq 24 \tag{2.12}$$

where $c > 0$ denotes a universal constant.

Proof. The existence of a centered GP $(\hat{f}_{\mathbf{x}})_{\mathbf{x}}$ with covariance k follows from general results about Gaussian processes. The existence of a continuous modification $(f_{\mathbf{x}})_{\mathbf{x}\in\mathcal{X}_{paths,L}}$ of follows from Dudley's theorem if

$$\int_0^1 \sqrt{\log_2 N(\epsilon, L)}d\epsilon < \infty$$

but by Theorem 2.3 we have

$$\int_0^1 \sqrt{\log_2 N(\epsilon, L)}d\epsilon \leq \sqrt{2(d+1)L\sqrt{M}}\int_0^1 \frac{1}{\sqrt{\epsilon}}d\epsilon < \infty.$$

Dudley's results immediately yield a modulus of continuity in probability. By standard arguments this can be strengthened to give an almost sure modulus of continuity. Concretely, we use the formulation given in Theorem 2.7.1 in Chapter 5 of [138] which guarantees that

$$\limsup_{\delta\to 0} \frac{\omega(\delta)}{\int_0^\delta \sqrt{N(\frac{\epsilon}{2}, L)}d\epsilon + c\delta\sqrt{\ln\ln\frac{1}{\delta}}} \leq 24.$$

The bound (2.12) follows immediately since first term in the denominator equals

$$\begin{aligned}\int_0^\delta \sqrt{\log_2 N\left(\frac{\epsilon}{2}, L\right)}d\epsilon &= \sqrt{2(d+1)L\sqrt{M}}2\sqrt{2}\sqrt{\delta} \\ &= \sqrt{\delta}4\sqrt{(d+1)L\sqrt{M}}.\end{aligned}$$

□

2.B Benchmark results

We report in Table 2.4 and Table 2.5 the negative log-predictive probabilities and accuracies of the GP models considered in Section 2.5. For each method-dataset pair, 5 models were trained with initialization described in [266, App. E.3]. The variance of the results is therefore due to random initialization and minibatch randomness. The RNN based models used the architectures detailed in [266, Tab. 4]. As non-Bayesian baselines, we report the results of recent frequentist TS classification methods from the respective publications, that is, [59, 18, 19, 137, 269, 232, 136]. Particularly for MLSTMFCN, we report the same results as in [232].

Table 2.4: Mean and standard deviation of NLPP on test sets over 5 independent runs

Dataset	GPSigLSTM	GPSigGRU	GPSig	GPLSTM	GPGRU	GPKConv1D
Arabic Digits	0.047 ± 0.030	0.023 ± 0.006	0.071 ± 0.021	0.082 ± 0.022	0.066 ± 0.010	0.050 ± 0.003
AUSLAN	0.106 ± 0.007	0.123 ± 0.045	0.550 ± 0.114	0.650 ± 0.071	0.248 ± 0.063	1.900 ± 0.139
Character Traj.	0.031 ± 0.007	0.258 ± 0.265	0.108 ± 0.005	2.506 ± 1.007	3.523 ± 0.635	0.409 ± 0.141
CMUsubject16	0.088 ± 0.020	0.040 ± 0.009	0.089 ± 0.027	0.270 ± 0.080	0.089 ± 0.039	0.255 ± 0.002
DigitShapes	0.008 ± 0.001	0.035 ± 0.051	0.021 ± 0.001	0.013 ± 0.002	0.727 ± 0.569	0.035 ± 0.003
ECG	0.402 ± 0.023	0.431 ± 0.037	0.356 ± 0.008	0.496 ± 0.018	0.601 ± 0.137	0.543 ± 0.019
Jap. Vowels	0.080 ± 0.031	0.053 ± 0.009	0.069 ± 0.003	0.061 ± 0.029	0.052 ± 0.005	0.067 ± 0.001
Kick vs Punch	0.301 ± 0.109	0.493 ± 0.128	0.224 ± 0.014	0.696 ± 0.046	0.674 ± 0.037	0.662 ± 0.017
LIBRAS	0.320 ± 0.045	0.346 ± 0.091	0.259 ± 0.021	0.911 ± 0.056	1.110 ± 0.248	1.608 ± 0.311
NetFlow	0.218 ± 0.009	0.259 ± 0.078	0.189 ± 0.014	0.251 ± 0.041	0.194 ± 0.011	0.168 ± 0.081
PEMS	0.704 ± 0.130	1.100 ± 0.064	0.520 ± 0.058	1.194 ± 0.308	0.784 ± 0.111	0.537 ± 0.010
PenDigits	0.289 ± 0.127	0.399 ± 0.206	0.146 ± 0.007	0.185 ± 0.027	0.187 ± 0.043	0.181 ± 0.005
Shapes	0.014 ± 0.004	0.012 ± 0.004	0.011 ± 0.002	0.016 ± 0.008	0.168 ± 0.142	0.012 ± 0.001
UWave	0.113 ± 0.011	0.121 ± 0.017	0.140 ± 0.004	0.745 ± 0.151	1.168 ± 1.063	0.189 ± 0.008
Wafer	0.048 ± 0.021	0.081 ± 0.011	0.105 ± 0.010	0.105 ± 0.086	0.029 ± 0.011	0.085 ± 0.002
Walk vs Run	0.030 ± 0.008	0.030 ± 0.008	0.023 ± 0.007	0.048 ± 0.040	0.028 ± 0.000	0.066 ± 0.001
Mean	0.175	0.238	0.180	0.514	0.603	0.423
Avg. rank ($n_{\mathbf{x}} < 300$)	2.800	2.900	2.200	4.700	4.000	4.400
Avg. rank ($n_{\mathbf{x}} \geq 300$)	2.333	3.333	2.833	4.833	4.333	3.333
Avg. rank (all)	2.625	3.062	2.438	4.750	4.125	4.000

Table 2.5: Mean and standard deviation of accuracies on test sets over 5 independent runs

Dataset	GPSigLSTM	GPSigGRU	GPSig	GPLSTM	GPGRU	GPKConv1D
Arabic Digits	0.992 ± 0.003	0.994 ± 0.002	0.979 ± 0.004	0.985 ± 0.004	0.986 ± 0.005	0.984 ± 0.001
AUSLAN	0.983 ± 0.003	0.978 ± 0.006	0.925 ± 0.014	0.880 ± 0.012	0.949 ± 0.014	0.784 ± 0.012
Character Traj.	0.991 ± 0.003	0.925 ± 0.078	0.979 ± 0.002	0.233 ± 0.331	0.114 ± 0.050	0.941 ± 0.013
CMUsubject16	1.000 ± 0.000	1.000 ± 0.000	0.979 ± 0.017	0.924 ± 0.051	0.993 ± 0.014	0.897 ± 0.000
DigitShapes	1.000 ± 0.000	0.988 ± 0.025	1.000 ± 0.000	1.000 ± 0.000	0.812 ± 0.153	1.000 ± 0.000
ECG	0.816 ± 0.029	0.832 ± 0.012	0.848 ± 0.010	0.782 ± 0.032	0.734 ± 0.033	0.760 ± 0.018
Jap. Vowels	0.981 ± 0.005	0.985 ± 0.004	0.982 ± 0.005	0.982 ± 0.004	0.986 ± 0.005	0.986 ± 0.002
Kick vs Punch	0.900 ± 0.063	0.820 ± 0.098	0.900 ± 0.000	0.620 ± 0.075	0.600 ± 0.110	0.700 ± 0.089
LIBRAS	0.921 ± 0.013	0.899 ± 0.031	0.923 ± 0.004	0.776 ± 0.019	0.742 ± 0.050	0.698 ± 0.026
NetFlow	0.931 ± 0.002	0.921 ± 0.012	0.937 ± 0.003	0.928 ± 0.011	0.926 ± 0.012	0.945 ± 0.027
PEMS	0.763 ± 0.016	0.775 ± 0.019	0.820 ± 0.014	0.745 ± 0.044	0.769 ± 0.020	0.794 ± 0.008
PenDigits	0.928 ± 0.030	0.902 ± 0.048	0.955 ± 0.002	0.953 ± 0.008	0.951 ± 0.008	0.946 ± 0.001
Shapes	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	0.867 ± 0.163	1.000 ± 0.000
UWave	0.970 ± 0.004	0.968 ± 0.006	0.964 ± 0.001	0.870 ± 0.029	0.763 ± 0.225	0.947 ± 0.002
Wafer	0.988 ± 0.005	0.978 ± 0.005	0.965 ± 0.004	0.966 ± 0.037	0.994 ± 0.002	0.984 ± 0.001
Walk vs Run	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
Mean	0.948	0.935	0.947	0.853	0.824	0.898
Avg. rank ($n_{\mathbf{x}} < 300$)	3.000	3.100	2.800	4.250	4.250	3.600
Avg. rank ($n_{\mathbf{x}} \geq 300$)	2.167	3.500	3.000	4.167	4.333	3.833
Avg. rank (all)	2.688	3.250	2.875	4.219	4.281	3.688

Table 2.6: Accuracies of frequentist time series classification methods

Dataset	SMTS	LPS	mvARF	DTW	ARKernel	gRSF	MLSTMFCN	MUSE
Arabic Digits	0.964	0.971	0.952	0.908	0.988	0.975	0.990	0.992
AUSLAN	0.947	0.754	0.934	0.727	0.918	0.955	0.950	0.970
Character Traj.	0.992	0.965	0.928	0.948	0.900	0.994	0.990	0.937
CMUsubject16	0.997	1.000	1.000	0.930	1.000	1.000	1.000	1.000
DigitShapes	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
ECG	0.818	0.820	0.785	0.790	0.820	0.880	0.870	0.880
Jap. Vowels	0.969	0.951	0.959	0.962	0.984	0.800	1.000	0.976
Kick vs Punch	0.820	0.900	0.976	0.600	0.927	1.000	0.900	1.000
LIBRAS	0.909	0.903	0.945	0.888	0.952	0.911	0.970	0.894
NetFlow	0.977	0.968	NA	0.976	NA	0.914	0.950	0.961
PEMS	0.896	0.844	NA	0.832	0.750	1.000	NA	NA
PenDigits	0.917	0.908	0.923	0.927	0.952	0.932	0.970	0.912
Shapes	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
UWave	0.941	0.980	0.952	0.916	0.904	0.929	0.970	0.916
Wafer	0.965	0.962	0.931	0.974	0.968	0.992	0.990	0.997
Walk vs Run	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000

Chapter 3

Seq2Tens

3.1 Introduction

A central task of learning is to find representations of the underlying data that efficiently and faithfully capture their structure. In the case of sequential data, one data point consists of a sequence of objects. This is a rich and non-homogeneous class of data and includes classical uni- or multi-variate time series (sequences of scalars or vectors), video (sequences of images), and text (sequences of letters). Particular challenges of sequential data are that each sequence entry can itself be a highly structured object and that data sets typically include sequences of different length which makes naive vectorization troublesome.

Contribution. Our main result is a generic method that takes a *static feature map* for a class of objects (e.g. a feature map for vectors, images, or letters) as input and turns this into a feature map for sequences of arbitrary length of such objects (e.g. a feature map for time series, video, or text). We call this feature map for sequences `Seq2Tens`; among its attractive properties are that it (i) provides a structured description of sequences; generalizing classical methods for strings, (ii) comes with theoretical guarantees such as universality, (iii) can be turned into modular and flexible neural network (`NN`) layers. The key ingredient to our approach is to embed the feature space of the static feature map into the tensor algebra, and then the product in this algebra is then used to "stitch together" the static features of the sequence entries in a structured way.

Outline. Section 3.2 formalizes the main ideas of `Seq2Tens`. Section 3.3 shows how low-rank (`LR`) constructions combined with `seq2seq` transforms allows one to efficiently use this rich algebraic structure. Section 3.4 applies the results of Sections 3.2 and 3.3 to build modular

and scalable NN layers for sequential data. Section 3.5 demonstrates the flexibility and modularity of this approach on both discriminative and generative benchmarks. Section 3.6 makes connections with previous work and summarizes this article. In the appendices we provide mathematical background, extensions, and detailed proofs for our theoretical results.

3.2 Capturing order by non-commutative multiplication

We denote the set of sequences of elements in a set $\mathcal{X}$ by

$$\mathsf{Seq}(\mathcal{X}) = \{\mathbf{x} = (\mathbf{x}_i)_{i=1,\dots,L} : \mathbf{x}_i \in \mathcal{X},\, L \geq 1\}$$

where $L \geq 1$ is some arbitrary length. Notice that in this chapter we use 1-based indexing for sequences, and for a sequence $\mathbf{x} = (\mathbf{x}_1, \dots, \mathbf{x}_L) \in \mathsf{Seq}(\mathcal{X})$, $\mathsf{L}_\mathbf{x} = L$. Even if $\mathcal{X}$ itself is a linear space, e.g. $\mathcal{X} = \mathbb{R}$, $\mathsf{Seq}(\mathcal{X})$ is never a linear space since there is no natural addition of two sequences of different length.

3.2.1 Seq2Tens in a nutshell

Given any vector space V we may construct the tensor algebra $\mathsf{T}((V))$ over V. We describe the space $\mathsf{T}((V))$ in detail in Section 1.1.2, but as for now the only thing that is important is that $\mathsf{T}((V))$ is also a vector space that includes V, and that it carries a non-commutative product, which is, in a precise sense, "the most general product" on V.

The main idea of Seq2Tens is that any "static feature map" for elements in $\mathcal{X}$, $\phi : \mathcal{X} \to V$, can be used to construct a new feature map $\Phi : \mathsf{Seq}(\mathcal{X}) \to \mathsf{T}((V))$ for sequences in $\mathcal{X}$ by using the algebraic structure of $\mathsf{T}((V))$: the non-commutative product on $\mathsf{T}((V))$ makes it possible to "stitch together" the individual features $\phi(\mathbf{x}_1), \dots, \phi(\mathbf{x}_L) \in V \subset \mathsf{T}((V))$ of the sequence $\mathbf{x}$ in the larger space $\mathsf{T}((V))$ by multiplication in $\mathsf{T}((V))$. With this we may define the feature map $\Phi(\mathbf{x})$ for a sequences $\mathbf{x} = (\mathbf{x}_1, \dots, \mathbf{x}_L) \in \mathsf{Seq}(\mathcal{X})$ as follows

(i) lift the map $\phi : \mathcal{X} \to V$ to a map $\varphi : \mathcal{X} \to \mathsf{T}((V))$,

(ii) map $\mathsf{Seq}(\mathcal{X}) \to \mathsf{Seq}(\mathsf{T}((V)))$ by $(\mathbf{x}_1, \dots, \mathbf{x}_L) \mapsto (\varphi(\mathbf{x}_1), \dots, \varphi(\mathbf{x}_L))$,

(iii) map $\mathsf{Seq}(\mathsf{T}((V))) \to \mathsf{T}((V))$ by multiplication $(\varphi(\mathbf{x}_1), \dots, \varphi(\mathbf{x}_L)) \mapsto \varphi(\mathbf{x}_1) \cdots \varphi(\mathbf{x}_L)$.

In a more concise form, we define Φ as

$$\Phi : \mathsf{Seq}(\mathcal{X}) \to \mathsf{T}((V)), \quad \Phi(\mathbf{x}) = \prod_{i=1}^{L} \varphi(\mathbf{x}_i) \tag{3.1}$$

where $\prod$ denotes multiplication in $\mathsf{T}((V))$. We refer to the resulting map Φ as the `Seq2Tens` map, which stands short for ***Seq**uences-**2**-**Tens**ors*. Why is this construction a good idea? First note, that step (i) is always possible since $V \subset \mathsf{T}((V))$ and we discuss the simplest such lift before Theorem 3.1 as well as other choices in [263, App. B]. Further, if ϕ, respectively φ, provides a faithful representation of objects in $\mathcal{X}$, then there is no loss of information in step (ii). Finally, since step (iii) uses "the most general product" to multiply $\varphi(\mathbf{x}_1)\cdots\varphi(\mathbf{x}_L)$ one expects that $\Phi(\mathbf{x}) \in \mathsf{T}((V))$ faithfully represents the sequence $\mathbf{x}$ as an element of $\mathsf{T}((V))$.

Indeed in Theorem 3.1 below we show an even stronger statement, namely that if the static feature map $\phi : \mathcal{X} \to V$ contains enough non-linearities so that non-linear functions from $\mathcal{X}$ to $\mathbb{R}$ can be approximated as *linear functions* of the static feature map ϕ, then the above construction extends this property to functions of sequences. Put differently, *if ϕ is a universal feature map for $\mathcal{X}$, then Φ is a universal feature map for* $\mathsf{Seq}(\mathcal{X})$; that is, any non-linear function $f(\mathbf{x})$ of a sequence $\mathbf{x}$ can be approximated as a linear functional of $\Phi(\mathbf{x})$, $f(\mathbf{x}) \approx \langle \ell, \Phi(\mathbf{x}) \rangle$. We also emphasize that the domain of Φ is the space $\mathsf{Seq}(\mathcal{X})$ of sequences of *arbitrary* (finite) length. The remainder of this Section gives details about steps (i),(ii),(iii) for the construction of Φ.

3.2.2 Construction

Lifting static feature maps. Step (i) in the construction of Φ requires turning a given feature map $\phi : \mathcal{X} \to V$ into a map $\varphi : \mathcal{X} \to \mathsf{T}((V))$. Throughout the rest of this article we use the lift

$$\varphi(\mathbf{x}) = (1, \phi(\mathbf{x}), 0, 0 \ldots) \in \mathsf{T}((V)). \tag{3.2}$$

We discuss other choices in [266, App. B], but attractive properties of the lift 3.2 are that (1) the evaluation of Φ against low-rank tensors becomes a simple recursive formula (Proposition 3.3,) (2) it is a generalization of sequence sub-pattern matching as used in string kernels ([263, App. B.3], (3) despite its simplicity it performs exceedingly well in practice (Section 3.4).

Extending to sequences of arbitrary length. Steps (i) and (ii) in the construction specify how the map $\Phi : \mathcal{X} \to \mathsf{T}((V))$ behaves on sequences of length-1, that is, single observations. Step (iii) amounts to the requirement that for any two sequences $\mathbf{x}, \mathbf{y} \in \mathsf{Seq}(V)$, their concatenation defined as $\mathbf{z} = (\mathbf{x}_0, \ldots, \mathbf{x}_{\mathsf{L}_x}, \mathbf{y}_0, \ldots, \mathbf{y}_{\mathsf{L}_y}) \in \mathsf{Seq}(V)$ can be understood in the feature space as (non-commutative) multiplication of their corresponding features

$$\Phi(\mathbf{z}) = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{y}).$$

In other words, we inductively extend the lift φ to sequences of arbitrary length by starting from sequences consisting of a single observation, which is given in (3.1). Repeatedly applying the definition of the tensor convolution product in (1.2) leads to the following explicit formula

$$\Phi_m(\mathbf{x}) = \sum_{\mathbf{i}\in\Delta_m(\mathsf{L}_\mathbf{x})} \mathbf{x}_{i_1} \otimes \cdots \otimes \mathbf{x}_{i_m} \in V^{\otimes m}, \;\; \Phi(\mathbf{x}) = (\Phi_m(\mathbf{x}))_{m\geq 0}, \tag{3.3}$$

where $\mathbf{x} = (\mathbf{x}_0, \ldots, \mathbf{x}_{\mathsf{L}_\mathbf{x}}) \in \mathsf{Seq}(V)$ and the summation is over non-contiguous subsequences.

Some intuition: generalized pattern matching. Our derivation of the feature map $\Phi(\mathbf{x}) =\in \mathsf{T}((V))$ was guided by general algebraic principles, but (3.3) provides an intuitive interpretation. It shows that for each $m \geq 1$, the entry $\Phi_m(\mathbf{x}) \in V^{\otimes m}$ constructs a summary of a long sequence $\mathbf{x} \in \mathsf{Seq}(V)$ based on subsequences $(\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_m})$ of $\mathbf{x}$ of length-m. It does this by taking the usual tensor product (see Section 1.1) $\mathbf{x}_{i_1} \otimes \cdots \otimes \mathbf{x}_{i_m} \in V^{\otimes m}$ and summing over all possible non-contiguous subsequences. This is completely analogous to how string kernels provide a structured description of text by looking at non-contiguous substrings of length-m (indeed, Appendix [263, App. B.3] makes this rigorous). However, the main difference is that the above construction works for arbitrary sequences and not just sequences of discrete letters.

The algebraic background allows to prove that Φ is universal, see Theorem 3.1 below.

Universality. A function $\phi : \mathcal{X} \to V$ is said to be *universal for* $\mathcal{X}$ if all continuous functions on $\mathcal{X}$ can be approximated as linear functions on the image of ϕ. One of the most powerful features of NNs is their universality [124]. A very attractive property of Φ is that it preserves universality: if $\phi : \mathcal{X} \to V$ is universal for $\mathcal{X}$, then $\Phi : \mathsf{Seq}(\mathcal{X}) \to \mathsf{T}((V))$ is universal for $\mathsf{Seq}(\mathcal{X})$. To make this precise, note that $V^{\otimes m}$ is a linear space and therefore any $\ell = (\ell_0, \ell_1, \ldots, \ell_M, 0, 0, \ldots) \in \mathsf{T}((V))$ consisting of M tensors $\ell_m \in V^{\otimes m}$, yields a linear functional on $\mathsf{T}((V))$; e.g. if $V = \mathbb{R}^d$ and we identify ℓ_m in coordinates as $\ell_m = (\ell_m^{i_1,\ldots,i_m})_{i_1,\ldots,i_m\in\{1,\ldots,d\}}$ then

$$\langle \ell, \mathbf{t} \rangle := \sum_{m=0}^{M} \langle \ell_m, \mathbf{t}_m \rangle = \sum_{m=0}^{M} \sum_{i_1,\ldots,i_m\in\{1,\ldots,d\}} \ell_m^{i_1,\ldots,i_m} \mathbf{t}_m^{i_1,\ldots,i_m}.$$

Thus linear functionals of the feature map Φ, are real-valued functions of sequences. Theorem 3.1 below shows that any continuous function $f : \mathsf{Seq}(\mathcal{X}) \to \mathbb{R}$ can by arbitrary well approximated by a $\ell \in \mathsf{T}((V))$, $f(\mathbf{x}) \approx \langle \ell, \Phi(\mathbf{x}) \rangle$.

Theorem 3.1 (Universality)**.** *Let $\phi : \mathcal{X} \to V$ be a universal map with a lift that satisfies some mild constraints, then the following map is universal:*

$$\Phi : \mathsf{Seq}(\mathcal{X}) \to \mathsf{T}((V)), \quad \mathbf{x} \mapsto \Phi(\mathbf{x}).$$

A detailed proof and the precise statement of Theorem 3.1 is given in [263, App. B].

3.3 Approximation by low-rank linear functionals

The combinatorial explosion of tensor coordinates and what to do about it. The universality of Φ suggests the following approach to represent a function $f : \mathsf{Seq}(\mathcal{X}) \to \mathbb{R}$ of sequences: First compute $\Phi(\mathbf{x})$ and then optimize over ℓ (and possibly also the hyperparameters of ϕ) such that $f(\mathbf{x}) \approx \langle \ell, \Phi(\mathbf{x}) \rangle = \sum_{m=0}^{M} \langle \ell_m, \Phi_m(\mathbf{x}) \rangle$. Unfortunately, tensors suffer from a combinatorial explosion in complexity, i.e. even just storing $\Phi_m(\mathbf{x}) \in V^{\otimes m} \subset \mathsf{T}((V))$ requires $O(\dim(V)^m)$ real numbers. Below we resolve this computational bottleneck as follows: in Proposition 3.3 we show that for a special class of low-rank elements $\ell \in \mathsf{T}((V))$, the functional $\mathbf{x} \mapsto \langle \ell, \Phi(\mathbf{x}) \rangle$ can be efficiently computed in both time and memory. This is somewhat analogous to a kernel trick since it shows that $\langle \ell, \Phi(\mathbf{x}) \rangle$ can be cheaply computed without explicitly computing the feature map $\Phi(\mathbf{x})$. However, Theorem 3.1 guarantees universality under no restriction on ℓ, thus restriction to rank-1 functionals limits the class of functions $f(\mathbf{x})$ that can be approximated. Nevertheless, by iterating these "low-rank functional" constructions in the form of `seq2seq` transformations this can be ameliorated. We give the details below but to gain intuition, we invite the reader to think of this iteration analogous to stacking layers in a neural network: each layer is a relatively simple non-linearity (e.g. a sigmoid composed with an affine function) but by composing such layers, complicated functions can be efficiently approximated.

Rank-1 functionals are computationally cheap. Degree $m = 2$ tensors are matrices and low-rank approximations are widely used in practice [270] to address quadratic complexity. The definition below generalizes the rank of matrices to tensors of any degree m.

Definition 3.2. *The* rank *(also called* `CP` rank *[31]) of a degree-m tensor $\mathbf{t}_m \in V^{\otimes m}$ is the smallest number $r \geq 0$ such that one may write*

$$\mathbf{t}_m = \sum_{i=0}^{r} \mathbf{v}_i^1 \otimes \cdots \otimes \mathbf{v}_i^m, \quad \mathbf{v}_i^1, \ldots, \mathbf{v}_i^m \in V.$$

We say that $\mathbf{t} = (\mathbf{t}_m)_{m\geq 0} \in \mathsf{T}((V))$ *has rank-*1 *(and truncation-M) if each* $\mathbf{t}_m \in V^{\otimes m}$ *is a rank-*1 *tensor and* $\mathbf{t}_i = 0$ *for* $i > M$.

The ranks of signature tensors related are investigated in [90]. A direct calculation shows that if ℓ is of rank-1, then $\langle \ell, \Phi(\mathbf{x})\rangle$ can be computed efficiently by inner product evaluations.

Proposition 3.3. *Let* $\ell = (\ell_m)_{m\geq 0} \in \mathsf{T}((V))$ *be of rank-*1 *and truncation-M. Then,*

$$\langle \ell, \Phi(\mathbf{x})\rangle = \sum_{m=0}^{M} \sum_{\mathbf{i}\in\Delta_m(\mathsf{L}_\mathbf{x})} \prod_{k=1}^{m} \langle \mathbf{v}_k^m, \phi(\mathbf{x}_{i_k})\rangle \tag{3.4}$$

where $\ell_m = \mathbf{v}_1^m \otimes \cdots \otimes \mathbf{v}_m^m \in V^{\otimes m}$, $\mathbf{v}_i^m \in V$ *and* $m = 0, \ldots, M$.

Note that the inner sum is taken over all non-contiguous subsequences of $\mathbf{x}$ of length-m, analogously to m-mers of strings; the proof of Proposition 3.3 is given in [263, App. B]. While (3.4) looks expensive, by casting it into a recursive formulation over time, it can be computed in $O(M^2 \cdot \mathsf{L}_\mathbf{x} \cdot d)$ time and $O(M^2 \cdot (\mathsf{L}_\mathbf{x} + c))$ memory, where d is the inner product evaluation time on V, while c is the memory footprint of a $v \in V$. This can further be reduced to $O(M \cdot \mathsf{L}_\mathbf{x} \cdot d)$ time and $O(M \cdot (\mathsf{L}_\mathbf{x} + c))$ memory by an efficient parameterization of the rank-1 element $\ell \in \mathsf{T}((V))$. We give further details and numerical scalability analysis in Appendix 3.A.

Low-rank Seq2Tens maps. The composition of a linear map $\mathcal{L} : \mathsf{T}((V)) \to \mathbb{R}^N$ with Φ can be computed cheaply in parallel using (3.4) when $\mathcal{L}$ is specified through a collection of $N \in \mathbb{N}$ rank-1 elements $\ell^1, \ldots, \ell^N \in \mathsf{T}((V))$ such that

$$\tilde{\Phi}_{\tilde{\theta}}(\mathbf{x}_1, \ldots, \mathbf{x}_L) := \mathcal{L} \circ \Phi(\mathbf{x}_1, \ldots, \mathbf{x}_L) = (\langle \ell^j, \Phi(\mathbf{x}_1, \ldots, \mathbf{x}_L)\rangle)_{j=1}^N \in \mathbb{R}^N.$$

We call the resulting map $\tilde{\Phi}_{\tilde{\theta}} : \mathsf{Seq}(\mathcal{X}) \to \mathbb{R}^N$ a ***Low-rank Seq2Tens*** map of width-N and truncation-M, where $M \in \mathbb{N}$ is the maximal degree of $\ell^1, \ldots, \ell^N$ such that $\ell_i^j = 0$ for $i > M$. The `LS2T` map is parametrized by (1) the component vectors $\mathbf{v}_{j,m}^k \in V$ of the rank-1 elements $\ell_m^j = \mathbf{v}_{j,m}^1 \otimes \cdots \otimes \mathbf{v}_{j,m}^m$, (2) by any parameters θ that the static feature map $\phi_\theta : \mathcal{X} \to V$. We denote these parameters by $\tilde{\theta} = (\theta, \ell^1, \ldots, \ell^N)$. In addition, by the subsequent composition of $\tilde{\Phi}_{\tilde{\theta}}$ with a linear functional $\mathbb{R}^N \to \mathbb{R}$, we get the following function subspace as hypothesis class

$$\tilde{\mathcal{H}} = \Big\{ \langle \sum_{j=1}^{N} \alpha_j \ell^j, \Phi(\mathbf{x}_1, \ldots, \mathbf{x}_L)\rangle \,:\, \alpha_j \in \mathbb{R} \Big\} \subsetneq \mathcal{H} = \Big\{ \langle \ell, \Phi(\mathbf{x}_1, \ldots, \mathbf{x}_L) : \ell \in \mathsf{T}((V)) \Big\}$$

Hence, we acquire an intuitive explanation of the (hyper)parameters: the width of the `LS2T`, $N \in \mathbb{N}$ specifies the maximal rank of the low-rank linear functionals of Φ that the `LS2T` can

represent, while the span of the rank-1 elements, $\mathsf{span}(\ell^1, \ldots, \ell^N)$ determine an N-dimensional subspace of the dual space of $\mathsf{T}((V))$ consisting of at most rank-N functionals.

Recall now that without rank restrictions on the linear functionals of Seq2Tens features, Theorem 3.1 would guarantee that any real-valued function $f : \mathsf{Seq}(\mathcal{X}) \to \mathbb{R}$ could be approximated by $f(\mathbf{x}) \approx \langle \ell, \Phi(\mathbf{x}_1, \ldots, \mathbf{x}_L) \rangle$. As pointed out before, the restriction of the hypothesis class to low-rank linear functionals of $\Phi(\mathbf{x}_1, \ldots, \mathbf{x}_L)$ would limit the class of functions of sequences that can be approximated. To ameliorate this, we use LS2T transforms in a seq2seq fashion that allows us to stack such low-rank functionals, significantly recovering expressiveness.

Sequence-to-sequence transforms. Motivated by the empirical successes of stacked RNNs [105, 104, 254], we build sequence-to-sequence (seq2seq) transformations. We can use LS2T to build seq2seq transformations in the following way: fix the static map $\phi_\theta : \mathcal{X} \to V$ parametrized by θ and rank-1 elements such that $\tilde{\theta} = (\theta, \ell^1, \ldots, \ell^N)$ and apply the resulting LS2T map $\tilde{\Phi}_{\tilde{\theta}}$:

$$\mathsf{Seq}(\mathcal{X}) \to \mathsf{Seq}(\mathbb{R}^N), \quad \mathbf{x} \mapsto \big(\tilde{\Phi}_{\tilde{\theta}}(\mathbf{x}_1), \tilde{\Phi}_{\tilde{\theta}}(\mathbf{x}_1, \mathbf{x}_2), \ldots, \tilde{\Phi}_{\tilde{\theta}}(\mathbf{x}_1, \ldots, \mathbf{x}_L)\big). \tag{3.5}$$

Note that the cost of computing the expanding window seq2seq transform in (3.5) is no more expensive than computing $\tilde{\Phi}_{\tilde{\theta}}(\mathbf{x}_1, \ldots, \mathbf{x}_L)$ itself due to the recursive nature of our algorithms, for further details see Appendix 3.A.

Deep seq2seq transforms. Inspired by the empirical successes of deep RNNs [105, 104, 254], we iterate the transformation (3.5) D-times:

$$\mathsf{Seq}(\mathcal{X}) \to \mathsf{Seq}(\mathbb{R}^{N_1}) \to \mathsf{Seq}(\mathbb{R}^{N_2}) \to \cdots \to \mathsf{Seq}(\mathbb{R}^{N_D}).$$

Each of these mappings $\mathsf{Seq}(\mathbb{R}^{N_i}) \to \mathsf{Seq}(\mathbb{R}^{N_{i+1}})$ is parametrized by the parameters $\tilde{\theta}_i$ of a static feature map ϕ_{θ_i} and a linear map $\mathcal{L}_i$ specified by N_i rank-1 elements of $\mathsf{T}((V))$; these parameters are collectively denoted by $\tilde{\theta}_i = (\theta_i, \ell_i^1, \ldots, \ell_i^{N_i})$. Evaluating the final sequence in $\mathsf{Seq}(\mathbb{R}^{N_D})$ at the last observation-time $t = L$, we get the deep LS2T map with depth-D

$$\tilde{\Phi}_{\tilde{\theta}_1, \ldots, \tilde{\theta}_D} : \mathsf{Seq}(\mathcal{X}) \to \mathbb{R}^{n_D}.$$

Making precise how the stacking of such low-rank seq2seq transformations approximates general functions requires more tools from algebra, and we provide a rigorous quantitative statement in [263, App. C]. Here, we just appeal to the analogy made with adding depth in neural networks mentioned earlier and empirically validate this in our experiments in Section 3.4.

3.4 Building neural networks with LS2T layers

The Seq2Tens map Φ built from a static feature map ϕ is universal if ϕ is universal, Theorem 3.1. NNs form a flexible class of universal feature maps with strong empirical success for data in $\mathcal{X} = \mathbb{R}^d$, and thus make a natural choice for ϕ. Combined with standard deep learning constructions, the framework of Sections 3.2 and 3.3 can build modular and expressive layers for sequences.

Neural LS2T layers. The simplest choice among many is to use as static feature map $\phi : \mathcal{X} = \mathbb{R}^d \to \mathbb{R}^h$ a feedforward network with depth-P, $\phi = \phi_P \circ \cdots \circ \phi_1$ where $\phi_j(\mathbf{x}) = \sigma(\mathbf{W}_j\mathbf{x} + \mathbf{b}_j)$ for $\mathbf{W}_j \in \mathbb{R}^{h\times d}$, $\mathbf{b}_j \in \mathbb{R}^h$. We can then lift this to a map $\varphi : \mathbb{R}^d \to \mathsf{T}((\mathbb{R}^h))$ as prescribed in (3.2). Hence, the resulting LS2T layer $\mathbf{x} \mapsto (\tilde{\Phi}_{\tilde{\theta}}(\mathbf{x}_1, \ldots, \mathbf{x}_i))_{i=1,\ldots,L}$ is a seq2seq transform $\mathsf{Seq}(\mathbb{R}^d) \to \mathsf{Seq}\mathbb{R}^h$ that is parametrized by $\tilde{\theta} = (\mathbf{W}_1, \mathbf{b}_1, \ldots, \mathbf{W}_P, \mathbf{b}_P, \ell_1^1, \ldots, \ell_1^{N_1})$.

Bidirectional LS2T layers. The transformation in (3.5) is completely causal in the sense that each step of the output sequence depends on the past. For generative models, it can behove us to make the output depend on both past and future, see [104, 14, 168]. Similarly to bidirectional RNNs [235, 106], we may achieve this by defining a bidirectional layer,

$$\tilde{\Phi}^{\mathrm{b}}_{(\tilde{\theta}_1,\tilde{\theta}_2)}(\mathbf{x}) : \mathsf{Seq}(\mathbb{R}^d) \to \mathsf{Seq}(\mathbb{R}^{N+N'}), \quad \mathbf{x} \mapsto (\tilde{\Phi}_{\tilde{\theta}_1}(\mathbf{x}_1, \ldots, \mathbf{x}_i), \tilde{\Phi}_{\tilde{\theta}_2}(\mathbf{x}_i, \ldots, \mathbf{x}_L))_{i=1}^L.$$

The sequential nature is kept intact by making the distinction between what classifies as past (the first N coordinates) and future (the last N' coordinates) information. This amounts to having a form of precognition in the model, and has been applied in e.g. dynamics generation [168], machine translation [252], and speech processing [104].

Convolutions and LS2T. We motivate to replace the time-distributed feedforward layers proposed in the paragraph above by temporal convolutions (CNN) instead. Although theory only requires the "preprocessing" layer to be a static feature map, we find that it is beneficial to capture some of the sequential information as well, e.g. using CNNs or RNNs. From a mathematical point of view, CNNs are a straightforward extension since they can be interpreted as time-distributed feedforward layers applied to the input sequence augmented with a $p \in \mathbb{N}$ number of its lags for CNN kernel size p (see [263, App. D.1] for further discussion).

In the following, we precede our deep LS2T blocks by one or more CNN layers. Intuitively, CNNs and LS2Ts are similar in that both transformations operate on subsequences of their input sequence. The main difference between the two lies in that *CNNs operate on contiguous subsequences*, and therefore, capture local, short-range nonlinear interactions between timesteps;

while LS2T*s (*(3.4)*) use all non-contiguous subsequences*, and hence, learn global, long-range interactions in time. This observation motivates that the inductive biases of the two types of layers (local/global time-interactions) are highly complementary in nature, and we suggest that the improvement in the experiments on the models containing vanilla CNN blocks are due to this.

3.5 Experiments

We demonstrate the modularity and flexibility of the above LS2T and its variants by applying it to (i) multivariate time series classification, (ii) mortality prediction in healthcare, (iii) generative modelling of sequential data. In all cases, we take a strong baseline model (FCN and GP-VAE, as detailed below) and upgrade it with LS2T layers. As Thm. 3.1 requires the Seq2Tens layers to be preceded by at least a static feature map, we expect these layers to perform best as an add-on on top of other models, which however can be quite simple, such as a CNN. The additional computation time is negligible (in fact, for FCN it allows to reduce the number of parameters significantly, while retaining performance), but it can yield improvements. This is remarkable, since the original models are already state-of-the-art on well-established benchmarks.

3.5.1 Multivariate time series classification

As the first task, we consider multivariate time series classification (TSC) on an archive of benchmark datasets collected by [17]. Numerous previous publications report results on this archive, which makes it possible to compare against several well-performing competitor methods from the TSC community. These baselines are detailed in [263, App. E.1]. This archive was also considered in a recent popular survey paper on DL for TSC [128], from where we borrow the two best performing models as deep learning baselines: FCN and ResNet. The FCN is a fully convolutional network which stacks 3 convolutional layers of kernel sizes $(8, 5, 3)$ and filters $(128, 256, 128)$ followed by a global average pooling (GAP) layer, hence employing global parameter sharing. We refer to this model as $\texttt{FCN}_{128}$. The ResNet is a residual network stacking 3 FCN blocks of various widths with skip-connections in between [113] and a final GAP layer.

The FCN is an interesting model to upgrade with LS2T layers, since the LS2T also employs parameter sharing across the sequence length, and as noted previously, convolutions are only able to learn local interactions in time, that in particular makes them ill-suited to picking up on long-range interactions, which is exactly where the LS2T can provide improvements. As our models, we consider three simple architectures: (i) $\texttt{LS2T}^3_{64}$ stacks 3 LS2T layers of order-2 and width-64; (ii) $\texttt{FCN}_{64}$-$\texttt{LS2T}^3_{64}$ precedes the $\texttt{LS2T}^3_{64}$ block by an $\texttt{FCN}_{64}$ block; a downsized version of $\texttt{FCN}_{128}$; (iii) $\texttt{FCN}_{128}$-$\texttt{LS2T}^3_{64}$ uses the full $\texttt{FCN}_{128}$ and follows it by a $\texttt{LS2T}^3_{64}$ block as before. Also,

Table 3.1: Posterior probabilities given by a Bayesian signed-rank test comparison of the proposed methods against the baselines. {>}, {<}, {=} refer to the respective events that the row method is better, the column method is better, or that they are equivalent.

MODEL	$\texttt{LS2T}^3_{64}$			$\texttt{FCN}_{64}\text{-}\texttt{LS2T}^3_{64}$			$\texttt{FCN}_{128}\text{-}\texttt{LS2T}^3_{64}$		
	$p(>)$	$p(=)$	$p(<)$	$p(>)$	$p(=)$	$p(<)$	$p(>)$	$p(=)$	$p(<)$
SMTS [18]	0.180	0.000	**0.820**	0.010	0.000	**0.990**	0.008	0.000	**0.992**
LPS [19]	0.191	0.002	**0.807**	0.012	0.001	**0.987**	0.006	0.001	**0.993**
mvARF [269]	0.011	0.140	**0.849**	0.000	0.126	**0.874**	0.000	0.088	**0.912**
DTW [225]	0.033	0.000	**0.967**	0.001	0.000	**0.999**	0.000	0.000	**1.000**
ARKernel [59]	0.100	0.097	**0.803**	0.000	0.021	**0.979**	0.000	0.015	**0.985**
gRSF [137]	0.481	0.011	**0.508**	0.028	0.013	**0.960**	0.022	0.013	**0.965**
MUSE [232]	0.405	0.128	**0.467**	0.001	0.074	**0.925**	0.001	0.077	**0.922**
MLSTMFCN [136]	**0.916**	0.043	0.041	0.123	0.071	**0.807**	0.055	0.110	**0.835**
$\texttt{FCN}_{128}$ [284]	**0.998**	0.002	0.000	0.363	0.186	**0.451**	0.169	0.011	**0.820**
ResNet [284]	**0.998**	0.002	0.001	0.056	0.240	**0.704**	0.016	0.048	**0.935**
$\texttt{LS2T}^3_{64}$	-	-	-	0.000	0.001	**0.999**	0.000	0.001	**0.999**
$\texttt{FCN}_{64}\text{-}\texttt{LS2T}^3_{64}$	**0.999**	0.001	0.000	-	-	-	0.020	0.387	**0.593**

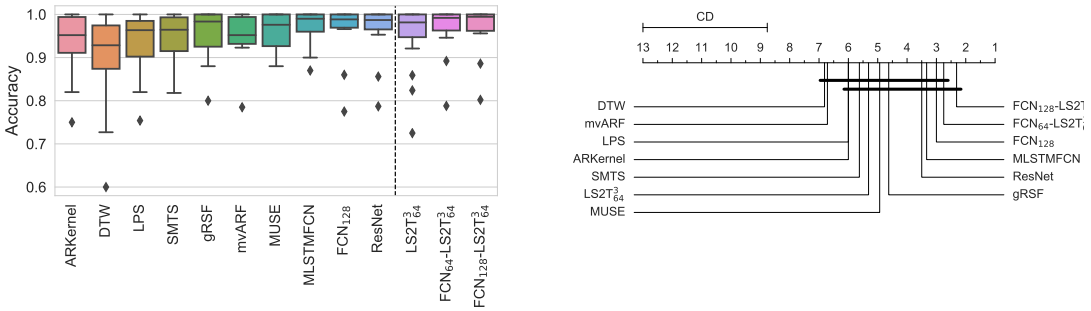

Figure 3.1: Box-plot of classification accuracies (left) and critical difference diagram (right).

both FCN-LS2T models employ skip-connections from the input to the LS2T block and from the FCN to the classification layer, allowing for the LS2T to directly see the input, and for the FCN to directly affect the final prediction. These hyperparameters were only subject to hand-tuning on a subset of the datasets, and the values we considered were $H, N \in \{32, 64, 128\}$, $M \in \{2, 3, 4\}$ and $D \in \{1, 2, 3\}$, where $H, N \in \mathbb{N}$ is the FCN and LS2T width, resp., while $M \in \mathbb{N}$ is the LS2T order and $D \in \mathbb{N}$ is the LS2T depth. We also employ techniques such as time-embeddings [174], sequence differencing and batch normalization, see [263, App. D.1]; [263, App. E.1] for further details on the experiment and Figure 2 in there for a visualization of the architectures.

Results. The full table of results is available in Table 3.6 in Appendix 3.B, where for the models that we trained ourselves, i.e. $\texttt{FCN}_{128}$, ResNet, $\texttt{LS2T}^3_{64}$, $\texttt{FCN}_{64}\text{-}\texttt{LS2T}^3_{64}$, $\texttt{FCN}_{128}\text{-}\texttt{LS2T}^3_{64}$, we report the mean and standard deviation of test set accuracies over 5 model trains. Fig-

Table 3.2: Average ranks of classifier accuracies across data domains with the best and second best highlighted for each row in **bold** and *italic*, respectively.

DOMAIN	ARKernel	DTW	LPS	SMTS	gRSF	mvARF	MUSE	MLSTMFCN	$\texttt{FCN}_{128}$	ResNet	$\texttt{LS2T}^{3}_{64}$	$\texttt{FCN}_{64}\text{-}\texttt{LS2T}^{3}_{64}$	$\texttt{FCN}_{128}\text{-}\texttt{LS2T}^{3}_{64}$
HANDWRITING	5.00	4.75	5.50	4.00	2.75	5.50	5.75	*2.00*	*2.00*	2.75	3.50	*2.00*	**1.75**
MOTION	5.33	7.50	4.67	5.67	4.50	4.67	5.00	3.50	**1.83**	2.17	4.17	2.33	*2.00*
SENSOR	8.67	6.00	6.50	5.50	4.00	11.50	**3.33**	5.00	5.50	6.00	8.25	4.25	*3.75*
SPEECH	6.00	10.50	10.00	9.00	10.00	10.50	5.50	3.00	3.50	4.00	6.50	*2.50*	**1.50**

ure 3.1 depicts the box-plot distributions of classification accuracies and the corresponding critical difference (CD) diagram. The CD diagram depicts the mean ranks of each method averaged over datasets with a calculated CD region using the Nemenyi test [198] for an alpha value of $\alpha = 0.1$. For the Bayesian signed-rank test [20], we used the implementation from https://github.com/janezd/baycomp, and the posterior probabilities are compared in Table 3.1. The posterior distributions themselves are visualized on Figure 4 [263, App. E.1]. The region of practical equivalence (rope) was set to $\texttt{rope} = 1 \times 10^{-3}$, that is, two accuracies were practically equivalent if they are at most the given distance from each other.

The results of the signed-rank test indicate that while $\texttt{LS2T}^{64}_{3}$ is only better with high probability ($p \geq 0.8$) than 5 of the baselines, $\texttt{FCN}_{64}\text{-}\texttt{LS2T}^{64}_{3}$ performs better than all except $\texttt{FCN}_{128}$ and ResNet with high probability ($p \geq 0.8$), while $\texttt{FCN}_{128}\text{-}\texttt{LS2T}^{64}_{3}$ performs better than *all baselines* with high probability ($p \geq 0.8$). Note however that $\texttt{FCN}_{64}\text{-}\texttt{LS2T}^{64}_{3}$ *has fewer parameters than* $\texttt{FCN}_{128}$ *by around* 60%, see Table 6 in [263, App. E.1], while providing comparable or slightly improved performance. In conclusion, we have verified that $\texttt{LS2T}^{64}_{3}$ already performs well on some classification tasks, and by preceding it with an FCN, its performance is elevated to outperforming all baselines. This observation is not surprising, as Thm. 3.1 warrants, the strength of the LS2T layer shines when it is preceded by state-space nonlinearities. Hence, we expect LS2Ts to find their best use as a modular building block being part of larger models. We also suggest the improvement on the vanilla FCN composed of CNN blocks is due to the LS2T enhancing the model's ability to pick up long-range interactions, see the discussion at the end of Section 3.4.

Domain analysis. We provide a breakdown of classifier performance in Table 3.2 across four distinct domains: handwriting, motion, sensor, speech. We observe the following (1) for handwriting datasets, both variants of $\texttt{FCN}_{128}\text{-}\texttt{LS2T}^{64}_{3}$ provides the best performance with $\texttt{FCN}_{64}\text{-}\texttt{LS2T}^{64}_{3}$, $\texttt{FCN}_{128}$ and MLSTMFCN coming in close second, and ResNet in third; (2) however, on motion datasets $\texttt{FCN}_{128}$ takes first place and $\texttt{FCN}_{128}\text{-}\texttt{LS2T}^{64}_{3}$ in close second and $\texttt{FCN}_{64}\text{-}\texttt{LS2T}^{64}_{3}$ in third place; (3) on sensor datasets, surprisingly, the deep learning baselines underperform with MUSE taking first place and $\texttt{FCN}_{128}\text{-}\texttt{LS2T}^{64}_{3}$ coming in second and $\texttt{FCN}_{64}\text{-}\texttt{LS2T}^{64}_{3}$ in third place, while $\texttt{FCN}_{128}$, MLSTMFCN, and ResNet underperform; (4) finally, on speech datasets, the models

Table 3.3: Comparison of FCN-LS2T and FCN on PHYSIONET2012 with baselines from [123].

MODEL	ACCURACY	AUPRC	AUROC
FCN-LS2T	**84.1 ± 1.6**	**53.9 ± 0.5**	85.6 ± 0.5
FCN	80.7 ± 1.7	52.8 ± 1.3	85.6 ± 0.2
GRU-D	80.0 ± 2.9	*53.7 ± 0.9*	**86.3 ± 0.3**
GRU-Simple	82.2 ± 0.2	42.2 ± 0.6	80.8 ± 1.1
IP-Nets	79.4 ± 0.3	51.0 ± 0.6	*86.0 ± 0.2*
Phased-LSTM	76.8 ± 5.2	38.7 ± 1.5	79.0 ± 1.0
Transformer	*83.7 ± 3.5*	52.8 ± 2.2	**86.3 ± 0.8**
Latent-ODE	76.0 ± 0.1	50.7 ± 1.7	85.7 ± 0.6
SeFT-Attn.	75.3 ± 3.5	52.4 ± 1.1	85.1 ± 0.4

FCN_{128}-LS2T_3^{64} and FCN_{64}-LS2T_3^{64} come in first and second place respectively with MLSTMFCN coming in third. Overall, we observe that the models FCN_{128}-LS2T_3^{64} and FCN_{64}-LS2T_3^{64} perform well across each data domain, coming in top 3 at all times, while the vanilla deep learning baselines without LS2T layers, FCN_{128}, MLSTMFCN and ResNet only work well on certain data modalities, but underperform on others, such as sensor readings as discussed above.

3.5.2 Mortality prediction

We consider the PHYSIONET2012 challenge dataset [99] for mortality prediction, which is a case of medical TSC as the task is to predict in-hospital mortality of patients after their admission to the ICU. This is a difficult ML task due to missingness in the data, low signal-to-noise ratio (SNR), and imbalanced class distributions with a prevalence ratio of around 14%. We extend the experiments conducted in [123], which we also use as very strong baselines. Under the same experimental setting, we train two models: FCN-LS2T as ours and the FCN as another baseline. For both models, we conduct a random search for all hyperparameters with 20 samples from a pre-specified search space, and the setting with best validation performance is used for model evaluation on the test set over 5 independent model trains, exactly the same way as it was done in [123]. We preprocess the data using the same method as in [38, eq. (9)] and additionally handle static features by tiling them along the time axis and adding them as extra coordinates. We additionally introduce in both models a SpatialDropout1D layer after all CNN and LS2T layers with the same tunable dropout rate to mitigate the low SNR of the dataset.

Results. Table 3.3 compares the performance of FCN-LS2T with that of FCN and the results from [123] on 3 metrics: (1) ACCURACY, (2) area under the precision-recall curve (AUPRC), (3) area under the ROC curve (AUROC). We can observe that *FCN-LS2T takes on average first place according*

to both ACCURACY *and* AUPRC*, outperforming* FCN *and all* SOTA *methods*, e.g. Transformer [275], GRU-D [38], SeFT [123], and also being competitive in terms of AUROC. This is very promising, and it suggests that LS2T layers might be particularly well-suited to complex and heterogenous datasets, such as medical time series, since the FCN-LS2T models significantly improved accuracy on ECG as well, another medical dataset in the previous experiment.

3.5.3 Generating sequential data

Finally, we demonstrate on sequential data imputation for time series and video that LS2Ts also provide good representations of sequences in generative models.

The GP-VAE model. In this experiment, we take as base model the recent GP-VAE [83], that provides state-of-the-art results for probabilistic sequential data imputation. The GP-VAE is essentially based on the HI-VAE [197] for handling missing data in variational autoencoders (VAEs) [144] adapted to the handling of time series data by the use of a Gaussian process (GP) prior [216] across time in the latent sequence space. The in-depth description of GPVAE can be found in [263, App. E.3].We extend the experiments conducted in [83], and we make one simple change to the GP-VAE architecture without changing any other hyperparameters or aspects: we introduce a single bidirectional LS2T layer (B-LS2T) into the encoder network that is used in the amortized representation of the means and covariances of the variational posterior. The B-LS2T layer is preceded by a time-embedding and differencing block, and succeeded by channel flattening and layer normalization as depicted on Figure 5 in [263, App. E.3]. The idea behind this experiment is to see if we can improve the performance of a highly complicated model that is composed of many interacting submodels, by an educated introduction of LS2T layers.

Results. To make the comparison, we ceteris paribus re-ran all experiments the authors originally included in their paper [83], which are imputation of HEALING MNIST, SPRITES, and PHYSIONET2012. The results are in Table 3.4, which report the same metrics as used in [83], i.e. negative log-likelihood (NLL, lower is better), mean squared error (MSE, lower is better) on test sets, and downstream classification performance of a linear classifier (AUROC, higher is better). For all other models beside our GP-VAE (B-LS2T), the results were borrowed from [83]. We observe that simply adding the B-LS2T layer improved the result in almost all cases, except for SPRITES, where the GP-VAE already achieved a very low MSE score. Additionally, when comparing GP-VAE to BRITS on PHYSIONET2012, the authors argue that although the BRITS achieves a higher AUROC score, the GP-VAE should not be disregarded as it fits a generative model to the data that enjoys the usual Bayesian benefits of predicting distributions instead of point predictions. The results display that by simply adding our layer into the architecture, we managed to elevate

Table 3.4: Performance comparison of GP-VAE (B-LS2T) with the baseline methods

METHOD	HMNIST			SPRITES	PHYSIONET
	NLL	MSE	AUROC	MSE	AUROC
MEAN IMPUTATION	-	0.168 ± 0.000	0.938 ± 0.000	0.013 ± 0.000	0.703 ± 0.000
FORWARD IMPUTATION	-	0.177 ± 0.000	0.935 ± 0.000	0.028 ± 0.000	0.710 ± 0.000
VAE	0.599 ± 0.002	0.232 ± 0.000	0.922 ± 0.000	0.028 ± 0.000	0.677 ± 0.002
HI-VAE	0.372 ± 0.008	0.134 ± 0.003	$\mathbf{0.962 \pm 0.001}$	0.007 ± 0.000	0.686 ± 0.010
GP-VAE	0.350 ± 0.007	0.114 ± 0.002	$\mathbf{0.960 \pm 0.002}$	$\mathbf{0.002 \pm 0.000}$	0.730 ± 0.006
GP-VAE (B-LS2T)	$\mathbf{0.251 \pm 0.008}$	$\mathbf{0.092 \pm 0.003}$	$\mathbf{0.962 \pm 0.001}$	$\mathbf{0.002 \pm 0.000}$	$\mathbf{0.743 \pm 0.007}$
BRITS	-	-	-	-	$\mathbf{0.742 \pm 0.008}$

the performance of GP-VAE to the same level while retaining these same benefits. We believe the reason for the improvement is a tighter amortization gap in the variational approximation [54] achieved by increasing the expressiveness of the encoder by the LS2T allowing it to pick up on long-range interactions in time. For further discussion, see [263, App. E.3].

3.6 Related work and Summary

Related Work. The literature on tensor models in ML is vast. Related to our approach we mention pars-pro-toto Tensor Networks [51], that use classical LR decompositions, such as CP [31], Tucker [268], tensor trains [202] and tensor rings [304]; further, CNNs have been combined with LR tensor techniques [53, 151] and extended to RNNs [139]; Tensor Fusion Networks [301] and its LR variants [175, 171, 125]; tensor-based gait recognition [257]. Our main contribution to this literature is the use of the tensor algebra $\mathsf{T}((V))$ with its convolution product $\cdot$, instead of $V^{\otimes m}$ with the outer product $\otimes$ that is used in the above papers. While counter-intuitive to work in a larger space $\mathsf{T}((V))$, the additional algebra structure of $(\mathsf{T}((V)), \cdot)$ is the main reason for the nice properties of Φ (*universality, making sequences of arbitrary length comparable, convergence in the continuous time limit*, see [263, App. B], which we believe are in turn the main reason for the *strong benchmark performance*. Stacked LR sequence transforms allow to exploit this rich algebraic structure with little computational overhead. Another related literature are path signatures in ML [178, 42, 101, 22, 266]. These arise as special case of Seq2Tens [263, App. B] and our main contribution to this literature is that Seq2Tens resolves a well-known computational bottleneck in this literature since it *never needs to compute and store a signature*, instead it *directly and efficiently learns the functional of the signature*.

Summary. We used a classical non-commutative structure to construct a feature map for sequences of arbitrary length. By stacking sequence transforms we turned this into scalable and

modular NN layers for sequence data. The main novelty is the use of the tensor algebra $\mathsf{T}((V))$ constructed from the static feature space V. While tensor algebras are classical in mathematics, their use in ML seems novel and underexplored. We would like to re-emphasize that $(\mathsf{T}((V)), \cdot)$ is not a mysterious abstract space: if you know the outer tensor product $\otimes$ then you can easily switch to the tensor convolution product $\cdot$ by taking sums of outer tensor products, as defined in (1.2). As our experiments show, the benefits of this algebraic structure are not just theoretical but can significantly elevate performance of already strong-performing models.

3.A Computational details

Here we give further information on the implementation of LS2T layers detailed in the main text. For simplicity, we fix the state-space of sequences to be $V = \mathbb{R}^d$ from here onwards.

3.A.1 Recursive computations

Next, we show how the computation of the maps Φ_m and $\Phi_{m,\theta}$ can be formulated as a joint recursion over the tensor levels and the sequence itself. Since Φ_m is given by a summation over all noncontiguous length-m subsequences with non-repetitions of a sequence $\mathbf{x} \in \mathsf{Seq}\mathbb{R}^d$, simple reasoning shows that Φ_m obeys the recursion across m and time for $2 \le l \le L$ and $1 \le m$

$$\Phi_m(\mathbf{x}_1, \dots \mathbf{x}_l) = \Phi_m(\mathbf{x}_1, \dots \mathbf{x}_{l-1}) + \Phi_{m-1}(\mathbf{x}_1, \dots, \mathbf{x}_{l-1}) \otimes \mathbf{x}_l, \tag{3.6}$$

with the initial conditions $\Phi_0 \equiv 1$, $\Phi_1(\mathbf{x}_1) = \mathbf{x}_1$ and $\Phi_m(\mathbf{x}_1) = \mathbf{0}$ for $m \ge 2$.

Let $\ell = (\ell_m)_{m \ge 0} \in T(\mathbb{R}^d)$ be a sequence of rank-1 tensors with $\ell_m = \mathbf{z}_{m,1} \otimes \cdots \otimes \mathbf{z}_{m,m} \in (\mathbb{R}^d)^{\otimes m}$ a rank-1 tensor of degree-m. Then, $\langle \ell_m, \Phi_m(\mathbf{x}) \rangle$ may be computed analogously to (3.6) using

$$\begin{aligned}
\langle \ell_m, \Phi_m(\mathbf{x}_1, \dots, \mathbf{x}_l) \rangle =& \langle \mathbf{z}_{m,1} \otimes \cdots \otimes \mathbf{z}_{m,m}, \Phi_m(\mathbf{x}_1, \dots, \mathbf{x}_l) \rangle \\
=& \langle \mathbf{z}_{m,1} \otimes \cdots \otimes \mathbf{z}_{m,m}, \Phi_m(\mathbf{x}_1, \dots, \mathbf{x}_{l-1}) \rangle \\
&+ \langle \mathbf{z}_{m,1} \otimes \cdots \otimes \mathbf{z}_{m,m-1}, \Phi_{m-1}(\mathbf{x}_1, \dots, \mathbf{x}_{l-1}) \rangle \langle \mathbf{z}_{m,m}, \mathbf{x}_l \rangle \\
=& \langle \ell_m, \Phi_m(\mathbf{x}_1, \dots \mathbf{x}_{l-1}) \rangle && (3.7) \\
&+ \langle \mathbf{z}_{m,1} \otimes \cdots \otimes \mathbf{z}_{m,m-1}, \Phi_{m-1}(\mathbf{x}_1, \dots, \mathbf{x}_{l-1}) \rangle \langle \mathbf{z}_{m,m}, \mathbf{x}_l \rangle && (3.8)
\end{aligned}$$

and the initial conditions can be rewritten as the identities $\langle z_{0,0}, \Phi_0 \rangle = 1$, $\langle \mathbf{z}_{m,1}, \Phi_1(\mathbf{x}) \rangle = \langle \mathbf{z}_{m,1}, \mathbf{x} \rangle$ and $\langle \mathbf{z}_{m,1} \otimes \cdots \otimes \mathbf{z}_{m,m}, \Phi_m(\mathbf{x}_1) \rangle = \mathbf{0}$ for $2 \le m$.

A slight inefficiency of the previous recursion given in (3.7), (3.8) is that one generally cannot substitute $\langle \ell_{m-1}, \Phi_m(\mathbf{x}_1, \dots, \mathbf{x}_{l-1})\rangle$ for the $\langle \mathbf{z}_{m,1} \otimes \cdots \otimes \mathbf{z}_{m,m-1}, \Phi_m(\mathbf{x}_1, \dots, \mathbf{x}_{l-1})\rangle\rangle$ term in (3.8), since $\ell_{m-1} \neq \mathbf{z}_{m,1} \otimes \cdots \otimes \mathbf{z}_{m,m-1}$ generally. This means that to construct the degree-m linear functional $\langle \ell_m, \Phi_m(\mathbf{x}_1, \dots, \mathbf{x}_l)\rangle$, one has to start from scratch by first constructing the degree-1 term $\langle \mathbf{z}_{m,1}, \Phi_1\rangle$ first, then the degree-2 term $\langle \mathbf{z}_{m,1} \otimes \mathbf{z}_{m,2}, \Phi_2\rangle$, and so forth. This further means in terms of complexities that while (3.6) has linear complexity in the largest value of m, henceforth denoted by $M \in \mathbb{N}$, (3.7), (3.8) has a quadratic complexity in M due to the non-recursiveness of the rank-1 tensors $(\ell_m)_m = (\mathbf{z}_{m,1} \otimes \cdots \otimes \mathbf{z}_{m,m})_m$.

The previous observation indicates that an even more memory and time efficient recursion can be devised by parametrizing the rank-1 tensors $(\ell_m)_m$ in a recursive way as follows: let $\ell_1 = \mathbf{z}_1 \in \mathbb{R}^d$ and define $\ell_m = \ell_{m-1} \otimes \mathbf{z}_m \in (\mathbb{R}^d)^{\otimes m}$ for $2 \leq m$, i.e. $\ell_m = \mathbf{z}_1 \otimes \cdots \otimes \mathbf{z}_m$ for $\{\mathbf{z}_1, \dots \mathbf{z}_m\} \subset \mathbb{R}^d$. This parameterization indeed allows to substitute ℓ_{m-1} in (3.8) so

$$\langle \ell_m, \Phi_m(\mathbf{x}_1, \dots, \mathbf{x}_l)\rangle = \langle \ell_m, \Phi_m(\mathbf{x}_1, \dots, \mathbf{x}_{l-1}\rangle + \langle \ell_{m-1}, \Phi_{m-1}(\mathbf{x}_1, \dots, \mathbf{x}_{l-1})\rangle \langle \mathbf{z}_m, \mathbf{x}_l\rangle, \quad (3.9)$$

and hence, due to the recursion across m for both ℓ_m and Φ_m, it is now linear in the maximal value of m, denoted by $M \in \mathbb{N}$. This results in a less flexible, but more efficient LS2T, due to the additional added recursivity constraint on the rank-1 weight tensors. We refer to this version as the *recursive variant*, while to the non-recursive construction as the *independent variant*.

Next, we show how the previous computations can be rewritten as a simple RNN-like recursion. For simplicity, we consider the recursive formulation, but the independent variant can also be formulated as such with a larger latent state size. Let $(\ell^j)_{j=1,\dots,n}$ be $n \in \mathbb{N}$ different rank-1 recursive weight tensors, i.e. $\ell^j = (\ell^j_m)_{m \geq 0}$, $\ell^j_m = \mathbf{z}^j_1 \otimes \dots \mathbf{z}^j_m$ for $\mathbf{z}^j_m \in \mathbb{R}^d$, $m \geq 0$ and $j = 1, \dots, n$. Also, denote $h^j_{m,i} := \langle \ell^j_m, \Phi_m(\mathbf{x}_1, \dots, \mathbf{x}_i)\rangle \in \mathbb{R}$, a scalar corresponding to the output of the jth linear functional on the mth tensor level for the sequence $(\mathbf{x}_1, \dots, \mathbf{x}_i)$. We collect all such functionals for given m and i into $\mathbf{h}_{m,i} := (h^1_{m,i}, \dots h^n_{m,i}) \in \mathbb{R}^n$, i.e. $\mathbf{h}_{m,i} = \tilde{\Phi}_{m,\tilde{\theta}}(\mathbf{x}_1, \dots, \mathbf{x}_i)$.

Additionally, we collect all weight vectors $\mathbf{z}^j_m \in \mathbb{R}^d$ for a given $m \in \mathbb{N}$ into the matrix $\mathbf{Z}_m := (\mathbf{z}^1_m, \dots, \mathbf{z}^n_m)^\top \in \mathbb{R}^{n \times d}$. Then, we may write the following vectorized version of (3.9):

$$\begin{aligned} \mathbf{h}_{1,i} &= \mathbf{h}_{1,i-1} + \mathbf{Z}_1 \mathbf{x}_i, \\ \mathbf{h}_{m,i} &= \mathbf{h}_{m,i-1} + \mathbf{h}_{m-1,i-1} \odot \mathbf{Z}_m \mathbf{x}_i \quad \text{for } m \geq 2, \end{aligned}$$

with the initial conditions $\mathbf{h}_{m,0} = \mathbf{0} \in \mathbb{R}^n$ for all $m \geq 1$, and $\odot$ denoting the Hadamard product.

Algorithm 5 Computing the LS2T layer with independent tensors across levels

Input: Sequences $(\mathbf{x}^j)_{j=1,\dots,n_\mathbf{x}} = (\mathbf{x}_1^j, \dots, \mathbf{x}_L^j)_{j=1,\dots,n_\mathbf{x}} \subset \mathsf{Seq}(\mathbb{R}^d)$,
rank-1 tensors $(\ell^k)_{k=1,\dots,n_\ell} = (\mathbf{z}_{m,1}^k \otimes \cdots \otimes \mathbf{z}_{m,m}^k)_{m=1,\dots,M}^{k=1,\dots,n_\ell} \subset T(\mathbb{R}^d)$, LS2T order $M \in \mathbb{N}$
Compute $A[m,i,j,l,k] \leftarrow \langle \mathbf{z}_{m,k}^j, \mathbf{x}_l^i \rangle$ for $m \in \{1,\dots,M\}$, $i \in \{1,\dots,n_\mathbf{x}\}$, $j \in \{1,\dots,n_\ell\}$, $l \in \{1,\dots,L\}$ and $k \in \{1,\dots,m\}$
for $m = 1$ **to** M **do**
    Assign $R \leftarrow A[m,:,:,:,1]$
    for $k = 2$ **to** m **do**
        Iterate $R \leftarrow A[m,:,:,:,k] \odot R[:,:,\boxplus + 1]$
    end for
    Save $Y_m \leftarrow \cdot R[:,:,\boxplus]$
end for
Output: Sequences $(Y_1, \dots, Y_M)$ each of shape $(n_\mathbf{x} \times L \times n_\ell)$

Algorithm 6 Computing the LS2T layer with recursive tensors across levels

Input: Sequences $(\mathbf{x}^j)_{j=1,\dots,n_\mathbf{x}} = (\mathbf{x}_1^j, \dots, \mathbf{x}_L^j)_{j=1,\dots,n_\mathbf{x}} \subset \mathsf{Seq}(\mathbb{R}^d)$,
rank-1 tensors $(\ell^k)_{k=1,\dots,n_\ell} = (\mathbf{z}_1^k \otimes \cdots \otimes \mathbf{z}_m^k)_{m=1,\dots,M}^{k=1,\dots,n_\ell} \subset T(\mathbb{R}^d)$, LS2T order $M \in \mathbb{N}$
Compute $A[m,i,j,l] \leftarrow \langle \mathbf{z}_m^j, \mathbf{x}_l^i \rangle$ for $m \in \{1,\dots,M\}$, $i \in \{1,\dots,n_\mathbf{x}\}$, $j \in \{1,\dots,n_\ell\}$ and $l \in \{1,\dots,L\}$
Assign $R \leftarrow A[1,:,:,:]$
Save $Y_1 \leftarrow R[:,:,\boxplus]$
for $m = 2$ **to** M **do**
    Update $R \leftarrow A[m,:,:,:] \odot R[:,:,\boxplus + 1]$
    Save $Y_m \leftarrow R[:,:,\boxplus]$
end for
Output: Sequences $(Y_1, \dots, Y_M)$ each of shape $(n_\mathbf{x} \times L \times n_\ell)$

3.A.2 Algorithms

We have shown previously that one may compute $\Phi_\theta(\mathbf{x}_1, \dots, \mathbf{x}_i) = (\Phi_{m,\theta}(\mathbf{x}_1, \dots, \mathbf{x}_i))_{m \geq 0}$ recursively in a vectorized way for a given sequence $(\mathbf{x}_1, \dots, \mathbf{x}_i) \in \mathsf{Seq}(\mathbb{R}^d)$. Now, in Algorithms 5 and 6, we additionally show how to further vectorize the previous computations across time and the batch. For this purpose, let $(\mathbf{x}^j)_{j=1,\dots,n_\mathbf{X}} \subset \mathsf{Seq}(\mathbb{R}^d)$ be $n_\mathbf{X} \in \mathbb{N}$ sequences in $\mathbb{R}^d$ and $(\ell^k)_{k=1,\dots,n_\ell} \subset T(\mathbb{R}^d)$ be n_ℓ be rank-1 tensors in $T(\mathbb{R}^d)$.

3.A.3 Complexity analysis

We give a complexity analysis of Algorithms 5 and 6. Inspection of Algorithm 5 says that it has $O(M^2 \cdot n_\mathbf{x} \cdot L \cdot n_\ell \cdot d)$ complexity in both time and memory with an additional memory cost of storing the $O(M^2 \cdot n_\ell \cdot d)$ number of parameters, the rank-1 elements $(\ell_m^k)_m$, which are stored in terms of their components $\mathbf{z}_{m,j}^k \in \mathbb{R}^d$. In contrast, Algorithm 5 has a time and memory cost of $O(M \cdot n_\mathbf{x} \cdot L \cdot n_\ell \cdot d)$, thus linear in M, and the recursive rank-1 elements are now only an additional $O(M \cdot n_\ell \cdot d)$ number of parameters.

Additionally to the big-O bounds on complexities, another important question is how well the computations can be parallelized, which can have a larger impact on computations when e.g. running on GPUs. Observing the algorithms, we can see that the only operation over the

Table 3.5: Forward pass computation time in seconds on a Gefore 2080Ti GPU for varying sequence length L, fixed batch size $N = 32$, state-space dimension $d = 64$ and output dimension $h = 64$.

L	Conv1D	LSTM	LS2T			LS2T-R		
			$M = 2$	$M = 6$	$M = 10$	$M = 2$	$M = 6$	$M = 10$
32	8.1×10^{-4}	1.2×10^{-1}	1.7×10^{-3}	4.5×10^{-3}	9.9×10^{-3}	1.7×10^{-3}	2.5×10^{-3}	3.4×10^{-3}
64	8.5×10^{-4}	2.3×10^{-1}	1.8×10^{-3}	4.5×10^{-3}	9.9×10^{-3}	1.8×10^{-3}	2.6×10^{-3}	3.4×10^{-3}
128	9.7×10^{-4}	4.6×10^{-1}	2.1×10^{-3}	4.9×10^{-3}	1.0×10^{-2}	2.1×10^{-3}	2.9×10^{-3}	3.8×10^{-3}
256	1.1×10^{-3}	9.3×10^{-1}	2.4×10^{-3}	5.2×10^{-3}	1.1×10^{-2}	2.4×10^{-3}	3.2×10^{-3}	4.0×10^{-3}
512	1.3×10^{-3}	1.8×10^{0}	3.2×10^{-3}	6.0×10^{-3}	1.1×10^{-2}	3.0×10^{-3}	4.0×10^{-3}	4.8×10^{-3}
1024	1.9×10^{-3}	3.7×10^{0}	4.4×10^{-3}	7.1×10^{-3}	1.2×10^{-2}	4.4×10^{-3}	5.1×10^{-3}	6.0×10^{-3}

time axis is the cumsum ($\boxplus$) operation in Lines 6, 8 (Algorithm 5) and Lines 4, 6 (Algorithm 6). The cumulative sum operates recursively on the whole time axis, and it can be parallelized using a work efficient scan algorithm to run in log-length time.

To gain further intuition about what kind of performance one can expect for our LS2T layers, we benchmarked the computation time of a forward pass for varying sequence lengths and varying hyperparameters of the model. For comparison, we ran the same experiment with an LSTM layer and a Conv1D layer with a filter size of 32. The input is a batch of sequences of shape $(n_{\mathbf{X}} \times L \times d)$, while the output has shape $(n_{\mathbf{X}} \times L \times h)$, where $d \in \mathbb{N}$ is the state-space dimension of the input sequences, while $h \in \mathbb{N}$ is simply the number of channels or hidden units in the layer. For our layers, we used our own implementation in Tensorflow, while for LSTM and Conv1D, we used the Keras implementation using the Tensorflow backend.

In Table 3.5, we report the average computation time of a forward pass over 100 trials, for fixed batch size $n_{\mathbf{X}} = 32$, state-space dimension $d = 64$, output dimension $h = 64$ and varying sequence lengths $L \in \{32, 64, 128, 256, 512, 1024\}$. LS2T and LS2T-R respectively refer to the independent and recursive variants, and $M \in \mathbb{N}$ denotes the truncation degree. We can observe that while the LSTM practically scales linearly in L, the scaling of LS2T is sublinear for all practical purposes, exhibiting a growth rate that is more close to that of the Conv1D layer, that is fully parallelizable. Specifically, while the LSTM takes 3.7 seconds to make a forward pass for $L = 1024$, all variants of the LS2T layer take less time than that *by a factor of at least a 100*. Additionally, we observe that LS2T exhibits a more aggressive growth rate with respect to the parameter M due to the quadratic complexity in M (although the numbers show only linear growth), while LS2T-R scales very favourably in M as well due to the linear complexity (the results indicate a sublinear growth rate).

3.B Further results

Table 3.6: Classifier accuracies on the multivariate TSC datasets with the best and second best highlighted for each row in **bold** and *italic*, respectively.

Dataset	ARKernel	DTW	LPS	SMTS	gRSF	mvARF	MUSE	MLSTMFCN	FCN_{128}	ResNet	$LS2T^3_{64}$	FCN_{64}-$LS2T^3_{64}$	FCN_{128}-$LS2T^3_{64}$
ArabicDigits	0.988	0.908	0.971	0.964	0.975	0.952	0.992	0.990	0.995(0.001)	0.995(0.002)	0.979(0.002)	*0.996*(0.001)	**0.997**(0.001)
AUSLAN	0.918	0.727	0.754	0.947	0.955	0.934	0.970	0.950	0.979(0.003)	0.971(0.003)	0.987(0.002)	**0.996**(0.001)	*0.995*(0.001)
Char. Traj.	0.900	0.948	0.965	0.992	*0.994*	0.928	0.937	0.990	0.992(0.001)	0.985(0.002)	0.980(0.003)	0.993(0.001)	**0.995**(0.000)
CMUsubject16	**1.000**	0.930	**1.000**	*0.997*	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**(0.000)	**1.000**(0.000)	**1.000**(0.000)	**1.000**(0.000)	**1.000**(0.000)
DigitShapes	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**(0.000)	**1.000**(0.000)	**1.000**(0.000)	**1.000**(0.000)	**1.000**(0.000)
ECG	0.820	0.790	0.820	0.818	0.880	0.785	0.880	0.870	0.860(0.018)	0.856(0.010)	0.824(0.016)	**0.892**(0.015)	*0.886*(0.014)
Jap Vowels	0.984	0.962	0.951	0.969	0.800	0.959	0.976	**1.000**	0.990(0.003)	0.989(0.003)	0.984(0.005)	0.991(0.003)	*0.994*(0.003)
Kick vs Punch	0.927	0.600	0.900	0.820	**1.000**	*0.976*	**1.000**	0.900	**1.000**(0.000)	**1.000**(0.000)	**1.000**(0.000)	**1.000**(0.000)	**1.000**(0.000)
LIBRAS	0.952	0.888	0.903	0.909	0.911	0.945	0.894	**0.970**	*0.966*(0.002)	*0.966*(0.008)	0.859(0.008)	0.946(0.005)	0.956(0.008)
NetFlow	nan	0.976	0.968	0.977	0.914	nan	0.961	0.950	0.970(0.003)	0.953(0.006)	0.921(0.014)	0.962(0.006)	0.962(0.005)
PEMS	0.750	0.832	0.844	0.896	1.000	nan	nan	nan	0.775(0.019)	0.787(0.008)	0.725(0.013)	0.788(0.025)	0.802(0.017)
PenDigits	0.952	0.927	0.908	0.917	0.932	0.923	0.912	**0.970**	*0.967*(0.002)	0.963(0.001)	0.956(0.002)	0.963(0.003)	0.962(0.002)
Shapes	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**(0.000)	**1.000**(0.000)	**1.000**(0.000)	**1.000**(0.000)	**1.000**(0.000)
UWave	0.904	0.916	**0.980**	0.941	0.929	0.952	0.916	0.970	*0.979*(0.001)	0.978(0.001)	0.958(0.001)	0.975(0.002)	0.976(0.001)
Wafer	0.968	0.974	0.962	0.965	*0.992*	0.931	**0.997**	0.990	0.987(0.005)	0.989(0.002)	0.983(0.003)	0.988(0.001)	0.990(0.001)
Walk vs Run	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**(0.000)	**1.000**(0.000)	**1.000**(0.000)	**1.000**(0.000)	**1.000**(0.000)
Avg. acc.	0.938	0.899	0.933	0.945	0.955	0.949	0.962	**0.970**	0.966	0.964	0.947	0.*968*	**0.970**
Med. acc.	0.952	0.929	0.964	0.964	0.984	0.952	0.976	0.990	0.988	0.987	0.982	*0.992*	**0.994**
Sd. acc.	0.073	0.111	0.073	0.059	0.058	0.055	0.043	0.039	0.061	0.059	0.079	0.056	0.054
Avg. rank	6.000	6.812	6.000	5.625	4.625	6.714	4.933	3.333	3.000	3.500	5.312	*2.750*	**2.312**
Med. rank	6.000	8.000	6.000	6.500	2.500	8.500	4.000	3.000	*2.500*	3.000	6.500	*2.500*	**2.000**
Sd. rank	4.071	4.246	4.397	3.810	3.964	4.514	4.044	2.610	2.066	2.251	3.591	1.880	1.493

Chapter 4

Capturing Graphs by Hypo-Elliptic Diffusions

4.1 Introduction

Obtaining a latent description of the non-Euclidean structure of a graph is central to many applications. One common approach is to construct features for each node that represents the local neighborhood; pooling these node features then provides a latent description of the whole graph. A classic way to arrive at such node features is by random walks: at the given node one starts a random walk, and extracts a neighbourhood summary from its sample trajectories. We revisit this random walk construction and are inspired by two classical mathematical results:

Hypo-elliptic Laplacian. In the case of Brownian motion $B = (B_t)_{t\geq 0}$ evolving in $\mathbb{R}^n$, the quantity $u(t,x) = \mathbb{E}[f(B_t)|B_0 = x]$ solves the heat equation $\partial_t u = \Delta u$ on $[0,\infty) \times \mathbb{R}^n$, $u(0,x) = f(x)$. Seminal work of Gaveau [93] shows that if one replaces $f(B_t)$ in the expectation by a functional of the whole trajectory, $F(B_s : s \in [0,t])$, then a path-dependent heat equation can be derived using the hypo-elliptic Laplacian in place of classic Laplacian.

Tensor Algebras. A simple way to capture a sequence – for us, a sequence of nodes visited by a random walk on a graph – is to associate with each sequence element an element in an algebra[1] and multiply these algebra elements together. If the algebra multiplication is commutative, the sequential structure is lost but if it is non-commutative, this captures the order in the sequence. In fact, by using the tensor algebra, this can be done faithfully and linear functionals of this algebra correspond to functionals on the space of sequences.

[1] An algebra is a vector space where one can multiply elements; e.g. the set of $n \times n$ matrices with matrix multiplication. This multiplication can be non-commutative; e.g. $A \cdot B \neq B \cdot A$ for general matrices A, B.

We generalize these ideas from the Euclidean case of $\mathbb{R}^d$ to the non-Euclidean case of graphs. In particular, we construct node features by sampling from a random walk started at the node, but instead of averaging over end points, we average over path-dependent functions. Informally, instead of asking a random walker that started at a node, "*What do you see now?*" after k steps, we ask "*What have you seen along your way?*". The above notions from mathematics about the hypo-elliptic Laplacian and the tensor algebra allow us to formalize this in the form of a generalized graph diffusion equation and we develop algorithms that make this a scalable method.

Related Work. From the ML literature, [206, 109] popularized the combination of deep learning architectures to capture random walk histories. Such ideas have been incorporated, sometimes implicitly, into graph neural networks (GNN) [231, 26, 233, 65, 112, 15, 145] that in turn build on convolutional approaches [160, 161, 109], as well as their combination with attention or message passing [191, 276, 95], and more recent improvements [296, 193, 184, 41] that provide and improve on theoretical guarantees. Another classic approach are graph kernels, see [23] for a recent survey; in particular, the seminal paper [150] explored the connection between diffusion equations and random walkers in a kernel learning context. More recently, [39] proposed sequence kernels to capture the random walk history. Furthermore, [52] uses the signature kernel maximum mean discrepancy (MMD) [44] as a metric for trees which implicitly relies on the algebra of tensors that we use, and [200] aggregates random walk histories to derive a kernel for graphs. Moreover, the concept of network motifs [190, 236] relates to similar ideas that describe a graph by node sequences. Further, the Bethe Hessian [224] has been successfully used in spectral clustering and shares the same goal of capturing path-dependence via "deformed Laplacians", although the mathematical approach is very different to ours. Directly related to our approach is the topic of learning diffusion models [147, 37, 258, 79, 36] on graphs. While similar ideas on random walks and diffusion for graph learning have been developed by different communities, our proposed method leverages these perspectives by capturing random walk histories through a novel diffusion operation.

Our main mathematical influence is the seminal work of Gaveau [93] from the 1970s that shows how Brownian motion can be lifted into a Markov process evolving in a tensor algebra to capture path-dependence. This leads to a heat equation governed by the hypo-elliptic Laplacian. These insights had a large influence in PDE theory, see [218, 122], but it seems that their discrete counterpart on graphs has received no attention despite the well-developed literature on random walks on graphs and general non-Euclidean objects, [291, 67, 108, 274]. A key challenge to go from theory to application is handling the computational complexity. To do so, we build on ideas from [263] to design effective algorithms for the hypo-elliptic graph diffusion.

Contribution and Outline. We introduce the hypo-elliptic graph Laplacian which allows to effectively capture random walk histories through a generalized diffusion model.

- In Section 4.3, we introduce the hypo-elliptic variants of standard graph matrices such as the adjacency matrix and (normalized) graph Laplacians. These hypo-elliptic variants are formulated in terms of tensor-valued matrices rather than scalar-valued matrices, and can be manipulated using linear algebra in the same manner as the classical setting.
- The hypo-elliptic Laplacian leads to a corresponding diffusion model, and in Section 4.1, we show that the solution to this generalized diffusion equation summarizes the microscopic picture of *the entire history of random walks* and not just their location after k steps.
- This solution provides a rich description of the local neighbourhood about a node, which can either be used directly as node features or be pooled over the graph to obtain a latent description of the graph. Section 4.2 shows that these node features characterize random walks on the graph, and we provide an analogous statement for graphs in [265, App. E].
- One can solve the hypo-elliptic graph diffusion equation with linear algebra, but this is computationally prohibitive and Thm. 4.3 provides an efficient low-rank approximation.
- Finally, Section 4.5 provides experiments and benchmarks. A particular focus is to test the ability of our model to capture long-range interactions between nodes and the robustness of pooling operations, potentially making it more robust to "over-squashing" [7].

4.2 Sequence Features by Non-Commutative Multiplication.

We recall the algebraic construction of sequence features from Chapter 3. Let us define the space of finite, but potentially different length sequences in $\mathbb{R}^d$ started from $\mathbf{0}$ as

$$\mathsf{Seq}(\mathbb{R}^d) = \left\{\mathbf{x} = (\mathbf{0}, \mathbf{x}_1, \ldots, \mathbf{x}_L) : \mathbf{x}_0, \ldots, \mathbf{x}_L \in \mathbb{R}^d, L \in \mathbb{N}\right\},$$

where analogously to Section 1, we use 0-based indexing for sequences, and define the effective length of a sequence $\mathbf{x} = (\mathbf{0}, \mathbf{x}_1, \ldots, \mathbf{x}_L) \in \mathsf{Seq}(\mathbb{R}^d)$ by $\mathsf{L}_\mathbf{x} = L$.

Assume we are given an injective map, which we call the *algebra lifting*, $\varphi : \mathbb{R}^d \to H$ from $\mathbb{R}^d$ into an algebra H. We can use this to define a *sequence feature map*[2]

$$\widetilde{\varphi} : \mathsf{Seq}(\mathbb{R}^d) \to H, \quad \widetilde{\varphi}(\mathbf{x}) = \varphi(\delta\mathbf{x}_1) \cdots \varphi(\delta\mathbf{x}_{\mathsf{L}_\mathbf{x}}), \tag{4.1}$$

[2]There are variants of this sequence feature map, which are discussed in [263, App. B] and [265, App. G].

where $\delta \mathbf{x}_i = \mathbf{x}_i - \mathbf{x}_{i-1}$ for $i \geq 1$ are used to denote the *increments* of a sequence $\mathbf{x} = (x_0, \ldots, x_k)$. This map associates to any sequence $\mathbf{x} \in \mathsf{Seq}(\mathbb{R}^d)$ an element of the algebra H. If the multiplication in H is commutative, then the map $\widetilde{\varphi}$ would have no information about the order of increments, i.e. $\varphi(\delta_0 \mathbf{x}) \cdots \varphi(\delta_k \mathbf{x}) = \varphi(\delta_{\pi(0)} \mathbf{x}) \cdots \varphi(\delta_{\pi(k)} \mathbf{x})$ for any permutation π of $\{0, \ldots, k\}$. However, if the multiplication in H is "non-commutative enough" we expect $\widetilde{\varphi}$ to be injective.

A Free Construction. Many choices for H are possible, but intuitively it makes sense to use the "most general object" for H. The mathematically rigorous approach is to use the *tensor algebra over* $\mathbb{R}^d$, see Section 1.1. Despite this abstract motivation, the algebra H has a concrete form: it is realized as a sequence of tensors in $\mathbb{R}^d$ of increasing degree, and is defined by

$$H = \{\mathbf{v} = (\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, \ldots) : \mathbf{v}_m \in (\mathbb{R}^d)^{\otimes m},\ m \in \mathbb{N},\ \|\mathbf{v}\| < \infty\},$$

where by convention $(\mathbb{R}^d)^{\otimes 0} = \mathbb{R}$, and we describe the norm $\|\mathbf{v}\|$ in the paragraph below. For example, if $\mathbf{v} = (\mathbf{v}_m)_{m \geq 0} \in H$, then $\mathbf{v}_0$ is a scalar, $\mathbf{v}_1$ is a vector, $\mathbf{v}_2 \in (\mathbb{R}^d)^{\otimes 2}$ is a $d \times d$ matrix, and so on. The vector space structure of H is given by addition and scalar multiplication as

$$\mathbf{v} + \mathbf{w} = (\mathbf{v}_m + \mathbf{w}_m)_{m \geq 0} \in H \quad \text{and} \quad \lambda \mathbf{v} = (\lambda \mathbf{v}_m)_{m \geq 0} \in H$$

for $\lambda \in \mathbb{R}$, and the algebra structure is given by

$$\mathbf{v} \cdot \mathbf{w} = \left(\sum_{i=0}^{m} \mathbf{v}_i \otimes \mathbf{w}_{m-i} \right)_{m \geq 0} \in H. \tag{4.2}$$

An Inner Product. If $e^1, \ldots, e^d$ is a basis of $\mathbb{R}^d$, then every tensor $\mathbf{v}_m \in (\mathbb{R}^d)^{\otimes m}$ can be written as

$$\mathbf{v}_m = \sum_{1 \leq i_1, \ldots, i_m \leq d} c_{i_1, \ldots, i_m} e^{i_1} \otimes \cdots \otimes e^{i_m}.$$

This allows us to define an inner product $\langle \cdot, \cdot \rangle_m$ on $(\mathbb{R}^d)^{\otimes m}$ by extending

$$\langle e^{i_1} \otimes \cdots \otimes e^{i_m}, e^{j_1} \otimes \cdots \otimes e^{j_m} \rangle_m = \begin{cases} 1 & : i_1 = j_1, \ldots, i_m = j_m, \\ 0 & : \text{otherwise.} \end{cases}$$

to $(\mathbb{R}^d)^{\otimes m}$ by linearity. This gives us an inner product on H,

$$\langle \mathbf{v}, \mathbf{w} \rangle = \sum_{m \geq 0} \langle \mathbf{v}_m, \mathbf{w}_m \rangle_m$$

such that H is a Hilbert space; in particular we get a norm $\|\mathbf{v}\| = \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle}$. To sum up, the space H has a rich structure: it has a vector space structure, it has an algebra structure (a noncommutative product), and it is a Hilbert space (an inner product between elements of H gives a scalar).

Characterizing Random Walks. From (4.1), we have constructed a map $\widetilde{\varphi}$ that maps a sequence $\mathbf{x} \in \mathsf{Seq}(\mathbb{R}^d)$ of arbitrary length into the space H (see [265, App. C] for further details). Our aim is to apply this to the sequence of node attributes corresponding to random walks on a graph. Therefore, the expectation of $\widetilde{\varphi}$ should be able to characterize the distribution of the random walk. Formally the map $\widetilde{\varphi}$ is *characteristic* if the map $\mu \mapsto \mathbb{E}_{\mathbf{x}\sim\mu}[\widetilde{\varphi}(\mathbf{x})]$ from the space of probability measures on $\mathsf{Seq}(\mathbb{R}^d)$ into H is injective. Indeed, if the chosen lifting φ satisfies some mild conditions this holds for $\widetilde{\varphi}$; see [265, App. C] and [44] for further details.

Linear Functionals. The quantity $\mathbb{E}_{\mathbf{x}\sim\mu}[\widetilde{\varphi}(\mathbf{x})]$ characterizes the probability measure μ but is valued in the infinite-dimensional Hilbert space H. However, we can instead consider

$$\langle \boldsymbol{\ell}, \mathbb{E}_{\mathbf{x}\sim\mu}[\widetilde{\varphi}(\mathbf{x})] \rangle \text{ for } \boldsymbol{\ell} = (\boldsymbol{\ell}_0, \boldsymbol{\ell}_1, \boldsymbol{\ell}_2, \ldots, \boldsymbol{\ell}_M, 0, \ldots) \in H \text{ and } M \geq 1 \tag{4.3}$$

which is equivalent to knowing $\mathbb{E}_{\mathbf{x}\sim\mu}[\widetilde{\varphi}(\mathbf{x})]$; i.e. the set (4.3) characterizes μ. This is analogous to how one can use either the moment generating function of a real-valued random variable or its moments to characterize its distribution; the former is infinite-dimensional (a function), the latter is an infinite sequence of scalars. We extend a key insight from [263] in Section 4.4: a linear functional $\langle \boldsymbol{\ell}, \mathbb{E}_{\mathbf{x}\sim\mu}[\widetilde{\varphi}(\mathbf{x})] \rangle$ can be efficiently approximated using low-rank decompositions.

The Tensor Exponential. While we will continue to keep φ arbitrary for our main results (see [263] and App. 4.C for other choices), we will use the *tensor exponential* defined by

$$\exp_{\otimes} : \mathbb{R}^d \to H, \quad \exp_{\otimes}(x) = \left(\frac{x^{\otimes m}}{m!} \right)_{m \geq 0},$$

as the primary example throughout this paper and in the experiments in Section 4.5. With this choice, the induced sequence feature map is the discretized version of a classical object in analysis, called the path signature, see [265, App. C] and Section 1.2.6 for further details.

4.3 Hypo-Elliptic Diffusions

Throughout this section, we fix a labelled graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, f)$, where $\mathcal{V}$ is a set of n nodes $\mathcal{V} = \{1, \dots, n\}$, $\mathcal{E}$ denotes edges and $f : \mathcal{V} \to \mathbb{R}^d$ are continuous node attributes[3] which map each node to an element in the vector space $\mathbb{R}^d$. Two nodes $i, j \in \mathcal{V}$ are *adjacent* if $(i, j) \in \mathcal{E}$ is an edge, and we denote this by $i \sim j$. The *adjacency matrix* A of a graph is defined by $A_{i,j} = 1$, whenever $i \sim j$, and 0 otherwise. We denote by $\deg(i)$ the number of neighbours of node i.

Random Walks on Graphs. Let $(B_k)_{k\geq 0}$ be the simple random walk on $\mathcal{V}$, where the initial node is chosen uniformly. The *transition matrix* of this time-homogeneous Markov chain is

$$P_{i,j} = \mathbb{P}(B_k = j | B_{k-1} = i) = \begin{cases} \frac{1}{\deg(i)} & : i \sim j \\ 0 & : \text{otherwise.} \end{cases}$$

Denote by $(L_k)_{k\geq 0}$ the random walk lifted to the node attributes in $\mathbb{R}^d$, that is

$$L_k = f(B_k). \tag{4.4}$$

Recall that the *normalized graph Laplacian* for random walks is defined as $\mathcal{L} = I - D^{-1}A$, where D is diagonal degree matrix; in particular, the entry-wise definition is

$$\mathcal{L}_{i,j} = \begin{cases} -\frac{1}{\deg(i)} & : i \sim j \\ 1 & : i = j \\ 0 & : \text{otherwise.} \end{cases}$$

The discrete graph diffusion equation for $U_k \in \mathbb{R}^{n\times d}$ is given by

$$U_k - U_{k-1} = -\mathcal{L}U_{k-1}, \quad U_0^{(i)} = f(i) \tag{4.5}$$

where the initial condition $U_0 \in \mathbb{R}^{n\times d}$ is specified by the node attributes.[4] The probabilistic interpretation of the solution to this diffusion equation is classical and given as

$$U_k = (\mathbb{E}[L_k \,|\, B_0 = i])_{i=1}^n = P^k U_0. \tag{4.6}$$

[3]The labels given by the labelled graph are called *attributes*, while the computed updates are called *features*.

[4]The attributes over all nodes are given by an $n \times d$ matrix; in particular $U_k^{(i)}$ is the i^{th} row of the matrix.

This allows us to compute the solution u_k using the transition matrix $P = I - \mathcal{L}$.

Random Walks on Algebras. We incorporate the history of a random walker by considering

$$\mathbb{E}[\widetilde{\varphi}(\mathbf{L}_k)\,|\,B_0 = i] = \mathbb{E}[\varphi(\delta\mathbf{L}_1)\cdots\varphi(\delta\mathbf{L}_{\mathbf{L}_k})\,|\,B_0 = i] \tag{4.7}$$

where $\mathbf{L}_k = (L_0, \ldots, L_k)$. Since $\widetilde{\varphi}$ captures the whole history of the random walk $\mathbf{L}_k$ over node attributes, we expect this expectation to provide a richer summary of the neighborhood of node i than $\mathbb{E}[L_k|B_0 = i]$. The price is however, the computational complexity, since (4.7) is H-valued. We first show, that analogous to (4.5), the quantity (4.7) satisfies a diffusion equation that can be computed with linear algebra. To do so, we develop a graph analogue of the hypo-elliptic Laplacian and replace the scalar entries of the matrices with entries from the algebra H.

Matrix Rings over Algebras. We first revisit the adjacency matrix $A \in \mathbb{R}^{n\times n}$ and replace it by the *tensor adjacency matrix* $\widetilde{A} = (\widetilde{A})_{i,j} \in H^{n\times n}$, that is $\widetilde{A}$ is a matrix but instead of scalar entries its entries are elements in the algebra H. The matrix A has an entry at i, j if nodes i and j are connected; $\widetilde{A}$ replaces the i, j entry with an element of H that tells us how node attributes differ,

$$\widetilde{A}_{i,j} = \begin{cases} \varphi(f(j) - f(i)) & : i \sim j \\ 0 & : \text{otherwise.} \end{cases} \tag{4.8}$$

Matrix multiplication works for elements of $H^{n\times n}$ by replacing scalar multiplication by multiplication in H, that is $(\widetilde{B}\cdot\widetilde{C})_{i,j} = \sum_{k=1}^{n} \widetilde{B}_{i,k}\cdot\widetilde{C}_{k,j}$ for $\widetilde{B}, \widetilde{C} \in H^{n\times n}$ and $\widetilde{B}_{i,k}\cdot\widetilde{C}_{k,j}$ denotes multiplication in H as in eq. (4.2). For the classical adjacency matrix A, the k-th power counts the number of length k walks in the graph, so that $(A^k)_{i,j}$ is the number of walks of length k from node i to node j. We can take powers of $\widetilde{A}$ in the same way as in the classical case, where

$$(\widetilde{A}^k)_{i,j} = \sum_{\mathbf{x}} \varphi(\delta\mathbf{x}_1)\cdots\varphi(\delta\mathbf{x}_{\mathbf{L}_\mathbf{x}})$$

where the sum is taken over all length k walks $\mathbf{x} = (f(i), \ldots f(j))$ from node i to node j (full details are provided in Appendix 4.A) Since $\widetilde{\varphi}(\mathbf{x})$ characterizes each walk $\mathbf{x}$, the entry $\widetilde{A}^k_{i,j}$ can be interpreted as a summary of all walks which connect nodes i and j.

Hypo-elliptic Graph Diffusion. Similar to the tensor adjacency matrix, we define the *hypo-elliptic graph Laplacian* as the $n \times n$ matrix

$$\widetilde{\mathcal{L}} = I - D^{-1}\widetilde{A} \in H^{n\times n},$$

where D is the degree matrix embedded into $H^{n\times n}$ at tensor degree 0. The entry-wise definition

$$\widetilde{\mathcal{L}}_{i,j} = \begin{cases} \frac{-\varphi(f(j)-f(i))}{\deg(i)} & : i \sim j \\ 1 & : i = j \\ 0 & : \text{otherwise.} \end{cases} \tag{4.9}$$

We can now formulate the *hypo-elliptic graph diffusion equation* for $\mathbf{v}_k \in H^n$ as

$$\mathbf{v}_k - \mathbf{v}_{k-1} = -\widetilde{\mathcal{L}}\mathbf{v}_{k-1}, \quad \mathbf{v}_0^{(i)} = \varphi(f(i)). \tag{4.10}$$

Analogous to classical graph diffusion (4.6), the hypo-elliptic graph diffusion (4.10) has a probabilistic interpretation in terms of $\mathbf{L}$ as shown in Thm. 4.1 (the proof is given in App. 4.A).

Theorem 4.1. *Let $k \in \mathbb{N}$, $\mathbf{L}_k = (L_0, \dots, L_k)$ be the lifted random walk from* (4.4)*, and $\widetilde{P} = I - \widetilde{\mathcal{L}}$ the* tensor adjacency matrix*. The solution to the hypo-elliptic graph diffusion eq.* (4.10) *is*

$$\mathbf{v}_k = (\mathbb{E}[\varphi(\delta_1\mathbf{L}_k)\cdots\varphi(\delta_k\mathbf{L}_k)|B_0 = i])_{i=1}^n = \widetilde{P}^k\mathbb{1}_H.$$

Furthermore, if $F \in H^{n\times n}$ is the diagonal matrix with $F_{i,i} = \varphi(f(i))$, then

$$F\mathbf{v}_k = (\mathbb{E}[\widetilde{\varphi}(\mathbf{L}_k)|B_0 = i])_{i=1}^n.$$

In the classical diffusion equation, U_k captures the concentration of the random walkers after k time steps over the nodes. In the hypo-elliptic diffusion equation, $\mathbf{v}_k$ captures summaries of random walk histories after k time steps over the nodes since $\widetilde{\varphi}(\mathbf{L}_k)$ summarizes the whole trajectory $\mathbf{L}_k = (L_0, \dots, L_k)$ and not only the endpoint L_k.

Node Features and Graph Features. Theorem 4.1 can then be used to compute features $\Phi(i) \in H$ for individual nodes as well as a feature $\Psi(\mathcal{G})$ for the entire graph. The former is

given by i-th component $\mathbf{v}_k^{(i)}$ of the solution $\mathbf{v}_k = (\mathbf{v}_k^{(i)})_{i=1,\dots,n} \in H^n$ of eq. (4.10),

$$\Phi(i) = \mathbf{v}_k^{(i)} = \mathbb{E}[\widetilde{\varphi}(\mathbf{L}_k) \,|\, B_0 = i] = (F\widetilde{P}^k \mathbf{v}_0)^{(i)} \in H,$$

since the random walk B chooses the starting node $B_0 = i$ uniformly at random. The latter can be computed by mean pooling the node features, which also has a probabilistic interpretation

$$\Psi(\mathcal{G}) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{v}_k^{(i)} = \mathbb{E}[\widetilde{\varphi}(\mathbf{L}_k)] = n^{-1}(\mathbb{1}_H^T F\widetilde{P}^k \mathbf{v}_0) \in H, \tag{4.11}$$

where $\mathbb{1}_H^T = (1_H, \dots, 1_H) \in H^n$ is the all-ones vector in H and 1_H denotes the unit in H.

Characterizing Graphs with Random Walks. The graph and node features obtained through the hypo-elliptic diffusion equation are highly descriptive: they characterize the history of the random walk process if one includes the time parameterization, as described in [265, App. C].

Theorem 4.2. *Suppose Ψ is the graph feature map from eq.* (4.11) *induced by the tensor exponential algebra lifting including time parameterization. Let $\mathcal{G}$ and $\mathcal{G}'$ be two labelled graphs, and $\mathbf{L}_k = (L_0, \dots, L_k)$ and $\mathbf{L}'_k = (L'_0, \dots, L'_k)$ be the k-step lifted random walk as defined in eq.* (4.4). *Then, $\Psi(\mathcal{G}) = \Psi(\mathcal{G}')$ if and only if the distributions of $\mathbf{L}_k$ and $\mathbf{L}'_k$ are equal.*

It is instructive to contrast this result with the classical diffusion case; the latter only uses the marginal distribution of L_k to capture the graph structure, which at least intuitively has much less expressive power. Indeed, in [265, App. E], we show that for elementary graphs, this already leads to big differences in expressive power. Further, an analogous result holds for the node features, and we prove both results in [265, App. E]. While we use the tensor exponential in this article, many other choices of $\widetilde{\varphi}$ are possible and result in graph and node features with such properties: under mild conditions, if the algebra lifting $\varphi : \mathbb{R}^d \to H$ characterizes measures on $\mathbb{R}^d$, the resulting node feature map Φ characterizes the random walk, see [263], which in turn implies the above results. Possible variations are discussed in Appendix 4.C.

General (Hypo-elliptic) Diffusions and Attention. One can consider more general diffusion operators, such as the normalized Laplacian $\mathcal{K}$ of a weighted graph. We define its lifted operator $\widetilde{\mathcal{K}} \in H^{n\times n}$ analogous to eq. (4.9), resulting in a generalization of 4.1 with $\widetilde{\mathcal{K}}$ replacing $\widetilde{\mathcal{L}}$. In the flavour of convolutional GNNs [25], we consider a weighted adjacency matrix $A \in \mathbb{R}^{n\times n}$

$$A_{i,j} = \begin{cases} c_{i,j} & : i \sim j \\ 0 & : \text{otherwise,} \end{cases}$$

for $c_{i,j} > 0$. The corresponding normalized Laplacian $\mathcal{K}$ is given by $\mathcal{K} = I - D^{-1}A$, where D is a diagonal matrix with $D_{i,i} = \sum_{j \in \mathcal{N}(i)} c_{i,j}$. A common way to learn the coefficients is by introducing parameter sharing across graphs by modelling them as $c_{i,j} = \exp(a(f(i), f(j)))$ using a local attention mechanism, $a : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ [276]. In our implementation, we use additive attention [13] given by $a(f(i), f(j)) = \texttt{LeakyRelu}_{0.2}(W_s f(i) + W_t f(j))$, where $W_s, W_t \in \mathbb{R}^{1 \times d}$ are linear transformations for the source and target nodes, but different attention mechanisms can also be used; e.g. scaled dot-product attention [275]. Then, the corresponding transition matrix $P = D^{-1}A$ is defined as $P_{ij} = \texttt{softmax}_{k \in \mathcal{N}(i)}(a(f(i), f(k)))_j$, and the lifted transition matrix as

$$\widetilde{P} = \begin{cases} P_{i,j}\varphi(f(j) - f(i)) & : i \sim j \\ 0 & : \text{otherwise.} \end{cases}$$

The statements of Theorem 4.1 immediately generalize to this variation by replacing the expectation with respect to a non-uniform random walk. Hence, in this case the use of attention can be interpreted as learning the transition probabilities of a random walk on the graph.

4.4 Efficient Algorithms for Deep Learning

The previous sections show that the node feature $\Phi(i)$ provides a structured description of the neighborhood of node i and it is instructive to think of a linear functional $\langle \ell, \Phi(i) \rangle$ as answering a specific question about the node neighbourhood, see [265, App. E] for examples. The naive computation of $\langle \ell, \Phi(i) \rangle$ by first computing $\Phi(i)$ and taking the inner product is too expensive, especially when $\ell = (\ell_0, \ldots, \ell_M, 0, \ldots) \in H$ for large M. To address this we revisit two observations from [263]: first, for a rank-1 functional $\ell \in H$, the computation of $\langle \ell, \Phi(i) \rangle$ is computationally cheap. Second, restriction to small M limits the expressive power but can be counteracted by composition: any choice of d different functionals $\ell^1, \ldots, \ell^d \in H$ gives a label update $f(i) \mapsto (\langle \ell^j, \Phi(i) \rangle)_{j=1,\ldots,d} \in \mathbb{R}^d$ for the graph. Repeating such a label update a few times with low-degree M and rank-1 functionals turns out to be as powerful as computing one update for general functionals with arbitrary high M. The first observation should not be too surprising given the popularity of low-rank approximations; the second observation is reminiscent to constructing a high-degree polynomial by composing low-degree polynomials[5] or the width-vs-depth phenomenon in neural nets and we give more details below.

[5] For example, $1 + x + x^2$ composed with $1 + 2x^2$ yields the degree 4 polynomial $1 + (1 + 2x^2) + (1 + 2x^2)^2$.

Computing Rank-1 Functionals. First, we focus on a *rank*-1 linear functional $\boldsymbol{\ell} \in H$ given as

$$\boldsymbol{\ell} = (\boldsymbol{\ell}_m)_{m\geq 0} \text{ with } \boldsymbol{\ell}_m = u_{M-m+1} \otimes \cdots \otimes u_M \text{ and } \boldsymbol{\ell}_m = 0 \text{ for } m > M, \tag{4.12}$$

where $u_m \in \mathbb{R}^d$ for $m = 1, \ldots, M$ for a fixed $M \geq 1$. Thm 4.3 shows for such $\boldsymbol{\ell}$, the computation of $\langle \boldsymbol{\ell}, \hat{\Phi}(i) \rangle$, where $\hat{\Phi}(i)$ is the node feature without basepoint, can be done (a) efficiently by factoring this low-rank structure into the recursive computation, and (b) simultaneously for all nodes $i \in \mathcal{V}$ in parallel. This can then be used to compute rank-R functionals for $R > 1$, and for $\langle \boldsymbol{\ell}, \Phi(i) \rangle$; see App. 4.B, and for a pseudo-code implementation see Algorithm 1 in [265, App. F].

Theorem 4.3. *Let $\boldsymbol{\ell}$ be as in* (4.12) *and define $f_{k,m} \in \mathbb{R}^n$ for $m = 1, \ldots, M$ as*

$$f_{1,m} = \frac{1}{m!} (P \odot C^{u_{M-m+1}} \odot \cdots \odot C^{u_M}) \cdot \mathbb{1},$$

where $\mathbb{1}^T = (1, \ldots, 1) \in \mathbb{R}^n$ is the all-ones vector; and for $2 \leq k$ and $1 \leq m \leq M$ recursively as

$$f_{k,m} = P \cdot f_{k-1,m} + \sum_{r=1}^{m} \frac{1}{r!} (P \odot C^{u_{M-m+1}} \odot \cdots \odot C^{u_{M-m+r}}) \cdot f_{k-1,m-r}, \tag{4.13}$$

where the matrix $C^u = (C^u_{i,j}) \in \mathbb{R}^{n \times n}$ is defined as

$$C^u_{i,j} = \begin{cases} \langle u, f(j) - f(i) \rangle & : i \sim j, \\ 0 & : \textit{otherwise}. \end{cases}$$

Here $\odot$ denotes element-wise[6] *multiplication, while $\cdot$ denotes matrix multiplication. Then, it holds for $i \in \mathcal{V}$, random walk length $k \in \mathbb{Z}_+$, and tensor degree $m = 1, \ldots, M$, that*

$$f_{k,m}(i) = \langle \boldsymbol{\ell}_m, \hat{\Phi}_k(i) \rangle,$$

where $\hat{\Phi}_k(i) = \mathbb{E}[\varphi(\delta_1 \mathbf{L}_k) \cdots \varphi(\delta_k \mathbf{L}_k) \,|\, B_0 = i]$.

Overall, Eq. (4.13) computes $f_{k,m}(i)$ for all $i \in \mathcal{V}$, $k = 1, \ldots, K$, $m = 1, \ldots, M$ in $O(K \cdot M^2 \cdot N_E + M \cdot N_E \cdot d)$ operations, where $N_E \in \mathbb{N}$ denotes the number of edges; see App. 4.B. In particular, one does not need to compute $\Phi(i) \in H$ directly or store large tensors.

[6] For example $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \odot \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 5 & 12 \\ 21 & 32 \end{bmatrix}$.

Graph Labelling Layers. Fixing d rank 1-functionals $\boldsymbol{\ell}^1, \ldots, \boldsymbol{\ell}^d \in H$ induces a label update $f(i) \mapsto (\langle \boldsymbol{\ell}^i, \Phi(i) \rangle)_{i=1,\ldots,d} \in \mathbb{R}^d$. Theorem 4.3 allows us to compute this update in parallel for all nodes in $\mathcal{V}$. Such a label update is similar to hidden layer in a neural network (NN) and we can stack such updates, see Figure 4.1 in Appendix 4.D.1. As in NN, the d functionals in each "graph labelling layer" are optimized by gradient descent. Finally, note that a rank-R functional is the sum of R rank-1 functionals so we can immediately carry out the same construction with rank-R functionals by adding a linear mixing layer. To sum up, a graph labelling layer is determined by the random walk length k, the maximal tensor degree-M, maximal tensor rank-R and the functionals themselves are then found by optimization.

Using a single layer of low-rank functionals limits the expressiveness but stacking layers allows in practice to approximate general, high-degree M functionals. Some theoretical results can be found in [263]; however, here we simply appeal to the analogy with NN where stacking simple transformations provides a flexible functional class with good inductive bias.

4.5 Experiments

We implemented the above approach and call the resulting model **G**raph**2T**ens **N**etworks since it represents the neighbourhood of a node as a sequence of tensors, which is further pushed through a low-tensor-rank constrained linear mapping, similarly to how neural networks linearly transform their inputs pre-activation. A conceptual difference is that in our case the non-linearity is applied first and the projection secondly, albeit the computation is coupled between these steps. We provide further experiments and ablation studies of our models in App. 4.E.

Experimental Setup. The aim of our main experiment is to test the following key properties: (1) ability to capture long-range interactions between nodes in a graph, (2) robustness to pooling operations, hence making it less susceptible to the "over-squashing" phenomenon [7]. We do this by following the experiments in [294]. In particular, we show that our model is competitive with previous approaches for retaining long-range context in graph-level learning tasks but without computing all pairwise interactions between nodes, thus keeping the influence distribution localized [297]. We further give a detailed ablation study to show the robustness of our model to various architectural choices in App. 4.E.1. As a second experiment, we follow the previous applications of diffusion approaches to graphs that have mostly considered inductive learning tasks, e.g. on the citation datasets [37, 258, 36]. Our experimentation on these datasets are available in App. 4.E.2, where the model performs on par with short-range GNN models, but does not seem to benefit from added long-range information a-priori. However, when labels are dropped in a k-hop sanitized way as in [213], the performance decrease is less pronounced.

Datasets. We use two biological graph classification datasets (NCI1 and NCI109), that contain around ∼4000 biochemical compounds represented as graphs with ∼30 nodes on average [282, 1]. The task is to predict whether a compound contains anti-lung-cancer activity. The dataset is split in a ratio of $80\% - 10\% - 10\%$ for training, validation and testing. Previous work [7] has found that GNNs that only summarize local structural information can be outperformed by models accounting for global contextual relationships through the use of *fully-adjacent* layers. This was further improved on by [294], where a local neighbourhood encoder consisting of a GNN stack was upgraded with a Transformer submodule [275] for learning global interactions.

Model Details. We build a GNN architecture primarily motivated by the GraphTrans (small) model from [294], and only fine-tune the pre- and postprocessing layers(s), random walk length, functional degree and optimization settings. In detail, a preprocessing MLP layer with 128 hidden units is followed by a stack of 4 G2TN layers each with RW length-5, max rank-128, max tensor degree-2, all equipped with JK-connections [297] and a max aggregator. Afterwards, the node features are combined into a graph-level representation using gated attention pooling [169]. The pooled features are transformed using a final MLP layer with 256 hidden units, and then fed into a softmax classification layer. The pre- and postprocessing MLP layers employ skip-connections [113]. Both MLP and G2TN layers are followed by layer normalization [11], where GT2N layers normalize their rank-1 functionals independently across different tensor degrees, which corresponds to a particular realization of group normalization [293]. We randomly drop 10% of the features for all hidden layers during training [247]. The attentional variant, G2T(A)N also randomly drops 10% of its edges and uses 8 attention heads [276]. Training is performed by minimizing the categorical cross-entropy loss with an ℓ_2 regularization penalty of 10^{-4}. For optimization, Adam [143] is used with a batch size of 128 and an inital learning rate of 10^{-3} that is decayed via a cosine annealing schedule [177] over 200 epochs. Further intuition about the model and architectural choices are available in App. 4.D.1.

Baselines. We compare against (1) the baseline models reported in [294], (2) variations of GraphTrans, (3) other recently proposed hierarchical approaches for long-range graph tasks [213]. Groups of models in Table 4.1 are separated by dashed lines if they were reported in separate papers, and the first citation after the name is where the result first appeared. The number of GNN layers in HGNet are not discussed by [213], and we report it as implied by their code. We organize the models into three groups divided by solid lines: (a) baselines that only apply neighbourhood aggregations, and hierarchical or global pooling schemes, (b) baselines that first employ a local neighbourhood encoder, and afterwards fully densify the graph in some way so all nodes interact with each other *directly*, (c) our models that that belong to (a).

Table 4.1: Comparison of classification accuracies on NCI biological datasets, where we report mean and standard deviation over 10 random seeds for our models.

MODEL	GNN TYPE	GNN COUNT	NCI1 (%)	NCI109 (%)
Set2Set [164, 278]	GCN	3	68.6 ± 1.9	69.8 ± 1.2
SortPool [164, 303]	GCN	3	73.8 ± 1.0	74.0 ± 1.2
SAGPool_h [164]	GCN	3	67.5 ± 1.1	67.9 ± 1.4
SAGPool_g [164]	GCN	3	74.2 ± 1.2	74.1 ± 0.8
GIN [80, 296]	GIN	8	80.0 ± 1.4	-
GCN + VN [298, 95]	GCN	2	71.5	-
HGNet-EdgePool [298, 233]	GCN+RGCN	$3+2$	77.1	-
HGNet-Louvain [298, 233]	GCN+RGCN	$3+2$	75.1	-
GIN + FA [7, 296]	GIN	8	81.5 ± 1.2	-
GraphTrans (small) [294, 275]	GCN	3	81.3 ± 1.9	79.2 ± 2.2
GraphTrans (large) [294, 275]	GCN	4	$\mathbf{82.6 \pm 1.2}$	$\mathbf{82.3 \pm 2.6}$
G2T(A)N (OURS)	G2T(A)N	4	$\mathbf{81.9 \pm 1.2}$	78.0 ± 2.3
G2TN (OURS)	G2TN	4	80.7 ± 2.5	$\mathbf{78.9 \pm 2.5}$

Results. In Table 4.1, we report the mean and standard deviation of classification accuracy computed over 10 different seeds. Overall, both our models improve over all baselines in group (a) on both datasets, maximally by 1.9% on NCI1 and by 4.8% on NCI109. In group (b), G2T(A)N is solely outperformed by GraphTrans (large) on NCI1 by only 0.7%. Interestingly, the attention-free variation, G2TN , performs better on NCI109, where it performs very slightly worse than GraphTrans (small).

Discussion. The previous experiments demonstrate that our approach performs very favourably on long-range reasoning tasks compared to GNN-based alternatives without global pairwise node interactions. Several of the works we compare against have focused on extending GNNs to larger neighbourhoods by specifically designed graph coarsening and pooling operations, and we emphasize two important points: (1) our approach can efficiently capture large neighbourhoods without any need for coarsening, (2) it already performs well with simple mean-pooling as justified by Theorem 4.2 and experimentally supported by the ablation studies in 4.E.1. Although the Transformer-based GraphTrans slightly outperforms our model potentially due to its ability to learn global interactions, it is not entirely clear how much of the global graph structure it is able to infer from interactions of short-range neighbourhood summaries. Finally, Transformer models can be bottlenecked by their quadratic complexity in nodes, while our approach only scales with edges, and hence, it can be more favourable for large sparse graphs in terms of computations.

4.6 Conclusion

Inspired by classical results from analysis [93], we introduce the hypo-elliptic graph Laplacian. This yields a diffusion equation and also generalizes its classical probabilistic interpretation via random walks but now taking history into account. In addition to several attractive theoretical guarantees, we provide scalable algorithms. Our experiments show that this can lead to largely improved baselines for long-range reasoning tasks. A promising future research theme is to develop improvements for the classical Laplacian in this hypo-elliptic context; including lazy random walks [295]; nonlinear diffusions [37]; and source/sink terms [258]. Another theme could be to extend the geometric study [260] of over-squashing to this hypo-elliptic point of view which is naturally tied to sub-Riemannian geometry [249]. A limitation in our theoretical results is that for the iterations of low-rank approximations only partial results exist and expanding this is an interesting (algebra-heavy) topic.

4.A Further Details on Hypo-elliptic Graph Diffusion

In Section 4.3, we introduced tensor analogues of classical matrix operators, which were then used to define hypo-elliptic graph diffusion. These tensor-valued matrices allow us to efficiently represent random walk histories on a graph, which can be manipulated using matrix operations. In this section, we discuss further details on the tensor adjacency matrix, provide a proof of Theorem 4.1, and introduce a variation of hypo-elliptic graph diffusion.

As in Section 4.3, we fix a labelled graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, f)$, where the nodes are denoted by integers, $\mathcal{V} = \{1, \dots, n\}$, and $f : \mathcal{V} \to \mathbb{R}^d$ denotes the continuous node attributes.

Powers of the Tensor Adjacency Matrix. Recall that the powers of the classical adjacency matrix A counts the number of walks between two nodes on the graph. In particular, given $k \in \mathbb{N}$, and two nodes $i, j \in \mathcal{V}$, the result follows as a consequence of the sparsity pattern of A,

$$(A^k)_{i,j} = \sum_{i_1,\dots,i_{k-1}=1}^{n} A_{i,i_1} \cdot A_{i_1,i_2} \cdots A_{i_{k-1},j} = \sum_{i=i_0\sim\dots\sim i_k=j} 1.$$

Note that the product $A_{i,i_1} \cdots A_{i_{k-1},j} = 0$ unless each pair of consecutive indices are adjacent in the graph, namely $i_{q-1} \sim i_q$ for all $q = 1, \dots, k$. Applying the same procedure to the tensor adjacency matrix from eq. (4.8), we obtain a summary of all walks between two nodes, rather than just the number of walks. In particular,

$$
\begin{aligned}
(\widetilde{A}^k)_{i,j} &= \sum_{i=i_0\sim\cdots\sim i_k=j} \widetilde{A}_{i,i_1}\cdot\widetilde{A}_{i_1,i_2}\cdots\widetilde{A}_{i_{k-1},j} \\
&= \sum_{i=i_0\sim\cdots\sim i_k=j} \varphi(f(i_1)-f(i))\cdot\varphi(f(i_2)-f(i_1))\cdots\varphi(f(j)-f(i_{k-1})) \\
&= \sum_{i=i_0\sim\cdots\sim i_k=j} \varphi(\delta_1\mathbf{x})\cdots\varphi(\delta_k\mathbf{x}),
\end{aligned}
$$

where $\mathbf{x} = (f(i_0),\ldots,f(i_k))$ denotes the lifted sequence in the vector space $\mathbb{R}^d$. Note that this corresponds to the sequence feature map *without* the initial point $\delta_0\mathbf{x}$.

Powers of the Tensor Transition Matrix. We now consider powers of the classical transition matrix $P = I - \mathcal{L}$, where $\mathcal{L} = I - D^{-1}A$ is the normalized graph Laplacian. The entries of P^k provide length k random walk probabilities; in particular, we have

$$
(P^k)_{i,j} = \sum_{i=i_0\sim\ldots\sim i_k=j} \frac{1}{\deg(i_0)\deg(i_1)\ldots\deg(i_{k-1})} = \mathbb{P}[B_k = j | B_0 = i].
$$

The powers of the tensor transition matrix $\widetilde{P} = I - \widetilde{\mathcal{L}}$, where $\widetilde{\mathcal{L}} = I - D^{-1}\widetilde{A}$ is the hypo-elliptic Laplacian, will be the conditional expectation of the sequence feature map of the random walk process. In particular,

$$
(\widetilde{P})^k_{i,j} = \sum_{i=i_0\sim\cdots\sim i_k=j} \varphi(\delta_1\mathbf{x})\cdots\varphi(\delta_k\mathbf{x})\mathbb{P}[B_1 = i_1, B_2 = i_2, \ldots, B_k = i_k | B_0 = i]. \tag{4.14}
$$

Proof of Hypo-elliptic Diffusion Theorem. Recall from Section 4.3 that the *hypo-elliptic graph diffusion equation*[7] is defined by

$$
\mathbf{v}_k - \mathbf{v}_{k-1} = -\widetilde{\mathcal{L}}\mathbf{v}_{k-1}, \quad \mathbf{v}_0 = \mathbb{1}_H,
$$

where $\mathbb{1}_H^T = (1_H,\ldots,1_H) \in H^n$ is the all-ones vector in H. Using the above computations for powers of the tensor transition matrix, we can prove Thm. 4.1, which is restated here.

[7]There is a minor typo in 4.3 which does not alter our results; the initial condition should be $\mathbf{v}_0 = \mathbb{1}_H$ as given here. The statement of 4.1 and the resulting node/graph features are adjusted accordingly.

Theorem 4.4. *Let $k \in \mathbb{N}$, $\mathbf{L}_k = (L_0, \dots, L_k)$ be the lifted random walk from* (4.4)*, and $\widetilde{P} = I - \widetilde{\mathcal{L}}$ be the* tensor adjacency matrix*. The solution to the hypo-elliptic graph diffusion eq.* (4.10) *is*

$$\mathbf{v}_k = (\mathbb{E}[\varphi(\delta_1 \mathbf{L}_k) \cdots \varphi(\delta_k \mathbf{L}_k) | B_0 = i])_{i=1}^n = \widetilde{P}^k \mathbb{1}_H.$$

Furthermore, if $F \in H^{n \times n}$ is the diagonal matrix with $F_{i,i} = \varphi(f(i))$, then

$$F\mathbf{v}_k = (\mathbb{E}[\widetilde{\varphi}(\mathbf{L}_k) | B_0 = i])_{i=1}^n .$$

Proof of Theorem 4.1. We begin by proving the first equation. From the definition of hypo-elliptic diffusion, it is straightforward to see that

$$\mathbf{v}_k = (I - \widetilde{\mathcal{L}})\mathbf{v}_{k-1} = \widetilde{P}\mathbf{v}_{k-1} = \widetilde{P}^k \mathbf{v}_0 = \widetilde{P}^k \mathbb{1}_H.$$

We prove coordinate-wise that $\mathbf{v}_k = (\mathbb{E}[\varphi(\delta_1 \mathbf{L}_k) \cdots \varphi(\delta_k \mathbf{L}_k) | B_0 = i])_{i=1}^n$. Indeed, using the above and eq. (4.14), the i-th coordinate of $\mathbf{v}_k$ is

$$\begin{aligned} \mathbf{v}_k^{(i)} = (\widetilde{P}^k \mathbb{1}_H)^{(i)} = \sum_{j=1}^n (\widetilde{P}^k)_{i,j} &= \sum_{i=i_0 \sim \ldots \sim i_k} \varphi(\delta_1 \mathbf{x}) \cdots \varphi(\delta_k \mathbf{x}) \mathbb{P}[B_1 = i_1, \dots, B_k = i_k | B_0 = i] \\ &= \mathbb{E}[\varphi(\delta_1 \mathbf{L}_k) \cdots \varphi(\delta_k \mathbf{L}_k) | B_0 = i], \end{aligned}$$

where $\mathbf{x} = (f(i_0), \dots, f(i_k))$ is the lifted sequence corresponding to a walk $i_0 \sim \ldots \sim i_k$ on the graph. Next, we will also prove the second equation coordinate-wise. Using the above result, we have

$$(F\mathbf{v}_k)^{(i)} = \varphi(f(i)) \mathbb{E}[\varphi(\delta_1 \mathbf{L}_k) \cdots \varphi(\delta_k \mathbf{L}_k) | B_0 = i] = \mathbb{E}[\widetilde{\varphi}(\mathbf{L}_k) | B_0 = i],$$

where we use the fact that $\delta_0 \mathbf{L}_k = L_0 = f(i)$ when we condition $B_0 = i$. □

Forward Hypo-elliptic Diffusion. In the classical setting, we can consider both the forward and backward Kolmogorov equations. Throughout the main text and in the appendix so far, we have been considering the *backward* variants. In this section, we formulate the *forward* analogue of hypo-elliptic diffusion. In the classical graph setting, this corresponds to the following forward equation for $U_k \in \mathbb{R}^{n \times d}$ given by

$$U_k^T - U_{k-1}^T = -U_{k-1}^T \mathcal{L}, \quad U_0^{(i)} = f(i)$$

where the initial condition $U_0 \in \mathbb{R}^{n\times d}$ is specified by the node attributes. Note that because $P = D^{-1}A$ is right stochastic[8], this variation of the graph diffusion equation conserves mass in each coordinate of the node attributes at every time step.

Similarly we can formulate the forward hypo-elliptic graph diffusion equation for $\mathbf{v}_k \in H^n$ as

$$\mathbf{v}_k^T - \mathbf{v}_{k-1}^T = -\mathbf{v}_{k-1}^T \widetilde{\mathcal{L}}, \quad \mathbf{v}_0 = n^{-1}\mathbb{1}_H. \tag{4.15}$$

The solution of eq. 4.15 is given below.

Theorem 4.5. *Let $k \in \mathbb{N}$, $\mathbf{L}_k = (L_0, \dots, L_k)$ be the lifted random walk from* (4.4)*, and $\widetilde{P} = I - \widetilde{\mathcal{L}}$ be the* tensor adjacency matrix*. The solution to the forward hypo-elliptic graph diffusion* (4.15) *is*

$$\mathbf{v}_k^T = (\mathbb{P}[B_k = i]\mathbb{E}[\varphi(\delta_1\mathbf{L}_k)\cdots\varphi(\delta_k\mathbf{L}_k)|B_k = i])_{i=1}^n = \frac{1}{n}\mathbb{1}_H^T\widetilde{P}^k.$$

Furthermore, if $F \in H^{n\times n}$ is the diagonal matrix with $F_{i,i} = \varphi(f(i))$, then

$$\frac{1}{n}\mathbb{1}_H^T F\widetilde{P}^k = (\mathbb{P}[B_k = i]\mathbb{E}[\widetilde{\varphi}(\mathbf{L}_k)|B_k = i])_{i=1}^n.$$

Proof. The proof proceeds in the same way as the backward equation. By definition of the forward hypo-elliptic diffusion, we have

$$\mathbf{v}_k^T = \mathbf{v}_0^T\widetilde{P}^k = \frac{1}{n}\mathbb{1}_H^T\widetilde{P}^k.$$

Now, recall that the initial point of the random walk process is chosen uniformly over all nodes; in other words, $\mathbb{P}[B_0 = i] = \frac{1}{n}$. Then, we show $\mathbf{v}_k^T = (\mathbb{E}[\varphi(\delta_1\mathbf{L}_k)\cdots\varphi(\delta_k\mathbf{L}_k)|B_k = i])_{i=1}^n$ coordinate-wise as

$$\begin{aligned}
\mathbf{v}_k^{(i)} &= \frac{1}{n}\sum_{j=1}^n (\widetilde{P}^k)_{j,i} \\
&= \sum_{i_0\sim\dots\sim i_k = i} \varphi(\delta_1\mathbf{x})\cdots\varphi(\delta_k\mathbf{x})\mathbb{P}[B_1 = i_1, \dots, B_k = i|B_0 = i_0]\mathbb{P}[B_0 = i_0] \\
&= \sum_{i_0\sim\dots\sim i_k = i} \varphi(\delta_1\mathbf{x})\cdots\varphi(\delta_k\mathbf{x})\mathbb{P}[B_0 = j, \dots, B_{k-1} = i_{k-1}|B_k = i]\mathbb{P}[B_k = i] \\
&= \mathbb{P}[B_k = i]\mathbb{E}[\varphi(\delta_1\mathbf{L}_k)\cdots\varphi(\delta_k\mathbf{L}_k)|B_k = i],
\end{aligned}$$

where $\mathbf{x} = (f(i_0), \dots, f(i_k))$ is the lifted sequence corresponding to a walk $i_0 \sim \dots \sim i_k$ on the

[8] Each column sums to one and hence multiplying on the left by a row vector conserves its mass.

graph. We will now prove the second equation. Note that we have

$$(F\widetilde{P}^k)_{i,j} = \sum_{i=i_0\sim\ldots\sim i_k=j} \widetilde{\varphi}(\mathbf{x})\mathbb{P}[B_1 = i_1,\ldots,B_k = i_k \,|\, B_0 = i].$$

Then, following the same reasoning as the first equation, we obtain the desired result. □

Weighted Graphs. In the prior discussion, we considered simple random walks in which the walk chooses one of the nodes adjacent to the current node uniformly at random. We can instead consider more general random walks on graphs, which can be described using a weighted adjacency matrix,

$$A_{i,j} = \begin{cases} c_{i,j} & : i \sim j \\ 0 & : \text{otherwise.} \end{cases}$$

In this case, we define the diagonal weighted degree matrix to be $D_{i,i} = \sum_{i\sim j} A_{i,j}$. We now define a weighted random walk $(B_k)_{k\geq 0}$ on the vertices $\mathcal{V}$ where the transition matrix is given by

$$P_{i,j} \coloneqq D^{-1}A = \mathbb{P}(B_k = j \,|\, B_{k-1} = i) = \begin{cases} \frac{c_{i,j}}{\sum_{i\sim j'} c_{i,j'}} & : i \sim j \\ 0 & : \text{otherwise.} \end{cases}$$

With these weighted graphs, the powers of the adjacency and transition matrix can be interpreted in a similar manner as the standard case. Powers of the adjacency matrix provide total weights over walks, while the powers of the transition matrix provide the probability of a weighted walk going between two nodes after a specified number of steps. In particular,

$$(A^k)_{i,j} = \sum_{i\sim i_1\sim\ldots\sim i_{k-1}\sim j} c_{i,i_1}\ldots c_{i_{k-1},j}$$
$$(P^k)_{i,j} = \mathbb{P}[B_k = j \,|\, B_0 = i].$$

The tensor adjacency and tensor transition matrices are defined in the same manner as

$$\widetilde{A}_{i,j} \coloneqq \begin{cases} c_{i,j}\varphi(f(j)-f(i)) & : i \sim j \\ 0 & : \text{otherwise} \end{cases}$$

$$\widetilde{P}_{i,j} := D^{-1}\widetilde{A} = \begin{cases} P_{i,j}\varphi(f(j)-f(i)) & : i \sim j \\ 0 & : \text{otherwise} \end{cases}$$

Note that powers of this weighted tensor transition matrix are exactly the same as the unweighted case from eq. (4.14), and thus the weighted version of both Theorems 4.2 and 4.5 immediately follow.

4.B Details on the low-rank Algorithm.

In this section, we will provide further details and proofs on the low-rank approximation method discussed in Sec. 4.4. We can use low-rank tensors to define corresponding functionals of the features obtained via hypo-elliptic diffusion. The following is the definition of *CP-rank* [31].

Definition 4.6. *The* rank *of a level m tensor $\mathbf{v}_m \in (\mathbb{R}^d)^{\otimes m}$ is the smallest number $r \geq 0$ such that*

$$\mathbf{v}_m = \sum_{i=1}^{r} v_{i,1} \otimes \ldots \otimes v_{i,m}, \quad v_{i,j} \in \mathbb{R}^d.$$

We say that $\mathbf{v} = (\mathbf{v}_0, \mathbf{v}_1, \ldots) \in H$ is rank 1 *if all $\mathbf{v}_m$ are rank 1 tensors.*

We will now prove Theorem 4.3, which is stated using the node feature map without `ZeroStart` (App. 4.C). In particular, the node feature map is given by

$$\hat{\Phi}_k(i) = \mathbb{E}[\varphi(\delta_1 \mathbf{L}_k) \cdots \varphi(\delta_k \mathbf{L}_k) \,|\, B_0 = i] = (\widetilde{P}^k \mathbb{1}_H)^{(i)} \in H,$$

where we explicitly specify the walk length k in the subscript. Furthermore, if we also need to specify the tensor degree m, we will use two subscripts, where

$$\hat{\Phi}_{k,m}(i) \in (\mathbb{R}^d)^{\otimes m}$$

is the level m component of the hypo-elliptic diffusion with a walk length of k. Throughout the proof, we omit the H subscript for the all-ones vector, such that $\mathbb{1}^T = (1_H, \ldots, 1_H) \in H^n$, and we denote $\mathbb{1}_i^T = (0, \ldots, 1_H, \ldots, 0) \in H^n$ to be the unit vector in the i^{th} coordinate.

Proof of Theorem 4.3. First, we will show that $f_{1,m}(i) = \langle \boldsymbol{\ell}_m, \hat{\Phi}_1(i) \rangle$ for all $m = 1, \ldots, M$. By the definition of hypo-elliptic diffusion, we know that

$$\hat{\Phi}_1(i) = \mathbb{1}_i^T \widetilde{P} \mathbb{1} = \sum_{j=1} \widetilde{P}_{i,j} = \sum_{i \sim j} \frac{\exp_{\otimes}(f(j)-f(i))}{d_i}.$$

By explicitly expressing the level m component, and by factoring out the inner product, we get

$$\left\langle \ell_m, \frac{\exp_\otimes(f(j)-f(i))}{d_i} \right\rangle = \frac{1}{d_i m!} \prod_{r=M-m+1}^{M} \langle u_r, f(j)-f(i)\rangle$$
$$= \frac{1}{m!}(P_{i,j} \cdot C_{i,j}^{u_{M-m+1}} \cdot \ldots \cdot C_{i,j}^{u_M}).$$

Then by linearity of the inner product, we get $f_{1,m}(i) = \langle \ell_m, \hat{\Phi}_1(i)\rangle$.

Next, we continue by induction and suppose that $f_{k-1,m}(i) = \langle \ell_m, \hat{\Phi}_{k-1}(i)\rangle$ holds for all $m = 1, \ldots, M$. Starting once again from the definition of hypo-elliptic diffusion, we know that

$$\hat{\Phi}_k(i) = \mathbb{1}_i^T \widetilde{P} \hat{\Phi}_{k-1} = \sum_{i\sim j} \widetilde{P}_{i,j} \cdot \hat{\Phi}_{k-1}(j).$$

Fix a degree m. We explicitly write out the level m component of this equation by expanding $\widetilde{P}_{i,j}$ and the tensor product as

$$\begin{aligned}\hat{\Phi}_{k,m}(i) &= \sum_{i\sim j} \sum_{r=0}^{m} \frac{(f(j)-f(i))^{\otimes r}}{d_i r!} \cdot \hat{\Phi}_{k-1,m-r}(j) \\ &= \sum_{i\sim j} \frac{\hat{\Phi}_{k-1,m}(j)}{d_i} + \sum_{r=1}^{m} \frac{1}{r!} \sum_{i\sim j} \frac{(f(j)-f(i))^{\otimes r}}{d_i} \cdot \hat{\Phi}_{k-1,m-r}(j) \end{aligned} \tag{4.16}$$

Note that the first sum is equivalent to

$$\sum_{i\sim j} \frac{\hat{\Phi}_{k-1,m}(j)}{d_i} = \mathbb{1}_i^T P \cdot \hat{\Phi}_{k-1,m}.$$

Applying the linear functional ℓ_m and the induction hypothesis to this, we have

$$\left\langle \ell_m, \sum_{i\sim j} \frac{\hat{\Phi}_{k-1,m}(j)}{d_i} \right\rangle = \mathbb{1}_i^T P \cdot f_{k-1,m}.$$

For the second sum in (4.16), we factor the inner product and apply the induction hypothesis

$$\left\langle \ell_m, \sum_{i\sim j} \frac{(f(j)-f(i))^{\otimes r}}{d_i} \cdot \hat{\Phi}_{k-1,m-r}(j) \right\rangle = \sum_{j=1}^{n} P_{i,j} \cdot C_{i,j}^{u_{M-m+1}} \cdot \ldots \cdot C_{i,j}^{M-m+r} \cdot f_{k-1,m-r}.$$

Putting this all together, we get

$$\hat{\Phi}_{k,m}(i) = \mathbb{1}_i^T \left(P \cdot f_{k-1,m} + \sum_{r=1}^{m} \frac{1}{r!} (P \odot C^{u_{M-m+1}} \odot \ldots \odot C^{u_M}) \cdot f_{k-1,m-r} \right) = f_{k,m}(i).$$

□

Computational Complexity. We will now consider the computational complexity of our algorithms. We begin by noting that the naive approach of computing $\Phi_k(i) = (F\widetilde{P}^k \mathbb{1}_H)^{(i)}$ has the computational complexity of matrix multiplication; though this counts tensor operations, which itself requires $O(d^m)$ scalar multiplications at tensor degree m. This is computationally too expensive for practical applications.

Next, we consider the complexity of the recursive low-rank algorithm from Thm. 4.3, where the primary computational advantage is that we only perform *scalar* operations rather than *tensor* operations. We consider the recursive step from eq. (4.13), reproduced here for $m = M$,

$$f_{k,M} = P \cdot f_{k-1,M} + \sum_{r=1}^{M} \frac{1}{r!} (P \odot C^{u_1} \odot \cdots \odot C^{u_r}) \cdot f_{k-1,M-r}.$$

Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, f)$ with $n = |\mathcal{V}|$ nodes and $E = |\mathcal{E}|$ edges, both P and C^u are sparse $n \times n$ (scalar) matrices with $O(E)$ nonzero entries, and $f_{k,m}$ is an n-dimensional column vector. Recall that $\odot$ denotes element-wise multiplication, and thus both the sparse matrix-matrix multiplication and the sparse matrix-vector multiplication have complexity $O(E)$. Furthermore, the entry-wise products $C^{u_1} \odot \cdots \odot C^{u_r}$ differ by only one factor between $r = m$ and $r = m + 1$, and thus, computing $f_{k,M}$ assuming all lower $f_{k',m'}$ have been computed has complexity $O(ME)$. Taking into account the two recursive parameters results in a complexity of $O(kM^2E)$. Note that this is the complexity to compute features for *all nodes*. Once the $f_{k,m}$ are computed, the complexity of adding the zero starting point is $O(M)$.

4.C Variations and Hyperparameters of Hypo-Elliptic Diffusion

In this appendix, we summarize possible variations of the sequence feature map, leading to different hypo-elliptic diffusion features. The choice of variation is learned during training, and we also summarize the hyperparameters used for our features. While the theoretical results on characterizing random walks, such as Thm. 4.2, depend on specific choices of the sequence feature map, there exist analogous results for these variations, which can characterize random walks up to certain equivalences. Furthermore, the computation of these variations can be

performed in the same way: through tensorized linear algebra for exact solutions, and through an analogous low-rank method (as in Thm. 4.3) for approximate solutions.

We fix an algebra lifting $\varphi : \mathbb{R}^d \to H$ and let $\mathbf{x} = (x_0, \dots, x_k) \in \mathsf{Seq}(\mathbb{R}^d)$. The simplest sequence feature map to define simply multiplies the terms in the sequence together as

$$\widetilde{\varphi}(\mathbf{x}) = \varphi(x_0) \cdots \varphi(x_k). \tag{4.17}$$

Note that this is *not* the sequence feature map used in the main text. We will now discuss several variations of this map, where $\widetilde{\varphi}_{\text{inc,zs}}$ is the one primarily used in the main text and $\widetilde{\varphi}_{\text{inc,zs,tp}}$ is used to characterize random walks in Theorem 4.2 and [265, App. E].

Increments (`Diff`). Rather than directly multiplying terms in the sequence together, we can instead multiply the *increments* as

$$\widetilde{\varphi}_{\text{inc}}(\mathbf{x}) = \varphi(\delta_1 \mathbf{x}) \cdots \varphi(\delta_k \mathbf{x}),$$

where $\delta_i \mathbf{x} = x_i - x_{i-1}$ for $i \geq 1$. In both cases, the sequence feature map is the path signature of a continuous piecewise-linear path when we set $\varphi = \exp_\otimes$, as discussed in [265, App. C], and it is instructive to use this perspective to understand the effect of increments. If we use increments, the path corresponding to the sequence is

$$\mathbf{X}_{\text{inc}}(t) = x_i + (t - i)(x_{i+1} - x_i) \text{ for } t \in [i, i+1),$$

while if we do not use increments, the path corresponding to the sequence is

$$\mathbf{X}(t) = \sum_{j=0}^{i-1} x_j + (t - i) x_i \text{ for } t \in [i, i+1).$$

Thus, when we use increments the sequence $\mathbf{x}$ corresponds to vertices of the path $\mathbf{X}_{\text{inc}}$; if we do not, it corresponds to the vectors between vertices of the path $\mathbf{X}$. In practice, this variation corresponds to taking 1_{st} differences of $\mathbf{x}$ before using eq. (4.17).

Zero starting point (`ZeroStart`). The sequence feature map with increments, $\widetilde{\varphi}_{\text{inc}}$, as defined above is *translation-invariant*, meaning $\widetilde{\varphi}_{\text{inc}}(\mathbf{x} + a) = \widetilde{\varphi}_{\text{inc}}(\mathbf{x})$, where $\mathbf{x} + a = (x_0 + a, x_1 + a, \dots, x_k + a)$ for some $a \in \mathbb{R}^d$. In order to remove translation invariance, we can start each sequence at the origin $0 \in \mathbb{R}^d$ by pre-appending a 0 to each sequence. A concise way

to define the resulting *zero started* sequence feature map is

$$\widetilde{\varphi}_{\text{inc,zs}}(\mathbf{x}) = \varphi(\delta_0\mathbf{x})\cdots\varphi(\delta_k\mathbf{x}),$$

where we define $\delta_0\mathbf{x} = x_0$. This is the sequence feature map defined in eq. (4.1). Note that this does not change the sequence feature map if we do not use increments.

Time parameterization. When we relate sequences to piecewise linear paths as described in [265, App. C], we can use the fact that the path signature is invariant under reparameterization, or more generally, tree-like equivalence [111]. In terms of discrete sequences, this includes invariance with respect to 0 elements in the sequence (without increments), and repeated elements in the sequence (with increments). In order to remove this invariance, we can include *time parameterization* by setting

$$\widetilde{\varphi}_{-,\text{tp}}(\mathbf{x}) = \widetilde{\varphi}_{-}(\bar{\mathbf{x}}),$$

where $\bar{\mathbf{x}} = (\bar{x}_0, \ldots, \bar{x}_k) \in \mathsf{Seq}(\mathbb{R}^{d+1})$, with $\bar{x}_i = (i, x_i) \in \mathbb{R}^{d+1}$. This is a simple form of positional encoding, but other encodings are also possible, e.g. sinusoidal waves [275].

Algebra lifting (`AlgOpt`). Throughout this article, we have used the tensor exponential as the algebra lifting. However, we can also scale each level of the lifting independently, and keep these as hyperparameters to optimize. In particular, for a sequence $\mathbf{c} = (c_0, c_1, \ldots) \in \mathbb{R}^{\mathbb{N}}$, we define $\varphi^{\mathbf{c}} : \mathbb{R}^d \to H$ to be

$$\varphi^{\mathbf{c}}(x) = \left(c_m x^{\otimes m}\right)_{m=0}^{\infty},$$

where $1/m!$ is used as initialization for c_m and learned along with the other parameters.

The choice of which variant of the sequence feature map to use depends on which invariance properties are important for the specific problem. In practice, the choice can be learned during the training, which is done in our experiments in Section 4.5. Furthermore, the features obtained through the low-rank hypo-elliptic diffusion depend on three hyperparameters: (1) the length of random walks; (2) the number of low-rank functionals; (3) the maximal tensor degree; (4) the number of iterations (layers). Note that the first three hyperparameters can also potentially vary across different iterations.

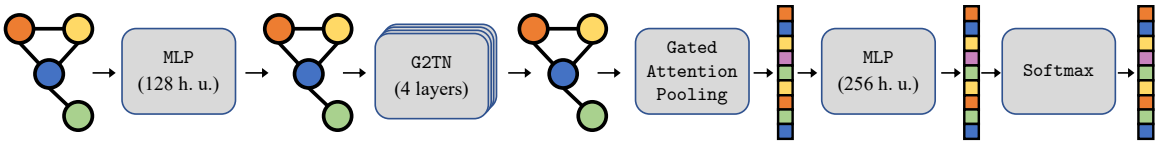

Figure 4.1: Visualization of the architecture used for NCI1 and NCI109 described in Section 4.5.

4.D Experiments

We have implemented the low-rank algorithm for our layers given in Theorem 4.3 using `Tensorflow`, `Keras` and `Spektral` [103]. Code is available at `https://github.com/tgcsaba/graph2tens`. All experiments were run on one of 3 computing clusters that were overall equipped with 5 NVIDIA Geforce 2080Ti, 2 Quadro GP100, 2 A100 GPUs. The largest GPU memory allocation any one of the experiments required was around ∼5GB. For the experiments ran using different random seeds, the seed was used to control the randomness in the (1) data splitting process, (2) parameter initialization, (3) optimization procedure. For an experiment with overall n_{runs} number of runs, the used seeds were $\{0, \dots, n_{\text{runs}} - 1\}$.

4.D.1 Model details.

The architecture described in Section 4.5 is visualized on Figure 4.1, where for `G2T(A)N` the `G2TN` stack is ceteris paribus replaced by the attentional variation. To further the discussion, here we provide more details on the model implementation.

Initialization. In our `G2TN` and `G2T(A)N` layers, we linearly project tensor-valued features using linear functionals that use the same low-rank parameterization as the *recursive* variation in [263, App. D.2]. There, the authors also propose in App. D.5 an initialization heuristic for setting their variances, that we have employed with a centered uniform distribution.

Regularization. We also apply ℓ_2 regularization for both experiments in Sections 4.5 and 4.E. Given a max. tensor degree $M \geq 1$ and a max. tensor rank $R \geq 1$ representing the layer width, there are overall RM rank-1 linear functionals of the form

$$\boldsymbol{\ell}_m^r = u_{M-m+1}^r \otimes \cdots \otimes u_M^r \quad \text{for } m = 1, \dots M \text{ and } r = 1, \dots, R. \tag{4.18}$$

Since in practice these are represented using only the component vectors $u_i^r \in \mathbb{R}^d$, a naive approach would compute the 2-norm of each component vector u_i^r to penalize the loss function. However, we found this approach to underperform compared to the following. Although our algorithm represents the functionals using a rank-1 decomposition and computes the projection of

Table 4.2: Accuracies computed over 5 seeds of G2T(A)N ablated by ceteris paribus changing a single model setting.

DATASET	NoDiff	NoZeroStart	NoAlgOpt	NoJK	NoSkip	NoNorm	AvgPool
NCI1	80.1 ± 0.7	79.5 ± 1.8	81.6 ± 1.6	82.1 ± 1.8	81.8 ± 0.9	81.6 ± 1.5	82.4 ± 0.9
NCI109	78.2 ± 1.2	77.7 ± 1.8	77.5 ± 1.3	77.6 ± 1.2	78.3 ± 1.3	79.8 ± 1.4	77.6 ± 1.3

Table 4.3: Accuracies computed over 5 seeds of G2TN ablated by ceteris paribus changing a single model setting.

DATASET	NoDiff	NoZeroStart	NoAlgOpt	NoJK	NoSkip	NoNorm	AvgPool
NCI1	79.1 ± 1.5	80.7 ± 0.6	78.9 ± 1.3	79.4 ± 1.9	81.0 ± 1.3	79.5 ± 1.1	80.3 ± 1.7
NCI109	77.8 ± 1.2	79.5 ± 1.5	76.5 ± 1.7	78.1 ± 1.9	77.0 ± 1.6	77.4 ± 2.7	77.6 ± 1.8

each tensor-valued node feature without explicitly building high-dimensional tensors, conceptually we still have tensors ($\boldsymbol{\ell}_m^r$) acting on tensors ($\Phi(i)$ for $i \in \mathcal{V}$), and hence the tensor norm, $\left\|\boldsymbol{\ell}_m^r\right\|_2$, should be used for regularization. Fortunately, this can also be computed efficiently:

$$\left\|\boldsymbol{\ell}_m^r\right\|_2 = \left\|u_{M-m+1}^r \otimes \cdots \otimes u_M^r\right\|_2 = \left\|u_{M-m+1}^r\right\|_2 \cdots \left\|u_M^r\right\|_2 .$$

Further, as is common, we replace the ℓ_2 norm with its squared value, sum over all functionals in (4.18), and multiply by the regularization parameter $\lambda > 0$ so the final penalty is

$$\texttt{L2Penalty} = \lambda \sum_{r=1}^{R} \sum_{m=1}^{M} \left\|u_{M-m+1}^r\right\|_2^2 \cdots \left\|u_M^r\right\|_2^2 .$$

4.E Details on experiments.

Here we extend the experimental results demonstrated in Section 4.5.

4.E.1 Ablation Studies

Further to using attention, we give ideas for variations on our models in Appendix 4.C which can be summarized briefly as: (i) using increments of node attributes (Diff), (ii) preprending a zero point to sequences (ZeroStart), (iii) optimizing over the algebra embedding (AlgOpt), all of which are built into our main models. Further, the previous architectural choices aimed at incorporating several commonly used tricks for training GNNs. We investigate the effect of the previous variations and ablate the architectural "tricks" by measuring the performance change resulting from ceteris paribus removing it. Table 4.2 shows the result for G2T(A)N . To summarize the main observations, the model is robust to all the architectural changes, removing

the layer norm even improves on NCI109. Importantly, replacing the attention pooling with mean pooling does not significantly affect the performance, but actually slightly improves on NCI1. Regarding variations, `AlgOpt` slightly improves on both datasets, while removing `Diff` and `ZeroStart` significantly degrades the accuracy on NCI1. The latter means that translations of node attributes are important, not just their distances.

We give the analogous ablation study for the `G2TN` model in Table 4.3, and compare the derived conclusions between the attentional and attention-free versions. First, we discuss the layer variations. Similarly to `G2T(A)N` , `NoDiff` slightly decreases the accuracy. However the conclusions regarding `NoZeroStart` and `NoAlgOpt` are different. In this case, removing `ZeroStart` actually improves the performance, while on `G2T(A)N` the opposite was true. An interpretation of this phenomenon is that only the attention mapping that is used to learn the random walks required information about translations of the layer's features, and not the tensor-features themselves. Another difference is that `NoAlgOpt` degrades the accuracy more significantly for `G2TN` . A possible explanation is that since `G2T(A)N` layers are more flexible thanks to their use of attention, they rely less on being able to learn the algebraic lift, while as `G2TN` layers are more rigid in their random walk definition, and benefit more from the added flexibility of `AlgOpt`. Additionally, it seems that `G2TN` is more sensitive to the various architectural options, and removing any of them, i.e. `NoJK`, `NoSkip`, `NoNorm` or `AvgPool`, degrades the accuracy by 1% or more on at least one dataset. Intuitively, it seems that overall the `G2T(A)N` model is more robust to the various architectural "tricks", and more adaptable due to its ability to learn the random walk.

4.E.2 Further Experiments

Citation datasets. Additionally to transductive learning on the biological datasets, we have carried out inductive learning tasks on some of the common citation datasets, i.e. Cora, Citeseer [237] and Pubmed [196]. We follow [240] in carrying out the experiment, and use the largest connected component for each dataset with 20 training examples per class, 30 validation examples per class, and the rest of the data used for testing. The hyperparameters of our models and optimization procedure were based on the settings of the GAT model in [240, Table 4], which we have slightly fine-tuned on Cora and used for the other datasets. In particular, a single layer of `G2TN` or `G2T(A)N` is used with 64 functionals, max. tensor degree 2 and random walk length 5. The dropout rate was tuned to 0.9, while the attentional dropout was set to 0.3 in `G2T(A)N` . Optimization is carried out with Adam [143], a fixed learning rate of 0.01 and ℓ_2 regularization strength 0.01. Training is stopped once the validation loss does not improve for 50 epochs, and restored to the best checkpoint. For both of our models, `NoDiff` is used that we found to improve on the results as opposed to using increments of node features. The dropout rate had to

Table 4.4: Accuracies of our models on the citation datasets computed over 100 seeds compared with the 4 consistently best performing baselines from [240].

DATASET	GCN	GAT	MONET	GRAPHSAGE (MEAN)	G2TN (OURS)	G2T(A)N (OURS)
CORA	81.5 ± 1.3	81.8 ± 1.3	81.3 ± 1.3	79.2 ± 7.7	$\mathbf{82.6 \pm 1.0}$	82.0 ± 1.1
CITESEER	$\mathbf{71.9 \pm 1.9}$	71.1 ± 1.9	71.2 ± 2.0	71.6 ± 1.9	69.4 ± 1.0	68.2 ± 1.3
PUBMED	77.8 ± 2.9	78.7 ± 2.3	78.6 ± 2.3	77.4 ± 2.2	$\mathbf{78.8 \pm 1.9}$	78.0 ± 1.9

Table 4.5: Accuracies of our models on k-hop sanitized citation datasets computed over 100 seeds compared with the 5 consistently best performing models from [213].

K	DATASET	GCN	GAT	G-U-NET	HGNET-EP	HGNET-L	G2TN (OURS)	G2T(A)N (OURS)
1	CORA	76.7	78.5	78.1	77.2	77.1	$\mathbf{81.8 \pm 1.3}$	80.9 ± 1.4
	CITESEER	64.2	66.4	63.0	64.3	64.10	$\mathbf{68.1 \pm 1.3}$	66.6 ± 1.4
	PUBMED	75.8	75.9	75.8	77.0	76.3	$\mathbf{78.7 \pm 1.9}$	77.5 ± 2.0
2	CORA	72.0	73.4	74.4	74.0	75.4	$\mathbf{79.4 \pm 2.8}$	78.3 ± 2.9
	CITESEER	58.3	59.4	57.3	57.8	59.9	$\mathbf{63.6 \pm 1.8}$	62.2 ± 1.9
	PUBMED	72.1	73.1	72.4	72.9	75.1	$\mathbf{77.1 \pm 1.7}$	76.3 ± 1.6

be tuned as high as 0.9, which suggests very strong overfitting, hence the additional complexity of `AlgOpt` was also contrabeneficial, and `NoAlgOpt` was used. As such, each model employs a single hidden layer, which is followed by layer normalization that was found to perform slightly better than other normalizations, e.g. graph-level normalization.

The results of our models over 100 runs are reported in Table 4.4 compared with the 4 consistently best performing baselines on these datasets from [240], i.e. GCN [145], GAT [276], MoNet [191], and GraphSage [112] with a mean aggregator. Firstly on Cora, both `G2TN` and `G2T(A)N` outperform the baselines with a more significant improvement for `G2TN` . For CiteSeer, our models are left somewhat behind compared to the baselines in terms of accuracy. Finally, they are again competitive on Pubmed, where `G2TN` takes the top score with a very slight lead. Two consistent observations are: (1) `G2TN` and `G2T(A)N` have a lower variance than all baselines, (2) `G2TN` consistently outperforms `G2T(A)N` . The latter may be attributed to the observation that due to the severe overfitting on these datasets, the additional complexity of the attention mechanism in `G2T(A)N` is unhelpful for generalization.

***K*-hop Sanitized Splits.** Recent work [213] has demonstrated that it is possible to make the previously considered citation datasets more suitable for testing the ability of a model to learn long-range information by dropping node labels in a structured way. Concretely, they use a label resampling strategy to guarantee that if a node is selected for a data split, none of its k-th degree neighbours are included in any splits, i.e. training, validation nor testing, allowing to reduce the effect of short-range "label imprinting". In practice, we select a maximal independent set from the graph with respect to the k-th power of the adjacency matrix with self-loops, and repeat the

Table 4.6: Computation time of one forward pass of one G2TN layer in milliseconds on a Nvidia GeForce 2080TI GPU for a graph with varying nodes (N) and edges (E), while node feature dimension is fixed at 128.

	$E = 5000$	$E = 10000$	$E = 20000$	$E = 40000$	$E = 80000$
$N = 500$	37	61	98	151	293
$N = 1000$	37	62	104	152	290
$N = 2000$	39	63	89	160	293
$N = 4000$	39	63	109	160	294
$N = 8000$	42	65	108	156	295

previous experiment with the same data splitting method, model choice and training settings as before. The experiment seed also controls the random maximal independent set that is selected.

The results of our models trained on the citation datasets sanitized this way are available in Table 4.5 computed over 100 seeds for $k = 1, 2$. As baseline results, we compare against the 5 best performing models from [213], where various GNN depths were also considered for each model, and we use the *best* reported result for each of the baselines. Overall, we can observe that all baseline models exhibit a very sharp drop in performance as k is increased, while for G2TN and G2T(A)N , the performance decrease is not nearly as pronounced. For both $k = 1$ and $k = 2$, our models perform better than the baselines on all datasets. This is explained by the fact that due to using random walks length of 5, the models can efficiently pick up on information outside of the sanitized neighbourhoods. As previously, G2TN performs better than G2T(A)N as the additional flexibility of the attention layer does not lead to improved generalization performance when severe overfitting is present. This experiment demonstrates that the proposed models efficiently pick up on long-range information within larger neighbourhoods, and it suggests that on inductive learning tasks they should be more robust to sparse labeling rates compared to common short-range GNN models.

4.E.3 Computation Time

Table 4.6 shows the computation time of one forward pass of a G2TN layer with various graphs. This empirically demonstrates that our layer does not depend on the number of nodes, and only depends linearly on the number of edges as expected from the theoretical complexity.

Table 4.7 shows the computation time of one forward pass of a G2TN layer with a fixed graph and various model parameters. Empirically, our layer scales roughly linearly in both the walk length k and the maximum tensor level M. Linear complexity in walk length is expected, but the linearity in the maximum tensor level M is due to parallelization of the algorithm presented in Thm. 4.3. This is discussed in the pseudocode section of [265, App. F].

Table 4.7: Computation time of one forward pass of one G2TN layer in milliseconds on a Nvidia GeForce 2080TI GPU for a fixed graph with $N = 2000$ nodes and $E = 10000$ edges. The walk length (k) and the maximum level M of the model are varied.

	$k=2$	$k=4$	$k=6$	$k=8$	$k=10$
$M=1$	14	26	45	59	70
$M=2$	20	40	72	81	101
$M=3$	26	55	95	115	163
$M=4$	34	73	101	150	188
$M=5$	42	92	139	187	237

4.E.4 Parameter Count

In both the NCI datasets, the G2TN and G2T(A)N have 505k and 519k trainable parameters respectively. Thus, our models are comparable in size to the GraphTrans, which is reported to have 500k trainable parameters [294].

4.E.5 Summary of Dataset Parameters

We provide a summary of dataset statistics and the model parameters in the experiments. Here, M denotes the maximum tensor degree, k denotes the walk length, and R denotes the maximum rank of the low-rank functions. Concretely, R is the number of node features used within the network, and is analogous to the width of a neural network. For the computation of the diameter of a disconnected graph, we take the largest diameter of any connected component.

Table 4.8: Dataset statistics and G2TN model parameters used.

	GRAPHS	TOTAL NODES	TOTAL EDGES	AVG. DIAM.	CLASSES	LAYERS	M	k	R
NCI1	4110	122K	132K	13.3	2	4	2	5	128
NCI109	4127	122K	132K	13.1	2	4	2	5	128
CORA	1	2485	5069	19	7	4	2	5	64
CITESEER	1	2110	3668	28	6	4	2	5	64
PUBMED	1	19K	44K	18	3	4	2	5	64

Chapter 5

Random Fourier Signature Features

5.1 Introduction

Machine learning has successfully been applied to tasks that require learning from complex and structured data types on non-Euclidean domains. Feature engineering on such domains is often tackled by exploiting the geometric structure and symmetries existing within the data [25]. Learning from sequential data (such as video, text, audio, time series, health data, etc.) is a classic, but an ongoing challenge due to the following properties:

- *Non-Euclidean data.* The data domain is nonlinear since there is no obvious and natural way of adding sequences of different length.
- *Time-space patterns.* Statistically significant patterns can be distributed over time and space, that is, capturing the order structure in which "events" arise is crucial.
- *Time-warping invariance.* The meaning of many sequences is often invariant to reparameterization also frequently called time-warping, at least to an extent; e.g. a sentence spoken quicker or slower contains (essentially) the same information.
- *Discretization and irregular sampling.* Sequences often arise by sampling along an irregularly spaced grid of an underlying continuous time process. A general methodology should be robust as the sampling gets finer, sequences approximate paths (continuous-time limit), or as the discretization grid varies between sequences.
- *Scalability.* Sequence datasets can quickly become massive, so the computational complexity should grow subquadratically, in terms of all of the state-space dimension, and the length and number of sequences.

The signature kernel k_S is the state-of-the-art kernel for sequential data [266, 228, 165] that addresses the first 4 of the above questions and can rely on the modular and powerful framework of kernel learning [234]. Its construction is motivated by classic ideas from stochastic analysis that give a structured description of a sequence by developing it into a series of tensors. We refer to [163] for a recent overview of its various constructions and applications. In the real-world, various phenomena are well-modelled by systems of differential equations. The path signature arises naturally in the context of controlled differential equations. The role of the signature here is to provide a basis for the effects of a driving signal on systems of controlled differential equations. In essence, it captures the interactions of a controlling signal with a nonlinear system. This explains the widespread applicability of signatures to various problems across the sciences [182]. There is also geometric intuition behind signatures, see Section 1.2.4 in [42].

Features vs Kernel/Primal vs Dual. Kernel learning circumvents the costly evaluation of a high- or infinite-dimensional feature map by replacing it with the computation of a Gram matrix which contains as entries the inner products of features between all pairs of data points. This can be very powerful since the inner product evaluation can often be done cheaply by the celebrated "kernel trick", even for infinite-dimensional feature spaces, but the price is that now the computational cost is quadratic in the number of samples, and downstream algorithms further often incur a cubic cost usually in the form of a matrix inversion. On the other hand, when finite-dimensional features can used for learning, the primal formulation of a learning algorithm can perform training and inference in a cost that is linear with respect to the sample size assuming that the feature dimension is fixed. This motivates the investigation of finite-dimensional approximations to kernels that mimic their expressiveness at a lower computational cost. It is an interesting question how the feature dimension should scale with the dataset size to maintain a given (optimal) learning performance in downstream tasks, which is investigated for instance by [219, 30, 251, 170, 246, 156].

Computational Cost of the Signature kernel. In the context of the signature kernel, one data point is itself a whole sequence. Hence, given a data set $\mathbf{X}$ consisting of $N \in \mathbb{Z}_+$ sequences where each sequence $\mathbf{x} = (\mathbf{x}_1, \dots, \mathbf{x}_{L_x})$ is of maximal length $L_{\mathbf{x}} \leq \ell \in \mathbb{Z}_+$ and has sequence entries $\mathbf{x}_i$ in a state-space of dimension d, then the existing algorithms to evaluate the Gram matrix of the k_S scale quadratically, i.e. as $O(N^2\ell^2 d)$, both in sequence length ℓ and number of sequences N. So far this has only been addressed by subsampling (either directly the sequence elements to reduce the length or by column subsampling via the Nyström approach [287]), which can lead to crude approximations and performance degradation on large-scale datasets.

Contribution. Random Fourier Features (RFF) [209] is a classic technique to enjoy both the benefits of the primal and dual approach. Here, a low-dimensional and random feature map is constructed, which although does not approximate the feature map of a translation-invariant kernel, its inner product is with high probability close to the kernel itself. The main contribution of this article is to carry out such a construction for the signature kernel. Concretely, we construct a random feature map on the domain of sequences called Random Fourier Signature Features (RFSF), such that its inner product is a random kernel $\tilde{k}_S$ for sequences that is both (i) an unbiased estimator for k_S, and (ii) has analogous probabilistic approximation guarantees to the classic RFF kernel. The challenge is that a direct application of the classic RFF technique is not feasible since this relies on Bochner's theorem which does not apply since the sequence domain is not even a linear space and the feature domain is non-Abelian, which makes the use of (generalizations of [88]) Bochner's theorem difficult due to the lack of sufficiently explicit representations. We tackle this challenge by combining the algebraic structure of signatures with probabilistic concentration arguments; a careful analysis of the error propagation yields uniform concentration guarantees similar to the RFF on $\mathbb{R}^d$. Then, we introduce dimensionality reduction techniques for random tensors further approximating $\tilde{k}_S$ to define the extremely scalable variants $\tilde{k}_S^{DP}$ and $\tilde{k}_S^{TRP}$ called RFSF-DP and RFSF-TRP saving considerable amounts of computation time and memory by low-dimensional projection of the feature set of the RFSF.

Hence, analogously to the classic RFF construction, the random kernels $\tilde{k}_S, \tilde{k}_S^{DP}, \tilde{k}_S^{TRP}$ simultaneously enjoy the expressivity of an infinite-dimensional feature space as well as linear complexity in sequence length. This overcomes the arguably biggest drawback of the signature kernel, which is the quadratic complexity in sample size and sequence length; the price for reducing the complexities by an order is that this approximation only holds with high probability. As in the case of the classic RFF, our experiments show that this is in general a very attractive tradeoff. Concretely, we demonstrate in the experiments that the proposed random features (1) provide comparable performance on moderate sized datasets to full-rank (quadratic time) signature kernels, (2) outperform other random feature approaches for time series on both moderate- and large-scale datasets, (3) allow scaling to datasets of a million time series.

Related Work. The signature kernel has found many applications; for example, it is used in ABC-Bayes [76], economic scenario validation [8], amortised likelihood estimation [77], the analysis of RNNs [82], analysis of trajectories in Lie groups [162], metrics for generative modelling [27, 141], or dynamic analysis of topological structures [96]. For a general overview see [163]. All of these applications can benefit from a faster computation of the signature kernel with theoretical guarantees. Previous approaches address the quadratic complexity of the signature kernel only by subsampling in one form or another: [146] combine a structured

Nyström type-low-rank approximation to reduce complexity in dimension of samples and sequence length, [266] combine this with inducing point and variational methods, [228] uses sequence-subsampling, [165] use diagonal approximations to Gram matrices in a variational setting. Related to this work is also the random nonlinear projections in [180]; further, [192] combine linear dimension projection in a general pipeline and [55] use signatures in reservoir computing. Directly relevant for this work is recent progress on tensorized random projections [250, 212]. Random Fourier Features [209, 211] are well-understood theoretically [253, 244, 246, 172, 10, 255, 35, 271, 35]. In particular, its generalization properties are studied in e.g. [12, 170, 251, 156], where it is shown that the feature dimension need only scale sublinearly in the dataset size for supervised learning, and a similar result also holds for kernel principal component analysis [246]. Several variations have been proposed over the years [159, 81, 49, 300, 48, 47, 50], even finding applications in deep learning [256]. Alternative random feature approaches for polynomial and Gaussian kernels based on tensor sketching have been proposed in e.g. [281, 280, 279]. Gaussian sketching has also been applied in the RKHS for kernel approximation [152]. For a survey, the reader is referred to [35, 173].

Outline. Section 5.2 provides background on the prerequisites of our work: Random Fourier Features, and Signature Features/Kernels. Section 5.3 contains our proposed methods with theoretical results; it introduces Random Fourier Signature Features (RFSF) $\tilde{\varphi}_{S_{\leq M}}$, RFSF kernels $\tilde{k}_{S_{\leq M}}$ (where $M \in \mathbb{Z}_+$ is the truncation level introduced later), and most importantly their theoretical guarantees. Theorem 5.3 quantifies the approximation $k_{S_{\leq M}}(\mathbf{x}, \mathbf{y}) \approx \tilde{k}_{S_{\leq M}}(\mathbf{x}, \mathbf{y})$ uniformly. Then, we discuss additional variants: the RFSF-DP kernel $\tilde{k}^{DP}_{S_{\leq M}}$ and the RFSF-TRP kernel $\tilde{k}^{TRP}_{S_{\leq M}}$, which build on the previous construction using dimensionality reduction with corresponding concentration results in Theorems 5.6 and 5.9. Section 5.4 compares the performance of the proposed scalable signature kernels against popular approaches on SVM multivariate time series classification, which demonstrates that the proposed kernel not only significantly improves the computational complexity of the signature kernel, it also provides comparable performance, and in some cases even improvements in accuracy as well. Hence, we take the best of both worlds: linear batch, sequence, and state-space dimension complexities, while approximately enjoying the expressivity of an infinite-dimensional RKHS with high probability.

5.2 Background

5.2.1 Notation

We denote the real numbers by $\mathbb{R}$, natural numbers by $\mathbb{N} = \{0, 1, 2, \ldots\}$, positive integers by $\mathbb{Z}_+ = \{1, 2, 3, \ldots\}$, the range of positive integers from 1 to $n \in \mathbb{Z}_+$ by $[n] = \{1, 2, \ldots, n\}$. Given $a, b \in \mathbb{R}$, we denote their maximum by $a \vee b = \max(a, b)$ and their minimum by $a \wedge b = \min(a, b)$. We define the collection of all ordered m-tuples with non-repeating entries by

$$\Delta_m(n) = \{1 \leq i_1 < i_2 < \cdots < i_m \leq n : i_1, i_2, \ldots, i_n \in [n]\}. \tag{5.1}$$

In general, $\mathcal{X}$ refers to a subset of the input domain, where the various objects are defined, generally taken to be a subset $\mathbb{R}^d$ unless otherwise stated. For a vector $\mathbf{x} \in \mathbb{R}^d$, we denote its ℓ_p norm by $\|\mathbf{x}\|_p = \left(\sum_{i=1}^d |x_i|^p\right)^{1/p}$. For a matrix $\mathbf{A} \in \mathbb{R}^{d \times e}$, we denote the spectral and the Frobenius norm by $\|\mathbf{A}\|_2 = \sup_{\|\mathbf{x}\|_2 = 1} \|\mathbf{A}\mathbf{x}\|_2$ and $\|\mathbf{A}\|_F = \left(\sum_{i=1}^e \|\mathbf{A}\mathbf{e}_i\|_2^2\right)^{1/2}$, where $\{\mathbf{e}_1, \ldots, \mathbf{e}_e\}$ is the canonical basis of $\mathbb{R}^e$. The transpose of a matrix $\mathbf{A}$ is denoted by $\mathbf{A}^\top$. For a differentiable $f : \mathbb{R}^d \to \mathbb{R}$, we denote its gradient at $\mathbf{x} \in \mathbb{R}^d$ by $\nabla f(\mathbf{x}) = (\partial f(\mathbf{x}) / \partial x_i)_{i=1}^d$, and its collection of partial derivatives with respect to $\mathbf{s} = (x_{i_1}, \ldots, x_{i_k})$ by $\partial_{\mathbf{s}} f(\mathbf{x}) = \left(\partial f(\mathbf{x}) / \partial x_{i_j}\right)_{j=1}^k$.

$\mathsf{Seq}(\mathcal{X})$ refers to sequences of finite, but unbounded length with values in the set $\mathcal{X}$:

$$\mathsf{Seq}(\mathcal{X}) = \{\mathbf{x} = (\mathbf{x}_0, \ldots, \mathbf{x}_L) : \mathbf{x}_i \in \mathcal{X}, L \in \mathbb{Z}_+\}.$$

We denote the effective length of a sequence $\mathbf{x} = (\mathbf{x}_0, \ldots, \mathbf{x}_L) \in \mathsf{Seq}(\mathcal{X})$ by $\mathsf{L}_{\mathbf{x}} = L$, and define the 1^{st}-order forward differencing operator as $\delta \mathbf{x}_i = \mathbf{x}_{i+1} - \mathbf{x}_i$. We define the 1-variation functional of a sequence $\mathbf{x} \in \mathsf{Seq}(\mathcal{X})$ as $\|\mathbf{x}\|_{1\text{-var}} = \sum_{i=1}^{\ell_{\mathbf{x}}-1} \|\delta \mathbf{x}_i\|_2$ as a measure of sequence complexity.

5.2.2 Signature Features for Sequential Data

A challenge in machine learning when constructing feature maps for datasets of sequences is that the sequence length can vary from instance to instance; the space of sequences $\mathsf{Seq}(\mathcal{X}) = \left\{(\mathbf{x}_i)_{i=1}^\ell : \mathbf{x}_0, \ldots, \mathbf{x}_\ell \in \mathcal{X} \text{ and } \ell \in \mathbb{Z}_+\right\}$ includes sequences of various lengths, and they should all get mapped to the same feature space, while preserving the information about the ordering of elements. A concatenation property of path signatures called Chen's identity [182, Thm. 2.9] turns concatentation into multiplication provides a principled approach to construct features for sequences. Below we recall the construction from Chapter 3 of the discrete-time order-1 signature features originally introduced in Section 1.2.6.

The key idea is to define the discrete-time signature of 1-step increments, and then glue features together by algebra multiplication to guarantee the Chen identity holds by construction. Now assume we are given a static feature map $\varphi : \mathcal{X} \to \mathcal{H}_\mathsf{k}$ into some Hilbert space $\mathcal{H}_\mathsf{k}$. Our task is to construct from this feature map for elements of $\mathcal{X}$, a feature map for sequences of arbitrary length in $\mathcal{X}$. A natural first step is to apply the feature map φ elementwise to a sequence $\mathbf{x} \in \mathsf{Seq}(\mathcal{X})$ to lift it into the feature space $\mathcal{H}_\mathsf{k}$ of φ, $\varphi(\mathbf{x}) = (\varphi(\mathbf{x}_i))_{i=0}^{\mathsf{L_x}} \in \mathsf{Seq}(\mathcal{H}_\mathsf{k})$. The challenge is now to construct a feature map for sequences in $\mathcal{H}_\mathsf{k}$. Simple aggregation of the individual features fails; e.g. summation of the individual features $\varphi(\mathbf{x}_i)$ would lose order information, vectorization $\big(\varphi(\mathbf{x}_0), \ldots, \varphi(\mathbf{x}_{\mathsf{L_x}})\big) \in {\mathcal{H}_\mathsf{k}}^{\mathsf{L_x}}$ makes sequences of different length not comparable. It turns out that multiplication is well-suited for this task in a suitable algebra.

Fortunately, there is a natural way to embed any Hilbert space $\mathcal{H}_\mathsf{k}$ into a larger Hilbert space $\mathsf{T}(\mathcal{H})$ that is also a non-commutative algebra, see Sec. 1.1.2. First, we take the 1^{st} differences,

$$\mathbf{x} \mapsto \delta\varphi(\mathbf{x}) = (\varphi(\mathbf{x}_i) - \varphi(\mathbf{x}_{i-1}))_{i=1}^{\mathsf{L_x}} \in {\mathcal{H}_\mathsf{k}}^{\mathsf{L_x}}, \quad \text{where } \mathbf{x} \in \mathsf{Seq}(\mathcal{X}) \tag{5.2}$$

since it is more natural to keep track of changes rather than absolute values. Then we identify $\mathcal{H}_\mathsf{k}$ as a subset of $\mathsf{T}(\mathcal{H})$. The simplest choice given the above construction of $\mathsf{T}(\mathcal{H})$ is

$$\iota : \mathbf{h} \mapsto (1, \mathbf{h}, \mathbf{0}, \mathbf{0}, \ldots) \in \mathsf{T}(\mathcal{H}) \quad \text{where } \mathbf{h} \in \mathcal{H}_\mathsf{k}. \tag{5.3}$$

A direct calculation shows that composing the maps (5.2), (5.3), and multiplying the individual entries in $\mathsf{T}(\mathcal{H})$ results in a sequence summary using all non-contiguous subsequences, since in each multiplication step a sequence entry is either selected once or not at all. This gives rise to the discretized signatures $\varphi_\mathsf{S} : \mathsf{Seq}(\mathcal{X}) \to \mathsf{T}(\mathcal{H})$ for $\mathbf{x} \in \mathsf{Seq}(\mathcal{X})$ with $\mathsf{L_x} \geq 2$:

$$\varphi_\mathsf{S}(\mathbf{x}) = \prod_{i=1}^{\mathsf{L_x}} \iota(\delta\varphi(\mathbf{x}_i)) = \left(\sum_{\mathbf{i} \in \Delta_m(\mathsf{L_x})} \delta\varphi(\mathbf{x}_{i_1}) \otimes \cdots \otimes \delta\varphi(\mathbf{x}_{i_m}) \right)_{m \geq 0}, \tag{5.4}$$

where $\Delta_m : \mathbb{Z}_+ \to \mathbb{Z}_+^m$ is as defined in (5.1) and $\mathbf{i} = (i_1, \ldots, i_m)$. Thus, the sequence feature is itself a sequence, however, now a sequence of tensors indexed by their degree $m \in \mathbb{N}$ in contrast to being indexed by the time index $i \in [\mathsf{L_x}]$. These sequence features are invariant to a natural transformation of time series called time-warping, but can also be made sensitive to it by including time as an extra coordinate with the mapping $\mathbf{x} = (\mathbf{x}_i)_{i=1}^{\mathsf{L_x}} \mapsto (t_i, \mathbf{x}_i)_{i=1}^{\mathsf{L_x}}$. It also possesses similar approximation properties to path signatures in Section (1.2.2), i.e. uniform approximation of functions of sequences on compact sets; see Appendices A and B in [263].

Despite the abstract derivation, the resulting feature map φ_S is—in principle—explicitly

computable when $\mathcal{H}_\mathsf{k} = \mathbb{R}^d$; see [142] for details. However, when the static feature map φ is high- or infinite-dimensional, this is not feasible and we discuss a kernel trick further below.

Remark 5.1. *We used the map ι, as defined in* (5.3)*, to embed $\mathcal{H}_\mathsf{k}$ into* $\mathrm{T}(\mathcal{H})$*. Other choices are possible, for example one could use the embedding $\hat{\iota} : \mathcal{H}_\mathsf{k} \to \mathrm{T}(\mathcal{H})$ for $\mathbf{h} \in \mathcal{H}_\mathsf{k}$*

$$\hat{\iota}(\mathbf{h}) = \left(1, \mathbf{h}, \frac{\mathbf{h}^{\otimes 2}}{2!}, \frac{\mathbf{h}^{\otimes 3}}{3!}, \dots\right) \in \mathrm{T}(\mathcal{H}).$$

This embedding is actually the classical choice in mathematics, but different choices of the embedding lead to, besides potential improvements in benchmarks, mildly different computational complexities and interesting algebraic questions [69, 263, 265].

Finally, it can be useful to only consider the first $M \in \mathbb{Z}_+$ tensors in the series $\varphi_\mathsf{S}(\mathbf{x})$ analogously to using the first M moments in classic polynomial regression to avoid overfitting. Hence, we define the M-truncated signature features for $M \in \mathbb{Z}_+$ as

$$\varphi_{\mathsf{S}_{\leq M}}(\mathbf{x}) = \big(1, \varphi_{\mathsf{S}_1}(\mathbf{x}), \dots, \varphi_{\mathsf{S}_M}(\mathbf{x}), \mathbf{0}, \mathbf{0}, \dots\big) \quad \text{for } \mathbf{x} \in \mathsf{Seq}(\mathcal{X}),$$

where $\varphi_{\mathsf{S}_m}(\mathbf{x})$ is the projection of $\varphi_\mathsf{S}(\mathbf{x})$ onto ${\mathcal{H}_\mathsf{k}}^{\otimes m}$. In practice, we regard $M \in \mathbb{Z}_+$, and the choice of the embedding as hyperparameters to optimize. From now on, we will focus on the order-1 case, but we note that all algorithms can be generalized fully to the higher-order case.

Signature Kernels. The signature is a powerful feature set for nonlinear regression on paths and sequences. A computational bottleneck associated with it is the dimensionality of the feature space $\mathrm{T}(\mathcal{H})$. As we are dealing with tensors, for $\mathcal{H}_\mathsf{k}$ finite-dimensional $\varphi_{\mathsf{S}_m}(\mathbf{x})$ is a tensor of degree-m which has $(\dim \mathcal{H}_\mathsf{k})^m$ coordinates that need to be computed. This can quickly become computationally expensive. For infinite-dimensional $\mathcal{H}_\mathsf{k}$, e.g. when $\mathcal{H}_\mathsf{k}$ is a reproducing kernel Hilbert space (RKHS), which is one of the most interesting settings due to the modelling flexibility, it is infeasible to directly compute φ_S. In Section 1.2.6, we discuss a kernel trick allows to compute the inner product of discretized signature features (5.4) up to a given degree $M \in \mathbb{Z}_+$ using dynamic programming, even when $\mathcal{H}_\mathsf{k}$ is infinite-dimensional. As mentioned before, here we focus on discrete-time, and our starting point is the approach of order-1 discretized signature kernel, which is often referred to as a non-geometric approximation to the signature kernel [69]. For the continuous-time treatment of signature kernels, see Section 1.2.4.

Above we described a generic way to turn a static feature map $\varphi : \mathcal{X} \to \mathcal{H}_\mathsf{k}$ into a feature map $\varphi_{\mathsf{S}_{\leq M}}(\mathbf{x})$ for sequences, see (5.4). The signature kernel is a powerful formalism that allows to transform any static kernel on $\mathcal{X}$ into a kernel for sequences that evolve in $\mathcal{X}$. Let $\mathsf{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$

be a continuous and bounded kernel, and from now on, let $\mathcal{H}_{\mathsf{k}}$ denote its RKHS, and $\varphi(\mathbf{x}) = \mathsf{k}_{\mathbf{x}} \equiv \mathsf{k}(\mathbf{x}, \cdot)$ the associated reproducing kernel lift for $\mathbf{x} \in \mathcal{X}$. We define the M-truncated (discretized) signature kernel $\mathsf{k}_{\mathsf{S}_{\leq M}} : \mathsf{Seq}(\mathcal{X}) \times \mathsf{Seq}(\mathcal{X}) \to \mathbb{R}$ for $M \in \mathbb{Z}_+$ as

$$\begin{aligned}\mathsf{k}_{\mathsf{S}_{\leq M}}(\mathbf{x}, \mathbf{y}) &= \left\langle \varphi_{\mathsf{S}_{\leq M}}(\mathbf{x}), \varphi_{\mathsf{S}_{\leq M}}(\mathbf{y}) \right\rangle_{\mathrm{T}(\mathcal{H})} = \sum_{m=0}^{M} \left\langle \varphi_{\mathsf{S}_m}(\mathbf{x}), \varphi_{\mathsf{S}_m}(\mathbf{y}) \right\rangle_{\mathcal{H}_{\mathsf{k}}^{\otimes m}} \\ &= \sum_{m=0}^{M} \mathsf{k}_{\mathsf{S}_m}(\mathbf{x}, \mathbf{y}) = \sum_{m=0}^{M} \sum_{\substack{\mathbf{i} \in \Delta_m(\mathsf{L}_{\mathbf{x}}) \\ \mathbf{j} \in \Delta_m(\mathsf{L}_{\mathbf{y}})}} \delta_{i_1, j_1} \mathsf{k}(\mathbf{x}_{i_1}, \mathbf{y}_{j_1}) \cdots \delta_{i_m, j_m} \mathsf{k}(\mathbf{x}_{i_m}, \mathbf{y}_{j_m}), \end{aligned} \tag{5.5}$$

where we defined the level-m (discretized) signature kernel $\mathsf{k}_{\mathsf{S}_m} : \mathsf{Seq}(\mathcal{X}) \times \mathsf{Seq}(\mathcal{X}) \to \mathbb{R}$ for $m \in [M]$ as $\mathsf{k}_{\mathsf{S}_m}(\mathbf{x}, \mathbf{y}) = \left\langle \varphi_{\mathsf{S}_m}(\mathbf{x}), \varphi_{\mathsf{S}_m}(\mathbf{y}) \right\rangle_{\mathcal{H}_{\mathsf{k}}^{\otimes m}}$, and $\delta_{i,j}$ denotes 2^{nd}-order cross-differencing such that $\delta_{i,j}\mathsf{k}(\mathbf{x}_i, \mathbf{y}_j) = \mathsf{k}(\mathbf{x}_{i+1}, \mathbf{y}_{j+1}) - \mathsf{k}(\mathbf{x}_{i+1}, \mathbf{y}_j) - \mathsf{k}(\mathbf{x}_i, \mathbf{y}_{j+1}) + \mathsf{k}(\mathbf{x}_i, \mathbf{y}_j)$ for $i \in [\mathsf{L}_{\mathbf{x}}]$ and $j \in [\mathsf{L}_{\mathbf{y}}]$.

Although (5.5) looks expensive to compute, [146] applies dynamic programming to efficiently compute $\mathsf{k}_{\mathsf{S}_{\leq M}}$ using a recursive algorithm. Importantly, (5.5) avoids computing tensors, and only depends on the entry-wise evaluations of the static kernel $\mathsf{k}(\mathbf{x}_i, \mathbf{y}_j)$. Indeed, this leads to a computational cost of $O((M + d)\mathsf{L}_{\mathbf{x}}\mathsf{L}_{\mathbf{y}})$, which is feasible for sequences evolving in high-dimensional state-spaces, but only with moderate sequence length. Note that the same bottleneck applies to PDE-based approaches. In part, the aim of this article is to alleviate this quadratic cost in sequence length, while approximately enjoying the modelling capability of working within an infinite-dimensional RKHS.

5.2.3 Random Fourier Features

Kernel methods allow to implicitly use an infinite-dimensional feature map $\varphi : \mathcal{X} \to \mathcal{H}_{\mathsf{k}}$ by evaluation of the inner product $\mathsf{k}(\mathbf{x}, \mathbf{y}) = \langle \varphi(\mathbf{x}), \varphi(\mathbf{y}) \rangle_{\mathcal{H}_{\mathsf{k}}}$, when $\mathcal{H}$ is a Hilbert space. This inner product can often be evaluated without direct computation of $\varphi(\mathbf{x})$ and $\varphi(\mathbf{y})$ via the kernel trick. Although this makes them a powerful tool due to the resulting flexibility, the price of this flexibility is a trade-off in complexity with respect to the number of samples $N \in \mathbb{Z}_+$. Disregarding the price of evaluating the kernel $\mathsf{k}(\mathbf{x}, \mathbf{y})$ momentarily, kernel methods require the computation of a Gram matrix with $O(N^2)$ entries, that further incurs an $O(N^3)$ computational cost by most downstream algorithms, such as KRR [239], GP [216], and SVM [234]. Several techniques reduce this complexity, and the focal point of this article is the Random Fourier Feature (RFF) technique of [209, 210, 211], which can be applied to any continuous, bounded, translation-invariant kernel on $\mathbb{R}^d$.[1] Throughout, we write with some abuse of notation $\mathsf{k}(\mathbf{x} - \mathbf{y}) \equiv \mathsf{k}(\mathbf{x}, \mathbf{y})$.

[1] A kernel is called translation-invariant if $\mathsf{k}(\mathbf{x}, \mathbf{y}) = \mathsf{k}(\mathbf{x} + \mathbf{z}, \mathbf{y} + \mathbf{z})$ for any $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{R}^d$.

A corollary of Bochner's theorem [220] is that any continuous, bounded, and translation-invariant kernel $\mathsf{k} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ can be represented as the Fourier transform of a non-negative finite measure Λ called the spectral measure associated to k, i.e. for $\mathbf{x}, \mathbf{y} \in \mathcal{X}$

$$\mathsf{k}(\mathbf{x}-\mathbf{y}) = \int_{\mathbb{R}^d} \exp(i\mathbf{w}^\top(\mathbf{x}-\mathbf{y}))\,\mathrm{d}\Lambda(\mathbf{w}).$$

We may, without loss of generality, assume that Λ is a probability measure such that $\Lambda(\mathbb{R}^d) = 0$, which amounts to working with the kernel $\mathsf{k}(\mathbf{x}-\mathbf{y})/\mathsf{k}(\mathbf{0})$. [209] proposed to draw $\tilde{d} \in \mathbb{Z}_+$ i.i.d. samples from Λ, $\mathbf{w}_1, \ldots, \mathbf{w}_{\tilde{d}} \overset{\text{i.i.d.}}{\sim} \Lambda$, to define the random feature map for $\mathbf{x} \in \mathcal{X}$ by

$$\tilde{\varphi} : \mathcal{X} \to \tilde{\mathcal{H}}_{\mathsf{k}} = \mathbb{R}^{2\tilde{d}}, \quad \tilde{\varphi}(\mathbf{x}) = \frac{1}{\sqrt{\tilde{d}}}\left(\cos\left(\mathbf{W}^\top\mathbf{x}\right), \sin\left(\mathbf{W}^\top\mathbf{x}\right)\right), \tag{5.6}$$

where $\mathbf{W} = (\mathbf{w}_i)_{i=1}^{\tilde{d}} \in \mathbb{R}^{d\times\tilde{d}}$. Then, the corresponding random kernel is defined for $\mathbf{x}, \mathbf{y} \in \mathcal{X}$ as

$$\tilde{\mathsf{k}} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}, \quad \tilde{\mathsf{k}}(\mathbf{x}, \mathbf{y}) = \langle \tilde{\varphi}(\mathbf{x}), \tilde{\varphi}(\mathbf{y}) \rangle_{\tilde{\mathcal{H}}_{\mathsf{k}}} = \frac{1}{\tilde{d}} \sum_{i=1}^{\tilde{d}} \cos\left(\mathbf{w}_i^\top(\mathbf{x}-\mathbf{y})\right) \tag{5.7}$$

to provide a probabilistic approximation to k. Indeed, it is a straightforward exercise to check that $\mathsf{k}(\mathbf{x}, \mathbf{y}) = \mathbb{E}\left[\tilde{\mathsf{k}}(\mathbf{x}, \mathbf{y})\right] \approx \tilde{\mathsf{k}}(\mathbf{x}, \mathbf{y})$. This approximation converges exponentially fast in $\tilde{d}$ and uniformly over compact subsets of $\mathbb{R}^d$ as proven in [209, Claim 1]. This bound was later tightened and extended to the derivatives of the kernel in the series of works [244, 255, 35], and we provide an adapted version under Theorem 5.26 in the supplement.

5.3 Methodology

The goal of this section is to build random features for sequences, that enjoy the benefit of linear sequence length and low-dimensional feature complexity with theoretical guarantees that the corresponding inner product is close to the M-truncated (discretized) signature kernel $\mathsf{k}_{\mathsf{S}_{\leq M}}$ with high probability. We construct these random features in a two step process: firstly, we reduce the feature space from infinite to finite (but high) dimensionality through a careful construction using random Fourier features (RFFs), and in the second step we apply further dimensionality reduction to reduce the complexity to an even lower dimensional space in order to aid in scalability. Although we present this construction as distinct steps, the steps are coupled during the computation, and the features can be computed directly without the initial step.

5.3.1 Random Fourier Signature Features

In Section 5.2, we recalled the RFF construction, which associates to a continuous, bounded, translation-invariant kernel $\mathsf{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ on $\mathcal{X}$ a spectral measure Λ, and approximates k by drawing samples from Λ to define the random features $\tilde{\varphi} : \mathcal{X} \to \tilde{\mathcal{H}}_{\mathsf{k}}$ (5.6), and the random kernel $\tilde{\mathsf{k}} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ (5.7). Afterwards, we presented a generic way to turn any such static features $\tilde{\varphi} : \mathcal{X} \to \tilde{\mathcal{H}}_{\mathsf{k}}$ for elements of $\mathcal{X}$ into sequence features for sequences that evolve in $\mathcal{X}$ via $\varphi_{\mathsf{S}_{\leq M}} : \mathsf{Seq}(\mathcal{X}) \to \mathsf{T}(\mathcal{H})$. Applying this construction with the RFF as feature map on $\mathcal{X}$ would already result in a random feature map for sequences, i.e. a map from $\mathsf{Seq}(\mathcal{X})$ into $\tilde{\mathcal{H}}_{\mathsf{kS}}$. Taking the inner product in $\tilde{\mathcal{H}}_{\mathsf{kS}}$ of this new random feature map for sequences would, however, only yield a biased estimator for the truncated signature kernel $\mathsf{k}_{\mathsf{S}_{\leq M}}$. We correct for this bias by revisiting our previous construction, and build an unbiased approximation to $\mathsf{k}_{\mathsf{S}_{\leq M}}$ using independent RFF copies in each tensor multiplication step. Then, we show in Theorem 5.3 that this random estimator comes with good probabilistic guarantees.

The probabilistic construction procedure is outlined in the following definition.

Definition 5.2. *Let* $\mathbf{W}^{(1)}, \dots, \mathbf{W}^{(M)} \overset{i.i.d.}{\sim} \Lambda^{\tilde{d}}$ *be i.i.d. random matrices sampled from* $\Lambda^{\tilde{d}}$ *for* RFF *dimension* $\tilde{d} \in \mathbb{Z}_+$*, and define the independent* RFF *maps* $\tilde{\varphi}_m : \mathcal{X} \to \tilde{\mathcal{H}}_{\mathsf{k}}$ *as in* (5.6)*, i.e.* $\tilde{\varphi}_m(\mathbf{x}) = \frac{1}{\sqrt{\tilde{d}}}\left(\cos(\mathbf{W}^{(m)\top}\mathbf{x}), \sin(\mathbf{W}^{(m)\top}\mathbf{x})\right)$ *for* $m \in [M]$ *and* $\mathbf{x} \in \mathcal{X}$*. The* M*-truncated Random Fourier Signature Feature (*RFSF*) map* $\tilde{\varphi}_{\mathsf{S}_{\leq M}} : \mathsf{Seq}(\mathcal{X}) \to \tilde{\mathcal{H}}_{\mathsf{kS}}$ *from sequences in* $\mathcal{X}$ *into the free algebra over* $\tilde{\mathcal{H}}_{\mathsf{k}}$ *is defined for truncation level* $M \in \mathbb{Z}_+$ *and* $\mathbf{x} \in \mathsf{Seq}(\mathcal{X})$ *as*

$$\tilde{\varphi}_{\mathsf{S}_{\leq M}}(\mathbf{x}) = \left(\sum_{\mathbf{i} \in \Delta_m(\mathsf{L}_{\mathbf{x}})} \delta\tilde{\varphi}_1(\mathbf{x}_{i_1}) \otimes \cdots \otimes \delta\tilde{\varphi}_m(\mathbf{x}_{i_m}) \right)_{m=0}^{M}. \tag{5.8}$$

Further, the RFSF *kernel* $\tilde{\mathsf{k}}_{\mathsf{S}_{\leq M}} : \mathsf{Seq}(\mathcal{X}) \times \mathsf{Seq}(\mathcal{X}) \to \mathbb{R}$ *can be computed for* $\mathbf{x}, \mathbf{y} \in \mathsf{Seq}(\mathcal{X})$ *as*

$$\begin{aligned} \tilde{\mathsf{k}}_{\mathsf{S}_{\leq M}}(\mathbf{x}, \mathbf{y}) &= \left\langle \tilde{\varphi}_{\mathsf{S}_{\leq M}}(\mathbf{x}), \tilde{\varphi}_{\mathsf{S}_{\leq M}}(\mathbf{y}) \right\rangle_{\tilde{\mathcal{H}}_{\mathsf{kS}}} = \sum_{m=0}^{M} \left\langle \tilde{\varphi}_{\mathsf{S}_m}(\mathbf{x}), \tilde{\varphi}_{\mathsf{S}_m}(\mathbf{y}) \right\rangle_{\tilde{\mathcal{H}}_{\mathsf{k}}^{\otimes m}} \\ &= \sum_{m=0}^{M} \tilde{\mathsf{k}}_{\mathsf{S}_m}(\mathbf{x}, \mathbf{y}) = \sum_{m=0}^{M} \sum_{\substack{\mathbf{i} \in \Delta_m(\mathsf{L}_{\mathbf{x}}) \\ \mathbf{j} \in \Delta_m(\mathsf{L}_{\mathbf{y}})}} \delta_{i_1, j_1} \tilde{\mathsf{k}}_1(\mathbf{x}_{i_1}, \mathbf{y}_{j_1}) \cdots \delta_{i_m, j_m} \tilde{\mathsf{k}}_m(\mathbf{x}_{i_m}, \mathbf{y}_{j_m}), \end{aligned} \tag{5.9}$$

where we defined the level-m RFSF *kernel* $\tilde{\mathsf{k}}_{\mathsf{S}_m} : \mathsf{Seq}(\mathcal{X}) \times \mathsf{Seq}(\mathcal{X}) \to \mathbb{R}$ *for* $m \in \mathbb{N}$ *as* $\tilde{\mathsf{k}}_{\mathsf{S}_m}(\mathbf{x}, \mathbf{y}) = \left\langle \tilde{\varphi}_{\mathsf{S}_m}(\mathbf{x}), \tilde{\varphi}_{\mathsf{S}_m}(\mathbf{y}) \right\rangle_{\tilde{\mathcal{H}}_{\mathsf{k}}^{\otimes m}}$ *with the convention that* $\tilde{\mathsf{k}}_{\mathsf{S}_0} \equiv 1$*, and* $\tilde{\mathsf{k}}_1, \dots, \tilde{\mathsf{k}}_M : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ *are independent* RFF *kernels defined as in* (5.7) *with the random weights* $\mathbf{W}^{(1)}, \dots, \mathbf{W}^{(M)} \in \mathbb{R}^{d \times \tilde{d}}$*.*

Since the feature map $\tilde{\varphi}_{\mathsf{S}_{\leq M}}$ can be directly evaluated in the feature space recursively, $\tilde{\mathsf{k}}_{\mathsf{S}_{\leq M}}$ has linear complexity in the sequence length. However, it requires computing high-dimensional

tensors, where the degree-m component $\tilde{\varphi}_{\mathsf{S}_m}(\mathbf{x}) \in \tilde{\mathcal{H}}_{\mathsf{k}}^{\otimes m}$ has $(\dim \tilde{\mathcal{H}}_{\mathsf{k}})^m = (2\tilde{d})^m$ coordinates, making it infeasible for large $m, \tilde{d} \in \mathbb{Z}_+$. Remark 5.4 discusses the computational complexity in detail. Further, note that the kernel can be evaluated by means of a kernel trick exactly analogously to the evaluation of (5.5), but in this case there are no computational gains compared to the infinite-dimensional signature kernel $\mathsf{k}_{\mathsf{S}_{\leq M}}(\mathbf{x}, \mathbf{y})$.

Next, we provide a theoretical analysis to show that the random kernel $\tilde{\mathsf{k}}_{\mathsf{S}_{\leq M}}(\mathbf{x}, \mathbf{y})$ converges to the ground truth signature kernel $\mathsf{k}_{\mathsf{S}_{\leq M}}(\mathbf{x}, \mathbf{y})$ exponentially fast and uniformly over compact state-spaces $\mathcal{X} \subseteq \mathbb{R}^d$, generalizing the result [209, Claim 2] to this non-Euclidean domain of sequences. Throughout the analysis, we need certain regularity properties of Λ in order to invoke quantitative versions of the law of large numbers, i.e. properties such as boundedness, existence of the moment-generating function, moment-boundedness, or belonging to certain Orlicz spaces of random variables. Boundedness of the spectral measure is too restrictive an assumption, since a continuous, bounded, translation-invariant kernel $\mathsf{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is characteristic if and only if the support of its spectral measure is $\mathbb{R}^d$, see [243, Prop. 8]. Hence, we instead work with the assumption that its moments are well-controllable, i.e. the tails of the distribution are not "too heavy". Specifically, we assume the Bernstein moment condition that

$$\mathbb{E}_{\mathbf{w}\sim\Lambda}\left[w_i^{2m}\right] \leq \frac{m! S^2 R^{m-2}}{2} \quad \text{for all } i \in [d] \tag{5.10}$$

for some $S, R > 0$. We show in the Supplementary Material under Lemmas 5.21 and 5.22, in a more general context, that this is equivalent to Λ being a sub-Gaussian probability measure; see e.g. [24, Sec 2.3] and [277, Sec. 2.5] about sub-Gaussianity. This of course includes the spectral measure of the Gaussian kernel defined for bandwidth $\sigma > 0$ and $\mathbf{x}, \mathbf{y} \in \mathcal{X}$ $\mathsf{k}(\mathbf{x}, \mathbf{y}) = \exp\left(-\|\mathbf{x}-\mathbf{y}\|_2^2 / 2\sigma^2\right)$, which has a Gaussian spectral distribution $\mathbf{w} \sim \mathcal{N}(0, 1/\sigma^2 \mathbf{I}_d)$, and therefore calculation gives $\mathbb{E}_{w\sim\mathcal{N}(0,1/\sigma^2)}\left[w^{2m}\right] = \frac{2^m \Gamma\left(m+\frac{1}{2}\right))}{\sigma^{2m}\sqrt{\pi}} < \frac{m!}{2}\left(\frac{2\sqrt{2}}{\sigma^2 \sqrt[4]{\pi}}\right)^2 \left(\frac{2}{\sigma^2}\right)^{m-2}$, since $\Gamma(m + 1/2) < \Gamma(m+1) = m!$. Hence Λ satisfies condition (5.10) with S, R as given here. Now we state our approximation theorem regarding $\tilde{\mathsf{k}}_{\mathsf{S}_m}$, which quantifies that it is a (sub-)exponentially good estimator of $\mathsf{k}_{\mathsf{S}_m}$ with high probability and uniformly.

Theorem 5.3. *Let $\mathsf{k} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be a continuous, bounded, translation-invariant kernel with spectral measure Λ, which satisfies* (5.10). *Let $\mathcal{X} \subset \mathbb{R}^d$ be compact and convex with diameter $|\mathcal{X}|$, $\mathcal{X}_\Delta = \{\mathbf{x} - \mathbf{y} : \mathbf{x}, \mathbf{y} \in \mathcal{X}\}$. Then, the following quantities are finite: $\sigma_\Lambda^2 = \mathbb{E}_{\mathbf{w}\sim\Lambda}\left[\|\mathbf{w}\|_2^2\right]$, $L = \left\|\mathbb{E}_{\mathbf{w}\sim\Lambda}\left[\mathbf{w}\mathbf{w}^\top\right]\right\|_2^{1/2}$, $E_{i,j} = \mathbb{E}_{\mathbf{w}\sim\Lambda}\left[\left|w_i w_j\right| \|\mathbf{w}\|_2\right]$ and $D_{i,j} = \sup_{\mathbf{z}\in\mathcal{X}_\Delta} \left\|\nabla\left[\frac{\partial^2 \mathsf{k}(\mathbf{z})}{\partial z_i \partial z_j}\right]\right\|_2$ for $i, j \in [d]$.*

Further, for any max. sequence 1*-var* $V > 0$*, and signature level* $m \in \mathbb{Z}_+$*, for* $\epsilon > 0$

$$\mathbb{P}\left[\sup_{\substack{\mathbf{x},\mathbf{y}\in\mathsf{Seq}(\mathcal{X})\\ \|\mathbf{x}\|_{1\text{-var}},\|\mathbf{y}\|_{1\text{-var}}\le V}} \left|\mathsf{k}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y}) - \tilde{\mathsf{k}}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y})\right| \ge \epsilon\right] \le$$
$$\le m \begin{cases} \left(C_{d,\mathcal{X}}\left(\frac{\beta_{d,m,V}}{\epsilon}\right)^{\frac{d}{d+1}} + d\right)\exp\left(-\frac{\tilde{d}}{2(d+1)(S^2+R)}\left(\frac{\epsilon}{\beta_{d,m,V}}\right)^2\right) & \text{for } \epsilon < \beta_{d,m,V} \\ \left(C_{d,\mathcal{X}}\left(\frac{\beta_{d,m,V}}{\epsilon}\right)^{\frac{d}{(d+1)m}} + d\right)\exp\left(-\frac{\tilde{d}}{2(d+1)(S^2+R)}\left(\frac{\epsilon}{\beta_{d,m,v}}\right)^{\frac{1}{m}}\right) & \text{for } \epsilon \ge \beta_{d,m,V}, \end{cases} \tag{5.11}$$

where $C_{d,\mathcal{X}} = 2^{\frac{1}{d+1}}16\,|\mathcal{X}|^{\frac{d}{d+1}}\sum_{i,j=1}^{d}(D_{i,j}+E_{i,j})^{\frac{d}{d+1}}$ *and* $\beta_{d,m,V} = m\left(2V^2\left(L^2\vee 1\right)\left(\sigma_\Lambda^2\vee d\right)\right)^m$.

The proof is provided in the supplement under Theorem 5.36. The result shows that the random kernel $\tilde{\mathsf{k}}_{\mathsf{S}_m}$ approximates the signature kernel $\mathsf{k}_{\mathsf{S}_m}$ uniformly over subsets of $\mathsf{Seq}(\mathcal{X})$ of sequences $\mathbf{x} \in \mathsf{Seq}(\mathcal{X})$ with maximal 1-variation V, $\|\mathbf{x}\|_{1\text{-var}} \le V$, assuming that the state-space $\mathcal{X} \subset \mathbb{R}^d$ is a convex and compact domain. The error bound is analogous to the classic RFF bounds, in the sense that the tail probability decreases exponentially fast as a function of the RFF dimension $\tilde{d}$. The functional form of the bound is inherited from Theorem 5.26, which provides an analogous result for the derivatives of RFF. This link follows from Lemma 5.35, which connects the concentration of the RFSF kernel to the second derivatives of RFF.

The main difference from the classic case, i.e. [209, Claim 1] and Theorem 5.26, is the appearance of $\beta_{d,m,V}$ which controls a regime change in the tail behaviour. Concretely, for $\epsilon < \beta_{d,m,V}$ (5.11) has a polynomial plus a sub-Gaussian tail, while for $\epsilon > \beta_{d,m,V}$ has a $(1/m)$-subexponential tail. This is not surprising as the inner summand in (5.9) is the m-fold tensor product of m independent RFF kernels, which makes the tail heavier exactly by an exponent of $1/m$. The constant itself, $\beta_{d,m,V}$, depends on (i) the maximal sequence 1-variation V, which measures a notion of time-warping invariant sequence complexity; (ii) the Lipschitz constant of the kernel L (see Examples 5.28 and 5.29); (iii) the trace of the second moment of Λ, $\sigma_\Lambda^2 = \mathbb{E}_{\mathbf{w}\sim\Lambda}\left[\|\mathbf{w}\|_2^2\right]$; (iv) the state-space dimension d; (v) and the signature level m itself.

Remark 5.4. *Algorithm 7 demonstrates the computation of the* RFSF *map* $\tilde{\varphi}_{\mathsf{S}_{\le M}}$ *given a dataset of sequences* $\mathbf{X} = (\mathbf{x}_i)_{i=1}^N \subset \mathsf{Seq}(\mathcal{X})$*. Upon inspection, we can deduce that the algorithm has a computational complexity of* $O\left(N\ell(Md\tilde{d}^M)\right)$*. Importantly, it is linear in* ℓ*, the sequence length, although scales polynomially in the* RFF *sample size* $\tilde{d}^M$*, and exponentially in the truncation-*M.

5.3.2 Dimensionality Reduction: Diagonal Projection

Previously, we introduced a featurized approximation $\tilde{\mathsf{k}}_{\mathsf{S}_{\le M}}$ to the signature kernel $\mathsf{k}_{\mathsf{S}_{\le M}}$, called the RFSF kernel, which reduces the computation from the infinite-dimensional RKHS to a finite-

dimensional feature space using random tensors. Although this makes the computation in the feature space viable of the RFSF map $\tilde{\varphi}_{\mathsf{S}_{\leq M}}$, it is still tensor-valued, which incurs a computational cost of $O(\tilde{d} + \tilde{d}^2 + \cdots + \tilde{d}^m)$ in the RFF dimension $\tilde{d} \in \mathbb{Z}_+$. Now, we take another step towards scalability and apply further dimensionality reduction. By examining the structure of these tensors, we introduce a diagonally projected variant called RFSF-DP that considerably reduces their sizes. We emphasize that the above RFSF construction is crucial: it approximates the inner product in an infinite-dimensional space, and now we further approximate it in an even lower dimensional space. The benefit is that one does not have to go through the computation of the initial RFSF map, but only the selected degrees of freedom have to be computed.

As a first observation, we notice that the computation of (5.9) can be reformulated, due to (5.7) and linearity of the differencing operator, in the following way:

$$\tilde{\mathsf{k}}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y}) = \frac{1}{\tilde{d}^m} \sum_{q_1,\ldots,q_m=1}^{\tilde{d}} \sum_{\substack{\mathbf{i}\in\Delta_m(\mathsf{L}_\mathbf{x}) \\ \mathbf{j}\in\Delta_m(\mathsf{L}_\mathbf{y})}} \prod_{p=1}^{m} \delta_{i_p,j_p} \cos\left(\mathbf{w}_{q_p}^{(p)\top}(\mathbf{x}_{i_p} - \mathbf{y}_{j_p})\right) \tag{5.12}$$

by spelling out the definition of the RFF kernel, where $\mathbf{w}_1^{(1)},\ldots,\mathbf{w}_{\tilde{d}}^{(m)} \overset{\text{i.i.d.}}{\sim} \Lambda$, such that $\mathbf{W}^{(p)} = \left(\mathbf{w}_1^{(p)},\ldots,\mathbf{w}_{\tilde{d}}^{(p)}\right) \in \mathbb{R}^{d\times\tilde{d}}$ as defined in Def. 5.2. Now, we may observe that there is a dependency structure among the samples being averaged in (5.12), since the outer summation is over the Cartesian product $(q_1,\ldots,q_m) \in [\tilde{d}]^{\times m}$, which suggests that we might be able to drastically reduce the degrees of freedom by restricting this summation to only go over an independent set of samples. One way to do this is to restrict to multi-indices of the form $\mathcal{I} = \left\{(q,\ldots,q) \in [\tilde{d}]^{\times m} \mid q \in [\tilde{d}]\right\}$, i.e. we diagonally project the index set, motivating the name of the approach stated in the following definition.

Definition 5.5. *Let* $\mathbf{w}_1^{(1)},\ldots,\mathbf{w}_{\tilde{d}}^{(M)} \overset{\text{i.i.d.}}{\sim} \Lambda$ *for* $\tilde{d} \in \mathbb{Z}_+$*, and define* $\hat{\varphi}_{m,q} : \mathcal{X} \to \hat{\mathcal{H}}_\mathsf{k} = \mathbb{R}^2$ *with sample size* $\hat{d} = 1$ *for* $q \in [\tilde{d}]$ *and* $m \in [M]$*, such that* $\hat{\varphi}_{m,q}(\mathbf{x}) = \left(\cos(\mathbf{w}_q^{(m)\top}\mathbf{x}), \sin(\mathbf{w}_q^{(m)\top}\mathbf{x})\right)$ *for* $\mathbf{x} \in \mathcal{X}$*. The* M*-truncated Diagonally Projected Random Fourier Signature Feature (*RFSF-DP*) map* $\tilde{\varphi}^{\mathsf{DP}}_{\mathsf{S}_{\leq M}} : \mathsf{Seq}(\mathcal{X}) \to \tilde{\mathcal{H}}^{\mathsf{DP}}_{\mathsf{kS}} = \bigoplus_{m=0}^{M}\left(\hat{\mathcal{H}}_\mathsf{k}^{\otimes m}\right)^{\tilde{d}}$ *is defined for truncation* $M \in \mathbb{Z}_+$ *and* $\mathbf{x} \in \mathsf{Seq}(\mathcal{X})$ *as*

$$\tilde{\varphi}^{\mathsf{DP}}_{\mathsf{S}_{\leq M}}(\mathbf{x}) = \frac{1}{\sqrt{\tilde{d}}} \left(\left(\sum_{\mathbf{i}\in\Delta_m(\mathsf{L}_\mathbf{x})} \delta\hat{\varphi}_{1,q}(\mathbf{x}_{i_1}) \otimes \cdots \otimes \delta\hat{\varphi}_{m,q}(\mathbf{x}_{i_m}) \right)_{q=1}^{\tilde{d}} \right)_{m=0}^{M}.$$

Then, the RFSF-DP *kernel can be directly computed for* $\mathbf{x},\mathbf{y} \in \mathsf{Seq}(\mathcal{X})$ *via*

$$\tilde{\mathsf{k}}^{\mathsf{DP}}_{\mathsf{S}_{\leq M}}(\mathbf{x},\mathbf{y}) = \left\langle \tilde{\varphi}^{\mathsf{DP}}_{\mathsf{S}_{\leq M}}(\mathbf{x}), \tilde{\varphi}^{\mathsf{DP}}_{\mathsf{S}_{\leq M}}(\mathbf{y}) \right\rangle_{\tilde{\mathcal{H}}^{\mathsf{DP}}_{\mathsf{kS}}} = \sum_{m=0}^{M} \left\langle \tilde{\varphi}^{\mathsf{DP}}_{\mathsf{S}_m}(\mathbf{x}), \tilde{\varphi}^{\mathsf{DP}}_{\mathsf{S}_m}(\mathbf{y}) \right\rangle_{\left(\hat{\mathcal{H}}_\mathsf{k}^{\otimes m}\right)^{\tilde{d}}}$$

$$= \sum_{m=0}^{M} \tilde{\mathsf{k}}^{\mathsf{DP}}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y}) = \frac{1}{\tilde{d}} \sum_{m=0}^{M} \sum_{q=1}^{\tilde{d}} \sum_{\substack{\mathbf{i}\in\Delta_m(\ell_{\mathbf{x}}-1)\\ \mathbf{j}\in\Delta_m(\ell_{\mathbf{y}}-1)}} \delta_{i_1,j_1} \hat{\mathsf{k}}_{1,q}(\mathbf{x}_{i_1},\mathbf{y}_{j_1}) \cdots \delta_{i_m,j_m} \hat{\mathsf{k}}_{m,q}(\mathbf{x}_{i_m},\mathbf{y}_{j_m}) \quad (5.13)$$

where we defined the level-m RFSF-DP *kernel* $\tilde{\mathsf{k}}^{\mathsf{DP}}_{\mathsf{S}_m} : \mathsf{Seq}(\mathcal{X}) \times \mathsf{Seq}(\mathcal{X}) \to \mathbb{R}$ *for* $m \in \mathbb{N}$ *and* $\mathbf{x},\mathbf{y} \in \mathsf{Seq}(\mathcal{X})$ *as* $\tilde{\mathsf{k}}_{\mathsf{S}}(\mathbf{x},\mathbf{y}) = \left\langle \tilde{\varphi}^{\mathsf{DP}}_{\mathsf{S}_m}(\mathbf{x}), \tilde{\varphi}^{\mathsf{DP}}_{\mathsf{S}_m}(\mathbf{y}) \right\rangle_{\left(\mathcal{H}_{\hat{\mathsf{k}}}^{\otimes m}\right)^{\tilde{d}}}$ *with the convention that* $\tilde{\mathsf{k}}^{\mathsf{DP}}_{\mathsf{S}_0} \equiv 1$*, and* $\hat{\mathsf{k}}_{m,q} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ *are independent* RFF *kernels with sample size* $\hat{d} = 1$ *defined for* $\mathbf{x},\mathbf{y} \in \mathcal{X}$ *as* $\hat{\mathsf{k}}_{m,q}(\mathbf{x},\mathbf{y}) = \left\langle \hat{\varphi}_{m,q}(\mathbf{x}), \hat{\varphi}_{m,q}(\mathbf{y}) \right\rangle_{\mathcal{H}_{\hat{\mathsf{k}}}}$ *with the random weights* $\mathbf{w}_q^{(m)} \in \mathbb{R}^d$ *for* $q \in [\tilde{d}], m \in [M]$.

Note that by the definition of the RFF kernels in (5.13), we may substitute that $\hat{\mathsf{k}}_{p,q}(\mathbf{x},\mathbf{y}) = \cos({\mathbf{w}_q^{(p)}}^{\top}(\mathbf{x}-\mathbf{y}))$ for $\mathbf{x},\mathbf{y} \in \mathcal{X}$, so (5.13) is equivalently written for $\mathbf{x},\mathbf{y} \in \mathsf{Seq}(\mathcal{X})$ as

$$\tilde{\mathsf{k}}^{\mathsf{DP}}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y}) = \frac{1}{\tilde{d}} \sum_{q=1}^{\tilde{d}} \sum_{\substack{\mathbf{i}\in\Delta_m(\ell_{\mathbf{x}}-1)\\ \mathbf{j}\in\Delta_m(\ell_{\mathbf{y}}-1)}} \delta_{i_1,j_1} \cos({\mathbf{w}_q^{(1)}}^{\top}(\mathbf{x}_{i_1}-\mathbf{y}_{j_1})) \cdots \delta_{i_m,j_m} \cos({\mathbf{w}_q^{(m)}}^{\top}(\mathbf{x}_{i_m}-\mathbf{y}_{j_m})),$$

which is what we set out to do in the above paragraph; that is, restrict the outer summation onto the diagonal projection of the index set. Another way to look at Definition 5.5 is that the RFSF-DP kernel in (5.13) is constructed by defining $\tilde{d}$ independent RFSF kernels, each with internal RFF sample size $\hat{d} = 1$, and then taking their average; the concatenation of their corresponding features are then the features of the RFSF-DP map. Note that for RFF sample size 1, each RFF map has dimension 2, i.e. $\mathcal{H}_{\hat{\mathsf{k}}} = \mathbb{R}^2$, and hence, the corresponding RFSF kernels have dimension $1+2+\cdots+2^M = (2^{M+1}-1)$, which by concatenation results in the overall dimensionality of the RFSF-DP kernel being $\dim \mathcal{H}_{\tilde{\mathsf{k}}_{\mathsf{S}}}^{\mathsf{TRP}} = \tilde{d}\left(2^{M+1}-1\right)$. This relates to the computational complexity of the RFSF-DP map; for details see Remark 5.7.

Next, we state our concentration result regarding the level-m RFSF-DP kernel $\tilde{\mathsf{k}}^{\mathsf{DP}}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y})$.

Theorem 5.6. *Let* $\mathsf{k} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ *be a continuous, bounded, translation-invariant kernel with spectral measure* Λ*, which satisfies* (5.10)*. Then, for level* $m \in \mathbb{Z}_+$*,* $\mathbf{x},\mathbf{y} \in \mathsf{Seq}(\mathcal{X})$*, and* $\epsilon > 0$

$$\mathbb{P}\left[\left|\tilde{\mathsf{k}}^{\mathsf{DP}}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y}) - \mathsf{k}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y})\right| \geq \epsilon\right] \leq 2\exp\left(-\frac{1}{4}\min\left\{\left(\frac{\sqrt{\tilde{d}}\epsilon}{2C_{d,m,\mathbf{x},\mathbf{y}}}\right)^2, \left(\frac{\tilde{d}\epsilon}{\sqrt{8}C_{d,m,\mathbf{x},\mathbf{y}}}\right)^{\frac{1}{m}}\right\}\right),$$

where $L = \left\|\mathbb{E}_{\mathbf{w}\sim\Lambda}\left[\mathbf{w}\mathbf{w}^{\top}\right]\right\|$ *is the Lipschitz constant of* k*, and* $C_{d,m,\mathbf{x},\mathbf{y}} > 0$ *is bounded by*

$$C_{d,m,\mathbf{x},\mathbf{y}} \leq \sqrt{8}e^4(2\pi)^{1/4}e^{1/24}(4e^3\left\|\mathbf{x}\right\|_{1\text{-var}}\left\|\mathbf{y}\right\|_{1\text{-var}}/m)^m\left((2d\max(S,R))^m + \left(L^2/\ln 2\right)^m\right).$$

The proof is provided in the supplement under Theorem 5.37. The result shows that the RFSF-DP kernel converges for any two sequences $\mathbf{x}, \mathbf{y} \in \mathsf{Seq}(\mathcal{X})$ with a $(1/m)$-subexponential convergence rate with respect to the sample size $\tilde{d} \in \mathbb{Z}_+$. Similarly to Theorem 5.3, the bound has a phase transition, where for small values of ϵ, it has a sub-Gaussian tail, while for larger values, it has a $(1/m)$-subexponential tail. A crucial difference from the previous bound is that now the phase transition happens at $\epsilon^\star = C_{d,m,\mathbf{x},\mathbf{y}} 2^{\frac{2m-3/2}{2m-1}} \tilde{d}^{\frac{1-m}{2m-1}}$, which depends on the sample size $\tilde{d}$. This means that for fixed value of $\epsilon > 0$, the phase transition always happens eventually as $\tilde{d}$ gets large enough, hence the convergence rate with respect to $\tilde{d}$ is $(1/m)$-subexponential regardless of the value of ϵ. The slightly reduced rate of convergence compared to the RFSF kernel in Theorem 5.3 is to be expected, since the sample size of the RFSF-DP kernel is analogously reduced by an exponent of $(1/m)$ with respect to $\tilde{d}$ in comparison. The constant $C_{d,m,\mathbf{x},\mathbf{y}}$, similarly to (5.11), depends on (i) the 1-variation of sequences $\|\mathbf{x}\|_{1\text{-var}}, \|\mathbf{y}\|_{1\text{-var}}$ that measure the complexity of the sequences; (ii) $L > 0$, the Lipschitz constant of the kernel k (see Examples 5.28, 5.29); (iii) the moment bound parameters $S, R > 0$ from condition (5.10); (iv) the state-space dimension d; and (v) the signature level m.

Remark 5.7. *Algorithm 8 demonstrates the computation of the* RFSF-DP *map* $\tilde{\varphi}^{\mathsf{DP}}_{\mathsf{S}_{\leq M}}$ *given a dataset of sequences* $\mathbf{X} = (\mathbf{x}_i)_{i=1}^N \subset \mathsf{Seq}(\mathcal{X})$. *Upon counting the operations, we deduce that the algorithm has a computational complexity* $O\left(N\ell\tilde{d}(Md + 2^M)\right)$. *Crucially, it is linear in both* ℓ, *the maximal sequence length, and* $\tilde{d}$, *the sample size of the random kernel.*

5.3.3 Dimensionality Reduction: Tensor Random Projection

Previously, we built the RFSF-DP map by subsampling an independent set from the samples that constitute RFSF kernel. Here, we propose an alternative dimensionality reduction technique that starts again from the RFSF map, and uses random projections to project this generally high-dimensional tensor onto a lower dimension. Random projections are a classic technique in data science for reducing the data dimension, while preserving its important structural properties. They are built upon the celebrated Johnson-Lindenstrauss lemma [135], which states that a set of points in a high-dimensional space can be embedded into a space of much lower dimension, while approximately preserving their geometry. Exploiting this property, we construct a tensor random projected (TRP) variant of our random kernel called RFSF-TRP, such that the computation is coupled between the RFSF and TRP maps, similarly to a kernel trick.

Tensorized random projections [250, 212] construct random projections for tensors with concise parameterization that respects their tensorial nature. Given tensors $\mathbf{s}, \mathbf{t} \in \left(\mathbb{R}^d\right)^{\otimes m}$ for $m \in \mathbb{Z}_+$, the TRP map with CP (CANDECOMP/PARAFAC [148]) rank-1 is built via a random functional $\mathsf{Pr} : \left(\mathbb{R}^d\right)^{\otimes m} \to \mathbb{R}$ such that $\mathsf{Pr}(\mathbf{s}) = \langle \mathbf{p}_1 \otimes \cdots \otimes \mathbf{p}_m, \mathbf{s} \rangle_{\left(\mathbb{R}^d\right)^{\otimes m}}$, where $\mathbf{p}_1, \ldots, \mathbf{p}_m \overset{\text{i.i.d.}}{\sim} \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$

are d-dimensional component vectors sampled from a standard normal distribution. Then, the inner product can be estimated as $\mathsf{Pr}(\mathbf{s})\mathsf{Pr}(\mathbf{t}) \approx \mathbb{E}[\mathsf{Pr}(\mathbf{s})\mathsf{Pr}(\mathbf{t})] = \langle \mathbf{s}, \mathbf{t} \rangle_{\left(\mathbb{R}^d\right)^{\otimes m}}$. Variance reduction is achieved by stacking $n \in \mathbb{Z}_+$ such random projections, each with i.i.d. component vectors $\mathbf{p}_1^{(1)}, \dots, \mathbf{p}_m^{(n)} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$. Hence, the TRP operator is defined as

$$\mathsf{TRP} : \left(\mathbb{R}^d\right)^{\otimes m} \to \mathbb{R}^n, \quad \mathsf{TRP}(\mathbf{s}) = \frac{1}{\sqrt{n}} \left(\left\langle \mathbf{p}_1^{(i)} \otimes \cdots \otimes \mathbf{p}_m^{(i)}, \mathbf{s} \right\rangle \right)_{i=1}^{n}. \tag{5.14}$$

On the one hand, this allows to represent the random projection map onto $\mathbb{R}^n$ using only $O(nmd)$ parameters as opposed to the $O(nd^m)$ parameters in a densely parametrized random projection; and on the other, it allows for downstream computations to exploit the low-rank structure of the operator, as we shall do so in the definition stated below.

Definition 5.8. *Let $\mathbf{W}^{(1)}, \dots, \mathbf{W}^{(M)} \overset{i.i.d.}{\sim} \Lambda^{\tilde{d}}$ be i.i.d. random matrices sampled from $\Lambda^{\tilde{d}}$ for* RFF *dimension $\tilde{d} \in \mathbb{Z}_+$, define the independent* RFF *maps $\tilde{\varphi}_m : \mathcal{X} \to \tilde{\mathcal{H}}_{\mathsf{k}}$ as in* (5.6)*, i.e. $\tilde{\varphi}_m(\mathbf{x}) = 1/\sqrt{\tilde{d}} \left(\cos(\mathbf{W}^{(m)\top}\mathbf{x}), \sin(\mathbf{W}^{(m)\top}\mathbf{x}) \right)$ for $m \in [M]$ and $\mathbf{x} \in \mathcal{X}$, and let $\mathbf{P}^{(1)}, \dots, \mathbf{P}^{(M)} \overset{i.i.d.}{\sim} \mathcal{N}^{\tilde{d}}\left(\mathbf{0}, \mathbf{I}_{2\tilde{d}}\right)$ be random matrices with i.i.d. standard normal entries. The M-truncated Tensor Random Projected Random Fourier Signature Feature (*RFSF-TRP*) map $\tilde{\varphi}^{\mathsf{TRP}}_{\mathsf{S}_{\leq M}} \mathsf{Seq}(\mathcal{X}) \to \tilde{\mathcal{H}}_{\mathsf{k}\mathsf{S}}^{\mathsf{TRP}} = \mathbb{R}^{M\tilde{d}}$ is defined for truncation level $M \in \mathbb{Z}_+$ and $\mathbf{x} \in \mathsf{Seq}(\mathcal{X})$ as*

$$\begin{aligned}
\tilde{\varphi}^{\mathsf{TRP}}_{\mathsf{S}_{\leq M}}(\mathbf{x}) &= \frac{1}{\sqrt{\tilde{d}}} \left(\left(\sum_{\mathbf{i} \in \Delta_m(\mathsf{L}_\mathbf{x})} \left\langle \mathbf{p}_q^{(1)}, \delta\tilde{\varphi}_1(\mathbf{x}_{i_1}) \right\rangle \cdots \left\langle \mathbf{p}_q^{(m)}, \delta\tilde{\varphi}_m(\mathbf{x}_{i_m}) \right\rangle \right)_{q=1}^{\tilde{d}} \right)_{m=0}^{M} \\
&= \frac{1}{\sqrt{\tilde{d}}} \left(\sum_{\mathbf{i} \in \Delta_m(\mathsf{L}_\mathbf{x})} \left(\mathbf{P}^{(1)\top} \delta\tilde{\varphi}_1(\mathbf{x}_{i_1}) \right) \odot \cdots \odot \left(\mathbf{P}^{(m)\top} \delta\tilde{\varphi}_m(\mathbf{x}_{i_m}) \right) \right)_{m=0}^{M},
\end{aligned} \tag{5.15}$$

where $\mathbf{P}^{(m)} = \left(\mathbf{p}_q^{(m)}\right)_{q=1}^{\tilde{d}} \in \mathbb{R}^{2\tilde{d} \times \tilde{d}}$, and $\odot$ denotes the Hadamard product[2]*. The* RFSF-TRP *kernel $\tilde{\mathsf{k}}^{\mathsf{TRP}}_{\mathsf{S}_{\leq M}} : \mathsf{Seq}(\mathcal{X}) \times \mathsf{Seq}(\mathcal{X}) \to \mathbb{R}$ can then be directly computed for sequences $\mathbf{x}, \mathbf{y} \in \mathsf{Seq}(\mathcal{X})$ by*

$$\begin{aligned}
\tilde{\mathsf{k}}^{\mathsf{TRP}}_{\mathsf{S}_{\leq M}}(\mathbf{x}, \mathbf{y}) &= \left\langle \tilde{\varphi}^{\mathsf{TRP}}_{\mathsf{S}_{\leq M}}(\mathbf{x}), \tilde{\varphi}^{\mathsf{TRP}}_{\mathsf{S}_{\leq M}}(\mathbf{y}) \right\rangle_{\tilde{\mathcal{H}}_{\mathsf{k}\mathsf{S}}} = \sum_{m=0}^{M} \left\langle \tilde{\varphi}^{\mathsf{TRP}}_{\mathsf{S}_m}(\mathbf{x}), \tilde{\varphi}^{\mathsf{TRP}}_{\mathsf{S}_m}(\mathbf{y}) \right\rangle_{\tilde{\mathcal{H}}_{\mathsf{k}}{}^{\otimes m}} \\
&= \sum_{m=0}^{M} \tilde{\mathsf{k}}^{\mathsf{TRP}}_{\mathsf{S}_m}(\mathbf{x}, \mathbf{y}) = \frac{1}{\tilde{d}} \sum_{m=0}^{M} \sum_{q=1}^{\tilde{d}} \sum_{\substack{\mathbf{i} \in \Delta_m(\mathsf{L}_\mathbf{x}) \\ \mathbf{j} \in \Delta_m(\mathsf{L}_\mathbf{y})}} \prod_{p=1}^{m} \left\langle \mathbf{p}_q^{(p)}, \delta\tilde{\varphi}_p(\mathbf{x}_{i_p}) \right\rangle \left\langle \mathbf{p}_q^{(p)}, \delta\tilde{\varphi}_p(\mathbf{y}_{j_p}) \right\rangle,
\end{aligned} \tag{5.16}$$

where we defined the level-m RFSF-TRP *kernel $\tilde{\mathsf{k}}^{\mathsf{TRP}}_{\mathsf{S}_m} : \mathsf{Seq}(\mathcal{X}) \times \mathsf{Seq}(\mathcal{X}) \to \mathbb{R}$ for $m \leq M$ as $\tilde{\mathsf{k}}^{\mathsf{TRP}}_{\mathsf{S}_m}(\mathbf{x}, \mathbf{y}) = \left\langle \tilde{\varphi}^{\mathsf{TRP}}_{\mathsf{S}_m}(\mathbf{x}), \tilde{\varphi}^{\mathsf{TRP}}_{\mathsf{S}_m}(\mathbf{y}) \right\rangle_{\tilde{\mathcal{H}}_{\mathsf{k}}{}^{\otimes m}}$ with the convention that $\tilde{\mathsf{k}}^{\mathsf{TRP}}_{\mathsf{S}_0} \equiv 1$.*

[2]The Hadamard product stands for component-wise multiplication of the vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, $\mathbf{x} \odot \mathbf{y} = (x_i y_i)_{i=1}^n$.

We remark that (5.15) is equivalent to the TRP operator (5.14) applied to the RFSF map (5.8) by exploiting bilinearity of the inner product, and using that it factorizes over the tensor components, as described in (1.1). Then, the unbiasedness of (5.16) follows from the fact that the TRP operator is an isometry under expectation, which is applied to the RFSF tensor $\tilde{\varphi}_{\mathsf{S}_m}$, therefore $\tilde{\mathsf{k}}^{\mathsf{TRP}}_{\mathsf{S}_m}$ kernel is conditionally an unbiased estimator of $\tilde{\mathsf{k}}_{\mathsf{S}_m}$ given the RFSF weights $\mathbf{W}^{(1)}, \dots, \mathbf{W}^{(m)} \in \mathbb{R}^{d \times \tilde{d}}$. By the tower rule for expectations, $\tilde{\mathsf{k}}^{\mathsf{TRP}}_{\mathsf{S}_m}$ is an unbiased estimator of $\mathsf{k}_{\mathsf{S}_m}$. The approximation quality is then governed by two factors: (i) how well the TRP projected kernel $\tilde{\mathsf{k}}^{\mathsf{TRP}}_{\mathsf{S}_m}$ approximates $\tilde{\mathsf{k}}_{\mathsf{S}_m}$; (ii) the quality of the approximation of $\tilde{\mathsf{k}}_{\mathsf{S}_m}$ with respect to $\mathsf{k}_{\mathsf{S}_m}$. Note that (ii) has already been discussed in Theorem 5.3. Here, we state the following theoretical result which quantifies (i). Combining these two results by means of triangle inequality and union bounding quantifies that $\tilde{\mathsf{k}}^{\mathsf{TRP}}_{\mathsf{S}_m}$ is a good estimator of $\mathsf{k}_{\mathsf{S}_m}$.

Theorem 5.9. *Let* $\mathsf{k} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ *be a continuous, bounded, translation-invariant kernel with spectral measure* Λ*, which satisfies* (5.10)*. Then, the following bound holds for* RFSF-TRP *kernel for signature level* $m \in \mathbb{Z}_+$ *sequences* $\mathbf{x}, \mathbf{y} \in \mathsf{Seq}(\mathcal{X})$ *and* $\epsilon > 0$

$$\mathbb{P}\left[\left|\tilde{\mathsf{k}}^{\mathsf{TRP}}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y}) - \tilde{\mathsf{k}}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y})\right| \geq \epsilon\right] \leq C_{d,\Lambda} \exp\left(-\left(\frac{m^2 \tilde{d}^{\frac{1}{2m}} \epsilon^{\frac{1}{m}}}{2\sqrt{2} e^3 R \left\|\mathbf{x}\right\|_{1\text{-var}} \left\|\mathbf{y}\right\|_{1\text{-var}}}\right)^{\frac{1}{2}}\right), \quad (5.17)$$

where the absolute constant is defined as $C_{d,\Lambda} = 2\left(1 + \frac{S}{2R} + \frac{S^2}{4R^2}\right)^d$.

The proof is given in the supplement under Theorem 5.38 utilizing the hypercontractivity of Gaussian polynomials [134] that is used to quantify the concentration of the TRP estimator. The concentration of the RFSF-TRP kernel is then governed by Theorems 5.3 and 5.9 combined. Together, they show that for smaller values of ϵ (i.e. the regime change as discussed below Theorem 5.3), the probability has a polynomial plus a sub-Gaussian tail, while for large ϵ, it has a $\left(\frac{1}{2m}\right)$-subexponential tail due to (5.17), and the dominant convergence rate with respect to $\tilde{d}$ is $\left(\frac{1}{4m}\right)$-subexponential. This means that in terms of convergence, RFSF-TRP is the slowest among the 3 variations introduced so far. However, it is also the most efficient in terms of overall dimension, hence downstream computational complexity as well, since $\tilde{\mathcal{H}}^{\mathsf{TRP}}_{\mathsf{k}_{\mathsf{S}}} = \mathbb{R}^{M\tilde{d}}$. Remark 5.10 discusses the computational complexity in detail.

Remark 5.10. *Algorithm 9 demonstrates the computation of the* RFSF-TRP *map* $\tilde{\varphi}^{\mathsf{TRP}}_{\mathsf{S}_{\leq M}}$ *given a dataset of sequences* $\mathbf{X} = (\mathbf{x}_i)_{i=1}^N \subset \mathsf{Seq}(\mathcal{X})$*. Counting the operations, here we can deduce that the algorithm has an* $O\left(MN\ell\tilde{d}(d+\tilde{d})\right)$ *computational complexity. This variation is also linear in* ℓ*, the maximal sequence length, although it is quadratic in* $\tilde{d}$*.*

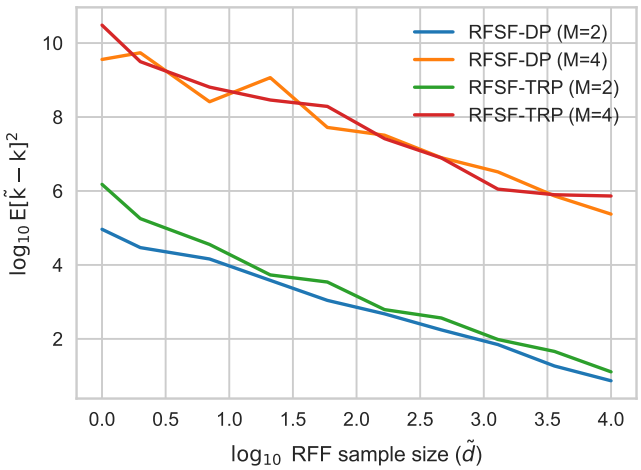

Figure 5.1: Approximation error of random kernels against RFF size on log-log plot.

5.3.4 Numerical evaluation

Here, we numerically evaluate the approximation error of the proposed scalable kernels, that is, RFSF-DP and RFSF-TRP. We do not include RFSF since its dimensionality shows polynomial explosion in the base sample size $\tilde{d}$ due to its tensor-based representation, which makes its computation infeasible for reasonable values of $\tilde{d}$. We generate d-dimensional synthetic time series of length-ℓ using a VAR(1) process $\tilde{\mathbf{x}} \in \mathrm{Seq}(\mathcal{X})$, such that $\tilde{\mathbf{x}}_0 = \mathbf{0}$ and $\tilde{\mathbf{x}}_{t+1} = {}^1/\sqrt{d} A \tilde{\mathbf{x}}_t + \epsilon_t$, where $A \sim \mathcal{N}^{d \times d}(0, 1)$ and $\epsilon_t \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_d)$. Then, we compute the normalized version $\mathbf{x} \in \mathrm{Seq}(\mathcal{X})$ of $\tilde{\mathbf{x}}$, which is rescaled to have 1-variation $V > 0$, i.e. $\mathbf{x}_t = V \tilde{\mathbf{x}}_t / \|\tilde{\mathbf{x}}\|_{1\text{-var}}$. We set $d = 10$, $\ell = 100$, $\sigma = 0.1$ and $V = 100$. We compute the squared deviation between the groundtruth signature kernel and the randomized approximations for two randomly sampled time series in this way. This process is repeated for 100 randomly sampled time series and 100 times resampled random kernel evaluations, giving rise to overall 10000 evaluations for each value of $\tilde{d}$. Figure 5.1 shows the average approximation error plotted against values of $\tilde{d}$ on a log-log plot. We can observe that both RFSF-DP and RFSF-TRP have approximately the same error curves for a given value of truncation level M, and the steepness appears to be the same across different levels of M. This means that RFSF-TRP is slightly more efficient in terms of dimensionality, since its dimension is $M\tilde{d}$ as opposed to $2^{M+1}\tilde{d}$ in RFSF-DP. We also observe that both curves are close to being linear, which indicates that the approximation error scales approximately as $O(\tilde{d}^{-\alpha})$ for some value of $\alpha > 0$.

Table 5.1: Computational complexities of kernels in our experiments; $N \in \mathbb{Z}_+$ is the number of time series, $\ell \in \mathbb{Z}_+$ is their length, $d \in \mathbb{Z}_+$ is their state-space dimension, $M \in \mathbb{Z}_+$ is the signature truncation level, $\tilde{d} \in \mathbb{Z}_+$ is the RF dimension, $W \in \mathbb{Z}_+$ is the warping length in RWS.

RFSF-DP	RFSF-TRP	KSig	KSigPDE	RWS	GAK	RBF	RFF
$O\big(N\ell\tilde{d}\big(Md+2^M\big)\big)$	$O\big(N\ell M\tilde{d}\big(d+\tilde{d}\big)\big)$	$O\big(N^2\ell^2(M+d)\big)$	$O\big(N^2\ell^2 d\big)$	$O(N\ell W d)$	$O\big(N^2\ell^2 d\big)$	$O\big(N^2\ell d\big)$	$O\big(N\ell d\tilde{d}\big)$

5.4 Experiments

5.4.1 Time series classification

We perform multivariate time series classification to investigate the performance of the scalable RFSF variants compared to the full-rank signature kernel and other quadratic time baseline kernels, and further, to demonstrate the scalability to large-scale datasets, where the quadratic sample complexity becomes prohibitive. We use support vector machine (SVM) [248] classification for classifying multivariate time series on datasets of various sizes. For quadratic time kernels, the dual SVM formulation is used, while for kernels with feature representations, we use the primal formulation that has linear complexity in the size of the dataset $n \in \mathbb{Z}_+$ aiding in scalability to truly large-scale datasets. For each considered kernel/feature, we use a GPU-based implementation provided in the KSig library[3] [264]. For large-scale experiments with the featurized kernels, linear SVM implementation is used from the cuML library [215], while the dual SVM on moderate-scale datasets uses the sklearn library [204]. For multi-class problems, we use the one-vs-one classification strategy. This study is also the largest scale comparison of signature kernels to date which extends the datasets considered in [228]. The hardware used was 2 computer clusters equipped with overall 8 NVIDIA 3080 Ti GPUs.

5.4.2 Methods

We compare the proposed variants RFSF-DP and RFSF-TRP to the baselines described here: (1) the M-truncated Signature Kernel [146] KSig formulated via the kernel trick, and is a quadratic time baseline; (2) the Signature-PDE Kernel [228] KSigPDE, which uses the 2^{nd}-order PDE solver and also has quadratic complexity; (3) the Global Alignment Kernel [58] GAK, one of the most popular sequence kernels to day and can can be related to the signature kernel, see [146, Sec. 5]; (4) Random Warping Series [292] RWS, which produces features by DTW alignments between the input and randomly sampled time series; (5) the RBF kernel, which treats the whole time series as a vector of length $\mathbb{R}^{\ell d}$, (6) Random Fourier Reatures [209] RFF, which also treats the time series as a long vector. We excluded RFSF from the comparison, as it is unfeasible

[3] https://github.com/tgcsaba/KSig

to compute it with reasonable sample sizes $\tilde{d}$ due to the polynomial explosion of dimensions in its tensor-based representation. The complexities are compared in Table 5.1.

5.4.3 Hyperparameter selection

For each dataset-kernel, we perform cross-validation to select the optimal hyperparameters that are optimized over the Cartesian product of the following options. For each method that requires a static kernel, we use the RBF kernel with bandwidth hyperparameter $\sigma > 0$. This is specified in terms of a rescaled median heuristic, i.e.

$$\sigma = \alpha \operatorname{med}\left\{\left\|\mathbf{x}_i - \mathbf{x}'_j\right\|_2 / 2 \,|\, i \in [\mathsf{L}_{\mathbf{x}}], j \in [\ell_{\mathbf{x}'}], \mathbf{x}, \mathbf{x}' \in \mathbf{X}\right\}, \quad \text{for } \alpha > 0, \tag{5.18}$$

where α is chosen from $\alpha \in \{10^{-3}, \dots, 10^3\}$ on a logarithmic grid with 19 steps. For each kernel that is not normalized by default (i.e. the GAK and RBF kernels are normalized, the former is because without normalization it blows up) , we select whether to normalize to unit norm in feature space via $\mathsf{k}(\mathbf{x}, \mathbf{y}) \mapsto \mathsf{k}(\mathbf{x}, \mathbf{y}) / \sqrt{\mathsf{k}(\mathbf{x}, \mathbf{x})\mathsf{k}(\mathbf{y}, \mathbf{y})}$. The SVM hyperparameter $C > 0$ is selected from $C \in \{10^0, 10^1, 10^2, 10^3, 10^4\}$. Further, motivated by previous work that investigates the effect of path augmentations in the context of signature methods [192], we chose 3 augmentations to cross-validate over. First is parameterization encoding, which gives the classifier the ability to remove the warping invariance of a given sequence kernel, adding the time index as an additional coordinate, i.e. for each time series in the dataset $\mathbf{x} \in \mathbf{X}$, we augment it via $\mathbf{x} = (\mathbf{x}_i)_{i=1}^{\mathsf{L}_{\mathbf{x}}} \mapsto (\beta i / \mathsf{L}_{\mathbf{x}}, \mathbf{x}_i)_{i=1}^{\mathsf{L}_{\mathbf{x}}}$, where $\beta > 0$ is the parameterization intensity chosen from $\beta \in \{10^0, 10^1, 10^2, 10^3, 10^4\}$. The second augmentation is the basepoint encoding, the role of which is to remove the translation invariance of signature features. Note that when the static base kernel is chosen to be a nonlinear kernel other than the Euclidean inner product, the signature kernel is not completely translation-invariant due to the state-space nonlinearities, but it is close being that by the L-Lipschitz property in Lemma 5.28 valid for of the static kernels considered in this work. The basepoint encoding adds an initial $\mathbf{0}$ step at the beginning of each time series, i.e. for $\mathbf{x} \in \mathbf{X}$, $\mathbf{x} = (\mathbf{x}_0, \dots, \mathbf{x}_{\mathsf{L}_{\mathbf{x}}}) \mapsto (\mathbf{0}, \mathbf{x}_0, \dots, \mathbf{x}_{\mathsf{L}_{\mathbf{x}}})$. The third augmentation is the lead-lag map, which is defined as $\mathbf{x} = (\mathbf{x}_0, \dots, \mathbf{x}_{\mathsf{L}_{\mathbf{x}}}) \mapsto \big((\mathbf{x}_0, \mathbf{x}_0), (\mathbf{x}_1, \mathbf{x}_0), (\mathbf{x}_1, \mathbf{x}_1), \dots, (\mathbf{x}_{\mathsf{L}_{\mathbf{x}}}, \mathbf{x}_{\mathsf{L}_{\mathbf{x}}}), (\mathbf{x}_{\mathsf{L}_{\mathbf{x}}}, \mathbf{x}_{\mathsf{L}_{\mathbf{x}}})\big)$. For the truncation-based signature kernels, we select the truncation level $M \in \mathbb{Z}_+$ from $M \in \{2, 3, 4, 5\}$. For RWS, we select the warping length from $W \in \{10, 20, \dots, 100\}$ as suggested by the authors. This makes RWS the most expensive feature-based kernel, and so as to fit within the same resource limitations, we omit cross-validating over the path augmentations. We select the standard deviation $\sigma > 0$ of the warping series from the same grid as α in (5.18). For all RF approaches, we set the RF dimension $\tilde{d} \in \mathbb{Z}_+$, so the overall dimension is 1000. Note that for

RFSF-DP and RFSF-TRP this is respectively $2^{M+1}\tilde{d}$ and $M\tilde{d}$, where $\tilde{d}$ is the base RFF sample size; for RWS it is the number of warping series $\tilde{d}$; while for RFF it is twice the number of samples $2\tilde{d}$.

5.4.4 Datasets

UEA Archive. The UEA archive [63] is a collection of overall 30 datasets for benchmarking classifiers on multivariate time series classification problems containing both binary and multi-class tasks. The data modality ranges from various sources e.g. human activity recognition, motion classification, ECG classification, EEG/MEG classification, audio spectra recognition, and others. The sizes of the datasets in terms of number of time series range from moderate ($\leq$ 1000 examples) to large ($\leq$ 30000), and includes various lengths between 8 and 18000. A summary of the dataset characteristics can be found in Table 2 in [63]. Pre-specified train-test splits are provided for each dataset, which we follow. We evaluate all considered kernels on the moderate datasets ($\leq$ 1000 time series), but because the non-feature-based become very expensive computationally beyond these sizes, we only evaluate feature-based approaches on medium and large datasets ($\geq$ 1000 time series). Each featurized approach is trained and evaluated 5 times on each dataset in order to account for the randomness in the hyperparameter selection procedure and evaluation.

Mental Workload Intensity Classification. We evaluate featurized approaches on a large-scale brain-activity recording data set called fNIRS2MW.[4] This dataset contains brain activity recordings collected from overall 68 participants during a 30-60 minute experimental session, where they were asked to carry out tasks of varying intensity. The collected time series are sliced into 30 second segments using a sliding window, and each segment is labelled with an intensity level (0-3), giving rise to overall $\sim$ 100000 segments, which we split in a ratio of $80-20$ for training and testing. We convert the task into a binary classification problem by assigning a label whether the task is low (0 or 1) or high (2 or 3) intensity.

Satellite Image Classification. As a massive scale task, we use a satellite imagery dataset[5] of $N = 10^6$ time series. Each length $\ell = 46$ time series corresponds to a vegetation index calculated from remote sensing data, and the task is to classify land cover types [207] by mapping vegetation profiles to various types of crops and forested areas corresponding to 24 classes. We split the dataset in a ratio of 90-10 for training and testing.

[4] https://github.com/tufts-ml/fNIRS-mental-workload-classifiers
[5] https://cloudstor.aarnet.edu.au/plus/index.php/s/pRLVtQyNhxDdCoM

Table 5.2: Comparison of SVM test accuracies on moderate multivariate time series classification datasets. For each row, the best result is highlighted in **bold**, and the second best in *italic*.

	RFSF-DP	RFSF-TRP	KSig	KSigPDE	RWS	GAK	RBF	RFF
ARTICULARYWORDRECOGNITION	0.984	0.981	**0.990**	0.983	*0.987*	0.977	0.977	0.978
ATRIALFIBRILLATION	0.373	0.320	*0.400*	0.333	**0.427**	0.333	0.267	0.373
BASICMOTIONS	**1.000**	**1.000**	**1.000**	**1.000**	*0.995*	**1.000**	0.975	0.860
CRICKET	0.964	0.964	0.958	*0.972*	**0.978**	0.944	0.917	0.886
DUCKDUCKGEESE	0.636	*0.664*	**0.700**	0.480	0.492	0.500	0.420	0.372
ERING	0.921	0.936	0.841	*0.941*	**0.945**	0.926	0.937	0.915
EIGENWORMS	*0.817*	**0.837**	0.809	0.794	0.623	0.511	0.496	0.443
EPILEPSY	**0.949**	*0.942*	**0.949**	0.891	0.925	0.870	0.891	0.777
ETHANOLCONCENTRATION	0.457	0.439	**0.479**	*0.460*	0.284	0.361	0.346	0.325
FINGERMOVEMENTS	0.608	0.624	**0.640**	*0.630*	0.612	0.500	0.620	0.570
HANDMOVEMENTDIRECTION	*0.573*	0.568	**0.595**	0.527	0.403	**0.595**	0.541	0.454
HANDWRITING	0.434	0.400	0.479	0.409	**0.591**	*0.481*	0.307	0.249
HEARTBEAT	0.717	0.712	0.712	**0.722**	0.714	0.717	0.717	*0.721*
JAPANESEVOWELS	0.978	0.978	**0.986**	**0.986**	0.955	*0.981*	*0.981*	0.979
LIBRAS	0.898	**0.928**	*0.922*	0.894	0.837	0.767	0.800	0.800
MOTORIMAGERY	*0.516*	**0.526**	0.500	0.500	0.508	0.470	0.500	0.482
NATOPS	0.906	0.908	0.922	**0.928**	*0.924*	0.922	0.917	0.900
PEMS-SF	0.800	0.808	0.827	*0.838*	0.701	**0.855**	**0.855**	0.770
RACKETSPORTS	0.874	0.861	**0.921**	*0.908*	0.878	0.849	0.809	0.755
SELFREGULATIONSCP1	0.868	0.856	*0.904*	*0.904*	0.829	**0.915**	0.898	0.885
SELFREGULATIONSCP2	0.489	0.510	*0.539*	**0.544**	0.481	0.511	0.439	0.492
STANDWALKJUMP	0.387	0.333	*0.400*	*0.400*	0.347	0.267	**0.533**	0.267
UWAVEGESTURELIBRARY	0.882	0.881	**0.912**	0.866	*0.897*	0.887	0.766	0.846
AVG. ACC.	*0.740*	0.738	**0.756**	0.735	0.710	0.702	0.692	0.656
AVG. RANK	3.609	3.739	**2.348**	*2.957*	3.957	4.174	4.913	5.913

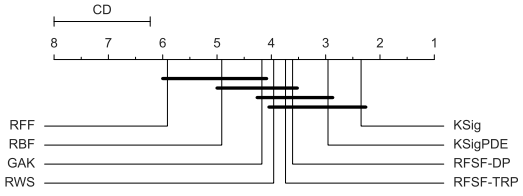

Figure 5.2: Critical difference diagram comparison on moderate datasets of considered approaches using two-tailed Nemenyi test [66].

Table 5.3: Comparison of accuracies on large-scale datasets of random features.

	RFSF-DP	RFSF-TRP	RWS	RFF
CHARACTERTRAJECTORIES	*0.990*	*0.990*	**0.991**	0.989
FACEDETECTION	*0.653*	**0.656**	0.642	0.572
INSECTWINGBEAT	*0.436*	**0.459**	0.227	0.341
LSST	0.589	*0.624*	**0.631**	0.423
PENDIGITS	*0.983*	0.982	**0.989**	0.980
PHONEMESPECTRA	*0.204*	*0.204*	**0.205**	0.083
SITS1M	**0.745**	*0.740*	0.610	0.718
SPOKENARABICDIGITS	**0.981**	*0.980*	**0.981**	0.964
FNIRS2MW	**0.659**	*0.658*	0.621	0.642
AVG. ACC.	*0.693*	**0.699**	0.655	0.635
AVG. RANK	**1.778**	*1.889*	2.222	3.333

5.4.5 Results

Table 5.2 compares test accuracies on moderate size multivariate time series classification datasets with $N \leq 1000$ from the UEA archive. KSig provides state-of-the-art performance among all sequence kernels with taking the highest aggregate score in terms of all of average accuracy, average rank, and number of first places. Our proposed random feature variants RFSF-DP and

Table 5.4: Average ranks of SVM classifier accuracies per data domain. For each row, the best result is highlighted in **bold**, and the second best in *italic*.

DOMAIN	RFSF-DP	RFSF-TRP	KSig	KSigPDE	RWS	GAK	RBF	RFF
AUDIO	3.333	3.667	**2.333**	*2.667*	4.667	3.000	4.000	4.333
ECG	3.000	5.000	**2.000**	3.000	*2.500*	5.000	3.500	4.500
EEG/MEG	4.200	3.400	**2.000**	*2.800*	5.800	3.800	4.600	5.400
HAR	3.556	3.556	*2.667*	3.000	**2.222**	4.111	5.333	6.667
MOTION	*2.500*	3.000	**2.000**	4.000	3.500	6.500	7.000	7.000
OTHER	4.000	4.000	**2.000**	**2.000**	7.500	*3.000*	3.500	6.500

RFSF-TRP provide comparable performance on most of the datasets in terms of accuracy, and they are only outperformed by KSig and KSigPDE with respect to average accuracy and rank. Interestingly, RFSF-TRP has more first place rankings, but RFSF-DP performs slightly better on average. This shows that on datasets of these sizes, using either of RFSF-DP and RFSF-TRP does not sacrifice model performance - even leading to improvements in some cases, potentially due to the implicit regularization effect of restricting to a finite-dimensional feature space - and it can already provide speedups. We visualize the critical difference diagram comparison of all considered approaches in Figure 5.2.

Table 5.3 demonstrates the performance of scalable approaches, i.e. RFSF-DP, RFSF-TRP, RWS and RFF on the remaining UEA datasets ($N \geq 1000$), the dataset fNIRS2MW ($N = 10^5$), and the satellite dataset SITS1M ($N = 10^6$). We find it infeasible to perform full cross-validation for quadratic time kernels on these datasets due to expensive kernel computations and downstream cost of dual SVM. The results show that both variants RFSF-DP and RFSF-TRP perform significantly better on average with respect to accuracy and rank then both RWS and RFF. Note when RWS takes first place, it only improves over our approach marginally, however, when it underperforms, it often does so severely. This is not surprising as both RFSF-DP and RFSF-TRP approximate the signature kernel, which is a universal kernel on time series; it is theoretically capable of learning from any kind of time series data as supported by its best overall performance above.

Domain analysis. Next, we provide a domain analysis on the moderate sized TSC datasets in Table 5.4 split across 6 data domains [63]: audio spectra classification (Audio), electrocardiogram (ECG) recordings, electroencephalography and magnetoencephalography (EEG/MEG), human activitiy recognition (HAR) based on accelerometer or gyroscope data, motion recognition (Motion) using other data sources, and other data modalities. We make the following observations: (1) on Audio, KSig and KSigPDE come in first and second respectively while GAK is in third place; (2) on ECG, KSig is first and RWS is second while RFSF-DP and KSigPDE are tied for third place, RFSF-TRP underperforms on this modality; (3) on EEG/MEG, KSig and KSigPDE

are first and second again, while RFSF-TRP is third, and RFSF-DP somewhat underperforms; (4) for HAR datasets, RWS is first, KSig is second, and KSigPDE is third, while RFSF-DP and RFSF-TRP are tied for fourth place; (5) on Motion datasets KSig is first, RFSF-DP is second, and RFSF-TRP is third; (6) on Other datasets, KSig and KSigPDE are tied for first place, while GAK comes in second, and RBF in third. Overall, we observe that KSig takes undisputed first place, while KSigPDE often comes in close second. The performance of RFSF-TRP and RFSF-DP vary across domains, one often outperforming the other indicating that the way they approximate the signature kernel captures patterns in the data differently. Note that RWS outperforms the RFSF variants on some data modalities, however, it also often underperforms severely while at least one RFSF variation always performs at least moderately well on each data domain.

5.5 Conclusion

We constructed a random kernel $\tilde{\mathsf{k}}_{\mathsf{S}_{\leq M}}$ for sequences that benefits from (i) lifting the original sequence to an infinite-dimensional RKHS $\mathcal{H}_{\mathsf{k}}$, (ii) linear complexity in sequence length, (iii) being with high probability close to the signature kernel k_{S}. Thereby it combines the strength of the signature kernel k_{S} which is to implicitly use the iterated integrals of a sequence that has an infinite-dimensional RKHS $\mathcal{H}_{\mathsf{k}}$ as state-space with the strength of (unkernelized) signature features φ_{S} that only require linear time complexity. Our main theoretical result extends the theoretical guarantees for translation-invariant kernels on linear spaces to the signature kernel defined on the nonlinear domain $\mathsf{Seq}(\mathcal{X})$; however, the proofs differ from the classic case and require to analyse the error propagation in tensor space. A second step is more straightforward, and combines this approach with random projections in finite-dimensions for tensors to reduce the complexity in memory further. The advantages and disadvantages of the resulting approach are analogous to the classic RFF technique on $\mathbb{R}^d$, namely a reduction of computational complexity by an order for the price of an approximation that only holds with high probability. As in the classic RFF case, our experiments indicate that this is in general a favourable tradeoff.

In the future, it would be interesting both theoretically and empirically to replace the vanilla Monte Carlo integration in the RFF construction by block-orthogonal random matrices as done in [300]. Further, our random features can also be used to define an unbiased appoximation to the inner product of expected signatures, which has found usecases, among many, in nonparametric hypothesis testing and market regime detection [44, 129], training of generative models [199, 130], and graph representation learning [265].

5.A Concentration of measure

5.A.1 Classic inequalities

The following inequalities are classic, and their proofs are in analysis textbooks, e.g. [75, 221]. Firstly, Jensen's inequality is useful for convex (concave) expectations.

Lemma 5.11 (Jensen's inequality)**.** *Let X be an integrable random variable, and $f : \mathbb{R} \to \mathbb{R}$ a convex function, such that $f(X)$ is also integrable. Then, the following inequality holds:*

$$f(\mathbb{E}[X]) \leq \mathbb{E}[f(X)].$$

Hölder's inequality is a generalization of the Cauchy-Schwarz inequality to L^p spaces.

Lemma 5.12 (Hölder's inequality)**.** *Let $p, q \geq 1$ such that $\frac{1}{p} + \frac{1}{q} = 1$. Let X and Y respectively be L^p and L^q integrable random variables, i.e. $\mathbb{E}[|X|^p] < \infty$ and $\mathbb{E}[|Y|^q] < \infty$. Then, XY is integrable, and it holds that*

$$\mathbb{E}[|XY|] \leq \mathbb{E}^{1/p}[|X|^p]\mathbb{E}^{1/q}[|Y|^q].$$

Although not inherently a probabilistic inequality, Young's inequality can be used to decouple products of random variables.

Lemma 5.13 (Young's inequality)**.** *Let $p, q > 0$ with $\frac{1}{p} + \frac{1}{q} = 1$. Then, for every $a, b \geq 0$*

$$ab \leq \frac{a^p}{p} + \frac{b^q}{q}.$$

Lemma 5.14 (Reverse Young's inequality)**.** *Let $p, q > 0$ such that $\frac{1}{p} - \frac{1}{q} = 1$. Then, for every $a \geq 0$ and $b > 0$, it holds that*

$$ab \geq \frac{a^p}{p} - \frac{b^{-q}}{q}.$$

Proof. Apply Young's inequality with $p' = \frac{1}{p}$ and $q' = \frac{q}{p}$ to $a' = (ab)^p$ and $b' = b^{-p}$. □

5.A.2 Subexponential concentration

Next, we state a variation on the well-known Bernstein inequality, which holds for random variables in the subexponential class. The condition (5.21) on a random variable X, in this case, is

formulated as a moment-growth bound, called a Bernstein moment condition. We show in Lemmas 5.21 and 5.22 (in a more general setting) that (5.21) is equivalent to the random variable being subexponential. Further, note that the condition itself is given in terms of non-centered random variables, while the statement of the theorem is about their centered counterparts. For similar results, see [272, Sec. 2.2.2].

Theorem 5.15 (Bernstein inequality - one-tailed)**.** *Let $X_1, \dots, X_n$ be independent random variables satisfying the following moment-growth condition for some $S, R > 0$,*

$$\mathbb{E}\left[X_i^k\right] \leq \frac{k! S^2 R^{k-2}}{2} \quad \text{for all } k \geq 2. \tag{5.19}$$

Then, it holds for $\tilde{X}_i = X_i - \mathbb{E}[X_i]$ that

$$\mathbb{P}\left[\sum_{i=1}^n \tilde{X}_i \geq t\right] \leq \exp\left(\frac{-t^2}{2(nS^2 + Rt)}\right).$$

Proof. We have for $\lambda > 0$

$$\begin{aligned}\mathbb{P}\left[\sum_{i=1}^n \tilde{X}_i \geq t\right] &\overset{(a)}{\leq} \exp(-\lambda t)\mathbb{E}\left[\exp\left(\lambda \sum_{i=1}^n \tilde{X}_i\right)\right] \\ &\overset{(b)}{=} \exp\left(-\lambda t - \lambda \sum_{i=1}^n \mathbb{E}[X_i]\right) \prod_{i=1}^n \mathbb{E}[\exp(\lambda X_i)],\end{aligned} \tag{5.20}$$

where (a) holds for any $\lambda > 0$ by the Chernoff bound applied to $\sum_{i=1}^n \tilde{X}_i$, and in (b) we used the independence of X_i-s. Bounding the moment-generating function of X_i, one gets

$$\begin{aligned}\mathbb{E}[\exp(\lambda X_i)] &\overset{(d)}{=} 1 + \lambda \mathbb{E}[X_i] + \sum_{k=2}^{\infty} \frac{\lambda^k}{k!} \mathbb{E}\left[X_i^k\right] \overset{(e)}{\leq} 1 + \lambda \mathbb{E}[X_i] + \frac{S^2\lambda^2}{2} \sum_{k=0}^{\infty} \lambda^k R^k \\ &\overset{(f)}{=} 1 + \lambda \mathbb{E}[X_i] + \frac{S^2\lambda^2}{2(1-\lambda R)} \overset{(g)}{\leq} \exp\left(\lambda \mathbb{E}[X_i] + \frac{S^2\lambda^2}{2(1-\lambda R)}\right),\end{aligned}$$

where (d) is due to the Taylor expansion of the exponential function, (e) implied by (5.19), (f) holds by the sum of geometric series for any $\lambda < 1/R$, (g) follows from the inequality $1 + x \leq \exp(x)$ for all $x \in \mathbb{R}$. Using this to bound (5.20),

$$\begin{aligned}\mathbb{P}\left[\sum_{i=1}^n \tilde{X}_i \geq t\right] &\leq \exp\left(-\lambda t - \lambda \sum_{i=1}^n \mathbb{E}[X_i]\right) \prod_{i=1}^n \exp\left(\lambda \mathbb{E}[X_i] + \frac{S^2\lambda^2}{2(1-\lambda R)}\right) \\ &= \exp\left(-\lambda t + \frac{nS^2\lambda^2}{2(1-\lambda R)}\right).\end{aligned}$$

Finally, choosing $\lambda = t/(nS^2 + Rt)$ gives the statement. □

chapter 5.16 (Bernstein inequality - two-tailed)**.** *Let $X_1, \ldots, X_n$ be independent random variables satisfying the following moment-growth condition for some $S, R > 0$,*

$$\mathbb{E}\left[|X_i|^k\right] \leq \frac{k! S^2 R^{k-2}}{2} \quad \text{for all } k \geq 2. \tag{5.21}$$

Then, it holds for $\tilde{X}_i = X_i - \mathbb{E}[X_i]$ that

$$\mathbb{P}\left[\left|\sum_{i=1}^{n} \tilde{X}_i\right| \geq t\right] \leq 2\exp\left(\frac{-t^2}{2(nS^2 + Rt)}\right).$$

Proof. Applying Theorem 5.15 to $\sum_{i=1}^n \tilde{X}_i$ and $-\sum_{i=1}^n \tilde{X}_i$, and combining the two bounds gives the two-tailed result. □

5.A.3 α-exponential concentration

In this section, we introduce a specific class of Orlicz norms for heavy-tailed random variables, which generalizes subexponential distributions.

Definition 5.17 (α-exponential Orlicz norm)**.** *Let $\alpha > 0$ and define the function*

$$\Psi_\alpha : \mathbb{R} \to \mathbb{R}, \quad \Psi_\alpha(x) = \exp(x^\alpha) - 1 \quad \text{for all } x \in \mathbb{R}.$$

The α-exponential Orlicz (quasi-)norm of a random variable X is defined as

$$\|X\|_{\Psi_\alpha} = \inf\left\{t > 0 : \mathbb{E}\left[\Psi_\alpha\left(\frac{|X|}{t}\right)\right] \leq 1\right\},$$

adhering to the standard convention that $\inf \emptyset = \infty$.

If a random variable X satisfies $\|X\|_{\Psi_\alpha} < \infty$, it is either called an α-(sub)exponential random variable (if $\alpha < 1$) [229, 100, 35], or sub-Weibull of order-α [153, 302].

An alternative characterization of the α-exponential norm is the following tail-bound.

Remark 5.18 (Tail bound)**.** *Let $\alpha > 0$ and X be a random variable such that $\|X\|_{\Psi_\alpha} < \infty$. Then,*

$$\mathbb{P}[|X| \geq \epsilon] \leq 2\exp\left(-\frac{\epsilon^\alpha}{\|X\|_{\Psi_\alpha}^\alpha}\right).$$

Proof. Due to Definition 5.17, $\mathbb{E}\left[\exp\left(\frac{|X|^\alpha}{\|X\|_{\Psi_\alpha}^\alpha}\right)\right] \leq 2$, hence by Markov's inequality

$$\mathbb{P}\left[|X| \geq \epsilon\right] = \mathbb{P}\left[\frac{|X|^\alpha}{\|X\|_{\Psi_\alpha}^\alpha} \geq \frac{\epsilon^\alpha}{\|X\|_{\Psi_\alpha}^\alpha}\right] \leq e^{-\frac{\epsilon^\alpha}{\|X\|_{\Psi_\alpha}^\alpha}} \mathbb{E}\left[\exp\left(\frac{|X|^\alpha}{\|X\|_{\Psi_\alpha}^\alpha})\right)\right] \leq 2e^{-\frac{\epsilon^\alpha}{\|X\|_{\Psi_\alpha}^\alpha}}.$$

□

Note that although $\|\cdot\|_{\Psi_\alpha}$ is often referred to as a norm, it does not satisfy the triangle inequality for $\alpha < 1$, although we can still relate the norm of the sum to the sum of the norms.

Lemma 5.19 (Generalized triangle inequality for Orlicz norm)**.** *It holds for any random variables X, Y and $\alpha > 0$ that*

$$\|X + Y\|_{\Psi_\alpha} \leq C_\alpha \left(\|X\|_{\Psi_\alpha} + \|Y\|_{\Psi_\alpha}\right),$$

where $C_\alpha = 2^{1/\alpha}$ if $\alpha < 1$ and 1 otherwise.

Proof. See Lemma A.3. in [100]. □

A useful property of α-exponential norms is that they satisfy a Hölder-type inequality.

Lemma 5.20 (Hölder inequality for Orlicz norm)**.** *It holds for any random variables $X_1, \ldots, X_k$ and $\alpha_1, \ldots, \alpha_k > 0$ that*

$$\left\|\prod_{i=1}^k X_i\right\|_{\Psi_\alpha} \leq \prod_{i=1}^k \|X_i\|_{\Psi_{\alpha_i}},$$

where $\alpha = \left(\sum_{i=1}^k \alpha_i^{-1}\right)^{-1}$.

Proof. See Lemma A.1. in [100]. □

Next, we show in the following two lemmas that a random variable X is α-exponential if and only if $|X|^\alpha$ satisfies a Bernstein moment-growth condition.

Lemma 5.21 (Bernstein condition implies Orlicz norm bound)**.** *Let $\alpha > 0$ and X be a random variable that satisfies for $S, R > 0$ that*

$$\mathbb{E}\left[|X|^{k\alpha}\right] \leq \frac{k! S^2 R^{k-2}}{2} \quad \text{for all } k \geq 2. \tag{5.22}$$

Then, it holds that

$$\|X\|_{\Psi_\alpha} \leq 2\,(S \vee R)^{1/\alpha}\,.$$

Proof. Firstly, due to Jensen's inequality (Lemma 5.11) and (5.22), we have

$$\mathbb{E}\left[|X|^\alpha\right] \leq \mathbb{E}^{1/2}\left[|X|^{2\alpha}\right] \leq S. \tag{5.23}$$

We proceed similarly to the proof of Theorem 5.15. For $t > 0$, we have

$$\begin{aligned}\mathbb{E}\left[\exp\frac{|X|^\alpha}{t^\alpha}\right] &\overset{\text{(a)}}{=} 1 + \frac{\mathbb{E}\left[|X|^\alpha\right]}{t^\alpha} + \sum_{m=2}^{\infty}\frac{\mathbb{E}\left[|X|^{m\alpha}\right]}{t^{m\alpha}m!} \overset{\text{(b)}}{\leq} 1 + \frac{S}{t^\alpha} + \frac{S^2}{2t^{2\alpha}}\sum_{m=0}^{\infty}\left(\frac{R}{t^\alpha}\right)^m \\ &\overset{\text{(c)}}{=} \underbrace{1 + \frac{S}{t^\alpha} + \frac{S^2}{2t^{2\alpha}}\frac{t^\alpha}{t^\alpha - R}}_{f(t)},\end{aligned}$$

where (a) is due to the Taylor expansion of the exponential function, (b) is due to (5.22) and (5.23), (c) is the sum of a geometric series for $R < t^\alpha$. Defining $f(t) = 1 + \frac{S}{t^\alpha}\left(1 + \frac{S}{2(t^\alpha - R)}\right)$ and solving for $f(t) \leq 2$ leads to the quadratic inequality

$$0 \leq t^{2\alpha} - (S+R)t^\alpha + SR - \frac{S^2}{2},$$

which has roots $t_{1,2}^\alpha = \frac{1}{2}\left(S + R \pm \sqrt{(S-R)^2 + 2S^2}\right)$. We discard the left branch, which violates the condition $R < t^\alpha$ since

$$t_2^\alpha = \frac{1}{2}\left(S + R - \sqrt{(S-R)^2 + 2S^2}\right) \leq \frac{1}{2}\left(S + R - |S-R|\right) = S \wedge R \leq R.$$

Now, as the inequality $\sqrt{x^2 + y^2} \leq |x| + |y|$ holds for all $x, y \in \mathbb{R}$, we get

$$\begin{aligned}t_1^\alpha &\leq \frac{1}{2}\left(S + R + \sqrt{(S-R)^2 + 2S^2}\right) \leq \frac{1}{2}\left(S + R + |S-R| + \sqrt{2}S\right) \\ &= S \vee R + \frac{\sqrt{2}}{2}S \leq 2(S \vee R)\end{aligned}$$

and hence choosing $t \geq (2(S \vee R))^{1/\alpha} \geq t_1$ implies $f(t) \leq 2$, and we are done. □

The other direction is proven in the following lemma.

Lemma 5.22 (Finite Orlicz norm implies Bernstein condition)**.** *Let $\alpha > 0$ and X be a random variable such that $\|X\|_{\Psi_\alpha} < \infty$. Then,*

$$\mathbb{E}\left[|X|^{k\alpha}\right] \leq \frac{k! S^2 R^{k-2}}{2} \quad \text{for all } k \geq 2,$$

where $S = \sqrt{2}\,\|X\|_{\Psi_\alpha}^{\alpha}$ and $R = \|X\|_{\Psi_\alpha}^{\alpha}$.

Proof. We have that

$$\begin{aligned}\mathbb{E}\left[|X|^{k\alpha}\right] &\overset{(a)}{=} k!\,\|X\|_{\Psi_\alpha}^{k\alpha}\,\mathbb{E}\left[\frac{1}{k!}\left(\frac{|X|}{\|X\|_{\Psi_\alpha}}\right)^{k\alpha}\right] \overset{(b)}{\leq} k!\,\|X\|_{\Psi_\alpha}^{k\alpha}\,\mathbb{E}\left[\exp\left(\frac{|X|^{\alpha}}{\|X\|_{\Psi_\alpha}^{\alpha}}\right) - 1\right] \\ &\overset{(c)}{\leq} k!\,\|X\|_{\Psi_\alpha}^{k\alpha} \overset{(d)}{=} \frac{k!\left(\sqrt{2}\,\|X\|_{\Psi_\alpha}^{\alpha}\right)^2\left(\|X\|_{\Psi_\alpha}^{\alpha}\right)^{k-2}}{2},\end{aligned}$$

where (a) is simply multiplying and dividing by the same values, (b) is the inequality $1 + x^k/k! \leq e^x$ for $x > 0$, (c) follows from Definition 5.17, (d) is reorganizing terms to the required form. □

We adapt the following concentration inequality from [153] for α-exponential summation.

Theorem 5.23 (Concentration inequality for α-subexponential summation)**.** *Let $\alpha \in (0,1)$ and $X_1, \ldots, X_n$ be independent, centered random variables with $\|X_i\|_{\Psi_\alpha} \leq M_\alpha$ for all $i \in [n]$. Then, it holds for $t > 0$ that*

$$\mathbb{P}\left[\left|\sum_{i=1}^{n} X_i\right| \geq C_\alpha\left(2\sqrt{nt} + \sqrt{2}4^{1/\alpha}t^{1/\alpha}\right)\right] \leq 2e^{-t}, \tag{5.24}$$

where $C_\alpha = \sqrt{8}e^4(2\pi)^{1/4}e^{1/24}(2e/\alpha)^{1/\alpha}M_\alpha$. Alternatively, it holds for $\epsilon > 0$ that

$$\mathbb{P}\left[\left|\sum_{i=1}^{n} X_i\right| \geq \epsilon\right] \leq 2\exp\left(-\frac{1}{4}\min\left\{\left(\frac{\epsilon}{2\sqrt{n}C_\alpha}\right)^2, \left(\frac{\epsilon}{\sqrt{8}C_\alpha}\right)^{\alpha}\right\}\right). \tag{5.25}$$

Proof. The inequality (5.24) follows directly from [153, Theorem 3.1].

To show (5.25), let $g_1(t) = 2C_\alpha\sqrt{nt}$ and $g_2(t) = \sqrt{2}C_\alpha 4^{1/\alpha}t^{1/\alpha}$. Now, for $t > 0$

$$\mathbb{P}\left[\left|\sum_{i=1}^{n} X_i\right| \geq 2\left(g_1(t) \vee g_2(t)\right)\right] \leq \mathbb{P}\left[\left|\sum_{i=1}^{n} X_i\right| \geq g_1(t) + g_2(t)\right] \leq 2e^{-t},$$

which is equivalently written as

$$\mathbb{P}\left[\left|\sum_{i=1}^{n} X\right| \geq \epsilon\right] \leq 2\exp\left(-\left(g_1^{-1}(\epsilon/2) \wedge g_2^{-1}(\epsilon/2)\right)\right).$$

□

5.A.4 Hypercontractivity

Here we provide an alternative approach for the concentration of heavy-tailed random variables, specifically for polynomials of Gaussian random variables. The following lemma states a moment bound for such Gaussian chaoses, considered in e.g. [134, 24].

Lemma 5.24 (Moment bound for Gaussian polynomial)**.** *Consider a degree-p polynomial $f(X) = f(X_1, \ldots, X_n)$ of independent centered Gaussian random variables, $X_1, \ldots X_n \overset{i.i.d.}{\sim} \mathcal{N}(0,1)$. Then, for all $k \geq 2$*

$$\mathbb{E}^{1/k}\left[|f(X)|^k\right] \leq (k-1)^{p/2}\mathbb{E}^{1/2}\left[|f(X)|^2\right].$$

Theorem 5.25 (Concentration inequality for Gaussian polynomial)**.** *Consider a degree-p polynomial $f(X) = f(X_1, \ldots, X_n)$ of independent centered Gaussian random variables, $X_1, \ldots, X_n \overset{i.i.d.}{\sim} \mathcal{N}(0,1)$. Then, for $p \geq 2$ and $\epsilon > 0$*

$$\mathbb{P}\left[|f(X) - \mathbb{E}[f(X)]| \geq \epsilon\right] \leq 2\exp\left(-\frac{\epsilon^{2/p}}{2\sqrt{2}e\mathbb{V}^{1/p}[f(X)]}\right),$$

where $\mathbb{V}[\cdot]$ denotes the variance of a random variable.

Proof. Without loss of generality, we may assume that $\mathbb{E}[f(X)] = 0$ and $\mathbb{V}[f(X)] = 1$. Since Lemma 5.24 holds, we have for $p, k \geq 2$

$$\mathbb{E}\left[|f(X)|^{2k/p}\right] \overset{(a)}{\leq} \mathbb{E}^{2/p}\left[|f(X)|^k\right] \overset{(b)}{\leq} (k-1)^k \leq k^k \overset{(c)}{\leq} k!e^k \leq \frac{k!(\sqrt{2}e)^2 e^{k-2}}{2},$$

where (a) holds due to Jensen inequality since $(\cdot)^{2/p}$ is concave, (b) is Lemma 5.24, and (c) is due to Stirling's approximation. Hence, $|f(X)|^{2/p}$ satisfies a Bernstein condition with $S = \sqrt{2}e$ and $R = e$. Therefore, by Lemma 5.21, we have $\|f(X)\|_{\Psi_{2/p}} \leq (2\sqrt{2}e)^{p/2}$.

Then, by Remark 5.18, it holds that

$$\mathbb{P}\left[|f(X)| \geq \epsilon\right] \leq 2\exp\left(-\frac{\epsilon^{2/p}}{\|f(X)\|_{\Psi_{2/p}}^{2/p}}\right) \leq 2\exp\left(-\frac{\epsilon^{2/p}}{2\sqrt{2}e}\right).$$

□

5.B Random Fourier Features

[209] provides a uniform bound for the RFF error over a compact and convex domain $\mathcal{X} \subset \mathbb{R}^d$ with diameter $|\mathcal{X}|$ for some absolute constant $C > 0$,

$$\mathbb{P}\left[\sup_{\mathbf{x},\mathbf{y}\in\mathcal{X}} \left|\tilde{\mathsf{k}}(\mathbf{x},\mathbf{y}) - \mathsf{k}(\mathbf{x},\mathbf{y})\right| \geq \epsilon\right] \leq C\left(\frac{\sigma_\Lambda |\mathcal{X}|}{\epsilon}\right)^2 \exp\left(-\frac{-\tilde{d}\epsilon^2}{4(d+2)}\right), \tag{5.26}$$

where $\sigma_\Lambda^2 = \mathbb{E}_{\mathbf{w}\sim\Lambda}\left[\|\mathbf{w}\|^2\right]$, and [253] shows that $C \leq 66$. Equation (5.26) implies [244, Sec. 2]

$$\sup_{\mathbf{x},\mathbf{y}\in\mathcal{X}} \left|\tilde{\mathsf{k}}(\mathbf{x},\mathbf{y}) - \mathsf{k}(\mathbf{x},\mathbf{y})\right| = O_p\left(|\mathcal{X}|\sqrt{\tilde{d}^{-1}\log\tilde{d}}\right),$$

where the $X_n = O_p(a_n)$ notation for $a_n > 0$ refers to $\lim_{C\to 0} \limsup_{n\to\infty} \mathbb{P}\left[\frac{|X_n|}{a_n} > C\right] = 0$.

The converse result[6] for RFF derivatives was shown in [244, Thm. 5]. The idea of the proof is to cover the input domain by an ϵ-net, and control the approximation error on the centers, while simultaneously controlling the Lipschitz constant of the error to get the bound to hold uniformly. We provide an adapted version with two main differences: (1) using the Bernstein inequality from Corollary 5.16, where the Bernstein condition is given in terms of non-centered random variables, and (2) using the covering numbers of [56]. We will use this theorem in proving Theorem 5.3 for controlling the approximation error of the derivatives of RFFs.

Given $\mathbf{p} \in \mathbb{N}^d$, we denote $|\mathbf{p}| = p_1 + \ldots + p_d$, for a function $f : \mathbb{R}^d \to \mathbb{R}$ the $\mathbf{p}$-th partial derivative by $\partial^{\mathbf{p}} f(\mathbf{z}) = \frac{\partial^{|\mathbf{p}|} f(\mathbf{z})}{\partial^{p_1} z_1 \ldots \partial^{p_d} z_d}$, for a vector $\mathbf{w} \in \mathbb{R}^d$ the $\mathbf{p}$-th power by $\mathbf{w}^{\mathbf{p}} = w_1^{p_1} \cdots w_d^{p_d}$.

Theorem 5.26 (Concentration inequality for RFF kernel derivatives)**.** *Let $\mathbf{p}, \mathbf{q} \in \mathbb{N}^d$ and $\mathsf{k} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be a continuous, bounded, translation-invariant kernel such that $\mathbf{z} \mapsto \nabla[\partial^{\mathbf{p},\mathbf{q}}\mathsf{k}(\mathbf{z})]$ is continuous. Let $\mathcal{X} \subset \mathbb{R}^d$ be a compact and convex domain with diameter $|\mathcal{X}|$, and denote by $D_{\mathbf{p},\mathbf{q},\mathcal{X}} = \sup_{\mathbf{z}\in\mathcal{X}_\Delta} \|\nabla[\partial^{\mathbf{p},\mathbf{q}}\mathsf{k}(\mathbf{z})]\|_2$, where $\mathcal{X}_\Delta = \{\mathbf{x}-\mathbf{y} \,|\, \mathbf{x},\mathbf{y} \in \mathcal{X}\}$. Let Λ be the spectral measure of*

[6] These error guarantees can also be improved exponentially both for the approximation of kernel values [244] and kernel derivatives [255, 35] in terms of the size of the domain where it holds, i.e. $|\mathcal{X}|$.

k *satisfying that* $E_{\mathbf{p},\mathbf{q}} = \mathbb{E}_{\mathbf{w}\sim\Lambda}\left[\left|\mathbf{w}^{\mathbf{p}+\mathbf{q}}\right| \|\mathbf{w}\|_2\right] < \infty$*, and the Bernstein moment condition for some* $S, R > 0$*,*

$$\mathbb{E}_{\mathbf{w}\sim\Lambda}\left[\left|\mathbf{w}^{\mathbf{p}+\mathbf{q}}\right|^k\right] \leq \frac{k! S^2 R^{k-2}}{2} \quad \textit{for all } k \geq 2.$$

Then, for $C_{\mathbf{p},\mathbf{q},\mathcal{X}} = |\mathcal{X}|\,(D_{\mathbf{p},\mathbf{q},\mathcal{X}} + E_{\mathbf{p},\mathbf{q}})$ *and* $\epsilon > 0$*,*

$$\mathbb{P}\left[\sup_{\mathbf{x},\mathbf{y}\in\mathcal{X}} \left|\partial^{\mathbf{p},\mathbf{q}}\tilde{\mathsf{k}}(\mathbf{x},\mathbf{y}) - \partial^{\mathbf{p},\mathbf{q}}\mathsf{k}(\mathbf{x},\mathbf{y})\right| \geq \epsilon\right] \leq 16\left(\frac{C_{\mathbf{p},\mathbf{q},\mathcal{X}}}{\epsilon}\right)^{\frac{d}{d+1}} \exp\left(\frac{-\tilde{d}\epsilon^2}{4(d+1)(2S^2 + R\epsilon)}\right).$$

Proof. We adapt the proof of [209, 244]. Note that as $\mathcal{X}$ is compact, so is $\mathcal{X}_\Delta$, and it can be covered by an ϵ-net of at most $T = (4|\mathcal{X}|/r)^d$ balls of radius $r > 0$ [56, Prop. 5] with centers $\mathbf{z}_1, \ldots, \mathbf{z}_T \in \mathcal{X}_\Delta$. Since for all $\mathbf{z} \in \mathcal{X}_\Delta$ there exists $i \in \{1, \ldots, T\}$ such that $\|\mathbf{z} - \mathbf{z}_i\|_2 \leq r$, it holds for $f(\mathbf{z}) = \partial^{\mathbf{p},\mathbf{q}}\tilde{\mathsf{k}}(\mathbf{z}) - \partial^{\mathbf{p},\mathbf{q}}\mathsf{k}(\mathbf{z})$ and $L_f = \sup_{\mathbf{s}\in\mathcal{X}_\Delta} \|\nabla f(\mathbf{s})\|_2$ that

$$|f(\mathbf{z}) - f(\mathbf{z}_i)| \overset{\text{(a)}}{\leq} \sup_{\mathbf{s}\in\mathcal{X}_\Delta} \|\nabla f(\mathbf{s})\|_2 \|\mathbf{z} - \mathbf{z}_i\|_2 \overset{\text{(b)}}{\leq} \sup_{\mathbf{s}\in\mathcal{X}_\Delta} \|\nabla f(\mathbf{s})\|_2 r = L_f r, \tag{5.27}$$

where (a) is due to the mean-value theorem followed by Cauchy-Schwarz inequality, and (b) is since $\|\mathbf{z} - \mathbf{z}_i\| \leq r$. Now, by triangle inequality, it holds for any $\mathbf{z} \in \mathcal{X}_\Delta$ that

$$\|\nabla f(\mathbf{z})\|_2 \leq \left\|\nabla\left[\partial^{\mathbf{p},\mathbf{q}}\tilde{\mathsf{k}}(\mathbf{z})\right]\right\|_2 + \|\nabla\left[\partial^{\mathbf{p},\mathbf{q}}\mathsf{k}(\mathbf{z})\right]\|_2 \leq \left\|\nabla\left[\partial^{\mathbf{p},\mathbf{q}}\tilde{\mathsf{k}}(\mathbf{z})\right]\right\|_2 + D_{\mathbf{p},\mathbf{q},\mathcal{X}}. \tag{5.28}$$

Differentiating $\tilde{\mathsf{k}}(\mathbf{z})$ as defined in (5.7), we get by the chain rule and triangle inequality that

$$\left\|\nabla\left[\partial^{\mathbf{p},\mathbf{q}}\tilde{\mathsf{k}}(\mathbf{z})\right]\right\|_2 = \frac{1}{\tilde{d}}\left\|\sum_{j=1}^{\tilde{d}} \mathbf{w}_j \mathbf{w}_j^{\mathbf{p}+\mathbf{q}} \cos^{(|\mathbf{p}+\mathbf{q}|)}(\mathbf{w}_j^\top \mathbf{z})\right\|_2 \leq \frac{1}{\tilde{d}}\sum_{j=1}^{\tilde{d}} \left|\mathbf{w}_j^{\mathbf{p}+\mathbf{q}}\right| \left\|\mathbf{w}_j\right\|_2, \tag{5.29}$$

where $\cos^{(n)}$ denotes the n-th derivative of cos for $n \in \mathbb{N}$. The main idea of the proof is then

$$\bigcap_{i=1}^{T} \{|f(\mathbf{z}_i)| < \epsilon/2\} \bigcap \{L_f < \epsilon/2r\} \subseteq \{|f(\mathbf{z})| < \epsilon \,|\, \forall \mathbf{z} \in \mathcal{X}_\Delta\},$$

since $|f(\mathbf{z})| \leq |f(\mathbf{z}) - f(\mathbf{z}_i)| + |f(\mathbf{z}_i)|$ and (5.27) holds. Therefore, taking the complement and bounding the union, we get our governing inequality for the uniform error over $\mathcal{X}_\Delta$:

$$\mathbb{P}\left[\sup_{\mathbf{z}\in\mathcal{X}_\Delta} |f(\mathbf{z})| \geq \epsilon\right] \leq \sum_{i=1}^{T} \mathbb{P}\left[|f(\mathbf{z}_i)| \geq \epsilon/2\right] + \mathbb{P}\left[L_f \geq \epsilon/2r\right]. \tag{5.30}$$

Now, we need to bound all probabilities on the RHS. First, we deal with L_f,

$$\mathbb{P}\left[L_f \geq \epsilon/2r\right] \overset{(c)}{\leq} \mathbb{E}\left[L_f\right]\frac{2r}{\epsilon} \overset{(d)}{\leq} \left(D_{\mathbf{p},\mathbf{q},\mathcal{X}} + \frac{1}{\tilde{d}}\sum_{j=1}^{\tilde{d}} \mathbb{E}_{\mathbf{w}_j\sim\Lambda}\left[\left|\mathbf{w}_j^{\mathbf{p}+\mathbf{q}}\right| \left\|\mathbf{w}_j\right\|_2\right]\right)\frac{2r}{\epsilon}$$
$$\leq \left(D_{\mathbf{p},\mathbf{q},\mathcal{X}} + E_{\mathbf{p},\mathbf{q}}\right)\frac{2r}{\epsilon}, \tag{5.31}$$

where (c) is Markov's inequality, (d) is (5.28) and (5.29). To deal with the centers in (5.30), note $\partial^{\mathbf{p},\mathbf{q}}\tilde{\mathsf{k}}(\mathbf{z})$ can be written as a sample average of $\tilde{d}$ i.i.d. terms as per (5.7) since

$$\partial^{\mathbf{p},\mathbf{q}}\tilde{\mathsf{k}}(\mathbf{z}) = \partial^{\mathbf{p},\mathbf{q}}\left(\frac{1}{\tilde{d}}\sum_{i=1}^{\tilde{d}}\cos(\mathbf{w}_j^\top\mathbf{z})\right) = \frac{1}{\tilde{d}}\sum_{i=1}^{\tilde{d}}\partial^{\mathbf{p},\mathbf{q}}\cos(\mathbf{w}_j^\top\mathbf{z})$$

so that the Bernstein inequality (Cor. 5.16) is applicable. For $j = 1,\ldots,\tilde{d}$, we have

$$\mathbb{E}_{\mathbf{w}_j\sim\Lambda}\left|\partial^{\mathbf{p},\mathbf{q}}\cos\left(\mathbf{w}_j^\top\mathbf{z}\right)\right|^m = \mathbb{E}_{\mathbf{w}_j\sim\Lambda}\left|\mathbf{w}_j^{\mathbf{p}+\mathbf{q}}\cos^{(|\mathbf{p}+\mathbf{q}|)}(\mathbf{w}_j^\top\mathbf{z})\right|^m \leq \mathbb{E}_{\mathbf{w}_j\sim\Lambda}\left|\mathbf{w}_j^{\mathbf{p}+\mathbf{q}}\right|^m \leq \frac{k!S^2R^{k-2}}{2},$$

and that $f(\mathbf{z}) = \partial^{\mathbf{p},\mathbf{q}}\tilde{\mathsf{k}}(\mathbf{z}) - \partial^{\mathbf{p},\mathbf{q}}\mathsf{k}(\mathbf{z}) = \partial^{\mathbf{p},\mathbf{q}}\tilde{\mathsf{k}}(\mathbf{z}) - \mathbb{E}\left[\partial^{\mathbf{p},\mathbf{q}}\tilde{\mathsf{k}}(\mathbf{z})\right]$ by the dominated convergence theorem. Hence, we may call the Bernstein inequality (Cor. 5.16) to control $f(\mathbf{z}_i)$ so that

$$\mathbb{P}\left[|f(\mathbf{z}_i)| \geq \epsilon/2\right] = \mathbb{P}\left[\left|\partial^{\mathbf{p},\mathbf{q}}\tilde{\mathsf{k}}(\mathbf{z}) - \partial^{\mathbf{p},\mathbf{q}}\mathsf{k}(\mathbf{z})\right| \geq \epsilon/2\right] \leq 2\exp\left(\frac{-\tilde{d}\epsilon^2}{4(2S^2+R\epsilon)}\right). \tag{5.32}$$

Combining the bounds for $|f(\mathbf{z}_i)|$ (5.32), and for L_f (5.31), into (5.30) yields

$$\mathbb{P}\left[\sup_{\mathbf{z}\in\mathcal{M}_\Delta}|f(\mathbf{z})| \geq \epsilon\right] \leq 2\left(\frac{4|\mathcal{X}|}{r}\right)^d \exp\left(\frac{-\tilde{d}\epsilon^2}{4(2S^2+R\epsilon)}\right) + \left(D_{\mathbf{p},\mathbf{q},\mathcal{X}} + E_{\mathbf{p},\mathbf{q}}\right)\frac{2r}{\epsilon},$$

which has the form $g(r) = \tau_1 r^{-d} + \tau_2 r$, that is minimized by choosing $r^\star = (d\tau_1/\tau_2)^{\frac{1}{d+1}}$. This choice sets it to the form $g(r^\star) = \tau_1^{\frac{1}{d+1}}\tau_2^{\frac{d}{d+1}}\left(d^{\frac{1}{d+1}} + d^{\frac{-d}{d+1}}\right)$, so that by substituting back in

$$\mathbb{P}\left[\sup_{\mathbf{z}\in\mathcal{M}_\Delta}|f(\mathbf{z})| \geq \epsilon\right] \leq F_d 2^{\frac{3d+1}{d+1}}\left(\frac{|\mathcal{X}|\left(D_{\mathbf{p},\mathbf{q},\mathcal{X}} + E_{\mathbf{p},\mathbf{q}}\right)}{\epsilon}\right)^{\frac{d}{d+1}} \exp\left(\frac{-\tilde{d}\epsilon^2}{4(d+1)(2S^2+R\epsilon)}\right).$$

Finally, we note that for $d \geq 1$, $F_d = d^{\frac{1}{d+1}} + d^{\frac{-d}{d+1}} \leq 2$ and $2^{\frac{3d+1}{d+1}} \leq 8$. □

5.C Bounds on Signature Kernels

We first set the ground for proving our main theorems by introducing the notion of L-Lipschitz kernels to help control distance distortions in the feature space. This subclass of kernels will be useful for us in relating the 1-variation of sequences in feature space to that in the input space. After this, we will prove various smaller lemmas and supplementary results for signature kernels, which will lead up to Lemma 5.35, which is our main tool for proving Theorem 5.3, and it relates the concentration of our RFSF kernel $\tilde{\mathsf{k}}_S$ to the second derivatives of the RFF kernel $\tilde{\mathsf{k}}$. Then, the proof of Theorem 5.3 quantifying the concentration of the RFSF kernel, $\tilde{\mathsf{k}}_S$, will follow from putting together Lemma 5.35 with Theorem 5.26, and we will also make use of the Bernstein inequality from Theorem 5.15. The proof of Theorem 5.6 for the RFSF-DP kernel, $\tilde{\mathsf{k}}_S^{\mathsf{DP}}$, will follow by combining the results of this section with α-exponential concentration, in particular, Theorem 5.23. Finally, Theorem 5.9 for the RFSF-TRP kernel, $\tilde{\mathsf{k}}_S^{\mathsf{TRP}}$, will be proven using lemmas from this section, and the hypercontractivity result from Theorem 5.25.

5.C.1 Distance bounds in the RKHS

Definition 5.27 (Lipschitz kernel)**.** *Let $(\mathcal{X}, d)$ be a metric space. We call a kernel $\mathsf{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ with RKHS $\mathcal{H}_\mathsf{k}$ an L-Lipschitz kernel over $\mathcal{X}$ for some $L > 0$ if it holds for all $\mathbf{x}, \mathbf{y} \in \mathcal{X}$ that*

$$\left\|\mathsf{k}_\mathbf{x} - \mathsf{k}_\mathbf{y}\right\|_{\mathcal{H}_\mathsf{k}} = \sqrt{\mathsf{k}(\mathbf{x},\mathbf{x}) + \mathsf{k}(\mathbf{y},\mathbf{y}) - 2\mathsf{k}(\mathbf{x},\mathbf{y})} \leq L d(\mathbf{x},\mathbf{y}),$$

where $\mathsf{k}_\mathbf{x} = \mathsf{k}(\mathbf{x}, \cdot), \mathsf{k}_\mathbf{y} = \mathsf{k}(\mathbf{y}, \cdot) \in \mathcal{H}_\mathsf{k}$.

Example 5.28 (Finite 2nd spectral moment implies Lipschitz)**.** *Let $\mathsf{k} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be a continuous, bounded and translation-invariant kernel with RKHS $\mathcal{H}_\mathsf{k}$ and spectral measure Λ, such that $\sigma_\Lambda^2 = \mathbb{E}_{\mathbf{w}\sim\Lambda}\left[\|\mathbf{w}\|_2^2\right] < \infty$. Then, it holds that k is $\left\|\mathbb{E}_{\mathbf{w}\sim\Lambda}\left[\mathbf{w}\mathbf{w}^\top\right]\right\|_2^{1/2}$-Lipschitz, so for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ one has*

$$\left\|\mathsf{k}_\mathbf{x} - \mathsf{k}_\mathbf{y}\right\|_{\mathcal{H}_\mathsf{k}} \leq \left\|\mathbb{E}_{\mathbf{w}\sim\Lambda}\left[\mathbf{w}\mathbf{w}^\top\right]\right\|_2^{1/2} \|\mathbf{x} - \mathbf{y}\|_2 .$$

Proof. We have for $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ that

$$\begin{aligned}
\left\|\mathsf{k}_\mathbf{x} - \mathsf{k}_\mathbf{y}\right\|_{\mathcal{H}_\mathsf{k}}^2 &\overset{\text{(a)}}{=} \mathsf{k}(\mathbf{x},\mathbf{x}) + \mathsf{k}(\mathbf{y},\mathbf{y}) - 2\mathsf{k}(\mathbf{x},\mathbf{y}) \overset{\text{(b)}}{=} \int_{\mathbb{R}^d} 2 - 2\exp\left(i\mathbf{w}^\top(\mathbf{x}-\mathbf{y})\right) \mathrm{d}\Lambda(\mathbf{w}) \\
&\overset{\text{(c)}}{=} \int_{\mathbb{R}^d} 2 - 2\cos\left(\mathbf{w}^\top(\mathbf{x}-\mathbf{y})\right) \mathrm{d}\Lambda(\mathbf{w}) \overset{\text{(d)}}{\leq} \int_{\mathbb{R}^d} \left(\mathbf{w}^\top(\mathbf{x}-\mathbf{y})\right)^2 \mathrm{d}\Lambda(\mathbf{w}) \\
&\overset{\text{(e)}}{=} (\mathbf{x}-\mathbf{y})^\top \mathbb{E}_{\mathbf{w}\sim\Lambda}\left[\mathbf{w}\mathbf{w}^\top\right](\mathbf{x}-\mathbf{y}) \overset{\text{(f)}}{\leq} \left\|\mathbb{E}_{\mathbf{w}\sim\Lambda}\left[\mathbf{w}\mathbf{w}^\top\right]\right\|_2 \|\mathbf{x}-\mathbf{y}\|_2^2 ,
\end{aligned}$$

where (a) holds due to the reproducing property, (b) is Bochner's theorem, (c) is because the imaginary part of the integral evaluates to 0 as the kernel is real-valued, (d) is due to the inequality $1 - t^2/2 \leq \cos(t)$ for all $t \in \mathbb{R}$, (e) is because $\left(\mathbf{w}^\top(\mathbf{x}-\mathbf{y})\right)^2 = (\mathbf{x}-\mathbf{y})^\top \mathbf{w}\mathbf{w}^\top(\mathbf{x}-\mathbf{y})$, and (f) is Cauchy-Schwarz inequality combined with the definition of the spectral norm. □

Example 5.29 (Random Lipschitz bound for RFF)**.** *Let $\tilde{\mathsf{k}} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be an* RFF *kernel defined as in* (5.6) *corresponding to some spectral measure Λ, and let $\tilde{\mathcal{H}}_{\mathsf{k}}$ denote its feature space corresponding to the* RFF *map $\tilde{\varphi} : \mathbb{R}^d \to \tilde{\mathcal{H}}_{\mathsf{k}}$ defined as in* (5.6)*, so that given $\mathbf{w}_1, \ldots, \mathbf{w}_{\tilde{d}} \overset{i.i.d.}{\sim} \Lambda$, we have for $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ that*

$$\tilde{\mathsf{k}}(\mathbf{x}, \mathbf{y}) = \frac{1}{\tilde{d}} \sum_{j=1}^{\tilde{d}} \cos\left(\mathbf{w}_j^\top(\mathbf{x}-\mathbf{y})\right).$$

Let $\mathbf{W} = (\mathbf{w}_1, \ldots, \mathbf{w}_{\tilde{d}}) \in \mathbb{R}^{d \times \tilde{d}}$ be the random matrix with column vectors $\mathbf{w}_1, \ldots, \mathbf{w}_{\tilde{d}} \overset{i.i.d.}{\sim} \Lambda$. Then, $\tilde{\mathsf{k}}$ is $\left(\frac{\|\mathbf{W}\|_2}{\sqrt{\tilde{d}}}\right)$-Lipschitz, so that for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, we have the inequality

$$\|\tilde{\varphi}(\mathbf{x}) - \tilde{\varphi}(\mathbf{y})\|_{\tilde{\mathcal{H}}_{\mathsf{k}}} \leq \frac{\|\mathbf{W}\|_2}{\sqrt{\tilde{d}}} \|\mathbf{x}-\mathbf{y}\|_2 .$$

Proof. The proof follows analogously to that of Example 5.28. Let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, then

$$\begin{aligned}\|\tilde{\varphi}(\mathbf{x}) - \tilde{\varphi}(\mathbf{y})\|^2_{\tilde{\mathcal{H}}_{\mathsf{k}}} &\overset{(a)}{=} 2 - \frac{2}{\tilde{d}} \sum_{i=1}^{\tilde{d}} \cos(\mathbf{w}_i^\top(\mathbf{x}-\mathbf{y})) \overset{(b)}{\leq} \frac{1}{\tilde{d}} \sum_{i=1}^{\tilde{d}} \left(\mathbf{w}_i^\top(\mathbf{x}-\mathbf{y})\right)^2 \\ &\overset{(c)}{=} \frac{1}{\tilde{d}} (\mathbf{x}-\mathbf{y})^\top \mathbf{W}\mathbf{W}^\top(\mathbf{x}-\mathbf{y}) \overset{(d)}{\leq} \frac{1}{\tilde{d}} \|\mathbf{W}\|_2^2 \|\mathbf{x}-\mathbf{y}\|_2^2,\end{aligned}$$

where (a) is due to the cosine identity $\cos(a-b) = \cos(a)\cos(b) + \sin(a)\sin(b)$ for all $a, b \in \mathbb{R}$, (b) is due to the inequality $1 - t^2/2 \leq \cos(t)$ for all $t \in \mathbb{R}$, (c) is because $\left(\mathbf{w}_i^\top(\mathbf{x}-\mathbf{y})\right)^2 = (\mathbf{x}-\mathbf{y})^\top \mathbf{w}_i \mathbf{w}_i^\top(\mathbf{x}-\mathbf{y})$ and $\mathbf{W}\mathbf{W}^\top = \sum_{i=1}^{\tilde{d}} \mathbf{w}_i \mathbf{w}_i^\top$, (d) is due to the Cauchy-Schwarz inequality combined with the definition of the spectral norm. □

Bounds for the Signature Kernel. A well-known property of signature features that they decay factorially fast with respect to the tensor level $m \in \mathbb{N}$.

Lemma 5.30 (Norm bound for signature features)**.** *Let $L > 0$ and $\mathsf{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be an L-Lipschitz kernel with RKHS $\mathcal{H}_{\mathsf{k}}$. Then, we have for the level-m signature features $\varphi_{\mathsf{S}_m}(\mathbf{x})$ of the*

sequence $\mathbf{x} \in \mathsf{Seq}(\mathcal{X})$ *that*

$$\left\|\varphi_{\mathsf{S}_m}(\mathbf{x})\right\|_{\mathcal{H}_\mathsf{k}^{\otimes m}} \leq \frac{(L\,\|\mathbf{x}\|_{1\text{-var}})^m}{m!}.$$

Proof. We have

$$\begin{aligned}
\left\|\varphi_{\mathsf{S}_m}(\mathbf{x})\right\|_{\mathcal{H}_\mathsf{k}^{\otimes m}} &= \left\| \sum_{\mathbf{i}\in\Delta_m(\mathsf{L}_\mathbf{x})} \delta\mathsf{k}_{\mathbf{x}_{i_1}} \otimes \cdots \otimes \delta\mathsf{k}_{\mathbf{x}_{i_m}} \right\|_{\mathcal{H}^{\otimes m}} \\
&\overset{\text{(a)}}{\leq} \sum_{\mathbf{i}\in\Delta_m(\mathsf{L}_\mathbf{x})} \left\|\delta\mathsf{k}_{\mathbf{x}_{i_1}}\right\|_{\mathcal{H}} \cdots \left\|\delta\mathsf{k}_{\mathbf{x}_{i_m}}\right\|_{\mathcal{H}} \\
&\overset{\text{(b)}}{\leq} L^m \sum_{\mathbf{i}\in\Delta_m(\mathsf{L}_\mathbf{x})} \left\|\delta\mathbf{x}_{i_1}\right\|_2 \cdots \left\|\delta\mathbf{x}_{i_m}\right\|_2 \\
&\overset{\text{(c)}}{\leq} \frac{\left(L\sum_{i=1}^{\mathsf{L}_\mathbf{x}} \|\delta\mathbf{x}_i\|_2\right)^m}{m!} \overset{\text{(d)}}{=} \frac{(L\,\|\mathbf{x}\|_{1\text{-var}})^m}{m!},
\end{aligned}$$

where (a) follows from triangle inequality followed by factorizing the tensor norm, (b) is the L-Lipschitzness property, (c) from completing the multinomial expansion and normalizing by the number of permutations (note that $\Delta_m(\mathsf{L}_\mathbf{x})$ contains a single permutation of all such multi-indices that have nonrepeating entries), and (d) is the definition of sequence 1-variation. □

The following bound for the signature kernel is a direct consequence of the previous lemma.

chapter 5.31 (Upper bound for signature kernel)**.** *Let* $\mathsf{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ *be an L-Lipschitz kernel, and* $\mathsf{k}_{\mathsf{S}_m} : \mathsf{Seq}(\mathcal{X}) \times \mathsf{Seq}(\mathcal{X}) \to \mathbb{R}$ *the level-m signature kernel built from* k*. Then, we have the following bound for* $\mathbf{x}, \mathbf{y} \in \mathsf{Seq}(\mathcal{X})$

$$\left|\mathsf{k}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y})\right| \leq \frac{\left(L^2\,\|\mathbf{x}\|_{1\text{-var}}\,\|\mathbf{y}\|_{1\text{-var}}\right)^m}{(m!)^2}.$$

Proof. Note that without a kernel trick, $\mathsf{k}_{\mathsf{S}_m}$ is written for $\mathbf{x}, \mathbf{y} \in \mathsf{Seq}(\mathcal{X})$ as the inner product

$$\begin{aligned}
\left|\mathsf{k}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y})\right| &= \left| \left\langle \sum_{\mathbf{i}\in\Delta_m(\mathsf{L}_\mathbf{x})} \delta\mathsf{k}_{\mathbf{x}_{i_1}} \otimes \cdots \otimes \delta\mathsf{k}_{\mathbf{x}_{i_m}}, \sum_{\mathbf{j}\in\Delta_m(\mathsf{L}_\mathbf{y})} \delta\mathsf{k}_{\mathbf{y}_{j_1}} \otimes \cdots \otimes \delta\mathsf{k}_{\mathbf{y}_{j_m}} \right\rangle_{\mathcal{H}_\mathsf{k}^{\otimes m}} \right| \\
&\overset{\text{(a)}}{\leq} \left\| \sum_{\mathbf{i}\in\Delta_m(\mathsf{L}_\mathbf{x})} \delta\mathsf{k}_{\mathbf{x}_{i_1}} \otimes \cdots \otimes \delta\mathsf{k}_{\mathbf{x}_{i_m}} \right\|_{\mathcal{H}_\mathsf{k}^{\otimes m}} \left\| \sum_{\mathbf{j}\in\Delta_m(\mathsf{L}_\mathbf{y})} \delta\mathsf{k}_{\mathbf{y}_{j_1}} \otimes \cdots \otimes \delta\mathsf{k}_{\mathbf{y}_{j_m}} \right\|_{\mathcal{H}_\mathsf{k}^{\otimes m}} \\
&\overset{\text{(b)}}{\leq} \frac{\left(L^2\,\|\mathbf{x}\|_{1\text{-var}}\,\|\mathbf{y}\|_{1\text{-var}}\right)^m}{(m!)^2},
\end{aligned}$$

where (a) follows from the Cauchy-Schwarz inequality, and (b) is implied by Lemma 5.30. □

A similar upper bound to Lemma 5.30 also holds for the RFSF kernel $\tilde{\mathsf{k}}_{\mathsf{S}_m}$, that now depends on the norms of the random matrices $\mathbf{W}^{(1)}, \dots, \mathbf{W}^{(m)}$, hence is itself random.

Lemma 5.32 (Random norm bound for RFSF)**.** *Let $\tilde{\varphi}_{\mathsf{S}_m} : \mathsf{Seq}(\mathcal{X}) \to \tilde{\mathcal{H}}_{\mathsf{k}\mathsf{S}}$ be the level-m RFSF map defined as in* (5.8) *built from some spectral measure Λ. Then, we have for $\mathbf{x} \in \mathsf{Seq}(\mathcal{X})$ that*

$$\left\|\tilde{\varphi}_{\mathsf{S}_m}(\mathbf{x})\right\|_{\tilde{\mathcal{H}}_{\tilde{\mathsf{k}}}{}^{\otimes m}} \leq \frac{\left\|\mathbf{W}^{(1)}\right\|_2 \cdots \left\|\mathbf{W}^{(m)}\right\|_2}{m!} \left(\frac{\|\mathbf{x}\|_{1\text{-var}}}{\sqrt{\tilde{d}}}\right)^m,$$

where $\mathbf{W}^{(1)}, \dots, \mathbf{W}^{(m)} \overset{i.i.d.}{\sim} \Lambda^{\tilde{d}}$ are random matrices sampled from $\Lambda^{\tilde{d}}$.

Proof. The proof follows analogously to Lemma 5.30. We have for $\mathbf{x} \in \mathsf{Seq}(\mathcal{X})$ that

$$\begin{aligned}
\left\|\tilde{\varphi}_{\mathsf{S}_m}(\mathbf{x})\right\|_{\tilde{\mathcal{H}}_{\tilde{\mathsf{k}}}{}^{\otimes m}} &= \left\| \sum_{\mathbf{i} \in \Delta_m(\mathsf{L}_\mathbf{x})} \delta\tilde{\varphi}_1(\mathbf{x}_{i_1}) \otimes \cdots \otimes \delta\tilde{\varphi}_m(\mathbf{x}_{i_m}) \right\|_{\tilde{\mathcal{H}}_{\tilde{\mathsf{k}}}{}^{\otimes m}} \\
&\overset{(a)}{\leq} \sum_{\mathbf{i} \in \Delta_m(\mathsf{L}_\mathbf{x})} \left\|\delta\tilde{\varphi}_1(\mathbf{x}_{i_1})\right\|_{\tilde{\mathcal{H}}_{\tilde{\mathsf{k}}}} \cdots \left\|\delta\tilde{\varphi}_m(\mathbf{x}_{i_m})\right\|_{\tilde{\mathcal{H}}_{\tilde{\mathsf{k}}}} \\
&\overset{(b)}{\leq} \frac{\left\|\mathbf{W}^{(1)}\right\|_2 \cdots \left\|\mathbf{W}^{(m)}\right\|_2}{\tilde{d}^{m/2}} \sum_{\mathbf{i} \in \Delta_m(\mathsf{L}_\mathbf{x})} \left\|\delta\mathbf{x}_{i_1}\right\|_2 \cdots \left\|\delta\mathbf{x}_{i_m}\right\|_2 \\
&\overset{(c)}{\leq} \frac{\left\|\mathbf{W}^{(1)}\right\|_2 \cdots \left\|\mathbf{W}^{(m)}\right\|_2}{\tilde{d}^{m/2}} \frac{\|\mathbf{x}\|_{1\text{-var}}^m}{m!},
\end{aligned}$$

where (a) is the triangle inequality and factorization of tensor norm, (b) is using the Lipschitzness of RFFs from Example 5.29, (c) is the same as steps (c)-(d) in Lemma 5.30. □

Then, the following is again an application of the Cauchy-Schwarz inequality.

chapter 5.33 (Random upper bound for RFSF kernel)**.** *Let $\tilde{\mathsf{k}}_{\mathsf{S}_m} : \mathsf{Seq}(\mathcal{X}) \times \mathsf{Seq}(\mathcal{X}) \to \mathbb{R}$ be the level-m RFSF kernel defined as in* (5.9) *built from some spectral measure Λ. Then, for all $\mathbf{x}, \mathbf{y} \in \mathsf{Seq}(\mathcal{X})$*

$$\left|\tilde{\mathsf{k}}_{\mathsf{S}_m}(\mathbf{x}, \mathbf{y})\right| \leq \frac{\left\|\mathbf{W}^{(1)}\right\|_2^2 \cdots \left\|\mathbf{W}^{(m)}\right\|_2^2}{(m!)^2} \left(\frac{\|\mathbf{x}\|_{1\text{-var}} \|\mathbf{y}\|_{1\text{-var}}}{\tilde{d}}\right)^m$$

where $\mathbf{W}^{(1)}, \dots, \mathbf{W}^{(m)} \overset{i.i.d.}{\sim} \Lambda^{\tilde{d}}$ are random matrices sampled from $\Lambda^{\tilde{d}}$.

The following lemma does not concern signatures, but will be useful to us later in the proof of Lemma 5.35 by providing a mean-value theorem for the cross-differencing operator $\delta_{i,j}$.

Lemma 5.34 (2^{nd} order mean-value theorem)**.** *Let $f : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be a twice differentiable function, and $\mathcal{X} \subset \mathbb{R}^d$ be a convex and compact set. Then, we have for any $\mathbf{u}, \mathbf{v}, \mathbf{x}, \mathbf{y} \in \mathcal{X}$ that*

$$f(\mathbf{x},\mathbf{y}) - f(\mathbf{x},\mathbf{v}) - f(\mathbf{u},\mathbf{y}) + f(\mathbf{u},\mathbf{v}) \leq \sup_{\mathbf{s},\mathbf{t}\in\mathcal{X}} \left\| \partial^2_{\mathbf{s},\mathbf{t}} f(\mathbf{s},\mathbf{t}) \right\|_2 \|\mathbf{x}-\mathbf{u}\|_2 \|\mathbf{y}-\mathbf{v}\|_2 ,$$

where $\partial^2_{\mathbf{s},\mathbf{t}} f(\mathbf{s},\mathbf{t}) = \left(\frac{\partial^2 f(\mathbf{s},\mathbf{t})}{\partial s_i \partial t_j} \right)_{i,j=1}^d$ refers to the submatrix of the Hessian of cross-derivatives.

Proof. Keeping $\mathbf{v}, \mathbf{y} \in \mathcal{X}$ as fixed, we may define $g : \mathbb{R}^d \to \mathbb{R}$ as $g(\cdot) = f(\cdot,\mathbf{y}) - f(\cdot,\mathbf{v})$, so that the expression above can be written as $g(\mathbf{x}) - g(\mathbf{u})$, and by the convexity of $\mathcal{X}$, we may apply the mean-value theorem to find that $\exists s \in (0,1)$ such that

$$g(\mathbf{x}) - g(\mathbf{u}) = \langle \nabla g(s\mathbf{x} + (1-s)\mathbf{u}), \mathbf{x}-\mathbf{u} \rangle \leq \sup_{\mathbf{s}\in\mathcal{X}} \|\nabla g(\mathbf{s})\|_2 \|\mathbf{x}-\mathbf{u}\|_2 , \tag{5.33}$$

where $\nabla g(\mathbf{s}) = \left(\frac{\partial g(\mathbf{s})}{\partial s_i} \right)_{i=1}^d$ denotes the gradient of g, while the second inequality follows from the Cauchy-Schwarz inequality and the compactness of $\mathcal{X}$ (so that the sup exists). Also note that $\nabla g(\mathbf{s}) = \partial_{\mathbf{s}} f(\mathbf{s},\mathbf{y}) - \partial_{\mathbf{s}} f(\mathbf{s},\mathbf{v})$, so defining $h : \mathbb{R}^d \to \mathbb{R}^d$ as $h(\cdot) = \partial_{\mathbf{s}} f(\mathbf{s},\cdot)$ and applying the vector-valued mean-value inequality [221, Thm. 9.19] to h gives that $\exists t \in (0,1)$ such that

$$\|h(\mathbf{y}) - h(\mathbf{v})\|_2 \leq \|\mathrm{J}_h(t\mathbf{y} + (1-t)\mathbf{v})\|_2 \|\mathbf{y}-\mathbf{v}\|_2 \leq \sup_{\mathbf{t}\in\mathcal{X}} \|\mathrm{J}_h(\mathbf{t})\|_2 \|\mathbf{y}-\mathbf{v}\|_2 , \tag{5.34}$$

where $\mathrm{J}_h(\mathbf{t}) = \left(\frac{\partial h_i(\mathbf{t})}{\partial t_j} \right)_{i,j=1}^d$ refers to the Jacobian of h. Putting the inequalities (5.33) and (5.34) together and substituting back the function f gives the desired result. □

This is our final lemma in our exposition of supplementary results about the (random) signature kernels, and it will be our main tool for proving Theorem 5.3.

Lemma 5.35 (Uniform upper bound for deviation of RFSF kernel)**.** *Let $\mathcal{X} \subset \mathbb{R}^d$ be a convex and compact set, $\mathsf{k} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ a continuous, bounded, translation-invariant L-Lipschitz kernel and $\tilde{\mathsf{k}} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ the corresponding* RFF *kernel. Then, the level-m ($m \in \mathbb{N}$) signature and* RFSF *kernels are uniformly close for $V > 0$ by*

$$\begin{aligned} &\sup_{\substack{\mathbf{x},\mathbf{y}\in\mathsf{Seq}(\mathcal{X}) \\ \|\mathbf{x}\|_{1\text{-var}},\|\mathbf{y}\|_{1\text{-var}}\leq V}} \left| \tilde{\mathsf{k}}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y}) - \mathsf{k}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y}) \right| \\ &\leq V^{2m} \sum_{k=1}^{m} \frac{L^{2(m-k)}}{\tilde{d}^{k-1}((k-1)!)^2} \left\| \mathbf{W}^{(1)} \right\|_2^2 \cdots \left\| \mathbf{W}^{(k-1)} \right\|_2^2 \sup_{\mathbf{s},\mathbf{t}\in\mathcal{X}} \left\| \partial^2_{\mathbf{s},\mathbf{t}} \tilde{\mathsf{k}}_k(\mathbf{s},\mathbf{t}) - \partial^2_{\mathbf{s},\mathbf{t}} \mathsf{k}(\mathbf{s},\mathbf{t}) \right\|_2 , \end{aligned}$$

where $\tilde{\mathsf{k}}_1, \dots, \tilde{\mathsf{k}}_m$ *are independent* RFF *kernels with weights* $\mathbf{W}^{(1)}, \dots, \mathbf{W}^{(m)} \overset{i.i.d.}{\sim} \Lambda^{\tilde{d}}$*, and* $\partial^2_{\mathbf{s},\mathbf{t}} f(\mathbf{s},\mathbf{t}) = \left(\frac{\partial^2 f(\mathbf{s},\mathbf{t})}{\partial s_i \partial t_j}\right)_{i,j=1}^{d}$ *for a twice-differentiable function* $f : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$.

Proof. First of all, by Lemma 5.31 and Lemma 5.33, the supremum exists. In the following, given a sequence $\mathbf{x} \in \mathsf{Seq}(\mathcal{X})$, we denote its $0 : l$ slice for some $l \in [\mathsf{L}_\mathbf{x}]$ by $\mathbf{x}_{:l} = (\mathbf{x}_0, \dots, \mathbf{x}_l)$. Then, it holds for any $m \geq 1$ recursively for the signature kernel that

$$\mathsf{k}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y}) = \sum_{k=1}^{\mathsf{L}_\mathbf{x}} \sum_{l=1}^{\mathsf{L}_\mathbf{y}} \mathsf{k}_{\mathsf{S}_{m-1}}(\mathbf{x}_{0:k-1}, \mathbf{y}_{0:l-1}) \delta_{k,l} \mathsf{k}(\mathbf{x}_k, \mathbf{y}_l),$$

and analogously for the RFSF kernel that

$$\tilde{\mathsf{k}}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y}) = \sum_{k=1}^{\mathsf{L}_\mathbf{x}} \sum_{l=1}^{\mathsf{L}_\mathbf{y}} \tilde{\mathsf{k}}_{\mathsf{S}_{m-1}}(\mathbf{x}_{0:k-1}, \mathbf{y}_{0:l-1}) \delta_{k,l} \tilde{\mathsf{k}}_m(\mathbf{x}_k, \mathbf{y}_l).$$

Combining these recursions together, we have for the uniform error that

$$\begin{aligned}
\epsilon_m &= \sup_{\substack{\mathbf{x},\mathbf{y} \in \mathsf{Seq}(\mathcal{X}) \\ \|\mathbf{x}\|_{1\text{-var}}, \|\mathbf{y}\|_{1\text{-var}} \leq V}} \left| \tilde{\mathsf{k}}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y}) - \mathsf{k}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y}) \right| \\
&= \sup_{\substack{\mathbf{x},\mathbf{y} \in \mathsf{Seq}(\mathcal{X}) \\ \|\mathbf{x}\|_{1\text{-var}}, \|\mathbf{y}\|_{1\text{-var}} \leq V}} \left| \sum_{k=1}^{\mathsf{L}_\mathbf{x}} \sum_{l=1}^{\mathsf{L}_\mathbf{y}} \tilde{\mathsf{k}}_{\mathsf{S}_{m-1}}(\mathbf{x}_{0:k-1}, \mathbf{y}_{0:l-1}) \delta_{k,l} \tilde{\mathsf{k}}_m(\mathbf{x}_k, \mathbf{y}_l) - \mathsf{k}_{\mathsf{S}_{m-1}}(\mathbf{x}_{0:k-1}, \mathbf{y}_{0:l-1}) \delta_{k,l} \mathsf{k}(\mathbf{x}_k, \mathbf{y}_l) \right| \\
&\overset{(a)}{\leq} \sup_{\substack{\mathbf{x},\mathbf{y} \in \mathsf{Seq}(\mathcal{X}) \\ \|\mathbf{x}\|_{1\text{-var}}, \|\mathbf{y}\|_{1\text{-var}} \leq V}} \left| \sum_{k=1}^{\mathsf{L}_\mathbf{x}} \sum_{l=1}^{\mathsf{L}_\mathbf{y}} \tilde{\mathsf{k}}_{\mathsf{S}_{m-1}}(\mathbf{x}_{0:k-1}, \mathbf{y}_{0:l-1}) (\delta_{k,l} \tilde{\mathsf{k}}_m(\mathbf{x}_k, \mathbf{y}_l) - \delta_{k,l} \mathsf{k}(\mathbf{x}_k, \mathbf{y}_l)) \right| \\
&\quad + \sup_{\substack{\mathbf{x},\mathbf{y} \in \mathsf{Seq}(\mathcal{X}) \\ \|\mathbf{x}\|_{1\text{-var}}, \|\mathbf{y}\|_{1\text{-var}} \leq V}} \left| \sum_{k=1}^{\mathsf{L}_\mathbf{x}} \sum_{l=1}^{\mathsf{L}_\mathbf{y}} (\tilde{\mathsf{k}}_{\mathsf{S}_{m-1}}(\mathbf{x}_{0:k-1}, \mathbf{y}_{0:l-1}) - \tilde{\mathsf{k}}_{\mathsf{S}_{m-1}}(\mathbf{x}_{0:k-1}, \mathbf{y}_{0:l-1})) \delta_{k,l} \mathsf{k}(\mathbf{x}_k, \mathbf{y}_l) \right| \\
&\overset{(b)}{\leq} \sup_{\substack{\mathbf{x},\mathbf{y} \in \mathsf{Seq}(\mathcal{X}) \\ \|\mathbf{x}\|_{1\text{-var}}, \|\mathbf{y}\|_{1\text{-var}} \leq V}} \sum_{k=1}^{\mathsf{L}_\mathbf{x}} \sum_{l=1}^{\mathsf{L}_\mathbf{y}} \left| \tilde{\mathsf{k}}_{\mathsf{S}_{m-1}}(\mathbf{x}_{0:k-1}, \mathbf{y}_{0:l-1}) \right| \left| \delta_{k,l} \tilde{\mathsf{k}}_m(\mathbf{x}_k, \mathbf{y}_l) - \delta_{k,l} \mathsf{k}(\mathbf{x}_k, \mathbf{y}_l) \right| \\
&\quad + \sup_{\substack{\mathbf{x},\mathbf{y} \in \mathsf{Seq}(\mathcal{X}) \\ \|\mathbf{x}\|_{1\text{-var}}, \|\mathbf{y}\|_{1\text{-var}} \leq V}} \sum_{k=1}^{\mathsf{L}_\mathbf{x}} \sum_{l=1}^{\mathsf{L}_\mathbf{y}} \left| \tilde{\mathsf{k}}_{\mathsf{S}_{m-1}}(\mathbf{x}_{0:k-1}, \mathbf{y}_{0:l-1}) - \mathsf{k}_{\mathsf{S}_{m-1}}(\mathbf{x}_{0:k-1}, \mathbf{y}_{0:l-1}) \right| \left| \delta_{k,l} \mathsf{k}(\mathbf{x}_k, \mathbf{y}_l) \right| \\
&\overset{(c)}{\leq} \underbrace{\sup_{\substack{\mathbf{x},\mathbf{y} \in \mathsf{Seq}(\mathcal{X}) \\ \|\mathbf{x}\|_{1\text{-var}}, \|\mathbf{y}\|_{1\text{-var}} \leq V}} \left| \tilde{\mathsf{k}}_{\mathsf{S}_{m-1}}(\mathbf{x},\mathbf{y}) \right|}_{\text{(i)}} \underbrace{\sup_{\substack{\mathbf{x},\mathbf{y} \in \mathsf{Seq}(\mathcal{X}) \\ \|\mathbf{x}\|_{1\text{-var}}, \|\mathbf{y}\|_{1\text{-var}} \leq V}} \sum_{k=1}^{\mathsf{L}_\mathbf{x}} \sum_{l=1}^{\mathsf{L}_\mathbf{y}} \left| \delta_{k,l} \tilde{\mathsf{k}}_m(\mathbf{x}_k, \mathbf{y}_l) - \delta_{k,l} \mathsf{k}(\mathbf{x}_k, \mathbf{y}_l) \right|}_{\text{(ii)}}
\end{aligned}$$

$$+ \underbrace{\sup_{\substack{\mathbf{x},\mathbf{y}\in\mathrm{Seq}(\mathcal{X})\\ \|\mathbf{x}\|_{1\text{-var}},\|\mathbf{y}\|_{1\text{-var}}\le V}} \left|\tilde{\mathsf{k}}_{\mathsf{S}_{m-1}}(\mathbf{x},\mathbf{y}) - \mathsf{k}_{\mathsf{S}_{m-1}}(\mathbf{x},\mathbf{y})\right|}_{\text{(iii)}} \underbrace{\sup_{\substack{\mathbf{x},\mathbf{y}\in\mathrm{Seq}(\mathcal{X})\\ \|\mathbf{x}\|_{1\text{-var}},\|\mathbf{y}\|_{1\text{-var}}\le V}} \sum_{k=1}^{\mathsf{L}_\mathbf{x}}\sum_{l=1}^{\mathsf{L}_\mathbf{y}} \left|\delta_{k,l}\mathsf{k}(\mathbf{x}_k,\mathbf{y}_l)\right|}_{\text{(iv)}}, \tag{5.35}$$

where (a) follows from adding and subtracting the cross-terms and applying triangle inequality, (b) follows from applying triangle inequality over the summations, (c) follows from noting that if $\|\mathbf{x}\|_{1\text{-var}}, \|\mathbf{y}\|_{1\text{-var}} \le V$ then so is $\|\mathbf{x}_{0:k-1}\|_{1\text{-var}}, \|\mathbf{y}_{0:l-1}\|_{1\text{-var}} \le V$ for $k \in [\mathsf{L}_\mathbf{x}]$ and $l \in [\mathsf{L}_\mathbf{y}]$, and thus justifiably pulling out the supremums.

Now, we deal with terms (i)–(iv) individually. For (i), we have Corollary 5.33, so

$$\sup_{\substack{\mathbf{x},\mathbf{y}\in\mathrm{Seq}(\mathcal{X})\\ \|\mathbf{x}\|_{1\text{-var}},\|\mathbf{y}\|_{1\text{-var}}\le V}} \left|\tilde{\mathsf{k}}_{\mathsf{S}_{m-1}}(\mathbf{x},\mathbf{y})\right| \le \frac{\left\|\mathbf{W}^{(1)}\right\|_2^2 \cdots \left\|\mathbf{W}^{(m-1)}\right\|_2^2}{((m-1)!)^2}\left(\frac{V^2}{\tilde{d}}\right)^{m-1}. \tag{5.36}$$

To deal with (ii), we can apply Lemma 5.34 with $f = \tilde{\mathsf{k}}_m - \mathsf{k}$ to get

$$\begin{aligned}
&\sup_{\substack{\mathbf{x},\mathbf{y}\in\mathrm{Seq}(\mathcal{X})\\ \|\mathbf{x}\|_{1\text{-var}},\|\mathbf{y}\|_{1\text{-var}}\le V}} \sum_{k=1}^{\mathsf{L}_\mathbf{x}}\sum_{l=1}^{\mathsf{L}_\mathbf{y}} \left|\delta_{k,l}\tilde{\mathsf{k}}_m(\mathbf{x}_k,\mathbf{y}_l) - \delta_{k,l}\mathsf{k}(\mathbf{x}_k,\mathbf{y}_l)\right| \\
&\le \sup_{\mathbf{s},\mathbf{t}\in\mathcal{X}} \left\|\partial^2_{\mathbf{s},\mathbf{t}}\tilde{\mathsf{k}}_m(\mathbf{s},\mathbf{t}) - \partial^2_{\mathbf{s},\mathbf{t}}\mathsf{k}(\mathbf{s},\mathbf{t})\right\|_2 \sup_{\substack{\mathbf{x},\mathbf{y}\in\mathrm{Seq}(\mathcal{X})\\ \|\mathbf{x}\|_{1\text{-var}},\|\mathbf{y}\|_{1\text{-var}}\le V}} \sum_{k=1}^{\mathsf{L}_\mathbf{x}}\sum_{l=1}^{\mathsf{L}_\mathbf{y}} \|\delta\mathbf{x}_k\|_2 \|\delta\mathbf{y}_l\|_2 \\
&\le V^2 \sup_{\mathbf{s},\mathbf{t}\in\mathcal{X}} \left\|\partial^2_{\mathbf{s},\mathbf{t}}\tilde{\mathsf{k}}_m(\mathbf{s},\mathbf{t}) - \partial^2_{\mathbf{s},\mathbf{t}}\mathsf{k}(\mathbf{s},\mathbf{t})\right\|_2.
\end{aligned} \tag{5.37}$$

For (iii), we note that it is simply ϵ_{m-1}. Finally, we can write (iv) as an inner product and apply Cauchy-Schwarz and L-Lipschitzness of k so that

$$\begin{aligned}
&\sup_{\substack{\mathbf{x},\mathbf{y}\in\mathrm{Seq}(\mathcal{X})\\ \|\mathbf{x}\|_{1\text{-var}},\|\mathbf{y}\|_{1\text{-var}}\le V}} \sum_{k=1}^{\mathsf{L}_\mathbf{x}}\sum_{l=1}^{\mathsf{L}_\mathbf{y}} \left|\delta_{k,l}\mathsf{k}(\mathbf{x}_k,\mathbf{y}_l)\right| = \sup_{\substack{\mathbf{x},\mathbf{y}\in\mathrm{Seq}(\mathcal{X})\\ \|\mathbf{x}\|_{1\text{-var}},\|\mathbf{y}\|_{1\text{-var}}\le V}} \sum_{k=1}^{\mathsf{L}_\mathbf{x}}\sum_{l=1}^{\mathsf{L}_\mathbf{y}} \left|\left\langle \delta\mathsf{k}_{\mathbf{x}_k}, \delta\mathsf{k}_{\mathbf{y}_l}\right\rangle\right| \\
&\le \sup_{\substack{\mathbf{x},\mathbf{y}\in\mathrm{Seq}(\mathcal{X})\\ \|\mathbf{x}\|_{1\text{-var}},\|\mathbf{y}\|_{1\text{-var}}\le V}} \sum_{k=1}^{\mathsf{L}_\mathbf{x}}\sum_{l=1}^{\mathsf{L}_\mathbf{y}} \left\|\delta\mathsf{k}_{\mathbf{x}_k}\right\|_{\mathcal{H}_\mathsf{k}} \left\|\delta\mathsf{k}_{\mathbf{y}_l}\right\|_{\mathcal{H}_\mathsf{k}} \le L^2V^2.
\end{aligned} \tag{5.38}$$

Putting equations (5.36), (5.37), (5.38) together in (5.35), we get that

$$\epsilon_m = \sup_{\substack{\mathbf{x},\mathbf{y}\in\mathrm{Seq}(\mathcal{X})\\ \|\mathbf{x}\|_{1\text{-var}},\|\mathbf{y}\|_{1\text{-var}}\le V}} \left|\tilde{\mathsf{k}}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y}) - \mathsf{k}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y})\right|$$

$$\leq \frac{V^{2m}}{\tilde{d}^{m-1}((m-1)!)^2} \sup_{\mathbf{s},\mathbf{t}\in\mathcal{X}} \left\| \partial^2_{\mathbf{s},\mathbf{t}} \tilde{\mathsf{k}}_m(\mathbf{s},\mathbf{t}) - \partial^2_{\mathbf{s},\mathbf{t}} \mathsf{k}(\mathbf{s},\mathbf{t}) \right\|_2 \prod_{p=1}^{m-1} \left\| \mathbf{W}^{(p)} \right\|_2^2 + L^2 V^2 \epsilon_{m-1}, \quad (5.39)$$

which gives us a recursion for estimating ϵ_m. The initial step, $m = 1$, can be estimated by

$$\begin{aligned}
\epsilon_1 &= \sup_{\substack{\mathbf{x},\mathbf{y}\in \mathsf{Seq}(\mathcal{X}) \\ \|\mathbf{x}\|_{1\text{-var}},\|\mathbf{y}\|_{1\text{-var}}\leq V}} \left| \tilde{\mathsf{k}}_{\mathsf{S}_1}(\mathbf{x},\mathbf{y}) - \mathsf{k}_{\mathsf{S}_1}(\mathbf{x},\mathbf{y}) \right| \\
&= \sup_{\substack{\mathbf{x},\mathbf{y}\in \mathsf{Seq}(\mathcal{X}) \\ \|\mathbf{x}\|_{1\text{-var}},\|\mathbf{y}\|_{1\text{-var}}\leq V}} \left| \sum_{k=1}^{\mathsf{L}_\mathsf{x}} \sum_{l=1}^{\mathsf{L}_\mathsf{y}} \delta_{k,l} \tilde{\mathsf{k}}_1(\mathbf{x}_k,\mathbf{y}_l) - \delta_{k,l} \mathsf{k}(\mathbf{x}_k,\mathbf{y}_l) \right| \\
&= \sup_{\substack{\mathbf{x},\mathbf{y}\in \mathsf{Seq}(\mathcal{X}) \\ \|\mathbf{x}\|_{1\text{-var}},\|\mathbf{y}\|_{1\text{-var}}\leq V}} \sum_{k=1}^{\mathsf{L}_\mathsf{x}} \sum_{l=0}^{\mathsf{L}_\mathsf{y}} \left| \delta_{k,l} \tilde{\mathsf{k}}_1(\mathbf{x}_k,\mathbf{y}_l) - \delta_{k,l} \mathsf{k}(\mathbf{x}_k,\mathbf{y}_l) \right| \\
&\leq V^2 \sup_{\mathbf{s},\mathbf{t}\in\mathcal{X}} \left\| \partial^2_{\mathbf{s},\mathbf{t}} \tilde{\mathsf{k}}_1(\mathbf{s},\mathbf{t}) - \partial^2_{\mathbf{s},\mathbf{t}} \mathsf{k}(\mathbf{s},\mathbf{t}) \right\|_2, \qquad (5.40)
\end{aligned}$$

which is actually analogous to (5.39) since $\epsilon_0 = 0$. Now, we may unroll the recursion (5.39) with the initial condition (5.40), and we get

$$\epsilon_m \leq V^{2m} \sum_{k=1}^{m} \frac{L^{2(m-k)}}{\tilde{d}^{k-1}((k-1)!)^2} \sup_{\mathbf{s},\mathbf{t}\in\mathcal{X}} \left\| \partial^2_{\mathbf{s},\mathbf{t}} \tilde{\mathsf{k}}_k(\mathbf{s},\mathbf{t}) - \partial^2_{\mathbf{s},\mathbf{t}} \mathsf{k}(\mathbf{s},\mathbf{t}) \right\|_2 \prod_{p=1}^{k-1} \left\| \mathbf{W}^{(p)} \right\|_2^2.$$

□

5.C.2 Proofs of main concentration results

Here, we provide proofs of the main concentration results, i.e. Theorems 5.3, 5.6, 5.9, respectively under Theorems 5.36, 5.37, 5.38.

Theorem 5.36 (Concentration inequality for RFSF kernel)**.** *Let $\mathcal{X} \subset \mathbb{R}^d$ be a compact and convex set with diameter $|\mathcal{X}|$, and $\mathcal{X}_\Delta = \{\mathbf{x}-\mathbf{y} : \mathbf{x},\mathbf{y} \in \mathcal{X}\}$. Let $\mathsf{k} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be a continuous, bounded, translation-invariant kernel with spectral measure Λ, which satisfies for some $S, R > 0$ that*

$$\mathbb{E}_{\mathbf{w}\sim\Lambda}\left[|w_i|^{2k} \right] \leq \frac{k! S^2 R^{k-2}}{2} \quad \text{for all } i \in [d] \text{ and } k \geq 2. \qquad (5.41)$$

Then, the following quantities are finite: $\sigma^2_\Lambda = \mathbb{E}_{\mathbf{w}\sim\Lambda}\left[\|\mathbf{w}\|_2^2 \right]$, $L = \left\| \mathbb{E}_{\mathbf{w}\sim\Lambda}\left[\mathbf{w}\mathbf{w}^\top \right] \right\|_2^{1/2}$, $E_{i,j} = \mathbb{E}_{\mathbf{w}\sim\Lambda}\left[\left| w_i w_j \right| \|\mathbf{w}\|_2 \right]$ and $D_{i,j} = \sup_{\mathbf{z}\in\mathcal{X}_\Delta} \left\| \nabla \left[\frac{\partial^2 \mathsf{k}(\mathbf{z})}{\partial z_i \partial z_j} \right] \right\|_2$ for $i,j \in [d]$. Further, for any maximal sequence 1-variation $V > 0$, and signature level $m \in \mathbb{Z}_+$, it holds for the level-m RFSF kernel $\tilde{\mathsf{k}}_{\mathsf{S}_m} : \mathsf{Seq}(\mathcal{X}) \times \mathsf{Seq}(\mathcal{X}) \to \mathbb{R}$ defined as in (5.9) and the signature kernel $\mathsf{k}_{\mathsf{S}_m} : \mathsf{Seq}(\mathcal{X}) \times \mathsf{Seq}(\mathcal{X}) \to \mathbb{R}$

defined as in (5.5) *for* $\epsilon > 0$ *that*

$$\mathbb{P}\left[\sup_{\substack{\mathbf{x},\mathbf{y}\in\mathsf{Seq}(\mathcal{X})\\ \|\mathbf{x}\|_{1\text{-var}},\|\mathbf{y}\|_{1\text{-var}}\le V}} \left|\mathsf{k}_{S_m}(\mathbf{x},\mathbf{y}) - \tilde{\mathsf{k}}_{S_m}(\mathbf{x},\mathbf{y})\right| \ge \epsilon\right] \le$$
$$\le m \begin{cases} \left(C_{d,\mathcal{X}}\left(\frac{\beta_{d,m,V}}{\epsilon}\right)^{\frac{d}{d+1}} + d\right)\exp\left(-\frac{\tilde{d}}{2(d+1)(S^2+R)}\left(\frac{\epsilon}{\beta_{d,m,V}}\right)^2\right) & \text{for } \epsilon < \beta_{d,m,V} \\ \left(C_{d,\mathcal{X}}\left(\frac{\beta_{d,m,V}}{\epsilon}\right)^{\frac{d}{(d+1)m}} + d\right)\exp\left(-\frac{\tilde{d}}{2(d+1)(S^2+R)}\left(\frac{\epsilon}{\beta_{d,m,V}}\right)^{\frac{1}{m}}\right) & \text{for } \epsilon \ge \beta_{d,m,V}, \end{cases}$$

where $C_{d,\mathcal{X}} = 2^{\frac{1}{d+1}} 16\,|\mathcal{X}|^{\frac{d}{d+1}} \sum_{i,j=1}^{d} \left(D_{i,j} + E_{i,j}\right)^{\frac{d}{d+1}}$ *and* $\beta_{d,m,V} = m\left(2V^2\left(L^2 \vee 1\right)\left(\sigma_\Lambda^2 \vee d\right)\right)^m$.

Proof. Finite quantities. To start off with, due to (5.41) with $m = 2$ and Jensen's inequality (Lemma 5.11), we get

$$\mathbb{E}\left[|w_i|^2\right] \le \mathbb{E}^{1/2}\left[|w_i|^4\right] < \infty \quad \text{for all } i \in [d].$$

Hence, by linearity of the expectation $\sigma_\Lambda^2 = \mathbb{E}\left[\|\mathbf{w}\|^2\right] = \sum_{i=1}^{d} \mathbb{E}\left[w_i^2\right] < \infty$. Next, due to Hölder's inequality (Lemma 5.12), it holds that

$$\mathbb{E}\left[w_i w_j\right] \le \mathbb{E}\left[\left|w_i w_j\right|\right] \le \mathbb{E}^{1/2}\left[|w_i|^2\right]\mathbb{E}^{1/2}\left[\left|w_j\right|^2\right] < \infty \quad \text{for all } i,j \in [d],$$

therefore $L < \infty$. Further, applying Hölder's inequality twice followed by Jensen's inequality,

$$\begin{aligned}\mathbb{E}\left[\left|w_i w_j w_k\right|\right] &\le \mathbb{E}^{2/3}\left[\left|w_i w_j\right|^{3/2}\right]\mathbb{E}^{1/3}\left[|w_k|^3\right] \le \mathbb{E}^{1/3}\left[|w_i|^3\right]\mathbb{E}^{1/3}\left[\left|w_j\right|^3\right]\mathbb{E}^{1/3}\left[|w_k|^3\right] \\ &\le \mathbb{E}^{1/6}\left[|w_i|^6\right]\mathbb{E}^{1/6}\left[\left|w_j\right|^6\right]\mathbb{E}^{1/6}\left[|w_k|^6\right] < \infty \quad \text{for all } i,j,k \in [d], \end{aligned} \tag{5.42}$$

which is finite due to (5.41) with $m = 3$. Now, because of the ℓ^1-ℓ^2 norm inequality,

$$\mathbb{E}\left[\left|w_i w_j\right| \|\mathbf{w}\|_2\right] \le \mathbb{E}\left[\left|w_i w_j\right| \|\mathbf{w}\|_1\right] = \sum_{k=1}^{d} \mathbb{E}\left[\left|w_i w_j w_k\right|\right] < \infty,$$

hence $\bar{E} < \infty$. Next, as per [230, Thm. 1.2.1.(iii)], as (5.42) holds for all $i,j,k \in [d]$, k is 3-times continuously differentiable, which combined with the compactness of $\mathcal{X}$, hence that of

$\mathcal{X}_\Delta$, gives $\sup_{\mathbf{z}\in\mathcal{X}_\Delta}\left|\frac{\partial^3 \mathsf{k}(\mathbf{z})}{\partial z_i \partial z_j \partial z_k}\right| < \infty$. Finally, from the ℓ^1-ℓ^2 inequality again, we get that

$$\sup_{\mathbf{z}\in\mathcal{X}_\Delta}\left\|\nabla\left[\frac{\partial^2 \mathsf{k}(\mathbf{z})}{\partial z_i \partial z_j}\right]\right\|_2 \leq \sup_{\mathbf{z}\in\mathcal{X}_\Delta}\left\|\nabla\left[\frac{\partial^2 \mathsf{k}(\mathbf{z})}{\partial z_i \partial z_j}\right]\right\|_1 \leq \sum_{k=1}^{d}\sup_{\mathbf{z}\in\mathcal{X}_\Delta}\left|\frac{\partial^3 \mathsf{k}(\mathbf{z})}{\partial z_i \partial z_j \partial z_k}\right| < \infty,$$

which shows the finiteness of $\bar{D}$. This finishes showing that the stated quantities are finite.

Splitting the bound. To start proving our main inequality, first note that as per Example 5.28, k is L-Lipschitz (see Def. 5.27). Hence, Lemma 5.35 yields that

$$\begin{aligned}\sup_{\substack{\mathbf{x},\mathbf{y}\in\mathsf{Seq}(\mathcal{X})\\ \|\mathbf{x}\|_{1\text{-var}},\|\mathbf{y}\|_{1\text{-var}}\leq V}} & \left|\tilde{\mathsf{k}}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y}) - \tilde{\mathsf{k}}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y})\right| \\ &\leq V^{2m}\sum_{k=1}^{m}\frac{L^{2(m-k)}}{\tilde{d}^{k-1}((k-1)!)^2}\sup_{\mathbf{s},\mathbf{t}\in\mathcal{X}}\left\|\partial^2_{\mathbf{s},\mathbf{t}}\tilde{\mathsf{k}}_k(\mathbf{s},\mathbf{t}) - \partial^2_{\mathbf{s},\mathbf{t}}\mathsf{k}(\mathbf{s},\mathbf{t})\right\|_2\prod_{p=1}^{k-1}\left\|\mathbf{W}^{(p)}\right\|_2^2 \quad (5.43)\end{aligned}$$

We bound the summand in the previous line in probability for each $k\in[m]$. For brevity, denote $\alpha_{m,k} = \frac{V^{2m}L^{2(m-k)}}{((k-1)!)^2}$, and first consider the case $k\geq 2$, so that we have

$$\begin{aligned}P_k(\epsilon) &= \mathbb{P}\left[\frac{\alpha_{m,k}}{\tilde{d}^{k-1}}\left\|\mathbf{W}^{(1)}\right\|_2^2\cdots\left\|\mathbf{W}^{(k-1)}\right\|_2^2\sup_{\mathbf{s},\mathbf{t}\in\mathcal{X}}\left\|\partial_{\mathbf{s},\mathbf{t}}\tilde{\mathsf{k}}_k(\mathbf{s},\mathbf{t}) - \partial_{\mathbf{s},\mathbf{t}}\mathsf{k}(\mathbf{s},\mathbf{t})\right\|_2 \geq \epsilon\right] \\ &\overset{(a)}{\leq} \mathbb{P}\left[\alpha_{m,k}\left(\frac{\left\|\mathbf{W}^{(1)}\right\|_2^2 + \ldots + \left\|\mathbf{W}^{(k-1)}\right\|_2^2}{\tilde{d}(k-1)}\right)^{k-1}\sup_{\mathbf{s},\mathbf{t}\in\mathcal{X}}\left\|\partial_{\mathbf{s},\mathbf{t}}\tilde{\mathsf{k}}_k(\mathbf{s},\mathbf{t}) - \partial_{\mathbf{s},\mathbf{t}}\mathsf{k}(\mathbf{s},\mathbf{t})\right\|_2 \geq \epsilon\right] \\ &\overset{(b)}{=} \mathbb{P}\left[\underbrace{\frac{\left\|\mathbf{W}^{(1)}\right\|_2^2 + \ldots + \left\|\mathbf{W}^{(k-1)}\right\|_2^2}{\tilde{d}(k-1)}}_{(A_k)}\underbrace{\sup_{\mathbf{s},\mathbf{t}\in\mathcal{X}}\left\|\partial_{\mathbf{s},\mathbf{t}}\tilde{\mathsf{k}}_k(\mathbf{s},\mathbf{t}) - \partial_{\mathbf{s},\mathbf{t}}\mathsf{k}(\mathbf{s},\mathbf{t})\right\|_2^{\frac{1}{k-1}}}_{(B_k)} \geq \left(\frac{\epsilon}{\alpha_{m,k}}\right)^{\frac{1}{k-1}}\right],\end{aligned}$$

where in (a) we used the arithmetic-geometric mean inequality, and in (b) we divided both sides by $\alpha_{m,k}$ and took the $(k-1)$th root. Further, setting $t = \left(\frac{\epsilon}{\alpha_{m,k}}\right)^{\frac{1}{k-1}}$, we have for $\gamma > 0$

$$\begin{aligned}P_k(\epsilon) \leq \quad & \mathbb{P}\left[A_k\cdot B_k \geq t\right] \overset{(c)}{\leq} \mathbb{P}\left[(A_k-\gamma)B_k \geq \frac{t}{2}\right] + \mathbb{P}\left[B_k \geq \frac{t}{2\gamma}\right] \\ \overset{(d)}{\leq} \quad & \inf_{\tau>0}\left\{\mathbb{P}\left[A_k - \gamma \geq \frac{\tau}{2}\right] + \mathbb{P}\left[B_k \geq \frac{t}{\tau}\right]\right\} + \mathbb{P}\left[B_k \geq \frac{t}{2\gamma}\right], \quad (5.44)\end{aligned}$$

where in (c) we added and subtracted γB_k and applied a union bound, while in (d) we combined a union bound with the relation $\{XY \geq \epsilon\} \subseteq \{X \geq \tau\}\bigcup\{Y \geq \epsilon/\tau\}$ which holds for any $\tau > 0$.

Our aim is now to obtain good probabilistic bounds on A_k and B_k to use in (5.44) with the

specific choice of $\gamma = \mathbb{E}[A_k] = \sigma_\Lambda^2$.

Bounding A_k. By the inequality between the spectral and Frobenius norms, we have

$$\sum_{p=1}^{k-1} \left\| \mathbf{W}^{(p)} \right\|_2^2 \leq \sum_{p=1}^{k-1} \left\| \mathbf{W}^{(p)} \right\|_F^2 = \sum_{i=1}^{d} \sum_{p=1}^{k-1} \sum_{j=1}^{\tilde{d}} \left(w_{i,j}^{(p)} \right)^2 .$$

Now note that $w_{i,j}^{(p)}$ are i.i.d. copies of the i^{th} marginal of Λ for all $j \in \left[\tilde{d}\right]$ and $p \in [k-1]$, so that via (5.41) $\left(w_{i,j}^{(p)} \right)^2$ satisfies the Bernstein moment condition

$$\mathbb{E}\left[\left(w_{i,j}^{(p)} \right)^{2k} \right] \leq \frac{k! S^2 R^{k-2}}{2} \quad \text{for all } i \in [d], j \in [\tilde{d}], p \in [m], k \geq 2.$$

Hence, we may apply the Bernstein inequality from Theorem 5.15 so that for $i \in [d]$

$$\mathbb{P}\left[\frac{1}{\tilde{d}(k-1)} \sum_{p=1}^{k-1} \sum_{j=1}^{\tilde{d}} \left(w_{i,j}^{(p)} \right)^2 - \sigma_i^2 \geq \epsilon \right] \leq \exp\left(\frac{-\tilde{d}(k-1)\epsilon^2}{2(S^2 + R\epsilon)} \right),$$

where $\sigma_i^2 = \mathbb{E}_{\mathbf{w}\sim\Lambda}\left[w_i^2 \right]$. Combining these bounds for all $i \in [d]$ and denoting $\sigma_\Lambda^2 = \sum_{i=1}^d \sigma_i^2$,

$$\begin{aligned}
\mathbb{P}\left[A_k - \sigma_\Lambda^2 \geq \epsilon \right] &= \mathbb{P}\left[\frac{1}{\tilde{d}(k-1)} \sum_{i=1}^{d} \sum_{p=1}^{k-1} \sum_{j=1}^{\tilde{d}} \left(w_{i,j}^{(p)} \right)^2 - \sigma_\Lambda^2 \geq \epsilon \right] \\
&\leq \sum_{i=1}^{d} \mathbb{P}\left[\frac{1}{\tilde{d}(k-1)} \sum_{p=1}^{k-1} \sum_{j=1}^{\tilde{d}} \left(w_{i,j}^{(p)} \right)^2 - \sigma_i^2 \geq \frac{\epsilon}{d} \right] \\
&\leq d \exp\left(\frac{-\tilde{d}(k-1)\left(\frac{\epsilon}{d}\right)^2}{2\left(S^2 + R\frac{\epsilon}{d}\right)} \right).
\end{aligned} \tag{5.45}$$

Hence, we have the required probabilistic bound for the term in (5.44) containing A_k.

Bounding B_k. One can bound the spectral norm by the max norm so that

$$\begin{aligned}
B_k^{k-1} = \sup_{\mathbf{s},\mathbf{t}\in\mathcal{X}} \left\| \partial_{\mathbf{s},\mathbf{t}} \tilde{\mathsf{k}}_k(\mathbf{s},\mathbf{t}) - \partial_{\mathbf{s},\mathbf{t}} \mathsf{k}(\mathbf{s},\mathbf{t}) \right\|_2 &\leq \sup_{\mathbf{s},\mathbf{t}\in\mathcal{X}} \left\| \partial_{\mathbf{s},\mathbf{t}} \tilde{\mathsf{k}}_k(\mathbf{s},\mathbf{t}) - \partial_{\mathbf{s},\mathbf{t}} \mathsf{k}(\mathbf{s},\mathbf{t}) \right\|_{\max} \\
&= \max_{i,j=1,\ldots,d} \sup_{\mathbf{s},\mathbf{t}\in\mathcal{X}} \left| \frac{\partial^2 \tilde{\mathsf{k}}_k(\mathbf{s},\mathbf{t})}{\partial s_i \partial t_j} - \frac{\partial^2 \mathsf{k}(\mathbf{s},\mathbf{t})}{\partial s_i \partial t_j} \right|.
\end{aligned} \tag{5.46}$$

Let $i, j \in [d]$ and denote $E_{i,j} = \mathbb{E}_{\mathbf{w}\sim\Lambda}\left[\left| w_i w_j \right| \|\mathbf{w}\|_2 \right]$ and $D_{i,j} = \sup_{\mathbf{z}\in\mathcal{X}_\Delta} \|\nabla\left[\partial^{\mathbf{e}_i,\mathbf{e}_j} \mathsf{k}(\mathbf{z})\right]\|_2$, which

are finite as previously shown. Due to Hölder's inequality (Lemma 5.12) and (5.41),

$$\mathbb{E}\left[\left|w_i w_j\right|^m\right] \leq \mathbb{E}^{1/2}\left[w_i^{2m}\right]\mathbb{E}^{1/2}\left[w_j^{2m}\right] < \infty.$$

Recall that k is 3-times continuously differentiable, so that the conditions required by Theorem 5.26 are satisfied, that we now call to our aid in controlling the RFF kernel derivatives,

$$\mathbb{P}\left[\sup_{\mathbf{s},\mathbf{t}\in\mathcal{X}}\left|\frac{\partial^2\tilde{\mathsf{k}}_k(\mathbf{s},\mathbf{t})}{\partial s_i \partial t_j} - \frac{\partial^2\mathsf{k}(\mathbf{s},\mathbf{t})}{\partial s_i \partial t_j}\right| \geq \epsilon\right] \leq 16 C'_{d,\mathcal{X},i,j}\epsilon^{-\frac{d}{d+1}}\exp\left(\frac{-\tilde{d}\epsilon^2}{4(d+1)(2S^2+R\epsilon)}\right),$$

where we defined $C'_{d,\mathcal{X},i,j} = \left(|\mathcal{X}|\left(D_{i,j}+E_{i,j}\right)\right)^{\frac{d}{d+1}}$. Hence, noting that the max satisfies the relation $\{\max_i \xi_i \geq \epsilon\} = \bigcup_i \{\xi_i \geq \epsilon\}$ and union bounding (5.46) in probability, we get that

$$\begin{aligned}\mathbb{P}\left[B_k^{k-1} \geq \epsilon\right] &\leq \sum_{i,j=1}^{d}\mathbb{P}\left[\sup_{\mathbf{s},\mathbf{t}\in\mathcal{X}}\left|\frac{\partial^2\tilde{\mathsf{k}}_k(\mathbf{s},\mathbf{t})}{\partial s_i \partial t_j} - \frac{\partial^2\mathsf{k}(\mathbf{s},\mathbf{t})}{\partial s_i \partial t_j}\right| \geq \epsilon\right] \\ &\leq 16 C'_{d,\mathcal{X}}\epsilon^{-\frac{d}{d+1}}\exp\left(\frac{-\tilde{d}\epsilon^2}{4(d+1)(2S^2+R\epsilon)}\right), \end{aligned} \tag{5.47}$$

where we denote $C'_{d,\mathcal{X}} = \sum_{i,j=1}^{d} C'_{d,\mathcal{X},i,j} = |\mathcal{X}|^{\frac{d}{d+1}}\sum_{i,j=1}^{d}\left(D_{i,j}+E_{i,j}\right)^{\frac{d}{d+1}}$.

Putting it together. Now that we have our bounds for A_k and B_k, we put everything together, that is, plug the bounds (5.45) and (5.47) into (5.44), so that we get

$$\begin{aligned}P_k(\epsilon) &\leq \inf_{\tau>0}\left\{d\exp\left(\frac{-\tilde{d}(k-1)\left(\frac{\tau}{2d}\right)^2}{2\left(S^2+R\frac{\tau}{2d}\right)}\right) + 16C'_{d,\mathcal{X}}\left(\frac{\tau}{t}\right)^{\frac{d(k-1)}{d+1}}\exp\left(\frac{-\tilde{d}\left(\frac{t}{\tau}\right)^{2(k-1)}}{4(d+1)\left(2S^2+R\left(\frac{t}{\tau}\right)^{k-1}\right)}\right)\right\} \\ &\quad + 16C'_{d,\mathcal{X}}\left(\frac{2\sigma_\Lambda^2}{t}\right)^{\frac{d(k-1)}{d+1}}\exp\left(\frac{-\tilde{d}\left(\frac{t}{2\sigma_\Lambda^2}\right)^{2(k-1)}}{4(d+1)\left(2S^2+R\left(\frac{t}{2\sigma_\Lambda^2}\right)^{k-1}\right)}\right) \\ &\overset{(e)}{\leq} d\exp\left(\frac{-\tilde{d}(k-1)\left(\frac{t^{\frac{k-1}{k}}}{2d}\right)^2}{2\left(S^2+R\frac{t^{\frac{k-1}{k}}}{2d}\right)}\right) + 16C'_{d,\mathcal{X}}\left(\frac{1}{t}\right)^{\frac{d(k-1)}{(d+1)k}}\exp\left(\frac{-\tilde{d}t^{\frac{2(k-1)}{k}}}{4(d+1)\left(2S^2+Rt^{\frac{k-1}{k}}\right)}\right) \\ &\quad + 16C'_{d,\mathcal{X}}\left(\frac{2\sigma_\Lambda^2}{t}\right)^{\frac{d(k-1)}{d+1}}\exp\left(\frac{-\tilde{d}\left(\frac{t}{2\sigma_\Lambda^2}\right)^{2(k-1)}}{4(d+1)\left(2S^2+R\left(\frac{t}{2\sigma_\Lambda^2}\right)^{k-1}\right)}\right) \\ &\overset{(f)}{=} d\exp\left(\frac{-\tilde{d}(k-1)\left(\frac{\left(\epsilon/\alpha_{m,k}\right)^{\frac{1}{k}}}{2d}\right)^2}{2\left(S^2+R\frac{\left(\epsilon/\alpha_{m,k}\right)^{\frac{1}{k}}}{2d}\right)}\right) + 16C'_{d,\mathcal{X}}\left(\frac{\alpha_{m,k}}{\epsilon}\right)^{\frac{d}{(d+1)k}}\exp\left(\frac{-\tilde{d}\left(\frac{\epsilon}{\alpha_{m,k}}\right)^{\frac{2}{k}}}{4(d+1)\left(2S^2+R\left(\frac{\epsilon}{\alpha_{m,k}}\right)^{\frac{1}{k}}\right)}\right)\end{aligned}$$

$$+ 16C'_{d,\mathcal{X}} \left(\frac{\alpha_{m,k} \left(2\sigma_\Lambda^2\right)^{k-1}}{\epsilon} \right)^{\frac{d}{d+1}} \exp\left(\frac{-\tilde{d}\left(\frac{\epsilon}{\alpha_{m,k}\left(2\sigma_\Lambda^2\right)^{k-1}} \right)^2}{4(d+1)\left(2S^2 + R\frac{\epsilon}{\alpha_{m,k}\left(2\sigma_\Lambda^2\right)^{k-1}}\right)} \right),$$

where (a) follows from substituting (5.47) and (5.45) into (5.44) with the choice of $\gamma = \sigma_\Lambda^2$, (b) from choosing $\tau = t^{\frac{k-1}{k}}$, and (c) from putting back $t = (\epsilon/\alpha_{m,k})^{\frac{1}{k-1}}$.

Note that the previous applies for all $k \geq 2$. For $k = 1$, we have by (5.47) that

$$\begin{aligned} P_1(\epsilon) &= \mathbb{P}\left[\sup_{\mathbf{s},\mathbf{t}\in\mathcal{X}} \left\| \partial_{\mathbf{s},\mathbf{t}} \tilde{\mathsf{k}}_1(\mathbf{s},\mathbf{t}) - \partial_{\mathbf{s},\mathbf{t}} \mathsf{k}(\mathbf{s},\mathbf{t}) \right\|_2 \geq \frac{\epsilon}{\alpha_{m,1}} \right] \\ &\leq 16C'_{d,\mathcal{X}} \left(\frac{\alpha_{m,1}}{\epsilon} \right)^{\frac{d}{d+1}} \exp\left(\frac{-\tilde{d}\left(\frac{\epsilon}{\alpha_{m,1}} \right)^2}{4(d+1)\left(2S^2 + R\frac{\epsilon}{\alpha_{m,1}}\right)} \right). \end{aligned}$$

Combining and simplifying. We can now combine the bounds for $P_1, \ldots, P_m$ into (5.43),

$$\begin{aligned} &\mathbb{P}\left[\sup_{\substack{\mathbf{x},\mathbf{y}\in\mathsf{Seq}(\mathcal{X}) \\ \|\mathbf{x}\|_{1\text{-var}},\|\mathbf{y}\|_{1\text{-var}}\leq V}} \left| \tilde{\mathsf{k}}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y}) - \mathsf{k}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y}) \right| \geq \epsilon \right] \leq \sum_{k=1}^{m} P_k\left(\frac{\epsilon}{m}\right) \\ &\overset{(g)}{\leq} 2^{\frac{1}{d+1}} 8C'_{d,\mathcal{X}} \sum_{k=1}^{m} \left(\frac{2^k \sigma_\Lambda^{2\,k-1} m\alpha_{m,k}}{\epsilon} \right)^{\frac{d}{d+1}} \exp\left(-\frac{\tilde{d}}{2(d+1)} \cdot \frac{\left(\frac{\epsilon}{2^k \sigma_\Lambda^{2(k-1)} m\alpha_{m,k}} \right)^2}{S^2 + R\frac{\epsilon}{2^k \sigma_\Lambda^{2(k-1)} m\alpha_{m,k}}} \right) \\ &\quad + 2^{\frac{1}{d+1}} 8C'_{d,\mathcal{X}} \sum_{k=2}^{m} \left(\frac{2^k m\alpha_{m,k}}{\epsilon} \right)^{\frac{d}{(d+1)k}} \exp\left(-\frac{\tilde{d}}{2(d+1)} \cdot \frac{\left(\frac{\epsilon}{2^k m\alpha_{m,k}} \right)^{\frac{2}{k}}}{S^2 + R\left(\frac{\epsilon}{2^k m\alpha_{m,k}} \right)^{\frac{1}{k}}} \right) \\ &\quad + d \sum_{k=2}^{M} \exp\left(-\frac{\tilde{d}(k-1)}{2} \frac{\left(\frac{\epsilon}{2^k d^k m\alpha_{m,k}} \right)^{\frac{2}{k}}}{S^2 + R\left(\frac{\epsilon}{2^k d^k m\alpha_{m,k}} \right)^{\frac{1}{k}}} \right) \\ &\overset{(h)}{\leq} 2^{\frac{1}{d+1}} 8C'_{d,\mathcal{X}} \sum_{k=1}^{m} \left(\frac{2^k \left(\sigma_\Lambda^2 \vee d\right)^k m\alpha_{m,k}}{\epsilon} \right)^{\frac{d}{d+1}} \exp\left(-\frac{\tilde{d}}{2(d+1)} \cdot \frac{\left(\frac{\epsilon}{2^k \left(\sigma_\Lambda^2 \vee d\right)^k m\alpha_{m,k}} \right)^2}{S^2 + R\frac{\epsilon}{2^k \left(\sigma_\Lambda^2 \vee d\right)^k m\alpha_{m,k}}} \right) \\ &\quad + 2^{\frac{1}{d+1}} 8C'_{d,\mathcal{X}} \sum_{k=1}^{m} \left(\frac{2^k \left(\sigma_\Lambda^2 \vee d\right)^k m\alpha_{m,k}}{\epsilon} \right)^{\frac{d}{(d+1)k}} \exp\left(-\frac{\tilde{d}}{2(d+1)} \cdot \frac{\left(\frac{\epsilon}{2^k \left(\sigma_\Lambda^2 \vee d\right)^k m\alpha_{m,k}} \right)^{\frac{2}{k}}}{S^2 + R\left(\frac{\epsilon}{2^k \left(\sigma_\Lambda^2 \vee d\right)^k m\alpha_{m,k}} \right)^{\frac{1}{k}}} \right) \\ &\quad + d \sum_{k=1}^{m} \exp\left(-\frac{\tilde{d}}{2(d+1)} \frac{\left(\frac{\epsilon}{2^k \left(\sigma_\Lambda^2 \vee d\right)^k m\alpha_{m,k}} \right)^{\frac{2}{k}}}{S^2 + R\left(\frac{\epsilon}{2^k \left(\sigma_\Lambda^2 \vee d\right)^k m\alpha_{m,k}} \right)^{\frac{1}{k}}} \right) \end{aligned}$$

$$
\begin{aligned}
&\overset{(f)}{\leq} 2^{\frac{1}{d+1}} 8 C'_{d,\mathcal{X}} \sum_{k=1}^{m} \left(\frac{\beta_{d,m,V}}{\epsilon} \right)^{\frac{d}{d+1}} \exp\left(-\frac{\tilde{d}}{2(d+1)} \frac{\left(\frac{\epsilon}{\beta_{d,m,V}}\right)^2}{S^2 + R\frac{\epsilon}{\beta d,m,V}} \right) \\
&+ 2^{\frac{1}{d+1}} 8 C'_{d,\mathcal{X}} \sum_{k=1}^{m} \left(\frac{\beta_{d,m,V}}{\epsilon} \right)^{\frac{d}{(d+1)k}} \exp\left(-\frac{\tilde{d}}{2(d+1)} \frac{\left(\frac{\epsilon}{\beta_{d,m,V}}\right)^{\frac{2}{k}}}{S^2 + R\left(\frac{\epsilon}{\beta_{d,m,V}}\right)^{\frac{1}{k}}} \right) \\
&+ d \sum_{k=1}^{m} \exp\left(-\frac{\tilde{d}}{2(d+1)} \frac{\left(\frac{\epsilon}{\beta_{d,m,V}}\right)^{\frac{2}{k}}}{S^2 + R\left(\frac{\epsilon}{\beta_{d,m,V}}\right)^{\frac{1}{k}}} \right),
\end{aligned}
$$

where (g) follows from rearranging the expressions from (f), while (h) from unifying the coefficients and that $1 \leq d \leq \max(\sigma_\Lambda^2, d)$ and $f(x) = x^2/(a+bx)$ is monotonically increasing in x on the positive half-line for $a, b > 0$, while (f) from $\alpha_{m,k} = V^{2m} L^{2(m-k)}/((k-1)!)^2 \leq (VL)^{2m}$, using that $f(x)$ is increasing, and defining $\beta_{d,m,V} = m\left(2V^2(L^2 \vee 1)(\sigma_\Lambda^2 \vee d)\right)^m$.

Conclusion. Finally, we split the bound into two cases: the first case is if the error is big, i.e. $\epsilon \geq \beta_{d,m,V} = m\left(2V^2(L^2 \vee 1)(\sigma_\Lambda^2 \vee d)\right)^m$, when we decrease all the exponents to $\frac{1}{m}$,

$$
\begin{aligned}
&\mathbb{P}\left[\sup_{\substack{\mathbf{x},\mathbf{y} \in \mathsf{Seq}(\mathcal{X}) \\ \|\mathbf{x}\|_{1\text{-var}}, \|\mathbf{y}\|_{1\text{-var}} \leq V}} \left| \tilde{\mathsf{k}}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y}) - \mathsf{k}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y}) \right| \geq \epsilon \right] \\
&\leq m \cdot \left(2^{\frac{1}{d+1}} 16 C'_{d,\mathcal{X}} \left(\frac{\beta_{d,m,V}}{\epsilon} \right)^{\frac{d}{(d+1)m}} + d \right) \exp\left(-\frac{\tilde{d}}{2(d+1)} \frac{\left(\frac{\epsilon}{\beta_{d,m,V}}\right)^{\frac{2}{m}}}{S^2 + R\left(\frac{\epsilon}{\beta_{d,m,V}}\right)^{\frac{1}{m}}} \right)
\end{aligned}
$$

The other when the error is small, i.e. $\epsilon < \beta_{d,m,V}$, when we increase all the exponents to 1

$$
\begin{aligned}
&\mathbb{P}\left[\sup_{\substack{\mathbf{x},\mathbf{y} \in \mathsf{Seq}(\mathcal{X}) \\ \|\mathbf{x}\|_{1\text{-var}}, \|\mathbf{y}\|_{1\text{-var}} \leq V}} \left| \tilde{\mathsf{k}}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y}) - \mathsf{k}_{\mathsf{S}}(\mathbf{x},\mathbf{y}) \right| \geq \epsilon \right] \\
&\leq m \cdot \left(2^{\frac{1}{d+1}} 16 C'_{d,\mathcal{X}} \left(\frac{\beta_{d,m,V}}{\epsilon} \right)^{\frac{d}{d+1}} + d \right) \exp\left(-\frac{\tilde{d}}{2(d+1)} \frac{\left(\frac{\epsilon}{\beta_{d,m,V}}\right)^2}{S^2 + R\left(\frac{\epsilon}{\beta_{d,m,V}}\right)} \right).
\end{aligned}
$$

The claimed estimate follows by denoting $C_{d,\mathcal{X}} = 2^{\frac{1}{d+1}} 16 C'_{d,\mathcal{X}}$ and simplifying. □

Next, we prove Theorem 5.6 to show an approximation bound for the `RFSF-DP` kernel.

Theorem 5.37 (Concentration inequality for `RFSF-DP` kernel)**.** *Let* $\mathsf{k} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ *be a continuous, bounded, translation-invariant kernel with spectral measure* Λ*, which satisfies for some*

$S, R > 0$ *that*

$$\mathbb{E}_{\mathbf{w}\sim\Lambda}\left[|w_i|^{2k}\right] \leq \frac{k! S^2 R^{k-2}}{2} \quad \textit{for all } i \in [d] \textit{ and } k \geq 2. \tag{5.48}$$

Then, for signature level $m \in \mathbb{Z}_+$ *and* $\mathbf{x}, \mathbf{y} \in \mathsf{Seq}(\mathcal{X})$*, it holds for* $\epsilon > 0$ *that:*

$$\mathbb{P}\left[\left|\tilde{\mathsf{k}}^{\mathsf{DP}}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y}) - \mathsf{k}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y})\right| \geq \epsilon\right] \leq 2\exp\left(-\frac{1}{4}\min\left\{\begin{array}{c}\left(\frac{\sqrt{\tilde{d}}\epsilon}{2C_{d,m}\|\mathbf{x}\|^m_{1\text{-var}}\|\mathbf{y}\|^m_{1\text{-var}}}\right)^2, \\ \left(\frac{\tilde{d}\epsilon}{\sqrt{8}C_{d,m}\|\mathbf{x}\|^m_{1\text{-var}}\|\mathbf{y}\|^m_{1\text{-var}}}\right)^{\frac{1}{m}}\end{array}\right\}\right),$$

where the absolute constant $C_{d,m} > 0$ *satisfies that*

$$C_{d,m} \leq \sqrt{8}e^4(2\pi)^{1/4}e^{1/24}(4e^3/m)^m\left((2d\max(S,R))^m + \left(L^2/\ln 2\right)^m\right).$$

Proof. Let $\hat{\mathsf{k}}^{(1)}_{\mathsf{S}_m}, \ldots, \hat{\mathsf{k}}^{(\tilde{d})}_{\mathsf{S}_m}$ be independent copies of the RFSF kernel, each with internal RFF sample size $\hat{d} = 1$, such that $\tilde{\mathsf{k}}^{\mathsf{DP}}_{\mathsf{S}_m} = \frac{1}{\tilde{d}}\sum_{k=1}^{\tilde{d}} \hat{\mathsf{k}}^{(k)}_{\mathsf{S}_m}$. Our goal is to call Theorem 5.23 with $\alpha = \frac{1}{m}$, and therefore, we compute an upper bound on the $\Psi_{1/m}$-norm of $\tilde{\mathsf{k}}^{(k)}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y}) - \mathsf{k}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y})$ for all $k \in [\tilde{d}]$; for a definition of the α-exponential norm, see Definition 5.17.

By Lemma 5.32, it holds for any $\mathbf{x}, \mathbf{y} \in \mathsf{Seq}(\mathcal{X})$ and $k \in [m]$ that

$$\left|\hat{\mathsf{k}}^{(k)}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y})\right| \leq \frac{\left(\|\mathbf{x}\|_{1\text{-var}}\|\mathbf{y}\|_{1\text{-var}}\right)^m}{(m!)^2}\left\|\mathbf{w}^{(1)}_k\right\|_2^2 \cdots \left\|\mathbf{w}^{(m)}_k\right\|_2^2,$$

where $\mathbf{w}^{(1)}_k, \ldots, \mathbf{w}^{(m)}_k \overset{\text{i.i.d.}}{\sim} \Lambda$ are the random weights that parametrize $\hat{\mathsf{k}}^{(k)}_{\mathsf{S}_m}$ for all $k \in [\tilde{d}]$. Now, calling Lemma 5.21 with $\alpha = 1$ yields that, due to (5.48), the following holds:

$$\left\|w^{(p)2}_{k,i}\right\|_{\Psi_1} \leq 2\max(S,R) \quad \text{for all } i \in [d], k \in [\tilde{d}], p \in [m].$$

Note that for $\alpha = 1$, $\|\cdot\|_{\Psi_\alpha}$ satisfies the triangle inequality (see Lemma 5.19), and hence

$$\left\|\left\|\mathbf{w}^{(p)}_k\right\|_2^2\right\|_{\Psi_1} \leq \sum_{i=1}^{d}\left\|w^{(p)2}_{k,i}\right\|_{\Psi_1} \leq 2d\max(S,R) \quad \text{for all } k \in [\tilde{d}], p \in [m].$$

As $\|\cdot\|_{\Psi_\alpha}$ is positive homogenous and satisfies a Hölder-type inequality (see Lemma 5.20):

$$\left\|\hat{\mathsf{k}}^{(k)}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y})\right\|_{\Psi_{1/m}} \overset{\text{(a)}}{\leq} \frac{\left(\|\mathbf{x}\|_{1\text{-var}}\|\mathbf{y}\|_{1\text{-var}}\right)^m}{(m!)^2}\left\|\left\|\mathbf{w}^{(1)}_k\right\|_2^2\right\|_{\Psi_1} \cdots \left\|\left\|\mathbf{w}^{(m)}_k\right\|_2^2\right\|_{\Psi_1}$$

$$\overset{\text{(b)}}{\leq} \frac{\left(2d \left\|\mathbf{x}\right\|_{1\text{-var}} \left\|\mathbf{y}\right\|_{1\text{-var}} \max(S,R)\right)^m}{(m!)^2} \quad \text{for all } k \in [\tilde{d}], \tag{5.49}$$

where in (a) we used Corollary 5.33, in (b) we used (5.49). We are almost ready to use Theorem 5.23, but it requires the $\|\cdot\|_{\Psi_{1/m}}$ bound in terms of centered random variables. Although $\|\cdot\|_{\Psi_\alpha}$ does not satisfy the triangle inequality for $\alpha \in (0,1)$, it obeys that (see [100, Lemma A.3.]) $\|X+Y\|_{\Psi_\alpha} \leq 2^{1/\alpha}\left(\|X\|_{\Psi_\alpha} + \|Y\|_{\Psi_\alpha}\right)$ for any random variables X and Y. For a constant $c \in \mathbb{R}$, we have $\|c\|_{\Psi_{1/m}} = \frac{|c|}{\ln^m 2}$, and hence by Lemma 5.31 we have that

$$\left\|\mathsf{k}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y})\right\|_{\Psi_{1/m}} \leq \frac{\left(L^2 \left\|\mathbf{x}\right\|_{1\text{-var}} \left\|\mathbf{y}\right\|_{1\text{-var}} / \ln 2\right)^m}{(m!)^2},$$

where $L = \left\|\mathbb{E}_{\mathbf{w}\sim\Lambda}\left[\mathbf{w}\mathbf{w}^\top\right]\right\|_2$ is the Lipschitz constant of the kernel $\mathsf{k} : \mathcal{X}\times\mathcal{X} \to \mathbb{R}$. This gives

$$\begin{aligned}\left\|\hat{\mathsf{k}}^{(k)}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y}) - \mathsf{k}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y})\right\|_{\Psi_{1/m}} &\leq 2^m\left(\left\|\hat{\mathsf{k}}^{(k)}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y})\right\|_{\Psi_{1/m}} + \left\|\mathsf{k}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y})\right\|_{\Psi_{1/m}}\right) \\ &\leq \frac{\left(2\left\|\mathbf{x}\right\|_{1\text{-var}}\left\|\mathbf{y}\right\|_{1\text{-var}}\right)^m}{(m!)^2}\left((2d\max(S,R))^m + \left(L^2/\ln 2\right)^m\right).\end{aligned}$$

Finally, we have the required Orlicz norm bound for invoking Theorem 5.23, so that we get

$$\begin{aligned}&\mathbb{P}\left[\left|\tilde{\mathsf{k}}^{\mathsf{DP}}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y}) - \mathsf{k}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y})\right| \geq \epsilon\right] \\ &\quad\leq 2\exp\left(-\frac{1}{4}\min\left\{\left(\frac{\sqrt{\tilde{d}}\epsilon}{2C_{d,m}}\right)^2, \left(\frac{\tilde{d}\epsilon}{\sqrt{8}C_{d,m}}\right)^{\frac{1}{m}}\right\}\right),\end{aligned}$$

where the constant $C_{d,m} > 0$ is defined as

$$C_{d,m} = \sqrt{8}e^4(2\pi)^{1/4}e^{1/24}\frac{(4em\left\|\mathbf{x}\right\|_{1\text{-var}}\left\|\mathbf{y}\right\|_{1\text{-var}})^m}{(m!)^2}\left((2d\max(S,R))^m + \left(L^2/\ln 2\right)^m\right),$$

and invoking Stirling's approximation $\frac{1}{m!} \leq \left(\frac{e}{m}\right)^m$ gives the stated result. □

Now, we prove the analogous result for $\tilde{\mathsf{k}}^{\mathsf{TRP}}_{\mathsf{S}_m}$.

Theorem 5.38 (Concentration inequality for RFSF-TRP kernel)**.** *Let $\mathsf{k} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be a continuous, bounded, translation-invariant kernel with spectral measure Λ, which satisfies for some $S, R > 0$ that*

$$\mathbb{E}_{\mathbf{w}\sim\Lambda}\left[|w_i|^{2k}\right] \leq \frac{k!S^2R^{k-2}}{2} \quad \textit{for all } i \in [d] \textit{ and } k \geq 2. \tag{5.50}$$

Then, for the level-m RFSF–TRP *kernel as defined in* (5.16)*, we have for* $\mathbf{x}, \mathbf{y} \in \mathsf{Seq}(\mathcal{X})$ *and* $\epsilon > 0$

$$\mathbb{P}\left[\left|\tilde{\mathsf{k}}^{\mathsf{TRP}}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y}) - \tilde{\mathsf{k}}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y})\right| \geq \epsilon\right] \leq C_{d,\Lambda} \exp\left(-\left(\frac{m^2 \tilde{d}_{\mathsf{TRP}}^{\frac{1}{2m}} \epsilon^{\frac{1}{m}}}{2\sqrt{2}e^3 R \,\|\mathbf{x}\|_{1\text{-var}} \,\|\mathbf{y}\|_{1\text{-var}}}\right)^{\frac{1}{2}}\right),$$

where $C_{d,\Lambda} = 2\left(1 + \frac{S}{2R} + \frac{S^2}{4R^2}\right)^d$.

Proof. First, we consider the conditional probability $\mathbb{P}\left[\left|\tilde{\mathsf{k}}^{\mathsf{TRP}}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y}) - \tilde{\mathsf{k}}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y})\right| \geq \epsilon \middle| \mathbf{W}\right]$ by conditioning on the RFSF weights $\mathbf{W} = (\mathbf{W}^{(1)}, \dots, \mathbf{W}^{(m)})$, so that the only source of randomness comes from the TRP weights $\mathbf{P} = (\mathbf{P}^{(1)}, \dots, \mathbf{P}^{(m)})$. The idea is to call Theorem 5.25 to estimate the conditional probability, and then take expectation over $\mathbf{W}$. Since Theorem 5.25 quantifies the concentration of a Gaussian polynomial around its mean in terms of its variance, we first compute the conditional statistics of $\tilde{\mathsf{k}}^{\mathsf{TRP}}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y})$.

Conditional expectation. Recall the definition of $\tilde{\varphi}^{\mathsf{TRP}}_{\mathsf{S}_m}(\mathbf{x})$ (5.15), where $\mathbf{p}_1^{(1)}, \dots, \mathbf{p}_{\tilde{d}_{\mathsf{TRP}}}^{(m)} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \mathbf{I}_{2\tilde{d}})$, and $\mathbf{x} \in \mathsf{Seq}(\mathcal{X})$, so that

$$\tilde{\varphi}^{\mathsf{TRP}}_{\mathsf{S}_m}(\mathbf{x}) = \frac{1}{\sqrt{\tilde{d}_{\mathsf{TRP}}}} \left(\sum_{\mathbf{i} \in \Delta_m(\mathsf{L}_\mathbf{x})} \prod_{p=1}^{m} \left\langle \mathbf{p}_i^{(p)}, \delta\tilde{\varphi}_p(\mathbf{x}_{i_p}) \right\rangle \right)_{i=1}^{\tilde{d}_{\mathsf{TRP}}}.$$

Hence, $\tilde{\mathsf{k}}^{\mathsf{TRP}}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y})$ is written for $\mathbf{x}, \mathbf{y} \in \mathsf{Seq}(\mathcal{X})$ as

$$\tilde{\mathsf{k}}^{\mathsf{TRP}}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y}) = \frac{1}{\tilde{d}_{\mathsf{TRP}}} \sum_{i=1}^{\tilde{d}_{\mathsf{TRP}}} \underbrace{\sum_{\substack{\mathbf{i} \in \Delta_m(\mathsf{L}_\mathbf{x}) \\ \mathbf{j} \in \Delta_m(\mathsf{L}_\mathbf{y})}} \prod_{p=1}^{m} \left\langle \mathbf{p}_i^{(p)}, \delta\tilde{\varphi}_p(\mathbf{x}_{i_p}) \right\rangle \left\langle \mathbf{p}_i^{(p)}, \delta\tilde{\varphi}_p(\mathbf{y}_{i_p}) \right\rangle}_{A_i},$$

which is a sample average of $\tilde{d}_{\mathsf{TRP}}$ i.i.d. terms, i.e. $\tilde{\mathsf{k}}^{\mathsf{TRP}}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y}) = \frac{1}{\tilde{d}_{\mathsf{TRP}}} \sum_{i=1}^{\tilde{d}_{\mathsf{TRP}}} A_i$. We only have to verify that A_i is conditionally an unbiased approximator of $\tilde{\mathsf{k}}_{\mathsf{S}_m}$ given $\mathbf{W}$.

$$\begin{aligned}
\mathbb{E}[A_i \,|\, \mathbf{W}] &\overset{\text{(a)}}{=} \sum_{\substack{\mathbf{i} \in \Delta_m(\mathsf{L}_\mathbf{x}) \\ \mathbf{j} \in \Delta_m(\mathsf{L}_\mathbf{y})}} \prod_{p=1}^{m} \mathbb{E}\left[\left\langle \mathbf{p}_i^{(p)}, \delta\tilde{\varphi}_p(\mathbf{x}_{i_p}) \right\rangle \left\langle \mathbf{p}_i^{(p)}, \delta\tilde{\varphi}_p(\mathbf{y}_{j_p}) \right\rangle \,|\, \mathbf{W} \right] \\
&\overset{\text{(b)}}{=} \sum_{\substack{\mathbf{i} \in \Delta_m(\mathsf{L}_\mathbf{x}) \\ \mathbf{j} \in \Delta_m(\mathsf{L}_\mathbf{y})}} \prod_{p=1}^{m} \left\langle \mathbb{E}\left[\mathbf{p}_i^{(p)} \otimes \mathbf{p}_i^{(p)} \right], \delta\tilde{\varphi}_p(\mathbf{x}_{i_p}) \otimes \delta\tilde{\varphi}_p(\mathbf{y}_{j_p}) \right\rangle
\end{aligned}$$

$$\overset{(c)}{=} \sum_{\substack{\mathbf{i}\in\Delta_m(\mathsf{L}_\mathbf{x})\\ \mathbf{j}\in\Delta_m(\mathsf{L}_\mathbf{y})}} \prod_{p=1}^{m} \left\langle I_{\tilde{d}}, \delta\tilde{\varphi}_p(\mathbf{x}_{i_p}) \otimes \delta\tilde{\varphi}_p(\mathbf{y}_{j_p}) \right\rangle$$

$$\overset{(d)}{=} \sum_{\substack{\mathbf{i}\in\Delta_m(\mathsf{L}_\mathbf{x})\\ \mathbf{j}\in\Delta_m(\mathsf{L}_\mathbf{y})}} \prod_{p=1}^{m} \left\langle \delta\tilde{\varphi}_p(\mathbf{x}_{i_p}), \delta\tilde{\varphi}_p(\mathbf{y}_{j_p}) \right\rangle,$$

where (a) follows from linearity of expectation and independence of the $\mathbf{p}_i^{(p)}$'s for $p \in [m]$, (b) from bilinearity of inner product, and independence of $\mathbf{P}$ and $\mathbf{W}$, (c) from substituting the covariance, (d) is since the outer product is projected onto the diagonal.

Conditional variance. We compute the conditional variance of A_i given $\mathbf{W}$:

$$\begin{aligned}
&\mathbb{E}\left[A_{m,i}^2 \mid \mathbf{W}\right] \\
&\overset{(e)}{=} \sum_{\substack{\mathbf{i},\mathbf{k}\in\Delta_m(\mathsf{L}_\mathbf{x})\\ \mathbf{j},\mathbf{l}\in\Delta_m(\mathsf{L}_\mathbf{y})}} \prod_{p=1}^{m} \mathbb{E}\left[\left\langle \mathbf{p}_i^{(p)}, \delta\tilde{\varphi}_p(\mathbf{x}_{i_p})\right\rangle \left\langle \mathbf{p}_i^{(p)}, \delta\tilde{\varphi}_p(\mathbf{y}_{j_p})\right\rangle \left\langle \mathbf{p}_i^{(p)}, \delta\tilde{\varphi}_p(\mathbf{x}_{k_p})\right\rangle \left\langle \mathbf{p}_i^{(p)}, \delta\tilde{\varphi}_p(\mathbf{y}_{l_p})\right\rangle \mid \mathbf{W}\right] \\
&\overset{(f)}{=} \sum_{\substack{\mathbf{i},\mathbf{k}\in\Delta_m(\mathsf{L}_\mathbf{x})\\ \mathbf{j},\mathbf{l}\in\Delta_m(\mathsf{L}_\mathbf{y})}} \prod_{p=1}^{m} \Bigg(\mathbb{E}\left[\left\langle \mathbf{p}_i^{(p)}, \delta\tilde{\varphi}_p(\mathbf{x}_{i_p})\right\rangle \left\langle \mathbf{p}_i^{(p)}, \delta\tilde{\varphi}_p(\mathbf{y}_{j_p})\right\rangle \mid \mathbf{W}\right] \mathbb{E}\left[\left\langle \mathbf{p}_i^{(p)}, \delta\tilde{\varphi}_p(\mathbf{x}_{k_p})\right\rangle \left\langle \mathbf{p}_i^{(p)}, \delta\tilde{\varphi}_p(\mathbf{y}_{l_p})\right\rangle \mid \mathbf{W}\right] \\
&\qquad + \mathbb{E}\left[\left\langle \mathbf{p}_i^{(p)}, \delta\tilde{\varphi}_p(\mathbf{x}_{i_p})\right\rangle \left\langle \mathbf{p}_i^{(p)}, \delta\tilde{\varphi}_p(\mathbf{x}_{k_p})\right\rangle \mid \mathbf{W}\right] \mathbb{E}\left[\left\langle \mathbf{p}_i^{(p)}, \delta\tilde{\varphi}_p(\mathbf{y}_{j_p})\right\rangle \left\langle \mathbf{p}_i^{(p)}, \delta\tilde{\varphi}_p(\mathbf{y}_{l_p})\right\rangle \mid \mathbf{W}\right] \\
&\qquad + \mathbb{E}\left[\left\langle \mathbf{p}_i^{(p)}, \delta\tilde{\varphi}_p(\mathbf{x}_{i_p})\right\rangle \left\langle \mathbf{p}_i^{(p)}, \delta\tilde{\varphi}_p(\mathbf{y}_{l_p})\right\rangle \mid \mathbf{W}\right] \mathbb{E}\left[\left\langle \mathbf{p}_i^{(p)}, \delta\tilde{\varphi}_p(\mathbf{x}_{k_p})\right\rangle \left\langle \mathbf{p}_i^{(p)}, \delta\tilde{\varphi}_p(\mathbf{y}_{j_p})\right\rangle \mid \mathbf{W}\right] \Bigg) \\
&\overset{(g)}{=} \sum_{\substack{\mathbf{i},\mathbf{k}\in\Delta_m(\mathsf{L}_\mathbf{x})\\ \mathbf{j},\mathbf{l}\in\Delta_m(\mathsf{L}_\mathbf{y})}} \prod_{p=1}^{m} \Big(\left\langle \delta\tilde{\varphi}_p(\mathbf{x}_{i_p}), \delta\tilde{\varphi}_p(\mathbf{y}_{j_p})\right\rangle \left\langle \delta\tilde{\varphi}_p(\mathbf{x}_{k_p}), \delta\tilde{\varphi}_p(\mathbf{y}_{l_p})\right\rangle \\
&\qquad + \left\langle \delta\tilde{\varphi}_p(\mathbf{x}_{i_p}), \delta\tilde{\varphi}_p(\mathbf{x}_{k_p})\right\rangle \left\langle \delta\tilde{\varphi}_p(\mathbf{y}_{j_p}), \delta\tilde{\varphi}_p(\mathbf{y}_{l_p})\right\rangle \\
&\qquad + \left\langle \delta\tilde{\varphi}_p(\mathbf{x}_{i_p}), \delta\tilde{\varphi}_p(\mathbf{y}_{l_p})\right\rangle \left\langle \delta\tilde{\varphi}_p(\mathbf{x}_{k_p}), \delta\tilde{\varphi}_p(\mathbf{y}_{j_p})\right\rangle \Big) \\
&\overset{(h)}{\leq} \sum_{\substack{\mathbf{i},\mathbf{k}\in\Delta_m(\mathsf{L}_\mathbf{x})\\ \mathbf{j},\mathbf{l}\in\Delta_m(\mathsf{L}_\mathbf{y})}} 3^m \prod_{p=1}^{m} \left\|\delta\tilde{\varphi}_p(\mathbf{x}_{i_p})\right\| \left\|\delta\tilde{\varphi}_p(\mathbf{y}_{j_p})\right\| \left\|\delta\tilde{\varphi}_p(\mathbf{x}_{k_p})\right\| \left\|\delta\tilde{\varphi}_p(\mathbf{y}_{l_p})\right\| \\
&\overset{(i)}{=} 3^m \left(\sum_{\mathbf{i}\in\Delta_m(\mathsf{L}_\mathbf{x})} \prod_{p=1}^{m} \left\|\delta\tilde{\varphi}_p(\mathbf{x}_{i_p})\right\| \right)^2 \left(\sum_{\mathbf{j}\in\Delta_m(\mathsf{L}_\mathbf{y})} \prod_{p=1}^{m} \left\|\delta\tilde{\varphi}_p(\mathbf{y}_{j_p})\right\| \right)^2 \\
&\overset{(j)}{\leq} \frac{1}{(m!)^4} \left(\frac{3\left\|\mathbf{x}\right\|_{1\text{-var}}^2 \left\|\mathbf{y}\right\|_{1\text{-var}}^2}{\tilde{d}^2} \right)^m \prod_{p=1}^{m} \left\|\mathbf{W}^{(p)}\right\|_2^4,
\end{aligned}$$

where (e) follows from linearity of expectation and independence of the $\mathbf{p}_i^{(p)}$'s for $p \in [m]$, (f) from Isserlis' theorem [131], (g) is the same as (a)-(d), (h) is the Cauchy-Schwarz inequality,

(i) from factorizing the summation, (j) is the same as Lemma 5.32.

Therefore, we have due to Lemma 5.33 for the variance that

$$\mathbb{V}\left[A_{m,i} \,|\, \mathbf{W}\right] = \mathbb{E}\left[A_{m,i}^2 \,|\, \mathbf{W}\right] - \mathbb{E}^2\left[A_{m,i} \,|\, \mathbf{W}\right] \leq \frac{3^m+1}{(m!)^4}\left(\frac{\|\mathbf{x}\|_{1\text{-var}}^2 \|\mathbf{y}\|_{1\text{-var}}^2}{\tilde{d}^2}\right)^m \prod_{p=1}^{m} \left\|\mathbf{W}^{(p)}\right\|_2^4.$$

Let $\beta_m(\mathbf{x},\mathbf{y}) = \frac{3^m+1}{(m!)^4}\|\mathbf{x}\|_{1\text{-var}}^{2m}\|\mathbf{y}\|_{1\text{-var}}^{2m}$. Then, as $\tilde{\mathsf{k}}_{\mathsf{S}_m}^{\mathsf{TRP}}(\mathbf{x},\mathbf{y}) \,|\, \mathbf{W}$ is a sample average,

$$\mathbb{V}\left[\tilde{\mathsf{k}}_{\mathsf{S}_m}^{\mathsf{TRP}}(\mathbf{x},\mathbf{y}) \,|\, \mathbf{W}\right] \leq \frac{\beta_m(\mathbf{x},\mathbf{y})}{\tilde{d}_{\mathsf{TRP}}\tilde{d}^{2m}} \prod_{p=1}^{m} \left\|\mathbf{W}^{(p)}\right\|_2^4. \tag{5.51}$$

Conditional bound. Since $\tilde{\mathsf{k}}_{\mathsf{S}_m}^{\mathsf{TRP}}(\mathbf{x},\mathbf{y}) \,|\, \mathbf{W}$ is a Gaussian polynomial of degree-$2m$, with expectation $\tilde{\mathsf{k}}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y})$, and variance (5.51), we have by Theorem 5.25 for $\epsilon > 0$ that

$$\begin{aligned}\mathbb{P}\left[\left|\tilde{\mathsf{k}}_{\mathsf{S}_m}^{\mathsf{TRP}}(\mathbf{x},\mathbf{y}) - \tilde{\mathsf{k}}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y})\right| \geq \epsilon \,|\, \mathbf{W}\right] &\leq 2\exp\left(-\frac{\epsilon^{\frac{1}{m}}}{2\sqrt{2}e\mathbb{V}^{\frac{1}{2m}}\left[\tilde{\mathsf{k}}_{\mathsf{S}_m}^{\mathsf{TRP}}(\mathbf{x},\mathbf{y}) \,|\, \mathbf{W}\right]}\right) \\ &\leq 2\exp\left(-\frac{\tilde{d}_{\mathsf{TRP}}^{\frac{1}{2m}}\tilde{d}\epsilon^{\frac{1}{m}}}{2\sqrt{2}e\beta_m^{\frac{1}{2m}}(\mathbf{x},\mathbf{y})\prod_{p=1}^{m}\left\|\mathbf{W}^{(p)}\right\|_2^{\frac{2}{m}}}\right). \end{aligned} \tag{5.52}$$

Undoing the conditioning. We take the expectation in (5.52) so that

$$\begin{aligned}\mathbb{P}\left[\left|\tilde{\mathsf{k}}_{\mathsf{S}_m}^{\mathsf{TRP}}(\mathbf{x},\mathbf{y}) - \tilde{\mathsf{k}}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y})\right| \geq \epsilon\right] &= \mathbb{E}\left[\mathbb{P}\left[\left|\tilde{\mathsf{k}}_{\mathsf{S}_m}^{\mathsf{TRP}}(\mathbf{x},\mathbf{y}) - \tilde{\mathsf{k}}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y})\right| \geq \epsilon \,|\, \mathbf{W}\right]\right] \\ &\leq 2\mathbb{E}\left[\exp\left(-\frac{\tilde{d}_{\mathsf{TRP}}^{\frac{1}{2m}}\tilde{d}\epsilon^{\frac{1}{m}}}{2\sqrt{2}e\beta_m(\mathbf{x},\mathbf{y})^{\frac{1}{2m}}\prod_{p=1}^{m}\left\|\mathbf{W}^{(p)}\right\|_2^{\frac{2}{m}}}\right)\right] \\ &\overset{(k)}{\leq} 2\mathbb{E}\left[\exp\left(-\frac{\lambda\tilde{d}_{\mathsf{TRP}}^{\frac{1}{2m}}\epsilon^{\frac{1}{m}}}{2\sqrt{2}e\beta_m(\mathbf{x},\mathbf{y})^{\frac{1}{2m}}}\frac{\tilde{d}}{\lambda\prod_{p=1}^{m}\left\|\mathbf{W}^{(p)}\right\|_2^{\frac{2}{m}}}\right)\right] \\ &\overset{(l)}{\leq} 2\mathbb{E}\left[\exp\left(-2\left(\frac{\lambda\tilde{d}_{\mathsf{TRP}}^{\frac{1}{2m}}\epsilon^{\frac{1}{m}}}{2\sqrt{2}e\beta_m(\mathbf{x},\mathbf{y})^{\frac{1}{2m}}}\right)^{\frac{1}{2}} + \frac{\lambda\prod_{p=1}^{m}\left\|\mathbf{W}^{(p)}\right\|_2^{\frac{2}{m}}}{\tilde{d}}\right)\right] \\ &\overset{(m)}{\leq} 2\mathbb{E}\left[\exp\left(-\left(\frac{\sqrt{2}\lambda\tilde{d}_{\mathsf{TRP}}^{\frac{1}{2m}}\epsilon^{\frac{1}{m}}}{e\beta_m(\mathbf{x},\mathbf{y})^{\frac{1}{2m}}}\right)^{\frac{1}{2}} + \frac{\lambda\sum_{p=1}^{m}\left\|\mathbf{W}^{(p)}\right\|_2^2}{m\tilde{d}}\right)\right], \end{aligned} \tag{5.53}$$

where (k) follows form multiplying and dividing with a $\lambda > 0$, (l) from applying Lemma 5.14

with $p = \frac{1}{2}$ and $q = 1$, and (m) from the arithmetic-geometric mean inequality.

Bounding the MGF of ${w_{i,j}^{(p)}}^2$ for $p \in [m], i \in [d], j \in [\tilde{d}]$, we have that

$$\begin{aligned}\mathbb{E}\left[\exp\left(\lambda {w_{i,j}^{(p)}}^2\right)\right] &\overset{(n)}{\leq} \sum_{k\geq 0} \mathbb{E}\left[{w_{i,j}^{(p)}}^k\right]\frac{\lambda^k}{k!} \overset{(o)}{\leq} 1 + \lambda S + \frac{\lambda^2 S^2}{2}\sum_{k\geq 0}(\lambda R)^k \\ &\overset{(p)}{=} 1 + \lambda S + \frac{\lambda^2 S^2}{2}\frac{1}{1-\lambda R} \overset{(q)}{=} 1 + \frac{S}{2R} + \frac{S^2}{4R^2},\end{aligned}$$

where (n) is the Taylor expansion, (o) is the condition (5.50) and applying Jensen inequality to the degree-1 term, (p) is the geometric series for $\lambda < \frac{1}{R}$, and (q) is choosing $\lambda = \frac{1}{2R}$. Hence,

$$\begin{aligned}\mathbb{E}\left[\exp\left(\frac{\lambda\sum_{p=1}^m \left\|\mathbf{W}^{(p)}\right\|_2^2}{m\tilde{d}}\right)\right] &= \mathbb{E}\left[\exp\left(\frac{\lambda\sum_{p=1}^m\sum_{i=1}^d\sum_{j=1}^{\tilde{d}} {w_{i,j}^{(p)}}^2}{m\tilde{d}}\right)\right] \\ &\overset{(r)}{\leq} \mathbb{E}^{1/(m\tilde{d})}\left[\exp\left(\lambda\sum_{p=1}^m\sum_{i=1}^d\sum_{j=1}^{\tilde{d}} {w_{i,j}^{(p)}}^2\right)\right] \\ &\overset{(s)}{\leq} \left(1 + \frac{S}{2R} + \frac{S^2}{4R^2}\right)^d,\end{aligned}$$

where (r) is due to the Jensen inequality (Lemma 5.11), and (s) follows from the independence of the $w_{i,j}^{(p)}$'s for $p \in [m], i \in [d], j \in [\tilde{d}]$. Finally, plugging this into (5.53), we get that

$$\mathbb{P}\left[\left|\tilde{\mathsf{k}}_{\mathsf{S}_m}^{\mathsf{TRP}}(\mathbf{x},\mathbf{y}) - \tilde{\mathsf{k}}_{\mathsf{S}_m}(\mathbf{x},\mathbf{y})\right| \geq \epsilon\right] \leq 2\left(1 + \frac{S}{2R} + \frac{S^2}{4R^2}\right)^d \exp\left(-\left(\frac{\tilde{d}_{\mathsf{TRP}}^{\frac{1}{2m}}\epsilon^{\frac{1}{m}}}{\sqrt{2}eR\beta_m(\mathbf{x},\mathbf{y})^{\frac{1}{2m}}}\right)^{\frac{1}{2}}\right).$$

Finally, note that $\beta_m(\mathbf{x},\mathbf{y})^{\frac{1}{2m}} = \left(\frac{3^m+1}{(m!)^4}\right)^{\frac{1}{2m}} \|\mathbf{x}\|_{1\text{-var}} \|\mathbf{y}\|_{1\text{-var}} \leq \frac{2e^2}{m^2}\|\mathbf{x}\|_{1\text{-var}}\|\mathbf{y}\|_{1\text{-var}}$, since $3^m + 1 \leq 4^m$ for $m \geq 1$, and $\frac{1}{m!} \leq \left(\frac{e}{m}\right)^m$ due to Stirling's approximation. □

5.D Algorithms

Algorithm 7 Computing the RFSF map $\tilde{\varphi}_{\mathsf{S}_{\leq M}}$.

Input: Sequences $\mathbf{X} = (\mathbf{x}_i)_{i=1}^N \subset \mathsf{Seq}(\mathcal{X})$, measure Λ, truncation $M \in \mathbb{Z}_+$, RFF sample size $\tilde{d} \in \mathbb{Z}_+$
Optional: Add time-parameterization $\mathbf{x}_i \leftarrow (\mathbf{x}_{i,t}, t/\ell_{\mathbf{x}_i})_{t=1}^{\ell_{\mathbf{x}_i}}$ for all $i \in [N]$
Tabulate to uniform length $\ell = \max_{j\in[N]} \ell_{\mathbf{x}_j}$ by $\mathbf{x}_i \leftarrow (\mathbf{x}_{i,1}, \ldots, \mathbf{x}_{i,\ell_{\mathbf{x}_i}}, \ldots, \mathbf{x}_{i,\ell_{\mathbf{x}_i}})$ for all $i \in [N]$
Sample independent RFF weights $\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(M)} \overset{\text{i.i.d.}}{\sim} \Lambda^{\tilde{d}}$
Initialize an array U with shape $[M, N, \ell-1, 2\tilde{d}]$
Compute increments $U[m,i,t,:] \leftarrow \tilde{\varphi}_m(\mathbf{x}_{i,t+1}) - \tilde{\varphi}_m(\mathbf{x}_{i,t})$ for $m \in [M]$, $i \in [N]$, $t \in [\ell-1]$
Initialize array $V \leftarrow U[1,:,:,:]$
Collapse into level-1 features $P_1 \leftarrow V[:,\Sigma,:]$
for $m = 2$ **to** M **do**
    Update with next increment $V \leftarrow V[:,\boxplus+1,:] \boxtimes_3 U[m,:,:,:]$
    Collapse into level-m features $P_m \leftarrow V[:,\Sigma,:]$
end for
Output: Arrays of RFSF features per signature level $P_1, \ldots, P_M$.

Algorithm 8 Computing the RFSF-DP map $\tilde{\varphi}^{\mathsf{DP}}_{\mathsf{S}_{\leq M}}$.

Input: Sequences $\mathbf{X} = (\mathbf{x}_i)_{i=1}^N \subset \mathsf{Seq}(\mathcal{X})$, measure Λ, truncation $M \in \mathbb{Z}_+$, RFSF-DP sample size $\tilde{d} \in \mathbb{Z}_+$
Optional: Add time-parameterization $\mathbf{x}_i \leftarrow (\mathbf{x}_{i,t}, t/\ell_{\mathbf{x}_i})_{t=1}^{\ell_{\mathbf{x}_i}}$ for all $i \in [N]$
Tabulate to uniform length $\ell = \max_{j\in[N]} \ell_{\mathbf{x}_j}$ by $\mathbf{x}_i \leftarrow (\mathbf{x}_{i,1}, \ldots, \mathbf{x}_{i,\ell_{\mathbf{x}_i}}, \ldots, \mathbf{x}_{i,\ell_{\mathbf{x}_i}})$ for all $i \in [N]$
Sample independent RFF weights $\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(M)} \overset{\text{i.i.d.}}{\sim} \Lambda^{\tilde{d}}$
Initialize an array U with shape $[M, N, \ell-1, \tilde{d}, 2]$
Compute increments $U[m,i,t,k,:] \leftarrow \hat{\varphi}_{m,k}(\mathbf{x}_{i,t+1}) - \hat{\varphi}_{m,k}(\mathbf{x}_{i,t})$ for $m \in [M]$, $i \in [N]$, $t \in [\ell-1]$, $k \in [\tilde{d}]$
Initialize array $V \leftarrow \frac{1}{\sqrt{\tilde{d}}} U[1,:,:,:,:]$
Collapse into level-1 features $P_1 \leftarrow V[:,\Sigma,:,:]$
for $m = 2$ **to** M **do**
    Update with next increment $V \leftarrow V[:,\boxplus+1,:,:] \boxtimes_4 U[m,:,:,:,:]$
    Collapse into level-m features $P_m \leftarrow V[:,\Sigma,:,:]$
end for
Output: Arrays of RFSF-DP features per signature level $P_1, \ldots, P_M$.

Algorithm 9 Computing the RFSF-TRP map $\tilde{\varphi}^{\mathsf{TRP}}_{\mathsf{S}_{\leq M}}$.

Input: Sequences $\mathbf{X} = (\mathbf{x}_i)_{i=1}^N \subset \mathsf{Seq}(\mathcal{X})$, measure Λ, truncation $M \in \mathbb{Z}_+$, RFSF and TRP sample size $\tilde{d} \in \mathbb{Z}_+$
Optional: Add time-parameterization $\mathbf{x}_i \leftarrow (\mathbf{x}_{i,t}, t/\ell_{\mathbf{x}_i})_{t=1}^{\ell_{\mathbf{x}_i}}$ for all $i \in [N]$
Tabulate to uniform length $\ell = \max_{j\in[N]} \ell_{\mathbf{x}_j}$ by $\mathbf{x}_i \leftarrow (\mathbf{x}_{i,1}, \ldots, \mathbf{x}_{i,\ell_{\mathbf{x}_i}}, \ldots, \mathbf{x}_{i,\ell_{\mathbf{x}_i}})$ for all $i \in [N]$
Sample independent RFF weights $\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(M)} \overset{\text{i.i.d.}}{\sim} \Lambda^{\tilde{d}}$
Sample standard normal matrices $\mathbf{P}^{(1)}, \ldots, \mathbf{P}^{(M)} \overset{\text{i.i.d.}}{\sim} \mathcal{N}^{\tilde{d}}(0, \mathbf{I}_{2\tilde{d}})$
Initialize an array U with shape $[M, N, \ell-1, \tilde{d}]$
Compute projected increments $U[m,i,t,:] \leftarrow \mathbf{P}^{(m)\top}\left(\tilde{\varphi}_m(\mathbf{x}_{i,t+1}) - \tilde{\varphi}_m(\mathbf{x}_{i,t})\right)$ for $m \in [M]$, $i \in [N]$, $t \in [\ell-1]$
Initialize array $V \leftarrow \frac{1}{\sqrt{\tilde{d}}} U[1,:,:,:]$
Collapse into level-1 features $P_1 \leftarrow V[:,\Sigma,:]$
for $m = 2$ **to** M **do**
    Update with next increment $V \leftarrow V[:,\boxplus+1,:] \odot U[m,:,:,:]$
    Collapse into level-m features $P_m \leftarrow V[:,\Sigma,:]$
end for
Output: Arrays of RFSF-TRP features per signature level $P_1, \ldots, P_M$.

Chapter 6

Recurrent Sparse Spectrum Signature Gaussian processes

6.1 Introduction

Time series forecasting plays a central role in various domains, including finance [238], renewable energy [283, 2], and healthcare [28]. Ongoing research faces several challenges, such as non-linear data domains, ordered structure, time-warping invariance, discrete and irregular sampling, and scalability. Among the numerous existing approaches, we focus on signature approach, which address the first four challenges and have demonstrated state-of-the-art performance as a kernel for sequential data [267]. However, the trade-off between accuracy and scalability remains unresolved. While the exact computation of the signature kernel requires quadratic complexity for sequence length L [146], existing approximate methods reduce complexity to linear, at the cost of degraded performance on large-scale datasets. Moreover, these methods treat the signature as a global feature, which limits their ability to capture recent local information effectively. This is particularly crucial for time series forecasting, where capturing near-term correlations are often as important as identifying long-term trends.

Contributions. We propose Random Fourier Decayed Signature Features with Gaussian Processes, which dynamically adjust its context length based on the data, prioritizing more recent information, and call the resulting model Recurrent Sparse Spectrum Signature Gaussian Process (RS^3GP). By incorporating variational inference, our model learns the decay parameters in a data-driven manner, allowing it to transform time series data into a joint predictive distribution at scale while maintaining accuracy. Our approach outperforms other GP baselines

for time series on both small and large datasets and demonstrates comparable performance to state-of-the-art deep learning diffusion models while offering significantly faster training times.

6.1.1 Related work

Gaussian process approach. There are two prominent approaches: (1) specialized kernels for sequences [176, 58, 60, 5], and (2) state-space models [86, 84, 187, 78, 71, 127]. These approaches can complement each other; modeling the latent system as a higher-order Markov process allows sequence kernels to capture the influence of past states. The random projection approach, also known as sparse spectrum approximation in the Gaussian process community [158, 288, 89, 62, 167].

Deep learning approach. From classic LSTMs [121] to transformers [275], deep learning methods have been widely applied to sequential data. In the context of probabilistic forecasting, diffusion models [242, 119, 149] are considered among the state-of-the-art techniques. Although deep learning models can approximate any continuous function, they often require a large number of parameters, suffer from high variance, and offer limited interpretability. This creates opportunities for alternative approaches, which can serve not only as competitors but also as complementary components within larger models.

6.2 Background

6.2.1 Signature Features and Kernels

The path signature $\mathsf{S}(\mathbf{x})$ provides a graded description of a path $\mathbf{x} : [0, T] \to \mathbb{R}^d$ by mapping it to a hierarchy of tensors $\mathsf{S}(\mathbf{x}) = (1, \mathsf{S}_1(\mathbf{x}), \mathsf{S}_2(\mathbf{x}), \ldots) \in \prod_{m\geq 0}(\mathbb{R}^d)^{\otimes m}$ of increasing degrees. Among its attractive properties are that: it maps from a nonlinear domain (there is not natural addition of paths of different length) to a linear space $\prod(\mathbb{R}^d)^{\otimes m}$; $\mathbf{x} \mapsto \mathsf{S}(\mathbf{x})$ is injective up to time-parameterization (thus naturally factoring out time-warping) and it can be made injective by adding time as a path coordinate (i.e. setting $x_t^0 = t$); it linearizes path functionals, that is we can approximate non-linear path functionals $f(\mathbf{x})$ as linear functionals of signatures $f(\mathbf{x}) \approx \langle \ell, \mathsf{S}(\mathbf{x}) \rangle$, see Section 1.2.2 for details. Additionally, if the path is random, the expected path signature characterizes the distribution, $\mu \mapsto \mathbb{E}_{\mathbf{x}\sim\mu}[\mathsf{S}(\mathbf{x})]$ is injective, see [44]. All this makes it a viable transformation for extracting information from time series and paths in machine learning.

Discrete-time signature features. In practice, we do not have access to continuous paths but are only given a sequence $\mathbf{x} = (\mathbf{x}_0, \dots, \mathbf{x}_L) \in \mathsf{Seq}(\mathbb{R}^d)$ in $\mathbb{R}^d$. Here, we focus on the higher-order discretized signature variation from Section 1.2.6, where the order parameter is set to the truncation level $p = M$. Hence, in this case we can simply identify $\mathbf{x}$ with a continuous time path by its linear interpolation for any choice of time parameterization. To compute signatures, denote with $\bar{\Delta}_m(n)$ the collection of all ordered m-tuples such that

$$\bar{\Delta}_m(n) = \{1 \leq i_1 \leq \cdots \leq i_m \leq n \,:\, i_1, \dots, i_m \in [n]\}\,.$$

Then, it can be shown using Chen's relation (Proposition 1.14) that its signature can be computed by the recursion for $1 \leq m$, and $1 \leq l \leq L$:

$$\mathsf{S}_m(\mathbf{x}_{0:l}) = \mathsf{S}_m(\mathbf{x}_{0:l-1}) + \sum_{p=1}^{m} \mathsf{S}_{m-p}(\mathbf{x}_{0:l-1}) \otimes \frac{(\delta \mathbf{x}_l)^{\otimes p}}{p!} \tag{6.1}$$

with the identification $S_m(\mathbf{x}_{0:0}) \equiv 0$, and the first-order difference operator is defined as $\delta \mathbf{x}_k = \mathbf{x}_k - \mathbf{x}_{k-1}$. Unrolling this along the length of the time series with respect to the first term, we get

$$\mathsf{S}_m(\mathbf{x}_{0:l}) = \sum_{k=1}^{l} \sum_{p=1}^{m} \mathsf{S}_{m-p}(\mathbf{x}_{0:k-1}) \otimes \frac{(\delta \mathbf{x}_k)^{\otimes p}}{p!}.$$

Recall from Section 1.2.6 that this expression can be evaluated in closed form:

$$\mathsf{S}_m(\mathbf{x}_{0:k}) = \sum_{\mathbf{i} \in \bar{\Delta}_m(\mathsf{L}_\mathbf{x})} \frac{1}{\mathbf{i}!} \delta \mathbf{x}_{i_1} \otimes \cdots \otimes \delta \mathbf{x}_{i_m}, \tag{6.2}$$

where $\mathbf{i}! = k_1! \cdots k_q!$ such that there are $q \in \mathbb{Z}_+$ unique indices in the multi-index $\mathbf{i}$ and $k_1, \cdots, k_q$ are the number of times they are repeated. It will be important later on that (6.1) allows to compute the signature up to all time steps using a paralellizable scan operation.

Discrete-time signature kernels. As $\mathsf{S}_m(\mathbf{x})$ is tensor-valued, it has d^m coordinates, which makes its computation infeasible for high-dimensional state-spaces or high m. Kernelization allows to alleviate this by first lifting sequences and paths from $\mathbb{R}^d$ into a RKHS and subsequently computing inner products of signature features in this RKHS. This alleviates the computational bottleneck and further adds more expressivity by first lifting the paths into the RKHS; see Section

1.2.4. Concretely, the signature kernel $\mathsf{k}_{\mathsf{S}} : \mathsf{Seq}(\mathbb{R}^d) \times \mathsf{Seq}(\mathbb{R}^d) \to \mathbb{R}$ is defined as

$$\mathsf{k}_{\mathsf{S}}(\mathbf{x}, \mathbf{y}) = \sum_{m=0}^{M} \mathsf{k}_{S_m}(\mathbf{x}, \mathbf{y}), \tag{6.3}$$

where $M \in \mathbb{Z}_+ \cup \{\infty\}$ is the truncation level. Although truncating the series (6.3) may seem like a limitation, it often happens that a careful selection of finite truncation level outperforms the untruncated case, since it can mitigate overfitting.

If we identify two sequences $\mathbf{x}, \mathbf{y} \in \mathsf{Seq}(\mathbb{R}^d)$ with paths given by their linear interpolations for some choice of time parameterization, then we get the analogous expression to (6.2):

$$\mathsf{k}_{S_m}(\mathbf{x}, \mathbf{y}) = \sum_{\substack{\mathbf{i} \in \bar{\Delta}(\mathsf{L}_{\mathbf{x}}) \\ \mathbf{j} \in \bar{\Delta}(\mathsf{L}_{\mathbf{y}})}} \frac{1}{\mathbf{i}!\mathbf{j}!} \delta_{i_1, j_1} \mathsf{k}(\mathbf{x}_{i_1}, \mathbf{y}_{j_1}) \cdots \delta_{i_m, j_m} \mathsf{k}(\mathbf{x}_{i_m}, \mathbf{y}_{j_m}), \tag{6.4}$$

where $\mathbf{i}!$ is as previously, and $\delta_{i,j}$ is a second-order differencing operator, such that $\delta_{i,j}\mathsf{k}(\mathbf{x}_i, \mathbf{y}_j) = \mathsf{k}(\mathbf{x}_{i+1}, \mathbf{y}_{j+1}) - \mathsf{k}(\mathbf{x}_{i+1}, \mathbf{y}_j) - \mathsf{k}(\mathbf{x}_i, \mathbf{y}_{j+1}) + \mathsf{k}(\mathbf{x}_i, \mathbf{y}_j)$. A similar recursion to (6.1) exists to compute (6.4), see Algorithm 2 in Section 1.3. For further details, see Sections 1.2.4 and 1.2.6.

6.2.2 Random Fourier Features

Kernelization circumvents the computational burden of a high- or infinite-dimensional feature space, but the cost is the associated quadratic complexity in the number of samples which is due to the evaluation of the Gram matrix. Random Fourier Features [209] address this by constructing for a given kernel k on $\mathbb{R}^d$ a random feature map for which the inner product is a good random approximation to the original kernel. Many variations proposed, see [173, 35].

Existing variations are based on Bochner's theorem, which allows for a spectral representation of stationary kernels. Let $\mathsf{k} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be a continuous, bounded, stationary kernel. Then, Bochner's theorem states that there exists a non-negative finite measure Λ over $\mathbb{R}^d$, such that k is its Fourier transform. In other words, for $\mathbf{x}, \mathbf{y}, \in \mathbb{R}^d$, k can be be represented as

$$\begin{aligned} \mathsf{k}(\mathbf{x}, \mathbf{y}) &= \int_{\mathbb{R}^d} \exp(i\boldsymbol{\omega}^\top(\mathbf{x} - \mathbf{y}))\, \mathrm{d}\Lambda(\boldsymbol{\omega}) \\ &= \int_{\mathbb{R}^d} \cos(\boldsymbol{\omega}^\top(\mathbf{x} - \mathbf{y})\, \mathrm{d}\Lambda(\boldsymbol{\omega}), \end{aligned} \tag{6.5}$$

since the kernel is real-valued. Without loss of generality, we may assume that $\Lambda(\mathbb{R}^d) = 1$ so Λ is a probability measure, since it amounts to rescaling the kernel.

Now, we may draw Monte Carlo samples to approximate (6.5), $\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_D \overset{\text{i.i.d.}}{\sim} \Lambda$ for $D \in \mathbb{Z}_+$:

$$\mathsf{k}(\mathbf{x},\mathbf{y}) \approx \frac{1}{D}\sum_{i=1}^{D}\cos(\boldsymbol{\omega}_i^\top(\mathbf{x}-\mathbf{y})). \tag{6.6}$$

There are two ways to proceed from here: one is to use the cosine identity $\cos(x-y) = \cos(x)\cos(y)+\sin(x)\sin(y)$, or to further approximate the sum in (6.6) using the identity

$$\cos(x-y) = \mathbb{E}_{b\sim\mathcal{U}(0,2\pi)}\left[\sqrt{2}\cos(x+b)\sqrt{2}\cos(y+b)\right] \tag{6.7}$$

see [89, App. A]. We approximate the expectation (6.6) by drawing 1 sample for each term:

$$\mathsf{k}(\mathbf{x},\mathbf{y}) \approx \frac{2}{D}\sum_{i=1}^{D}\cos(\boldsymbol{\omega}_i^\top\mathbf{x}+b_i)\cos(\boldsymbol{\omega}_i^\top\mathbf{y}+b_i),$$

where $b_1,\ldots,b_D \sim \mathcal{U}(0,2\pi)$. This represents each sample using a 1-dimensional feature, which will be important for us later on in Section 6.3.1 when constructing our variation of the Random Fourier Signature Feature map. Now, let $\Omega = (\boldsymbol{\omega}_1,\ldots,\boldsymbol{\omega}_D) \in \mathbb{R}^{d\times D}$ and $\mathbf{b} = (b_1,\ldots,b_D) \in \mathbb{R}^D$. Then, the RFF feature map can be written concisely as

$$\tilde{\varphi}(\mathbf{x}) = \sqrt{\frac{2}{p}}\cos\left(\Omega^\top\mathbf{x}+\mathbf{b}\right). \tag{6.8}$$

This offers to approximate a stationary kernel using a random, finite-dimensional feature map, and consequently to reformulate downstream algorithms in weight-space, which avoids the usual cubic costs in the number of data points. Probabilistic convergence of the RFF is studied in the series of works: [209], optimal rates were derived in [244], extended to the kernel derivatives in [255], conditions on the spectral measure relaxed in [35].

6.3 Recurrent Sparse Spectrum Signature Gaussian processes

6.3.1 Revisiting Random Fourier Signature Features

Now, we revisit how RFFs can be combined with signatures to extract random features from a time series. To do so, we adapt the construction of Random Fourier Signature Features (RFSF) from Chapter 5. We provide an adapted version of the diagonally projected RFSF variant (RFSF-DP), with a twist, which allows us to devise even more computationally efficient features.

We will focus on the discrete-time setting, but the construction applies to continuous-time in

an analogous manner. Firstly, we will build a 1-dimensional random estimator for the signature kernel. Let $\mathsf{k} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be a continuous, bounded, stationary kernel with spectral measure Λ, $\boldsymbol{\omega}^{(1)}, \ldots, \boldsymbol{\omega}^{(m)} \sim \Lambda$ and $b^{(1)}, \ldots, b^{(m)} \sim \mathcal{U}(0, 2\pi)$, and $\phi^{(p)}(\mathbf{x}) := \cos({\boldsymbol{\omega}^{(p)}}^\top \mathbf{x} + b^{(p)})$ for $\mathbf{x} \in \mathbb{R}^d$. Then, for sequences $\mathbf{x}, \mathbf{y} \in \mathsf{Seq}(\mathbb{R}^d)$, it holds for the signature kernel $\mathsf{k}_{\mathsf{S}_m}$ as defined in (6.4) that

$$\mathsf{k}_{\mathsf{S}_m}(\mathbf{x}, \mathbf{y}) = \mathbb{E}\left[\sum_{\substack{\mathbf{i} \in \tilde{\Delta}_m(\mathsf{L}_\mathbf{x}) \\ \mathbf{j} \in \tilde{\Delta}_m(\mathsf{L}_\mathbf{y})}} \frac{1}{\mathbf{i}!\mathbf{j}!} \prod_{p=1}^{m} \sqrt{2}\delta\phi^{(p)}(\mathbf{x}_{i_1}) \sqrt{2}\delta\phi^{(p)}(\mathbf{y}_{j_1})\right]. \tag{6.9}$$

The unbiasedness of the expectation is straightforward to check, since due to linearity and independence of $\phi^{(p)}$'s, it reduces to verifying the relations (6.7) and (6.5), which in turn recovers the formulation of the signature kernel from (6.4). Next, we draw $D \in \mathbb{Z}_+$ Monte Carlo samples to approximate (6.9), hence, let $\boldsymbol{\omega}_1^{(1)}, \ldots, \boldsymbol{\omega}_D^{(m)} \overset{\text{i.i.d.}}{\sim} \Lambda$ and $b_1^{(1)}, \ldots, b_D^{(m)} \overset{\text{i.i.d.}}{\sim} \mathcal{U}(0, 2\pi)$, and $\phi_k^{(p)}(\mathbf{x}) = \cos({\boldsymbol{\omega}_k^{(p)}}^\top \mathbf{x} + b_k^{(p)})$ for $\mathbf{x} \in \mathbb{R}^d$. Then,

$$\mathsf{k}_{\mathsf{S}_m}(\mathbf{x}, \mathbf{y}) \approx \frac{1}{D} \sum_{k=1}^{D} \sum_{\substack{\mathbf{i} \in \tilde{\Delta}_m(\mathsf{L}_\mathbf{x}) \\ \mathbf{j} \in \tilde{\Delta}_m(\mathsf{L}_\mathbf{y})}} \frac{1}{\mathbf{i}!\mathbf{j}!} \prod_{p=1}^{m} \sqrt{2}\delta\phi_k^{(p)}(\mathbf{x}_{i_1}) \sqrt{2}\delta\phi_k^{(p)}(\mathbf{y}_{j_1}). \tag{6.10}$$

We call this approximation Random Fourier Signature Features (RFSF), and the features corresponding to the kernel on the RHS of (6.10) can be represented in a finite-dimensional feature space. Let us collect the frequencies $\Omega^{(p)} = (\boldsymbol{\omega}_1^{(p)}, \ldots, \boldsymbol{\omega}_D^{(p)}) \in \mathbb{R}^{d \times D}$ and phases $\mathbf{b}^{(p)} = (b_1^{(p)}, \ldots, b_D^{(p)})^\top \in \mathbb{R}^D$, and define the RFF maps as in (6.8) so we have $\varphi^{(p)}(\mathbf{x}) = \cos({\Omega^{(p)}}^\top \mathbf{x} + \mathbf{b}^{(p)})$. Then, the RFSF map $\Phi_m : \mathsf{Seq}(\mathbb{R}^d) \to \mathbb{R}^D$ is defined for $\mathbf{x} \in \mathsf{Seq}(\mathbb{R}^d)$ as

$$\Phi_m(\mathbf{x}) = \sqrt{\frac{2^m}{D}} \sum_{\mathbf{i} \in \tilde{\Delta}_m(K)} \frac{1}{\mathbf{i}!} \bigodot_{p=1}^{m} \delta\varphi^{(p)}(\mathbf{x}_{i_p}), \tag{6.11}$$

where $\odot$ refers to the Hadamard product. Although (6.11) looks difficult to compute, we can establish a recursion across time and signature levels. The update rule analogous to (6.1) is

$$\Phi_m(\mathbf{x}_{0:l}) = \Phi_m(\mathbf{x}_{0:l-1}) + \sum_{p=1}^{m} \frac{1}{p!} \Phi_{m-p}(\mathbf{x}_{0:l-1}) \bigodot_{q=m-p+1}^{m} \delta\varphi^{(q)}(\mathbf{x}_l), \tag{6.12}$$

which can be computed efficiently using parallelizable scan operation. Additionally, we overload the definition of the differencing operator δ to mean fractional differencing for some learnable differencing order $\alpha \in (0, 1)$. Thus, in (6.11), we learn a separate differencing order parameter

for each feature channel, see Appendix 6.A.1.

In practive, we use the first $M \in \mathbb{Z}_+$ levels in conjunction, and normalize each RFSF level to unit norm, so that the full RFSF map $\Phi : \mathsf{Seq}(\mathbb{R}^d) \to \mathbb{R}^{MD+1}$ is

$$\Phi(\mathbf{x}) = \left(1, \frac{\Phi_1(\mathbf{x})}{\|\Phi_1(\mathbf{x})\|_2}, \dots, \frac{\Phi_M(\mathbf{x})}{\|\Phi_M(\mathbf{x})\|_2}\right).$$

We note that $\Phi(\mathbf{x})$ can be computed with complexity $O((M+W)LD + MLDd)$, where $W \in \mathbb{Z}_+$ is the window size for fractional differencing, see Appendix 6.A.2. This is more efficient than the scalable variants of Chapter 5, i.e. it avoids the 2^M factor as in RFSF-DP, and it also avoids the D^2 factor as in RFSF-TRP.

6.3.2 A forgetting mechanism for signatures

The recursive step in RFSF (6.12) allows to compute the feature map up to all time steps of a time series, similarly to a recurrent neural network (RNN), in one forward pass. This suggests to perform sequence-to-sequence regression (including time series forecasting, when the target sequence is future values of the time series) by considering the RFSF map over an expanding window. However, signatures have no built-in forgetting mechanism, and it is well-known that the level-m signature feature has magnitude $O(\|\mathbf{x}\|_{1\text{-var}}^m / m!)$, and the same is true for (6.11), which shows that RFSF only accumulates information without a way of forgetting. This is in contrast to modern RNNs, which have built-in gating mechanisms that allow or disallow the flow of historical information, allowing them to focus on more recent information.

In this section, we propose a novel forgetting mechanism tailored for RFSF, but which can be applied to signature features in general. We tackle this by introducing time step dependent decay factors, which multiply each increment in the formulation (6.11). We assume exponential decay in time, so let $\boldsymbol{\lambda} \in \mathbb{R}^D$ be channel-wise decay factors. Then, we define Random Fourier Decayed Signature Features (RFDSF) as

$$\Phi_m(\mathbf{x}_{0:l}) = \sqrt{\frac{2^m}{D}} \sum_{\mathbf{i} \in \bar{\Delta}_m(K)} \frac{1}{\mathbf{i}!} \bigodot_{p=1}^{m} \boldsymbol{\lambda}^{\odot(l-i_p)} \odot \delta\varphi^{(p)}(\mathbf{x}_{i_p}). \tag{6.13}$$

A recursion analogous to (6.12) allows to compute (6.13):

$$\Phi_m(\mathbf{x}_{0:l}) = \boldsymbol{\lambda}^{\odot m} \odot \Phi_m(\mathbf{x}_{0:l-1}) + \sum_{p=1}^{m} \frac{1}{p!} \boldsymbol{\lambda}^{\odot(m-p)} \odot \Phi_{m-p}(\mathbf{x}_{0:l-1}) \bigodot_{q=m-p+1}^{m} \delta\varphi^{(q)}(\mathbf{x}_l),$$

which can be unrolled over the length of a sequence so

$$\Phi_m(\mathbf{x}_{0:l}) = \sum_{k=1}^{l} \boldsymbol{\lambda}^{\odot m(l-k)} \sum_{p=1}^{m} \frac{1}{p!} \boldsymbol{\lambda}^{\odot(m-p)} \odot \Phi_{m-p}(\mathbf{x}_{0:l-1}) \bigodot_{q=m-p+1}^{m} \delta\varphi^{(q)}(\mathbf{x}_k), \tag{6.14}$$

which is an exponential decay over time steps of an appropriately chosen sequence depending on signature levels lower than m. Hence, (6.14) is still computable by a parallelizable scan operation, allowing for sublinear time computations. This makes the feature map well-suited for long time series, due to the adjustment of context length by the decay factors, and the possibility for parallelization across time. The theoretical complexity is the same as previously, i.e. $O((M+W)MLD+MLDd)$, see Appendix 6.A.2, but we emphasize that a work-efficient scan algorithm allows to compute the recursion on a GPU log-linearly.

6.3.3 Recurrent Sparse Spectrum Signature Gaussian processes

Next, we construct our (Variational) Recurrent Sparse Spectrum Signature Gaussian Process ((V)RS^{3}GP) model. We define a Gaussian process, which treats the hyperparameters of the random covariance function in a probabilistic fashion [89] by incorporating them into inference. We formulate our variational GP model in the feature space, as opposed to in the function space.

Bayesian formulation. We focus on supervised learning, and assume our dataset consists of multiple input time series $\mathbf{x} \in \mathsf{Seq}(\mathbb{R}^d)$ each with a corresponding univariate output time series $\mathbf{y} \in \mathsf{Seq}(\mathbb{R})$, such that $\mathsf{L}_\mathbf{x} = \mathsf{L}_\mathbf{y}$. We note that the multivariate case can also be handled by stacking multiple GP priors, one for each output coordinate.

As the signature is a universal feature map, we expect that a linear layer on top of the computed RFDSF features will work well for approximating functions of time series. We assume a probabilistic model, which models the prediction process as a linear layer on top of RFDSF, corrupted by Gaussian noise, that is,

$$y_l = \mathbf{w}^\top \Phi(\mathbf{x}_{0:l}) + \epsilon_l \quad \text{for } l = 0, \dots, L,$$

where $\mathbf{w} \in \mathbb{R}^{MD+1}$, such that M is the signature truncation level and D is the RFF dimension. We place an i.i.d. standard Gaussian prior on $\mathbf{w}$, $p(\mathbf{w}) = \mathcal{N}(0, I_{MD+1})$. The model noise is i.i.d. Gaussian with learnable variance $\sigma_y^2 > 0$, so that $p(y \,|\, \mathbf{w}, \boldsymbol{\Omega}, \mathbf{B}, \mathbf{x}) = \mathcal{N}(\mathbf{w}^\top \Phi(\mathbf{x}), \sigma_y^2)$, where we explicitly denoted the dependence on the hyperparameters of the RFSF map Φ, frequencies $\boldsymbol{\Omega} = (\Omega^{(1)}, \dots, \Omega^{(M)}) \in \mathbb{R}^{M \times d \times D}$, and phases $B = (\mathbf{b}^{(1)}, \dots, \mathbf{b}^{(M)}) \in \mathbb{R}^{M \times D}$. Next, we specialize to the case of the ARD Gaussian kernel, which has $\Lambda = \mathcal{N}(\mathbf{0}, D^{-1})$, where $D \in \mathbb{R}_+^{D \times D}$ is a diagonal

matrix of lengthscales. We assign independent lengthscales to each random feature map in (6.13). Then, following previous work [89, 57], we place the priors on the RFF parameters

$$p(\mathbf{\Omega}) = \prod_{m=1}^{M} p(\Omega^{(m)}) = \prod_{m=1}^{m} \mathcal{N}^D(\mathbf{0}, D_m^{-1}), \tag{6.15}$$
$$p(B) = \prod_{m=1}^{M} p(\mathbf{b}^{(m)}) = \prod_{m=1}^{M} \mathcal{U}^D(0, 2\pi),$$

where $D_m = \text{diag}(\ell_{m,1}, \ldots, \ell_{m,d})$ refers to the diagonal lengthscale matrix of the m^{th} RFF map.

As noted in [57], this model corresponds to a 2-layer Bayesian neural network, where the first layer is given by the RFDSF activations with priors on it as given by (6.15), while the second layer is a linear readout layer with a Gaussian prior on it. Inference in this case is analytically intractable. For a Gaussian likelihood, although $\mathbf{w}$ can be marginalized out in the log-likelihood, it is not clear how to handle the integrals with respect to $\mathbf{\Omega}$ and B. Further, disregarding the latter issue of additional hyperparameters, it only admits full-batch training, and this limits scalability to datasets with large numbers of input-output pairs. To this end, we introduce a variational approximation, which allows for both scalability and flexibility.

Variational treatment. We variationally approximate the posterior to define an evidence lower bound (ELBO). We define factorized variational distributions over the parameters $\mathbf{w} \in \mathbb{R}^{MD+1}$, frequencies $\mathbf{\Omega} \in \mathbb{R}^{M \times d \times D}$, and phases $B \in \mathbb{R}^{M \times D}$. The variational over $\mathbf{w} \in \mathbb{R}^{MD+1}$ is as usual given by a Gaussian $q(\mathbf{w}) = \mathcal{N}(\boldsymbol{\mu}_{\mathbf{w}}, \Sigma_{\mathbf{w}})$, where $\boldsymbol{\mu}_{\mathbf{w}} \in \mathbb{R}^{MD+1}$ is the variational mean and $\Sigma_{\mathbf{w}} \in \mathbb{R}^{(MD+1)\times(MD+1)}$ is the variational covariance matrix, which is symmetric and positive definite, represented in terms of its Cholesky factor, $L_{\mathbf{w}} \in \mathbb{R}^{(MD+1)\times(MD+1)}$, such that $\Sigma_{\mathbf{w}} = L_{\mathbf{w}} L_{\mathbf{w}}^{\top}$. The other parameters get the factorized variationals

$$q(\mathbf{\Omega}) = \prod_{m=1}^{M} q(\Omega^{(m)}) = \prod_{m=1}^{M} \prod_{i=1}^{D} \prod_{j=1}^{d} \mathcal{N}(\mu_{mij}, \sigma_{mij}^2),$$
$$q(B) = \prod_{m=1}^{M} q(\mathbf{b}^{(m)}) = \prod_{m=1}^{M} \prod_{i=1}^{D} \mathcal{B}_{[0,2\pi]}(\alpha_{mi}, \beta_{mi}),$$

where $\alpha_{mi}, \beta_{mi} > 0$, so that the posteriors over the component frequencies of ω are independent Gaussians, while the posteriors over the phases b are independent beta distributions on $[0, 2\pi]$.

The KL-divergence to the posterior is then minimized:

$$\mathsf{D}_{\mathsf{KL}}\left[q(\mathbf{w}, \mathbf{\Omega}, B) \,\|\, p(\mathbf{w}, \mathbf{\Omega}, B \,|\, \mathbf{y})\right] = \int q(\mathbf{w}, \mathbf{\Omega}, B) \log \frac{q(\mathbf{w}, \mathbf{\Omega}, B)}{p(\mathbf{w}, \mathbf{\Omega}, B \,|\, \mathbf{y})} \, \mathrm{d}\mathbf{w} \, \mathrm{d}\mathbf{\Omega} \, \mathrm{d}B.$$

Then, by the usual calculations, we get the ELBO, $\mathcal{L}_{\text{ELBO}} \leq \log p(\mathbf{y})$, such that

$$\mathcal{L}_{\text{ELBO}} = \underbrace{\sum_{i=1}^{N} \mathbb{E}_q \left[\log p(y_i \mid \mathbf{w}, \boldsymbol{\Omega}, B)\right]}_{\text{data-fit term}} - \underbrace{\mathsf{D}_{\mathsf{KL}}\left[q(\mathbf{w}) \,\|\, p(\mathbf{w})\right] - \mathsf{D}_{\mathsf{KL}}\left[q(\boldsymbol{\Omega}) \,\|\, p(\boldsymbol{\Omega})\right] - \mathsf{D}_{\mathsf{KL}}\left[q(B) \,\|\, p(B)\right]}_{\text{KL regularizers}} \tag{6.16}$$

The predictive distribution for a set of (possibly overlapping) sequences $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N) \subset \mathsf{Seq}(\mathbb{R}^d)$ is then, denoting $\Phi(\mathbf{X}) = [\Phi(\mathbf{x}_1), \ldots, \Phi(\mathbf{x}_N)]^\top \in \mathbb{R}^{N \times (MD+1)}$ and $\mathbf{f} = \Phi(\mathbf{X})\mathbf{w} \in \mathbb{R}^N$,

$$q(\mathbf{f}) = \mathcal{N}\left(\Phi(\mathbf{X})\boldsymbol{\mu}_{\mathbf{w}}^\top,\ \Phi(\mathbf{X}) L_{\mathbf{w}} L_{\mathbf{w}}^\top \Phi(\mathbf{X})^\top\right). \tag{6.17}$$

In order to evaluate the data-fit term in (6.16), and make inference about unseen points in (6.17), we require a way to handle the randomness in $\boldsymbol{\Omega}$ and B. Two solutions proposed in [57] are to perform Monte Carlo sampling or to sample them once at the start of training and keep them fixed while learning their distributional hyperparameters using the reparameterization trick [144]. Our observations align with theirs, and we found that resampling leads to high variance and non-convergence. The reason for this is likely that the factorized variational distributions do not model correlations between $\mathbf{w}$ and $\boldsymbol{\Omega}, B$. Hence, we use a fixed random outcome throughout with the reparameterization trick, see Appendix 6.A.3 for details.

Training objective. Although the ELBO (6.16) lower bounds the log-likelihood, it suffers from well-known pathologies. One of these is that it greatly underestimates the latent function uncertainty, and mainly relies on the observation noise for uncertainty calibration, which leads to overestimation. This is clearly unideal for time series forecasting, where heteroscedasticity is often present, and since useful information carried by the kernel about the geometry of the data space is lost. There are several solutions proposed by [133], and we adopt the approach called Parametric Predictive GP Regression (PPGPR), which restores symmetry between the training objective and the test time predictive distribution by treating both latent function uncertainty and observation noise on equal footing. The key idea is to modify the data-fit term in (6.16) taking the log outside the q-expectation so we have

$$\mathcal{L}_{\text{PPGPR}} = \sum_{i=1}^{N} \log \mathbb{E}_q \left[p(y_i \mid \mathbf{w}, \boldsymbol{\Omega}, B\right] - \mathsf{D}_{\mathsf{KL}}\left[q(\mathbf{w}) \,\|\, p(\mathbf{w})\right] - \mathsf{D}_{\mathsf{KL}}\left[q(\boldsymbol{\Omega}) \,\|\, p(\boldsymbol{\Omega})\right] - \mathsf{D}_{\mathsf{KL}}\left[q(B) \,\|\, p(B)\right]$$

which although is not a lower-bound anymore to the log-likelihood, leads to better calibrated uncertainties. However, as noted by [133], this objective can lead to underfitting in data regions, where a good fit is harder to achieve, and instead relying on uncertainty overestimation. We account for this by modifying the objective function by introducing a penalty term for the latent

function variance. Let $q(f_i) = \mathcal{N}(\mu_i, \sigma_i^2)$. Then, we modify the objective as

$$\mathcal{L} = \sum_{i=1}^{N} \log \mathbb{E}_q \left[p(y_i \mid \mathbf{w}, \mathbf{\Omega}, B \right] - \mathsf{D}_{\mathsf{KL}} \left[q(\mathbf{w}) \,\|\, p(\mathbf{w}) \right] - \mathsf{D}_{\mathsf{KL}} \left[q(\mathbf{\Omega}) \,\|\, p(\mathbf{\Omega}) \right] - \mathsf{D}_{\mathsf{KL}} \left[q(B) \,\|\, p(B) \right] - \alpha \sum_{i=1}^{N} \sigma_i^2,$$

where $\alpha > 0$ is a regularization hyperparameter. This promotes the optimizer to escape local optima, and rely on a better fit of the predictive mean, rather than on uncertainty overestimation to fit the data. We give further details in Appendix 6.A.4.

6.4 Experiments

Implementation. We implemented our models and GP baselines using PyTorch [203] and GPyTorch [91]. The computing cluster used has 4 NVIDIA RTX 3080 TI GPUs.

6.4.1 Synthetic Dataset

Dataset. We handcrafted a dataset to test the ability of our (V)RS^{3}GP model of adapting its context length to the data on a task which requires reasoning on multiple time horizons. The dataset consists of a multi-sinusoidal wave with multiple frequencies, which includes both large and low frequencies, requiring reasoning over both short and long time periods.

Methods. We compare three models: Variational Recurrent Sparse Spectrum Gaussian Process (VRS^{3}GP), our model constructed in the previous section; RS^{3}GP, which ablates the previous model as it does not learn variationals over the covariance parameters and sets them equal to the prior; Sparse Variational Gaussian Process (SVGP) using the RBF kernel and a fixed hand-tuned context length. (V)RS^{3}GP uses $D = 200$ and $M = 5$, while SVGP uses 100 inducing points.

Qualitative analysis. Figure 6.1 qualitatively illustrates the predictive mean and uncertainty of the different approaches. SVGP achieves a perfect fit of the training data, but in the testing regime fails to properly capture the underlying dynamics. RS^{3}GP properly captures both short and long horizon dynamics, but in certain data regions underfits the dataset. VRS^{3}GP on the other hand fits the data perfectly, and also perfectly captures the temporal dynamics, being more flexible than RS^{3}GP due to variational parameter learning.

6.4.2 Real-World Datasets

In this section, we present empirical results on several real-world datasets.

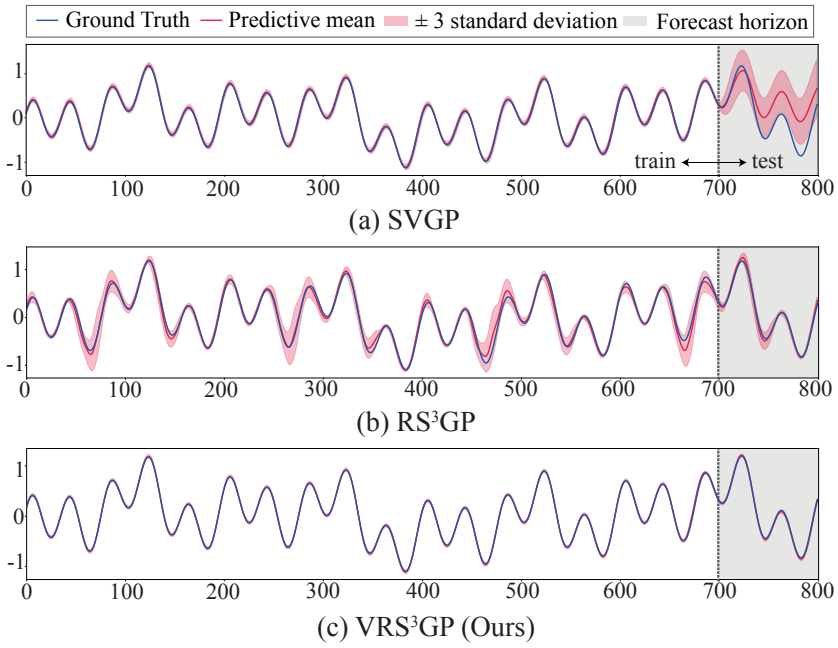

Figure 6.1: Predictive mean and uncertainty on a toy dataset composed of multi-sinusoidal waves with four distinct components including low and high frequencies, comparing various GP approaches. The true function is depicted in blue, while the predictive mean ±3 standard deviations are shown in red. The dataset consists of 700 training points, with a context window of 100 for SVGP, and a prediction horizon of 100 for all.

Datasets. We conducted experiments on eight univariate time series datasets from different domains, available in GluonTS [6]—Solar [154], Electricity [9], Traffic [9], Exchange [154], M4 [183], UberTLC [92], KDDCup [98], and Wikipedia [92]. We use GluonTS [6] to load the datasets, which has pre-specified train-test splits for each. See [262, App. B.1] for details.

Metric. We employed the continuous ranked probability score (CRPS) [97] for evaluating probabilistic forecasts. We approximate the CRPS by the normalized mean quantile loss, and report means and standard deviations over three independent runs, see [262, App. B.3].

Baselines. We extend the list of baselines from [149]. As such, we have included comprehensive baselines from three groups: classical statistics, deep learning, and other GP methods. For statistics, we included SeasonalNaïve, ARIMA, ETS, and a RidgeRegression model from the statistical literature [126]. Additionally, we compared against deep learning models that represent various architectural paradigms such as the RNN-based DeepAR [227], the CNN-based MQ-CNN [286], the state space model-based DeepState [214], the self-attention-based Transformer [275], and diffusion-model-based TSDiff [149]. For GP models, we add SVGP

Table 6.1: Forecasting results on eight benchmark datasets ranked by CRPS. The best and second best models have been shown as **bold** and underlined, respectively.

METHOD	SOLAR	ELECTRICITY	TRAFFIC	EXCHANGE	M4	UBERTLC	KDDCUP	WIKIPEDIA
SeasonalNaïve	0.512 ± 0.000	0.069 ± 0.000	0.221 ± 0.000	0.011 ± 0.000	0.048 ± 0.000	0.299 ± 0.000	0.561 ± 0.000	0.410 ± 0.000
ARIMA	0.545 ± 0.006	-	-	**0.008 ± 0.000**	0.044 ± 0.001	0.284 ± 0.001	0.547 ± 0.004	-
ETS	0.611 ± 0.040	0.072 ± 0.004	0.433 ± 0.050	**0.008 ± 0.000**	0.042 ± 0.001	0.422 ± 0.001	0.753 ± 0.008	0.715 ± 0.002
LINEAR	0.569 ± 0.021	0.088 ± 0.008	0.179 ± 0.003	0.011 ± 0.001	0.039 ± 0.001	0.360 ± 0.023	0.513 ± 0.011	1.624 ± 1.114
DeepAR	0.389 ± 0.001	0.054 ± 0.000	0.099 ± 0.001	0.011 ± 0.003	0.052 ± 0.006	**0.161 ± 0.002**	0.414 ± 0.027	0.231 ± 0.008
MQ-CNN	0.790 ± 0.063	0.067 ± 0.001	-	0.019 ± 0.006	0.046 ± 0.003	0.436 ± 0.020	0.516 ± 0.012	0.220 ± 0.001
DeepState	0.379 ± 0.002	0.075 ± 0.004	0.146 ± 0.018	0.011 ± 0.001	0.041 ± 0.002	0.288 ± 0.087	-	0.318 ± 0.019
Transformer	0.419 ± 0.008	0.076 ± 0.018	0.102 ± 0.002	0.010 ± 0.000	0.040 ± 0.014	0.192 ± 0.004	0.411 ± 0.021	**0.214 ± 0.001**
TSDiff	0.358 ± 0.020	**0.050 ± 0.002**	**0.094 ± 0.003**	0.013 ± 0.002	0.039 ± 0.006	0.172 ± 0.008	0.754 ± 0.007	0.218 ± 0.010
SVGP	**0.341 ± 0.001**	0.104 ± 0.037	-	0.011 ± 0.001	0.048 ± 0.001	0.326 ± 0.043	0.323 ± 0.007	-
DKLGP	0.780 ± 0.269	0.207 ± 0.128	-	0.014 ± 0.004	0.047 ± 0.004	0.279 ± 0.068	0.318 ± 0.010	-
RS^{3}GP	0.377 ± 0.004	0.057 ± 0.001	0.165 ± 0.001	0.012 ± 0.001	0.038 ± 0.003	0.354 ± 0.016	0.297 ± 0.007	0.310 ± 0.012
VRS^{3}GP	0.366 ± 0.003	0.056 ± 0.001	0.160 ± 0.002	0.011 ± 0.001	**0.035 ± 0.001**	0.347 ± 0.009	**0.291 ± 0.015**	0.295 ± 0.005

Table 6.2: Training time in hours

	SVGP	DKLGP	TSDiff	RS^{3}GP	VRS^{3}GP
SOLAR	1.90	2.41	2.75	0.27	0.45
ELECTRICITY	2.23	1.91	4.09	0.62	0.98
TRAFFIC	-	-	4.44	2.31	2.70
EXCHANGE	0.75	0.65	2.00	0.23	0.29
M4	1.68	1.94	1.96	0.86	1.02
UBERTLC	1.89	2.19	3.19	0.53	0.71
KDDCUP	2.03	1.55	2.10	0.65	0.93
WIKIPEDIA	-	-	6.17	3.52	6.10

[118], uses 500 inducing points with the RBF kernel; and DKLGP [289] which augments SVGP with a deep kernel using a 2-layer NN with 64 units. See [262, App. B.4] for further details.

Results. Table 6.1 shows the results of our models VRS^{3}GP and RS^{3}GP compared to baselines. We omit results for SVGP and DKLGP on Traffic and Wikipedia since inference time takes longer than 12 hours. Overall, our models achieve top scores on 2 datasets, outperforming the state-of-the-art diffusion baseline TSDiff, and provide comparable performance to deep learning baselines on the remaining 4/6 datasets. Importantly, they outperform linear baselines. They also outperform the GP baselines on 4/6 datasets. One of the datasets our model achieves the top score on, KDDCup, is the one with longest time series length, $L \sim 10000$, which suggests that this is due to its ability to adapt its context memory to long ranges. Moreover, Table 6.2 illustrates that our models VRS^{3}GP and RS^{3}GP are significantly quicker to train TSDiff, and both SVGP and DKLGP. This speed up is especially pronounced for datasets with long time series such as Solar, Electricity, Exchange, UberTLC and KDDCup, since our model is able to process a long time series in sublinear time due to parallelizability as detailed in Section 6.3. We further investigate the scalability in Appendix 6.A.5.

6.5 Conclusion and Limitations

In this work, we introduced the Random Fourier Decayed Signature Features for time series forecasting, incorporating a novel forgetting mechanism into signature features, which addresses the need to adaptively prioritize local, recent information in long time series. Our proposed model, the Recurrent Sparse Spectrum Signature Gaussian Process (RS^{3}GP), leverages variational inference and recurrent structure to efficiently transform time series data into a joint predictive distribution. We demonstrated that our approach outperforms traditional GP models and achieves comparable performance to state-of-the-art deep learning methods, while offering significantly faster training times. Limitations include reliance on the Gaussian likelihood, which may be unsuitable for non-Gaussian or heavy-tailed distributions, where asymmetric noise models are warranted, or in cases where explicit target constraints can be used to refine predictions. Additionally, while we introduced a decay mechanism for handling the trade-off between global and local information, more sophisticated forgetting mechanisms might be warranted for capturing complex dependencies in non-stationary time series. Future work could explore these contexts, and extend to the multivariate forecasting regime.

6.A Model details

6.A.1 Fractional Differencing

Fractional differencing [102] is a technique used in time series analysis to transform non-stationary data into stationary data while preserving long-term dependencies in the series. Unlike traditional differencing methods, which involve subtracting the previous value (integer differencing), fractional differencing provides a more flexible way of modelling persistence in time series by introducing a fractional order parameter. Hence, fractional differencing generalizes the concept of differencing by allowing the differencing parameter q to take non-integer values.

The general form of fractional differencing is expressed through a binomial expansion:

$$\delta^q X_t = (1-B)^q X_t = \sum_{k=0}^{\infty} \binom{q}{k} (-1)^k X_{t-k}, \tag{6.18}$$

where B is the backshift operator, $q > 0$ is the fractional differencing order parameter, and $\binom{q}{k}$ is the generalized binomial coefficient, i.e.

$$\binom{q}{k} = \frac{\Gamma(q+1)}{\Gamma(k+1)\Gamma(q-k+1)}, \tag{6.19}$$

where Γ is the Gamma function. This formula creates a weighted sum of past values, where the weights decay gradually depending on q. Note that for integer values of q, the formula collapses to standard differencing.

In our case in equation (6.11) and (6.13), we apply fractional differencing to each channel in the lifted time series it is applied to, such that the RFF maps $\varphi^{(p)}$ each have D channels, and each channel is convolved with an individual filter as in (6.18) depending on the channel-wise fractional differencing parameter. In practice, we limit the summation in (6.18) to a finite window $W \in \mathbb{Z}_+$, since the sequence decays to zero fast this does not lose much.

6.A.2 Algorithms

Additionally to the notation defined in Section 1.3.1, we extend the possible array operations. Recall that for arrays 1-based indexing is used. Elements outside the bounds of an array are treated as zeros as opposed to circular wrapping. Let A and B be k-fold arrays with shape $(n_1 \times \cdots \times n_k)$, and let $i_j \in [n_j]$ for $j \in [k]$. We define the following channelwise array operations:

(i) Let $\boldsymbol{\lambda} \in \mathbb{R}^{n_k}$. The channelwise geometric scan along axis j:

$$A[\ldots, :, \boxplus^{\boldsymbol{\lambda}}, :, \ldots][\ldots, i_{j-1}, i_j, , i_{j+1} \ldots] := \sum_{\kappa=1}^{i_j} \lambda_{i_k}^{i_j-\kappa} A[\ldots, i_{j-1}, \kappa, i_{j+1}, \ldots].$$

(ii) Let $\mathbf{q} \in \mathbb{R}^{n_k}$ and $W \in \mathbb{Z}_+$. The channelwise fractional difference along axis j:

$$A[\ldots, :, \boxminus_W^{\mathbf{q}}, :, \ldots][\ldots, i_{j-1}, i_j, , i_{j+1} \ldots] := \sum_{\kappa=0}^{W-1} \binom{q_{i_k}}{\kappa} A[\ldots, i_{j-1}, i_j - \kappa, i_{j+1}, \ldots],$$

where $\binom{q_{i_k}}{\kappa}$ is as in (6.19).

Algorithm 11 Computing the RFDSF map.

Input: Time series $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_L) \in \mathsf{Seq}(\mathbb{R}^d)$, spectral measure Λ, truncation $M \in \mathbb{Z}_+$, RFF dimension $D \in \mathbb{Z}_+$, frac. diff. orders $\mathbf{q} \in \mathbb{R}^D$, frac.diff. window $W \in \mathbb{Z}_+$, channelwise decay factors $\boldsymbol{\lambda} \in \mathbb{R}^d$
Sample independent RFF frequencies $\Omega^{(1)}, \ldots, \Omega^{(M)} \overset{\text{i.i.d.}}{\sim} \Lambda^D$ and phases $\mathbf{b}^{(1)}, \ldots, \mathbf{b}^{(M)} \overset{\text{i.i.d.}}{\sim} \mathcal{U}(0, 2\pi)^D$
Initialize an array U with shape $[M, L, D]$
Compute RFFs per time step $U[m, l, :] \leftarrow \cos({\Omega^{(m)}}^\top \mathbf{x}_l + \mathbf{b}^{(m)})$ for $m \in [M]$ and $l \in [L]$
Compute fractional differences $U \leftarrow U[:, \boxminus_W^{\mathbf{q}}, :]$
Accumulate into level-1 features $P_1 \leftarrow U[1, \boxplus^{\boldsymbol{\lambda}}, :]$
Initialize list $R \leftarrow [U[1, :, :]]$
for $m = 2$ **to** M **do**
  Update with next step $P' \leftarrow \boldsymbol{\lambda}^{\odot(m-1)} \odot P_{m-1}[+1, :] \odot U[m, :, :]$
  Initialize new list $R' \leftarrow [P']$
  for $p = 2$ to m **do**
    Update with current step $Q \leftarrow \frac{1}{p} R[p-1] \odot U[m, :, :]$
    Append to $R' += [Q]$
  end for
  Aggregate into level-m features $P_m \leftarrow R'[\Sigma][\boxplus^{\boldsymbol{\lambda}^{\odot m}}, :]$
  Roll list $R \leftarrow R'$
end for
Output: Arrays of RFDSF features per signature level and time step $P_1, \ldots, P_M$.

Algorithm 10 Computing the RFSF map.

Input: Time series $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_L) \in \mathsf{Seq}(\mathbb{R}^d)$, spectral measure Λ, truncation $M \in \mathbb{Z}_+$, RFF dimension $D \in \mathbb{Z}_+$, frac. diff. orders $\mathbf{q} \in \mathbb{R}^D$, frac.diff. window $W \in \mathbb{Z}_+$
Sample independent RFF frequencies $\Omega^{(1)}, \ldots, \Omega^{(M)} \overset{\text{i.i.d.}}{\sim} \Lambda^D$ and phases $\mathbf{b}^{(1)}, \ldots, \mathbf{b}^{(M)} \overset{\text{i.i.d.}}{\sim} \mathcal{U}(0, 2\pi)^D$
Initialize an array U with shape $[M, L, D]$
Compute RFFs per time step $U[m, l, :] \leftarrow \cos({\Omega^{(m)}}^\top \mathbf{x}_l + \mathbf{b}^{(m)})$ for $m \in [M]$ and $l \in [L]$
Compute fractional differences $U \leftarrow U[:, \boxminus_W^{\mathbf{q}}, :]$
Accumulate into level-1 features $P_1 \leftarrow U[1, \boxplus, :]$
Initialize list $R \leftarrow [U[1, :, :]]$
for $m = 2$ **to** M **do**
  Update with next step $P' \leftarrow P_{m-1}[+1, :] \odot U[m, :, :]$
  Initialize new list $R' \leftarrow [P']$
  for $p = 2$ to m **do**
    Update with current step $Q \leftarrow \frac{1}{p} R[p-1] \odot U[m, :, :]$
    Append to $R' += [Q]$
  end for
  Aggregate into level-m features $P_m \leftarrow R'[\Sigma][\boxplus, :]$
  Roll list $R \leftarrow R'$
end for
Output: Arrays of RFSF features per signature level and time step $P_1, \ldots, P_M$.

6.A.3 Treatment of Random Parameters

Reparameterization. First of all, in order to be able to learn the distributional parameters of the random parameters, we need to reparameterize them [144]. That is, we reparametrize random variable X, in terms of a simpler random variable, say ϵ, such that the distributional parameters of X can be captured by a continuous function. For a Gaussian distribution, this is easy to do, since for $X \sim \mathcal{N}(\mu, \sigma^2)$, we have the reparameterization $X = \sigma\epsilon + \mu$, where $\epsilon \sim \mathcal{N}(0, 1)$.

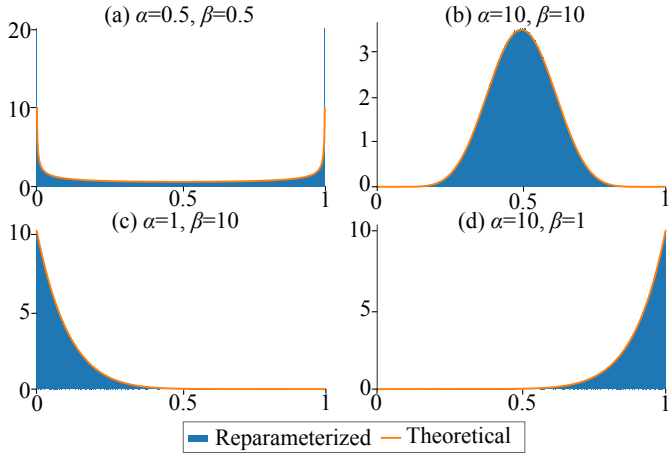

Figure 6.2: Reparameterizing the beta distribution for various shape parameters given fixed random outcomes.

For a beta-distributed random variable, this is more tricky, and it is unfeasible to do exact reparameterization. The problem is equivalent to reparameterizing a Gamma random variable, since for $X = Y_1/(Y_1 + Y_2)$ such that $Y_1 \sim \Gamma(\alpha, 1)$, $Y_2 \sim \Gamma(\beta, 1)$, it holds that $X \sim \mathcal{B}(\alpha, \beta)$. Previous work [195] combined accepted samples from the Gamma acceptance-rejection sampler with the shape augmentation trick for Gamma variables, so that the probability of acceptance is close to 1 in order to reparameterize Gamma random variables. We numerically validated this approach, and for fixed randomness, it is able to continuously transport between beta distributions with different shape parameters. The below figure shows a density plot of the reparameterization for shape augmentation parameter $B = 10$ and $n = 10^7$ samples. We see that the theoretical densities are reproduced exactly by the reparameterized distributions.

Resample or not to resample. The evaluation of the KL divergences can be done analytically, and the question becomes how to sample from the data-fit term, and how to treat the integrals with respect to the random parameters during inference. There are two approaches proposed in [57]. First, is to simply apply vanilla Monte Carlo sampling to the random hyperparameters, that is, resample them at each step optimization step. This leads to high variance and slow convergence of the ELBO, or it may not even converge within optimization budgets. The explanation given for this is that the since the variational factorizes across the "layers" of the model, it does not maintain the proper correlations needed to propagate the uncertainty. Hence, instead of resampling the random parameters during training, we retort to sampling them once at the start of training, and then keeping them fixed using the reparameterization trick throughout.

6.A.4 Fixing Pathologies in the ELBO

Next, we will focus on why the ELBO is unsuitable for capturing latent uncertainty, and relies solely on the observation noise for uncertainty calibration. Let us recall the ELBO from (6.16):

$$\mathcal{L}_{\text{ELBO}} = \underbrace{\sum_{i=1}^{N} \mathbb{E}_q\left[\log p(y_i \,|\, \mathbf{w}, \boldsymbol{\Omega}, B)\right]}_{\text{data-fit term}} - \underbrace{\mathsf{D}_{\mathsf{KL}}\left[q(\mathbf{w}) \,\|\, p(\mathbf{w})\right] - \mathsf{D}_{\mathsf{KL}}\left[q(\boldsymbol{\Omega}) \,\|\, p(\boldsymbol{\Omega})\right] - \mathsf{D}_{\mathsf{KL}}\left[q(B) \,\|\, p(B)\right]}_{\text{KL regularizers}}.$$

Clearly, the KL regularizers do not play a role in uncertainty calibration to the data, and we will focus on a datafit term $\mathbb{E}_q\left[\log p(y_i \,|\, \mathbf{w}, \boldsymbol{\Omega}, B)\right]$. Let $f_i = \mathbf{w}^\top \Phi(\mathbf{x}_i)$, and $q(f_i) = \mathcal{N}(\mu_i, \sigma_i)$ be given as in (6.17). Now, note that $p(y_i \,|\, \mathbf{w}, \boldsymbol{\Omega}, B) = \mathcal{N}(y_i \,|\, f_i, \sigma_y^2)$. Hence,

$$\mathbb{E}_q\left[\log p(y_i \,|\, \mathbf{w}, \boldsymbol{\Omega}, B)\right] = -\frac{1}{2}\log(2\pi\sigma_y^2) - \frac{1}{2\sigma_y^2}\mathbb{E}_q[y_i - f_i]^2.$$

The first term clearly plays no role in the data calibration, but serves as a penalty, while the second term matches the predictive distribution to the data. Calculating the expectation,

$$\frac{1}{2\sigma_y^2}\mathbb{E}_q\left[y_i - f_i\right]^2 = \frac{1}{2\sigma_y^2}\left((y_i - \mu_i)^2 + \sigma_i^2\right).$$

Putting these two together, we get that

$$\mathbb{E}_q\left[\log p(y_i \,|\, \mathbf{w}, \boldsymbol{\Omega}, B)\right] = -\frac{1}{2}\log(2\pi\sigma_y^2) - \frac{1}{2\sigma_y^2}\left((y_i - \mu_i)^2 + \sigma_i^2\right). \tag{6.20}$$

Hence, the data-fit is solely determined by the closeness of the predictive mean to the underlying observation weighted the inverse of the observation noise level, while the latent function uncertainty is simply penalized without playing a role in the data-fit calibration. This explains why the ELBO underestimates the latent function uncertainty and only relies on the observation noise for uncertainty calibration.

Next, we consider the PPGPR loss. With the above notation, consider

$$\mathcal{L}_{\text{PPGPR}} = \underbrace{\sum_{i=1}^{N} \log \mathbb{E}_q\left[p(y_i \,|\, \mathbf{w}, \boldsymbol{\Omega}, B)\right]}_{\text{data-fit term}} - \underbrace{\mathsf{D}_{\mathsf{KL}}\left[q(\mathbf{w}) \,\|\, p(\mathbf{w})\right] - \mathsf{D}_{\mathsf{KL}}\left[q(\boldsymbol{\Omega}) \,\|\, p(\boldsymbol{\Omega})\right] - \mathsf{D}_{\mathsf{KL}}\left[q(B) \,\|\, p(B)\right]}_{\text{KL regularizers}}.$$

Now, the data-fit term is given by the log outside of the q-expectation so that $\mathbb{E}_q[p(y_i \mid \mathbf{w}, \mathbf{\Omega}, B)] = \mathbb{E}_q[\mathcal{N}(y_i \mid f_i, \sigma_y^2)]$, which is a convolution of two Gaussians, hence we get

$$\begin{aligned} &\log \mathbb{E}_q[p(y_i \mid \mathbf{w}, \mathbf{\Omega}, B)] \\ &= \log \mathcal{N}(y_i \mid \mu_i, \sigma_f^2 + \sigma_y^2) = -\frac{1}{2}\log(2\pi(\sigma_i^2 + \sigma_y^2)) - \frac{1}{2(\sigma_i^2 + \sigma_y^2)}(y_i - \mu_i)^2. \end{aligned} \tag{6.21}$$

Contrasting this with (6.20), we see that (6.21) treats the latent function variance σ_i^2 and σ_y^2 on equal footing, hence restoring symmetry in the objective function. This is why PPGPR often leads to better predictive variances, since both the latent function uncertainty and observation noise are taken into account during uncertainty calibration, which makes use of the expressive power contained in the kernel function regarding the geometry of the data-space.

6.A.5 Scalability

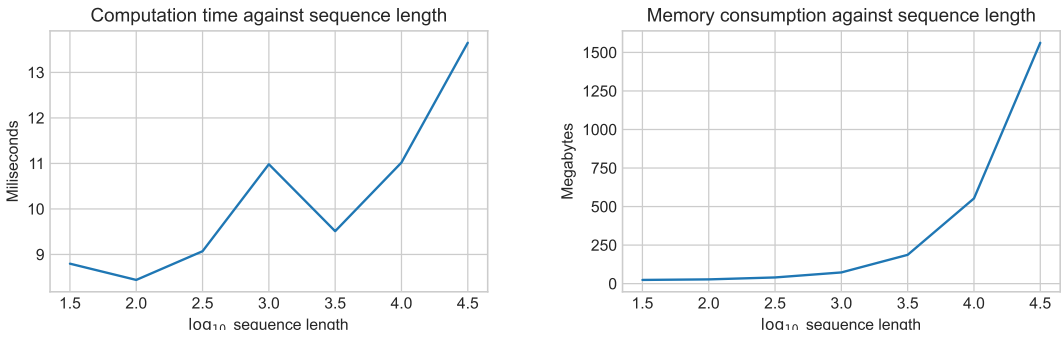

Figure 6.3: Computation time (left) and memory consumption (right) of VRS^{3}GP measured against scaling the sequence length on a logarithmic scale. The hyperparameters of the model are $D = 200$ and $M = 5$, and the input time series is univariate augmented with $l = 9$ lags.

Conclusion

Summary

The thesis explores the integration of path signatures into machine learning frameworks to address challenges in modeling sequential and structured data. Path signatures, rooted in rough path theory, serve as a powerful mathematical tool that generalizes polynomial features and captures complex dynamics in sequential data. Despite their theoretical strengths, computational challenges such as the combinatorial growth of tensor dimensions have limited their scalability, while its kernelized variation suffers from quadratic complexity in the sequence length. This thesis addresses these limitations while preserving the expressiveness of signatures.

Chapter 1 introduces path signatures by laying the necessary mathematical foundation through tensor algebras and their properties. It highlights path signatures' universality in approximating continuous functions and their robustness to reparameterization, which makes them particularly effective for sequence modelling. The connection between path signatures and classical polynomial features motivates their suitability for extracting meaningful representations from sequential data, although scalability remains a concern. Additionally, oscillatory behaviour may deteriorate performance on complex datasets analogously to how polynomial regression is often outperformed by other nonlinearities, which motivates the use of the lifted signature kernel as introduced in [146] and discussed in Section 1.2.4.

Building upon this foundation, Chapter 2 applies path signatures to Gaussian Processes (GPs), embedding signature features into covariance functions. This approach enhances GPs' ability to model temporal dependencies in sequential data within a Bayesian framework. The introduction of sparse variational inference mitigates computational bottlenecks, enabling the model to scale effectively to large datasets of high-dimensional sequence data. The empirics demonstrate the advantages of the method, particularly in time series classification tasks where classic sequence kernels underperform. Synergy with sequence modelling methods in deep learning is also demonstrated, which can further enhance performance on real-world tasks.

Motivated by the success of combining signatures with deep learning architectures in Chapter 2, Chapter 3 introduces the `Seq2Tens` framework, which bridges path signatures with deep learning architectures. By employing low-rank linear functionals, `Seq2Tens` mitigates the computational expense of signature features while preserving their expressiveness. Stacking multiple layers of low-rank transformations efficiently enables deep learning-style flexibility, making `Seq2Tens` both efficient and powerful for tasks such as time series classification, mortality prediction, and generative modeling. The framework's success underscores its ability to combine the mathematical richness of signatures with the scalability of deep learning. Further on the theoretical side, the chapter formalizes the theory behind the discretized order-1 signature features introduced in Section 1.2.6 proving universality under mild conditions. This explains the practical usefulness of the order-1 signature, that we often make use of in the other chapters.

In Chapter 4, the application of path signatures is extended to graph learning by connecting expected signatures on graphs to hypo-elliptic diffusion processes. This approach introduces a hypo-elliptic graph Laplacian that encodes both local and long-range graph structures via diffusion dynamics. By integrating these features into neural network architectures, the method effectively models hierarchical and global characteristics in graph-structured data. This development demonstrates the versatility of signature methods beyond sequential data, achieving strong performance in tasks such as node and graph-level classification. This connects to Chapter 3 in the deep learning-style iteration of layers, since we employ the same low-rank representation in the weight-space as introduced in there.

Chapter 5 tackles the scalability limitations of signature kernels by introducing Random Fourier Signature Features (`RFSF`). This novel method approximates signature kernels using random feature maps, significantly reducing computational complexity while preserving kernel properties with high probability. Through dimensionality reduction techniques, the approach allows signature methods to handle large-scale datasets efficiently both in sample size and sequence length. Experiments confirm that this retains high performance across time series tasks, establishing it as a practical tool for large-scale machine learning applications. This work can be treated as an extension of the low-rank algorithms proposed in [146], which had a similar aim of reformulating the computations from the dual space to the feature space, reducing the computational complexity while preserving the representational power of kernelization.

Finally, Chapter 6 focuses on time series forecasting and introduces the Recurrent Sparse Spectrum Signature Gaussian Process (`RS`^{3}`GP`). This model combines Random Fourier Signature Features (`RFSF`) with a novel decay-based forgetting mechanism to dynamically adjust its context length allowing it fo focus on more recent data segments. By leveraging variational inference and efficient parallelization, `RS`^{3}`GP` captures both short-term and long-term dependencies efficiently, making it well-suited for complex high-dimensional forecasting tasks. The model

achieves competitive results against traditional GPs and state-of-the-art deep learning methods, highlighting its scalability and interpretability. This chapter can be treated as an extension of Chapters 2 and 5, since we combine the finite-rank signature kernel called Random Fourier Signature Features introduced in Chapter 5 with the variational GP framework of Chapter 2.

The thesis demonstrates an evolution of path signatures from a theoretical construct to a versatile tool in machine learning. By integrating path signatures with Gaussian processes, deep learning, and kernel methods, the work highlights how the core principles of path signatures can address longstanding challenges in sequential and structured data modeling. Furthermore, the adaptation of these principles to graph learning, scalable low-rank and random feature techniques, extends the applicability of path signatures beyond their traditional scope, offering solutions to real-world problems in both temporal and graph-structured domains. This exploration bridges the gap between theoretical mathematical constructs and practical machine learning methodologies. The findings pave the way for future advancements by emphasizing the balance between computational scalability and expressiveness, making the contributions of this thesis a promising foundation for applications of signatures in scalable AI systems.

Outlook

This thesis demonstrates that path signatures provide a powerful and mathematically grounded framework for learning from sequential and structured data. However, there are avenues for further exploration. In particular, next we detail open questions and further ideas that might be interesting to pursue in the future relating to each chapter.

In Section, 2.A we show on the theoretical side that there exists a continuous modification of the Gaussian process with the unlifted truncated signature kernel as covariance function. We do this by bounding the metric entropy of the space of uniformly bounded variation paths. There are several possible extensions for this result, such as considering the case of the lifted signature kernel by e.g. the RBF kernel as static kernel, generalizing to unbounded variation paths using tools from rough path theory, and extending the result to the untruncated signature kernel to provide theoretical support for the GP approach of [166].

In Section, 3.3 we introduce LS2T layers as (order-1) signature features projected by low-rank weight tensors. Although the low-rank constraint limits the expressiveness of these features, we empirically demonstrate that stacking such layers allows to achieve state-of-the-art performance. Some algebraic results regarding compositions of such layers are given in [263, App. C], but it would be interesting to provide an explicit characterization of the functions that can be recovered by compositions of low-rank sequence-to-sequence layers, and answer the question theoretically whether the iteration of such low-rank layers can recover universality.

Another observation, is that if the components vectors of the low-rank tensor projections are initialized as Gaussians, then the resulting activations at initialization correspond to the Tensor Random Projection map (TRP) from Section 5.3.3 applied to unlifted signature features. The result in Theorem 5.9 can be easily generalized to this case to show that in the infinite-width limit the activations of the LS2T layer at initialization correspond to a feature representation of the unlifted signature kernel. Consequently, a 2-layer neural network with an LS2T layer followed by a linear projection layer, both with Gaussian initializations, corresponds in the infinite-width limit to the Gaussian process with the signature covariance function from Chapter 2. Inspired by this, in the setting when the low-rank projections are learnt, it would be interesting to characterize the Neural Tangent Kernel (NTK) [132] of LS2T layers, and establish its relation to the signature kernel, potentially sheding light on the training dynamics. Finally, the extension of signature features with a forgetting mechanism from Section 6.3.2 can be incorporated into LS2T layers to make them amenable to the learning of both short- and long-range interactions, potentially ameliorating the need for the preprocessing CNN layers employed in our experiments.

In Section 4.4, we introduce hypo-elliptic diffusions on graphs corresponding to expected signatures of random walks on nodes. We apply low-rank tensor projections similarly to the predecing section to construct scalable graph learning architectures. These are called G2TN layers, and they generalize the LS2T layers of the previous chapter to the domain of graphs. The exact same questions about LS2T layers follow through, e.g. whether compositions of such low-rank layers can recover the characteristicness property of expected signatures. Further, it can analogously be derived that in the infinite-limit the layer activations converge to a feature representation for the expected signature kernel (i.e. a kernel for probability measures on path-space by considering the inner product of expected signatures), which motivates to investigate the NTK associated to G2TN layers. Additionally, there are two main issues associated with classical GNNs: oversquashing related to structural bottlenecks on graphs [261]; and oversmoothing caused by node features converging to a constant node vector as we increase the network depth [222]. Whether these issues are apparent in the context of G2TN layers is an open question. However, we conjecture that due to the noncommutativity of the sequence features used to represent random walks on graphs, our node features might be less plagued by oversmoothing, and that graph rewiring techniques may be equally useful in this setting for ameliorating oversquashing.

In Chapter 5, we construct various scalable, random approximations for the order-1 signature kernel that converge with high probability. The extension of this construction to the approximation of higher-order signature kernels (see Section 1.2.6) is straightforward, but the theoretical results require further work, especially the key lemma in the proof of Theorem 5.3, i.e. Lemma 5.35, where the recursive step requires to consider the higher-order terms (i.e. the repetitions). Further, extension to continuous-time signature kernels including the unbounded

variation setting is an interesting future direction to consider. On the applications' side, RFSF features can also be used to construct efficient approximations to signature MMDs, which could come handy in several contexts, such as hypothesis testing for large time series datasets [44], or as an efficient loss function for the training of generative models for time series [130].

Finally, Chapter 6 constructs a parametric Gaussian process model called RS^{3}GP for efficient forecasting of long time series with adaptive context horizon, that utilizes RFSF features combined with a forgetting mechanism. However, the model formulation can be applied in more general contexts, such as sequence-to-sequence regression and system dynamics identification. In particular, we carried out preliminary experiments on battery health prediction [4] with promising results, indicating that this could be an interesting application to consider. Finally, given the results in Section 6.4, we conclude that our models are much more efficient but somewhat lacking in expressiveness compared to the state-of-the-art deep learning models for probabilistic time series forecasting, such as TSDiff. This could be enhanced by constructing a deep GP using probabilistic iterations of RS^{3}GP layers in the same spirit as done in [57].

Finally, a critical direction lies in the interpretability of models that employ path signatures. Developing methods to explain the contributions of signature features to predictions would enhance their applicability in fields such as healthcare and finance, where transparency is essential. By bridging theoretical rigour with computational scalability, this thesis establishes path signatures as a robust tool for modern machine learning. The innovations introduced here pave the way for scalable deployment of signature features in diverse large-scale real-world applications.

Bibliography

[1] Pubchem. https://pubchem.ncbi.nlm.nih.gov/.

[2] Masaki Adachi, Yannick Kuhn, Birger Horstmann, Arnulf Latz, Michael A Osborne, and David A Howey. Bayesian model selection of lithium-ion battery models via Bayesian quadrature. *IFAC-PapersOnLine*, 56(2):10521–10526, 2023.

[3] R.J. Adler and J.E. Taylor. *Random Fields and Geometry*. Springer Monographs in Mathematics. Springer New York, 2009.

[4] Antti Aitio and David A Howey. Predicting battery end of life from solar off-grid system field data using machine learning. *Joule*, 5(12):3204–3220, 2021.

[5] Maruan Al-Shedivat, Andrew G Wilson, Yunus Saatchi, Zhiting Hu, and Eric P Xing. Learning scalable deep kernels with recurrent structure. *Journal of Machine Learning Research (JMLR)*, 18(82):1–37, 2017.

[6] Alexander Alexandrov, Konstantinos Benidis, Michael Bohlke-Schneider, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Danielle C Maddix, Syama Rangapuram, David Salinas, Jasper Schulz, et al. GluonTS: Probabilistic and neural time series modeling in Python. *Journal of Machine Learning Research (JMLR)*, 21(116):1–6, 2020.

[7] Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations*, 2021.

[8] Hervé Andrès, Alexandre Boumezoued, and Benjamin Jourdain. Signature-based validation of real-world economic scenarios. *ASTIN Bulletin: The Journal of the IAA*, 54(2):410–440, 2024.

[9] Arthur Asuncion, David Newman, et al. UCI machine learning repository, 2007.

[10] Haim Avron, Michael Kapralov, Cameron Musco, Christopher Musco, Ameya Velingker, and Amir Zandieh. Random Fourier features for kernel ridge regression: Approximation

bounds and statistical guarantees. In *International Conference on Machine Learning*, pages 253–262, 2017.

[11] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[12] Francis Bach. Sharp analysis of low-rank kernel matrix approximations. In *Conference on Learning Theory*, pages 185–209, 2013.

[13] Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*, 2015.

[14] Pierre Baldi, Søren Brunak, Paolo Frasconi, Giovanni Soda, and Gianluca Pollastri. Exploiting the past and the future in protein secondary structure prediction. *Bioinformatics*, 15(11):937–946, 1999.

[15] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. *Advances in neural information processing systems*, 29, 2016.

[16] Matthias Bauer, Mark van der Wilk, and Carl Edward Rasmussen. Understanding probabilistic sparse Gaussian process approximations. In *Advances in neural information processing systems*, pages 1533–1541, 2016.

[17] Mustafa Baydogan. *Multivariate Time Series Classification Datasets*. `http://mustafabaydogan.com`, 2015. [Accessed: 2020-02-05].

[18] Mustafa Gokce Baydogan and George Runger. Learning a symbolic representation for multivariate time series classification. *Data Mining and Knowledge Discovery*, 29(2):400–422, 2015.

[19] Mustafa Gokce Baydogan and George C. Runger. Time series representation and similarity based on local autopatterns. *Data Mining and Knowledge Discovery*, 30:476–509, 2015.

[20] Alessio Benavoli, Giorgio Corani, Francesca Mangili, Marco Zaffalon, and Fabrizio Ruggeri. A Bayesian wilcoxon signed-rank test based on the Dirichlet process. In *International conference on machine learning*, pages 1026–1034, 2014.

[21] Alain Berlinet and Christine Thomas-Agnan. *Reproducing kernel Hilbert spaces in probability and statistics*. Springer Science & Business Media, 2011.

[22] Patric Bonnier, Patrick Kidger, Imanol Perez Arribas, Cristopher Salvi, and Terry Lyons. Deep signature transforms. *33rd Conference on Neural Information Processing Systems, NeurIPS*, 2019.

[23] Karsten Borgwardt, Elisabetta Ghisu, Felipe Llinares-López, Leslie O'Bray, and Bastian Rieck. Graph kernels: State-of-the-art and future challenges. *arXiv preprint arXiv:2011.03854*, 2020.

[24] Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. *Concentration Inequalities: A Nonasymptotic Theory of Independence*. Oxford University Press, 2013.

[25] Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

[26] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[27] H Buehler, B Horvath, T Lyons, I Perez, and B Wood. Generating financial markets with signatures. *Risk.net*, 2021.

[28] C Bui, N Pham, A Vo, A Tran, A Nguyen, and T Le. Time series forecasting for healthcare diagnosis and prognostics with the focus on cardiovascular diseases. In *International Conference on the Development of Biomedical Engineering in Vietnam (BME)*, pages 809–818. Springer, 2018.

[29] Thang D. Bui, Josiah Yan, and Richard E. Turner. A Unifying Framework for Gaussian Process Pseudo-Point Approximations using Power Expectation Propagation. *Journal of Machine Learning Research*, 18(104):1–72, 2017.

[30] Luigi Carratino, Alessandro Rudi, and Lorenzo Rosasco. Learning with SGD and random features. In *Advances in Neural Information Processing Systems*, pages 10192–10203, 2018.

[31] J Douglas Carroll and Jih-Jie Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of "Eckart-young" decomposition. *Psychometrika*, 35(3):283–319, 1970.

[32] Thomas Cass, Terry Lyons, and Xingcheng Xu. Weighted signature kernels. *The Annals of Applied Probability*, 34(1A):585–626, 2024.

[33] Thomas Cass and Cristopher Salvi. Lecture notes on rough paths and applications to machine learning. *arXiv preprint arXiv:2404.06583*, 2024.

[34] Kian Ming A Chai. Variational multinomial logit Gaussian process. *Journal of Machine Learning Research*, 13(Jun):1745–1808, 2012.

[35] Linda Chamakh, Emmanuel Gobet, and Zoltán Szabó. Orlicz random Fourier features. *Journal of Machine Learning Research*, 21(145):1–37, 2020.

[36] Benjamin Paul Chamberlain, James Rowbottom, Davide Eynard, Francesco Di Giovanni, Xiaowen Dong, and Michael M. Bronstein. Beltrami flow and neural diffusion on graphs. *CoRR*, abs/2110.09443, 2021.

[37] Benjamin Paul Chamberlain, James R. Rowbottom, Maria I. Gorinova, Stefan Webb, Emanuele Rossi, and Michael M. Bronstein. GRAND: Graph neural diffusion. In *ICML*, 2021.

[38] Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. Recurrent neural networks for multivariate time series with missing values. *Scientific reports*, 8(1):6085, 2018.

[39] Dexiong Chen, Laurent Jacob, and Julien Mairal. Convolutional kernel networks for graph-structured data. In *International Conference on Machine Learning*, pages 1576–1586. PMLR, 2020.

[40] Kuo-Tsai Chen. Integration of paths—a faithful representation of paths by non-commutative formal power series. *Trans. Amer. Math. Soc.*, 89:395–407, 1958.

[41] Zhengdao Chen, Soledad Villar, Lei Chen, and Joan Bruna. On the equivalence between graph isomorphism testing and function approximation with gnns. *Advances in neural information processing systems*, 32, 2019.

[42] Ilya Chevyrev and Andrey Kormilitzin. A primer on the signature method in machine learning. *arXiv preprint arXiv:1603.03788*, 2016.

[43] Ilya Chevyrev, Vidit Nanda, and Harald Oberhauser. Persistence Paths and Signature Features in Topological Data Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2018.

[44] Ilya Chevyrev and Harald Oberhauser. Signature moments to characterize laws of stochastic processes. *Journal of Machine Learning Research*, 23(176):1–42, 2022.

[45] Kyunghyun Cho, B van Merrienboer, Caglar Gulcehre, F Bougares, H Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, 2014.

[46] Francois Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.

[47] Krzysztof Choromanski, Mark Rowland, Tamas Sarlos, Vikas Sindhwani, Richard Turner, and Adrian Weller. The geometry of random features. In *International Conference on Artificial Intelligence and Statistics*, pages 1–9, 2018.

[48] Krzysztof Choromanski, Mark Rowland, and Adrian Weller. The unreasonable effectiveness of structured random orthogonal embeddings. In *International Conference on Neural Information Processing Systems*, pages 218–227, 2017.

[49] Krzysztof Choromanski and Vikas Sindhwani. Recycling randomness with structure for sublinear time kernel expansions. In *International Conference on Machine Learning*, pages 2502–2510, 2016.

[50] Krzysztof Marcin Choromanski, Han Lin, Haoxian Chen, Arijit Sehanobish, Yuanzhe Ma, Deepali Jain, Jake Varley, Andy Zeng, Michael S Ryoo, Valerii Likhosherstov, Dmitry Kalashnikov, Vikas Sindhwani, and Adrian Weller. Hybrid random features. In *International Conference on Learning Representations*, 2022.

[51] Andrzej Cichocki, Namgil Lee, Ivan Oseledets, Anh-Huy Phan, Qibin Zhao, and Danilo P Mandic. Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions. *Foundations and Trends® in Machine Learning*, 9(4-5):249–429, 2016.

[52] Thomas Cochrane, Peter Foster, Varun Chhabra, Maud Lemercier, Terry Lyons, and Cristopher Salvi. Sk-tree: a systematic malware detection algorithm on streaming trees via the signature kernel. In *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*, pages 35–40. IEEE, 2021.

[53] Nadav Cohen, Or Sharir, and Amnon Shashua. On the expressive power of deep learning: A tensor analysis. In *Conference on learning theory*, pages 698–728, 2016.

[54] Chris Cremer, Xuechen Li, and David Duvenaud. Inference suboptimality in variational autoencoders. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1078–1086, 2018.

[55] Christa Cuchiero, Lukas Gonon, Lyudmila Grigoryeva, Juan-Pablo Ortega, and Josef Teichmann. Discrete-time signatures and randomness in reservoir computing. *IEEE Transactions on Neural Networks and Learning Systems*, 33(11):6321–6330, 2022.

[56] Felipe Cucker and Steve Smale. On the mathematical foundations of learning. *Bulletin of the American mathematical society*, 39(1):1–49, 2002.

[57] Kurt Cutajar, Edwin V. Bonilla, Pietro Michiardi, and Maurizio Filippone. Random feature expansions for deep Gaussian processes. In *International Conference on Machine Learning (ICML)*, pages 884–893, 2017.

[58] Marco Cuturi. Fast global alignment kernels. In *International Conference on Machine Learning*, pages 929–936, 2011.

[59] Marco Cuturi and Arnaud Doucet. Autoregressive Kernels For Time Series. *arXiv e-prints*, page arXiv:1101.0673, January 2011.

[60] Marco Cuturi and Arnaud Doucet. Autoregressive kernels for time series. *arXiv preprint arXiv:1101.0673*, 2011.

[61] Andreas Damianou and Neil Lawrence. Deep Gaussian processes. In *Artificial Intelligence and Statistics*, pages 207–215, 2013.

[62] Tri Dao, Christopher M De Sa, and Christopher Ré. Gaussian quadrature for kernel features. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017.

[63] Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn Keogh. The UCR time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305, 2019.

[64] Alexander G. de G. Matthews, Mark van der Wilk, Tom Nickson, Keisuke Fujii, Alexis Boukouvalas, Pablo León-Villagrá, Zoubin Ghahramani, and James Hensman. Gpflow: A Gaussian process library using TensorFlow. *Journal of Machine Learning Research*, 18:40:1–40:6, 2017.

[65] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016.

[66] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30, 2006.

[67] Persi Diaconis. Group representations in probability and statistics. *Lecture notes-monograph series*, 11:i–192, 1988.

[68] Joscha Diehl, Kurusch Ebrahimi-Fard, and Nikolas Tapia. Time-warping invariants of multidimensional time series. *Acta Applicandae Mathematicae*, 170(1):265–290, 2020.

[69] Joscha Diehl, Kurusch Ebrahimi-Fard, and Nikolas Tapia. Generalized iterated-sums signatures. *Journal of Algebra*, 632:801–824, 2023.

[70] Andreas Doerr, Christian Daniel, Martin Schiegg, Nguyen-Tuong Duy, Stefan Schaal, Marc Toussaint, and Trimpe Sebastian. Probabilistic recurrent state-space models. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1280–1289, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[71] Andreas Doerr, Christian Daniel, Martin Schiegg, Nguyen-Tuong Duy, Stefan Schaal, Marc Toussaint, and Trimpe Sebastian. Probabilistic recurrent state-space models. In *International Conference on Machine Learning (ICML)*, pages 1280–1289. PMLR, 2018.

[72] Timothy Dozat. Incorporating Nesterov Momentum into Adam. In *International Conference on Learning Representations*, 2015.

[73] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[74] Richard M Dudley. The sizes of compact subsets of Hilbert space and continuity of Gaussian processes. *Journal of Functional Analysis*, 1(3):290–330, 1967.

[75] Richard M Dudley. *Real analysis and probability*, volume 74. Cambridge University Press, 2002.

[76] Joel Dyer, Patrick Cannon, and Sebastian M Schmon. Approximate Bayesian computation with path signatures. In *The 40th Conference on Uncertainty in Artificial Intelligence*, 2024.

[77] Joel Dyer, Patrick W. Cannon, and Sebastian M. Schmon. Amortised likelihood-free inference for expensive time-series simulators with signatured ratio estimation. In *International Conference on Artificial Intelligence and Statistics*, pages 11131–11144, 2022.

[78] Stefanos Eleftheriadis, Tom Nicholson, Marc Deisenroth, and James Hensman. Identification of Gaussian process state space models. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017.

[79] Ahmed A. A. Elhag, Gabriele Corso, Hannes Stärk, and Michael M. Bronstein. Graph anisotropic diffusion. In *ICLR 2022 Workshop on Geometrical and Topological Representation Learning*, 2022.

[80] Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. A fair comparison of graph neural networks for graph classification. In *International Conference on Learning Representations*, 2019.

[81] Chang Feng, Qinghua Hu, and Shizhong Liao. Random feature mapping with signed circulant matrix projection. In *International Joint Conference on Artificial Intelligence*, page 3490–3496, 2015.

[82] Adeline Fermanian, Pierre Marion, Jean-Philippe Vert, and Gérard Biau. Framing RNN as a kernel method: A neural ODE approach. In *Advances in Neural Information Processing Systems*, pages 3121–3134, 2021.

[83] Vincent Fortuin, Dmitry Baranchuk, Gunnar Rätsch, and Stephan Mandt. GP-VAE: Deep probabilistic time series imputation. In *International Conference on Artificial Intelligence and Statistics*, pages 1651–1661. PMLR, 2020.

[84] Roger Frigola, Yutian Chen, and Carl Edward Rasmussen. Variational Gaussian process state-space models. *Advances in Neural Information Processing Systems (NeurIPS)*, 27, 2014.

[85] Roger Frigola, Fredrik Lindsten, Thomas B Schön, and Carl Edward Rasmussen. Bayesian inference and learning in Gaussian process state-space models with particle MCMC. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3156–3164. Curran Associates, Inc., 2013.

[86] Roger Frigola, Fredrik Lindsten, Thomas B Schön, and Carl Edward Rasmussen. Bayesian inference and learning in Gaussian process state-space models with particle MCMC. *Advances in Neural Information Processing Systems (NeurIPS)*, 26, 2013.

[87] Peter K Friz and Martin Hairer. *A course on rough paths*. Springer, 2020.

[88] Kenji Fukumizu, Arthur Gretton, Bernhard Schölkopf, and Bharath K. Sriperumbudur. Characteristic kernels on groups and semigroups. In *Advances in Neural Information Processing Systems*, page 473–480, 2008.

[89] Yarin Gal and Richard Turner. Improving the Gaussian process sparse spectrum approximation by representing uncertainty in frequency inputs. In *International Conference on Machine Learning (ICML)*, pages 655–664, 2015.

[90] Francesco Galuppi and Pierpaola Santarsiero. Rank and symmetries of signature tensors. *arXiv preprint arXiv:2407.20405*, 2024.

[91] Jacob Gardner, Geoff Pleiss, Kilian Q Weinberger, David Bindel, and Andrew G Wilson. GPytorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration. *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 2018.

[92] Jan Gasthaus, Konstantinos Benidis, Yuyang Wang, Syama Sundar Rangapuram, David Salinas, Valentin Flunkert, and Tim Januschowski. Probabilistic forecasting with spline quantile function rnns. In *International Conference on Artificial Intelligence and Statistics (AISTAS)*, pages 1901–1910. PMLR, 2019.

[93] Bernard Gaveau. Principe de moindre action, propagation de la chaleur et estimees sous elliptiques sur certains groupes nilpotents. *Acta Mathematica*, 139(none):95–153, January 1977.

[94] Zoubin Ghahramani. Bayesian non-parametrics and the probabilistic approach to modelling. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, 371:20110553, 02 2013.

[95] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[96] Chad Giusti and Darrick Lee. Signatures, lipschitz-free spaces, and paths of persistence diagrams. *SIAM Journal on Applied Algebra and Geometry*, 7(4):828–866, 2023.

[97] Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, 102(477):359–378, 2007.

[98] Rakshitha Wathsadini Godahewa, Christoph Bergmeir, Geoffrey I. Webb, Rob Hyndman, and Pablo Montero-Manso. Monash time series forecasting archive. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[99] AL Goldberger, LAN Amaral, L Glass, JM Hausdorff, P Ch Ivanov, RG Mark, JE Mietus, GB Moody, CK Peng, and HE Stanley. Components of a new research resource for complex physiologic signals. *PhysioBank, PhysioToolkit, and Physionet*, 2000.

[100] Friedrich Götze, Holger Sambale, and Arthur Sinulis. Concentration inequalities for polynomials in α-sub-exponential random variables. *Electronic Journal of Probability*, 26:1 – 22, 2021.

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

[102] Clive WJ Granger and Roselyne Joyeux. An introduction to long-memory time series models and fractional differencing. *Journal of time series analysis*, 1(1):15–29, 1980.

[103] Daniele Grattarola and Cesare Alippi. Graph neural networks in tensorflow and keras with spektral [application notes]. *IEEE Computational Intelligence Magazine*, 16(1):99–106, 2021.

[104] Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. Hybrid speech recognition with deep bidirectional lstm. In *2013 IEEE workshop on automatic speech recognition and understanding*, pages 273–278. IEEE, 2013.

[105] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE, 2013.

[106] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610, 2005.

[107] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773, 2012.

[108] Alexander Grigoryan. *Heat kernel and analysis on manifolds*, volume 47. American Mathematical Soc., 2009.

[109] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[110] Ben Hambly and Terry Lyons. Uniqueness for the Signature of a path of bounded variation and the reduced path group. *Ann. of Math. (2)*, 171(1):109–167, 2010.

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

[112] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[113] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[114] James M. Heather and Benjamin Chain. The sequence of sequencers: The history of sequencing dna. *Genomics*, 107(1):1–8, 2016.

[115] James Hensman, Alexander G. de G. Matthews, and Zoubin Ghahramani. Scalable variational gaussian process classification. In Guy Lebanon and S. V. N. Vishwanathan, editors, *AISTATS*, volume 38 of *JMLR Workshop and Conference Proceedings*. JMLR.org, 2015.

[116] James Hensman, Nicolas Durrande, and Arno Solin. Variational fourier features for Gaussian processes. *Journal of Machine Learning Research*, 18(151):1–52, 2018.

[117] James Hensman, Nicoló Fusi, and Neil D. Lawrence. Gaussian processes for big data. *CoRR*, abs/1309.6835, 2013.

[118] James Hensman, Nicolò Fusi, and Neil D Lawrence. Gaussian processes for big data. In *Uncertainty in Artificial Intelligence (UAI)*, 2013.

[119] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:6840–6851, 2020.

[120] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[121] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[122] Lars Hörmander. Hypoelliptic second order differential equations. *Acta Mathematica*, 119:147–171, 1967.

[123] Max Horn, Michael Moor, Christian Bock, Bastian Rieck, and Karsten Borgwardt. Set functions for time series. In *ICML*, 2020.

[124] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

[125] Ming Hou, Jiajia Tang, Jianhai Zhang, Wanzeng Kong, and Qibin Zhao. Deep multimodal multilinear fusion with high-order polynomial pooling. In *Advances in Neural Information Processing Systems*, pages 12136–12145, 2019.

[126] RJ Hyndman. *Forecasting: principles and practice*. OTexts, 2018.

[127] Alessandro Davide Ialongo, Mark Van Der Wilk, James Hensman, and Carl Edward Rasmussen. Overcoming mean-field approximations in recurrent Gaussian process models. In *International conference on machine learning (ICML)*, pages 2931–2940. PMLR, 2019.

[128] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33(4):917–963, July 2019.

[129] Zacharia Issa and Blanka Horvath. Non-parametric online market regime detection and regime clustering for multidimensional and path-dependent data structures. *arXiv preprint arXiv:2306.15835*, 2023.

[130] Zacharia Issa, Blanka Horvath, Maud Lemercier, and Cristopher Salvi. Non-adversarial training of neural sdes with signature kernel scores. In *Advances in Neural Information Processing Systems*, volume 36, pages 11102–11126, 2023.

[131] Leon Isserlis. On a formula for the product-moment coefficient of any order of a normal frequency distribution in any number of variables. *Biometrika*, 12(1/2):134–139, 1918.

[132] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

[133] Martin Jankowiak, Geoff Pleiss, and Jacob Gardner. Parametric Gaussian process regressors. In *International conference on machine learning (ICML)*, 2020.

[134] Svante Janson. *Gaussian Hilbert Spaces*. Cambridge University Press, 1997.

[135] William B Johnson, Joram Lindenstrauss, and Gideon Schechtman. Extensions of Lipschitz maps into Banach spaces. *Israel Journal of Mathematics*, 54(2):129–138, 1986.

[136] Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Samuel Harford. Multivariate lstm-fcns for time series classification. *Neural Networks*, 116:237–245, 2019.

[137] Isak Karlsson, Panagiotis Papapetrou, and Henrik Boström. Generalized random shapelet forests. *Data Min. Knowl. Discov.*, 30(5):1053–1085, September 2016.

[138] Davar Khoshnevisan. *Multiparameter processes: an introduction to random fields*. Springer Science & Business Media, 2002.

[139] Valentin Khrulkov, Oleksii Hrinchuk, and Ivan Oseledets. Generalized tensor models for recurrent neural networks. *arXiv preprint arXiv:1901.10801*, 2019.

[140] Patrick Kidger, Patric Bonnier, Imanol Perez Arribas, Cristopher Salvi, and Terry Lyons. Deep signature transforms. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 3099–3109. Curran Associates, Inc., 2019.

[141] Patrick Kidger, James Foster, Xuechen Li, Harald Oberhauser, and Terry J Lyons. Neural SDEs as infinite-dimensional GANs. In *International Conference on Machine Learning*, pages 5453–5463, 2021.

[142] 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.

[143] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[144] Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

[145] Thomas Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*, 2017.

[146] Franz J Király and Harald Oberhauser. Kernels for sequentially ordered data. *Journal of Machine Learning Research*, 20(31):1–45, 2019.

[147] Johannes Klicpera, Stefan Weissenberger, and Stephan Günnemann. Diffusion improves graph learning. *ArXiv*, abs/1911.05485, 2019.

[148] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[149] Marcel Kollovieh, Abdul Fatir Ansari, Michael Bohlke-Schneider, Jasper Zschiegner, Hao Wang, and Yuyang Bernie Wang. Predict, refine, synthesize: Self-guiding diffusion models for probabilistic time series forecasting. *Advances in Neural Information Processing Systems (NeurIPS)*, 36, 2024.

[150] Risi Kondor. Diffusion kernels on graphs and other discrete structures. In *ICML 2002*, 2002.

[151] Jean Kossaifi, Zachary C Lipton, Aran Khanna, Tommaso Furlanello, and Anima Anandkumar. Tensor regression networks. *arXiv preprint arXiv:1707.08308*, 2017.

[152] Samory Kpotufe and Bharath Sriperumbudur. Gaussian sketching yields a JL lemma in RKHS. In *International Conference on Artificial Intelligence and Statistics*, pages 3928–3937, 2020.

[153] Arun Kumar Kuchibhotla and Abhishek Chakrabortty. Moving beyond sub-Gaussianity in high-dimensional statistics: Applications in covariance estimation and linear regression. *Information and Inference: A Journal of the IMA*, 11(4):1389–1456, 2022.

[154] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pages 95–104, 2018.

[155] Serge Lang. *Algebra*. Springer, 2002.

[156] Samuel Lanthaler and Nicholas H. Nelsen. Error bounds for learning with vector-valued random features. In *Advances in Neural Information Processing Systems*, 2023.

[157] Miguel Lázaro-Gredilla and Anibal Figueiras-Vidal. Inter-domain Gaussian processes for sparse inference using inducing features. In *Advances in Neural Information Processing Systems*, pages 1087–1095, 2009.

[158] Miguel Lázaro-Gredilla, Joaquin Quinonero-Candela, Carl Edward Rasmussen, and Aníbal R Figueiras-Vidal. Sparse spectrum Gaussian process regression. *Journal of Machine Learning Research (JMLR)*, 11:1865–1881, 2010.

[159] Quoc Le, Tamás Sarlós, Alex Smola, et al. Fastfood-approximating kernel expansions in loglinear time. In *International Conference on Machine Learning*, page 8, 2013.

[160] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[161] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[162] Darrick Lee and Robert Ghrist. Path signatures on lie groups. *arXiv preprint arXiv:2007.06633*, 2020.

[163] Darrick Lee and Harald Oberhauser. The signature kernel. *arXiv preprint arXiv:2305.04625*, 2023.

[164] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-attention graph pooling. In *International conference on machine learning*, pages 3734–3743. PMLR, 2019.

[165] Maud Lemercier, Cristopher Salvi, Thomas Cass, Edwin V. Bonilla, Theodoros Damoulas, and Terry J Lyons. SigGPDE: Scaling sparse Gaussian processes on sequential data. In *International Conference on Machine Learning*, pages 6233–6242, 2021.

[166] Maud Lemercier, Cristopher Salvi, Thomas Cass, Edwin V Bonilla, Theodoros Damoulas, and Terry J Lyons. SigGPDE: Scaling sparse Gaussian processes on sequential data. In *International conference on machine learning (ICML)*, pages 6233–6242. PMLR, 2021.

[167] Kevin Li, Max Balakirsky, and Simon Mak. Trigonometric quadrature Fourier features for scalable Gaussian process regression. In *International Conference on Artificial Intelligence and Statistics (AISTAS)*, pages 3484–3492. PMLR, 2024.

[168] Yingzhen Li and Stephan Mandt. Disentangled sequential autoencoder. *arXiv preprint arXiv:1803.02991*, 2018.

[169] Yujia Li, Richard Zemel, Marc Brockschmidt, and Daniel Tarlow. Gated graph sequence neural networks. In *Proceedings of ICLR'16*, 2016.

[170] Zhu Li, Jean-Francois Ton, Dino Oglic, and Dino Sejdinovic. Towards a unified analysis of random Fourier features. In *International Conference on Machine Learning*, pages 3905–3914, 2019.

[171] Paul Pu Liang, Zhun Liu, Yao-Hung Hubert Tsai, Qibin Zhao, Ruslan Salakhutdinov, and Louis-Philippe Morency. Learning representations from imperfect time series data via tensor rank regularization. *arXiv preprint arXiv:1907.01011*, 2019.

[172] Zhenyu Liao, Romain Couillet, and Michael W Mahoney. A random matrix analysis of random Fourier features: beyond the Gaussian kernel, a precise phase transition, and the corresponding double descent. In *Advances in Neural Information Processing Systems*, pages 13939–13950, 2020.

[173] Fanghui Liu, Xiaolin Huang, Yudong Chen, and Johan AK Suykens. Random features for kernel approximation: A survey on algorithms, theory, and beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10):7128–7148, 2021.

[174] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 9628–9639, Red Hook, NY, USA, 2018. Curran Associates Inc.

[175] Zhun Liu, Ying Shen, Varun Bharadhwaj Lakshminarasimhan, Paul Pu Liang, Amir Zadeh, and Louis-Philippe Morency. Efficient low-rank multimodal fusion with modality-specific factors. *arXiv preprint arXiv:1806.00064*, 2018.

[176] Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text classification using string kernels. *Journal of Machine Learning Research (JMLR)*, 2(Feb):419–444, 2002.

[177] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *ICLR*, 2017.

[178] Terry Lyons. Rough paths, signatures and the modelling of functions on streams. In *Proceedings of the International Congress of Mathematicians*, 2014.

[179] Terry Lyons, Michael Caruana, and Thierry Lévy. *Differential Equations Driven by Rough Paths*. Éc. Été Probab. St.-Flour. Springer-Verlag, Berlin Heidelberg, 2007.

[180] Terry Lyons and Harald Oberhauser. Sketching the order of events. *arXiv preprint arXiv:1708.09708*, 2017.

[181] Terry Lyons and Weijun Xu. Inversion of signature for paths of bounded variation. *arXiv preprint arXiv:1112.0452*, 2011.

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

[183] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The M4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1):54–74, 2020.

[184] Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. *Advances in neural information processing systems*, 32, 2019.

[185] Alexander G de G Matthews. *Scalable Gaussian process inference using variational methods*. PhD thesis, Cambridge University, 2017.

[186] Alexander G de G Matthews, James Hensman, Richard Turner, and Zoubin Ghahramani. On sparse variational methods and the Kullback-Leibler divergence between stochastic processes. *Journal of Machine Learning Research*, 51:231–239, 2016.

[187] César Lincoln C. Mattos, Zhenwen Dai, Andreas Damianou, Jeremy Forth, Guilherme A. Barreto, and Neil D. Lawrence. Recurrent Gaussian processes. In *International Conference on Learning Representations (ICLR)*, volume 3, 2016.

[188] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Grossberger. Umap: Uniform manifold approximation and projection. *The Journal of Open Source Software*, 3(29):861, 2018.

[189] Charles A Micchelli, Yuesheng Xu, and Haizhang Zhang. Universal kernels. *Journal of Machine Learning Research*, 7(Dec):2651–2667, 2006.

[190] Ron Milo, Shai S. Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri B. Chklovskii, and Uri Alon. Network motifs: simple building blocks of complex networks. *Science*, 298 5594:824–7, 2002.

[191] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5115–5124, 2017.

[192] James Morrill, Adeline Fermanian, Patrick Kidger, and Terry Lyons. A generalised signature method for multivariate time series feature extraction. *arXiv preprint arXiv:2006.00873*, 2020.

[193] Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and Leman Go Neural: Higher-Order Graph Neural Networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):4602–4609, July 2019.

[194] Francis J Murray and J v Neumann. On rings of operators. *Annals of Mathematics*, pages 116–229, 1936.

[195] Christian Naesseth, Francisco Ruiz, Scott Linderman, and David Blei. Reparameterization gradients through acceptance-rejection sampling algorithms. In *International Conference on Artificial Intelligence and Statistics (AISTAS)*, pages 489–498, 2017.

[196] Galileo Namata, Ben London, Lise Getoor, Bert Huang, and U Edu. Query-driven active surveying for collective classification. In *10th International Workshop on Mining and Learning with Graphs*, volume 8, page 1, 2012.

[197] Alfredo Nazabal, Pablo M Olmos, Zoubin Ghahramani, and Isabel Valera. Handling incomplete heterogeneous data using vaes. *arXiv preprint arXiv:1807.03653*, 2018.

[198] P. Nemenyi. *Distribution-free Multiple Comparisons*. Princeton University, 1963.

[199] Hao Ni, Lukasz Szpruch, Marc Sabate-Vidales, Baoren Xiao, Magnus Wiese, and Shujian Liao. Sig-Wasserstein GANs for time series generation. In *International Conference on AI in Finance*, pages 1–8, 2022.

[200] Giannis Nikolentzos and Michalis Vazirgiannis. Random walk graph neural networks. *Advances in Neural Information Processing Systems*, 33:16211–16222, 2020.

[201] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[202] Ivan V Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

[203] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019.

[204] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[205] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[206] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[207] Francois Petitjean, Jordi Inglada, and Pierre Gancarski. Satellite image time series analysis under time warping. *IEEE transactions on geoscience and remote sensing*, 50(8):3081–3095, 2012.

[208] Joaquin Quiñonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6(Dec):1939–1959, 2005.

[209] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, pages 1177–1184, 2007.

[210] Ali Rahimi and Benjamin Recht. Uniform approximation of functions with random bases. In *Allerton Conference on Communication, Control, and Computing*, pages 555–561, 2008.

[211] Ali Rahimi and Benjamin Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. *Advances in Neural Information Processing Systems*, pages 1313–1320, 2008.

[212] Beheshteh Rakhshan and Guillaume Rabusseau. Tensorized random projections. In *International Conference on Artificial Intelligence and Statistics*, pages 3306–3316, 2020.

[213] Ladislav Rampášek and Guy Wolf. Hierarchical graph neural nets can capture long-range interactions. In *2021 IEEE 31st International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2021.

[214] Syama Sundar Rangapuram, Matthias W Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. Deep state space models for time series forecasting. *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 2018.

[215] Sebastian Raschka, Joshua Patterson, and Corey Nolet. Machine learning in Python: Main developments and technology trends in data science, machine learning, and artificial intelligence. *Information*, 11(4), 2020.

[216] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[217] Christophe Reutenauer. Free Lie algebras. *Handbook of Algebra*, 3:887–903, 2003.

[218] Linda Preiss Rothschild and Elias M. Stein. Hypoelliptic differential operators and nilpotent groups. *Acta Mathematica*, 137:247–320, 1976.

[219] Alessandro Rudi and Lorenzo Rosasco. Generalization properties of learning with random features. In *Advances in Neural Information Processing Systems*, pages 3218–3228, 2017.

[220] W. Rudin. *Fourier Analysis on Groups*. Dover Publications, 2017.

[221] Walter Rudin. *Principles of mathematical analysis*. McGraw-hill New York, 1976.

[222] T Konstantin Rusch, Michael M Bronstein, and Siddhartha Mishra. A survey on oversmoothing in graph neural networks. *arXiv preprint arXiv:2303.10993*, 2023.

[223] R.A. Ryan. *Introduction to Tensor Products of Banach Spaces*. Springer London, 2013.

[224] Alaa Saade, Florent Krzakala, and Lenka Zdeborová. Spectral clustering of graphs with the bethe hessian. In *NIPS*, 2014.

[225] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, February 1978.

[226] Hugh Salimbeni, Vincent Dutordoir, James Hensman, and Marc Deisenroth. Deep Gaussian processes with importance-weighted variational inference. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5589–5598, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[227] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International journal of forecasting*, 36(3):1181–1191, 2020.

[228] Cristopher Salvi, Thomas Cass, James Foster, Terry Lyons, and Weixin Yang. The signature kernel is the solution of a Goursat PDE. *SIAM Journal on Mathematics of Data Science*, 3(3):873–899, 2021.

[229] Holger Sambale. Some notes on concentration for α-subexponential random variables. In *High Dimensional Probability IX: The Ethereal Volume*, pages 167–192. Springer, 2023.

[230] Z. Sasvári. *Multivariate Characteristic and Correlation Functions*. De Gruyter, 2013.

[231] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[232] Patrick Schäfer and Ulf Leser. Multivariate time series classification with weasel+muse. *ArXiv*, abs/1711.11343, 2017.

[233] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pages 593–607. Springer, 2018.

[234] Bernhard Schölkopf and Alexander Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002.

[235] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

[236] Alice C Schwarze and Mason A Porter. Motifs for processes on networks. *SIAM Journal on Applied Dynamical Systems*, 20(4):2516–2557, 2021.

[237] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

[238] Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied soft computing*, 90:106181, 2020.

[239] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[240] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

[241] Edward Snelson and Zoubin Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *Advances in neural information processing systems*, pages 1257–1264, 2006.

[242] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning (ICML)*, pages 2256–2265. PMLR, 2015.

[243] Bharath Sriperumbudur, Kenji Fukumizu, and Gert Lanckriet. On the relation between universality, characteristic kernels and RKHS embedding of measures. In *International Conference on Artificial Intelligence and Statistics*, pages 773–780, 2010.

[244] Bharath Sriperumbudur and Zoltán Szabó. Optimal rates for random Fourier features. In *Advances in Neural Information Processing Systems*, pages 1144–1152, 2015.

[245] Bharath K Sriperumbudur, Kenji Fukumizu, and Gert RG Lanckriet. Universality, characteristic kernels and rkhs embedding of measures. *Journal of Machine Learning Research*, 12(Jul):2389–2410, 2011.

[246] Bharath K Sriperumbudur and Nicholas Sterge. Approximate kernel PCA: Computational versus statistical trade-off. *The Annals of Statistics*, 50(5):2713–2736, 2022.

[247] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[248] Ingo Steinwart and Andreas Christmann. *Support Vector Machines*. Springer Science & Business Media, 2008.

[249] Robert S Strichartz. Sub-riemannian geometry. *Journal of Differential Geometry*, 24(2):221–263, 1986.

[250] Yiming Sun, Yang Guo, Joel A Tropp, and Madeleine Udell. Tensor random projection for low memory dimension reduction. *arXiv preprint arXiv:2105.00105*, 2021.

[251] Yitong Sun, Anna Gilbert, and Ambuj Tewari. But how does it work in theory? Linear SVM with random features. In *Advances in Neural Information Processing Systems*, pages 3379–3388, 2018.

[252] Martin Sundermeyer, Tamer Alkhouli, Joern Wuebker, and Hermann Ney. Translation modeling with bidirectional recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 14–25, 2014.

[253] Dougal J Sutherland and Jeff Schneider. On the error of random Fourier features. In *Conference on Uncertainty in Artificial Intelligence*, pages 862–871, 2015.

[254] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc., 2014.

[255] Zoltan Szabo and Bharath Sriperumbudur. On kernel derivative approximation with random Fourier features. In *International Conference on Artificial Intelligence and Statistics (AISTAS)*, pages 827–836, 2019.

[256] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Advances in Neural Information Processing Systems*, pages 7537–7547, 2020.

[257] Dacheng Tao, Xuelong Li, Xindong Wu, and Stephen J Maybank. General tensor discriminant analysis and gabor features for gait recognition. *IEEE transactions on pattern analysis and machine intelligence*, 29(10):1700–1715, 2007.

[258] Matthew Thorpe, Tan Minh Nguyen, Hedi Xia, Thomas Strohmer, Andrea Bertozzi, Stanley Osher, and Bao Wang. GRAND++: Graph neural diffusion with a source term. In *International Conference on Learning Representations*, 2022.

[259] Michalis Titsias. Variational learning of inducing variables in sparse Gaussian processes. In David van Dyk and Max Welling, editors, *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pages 567–574, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, 16–18 Apr 2009. PMLR.

[260] Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M. Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. In *International Conference on Learning Representations*, 2022.

[261] Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M. Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. In *International Conference on Learning Representations*, 2022.

[262] Csaba Tóth, Masaki Adachi, Michael A Osborne, and Harald Oberhauser. Learning to forget: Bayesian time series forecasting using recurrent sparse spectrum signature Gaussian processes. *arXiv preprint arXiv:2412.19727*, 2024.

[263] Csaba Tóth, Patric Bonnier, and Harald Oberhauser. Seq2Tens: An efficient representation of sequences by low-rank tensor projections. In *International Conference on Learning Representations*, 2021.

[264] Csaba Tóth, Danilo Jr Dela Cruz, and Harald Oberhauser. A user's guide to KSig: GPU-accelerated computation of the signature kernel. *arXiv preprint arXiv:2501.07145*, 2025.

[265] Csaba Tóth, Darrick Lee, Celia Hacker, and Harald Oberhauser. Capturing graphs with hypo-elliptic diffusions. In *Advances in Neural Information Processing Systems*, pages 38803–38817, 2022.

[266] Csaba Tóth and Harald Oberhauser. Bayesian learning from sequential data using Gaussian processes with signature covariances. In *International Conference on Machine Learning*, pages 9548–9560, 2020.

[267] Csaba Tóth, Harald Oberhauser, and Zoltan Szabo. Random Fourier signature features. *arXiv preprint arXiv:2311.12214*, 2023.

[268] Ledyard R Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

[269] Kerem Sinan Tuncel and Mustafa Gokce Baydogan. Autoregressive forests for multivariate time series modeling. *Pattern Recognition*, 73:202–215, 2018.

[270] Madeleine Udell and Alex Townsend. Why are big data matrices approximately low rank? *SIAM Journal on Mathematics of Data Science*, 2019.

[271] Enayat Ullah, Poorya Mianjy, Teodor Vanislavov Marinov, and Raman Arora. Streaming kernel PCA with $\tilde{O}(\sqrt{n})$ random features. In *Advances in Neural Information Processing Systems*, 2018.

[272] AW van der Vaart and J. Wellner. *Weak Convergence and Empirical Processes: With Applications to Statistics*. Springer, 1996.

[273] Mark van der Wilk, Carl Edward Rasmussen, and James Hensman. Convolutional Gaussian processes. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, 2017.

[274] N Th Varopoulos. Analysis and geometry on groups. *Cambridge Tracts in Math.*, 100, 1992.

[275] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[276] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[277] R. Vershynin. *High-Dimensional Probability: An Introduction with Applications in Data Science*. Cambridge University Press, 2018.

[278] Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets. In *ICLR*, 2016.

[279] Jonas Wacker and Maurizio Filippone. Local random feature approximations of the Gaussian kernel. *Procedia Computer Science*, 207:987–996, 2022.

[280] Jonas Wacker, Motonobu Kanagawa, and Maurizio Filippone. Improved random features for dot product kernels. *arXiv preprint arXiv:2201.08712*, 2022.

[281] Jonas Wacker, Ruben Ohana, and Maurizio Filippone. Complex-to-real sketches for tensor products with applications to the polynomial kernel. In *International Conference on Artificial Intelligence and Statistics*, pages 5181–5212, 2023.

[282] Nikil Wale, Ian A. Watson, and George Karypis. Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowledge and Information Systems*, 14(3):347–375, March 2008.

[283] Huaizhi Wang, Zhenxing Lei, Xian Zhang, Bin Zhou, and Jianchun Peng. A review of deep learning for renewable energy forecasting. *Energy Conversion and Management*, 198:111799, 2019.

[284] Z. Wang, W. Yan, and T. Oates. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 1578–1585, 2017.

[285] Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. Transformers in time series: A survey. *arXiv preprint arXiv:2202.07125*, 2022.

[286] Ruofeng Wen, Kari Torkkola, Balakrishnan Narayanaswamy, and Dhruv Madeka. A multi-horizon quantile recurrent forecaster. *arXiv preprint arXiv:1711.11053*, 2017.

[287] Christopher Williams and Matthias Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems*, pages 682–688, 2000.

[288] Andrew G Wilson, Elad Gilboa, Arye Nehorai, and John P Cunningham. Fast kernel learning for multidimensional pattern extrapolation. *Advances in Neural Information Processing Systems (NeurIPS)*, 27, 2014.

[289] Andrew G Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial intelligence and statistics (AISTATS)*, pages 370–378. PMLR, 2016.

[290] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P. Xing. Deep kernel learning. In Arthur Gretton and Christian C. Robert, editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 370–378, Cadiz, Spain, 09–11 May 2016. PMLR.

[291] Wolfgang Woess. *Random Walks on Infinite Graphs and Groups*. Cambridge Tracts in Mathematics. Cambridge University Press, Cambridge, 2000.

[292] Lingfei Wu, Ian En-Hsu Yen, Jinfeng Yi, Fangli Xu, Qi Lei, and Michael Witbrock. Random warping series: A random features method for time-series embedding. In *International Conference on Artificial Intelligence and Statistics*, pages 793–802, 2018.

[293] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

[294] Zhanghao Wu, Paras Jain, Matthew Wright, Azalia Mirhoseini, Joseph E Gonzalez, and Ion Stoica. Representing long-range context for graph neural networks with global attention. *Advances in Neural Information Processing Systems*, 34:13266–13279, 2021.

[295] Louis-Pascal A. C. Xhonneux, Meng Qu, and Jian Tang. Continuous graph neural networks. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20, pages 10432–10441. JMLR.org, July 2020.

[296] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

[297] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning*, pages 5453–5462. PMLR, 2018.

[298] Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 4805–4815, Red Hook, NY, USA, December 2018. Curran Associates Inc.

[299] Takeo Yokonuma. *Tensor Spaces and Exterior Algebra*. American Mathematical Society, 1992.

[300] Felix Xinnan X Yu, Ananda Theertha Suresh, Krzysztof M Choromanski, Daniel N Holtmann-Rice, and Sanjiv Kumar. Orthogonal random features. In *Advances in Neural Information Processing Systems*, pages 1975–1983, 2016.

[301] Amir Zadeh, Minghai Chen, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. Tensor fusion network for multimodal sentiment analysis. *arXiv preprint arXiv:1707.07250*, 2017.

[302] Huiming Zhang and Song Xi Chen. Concentration inequalities for statistical inference. *Communications in Mathematical Research*, 37(1):1–85, 2021.

[303] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[304] Qibin Zhao, Masashi Sugiyama, Longhao Yuan, and Andrzej Cichocki. Learning efficient tensor representations with ring-structured networks. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8608–8612. IEEE, 2019.