SEQ2TENS: AN EFFICIENT REPRESENTATION OF SEQUENCES BY LOW-RANK TENSOR PROJECTIONS

Csaba Toth⋆ **Patric Bonnier**⋆ **Harald Oberhauser**⋆

⋆Mathematical Institute, University of Oxford
{toth, bonnier, oberhauser}@maths.ox.ac.uk

ABSTRACT

Sequential data such as time series, video, or text can be challenging to analyse as the ordered structure gives rise to complex dependencies. At the heart of this is non-commutativity, in the sense that reordering the elements of a sequence can completely change its meaning. We use a classical mathematical object – the free algebra – to capture this non-commutativity. To address the innate computational complexity of this algebra, we use compositions of low-rank tensor projections. This yields modular and scalable building blocks that give state-of-the-art performance on standard benchmarks such as multivariate time series classification and generative models for video.

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 for reasons that will become clear; among its attractive properties are that it (i) provides a structured, parsimonious 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 for sequence data. The key ingredient to our approach is to embed the feature space of the static feature map into a larger linear space that forms an algebra (a vector space equipped with a multiplication). The product in this algebra is then used to "stitch together" the static features of the individual sequence entries in a structured way. The construction that allows to do all this is classical in mathematics, and known as the *free algebra* (over the static feature space).

Outline. Section 2 formalizes the main ideas of Seq2Tens and introduces the free algebra $\mathrm{T}(V)$ over a space V as well as the associated product, the so-called *convolution tensor product*. Section 3 shows how low rank (LR) constructions combined with sequence-to-sequence transforms allows one to efficiently use this rich algebraic structure. Section 4 applies the results of Sections 2 and 3 to build modular and scalable NN layers for sequential data. Section 5 demonstrates the flexibility and modularity of this approach on both discriminative and generative benchmarks. Section 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.

2 CAPTURING ORDER BY NON-COMMUTATIVE MULTIPLICATION

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

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

where $L \geq 1$ is some arbitrary length. Even if $\mathcal{X}$ itself is a linear space, e.g. $\mathcal{X} = \mathbb{R}$, $\mathrm{Seq}(\mathcal{X})$ is never a linear space since there is no natural addition of two sequences of different length.

Seq2Tens in a nutshell. Given any vector space V we may construct the so-called free algebra $\mathrm{T}(V)$ over V. We describe the space $\mathrm{T}(V)$ in detail below, but as for now the only thing that is important is that $\mathrm{T}(V)$ 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 $\Phi : \mathrm{Seq}(\mathcal{X}) \to \mathrm{T}(V)$ for sequences in $\mathcal{X}$ by using the algebraic structure of $\mathrm{T}(V)$: the non-commutative product on $\mathrm{T}(V)$ makes it possible to "stitch together" the individual features $\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_L) \in V \subset \mathrm{T}(V)$ of the sequence $\mathbf{x}$ in the larger space $\mathrm{T}(V)$ by multiplication in $\mathrm{T}(V)$. With this we may define the feature map $\Phi(\mathbf{x})$ for a sequences $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_L) \in \mathrm{Seq}(\mathcal{X})$ as follows

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

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

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

In a more concise form, we define Φ as

$$\Phi : \mathrm{Seq}(\mathcal{X}) \to \mathrm{T}(V), \quad \Phi(\mathbf{x}) = \prod_{i=1}^{L} \varphi(\mathbf{x}_i) \tag{2}$$

where $\prod$ denotes multiplication in $\mathrm{T}(V)$. 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 $V \subset \mathrm{T}(V)$ and we discuss the simplest such lift before Theorem 2.1 as well as other choices in Appendix 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 $\Phi(\mathbf{x}) \in \mathrm{T}(V)$ faithfully represents the sequence $\mathbf{x}$ as an element of $\mathrm{T}(V)$.

Indeed in Theorem 2.1 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* $\mathrm{Seq}(\mathcal{X})$; 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 $\mathrm{Seq}(\mathcal{X})$ of sequences of *arbitrary* (finite) length. The remainder of this Section gives more details about steps (i),(ii),(iii) for the construction of Φ.

The free algebra $\mathrm{T}(V)$ over a vector space V. Let V be a vector space. We denote by $\mathrm{T}(V)$ the set of sequences of tensors indexed by their degree m,

$$\mathrm{T}(V) := \{\mathbf{t} = (\mathbf{t}_m)_{m \geq 0} \,|\, \mathbf{t}_m \in V^{\otimes m}\} \tag{3}$$

where by convention $V^{\otimes 0} = \mathbb{R}$. For example, if $V = \mathbb{R}^d$ and $\mathbf{t} = (\mathbf{t}_m)_{m \geq 0}$ is some element of $\mathrm{T}(\mathbb{R}^d)$, then its degree $m = 1$ component is a d-dimensional vector $\mathbf{t}_1$, its degree $m = 2$ component is a $d \times d$ matrix $\mathbf{t}_2$, and its degree $m = 3$ component is a degree 3 tensor $\mathbf{t}_3$. By defining addition and scalar multiplication as

$$\mathbf{s} + \mathbf{t} := (\mathbf{s}_m + \mathbf{t}_m)_{m \geq 0}, \quad c \cdot \mathbf{t} = (c\mathbf{t}_m)_{m \geq 0} \tag{4}$$

the set $\mathrm{T}(V)$ becomes a linear space. By identifying $v \in V$ as the element $(0, v, 0, \ldots, 0) \in \mathrm{T}(V)$ we see that V is a linear subspace of $\mathrm{T}(V)$. Moreover, while V is only a linear space, $\mathrm{T}(V)$ carries a product that turns $\mathrm{T}(V)$ into an algebra. This product is the so-called *tensor convolution product*, and is defined for $\mathbf{s}, \mathbf{t} \in \mathrm{T}(V)$ as

$$\mathbf{s} \cdot \mathbf{t} := \big(\sum_{i=0}^{m} \mathbf{s}_i \otimes \mathbf{t}_{m-i}\big)_{m \geq 0} = \big(1, \mathbf{s}_1 + \mathbf{t}_1, \mathbf{s}_2 + \mathbf{s}_1 \otimes \mathbf{t}_1 + \mathbf{t}_2, \ldots\big) \in \mathrm{T}(V) \tag{5}$$

where $\otimes$ denotes the usual outer tensor product; e.g. for vectors $u = (u_i), v = (v_i) \in \mathbb{R}^d$ the outer tensor product $u \otimes v$ is the $d \times d$ matrix $(u_i v_j)_{i,j=1,\ldots,d}$. We emphasize that like the outer tensor product $\otimes$, the tensor convolution product $\cdot$ is non-commutative, i.e. $\mathbf{s} \cdot \mathbf{t} \neq \mathbf{t} \cdot \mathbf{s}$. In a mathematically precise sense, $\mathrm{T}(V)$ *is the most general algebra that contains V; it is a "free construction"*. Since $\mathrm{T}(V)$ is realized as series of tensors of increasing degree, the *free algebra* $\mathrm{T}(V)$ is also known as the *tensor algebra* in the literature. Appendix A contains background on tensors and further examples.

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 $\varphi : \mathcal{X} \to \mathrm{T}(V)$. Throughout the rest of this article we use the lift

$$\varphi(\mathbf{x}) = (1, \phi(\mathbf{x}), 0, 0 \ldots) \in \mathrm{T}(V). \tag{6}$$

We discuss other choices in Appendix B, but attractive properties of the lift 6 are that (a) the evaluation of Φ against low rank tensors becomes a simple recursive formula (Proposition 3.3, (b) it is a generalization of sequence sub-pattern matching as used in string kernels (Appendix B.3, (c) despite its simplicity it performs exceedingly well in practice (Section 4).

Extending to sequences of arbitrary length. So far we have specified in steps (i) and (ii) how the map $\Phi : \mathcal{X} \to \mathrm{T}(V)$ behaves on single observations. Step (iii) in the construction amounts to the requirement that for any two sequences $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_K), \mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_L) \in \mathrm{Seq}(V)$, their concatenation defined as $\mathbf{z} = (\mathbf{x}_1, \ldots, \mathbf{x}_K, \mathbf{y}_1, \ldots, \mathbf{y}_L) \in \mathrm{Seq}(V)$ can be understood in the feature space as (non-commutative) multiplication of their corresponding features

$$\Phi(\mathbf{z}) = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{y}). \tag{7}$$

Using equation 7 we may inductively extend the lift φ to sequences of arbitrary length by starting from sequences consisting of a single observation, which is equivalent to equation 2. By repeatedly using the definition of the tensor convolution product in equation 5 this may also be written as

$$\Phi_m(\mathbf{x}) = \sum_{1 \leq i_1 < \cdots < i_m \leq L} \mathbf{x}_{i_1} \otimes \cdots \otimes \mathbf{x}_{i_m} \in V^{\otimes m}, \;\; \Phi(\mathbf{x}) = (\Phi_m(\mathbf{x}))_{m \geq 0}, \tag{8}$$

where $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_L) \in \mathrm{Seq}(V)$ and the summation is over non-contiguous subsequences of $\mathbf{x}$.

Some intuition: generalized pattern matching. Our derivation of the feature map $\Phi(\mathbf{x}) = (1, \Phi_1(\mathbf{x}), \Phi_2(\mathbf{x}), \ldots) \in \mathrm{T}(V)$ was guided by general algebraic principles, but equation 8 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 $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_L) \in \mathrm{Seq}(V)$ 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 outer tensor product $\mathbf{x}_{i_1} \otimes \cdots \otimes \mathbf{x}_{i_m} \in V^{\otimes m}$ and summing over all possible 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 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. Readers with less mathematical background might simply take this as motivation and regard equation 8 as definition. However, the algebraic background allows to prove that Φ is universal, see Theorem 2.1 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 neural nets is their universality (Hornik, 1991). A very attractive property of Φ is that it preserves universality: if $\phi : \mathcal{X} \to V$ is universal for $\mathcal{X}$, then $\Phi : \mathrm{Seq}(X) \to \mathrm{T}(V)$ is universal for $\mathrm{Seq}(\mathcal{X})$. To make this precise, note that $V^{\otimes m}$ is a linear space and therefore any $\ell = (\ell_0, \ell_1, \ldots, \ell_M, 0, 0, \ldots) \in$

$\mathrm{T}(V)$ consisting of M tensors $\ell_m \in V^{\otimes m}$, yields a linear functional on $\mathrm{T}(V)$; 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

$$\langle \ell, \mathbf{t} \rangle := \sum_{m=0}^{M} \langle \ell_m, \mathbf{t}_m \rangle = \sum_{m=0}^{M} \sum_{i_1,\ldots,i_m \in \{1,\ldots,d\}} \ell_m^{i_1,\ldots,i_m} \mathbf{t}_m^{i_1,\ldots,i_m}. \tag{9}$$

Thus linear functionals of the feature map Φ, are real-valued functions of sequences. Theorem 2.1 below shows that any continuous function $f : \mathrm{Seq}(\mathcal{X}) \to \mathbb{R}$ can by arbitrary well approximated by a $\ell \in \mathrm{T}(V)$, $f(\mathbf{x}) \approx \langle \ell, \Phi(\mathbf{x}) \rangle$.

Theorem 2.1. *Let $\phi : \mathcal{X} \to V$ be a universal map with a lift that satisfies some mild constraints, then the following map is universal:*

$$\Phi : \mathrm{Seq}(\mathcal{X}) \to \mathrm{T}(V), \quad \mathbf{x} \mapsto \Phi(\mathbf{x}). \tag{10}$$

A detailed proof and the precise statement of Theorem 2.1 is given in Appendix B.

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 $f : \mathrm{Seq}(\mathcal{X}) \to \mathbb{R}$ 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 in the sense that even just storing $\Phi_m(\mathbf{x}) \in V^{\otimes m} \subset \mathrm{T}(V)$ 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 $\ell \in \mathrm{T}(V)$, 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 2.1 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 sequence-to-sequence 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 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 LR approximations of matrices are widely used in practice (Udell & Townsend, 2019) to address the quadratic complexity. The definition below generalizes the rank of matrices (tensors of degree $m = 2$) to tensors of any degree m.

Definition 3.1. The *rank* (also called *CP rank* (Carroll & Chang, 1970)) of a degree m tensor $\mathbf{t}_m \in V^{\otimes m}$ is the smallest number $r \geq 0$ such that one may write

$$\mathbf{t}_m = \sum_{i=0}^{r} \mathbf{v}_i^1 \otimes \cdots \otimes \mathbf{v}_i^m, \quad \mathbf{v}_i^1, \ldots, \mathbf{v}_i^m \in V. \tag{11}$$

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

Remark 3.2. For $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_L) \in \mathrm{Seq}(V)$, the rank $r_m \in \mathbb{N}$ of $\Phi_m(\mathbf{x})$ satisfies $r_m \leq \binom{L}{m}$, while the rank and degree $r, d \in \mathbb{N}$ of $\Phi(\mathbf{x})$ satisfy $r \leq \binom{L}{K}$ for $K = \lfloor \frac{L}{2} \rfloor$ and $d \leq L$.

A direct calculation shows that if ℓ is of rank-1, then $\langle \ell, \Phi(\mathbf{x}) \rangle$ can be computed very efficiently by inner product evaluations in V.

Proposition 3.3. *Let $\ell = (\ell_m)_{m \geq 0} \in \mathrm{T}(V)$ be of rank-1 and degree M. If ϕ is lifted to φ as in equation 6, then*

$$\langle \ell, \Phi(\mathbf{x}) \rangle = \sum_{m=0}^{M} \sum_{1 \leq i_1 < \cdots < i_m \leq L} \prod_{k=1}^{m} \langle \mathbf{v}_k^m, \phi(\mathbf{x}_{i_k}) \rangle \tag{12}$$

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, analogous to m-mers of strings and we make this connection precise in Appendix B.3; the proof of Proposition 3.3 is given in Appendix B.1.1. While equation 12 looks expensive, by casting it into a recursive formulation over time, it can be computed in $O(M^2 \cdot L \cdot d)$ time and $O(M^2 \cdot (L+c))$ 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 $O(M \cdot L \cdot d)$ time and $O(M \cdot (L+c))$ memory by an efficient parametrization of the rank-1 element $\ell \in \mathrm{T}(V)$. We give further details in Appendices D.2, D.3, D.4.

Low-rank Seq2Tens maps. The composition of a linear map $\mathcal{L} : \mathrm{T}(V) \to \mathbb{R}^N$ with Φ can be computed cheaply in parallel using equation 12 when $\mathcal{L}$ is specified through a collection of $N \in \mathbb{N}$ rank-1 elements $\ell^1, \ldots, \ell^N \in \mathrm{T}(V)$ such that

$$\tilde{\Phi}_{\tilde{\theta}}(\mathbf{x}_1, \ldots, \mathbf{x}_L) := \mathcal{L} \circ \Phi(\mathbf{x}_1, \ldots, \mathbf{x}_L) = (\langle \ell^j, \Phi(\mathbf{x}_1, \ldots, \mathbf{x}_L)\rangle)_{j=1}^N \in \mathbb{R}^N. \tag{13}$$

We call the resulting map $\tilde{\Phi}_{\tilde{\theta}} : \mathrm{Seq}(\mathcal{X}) \to \mathbb{R}^N$ a ***Low-rank Seq2Tens*** map of width-N and order-M, where $M \in \mathbb{N}$ is the maximal degree of $\ell^1, \ldots, \ell^N$ such that $\ell^j_i = 0$ for $i > M$. The LS2T map is parametrized by (1) the component vectors $\mathbf{v}^k_{j,m} \in V$ of the rank-1 elements $\ell^j_m = \mathbf{v}^1_{j,m} \otimes \cdots \otimes \mathbf{v}^m_{j,m}$, (2) by any parameters θ that the static feature map $\phi_\theta : \mathcal{X} \to V$ may depend on. We jointly 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 for the LS2T

$$\tilde{\mathcal{H}} = \big\{ \langle \sum_{j=1}^N \alpha_j \ell^j, \Phi(\mathbf{x}_1, \ldots, \mathbf{x}_L)\rangle \,|\, \alpha_j \in \mathbb{R} \big\} \subsetneq \mathcal{H} = \big\{ \langle \ell, \Phi(\mathbf{x}_1, \ldots, \mathbf{x}_L) \,|\, \ell \in \mathrm{T}(V) \big\} \tag{14}$$

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

Recall now that without rank restrictions on the linear functionals of Seq2Tens features, Theorem 2.1 would guarantee that any real-valued function $f : \mathrm{Seq}(\mathcal{X}) \to \mathbb{R}$ 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 propose to use LS2T transforms in a sequence-to-sequence fashion that allows us to stack such low-rank functionals, significantly recovering expressiveness.

Sequence-to-sequence transforms. We can use LS2T to build sequence-to-sequence 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 $\tilde{\Phi}_{\tilde{\theta}}$ over expanding windows of $\mathbf{x}$:

$$\mathrm{Seq}(\mathcal{X}) \to \mathrm{Seq}(\mathbb{R}^N), \quad \mathbf{x} \mapsto \big(\tilde{\Phi}_{\tilde{\theta}}(\mathbf{x}_1), \tilde{\Phi}_{\tilde{\theta}}(\mathbf{x}_1, \mathbf{x}_2), \ldots, \tilde{\Phi}_{\tilde{\theta}}(\mathbf{x}_1, \ldots, \mathbf{x}_L)\big). \tag{15}$$

The cost of computing the expanding window sequence-to-sequence transform in equation 15 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.

Stacking sequence-to-sequence transforms. Inspired by the empirical successes of stacked RNNs (Graves et al., 2013b;a; Sutskever et al., 2014), we iterate the transformation 15 D-times:

$$\mathrm{Seq}(\mathcal{X}) \to \mathrm{Seq}(\mathbb{R}^{N_1}) \to \mathrm{Seq}(\mathbb{R}^{N_2}) \to \cdots \to \mathrm{Seq}(\mathbb{R}^{N_D}). \tag{16}$$

Each of these mappings $\mathrm{Seq}(\mathbb{R}^{N_i}) \to \mathrm{Seq}(\mathbb{R}^{N_{i+1}})$ 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 $\mathrm{T}(V)$; these parameters are collectively denoted by $\tilde{\theta}_i = (\theta_i, \ell^1_i, \ldots, \ell^{N_i}_i)$. Evaluating the final sequence in $\mathrm{Seq}(\mathbb{R}^{N_D})$ at last observation-time $t = L$, we get the D-times stacked LS2T map

$$\tilde{\Phi}_{\tilde{\theta}_1, \ldots, \tilde{\theta}_D} : \mathrm{Seq}(\mathcal{X}) \to \mathbb{R}^{n_D}. \tag{17}$$

Making precise how the stacking of such low-rank sequence-to-sequence transformations approximates general functions requires more tools from algebra, and we provide a rigorous quantitative statement in Appendix 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 4.

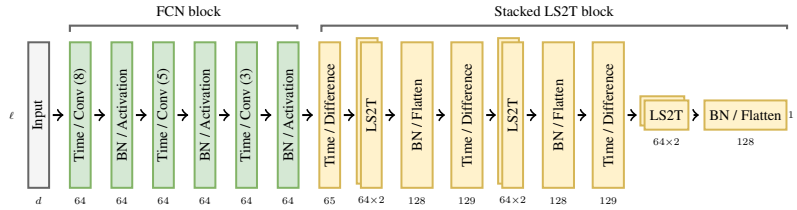

Figure 1: Depiction of the proposed models for time series classification. LS2T^3 only consists of a stacked Seq2Tens block (yellow), while FCN-LS2T^3 also precedes it with an FCN block (green).

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 2.1. 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 2 and 3 can build modular and expressive layers for sequence learning.

Neural LS2T layers. The simplest choice among many is to use as static feature map $\phi : \mathcal{X} = \mathbb{R}^d \to \mathbb{R}^M$ 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 $\mathbf{W}_j \in \mathbb{R}^{M \times d}$, $\mathbf{b}_j \in \mathbb{R}^M$. We can then lift this to a map $\varphi : \mathbb{R}^d \to \mathrm{T}(\mathbb{R}^M)$ as prescribed in equation 6. 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 sequence-to-sequence transform $\mathrm{Seq}(\mathbb{R}^d) \to \mathrm{Seq}(\mathbb{R}^M)$ 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 equation 15 is completely causal in the sense that each step of the output sequence depends only on past information. For generative models, it can behove us to make the output depend on both past and future information, see Graves et al. (2013a); Baldi et al. (1999); Li & Mandt (2018). Similarly to bidirectional RNNs and LSTMs (Schuster & Paliwal, 1997; Graves & Schmidhuber, 2005), we may achieve this by defining a bidirectional layer,

$$\tilde{\Phi}^{\mathrm{b}}_{(\tilde{\theta}_1, \tilde{\theta}_2)}(\mathbf{x}) : \mathrm{Seq}(\mathbb{R}^d) \to \mathrm{Seq}(\mathbb{R}^{N+N'}), \quad \mathbf{x} \mapsto (\tilde{\Phi}_{\tilde{\theta}_1}(\mathbf{x}_1, \ldots, \mathbf{x}_i), \tilde{\Phi}_{\tilde{\theta}_2}(\mathbf{x}_i, \ldots, \mathbf{x}_L))_{i=1}^L. \tag{18}$$

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 (Li & Mandt, 2018), machine translation (Sundermeyer et al., 2014), and speech processing (Graves et al., 2013a).

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 of the LS2T to be a state-space nonlinearity, we find that it is beneficial to capture some of the sequential information in the preprocessing layer 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 Appendix D.1 for further discussion).

In the following, we precede our stacked 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 LS2Ts (equation 12) 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 propose that the improvement in the experiments on the models containing vanilla CNN blocks are due to this complementarity.

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. 2.1 requires the Seq2Tens to be preceded by at

Table 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	LS2T^3			FCN-LS2T^3		
	$p(>)$	$p(=)$	$p(<)$	$p(>)$	$p(=)$	$p(<)$
LS2T^3	–	–	–	0.000	0.017	**0.983**
SMTS (BAYDOGAN & RUNGER, 2015A)	0.369	0.000	**0.631**	0.005	0.000	**0.995**
LPS (BAYDOGAN & RUNGER, 2015B)	0.477	0.002	**0.520**	0.001	0.000	**0.999**
MVARF (TUNCEL & BAYDOGAN, 2018)	0.021	0.168	**0.811**	0.000	0.089	**0.911**
DTW (SAKOE & CHIBA, 1978)	0.086	0.000	**0.914**	0.000	0.000	**1.000**
ARKERNEL (CUTURI & DOUCET, 2011)	0.136	0.104	**0.760**	0.000	0.044	**0.955**
GRSF (KARLSSON ET AL., 2016)	**0.851**	0.008	0.141	0.063	0.024	**0.913**
MLSTMFCN (KARIM ET AL., 2019)	**0.947**	0.038	0.014	0.342	0.190	**0.469**
MUSE (SCHÄFER & LESER, 2017)	**0.606**	0.194	0.200	0.021	0.197	**0.781**
FCN (WANG ET AL., 2017)	**0.971**	0.000	0.029	0.176	0.033	**0.791**
RESNET (WANG ET AL., 2017)	**0.962**	0.000	0.038	0.122	0.000	**0.878**

least a universal state-space nonlinearity, we expect these layers to perform best as an add-on on top of other models, which however can be quite simple, such as an FCN. The additional computation time is negligible (in fact, for FCN it allows us to reduce the number of parameters significantly), but it can yield substantial improvements. This is remarkable, since the results of the original models are already state-of-the-art on well-established (frequentist and Bayesian) benchmarks.

5.1 MULTIVARIATE TIME SERIES CLASSIFICATION

First, we consider multivariate time series classification (TSC) on a semi-standardized collection of benchmark datasets unified in Baydogan (2015). A wide range of previous publications report results on this archive, which makes it possible to compare against a wide range of baseline methods without bias in parameter settings. A subset of this archive was also considered in a recent, highly cited review paper on deep learning (DL) for TSC (Ismail Fawaz et al., 2019) from where we also borrow the best performing models as baselines. We provide further details on the experiment, baselines and datasets in Appendix E.1. As our own models, we introduce two simple architectures that utilize LS2T layers: (i) LS2T^3 stacks 3 LS2T layers of order 2 and width 64; (ii) FCN-LS2T^3 precedes the LS2T^3 block by a CNN block of 3 convolutional layers of width 64 and filter sizes of $8, 5, 3$; a downsized version of the FCN model of (Wang et al., 2017) with time-embeddings (Liu et al., 2018a) and without a final global average pooling (GAP) layer, see Fig. 1. The hyperparameters, i.e. CNN/LS2T width $N = 64$, LS2T order $M = 2$ and LS2T depth $D = 3$ were tuned using a small subset of the datasets from $N \in \{16, 64, 128\}$, $M \in \{2, 3, 4\}$ and $D \in \{1, 2, 3\}$.

Results. We trained the introduced models on each dataset 5 times while the baseline results were borrowed from their respective publications. In Appendix E.1, Figure 4 depicts the results as a box-plot of distributions and a CD diagram using the Nemenyi test (Nemenyi, 1963), while Table 7 shows the full list of means and standard deviations. Since mean-ranks based tests raise some paradoxical issues (Benavoli et al., 2016), it is customary to conduct pairwise comparisons using frequentist (Demšar, 2006) or Bayesian (Benavoli et al., 2017) hypothesis tests. We adopted the Bayesian signed-rank test from Benavoli et al. (2014), which is showcased in Table 1, while the Bayesian posteriors are visualized on Figure 5 in App. E.1. The results of the signed-rank test indicate that while LS2T^3 is better with moderate probability ($p \geq 0.6$) than 4 of the baselines, FCN-LS2T^3 performs better than all with high probability ($p \geq 0.75$), except for MLSTMFCN which is only slightly outperformed by FCN-LS2T^3. Note however that *FCN-LS2T^3 has fewer parameters than MLSTMFCN by more than* 60%, see Table 6 for a parameter comparison. Also, as MLSTMFCN is the concatenation of an FCN with an LSTM (Karim et al., 2019), the LSTM layer can bottleneck the computations for very long sequences as it scales linearly, while *LS2T layers effectively scale sublinearly in sequence length*, Appendix D.4 for a computational comparison. In conclusion, we have verified that LS2T^3 already performs well on some classification tasks, and by preceding it with an FCN, its performance is elevated to outperforming all baseline models. This observation is not surprising, as Thm. 2.1 warrants, the full strength of the LS2T layer shines when it is preceded by at least some 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 in time as previously discussed at the end of Section 4.

5.2 MORTALITY PREDICTION

We consider the 2012 PhysioNet challenge dataset (Goldberger et al., 2000) for mortality prediction, that consists of medical time series of 12,000 ICU stays over at least 48 hours. Overall, 6 static features were recorded and potentially up to 37 TS values were measured both irregularly and asynchronously for each stay with certain dimensions completely missing in some cases. The task is to predict whether a patient died during their hospital stay. This is a trickier dataset due to missingness, low signal-to-noise ratio, and unbalanced class distributions with a prevalence ratio of around 14%.

We extend the experiments conducted in Horn et al. (2020) containing very strong baselines, and under the same experimental setting we train on this dataset our model FCN-LS2T[3] and the FCN as control. We preprocess the data using the method introduced in Che et al. (2018, eq. (9)), which includes adding the observation mask and time since last measurement taken as extra TS dimensions, while we handle static features by tiling them along the time axis and also adding them as extra dims. For the models FCN-LS2T[3] and FCN, the same hyperparameters are used as in the TSC experiment with the additional introduction of a `SpatialDropout1D` layer following each CNN and LS2T layer using a dropout rate of $r = 0.1$ in order to deal with the low SNR of the dataset.

Results. Table 2 compares the results of our model, FCN-LS2T[3] with that of FCN and the baselines from Horn et al. (2020) through 3 metrics: (1) classification accuracy, (2) area under the precision-recall curve (AUPRC), (3) area under the ROC curve (AUROC). We observe that FCN-LS2T[3] significantly outperforms FCN according to all metrics. FCN-LS2T[3] is also competitive with the SOTA methods, such as the TRANSFORMER (Vaswani et al., 2017), SEFT (Horn et al., 2020), IP-NETS (Shukla & Marlin, 2019) and GRU-D (Che et al., 2018), achieving the best accuracy and taking fourth place with respect to both AUPRC and AUROC.

Table 2: Comparison of methods on mortality prediction with baselines lifted from Horn et al. (2020).

MODEL	ACCURACY	AUPRC	AUROC
FCN-LS2T[3]	$\mathbf{85.9 \pm 0.6}$	51.8 ± 0.7	85.1 ± 0.6
FCN	79.4 ± 0.6	50.2 ± 0.5	84.5 ± 0.4
GRU-D	80.0 ± 2.9	$\mathbf{53.7 \pm 0.9}$	$\mathbf{86.3 \pm 0.3}$
GRU-SIMPLE	82.2 ± 0.2	42.2 ± 0.6	80.8 ± 1.1
IP-NETS	79.4 ± 0.3	51.0 ± 0.6	$\mathit{86.0 \pm 0.2}$
PHASED-LSTM	76.8 ± 5.2	38.7 ± 1.5	79.0 ± 1.0
TRANSFORMER	$\mathit{83.7 \pm 3.5}$	$\mathit{52.8 \pm 2.2}$	$\mathbf{86.3 \pm 0.8}$
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

5.3 GENERATING SEQUENTIAL DATA

As another benchmarking experiment, we demonstrate that LS2T layers do not only provide good representations of sequences in discriminative, but also generative models. The considered problem is sequential data imputation for time series and video.

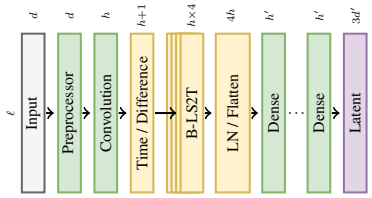

Figure 2: Encoder in GP-VAE (B-LS2T).

The GP-VAE model. In this experiment, we take as base model the recent GP-VAE (Fortuin et al., 2020), that provides state-of-the-art results for probabilistic sequential data imputation. The GP-VAE is essentially based on the HI-VAE (Nazabal et al., 2018) for handling missing data in variational autoencoders (VAEs) (Kingma & Welling, 2013) adapted to the handling of time series data by the use of a Gaussian process (GP) prior (Williams & Rasmussen, 2006) across time in the latent sequence space to capture temporal dynamics. Since the GP-VAE is a highly advanced model, its in-depth description is deferred to Appendix E.2.We extend the experiments conducted in Fortuin et al. (2020), 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 in Figure 2. 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 the naive introduction of LS2T layers.

Table 3: 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	-	-	-	-	$\mathbf{0.742 \pm 0.008}$

Results. To make the comparison, we ceteris paribus re-ran all experiments the authors originally included in their paper (Fortuin et al., 2020), which are imputation of Healing MNIST, Sprites, and Physionet 2012. The results are in Table 3, which report the same metrics as used in Fortuin et al. (2020), 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 Fortuin et al. (2020). 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 Physionet, 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 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 (Cremer et al., 2018) achieved by increasing the expressiveness of the encoder by the LS2T allowing it to pick up on long-range interactions in time. We provide further discussion in Appendix E.2.

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 (Cichocki et al., 2016), that use classical LR decompositions, such as CP (Carroll & Chang, 1970), Tucker (Tucker, 1966), tensor trains (Oseledets, 2011) and tensor rings (Zhao et al., 2019); further, CNNs have been combined with LR tensor techniques (Cohen et al., 2016; Kossaifi et al., 2017) and extended to RNNs (Khrulkov et al., 2019); Tensor Fusion Networks (Zadeh et al., 2017) and its LR variants (Liu et al., 2018b; Liang et al., 2019; Hou et al., 2019); tensor-based gait recognition (Tao et al., 2007). Our main contribution to this literature is the use of the free algebra $\mathrm{T}(V)$ 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 $\mathrm{T}(V)$, the additional algebra structure of $(\mathrm{T}(V), \cdot)$ is the main reason for the nice properties of Φ (*universality, making sequences of arbitrary length comparable, convergence in the continuous time limit*; see Appendix 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 (Lyons, 2014; Chevyrev & Kormilitzin, 2016; Graham, 2013; Bonnier et al., 2019; Toth & Oberhauser, 2020). These arise as special case of Seq2Tens (Appendix 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 free algebra $\mathrm{T}(V)$ constructed from the static feature space V. While free algebras are classical in mathematics, their use in ML seems novel and underexplored. We would like to re-emphasize that $(\mathrm{T}(V), \cdot)$ 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 equation 5. As our experiments show, the benefits of this algebraic structure are not just theoretical but can significantly elevate performance of already strong-performing models.

REFERENCES

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.

Robert Bamler and Stephan Mandt. Dynamic word embeddings. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 380–389. JMLR. org, 2017.

Mustafa Baydogan. Multivariate time series classification datasets. `http://mustafabaydogan.com`, 2015. [Accessed: 2020-06-11].

Mustafa Gokce Baydogan and George Runger. Learning a symbolic representation for multivariate time series classification. *Data Mining and Knowledge Discovery*, 29(2):400–422, 2015a.

Mustafa Gokce Baydogan and George C. Runger. Time series representation and similarity based on local autopatterns. *Data Mining and Knowledge Discovery*, 30:476–509, 2015b.

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*, pp. 1026–1034, 2014.

Alessio Benavoli, Giorgio Corani, and Francesca Mangili. Should we really use post-hoc tests based on mean-ranks? *The Journal of Machine Learning Research*, 17(1):152–161, January 2016. ISSN 1532-4435.

Alessio Benavoli, Giorgio Corani, Janez Demšar, and Marco Zaffalon. Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis. *The Journal of Machine Learning Research*, 18(1):2653–2688, 2017.

David M Blei and John D Lafferty. Dynamic topic models. In *Proceedings of the 23rd international conference on Machine learning*, pp. 113–120, 2006.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

P Bonnier, C Liu, and H Oberhauser. Adapted topologies and higher rank signatures. *arXiv preprint arXiv:2005.08897*, 2020.

Patric Bonnier, Patrick Kidger, Imanol Perez Arribas, Cristopher Salvi, and Terry Lyons. Deep signature transforms. *33rd Conference on Neural Information Processing Systems, NeurIPS*, 2019.

Wei Cao, Dong Wang, Jian Li, Hao Zhou, Lei Li, and Yitan Li. Brits: Bidirectional recurrent imputation for time series. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 31*, pp. 6775–6785. Curran Associates, Inc., 2018.

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.

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):1–12, 2018.

K. T. Chen. Iterated integrals and exponential homomorphisms. *Proc. London Math. Soc, 4, 502–512*, 1954.

K. T. Chen. Integration of paths, geometric invariants and a generalized Baker-Hausdorff formula. *Ann. of Math. (2), 65:163–178*, 1957.

K. T. Chen. Integration of paths - a faithful representation of paths by non-commutative formal power series. *Trans. Amer. Math. Soc. 89 (1958), 395–407*, 1958.

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

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.

Nadav Cohen, Or Sharir, and Amnon Shashua. On the expressive power of deep learning: A tensor analysis. In *Conference on learning theory*, pp. 698–728, 2016.

Chris Cremer, Xuechen Li, and David Duvenaud. Inference suboptimality in variational autoencoders. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 1078–1086, 2018.

N Cristianini and J Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge, 2000.

Marco Cuturi and Arnaud Doucet. Autoregressive Kernels For Time Series. *arXiv e-prints*, art. arXiv:1101.0673, Jan 2011.

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30, 2006.

J Diehl, K Ebrahimi-Fard, and N Tapia. Time-warping invariants of multidimensional time series. *arXiv preprint arXiv:1906.05823*, 2019.

Garoe Dorta, Sara Vicente, Lourdes Agapito, Neill DF Campbell, and Ivor Simpson. Structured uncertainty prediction networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5477–5485, 2018.

K. Ebrahimi-Fard and F. Patras. Cumulants, free cumulants and half-shuffles. *Proceedings of the Royal Society*, 2015.

Vincent Fortuin, Dmitry Baranchuk, Gunnar Rätsch, and Stephan Mandt. GP-VAE: Deep probabilistic time series imputation. In *International Conference on Artificial Intelligence and Statistics*, pp. 1651–1661. PMLR, 2020.

Samuel J. Gershman and Noah D. Goodman. Amortized inference in probabilistic reasoning. *Cognitive Science*, 36, 2014.

R Giles. A generalization of the strict topology. *Transactions of the American Mathematical Society*, 1971.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

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.

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

Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602 – 610, 2005.

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*, pp. 273–278. IEEE, 2013a.

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*, pp. 6645–6649. IEEE, 2013b.

I. Higgins, Loïc Matthey, A. Pal, C. Burgess, Xavier Glorot, M. Botvinick, S. Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*, 2017.

Max Horn, Michael Moor, Christian Bock, Bastian Rieck, and Karsten Borgwardt. Set functions for time series. In *ICML*, 2020.

Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4 (2):251–257, 1991.

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*, pp. 12136–12145, 2019.

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, Jul 2019. ISSN 1573-756X.

Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Samuel Harford. Multivariate lstm-fcns for time series classification. *Neural Networks*, 116:237 – 245, 2019. ISSN 0893-6080.

Isak Karlsson, Panagiotis Papapetrou, and Henrik Boström. Generalized random shapelet forests. *Data Min. Knowl. Discov.*, 30(5):1053–1085, September 2016. ISSN 1384-5810.

Nitish Shirish Keskar and Richard Socher. Improving generalization performance by switching from adam to SGD. *arXiv preprint arXiv:1712.07628*, 2017.

Valentin Khrulkov, Oleksii Hrinchuk, and Ivan Oseledets. Generalized tensor models for recurrent neural networks. *arXiv preprint arXiv:1901.10801*, 2019.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Franz J Király and Harald Oberhauser. Kernels for sequentially ordered data. *Journal of Machine Learning Research*, 2019.

Jean Kossaifi, Zachary C Lipton, Aran Khanna, Tommaso Furlanello, and Anima Anandkumar. Tensor regression networks. *arXiv preprint arXiv:1707.08308*, 2017.

Serge Lang. *Algebra*. Springer-Verlag New York, 2002.

C Leslie and R Kuang. Fast string kernels using inexact matching for protein sequences. *Journal of Machine Learning Research*, 2004.

Yingzhen Li and Stephan Mandt. Disentangled sequential autoencoder, 2018.

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.

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, pp. 9628–9639, Red Hook, NY, USA, 2018a. Curran Associates Inc.

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*, 2018b.

Terry Lyons. Rough paths, signatures and the modelling of functions on streams. *arXiv preprint arXiv:1405.4537*, 2014.

Dmytro Mishkin and Jiri Matas. All you need is a good init. *arXiv preprint arXiv:1511.06422*, 2015.

Alfredo Nazabal, Pablo M Olmos, Zoubin Ghahramani, and Isabel Valera. Handling incomplete heterogeneous data using vaes. *arXiv preprint arXiv:1807.03653*, 2018.

P. Nemenyi. *Distribution-free Multiple Comparisons*. Princeton University, 1963. URL `https://books.google.nl/books?id=nhDMtgAACAAJ`.

Ivan V Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

C Reutenauer. *Free Lie Algebras*. Clarendon press – Oxford, 1993.

W. Rudin. *Principles of Mathematical Analysis*. Cambridge University Press, 1965.

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, 1978.

Tim Sauer, James A Yorke, and Martin Casdagli. Embedology. *Journal of statistical Physics*, 65 (3-4):579–616, 1991.

Patrick Schäfer and Ulf Leser. Multivariate time series classification with weasel+muse. *ArXiv*, abs/1711.11343, 2017.

Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

Satya Narayan Shukla and Benjamin M Marlin. Interpolation-prediction networks for irregularly sampled time series. *arXiv preprint arXiv:1909.07782*, 2019.

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)*, pp. 14–25, 2014.

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 (eds.), *Advances in Neural Information Processing Systems 27*, pp. 3104–3112. Curran Associates, Inc., 2014. URL `http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf`.

Floris Takens. Detecting strange attractors in turbulence. In *Dynamical systems and turbulence, Warwick 1980*, pp. 366–381. Springer, 1981.

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.

C Toth and H Oberhauser. Bayesian learning from sequential data using gaussian processes with signature covariances. *ICML*, 2020.

Ledyard R Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3): 279–311, 1966.

Kerem Sinan Tuncel and Mustafa Gokce Baydogan. Autoregressive forests for multivariate time series modeling. *Pattern Recognition*, 73:202–215, 2018.

Madeleine Udell and Alex Townsend. Why are big data matrices approximately low rank? *SIAM Journal on Mathematics of Data Science*, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

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)*, pp. 1578–1585, 2017.

Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

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.

Cheng Zhang, Judith Bütepage, Hedvig Kjellström, and Stephan Mandt. Advances in variational inference. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):2008–2026, 2018.

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)*, pp. 8608–8612. IEEE, 2019.