A Generalised Signature Method for Time Series

James Morrill[1,2,*] **Adeline Fermanian**[3,*] **Patrick Kidger**[1,2,*] **Terry Lyons**[1,2]

[1] Mathematical Institute, University of Oxford
[2] The Alan Turing Institute, British Library
[3] Sorbonne Université
{morrill, kidger, tlyons}@maths.ox.ac.uk
adeline.fermanian@sorbonne-universite.fr

Abstract

The 'signature method' refers to a collection of feature extraction techniques for multimodal sequential data, derived from the theory of controlled differential equations. Variations exist as many authors have proposed modifications to the method, so as to improve some aspect of it. Here, we introduce a *generalised signature method* that contains these variations as special cases, and groups them conceptually into *augmentations*, *windows*, *transforms*, and *rescalings*. Within this framework we are then able to propose novel variations, and demonstrate how previously distinct options may be combined. We go on to perform an extensive empirical study on 26 datasets as to which aspects of this framework typically produce the best results. Combining the top choices produces a canonical pipeline for the generalised signature method, which demonstrates state-of-the-art accuracy on benchmark problems in multivariate time series classification.

1 Introduction

Multivariate time series offer certain challenges that are not commonly found in other areas of machine learning. The inputs may be of different length, the data may be irregularly sampled, and causality is sometimes a concern.

One approach is to construct models that directly accept such issues; for example recurrent neural networks handle varying lengths straightforwardly, whilst approaches differ on how best to have them handle irregularly sampled data (Che et al., 2018; Rubanova et al., 2019; Kidger et al., 2020a). Another option is to use feature extraction techniques, which normalise the data so that more standard techniques may then be applied; for example the shapelet transform (Ye and Keogh, 2009; Grabocka et al., 2014; Kidger et al., 2020b) or Gaussian process adapters (Li and Marlin, 2016; Futoma et al., 2017) fit into this category.

It is in this second category that the signature method belongs (Levin et al., 2013). This second option should not be thought of as somehow less desirable than the first. The signature method still exhibits a universal approximation theorem (Bonnier et al., 2019, Proposition A.6), and the results of these techniques are often more interpretable (Yang et al., 2017).

The approach taken by the signature method is to interpret a (multivariate, irregularly sampled, partially observed) time series as a discretisation of an underlying continuous path. We can then apply the *signature transform*, also known as the *path signature* or simply *signature*, which produces an infinite graded sequence of statistics characterizing the underlying path. The well-understood mathematics of rough path theory (Lyons et al., 2007; Friz and Victoir, 2010) show that the signature

[*] Equal contribution

Preprint. Under review.

acts as a natural basis for modelling functions on paths, making the signature a useful feature set for machine learning.

There is now an extensive literature on the signature method in machine learning. For example Király and Oberhauser (2019); Bonnier et al. (2019); Fermanian (2019) investigate transformations before the signature, Yang et al. (2017) discuss which regions of the data to operate on, Chevyrev and Kormilitzin (2016); Lai et al. (2017) discuss ways of rescaling the signature, and Li et al. (2017); Liao et al. (2019) consider the related *logsignature transform*. We discuss the existing literature in more detail below.

1.1 Contributions

We introduce a *generalised signature method* that contains these previously proposed variations as special cases, and in doing so are able to understand their conceptual groupings into what we term *augmentations*, *windows*, *transforms* and *rescalings*. By understanding their commonality, we are able to combine different variations, and then propose new options that fit into this framework.

We go on to examine which choices within this framework are most important to success. We perform an extensive empirical study across 26 datasets on the performance of the different options. To the best of our knowledge this is the first study of this type.

In doing so, we are then able to produce a ready-to-use canonical signature method. This represents a *domain-agnostic* starting point that may be adapted for the task at hand. We conclude by demonstrating that this canonical signature method gives state-of-the-art performance on benchmark problems in time series classification.

2 Context

2.1 Related work

A pedagogical introduction to the background theory is Lyons et al. (2007), whilst a comprehensive textbook is Friz and Victoir (2010). For a brief (four page) introduction to the signature transform, we recommend Bonnier et al. (2019, Appendix A). A similarly brief introduction to the related logsignature transform may be found in Liao et al. (2019, Section 2). For an introduction to the signature method as a whole we recommend Chevyrev and Kormilitzin (2016).

Many authors have considered transforms before the signature. Yang et al. (2017); Wu et al. (2020) consider the invisibility-reset transform, Kidger and Lyons (2020) allow for adding a basepoint to the input sequence, and Levin et al. (2013) introduce the now-standard time-augmentation transform. In each case the transformation is used to introduce sensitivity to certain kinds of perturbations. Next, Flint et al. (2016) introduce the lead-lag transform, Yang et al. (2017) introduce path disintegrations, and Bonnier et al. (2019) consider learnt transformations, in each case to incorporate additional information into the signature. Meanwhile Lyons and Oberhauser (2017) consider random projections and Liao et al. (2019) extend this to learnt projections, with the goal of dimensionality reduction.

The (log)signature transforms operate on streams of data. Windowing before applying the (log)signature transform is typical so as to achieve some locality of information. Most simply a single global window could be taken, but other options are sliding windows, expanding windows (Bonnier et al., 2019), or dyadic windows (Yang et al., 2017). Additionally a choice must be made between the signature and logsignature transforms, as must choices for the scaling of the terms in the signature (Lai et al., 2017; Chevyrev and Kormilitzin, 2016).

The differences between some of these choices made have been shown by Fermanian (2019) to significantly impact the performance of the methodology.

In terms of applications, some examples where the signature method has seen use include character recognition (Yang et al., 2016a,b; Reizenstein, 2019; Toth and Oberhauser, 2019), human action recognition (Li et al., 2017; Yang et al., 2017; Liao et al., 2019), and medicine (Arribas et al., 2018; Morrill et al., 2019; Howison et al., 2020). These applications have also prompted the creation of high performance software (Reizenstein and Graham, 2018; Kidger and Lyons, 2020).

2.2 Background theory

We recall the definition of the signature transform.

Definition 1. Let $\mathbf{x} = (x_1, \ldots, x_n)$ with each $x_i \in \mathbb{R}^d$. Let $T > 0$ and $0 = t_1 < t_2 < \cdots < t_{n-1} < t_n = T$ be arbitrary. Let $f_{\mathbf{x}} = (f_{\mathbf{x}}^1, \ldots, f_{\mathbf{x}}^d) \colon [0, T] \to \mathbb{R}^d$ be the unique continuous function such that $f_{\mathbf{x}}(t_i) = x_i$ and is affine on the intervals in between. Then the depth-N signature transform of $\mathbf{x}$ is given by

$$\mathrm{Sig}^N(\mathbf{x}) = \left(\left(\int \cdots \int_{0 < t_1 < \cdots < t_k < T} \prod_{j=1}^{k} \frac{\mathrm{d}f_{\mathbf{x}}^{i_j}}{\mathrm{d}t}(t_j) \mathrm{d}t_j \right)_{1 \leq i_1, \ldots, i_k \leq d} \right)_{1 \leq k \leq N}. \tag{1}$$

This definition is independent of the choice of T and t_i (Bonnier et al., 2019, Proposition A.7).

The signature transform has two key benefits. Given a sequence $\mathbf{x} = (x_1, \ldots, x_n)$, then the first is that the map $\mathbf{x} \mapsto \mathrm{Sig}^\infty(((1, x_1), (2, x_2), \ldots, (n, x_n)))$ uniquely determines $\mathbf{x}$ up to translations. The second benefit is that linear functions on the signature are dense in the set of continuous functions of $\mathbf{x}$, which is known as the 'universal nonlinearity' property. Precise statements of these properties may be found in Bonnier et al. (2019, Appendix A).

In comparison, the related logsignature transform (Liao et al., 2019) trades the universal nonlinearity property for a more compact representation of its data.

3 Method

Definition 2. Let $\mathcal{X}$ be a set. We denote the space of sequences over $\mathcal{X}$ as

$$\mathcal{S}(\mathcal{X}) = \{(x_1, \ldots, x_n) \,|\, x_i \in \mathcal{X}, n \in \mathbb{N}\}.$$

Given $\mathbf{x} = (x_1, \ldots, x_n) \in \mathcal{S}(\mathcal{X})$, we let $\mathrm{Length}(\mathbf{x}) = n$.

We assume that we observe some collection of sequences $\mathbf{x} \in \mathcal{S}(\mathbb{R}^d)$. If timestamps are included then these are an additional (increasing) sequence $\mathbf{t} \in \mathcal{S}(\mathbb{R})$ such that $\mathrm{Length}(\mathbf{x}) = \mathrm{Length}(\mathbf{t})$.

We begin by presenting each piece of the generalised method individually, before putting them together.

3.1 Augmentations

The first step is to transform an initial sequence $\mathbf{x} \in \mathcal{S}(\mathbb{R}^d)$ into one or several new sequences, which inspired by Bonnier et al. (2019) we call an augmentation.

For some $e, p \in \mathbb{N}$, we define an *augmentation* as a map

$$\phi \colon \mathcal{S}(\mathbb{R}^d) \to \mathcal{S}(\mathbb{R}^e)^p.$$

The widely used lead-lag transformation Flint et al. (2016); Chevyrev and Kormilitzin (2016); Yang et al. (2017) is an example of what we term an augmentation. Utilisation of the lead-lag augmentation enables us to capture the quadratic variation of a process, and takes the form

$$\phi(\mathbf{x}) = ((x_1, x_1), (x_2, x_1), (x_2, x_2), \ldots, (x_n, x_n)) \in \mathcal{S}(\mathbb{R}^{2d}).$$

Another important example is given by the stream preserving neural networks of Bonnier et al. (2019), which take $p = 1$ and ϕ some neural network, typically either convolutional or recurrent.

The time augmentation (Levin et al., 2013), invisibility-reset (Yang et al., 2017) and basepoint (Kidger and Lyons, 2020) operations are augmentations that are designed to add sensitivity to specific perturbations, and may often profitably be considered composed with other augmentations. The time augmentation adds sensitivity to parametrisation, whilst the basepoint and invisibility-reset augmentations add sensitivity to translations.

Multi-headed stream-preserving augmentation Let $\phi^1, \dots, \phi^p$ be stream preserving neural networks in the sense of Bonnier et al. (2019). Then inspired by the path disintegrations of Yang et al. (2017), we define a multi-headed stream-preserving augmentation (MHSP) as

$$\phi(\mathbf{x}) = (\phi^1(\mathbf{x}), \dots, \phi^p(\mathbf{x})) \in \mathcal{S}(\mathbb{R}^e)^p.$$

This simple extension gives a way for multiple groups of channels to interact, selected in a data-dependent way.

There are many pre-signature operations which have been proposed, and which we categorise as augmentations. See Appendix A for full details on each of them.

3.2 Windows

The second step is to choose a windowing operation, so as to select on which subsequences the (log)signature will be computed. This is analogous to rectangular window functions as might be used with a short time Fourier transform, and generalises the lifts of Bonnier et al. (2019).

We define a window to be a map

$$W = (W^1, \dots, W^q) \colon \mathcal{S}(\mathbb{R}^e) \to \mathcal{S}(\mathcal{S}(\mathbb{R}^e))^q,$$

for some $q \in \mathbb{N}$. The simplest possible window is the global window, defined by

$$W(\mathbf{x}) = (\mathbf{x}), \tag{2}$$

with $q = 1$, and which outputs the path itself. To get finer-scale information, we consider three other types of windows: sliding, expanding and hierarchical dyadic windows.

For $\mathbf{x} = (x_1, \dots, x_n) \in \mathcal{S}(\mathbb{R}^e)$ and $1 \leq i \leq j \leq n$, let $\mathbf{x}_{i,j} = (x_i, \dots, x_j) \in \mathcal{S}(\mathbb{R}^e)$ be a subsequence of $\mathbf{x}$. Then, a sliding window of length ℓ and step l is defined by

$$W(\mathbf{x}) = (\mathbf{x}_{1,\ell}, \mathbf{x}_{l+1,l+\ell}, \mathbf{x}_{2l+1,2l+\ell}, \dots), \tag{3}$$

and an expanding window of initial length ℓ and step l by

$$W(\mathbf{x}) = (\mathbf{x}_{1,\ell}, \mathbf{x}_{1,l+\ell}, \mathbf{x}_{1,2l+\ell}, \dots). \tag{4}$$

The expanding window produces subsequences of increasing length, and is analogous to the history processes of stochastic analysis. Each subsequence encodes local information only with respect to the preceding and succeeding subsequences; we will see in our experiments that despite this it performs surprisingly well. Both of these examples used $q = 1$.

Finally we consider a hierarchical dyadic window, analogous to the temporal disintegration of Yang et al. (2016b, 2017), which captures information at different scales. If the other window functions are analogous to the short time Fourier transform, then hierarchical dyadic windows are analogous to wavelets. Let $q \geq 1$ be fixed. Then for $i \in \{1, \dots, q\}$, let W^i be the sliding window of length and step both equal[2] to $n2^{-(i-1)}$. Then, the hierarchical dyadic window is defined as $W = (W^1, \dots, W^q)$. The larger the value of q, the finer scale on which the information is extracted.

3.3 The signature and logsignature transforms

Central to the signature methodology is of course the signature transform itself. Two choices must be made; whether to use the signature transform or the logsignature transform, and what depth to calculate the (log)signature transform to. This depth is the N in equation (1).

Both transforms produce essentially the same information, but represent it in different ways. A priori it is not clear which is more advantageous for machine learning. On the other hand, increasing the depth always produces more information, however it also introduces more feaures, and thus it must be chosen through a classical bias-variance trade-off.

The logsignature transform has multiple possible representations; common choices are a Hall basis, the Lyndon basis, or the non-Hall Lyndon word derived basis of Kidger and Lyons (2020). In practice it is only this latter one that we will consider, as it is the basis which may be computed most efficiently (Kidger and Lyons, 2020). For tree based models, the choice of basis is in principle an important one, and not one that we will explore here.

[2]For simplicity of presentation we assume that 2^{q-1} divides n but in practice some rounding may be required.

3.4 Rescaling

The signature transform produces a sequence of tensors, indexed by $k \in \{1, \dots, N\}$. The k-th term is of size $\mathcal{O}(1/k!)$, as it is computed by an integral over a k-dimensional simplex (Bonnier et al., 2019, Proposition A.5). It is typical that rescaling these terms to be $\mathcal{O}(1)$ will aid subsequent learning procedures. One option is to simply multiply the k-th term by $k!$, which we call post-signature scaling.

Pre-signature scaling However, it is possible that the previous option may suffer from numerical stability issues. Thus we also explore the performance of an option which may alleviate this, which is to multiply the input $\mathbf{x}$ by some scaling factor $\alpha \in \mathbb{R}$. Then the k-th term will be of size $\mathcal{O}(\alpha^k/k!)$, and so by taking $\alpha = (N!)^{1/N}$ the N-th term in the signature will be $\mathcal{O}(1)$; the trade-off is that Stirling's approximation then shows that the $N/2$-th term will be of size $\mathcal{O}(2^{N/2})$.

3.5 Putting the pieces together

Let $\phi = (\phi^1, \dots, \phi^p) \colon \mathcal{S}(\mathbb{R}^d) \to \mathcal{S}(\mathbb{R}^e)^p$ and $W = (W^1, \dots, W^q) \colon \mathcal{S}(\mathbb{R}^e) \to \mathcal{S}(\mathcal{S}(\mathbb{R}^e))^q$ be the augmentation and windowing maps. For each W^i let $W^i(\mathbf{x}) = (W^{i,j}(\mathbf{x}))_j$, so that each $W^{i,j}(\mathbf{x}) \in \mathcal{S}(\mathbb{R}^e)$.

Let S^N represent either the signature transform of depth N or the logsignature transform of depth N. Let ρ_{pre} represent either the pre-signature scaling by α, or the identity. Let ρ_{post} represent either the post-signature scaling by $k!$ or the identity.

Then given an input $\mathbf{x} \in \mathcal{S}(\mathbb{R}^d)$, the general framework for extracting signature features is given by the collection of

$$\mathbf{y}_{i,j,k} = (\rho_{\text{post}} \circ S^N \circ \rho_{\text{pre}} \circ W^{i,j} \circ \phi^k)(\mathbf{x}) \tag{5}$$

over all i, j, k. We refer to the procedure of computing $\mathbf{x} \mapsto (\mathbf{y}_{i,j,k})_{i,j,k}$ as the *generalised signature method*. Note that the range of j may depend on both i and $\text{Length}(\mathbf{x})$.

The collection $(\mathbf{y}_{i,j,k})_{i,j,k}$ may then be fed into any later machine learning algorithm, depending on the application. Note that the window choice defines the sequential nature of this collection. If the window is global, then (5) yields a collection $(\mathbf{y}_k)_k$ which does not have any ordering, whereas for other choices the collection $(\mathbf{y}_{i,j,k})_{i,j,k}$ forms a new sequence indexed by j. This sequential aspect may be used with for example recurrent neural networks, or ignored by stacking all coefficients together.

Applications to the general case of irregularly sampled, partially observed, multivariate time series will depend on the applicability of the chosen augmentation to such time series. Windowing is usually still a straightforward matter for such time series, whilst the (log)signature of such irregular time series may be computed in the same way as with regular ones (Kidger and Lyons, 2020).

4 Empirical study

4.1 Methodology

We perform a first-of-its-kind empirical study across 26 datasets to determine the most important aspects of this framework.

Datasets The datasets used are the Human Activities and Postural Transitions dataset provided by Reyes-Ortiz et al. (2016), the Speech Commands dataset provided by Warden (2018), and 24 datasets from the UEA time series classification archive, provided by Bagnall et al. (2018). (A few datasets from the UEA archive were excluded due to their high number of channels resulting in too large a computational burden; recall that the work required for the signature method scales as $\mathcal{O}(d^N)$, where d is the input channels and N is the depth of the (log)signature.)

Baseline We begin by defining a single baseline procedure, representing a simple and straightforward collection of choices for the generalised signature method. This baseline is to take the augmentation ϕ as appending time:

$$\phi(\mathbf{x}) = ((1, x_1), \dots (n, x_n)),$$

have the window W be the global window as in equation (2), have the transform be a signature transform of depth 3, and to use pre-signature scaling of the path. This means that the input features are the collection

$$\left(\mathrm{Sig}^3 \circ \rho_{\mathrm{pre}} \circ \phi\right)(\mathbf{x}).$$

Individual variations With respect to this baseline procedure, we then consider varying groups of options. Each such variation defines a particular form of the generalised signature method as in equation (5). Example variations are to switch to using a logsignature transform of depth 5, or to use a sliding window instead of a global window. We discuss the precise variations below.

Models On top of every variation, we then consider four different models: logistic regression, random forest, Gated Recurrent Unit (GRU) (Cho et al., 2014), and a residual Convolutional Neural Network (CNN) (He et al., 2015). We test nearly every combination of dataset, variation of the generalised signature method, and model. Different datasets and variations produce different numbers of features $\mathbf{y}_{i,j,k}$, so to reduce the computational burden we don't test those cases for which the number of features is greater than 10^5. Of the 9984 total combinations of dataset, variation, and model, this leaves out 1415 combinations. See Appendix B.2 for a break down of the omitted combinations by different cases.

Analysis We define the performance of a variation on a dataset as the best performance across the four models considered, to reflect the fact that different models are better suited for different problems. We then follow the methodology of Demšar (2006); Benavoli et al. (2016); Ismail Fawaz et al. (2019) to compare the variations across the multiple datasets. We first perform a Friedman test to reject the null hypothesis that all methods are equivalent. Then, we perform pairwise Wilcoxon signed-rank tests and use critical difference plots to visualize the performance of each signature method. A thick line indicates that the Wilcoxon test is not rejected at significance threshold of 5%, subject to Holm's alpha correction.

We refer the reader to Appendix B for further details on the methodology, such as precise architectural choices, learning rates, and so on.

4.2 Results

Due to the large number of variations and datasets considered, we present only the critical difference plots in the main paper. See Appendix C for all the tables of the underlying numerical values.

Augmentations We split the augmentations into two categories. The first category consists of those augmentations which remove the signature's invariance to translation (basepoint augmentation, invisibility-reset augmentation) or reparameterisation (time augmentation). See Figure 1.

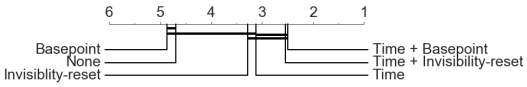

Figure 1: Performance of invariance-removing augmentations.

We see that augmenting with time, and either basepoint or invisibility-reset, are both typically important. This is expected; in general a problem need not be invariant to either translation or reparametersiation.

The second category consists of those augmentations which either seek to reduce dimensionality or introduce additional information. See Figure 2.

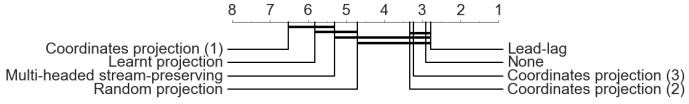

Figure 2: Performance of other augmentations.

Table 1: Average ranks for different augmentations by dataset characteristics. Lower is better.

Characteristic	Augmentation							
	None	Lead-lag	Coordinates projection (1)	(2)	(3)	Random projection	Learnt projection	MHSP
Data type								
EEG	4.88	4.83	6.50	3.13	5.67	4.38	**2.75**	**2.75**
HAR	2.25	**1.78**	7.20	3.50	2.90	4.75	6.50	6.50
MOTION	2.63	**1.75**	7.00	4.50	2.13	5.00	7.33	5.00
OTHER	2.88	3.92	5.44	**2.63**	3.29	4.69	6.00	5.21
Series length								
$<$50	3.20	**2.20**	7.40	3.20	2.70	5.10	7.00	5.20
50-100	2.20	**1.33**	6.00	4.10	2.75	6.20	4.80	5.10
100-500	**2.28**	2.57	7.28	3.50	2.63	4.17	6.63	5.33
$>$500	4.00	4.00	5.28	**2.64**	4.57	4.07	4.40	5.60
Dimension d								
2	4.67	3.5	6.33	4.33	4.0	**2.83**	6.67	3.67
3-5	2.5	**2.21**	6.36	3.43	3.14	4.64	6.67	6.83
6-8	3.25	**2.5**	6.94	3.0	3.56	5.0	5.29	6.0
$>$8	**2.25**	3.75	6.31	3.19	2.5	5.19	5.29	4.19

We see that most augmentations actually do not help matters, except for lead-lag which usually represents a good choice. We posit that the best augmentation is likely to be dataset dependent, so we additionally break this down by dataset characteristics in Table 1.

Here we indeed see that there is generally a better choice than doing nothing at all, but that this better choice is dependent on some characteristic of the dataset. For example on long or high-dimensional datasets, coordinate projections often perform well, whilst multi-headed stream preserving transformations do substantially better on EEG datasets. Lead-lag remains a strong choice in many cases.

Windows We consider the possibility of global, sliding, expanding, and dyadic windows. The results are shown in Figure 3.

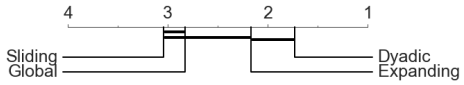

Figure 3: Performance of different windows.

We see that the dyadic window is significantly better than sliding and global windows. The poor performance of sliding windows is a little surprising, but tallies with the observations of Fermanian (2019). This is an important finding, as global and sliding windows tend to be commonly used with signature methods.

Signature versus logsignature transforms We consider the signature and logsignature transforms with depths ranging from 1 to 6. As higher depths always produce more information, we define the performance of the (log)signature transform as the best performance across all depths. With this metric, the average rank of the signature transform was 1.23 whilst the average rank of the logsignature transform was 1.77, corresponding to a p-value of 0.01 with the Wilcoxon signed-rank test. Thus we find that the signature transform performs significantly better than the logsignature transform.

Rescaling We consider using no rescaling, pre-signature rescaling, or post-signature rescaling. See Figure 4. We see that pre-signature rescaling performs significantly worse than the other two options.

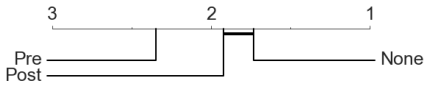

Figure 4: Performance of different rescalings

4.3 A canonical signature method

Given the results in the previous section, we derive a simple canonical signature method.

1. Unless the problem is known to be translation or reparameterisation invariant, then use the time and basepoint augmentations. Basepoint and invisbility-reset produce essentially the same performance, so basepoint is recommended over invisibility-reset due to its lower computational cost, as it does not introduce an extra channel.
2. The lead-lag augmentation should be considered, but we do not recommend it in general. This is because the performance improvement it offers is relatively slight, and it is an expensive augmentation, due to its doubling of the number of pre-signature channels; recall that the number of signature features scales according to $\mathcal{O}(d^N)$, where d is the input channels and N is the depth of the (log)signature.
3. Use hierarchical dyadic windows, and the signature transform; both have a depth hyperparameter that must be optimised.

We emphasise that this does not represent a best option for every application, but is meant to represent a compromise between broad applicability, ease of implementation, computational cost, and good performance. To validate that these choices work well collectively, we investigate its performance on all 30 datasets from the UEA archive.

Figure 5 shows the critical difference plot of the canonical signature pipeline, combined with a random forest, against the benchmark classifiers of Bagnall et al. (2018), which are nearest-neighbours with either Euclidean (ED) or dynamic time warping (DTW) based distances. These have been shown by Bagnall et al. (2017) to represent a strong baseline for domain-independent time series classification.

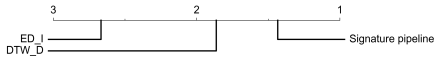

Figure 5: Performance on UEA datasets.

We see that our canonical signature method is significantly better than the previous benchmarks. Further comparisons to other benchmarks are beyond the scope of this paper; our primary goal is to compare the variations of the signature method against each other, and we refer to the existing literature for comparisons of the signature method against other methods.

See Appendix B.3 for further details.

4.4 Further results

See Appendix C for further results, in particular on the running times, sensitivity-inducing augmentations broken down by dataset type, an additional study on the most effective depth for the signature, and the precise numerical results for each individual test considered here.

5 Conclusion

We introduce a generalised signature method as a framework to capture recently proposed variations on the signature method. In doing so we are able to understand their conceptual groupings, and thus also understand how different variations may be combined. We go on to perform a first-of-its-kind extensive empirical investigation across 26 datasets, as to which elements of this framework are most important for performance in a domain-independent setting. As a result, we are able to present a canonical signature method that represents a best-practices domain-agnostic starting point, which may be fine tuned to any particular task.

Broader impact

Many variations have been proposed on the signature method, and so we hope that the present paper will be of benefit to the signature community, through its introduction of a common language for the method and introduction of a canonical domain-independent signature method. As signatures are a subset of the time series analysis community, we expect to indirectly be of benefit to the broader community as well. We do not expect any specific negative outcomes from the broader impact of this paper.

Acknowledgements

Thanks to Tony Bagnall for putting together the UEA datasets that were used in this paper. AF thanks Gérard Biau for stimulating discussions and insightful suggestions. JM was supported by the EPSRC grant EP/L015803/1 in collaboration with Iterex Therapuetics. AF was supported by a grant from Région Ile-de-France. PK was supported by the EPSRC grant EP/L015811/1. JM, PK, TL were supported by the Alan Turing Institute under the EPSRC grant EP/N510129/1.

References

Arribas, I. P., Goodwin, G. M., Geddes, J. R., Lyons, T., and Saunders, K. E. (2018). A signature-based machine learning model for distinguishing bipolar disorder and borderline personality disorder. *Translational psychiatry*, 8:1–7.

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

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

Benavoli, A., Corani, G., and Mangili, F. (2016). Should we really use post-hoc tests based on mean-ranks? *The Journal of Machine Learning Research*, 17:152–161.

Bonnier, P., Kidger, P., Perez Arribas, I., Salvi, C., and Lyons, T. (2019). Deep Signature Transforms. In *Advances in Neural Information Processing Systems*, pages 3099–3109.

Che, Z., Purushotham, S., Cho, K., Sontag, D., and Liu, Y. (2018). Recurrent neural networks for multivariate time series with missing values. *Scientific Reports*, 8:6085.

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

Cho, K., van Merrienboer, B., Gulcehre, C., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *EMNLP 2014*.

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

Fermanian, A. (2019). Embedding and learning with signatures. *arXiv:1911.13211*.

Flint, G., Hambly, B., and Lyons, T. (2016). Discretely sampled signals and the rough Hoff process. *Stochastic Processes and their Applications*, 126:2593–2614.

Friz, P. K. and Victoir, N. B. (2010). *Multidimensional Stochastic Processes as Rough Paths: Theory and Applications*, volume 120 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, Cambridge.

Futoma, J., Hariharan, S., and Heller, K. (2017). Learning to detect sepsis with a multitask Gaussian process RNN classifier. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1174–1182.

Grabocka, J., Schilling, N., Wistuba, M., and Schmidt-Thieme, L. (2014). Learning time-series shapelets. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 392–401.

Greff, K., Klein, A., Chovanec, M., Hutter, F., and Schmidhuber, J. (2017). The sacred infrastructure for computational research. In *Proceedings of the Python in Science Conferences-SciPy Conferences*.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep Residual Learning for Image Recognition. *arXiv:1512.03385*.

Howison, S., Nevado-Holgado, A., Swaminathan, S., Kormilitzin, A., Morrill, J., and Lyons, T. (2020). Utilisation of the signature method to identify the early onset of sepsis from multivariate physiological time series in critical care monitoring. *Critical Care Medicine*.

Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L., and Muller, P.-A. (2019). Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33:917–963.

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

Kidger, P., Morrill, J., Foster, J., and Lyons, T. (2020a). Neural controlled differential equations for irregular time series. *arXiv:2005.08926*.

Kidger, P., Morrill, J., and Lyons, T. (2020b). Generalised Interpretable Shapelets for Irregular Time Series. *arXiv:2005.13948*.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *Proceedings of International Conference on Learning Representations*.

Király, F. J. and Oberhauser, H. (2019). Kernels for sequentially ordered data. *Journal of Machine Learning Research*.

Kormilitzin, A., Saunders, K., Harrison, P., Geddes, J., and Lyons, T. (2016). Application of the signature method to pattern recognition in the cequel clinical trial. *arXiv:1606.02074*.

Lai, S., Jin, L., and Yang, W. (2017). Online signature verification using recurrent neural network and length-normalized path signature descriptor. In *Proceedings of the 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 400–405. IEEE.

Levin, D., Lyons, T., and Ni, H. (2013). Learning from the past, predicting the statistics for the future, learning an evolving system. *arXiv:1309.0260*.

Li, C., Zhang, X., and Jin, L. (2017). LPSNet: a novel log path signature feature based hand gesture recognition framework. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 631–639.

Li, S. C.-X. and Marlin, B. M. (2016). A scalable end-to-end Gaussian process adapter for irregularly sampled time series classification. In *Advances in Neural Information Processing Systems*, pages 1804–1812.

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

Lyons, T., Ni, H., and Oberhauser, H. (2014). A feature set for streams and an application to high-frequency financial tick data. In *Proceedings of the 2014 International Conference on Big Data Science and Computing*, page 5. ACM.

Lyons, T. and Oberhauser, H. (2017). Sketching the order of events. *arXiv:1708.09708*.

Lyons, T. J., Caruana, M., and Lévy, T. (2007). *Differential Equations driven by Rough Paths*, volume 1908 of *Lecture Notes in Mathematics*. Springer, Berlin.

Morrill, J., Kormilitzin, A., Nevado-Holgado, A., Swaminathan, S., Howison, S., and Lyons, T. (2019). The signature-based model for early detection of sepsis from electronic health records in the intensive care unit. *International Conference in Computing in Cardiology*.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in PyTorch.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

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

Reizenstein, J. and Graham, B. (2018). The iisignature library: efficient calculation of iterated-integral signatures and log signatures. *arXiv:1802.08252*.

Reyes-Ortiz, J.-L., Oneto, L., Samà, A., Parra, X., and Anguita, D. (2016). Transition-aware human activity recognition using smartphones. *Neurocomputing*, 171:754–767.

Rubanova, Y., Chen, T. Q., and Duvenaud, D. K. (2019). Latent ordinary differential equations for irregularly-sampled time series. In *Advances in Neural Information Processing Systems*, pages 5320–5330.

Tange, O. (2011). Gnu parallel - the command-line power tool. *The USENIX Magazine*, 36:42–47.

Toth, C. and Oberhauser, H. (2019). Variational Gaussian processes with signature covariances. *arXiv:1906.08215*.

Warden, P. (2018). Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv:1804.03209*.

Wu, Y., Ni, H., Lyons, T., and Hudson, R. (2020). Signature features with the visibility transformation. *arXiv:2004.04006*.

Yang, W., Jin, L., and Liu, M. (2016a). Deepwriterid: An end-to-end online text-independent writer identification system. *IEEE Intelligent Systems*, 31:45–53.

Yang, W., Jin, L., Ni, H., and Lyons, T. (2016b). Rotation-free online handwritten character recognition using dyadic path signature features, hanging normalization, and deep neural network. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 4083–4088. IEEE.

Yang, W., Lyons, T., Ni, H., Schmid, C., Jin, L., and Chang, J. (2017). Developing the path signature methodology and its application to landmark-based human action recognition. *arXiv:1707.03993*.

Ye, L. and Keogh, E. (2009). Time series shapelets: a new primitive for data mining. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 947–956.

Supplementary material

A Augmentations

We recall that an augmentation is a map

$$\phi\colon \mathcal{S}(\mathbb{R}^d) \to \mathcal{S}(\mathbb{R}^e)^p$$

We give below the precise definition of the different augmentations considered in the study, which are summarized in Table 2. These augmentations were not typically introduced using such language, so this serves as a reference for how the existing literature may be interpreted through the generalised signature method.

Throughout the section, we consider a sequence $\mathbf{x} = (x_1, \dots, x_n) \in \mathcal{S}(\mathbb{R}^d)$.

Table 2: Summary of the different augmentations

	e	p	Property
Fixed augmentations			
None	d	1	
Time	$d+1$	1	sensitivity to parametrization, uniqueness of the signature map
Invisibility-reset	$d+1$	1	sensitivity to translation
Basepoint	d	1	sensitivity to translation
Lead-lag	$2d$	1	information about quadratic variation, uniqueness of the signature map
Coordinates projection			dimensionality reduction
with singletons	1	d	
with pairs	2	$d(d-1)$	
with triplets	3	$d(d^2-1)$	
Random projections	e	p	dimensionality reduction
Learnt augmentations			
Learnt projections	e	p	data-dependent and linear
Stream-preserving neural network	e	1	data-dependent
Multi-headed stream-preserving NN	e	p	data-dependent

Time augmentation For a regularly sampled time series, the time augmentation is defined by

$$\phi(\mathbf{x}) = \big((1, x_1), \dots, (n, x_n)\big) \in \mathcal{S}(\mathbb{R}^{d+1}).$$

It ensures uniqueness of the signature transformation and removes the parametrization invariance (Levin et al., 2013).

If the time series is irregularly sampled, and there is an additional sequence of timestamps $\mathbf{t} = (t_1, \dots, t_n) \in \mathcal{S}(\mathbb{R})$ such that $\mathrm{Length}(\mathbf{t}) = \mathrm{Length}(\mathbf{x})$, then this is instead defined as

$$\phi_{\mathbf{t}}(\mathbf{x}) = \big((t_1, x_1), \dots, (t_n, x_n)\big) \in \mathcal{S}(\mathbb{R}^{d+1}).$$

Invisibility-reset augmentation First introduced by Yang et al. (2017), the invisibility-reset augmentation consists in adding a coordinate to the sequence $\mathbf{x}$ that is constant equal to 1 but drops to 0 at the last time step, i.e.,

$$\phi(\mathbf{x}) = \big((1, x_1), \dots, (1, x_{n-1}), (1, x_n), (0, x_n), (0, 0)\big) \in \mathcal{S}(\mathbb{R}^{d+1}).$$

This augmentation adds information on the initial position of the path, which is otherwise not included in the signature as it is a translation-invariant map.

Basepoint augmentation Introduced by Kidger and Lyons (2020), the basepoint augmentation has the same goal as the invisibility-reset augmentation: removing the translation-invariant property of the signature. It simply adds the point 0 at the beginning of the sequence:

$$\phi(\mathbf{x}) = (0, x_1, \ldots, x_n) \in \mathcal{S}(\mathbb{R}^d).$$

The main difference compared to the invisibility-reset augmentation is that the signature of $\mathbf{x}$ is contained in the signature of the invisibility-reset augmented path, whereas it is not in the signature of the basepoint augmented path. The price paid is that the invisibility-reset augmentation introduces redundancy into the signature, and is more computationally expensive due to the additional channel. (Recall that the signature method scales as $\mathcal{O}(d^N)$, where d is the input channels and N is the depth of the (log)signature.)

Lead-lag augmentation The lead-lag augmentation, introduced by Chevyrev and Kormilitzin (2016) and Flint et al. (2016) has been used in several applications (See for example Lyons et al. (2014); Kormilitzin et al. (2016); Yang et al. (2017)). It adds lagged copies of the path as new coordinates. This then explicitly captures the quadratic variation of the underlying process (Flint et al., 2016). As many different lags as desired may be added. If there is a single lag of a single timestep, then this corresponds to

$$\phi(\mathbf{x}) = ((x_1, x_1), (x_2, x_1), (x_2, x_2), \ldots, (x_n, x_n)) \in \mathcal{S}(\mathbb{R}^{2d}).$$

Coordinate projections For multidimensional streams, one may want to compute the signature of a subset of coordinates individually, rather than the signature of the whole stream; doing so restricts the interaction considered by the signature to just those between the projected coordinates. Let $\mathbf{x^1}, \ldots, \mathbf{x^d} \in \mathcal{S}(\mathbb{R})$ denote the different coordinates of $\mathbf{x} \in \mathcal{S}(\mathbb{R}^d)$. Let $\mathbf{t} = (1, \ldots, n) \in \mathcal{S}(\mathbb{R})$.

Then we define the singleton coordinate projection as

$$\phi(\mathbf{x}) = \big((\mathbf{t}, \mathbf{x^1}), (\mathbf{t}, \mathbf{x^2}), \ldots, (\mathbf{t}, \mathbf{x^d}) \in \mathcal{S}(\mathbb{R}^2)^d,$$

whilst considering all possible pairs of coordinates yields the augmentation

$$\phi(\mathbf{x}) = \big((\mathbf{t}, \mathbf{x^1}, \mathbf{x^2}), (\mathbf{t}, \mathbf{x^1}, \mathbf{x^3}), \ldots, (\mathbf{t}, \mathbf{x^d}, \mathbf{x^{d-1}}) \in \mathcal{S}(\mathbb{R}^2)^{d(d-1)},$$

and all possible triples yields the augmentation

$$\phi(\mathbf{x}) = \big((\mathbf{t}, \mathbf{x^1}, \mathbf{x^1}, \mathbf{x^2}), (\mathbf{t}, \mathbf{x^1}, \mathbf{x^1}, \mathbf{x^3}), \ldots, (\mathbf{t}, \mathbf{x^d}, \mathbf{x^d}, \mathbf{x^{d-1}}) \in \mathcal{S}(\mathbb{R}^3)^{d(d^2-1)}.$$

The decision to always include a time dimension is a somewhat arbitrary one, and it may alternatively be excluded if desired. (This is done so as to make sense of singleton coordinate projections; otherwise the result is a collection of univariate time series, for which the signature extracts only the increment due to the tree-like equivalence property.)

Random projections When the dimension of the input path is very large, Lyons and Oberhauser (2017) have proposed to project it into a smaller space by taking multiple random projections. Let $e < d$ and let $A_i : \mathbb{R}^d \to \mathbb{R}^e$ be random affine transformations indexed by $i \in \{1, \ldots, p\}$. Then ϕ is defined as

$$\phi(\mathbf{x}) = ((A_1 x_1, \ldots, A_1 x_n), \ldots, (A_p x_1, \ldots, A_p x_n)) \in \mathcal{S}(\mathbb{R}^e)^p.$$

Learnt projections Rather than taking random projections, Liao et al. (2019) learn it from the data. This takes exactly the same form as the random projections, except that the A_i are learnt.

Stream-preserving neural network Bonnier et al. (2019) introduce arbitrary learnt sequence-to-sequences maps prior to the signature transform, and refer to such maps, when parameterised as neural networks, as stream-preserving neural networks. For example these may be standard convolutional or recurrent architectures. In general this may be any learnt transformation

$$\phi \colon \mathcal{S}(\mathbb{R}^d) \to \mathcal{S}(\mathbb{R}^e).$$

Multi-headed stream-preserving neural network A straightforward extension of stream-preserving neural networks is to use multiple such networks, so as to avoid a potential bottleneck through the single signature map that it is eventually used in. Letting $\phi^1, \ldots, \phi^p$ be p different stream-preserving neural networks, then this gives an augmentation

$$\phi(\mathbf{x}) = (\phi^1(\mathbf{x}), \ldots, \phi^p(\mathbf{x})) \in (\mathcal{S}(\mathbb{R}^e))^p.$$

B Implementation details

B.1 General notes

Code All the code for this project is available at `https://github.com/jambo6/generalised-signature-method`.

Libraries The machine learning framework used was PyTorch (Paszke et al., 2017) version 1.3.1. Signatures and logsignatures were computed using the Signatory library (Kidger and Lyons, 2020) version 1.1.6. Scikit-learn (Pedregosa et al., 2011) version 0.22.1 was used for the logistic regression and random forest models. The experiments were tracked using the Sacred framework (Greff et al., 2017) version 0.8.1.

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

Architectures Two different GRU models were used on every dataset; a 'small' one with 32 hidden channels and 2 layers, and a 'large' one with 256 hidden channels and 3 layers.

Likewise, two different Residual CNN models were considered. The 'small' one used 6 blocks, each composed of batch normalisation, ReLU activation, convolution with 32 filters and kernel size 4, batch normalisation, ReLU activation, and a final convolution with 32 filters and kernel size 4, so that there are also 32 channels along the 'residual path'. A final two-hidden-layer neural network with 256 neurons was placed on the output. The 'large' is similar, except that it used 128 filters in both the blocks and the residual path, had 8 blocks, used a kernel size of 8, and the final neural network had 1024 neurons.

The logistic regression was performed three times with different amounts of L^2 regularisation, with scaling hyperparameters of 0.01, 0.2 and 1; for every experiment the regularization hyperparameter achieving the best accuracy on the test set was used.

The random forest used the default Scikit-learn implementation with a maximum depth of 6 and 100 trees.

Optimiser The GRU and CNN were optimised using Adam (Kingma and Ba, 2015). The learning rate was 0.01 for the GRU, and 0.001 for the residual CNN. The small models were trained for a maximum of 500 epochs; the large models were trained for a maximum of 1000 epochs. The learning rate was decreased by a factor of 10 if validation loss did not improve over a plateau of 10 epochs. Early stopping was used if the validation loss did not improve for 30 epochs. After training the parameters were always rolled back to those that demonstrated the best validation loss over the course of training. The batch size used varied by dataset; in each it was taken to be the power of two that meant that the number of batches per epoch was closest to 40.

Computing infrastructure Experiments were run on an Amazon AWS G3 Instance (g3.16xlarge) equipped with 4 Tesla M60s, parallelized using GNUParallel (Tange, 2011).

B.2 Analysis of variations of the signature method

Splits The UEA archive comes with a pre-defined train-test split, which we respect. We take an 80%/20% train/validation split in the training data, stratified by class label. For the Human Activities and Postural Transitions dataset, we take a 60%/15%/25% train/validation/test split from the whole dataset. For the Speech Commands dataset, we take a 68%/17%/15% train/validation/test split from the whole dataset. (These somewhat odd choices corresponding to taking either 25% or 15% of the dataset as test, and then splitting the remaining 80%/20% between train and validation.) These train/validation splits are only used for the training of the GRU and CNN classifiers.

Combinations In total we tested 8569 different combinations.

The variations tested are divided into groups. The first group consists of the sensitivity-adding augmentations, namely time, basepoint and invisibility-reset. Relative to the baseline model, we test every possible combination of these. (Including using none of them.)

The second group consists of those other augmentations, namely the lead-lag, singleton coordinate projection, pair coordinate projection, triplet coordinate projection, random projections, learnt projections, and multi-headed stream preserving neural networks, and finally also the case of no additional augmentation.

For the random projections, we consider four possibilities, with $e \in \{3, 6\}$ and $p \in \{2, 5\}$, all relative to the baseline model.

For no additional augmentation, lead-lag, coordinate projections, learnt projections, and the multi-headed stream preserving neural networks, we compose them with the time, time+basepoint and time+invisibility-reset augmentations (the clear best three from the first group), all relative to the baseline model.

For the learnt projections, we consider four different possibilities corresponding to $e \in \{3, 6\}$ and $p \in \{2, 5\}$; together with the time/time+basepoint/time+invisibility-reset cases this yields a total of twelve possibilities.

For the multi-headed stream-preserving neural networks, we again consider four different possibilities corresponding to $e \in \{3, 6\}$ and $p \in \{2, 5\}$, for a total of twelve possible augmentation strategies. In each the neural network operates elementwise, so as to map one sequence to another, and is given by a feedforward neural network of three hidden layers separated by ReLU activation functions. When $e = 3$ the hidden layers have 16 neurons each, and when $e = 6$ they have 32 neurons each.

For both the learnt projections and multi-headed stream-preserving neural networks, training these requires backpropagating through the model, so these were only considered for the GRU and residual CNN model. (The logistic regression model would in principle be possible as well, except that we ended up implementing this through Scikit-learn rather than PyTorch.)

We note that there are a great many possible ways of doing stream preserving neural networks, of which these are a small fraction. Their relatively weak performance here may likely be improved upon with greater tuning on an individual task, or the selection of better final models than were considered here.

The third group consisted of the different windows. Recall that the baseline model used a global window; we then consider varying this to two possible sliding windows, two possible expanding windows, and three possible dyadic windows. The two possible sliding/expanding windows are chosen so that either 5 or 20 windows are applied across the full length of the dataset. The three possible dyadic windows are depths 2, 3, 4. Thus in total there are 8 possible window combinations we consider.

The fourth group consists of rescaling options, namely no rescaling, pre-signature rescaling, and post-signature rescaling.

Omissions For the empirical study on the variations on the signature method, we excluded those UEA datasets with a dimension d over 60, so as to reduce the computational cost. This results removes 6 of the 30 datasets from the study, namely DuckDuckGeese, FaceDetection, Heartbeat, InsectWingbeat, MotorImagery, and PEMS-SF. These were nonetheless used in the demonstration of performance of the canonical signature method in Figure 5. Furthermore those combinations of dataset/variation/model which produced more than 10^5 signature features were omitted, to keep the computation managable. See Table 3.

B.3 The canonical signature pipeline

For each dataset, we implement the following steps. First, the sequences are augmented with time and basepoint augmentations. Then, we consider every combination of signature depth in $\{1, 2, 3, 4, 5, 6\}$ and hierarchical dyadic window depth in $\{2, 3, 4\}$. We consider 'post' rescaling of the signature features. For each of these choices, we perform a randomized grid search on a random forest classifier to optimize its number of trees and maximal depth parameters. We test 20 combinations randomly sampled from the following grids:

$$\begin{aligned} \text{n\_trees} &= [50, 100, 500, 1000], \\ \text{max\_depth} &= [2, 4, 6, 8, 12, 16, 24, 32, 45, 60, 80, \text{None}]. \end{aligned}$$

Table 3: Summary of the number of combinations considered and omitted.

Variations	# Variations	# Classifiers	# Omitted Combinations	# Total Combinations
Basic augmentations (Figure 1)	6	6	54	936
Other augmentations (Figure 2)				
Lead-lag/ None	3	6	100/27	468
Coordinates projection (1)/(2)/(3)	3	6	12/12/54	468
Random projections	4	6	32	624
Learnt projections / MHSP	12	4	348/176	1248
Windows (Figure 3)	8	6	227	1248
Signature/Logsignature transform	12	6	361	1872
Rescalings (Figure 4)	3	6	12	468
Total			1415	9984

Note that a maximal depth set to 'None' means that the trees are expanded until all leaves contain exactly one sample. Finally, we choose the combination of signature and hierarchical dyadic window depths which maximise the out-of-bag score.

C Additional results

C.1 Analysis of variations of the signature method

Running time To get a sense of the cost of each augmentation or window, we present the run times of each augmentation/model combination, and each window/model combination. (The times for varying between signature and logsignature, and between different rescalings, are largely insignificant.) See Table 4.

The run times are averaged over every UEA dataset. As the datasets are of very different sizes this thus represents quite a crude statistic, and in particular produces very large variances, so these are most meaningful simply with respect to each other.

Sensitivity-inducing augmentations broken down by dataset type Table 5 shows the average rank of each of the first group of augmentations (that add sensitivity to certain kinds of perturbation) by dataset type, where the types are taken from Bagnall et al. (2018). (This may be regarded as a companion to Table 1.)

It is interesting to note that for EEG data, it seems better not to consider the time augmentation, whereas it is the case for other applications. In particular the combination of time and basepoint augmentations achieve the best ranks for human action and motion recognition (HAR and MOTION in Table 5). Recognizing an action may not be translation-invariant nor invariant by time reparametrization.

Depth study on the signature transform In the main text we focused on the difference between the signature and logsignature transforms, and stated that larger depths must be chosen by a bias-variance tradeoff. Here we consider varying the depth together with the choice of signature or logsignature, and taking the best transform for each depth. See Figure 6. We see that larger depths do indeed generally correspond to increased performance, up to a point. The optimal depth will depend on the complexity of the task, as the number of features increases exponentially with the depth.

C.2 Complete results

We present in Tables 6, 7, 8, 9, 10 and 11 the performance of the different signature variations on each dataset. The tables were obtained by maximizing the test accuracy of the signature method

Table 4: Average run time (in seconds) for various experiments. mean (std), averaged over all UEA datasets.

	Classifier			
	CNN	GRU	Logistic regression	Random forest
Time augmentation+Global window (Baseline)	69.8 (98.0)	22.2 (31.8)	2.67 (7.09)	2.23 (4.84)
Augmentation				
None	48.1 (63.5)	16.8 (33.6)	3.55 (9.91)	66.3 (321)
Lead-lag	48.58 (69.99)	15.2 (18.1)	5.76 (11.7)	3.35 (6.04)
Coordinates projection (1)	32.8 (31.49)	13.4 (17.8)	1.37 (4.2)	12.2 (59.3)
Coordinates projection (2)	41.5 (51.4)	22.6 (62.3)	3.01 (8.54)	42.3 (203)
Coordinates projection (3)	41.3 (39.9)	19.1 (24.5)	5.41 (9.76)	6.3 (14.1)
Random projection	62.2 (70.1)	21.1 (31.2)	0.86 (1.25)	1.4 (2.47)
Learnt projection	917 (1288)	752 (972)	–	–
Multi-headed stream-preserving	1051 (1677)	1758 (4442)	–	–
Window				
Sliding	90.6 (120)	79.4 (175)	10.1 (27.4)	6.4 (16.0)
Expanding	102 (133)	68.7 (115)	9.98 (27.2)	7.17 (19.0)
Dyadic	725 (868)	56.9 (65.1)	12.5 (33.2)	7.59 (18.3)

Table 5: Average ranks for different augmentations by type of data. Lower is better.

	Augmentation					
Data type	None	Time	Basepoint	Invisibility-reset	Time + Basepoint	Time + Invisibility-reset
EEG	3.88	3.50	4.00	**2.00**	4.00	3.63
HAR	5.00	2.95	4.85	3.65	**2.00**	2.55
MOTION	5.25	2.75	5.75	3.88	**1.50**	1.88
OTHER	4.43	3.31	4.88	3.19	2.87	**2.31**

over the different classifiers considered. Recall that some values are omitted due to the large number of signature features that would be obtained.

C.3 Canonical signature method

In Table 12 we give the full results for our canonical signature method on all UEA datasets, together with the results of a 1-nearest neighbors algorithm with dyamic time wrapping or Euclidean distance, taken from Bagnall et al. (2018).

Figure 7 plots the performance of the canonical signature method against nearest neighbours with dynamic time warping. We see that the signature pipeline substantially improves the accuracy of several datasets, whereas it substantially decreases the accuracy for only one dataset (DuckDuckGeese).

Finally, we give in Table 13 the hyperparameters that were selected for each dataset in the signature pipeline model.

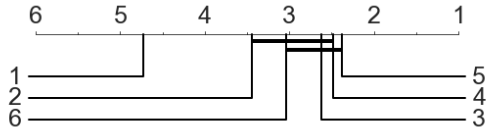

Figure 6: Critical differences plot for the depth study on the UEA datasets.

Table 6: Accuracy of sensitivity-inducing augmentations per dataset

	Augmentation					
Dataset	None	Time	Basepoint	Invisibility-reset	Time + Basepoint	Time + Invisibility-reset
ArticularyWordRecognition	96.0%	96.3%	95.7%	96.3%	97.7%	97.0%
AtrialFibrillation	46.7%	46.7%	40.0%	33.3%	40.0%	40.0%
BasicMotions	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
CharacterTrajectories	88.3%	93.2%	86.4%	88.7%	93.8%	93.7%
Cricket	91.7%	94.4%	94.4%	97.2%	97.2%	95.8%
ERing	80.0%	92.6%	77.4%	89.6%	91.9%	92.2%
EigenWorms	72.5%	79.4%	74.8%	76.3%	87.0%	81.7%
Epilepsy	84.8%	89.9%	91.3%	91.3%	97.1%	94.9%
EthanolConcentration	27.8%	29.3%	33.5%	41.8%	34.6%	41.4%
FingerMovements	55.0%	52.0%	57.0%	58.0%	55.0%	56.0%
HandMovementDirection	29.7%	33.8%	32.4%	33.8%	36.5%	32.4%
Handwriting	21.9%	30.8%	23.6%	24.6%	30.6%	28.7%
JapaneseVowels	85.4%	85.1%	97.3%	98.1%	97.3%	98.1%
LSST	42.0%	47.4%	44.0%	44.4%	50.9%	48.7%
Libras	72.8%	84.4%	65.0%	75.0%	80.0%	77.2%
NATOPS	81.7%	88.3%	79.4%	79.4%	91.1%	92.2%
PenDigits	91.1%	97.1%	88.3%	93.1%	96.8%	97.1%
PhonemeSpectra	4.7%	8.2%	4.3%	5.7%	10.0%	8.1%
RacketSports	78.9%	80.3%	78.9%	82.9%	82.9%	81.6%
SelfRegulationSCP1	81.6%	83.3%	76.8%	84.0%	75.4%	85.0%
SelfRegulationSCP2	57.2%	56.7%	56.1%	56.7%	56.1%	55.0%
SpokenArabicDigits	82.5%	85.5%	80.5%	88.0%	85.1%	90.1%
StandWalkJump	60.0%	46.7%	40.0%	46.7%	40.0%	46.7%
UWaveGestureLibrary	84.1%	87.5%	79.7%	82.8%	87.5%	83.4%
Human Activity	73.0%	76.6%	92.3%	92.2%	93.0%	93.8%
Speech Commands	71.4%	75.9%	74.7%	74.9%	79.7%	79.5%
Average rank	4.69	3.12	4.87	3.29	**2.5**	2.54

Table 7: Accuracy of other augmentations per dataset

Dataset	**Augmentation**							
	None	Lead-lag	Coordinates projection (1)	(2)	(3)	Random projection	Learnt projection	MHSP
ArticularyWordRecognition	97.7%	96.3%	83.3%	95.7%	97.0%	95.3%	73.7%	80.3%
AtrialFibrillation	46.7%	40.0%	53.3%	53.3%	46.7%	66.7%	46.7%	53.3%
BasicMotions	100.0%	100.0%	80.0%	100.0%	100.0%	100.0%	97.5%	87.5%
CharacterTrajectories	93.8%	95.3%	43.9%	93.2%	93.8%	93.3%	89.6%	91.1%
Cricket	97.2%	98.6%	90.3%	97.2%	95.8%	88.9%	69.4%	56.9%
ERing	92.6%	94.8%	79.3%	89.3%	91.9%	74.4%	62.2%	61.1%
EigenWorms	87.0%	87.8%	50.4%	84.0%	89.3%	78.6%	–	–
Epilepsy	97.1%	97.1%	55.8%	95.7%	95.7%	81.9%	67.4%	65.9%
EthanolConcentration	41.4%	39.9%	42.2%	43.3%	42.2%	30.4%	32.3%	30.0%
FingerMovements	56.0%	–	59.0%	58.0%	–	55.0%	60.0%	65.0%
HandMovementDirection	36.5%	31.1%	31.1%	40.5%	37.8%	37.8%	33.8%	44.6%
Handwriting	30.8%	33.5%	11.3%	27.8%	30.0%	21.6%	12.6%	13.2%
JapaneseVowels	98.1%	97.6%	94.1%	97.8%	97.6%	84.1%	95.4%	95.9%
LSST	50.9%	55.6%	43.5%	51.7%	52.8%	43.7%	34.4%	39.8%
Libras	84.4%	86.7%	47.8%	83.9%	85.0%	86.7%	73.3%	81.1%
NATOPS	92.2%	–	33.3%	90.6%	91.1%	85.6%	83.9%	81.7%
PenDigits	97.1%	98.3%	60.2%	96.8%	97.2%	96.7%	96.5%	97.4%
PhonemeSpectra	10.0%	–	4.5%	9.4%	10.6%	8.9%	7.2%	7.7%
RacketSports	82.9%	82.2%	53.3%	85.5%	84.2%	75.7%	73.7%	75.0%
SelfRegulationSCP1	85.0%	86.0%	61.8%	85.0%	84.0%	81.6%	86.7%	84.6%
SelfRegulationSCP2	56.7%	57.2%	55.6%	58.9%	55.6%	60.6%	59.4%	57.8%
SpokenArabicDigits	90.1%	96.6%	58.5%	86.0%	90.0%	83.0%	88.0%	86.0%
StandWalkJump	46.7%	40.0%	40.0%	53.3%	40.0%	53.3%	–	–
UWaveGestureLibrary	87.5%	88.8%	50.6%	85.6%	87.5%	86.2%	74.1%	75.6%
Human Activity	93.8%	93.6%	75.8%	93.2%	93.6%	69.2%	91.3%	91.5%
Speech Commands	79.7%	–	14.9%	77.1%	–	70.2%	–	76.1%
Average ranks	2.9	**2.77**	6.52	3.33	3.23	4.71	5.83	5.31

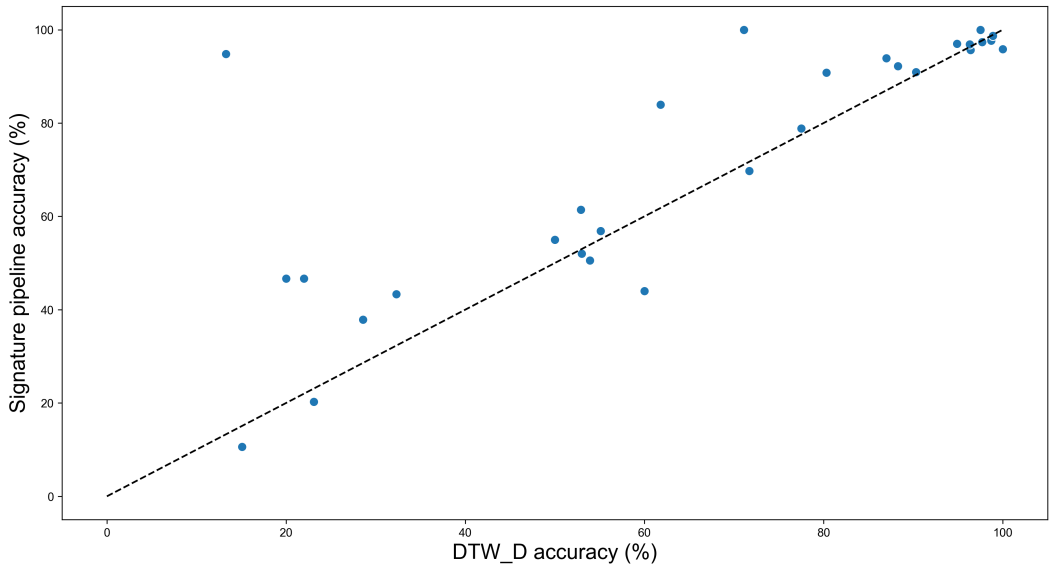

Figure 7: Test accuracy of the signature pipeline against DTW_D, the next best performin algorithim. Each point represents a dataset.

Table 8: Accuracy of windows per dataset

	Window			
Dataset	Global	Sliding	Expanding	Dyadic
ArticularyWordRecognition	96.3%	89.3%	99.0%	99.0%
AtrialFibrillation	46.7%	46.7%	46.7%	60.0%
BasicMotions	100.0%	100.0%	100.0%	100.0%
CharacterTrajectories	93.2%	94.6%	96.9%	97.1%
Cricket	97.2%	93.1%	97.2%	95.8%
ERing	90.7%	88.5%	91.9%	94.8%
EigenWorms	80.2%	74.8%	78.6%	76.3%
Epilepsy	89.9%	92.8%	92.0%	94.2%
EthanolConcentration	30.4%	38.8%	30.0%	35.7%
FingerMovements	50.0%	–	–	–
HandMovementDirection	33.8%	33.8%	36.5%	33.8%
Handwriting	30.1%	21.5%	30.2%	27.2%
JapaneseVowels	85.1%	76.5%	88.1%	89.2%
LSST	47.8%	43.1%	48.3%	46.6%
Libras	83.3%	85.6%	91.1%	90.0%
NATOPS	93.3%	84.4%	90.6%	–
PenDigits	95.8%	–	–	97.6%
PhonemeSpectra	8.6%	9.2%	9.6%	10.4%
RacketSports	82.2%	80.3%	84.2%	88.2%
SelfRegulationSCP1	82.6%	87.4%	84.0%	86.7%
SelfRegulationSCP2	56.7%	60.6%	54.4%	56.1%
SpokenArabicDigits	85.5%	91.5%	93.7%	96.6%
StandWalkJump	46.7%	53.3%	46.7%	53.3%
UWaveGestureLibrary	86.6%	79.4%	89.1%	89.7%
Human Activity	76.1%	73.0%	80.4%	81.7%
Speech Commands	75.9%	76.5%	82.3%	83.0%
Average ranks	2.83	3.04	2.17	**1.73**

Table 9: Accuracy of signature and logsignature transforms per dataset

	Transform	
Dataset	Signature	Logsignature
ArticularyWordRecognition	97.7%	97.3%
AtrialFibrillation	60.0%	53.3%
BasicMotions	100.0%	100.0%
CharacterTrajectories	93.8%	93.8%
Cricket	100.0%	100.0%
ERing	90.0%	89.3%
EigenWorms	79.4%	81.7%
Epilepsy	93.5%	91.3%
EthanolConcentration	31.9%	30.0%
FingerMovements	59.0%	56.0%
HandMovementDirection	40.5%	40.5%
Handwriting	35.3%	24.5%
JapaneseVowels	85.9%	86.8%
LSST	52.0%	46.4%
Libras	90.6%	87.8%
NATOPS	89.4%	91.7%
PenDigits	97.8%	97.5%
PhonemeSpectra	8.9%	7.6%
RacketSports	85.5%	84.9%
SelfRegulationSCP1	84.0%	83.3%
SelfRegulationSCP2	57.2%	56.1%
SpokenArabicDigits	87.5%	85.8%
StandWalkJump	53.3%	53.3%
UWaveGestureLibrary	90.0%	86.9%
Human Activity	78.7%	78.3%
Speech Commands	75.9%	76.3%
Average ranks	**1.25**	1.75

Table 10: Accuracy of rescaling choices per dataset

	Rescaling		
Dataset	None	Post	Pre
ArticularyWordRecognition	97.3%	97.0%	97.7%
AtrialFibrillation	53.3%	53.3%	46.7%
BasicMotions	100.0%	100.0%	100.0%
CharacterTrajectories	94.6%	94.6%	94.6%
Cricket	98.6%	97.2%	97.2%
ERing	93.7%	93.7%	93.0%
EigenWorms	80.9%	80.9%	79.4%
Epilepsy	92.0%	92.0%	91.3%
EthanolConcentration	31.2%	31.6%	30.0%
FingerMovements	54.0%	54.0%	50.0%
HandMovementDirection	35.1%	32.4%	29.7%
Handwriting	36.6%	36.4%	37.1%
JapaneseVowels	87.3%	85.9%	85.7%
LSST	55.8%	55.6%	55.4%
Libras	85.0%	86.1%	84.4%
NATOPS	92.8%	92.8%	91.7%
PenDigits	96.6%	96.7%	96.7%
PhonemeSpectra	8.0%	8.1%	8.2%
RacketSports	84.2%	84.2%	83.6%
SelfRegulationSCP1	79.5%	83.3%	84.6%
SelfRegulationSCP2	56.1%	57.2%	56.7%
SpokenArabicDigits	90.5%	90.5%	90.2%
StandWalkJump	46.7%	53.3%	46.7%
UWaveGestureLibrary	87.5%	87.2%	87.2%
Human Activity	85.0%	84.6%	85.1%
Speech Commands	77.0%	75.7%	75.9%
Average ranks	**1.73**	1.92	2.35

Table 11: Accuracy of (log)signature depth per dataset

	Depth					
Dataset	1	2	3	4	5	6
ArticularyWordRecognition	83.3%	96.0%	97.3%	97.7%	95.3%	–
AtrialFibrillation	40.0%	40.0%	60.0%	33.3%	40.0%	53.3%
BasicMotions	70.0%	100.0%	100.0%	100.0%	100.0%	92.5%
CharacterTrajectories	42.3%	88.0%	93.2%	93.8%	92.9%	93.8%
Cricket	30.6%	93.1%	97.2%	98.6%	100.0%	–
ERing	77.0%	89.6%	90.0%	89.3%	88.9%	84.8%
EigenWorms	46.6%	81.7%	79.4%	79.4%	–	–
Epilepsy	50.7%	78.3%	89.9%	93.5%	93.5%	93.5%
EthanolConcentration	25.5%	30.8%	30.0%	31.2%	31.9%	27.4%
FingerMovements	57.0%	58.0%	59.0%	–	–	–
HandMovementDirection	40.5%	36.5%	37.8%	39.2%	32.4%	–
Handwriting	7.3%	22.4%	32.4%	33.3%	35.3%	32.7%
JapaneseVowels	78.9%	85.9%	86.8%	84.3%	81.4%	–
LSST	40.9%	45.6%	47.6%	50.6%	52.0%	44.7%
Libras	51.7%	77.2%	85.0%	87.8%	88.9%	90.6%
NATOPS	35.0%	86.7%	91.7%	–	–	–
PenDigits	60.0%	90.4%	96.9%	97.7%	97.4%	97.8%
PhonemeSpectra	4.1%	7.6%	8.9%	–	–	–
RacketSports	44.1%	77.0%	78.9%	84.9%	85.5%	82.2%
SelfRegulationSCP1	53.6%	80.2%	84.0%	83.3%	81.9%	–
SelfRegulationSCP2	56.1%	55.0%	56.7%	54.4%	57.2%	–
SpokenArabicDigits	52.1%	85.8%	85.5%	87.5%	–	–
StandWalkJump	46.7%	46.7%	46.7%	46.7%	53.3%	46.7%
UWaveGestureLibrary	49.4%	83.1%	86.6%	87.8%	90.0%	88.1%
Human Activity	47.7%	78.3%	76.0%	78.7%	78.6%	–
Speech Commands	14.8%	69.6%	76.3%	–	–	–
Average ranks	4.73	3.44	2.62	2.48	**2.38**	3.04

Table 12: Results of the signature canonical pipeline with a Random Forest for the UEA archive. Best accuracies for each dataset are in bold.

	Classifcation method		
Dataset	ED_I	DTW_D	Signature pipeline
ArticularyWordRecognition	97.0%	**98.7%**	97.7%
AtrialFibrillation	26.7%	22.0%	**46.7%**
BasicMotions	67.6%	97.5%	**100.0%**
CharacterTrajectories	96.4%	**98.9%**	98.7%
Cricket	94.4%	**100.0%**	95.8%
DuckDuckGeese	27.5%	**60.0%**	44.0%
ERing	13.3%	13.3%	**94.8%**
EigenWorms	54.9%	61.8%	**84.0%**
Epilepsy	66.6%	**96.4%**	95.7%
EthanolConcentration	29.3%	32.3%	**43.3%**
FaceDetection	51.9%	52.9%	**61.4%**
FingerMovements	**55.0%**	53.0%	52.0%
HandMovementDirection	**27.8%**	23.1%	20.3%
Handwriting	20.0%	28.6%	**37.9%**
Heartbeat	61.9%	**71.7%**	69.8%
InsectWingbeat	12.8%	–	**63.7%**
JapaneseVowels	92.4%	94.9%	**97.0%**
LSST	45.6%	55.1%	**56.9%**
Libras	83.3%	87.0%	**93.9%**
MotorImagery	51.0%	50.0%	**55.0%**
NATOPS	85.0%	88.3%	**92.2%**
PEMS-SF	70.5%	71.1%	**100.0%**
PenDigits	97.3%	**97.7%**	97.4%
Phoneme	10.4%	**15.1%**	10.6%
RacketSports	86.8%	80.3%	**90.8%**
SelfRegulationSCP1	77.1%	77.5%	**78.8%**
SelfRegulationSCP2	48.3%	**53.9%**	50.6%
SpokenArabicDigits	96.7%	96.3%	**96.9%**
StandWalkJump	20.0%	20.0%	**46.7%**
UWaveGestureLibrary	88.1%	90.3%	**90.9%**
Average ranks	2.667	1.862	**1.433**

Table 13: Hyperparameters used for each dataset in the signature pipeline model.

	Signature hyperparmeters		**RF hyperparameters**		**Other**
Dataset	Depth	Dyadic depth	Max depth	Num estimators	Training time (s)
ArticularyWordRecognition	2	2	45	500	60.3
AtrialFibrillation	1	2	None	50	35.9
BasicMotions	2	2	24	100	19.3
CharacterTrajectories	4	2	80	500	181.4
Cricket	2	4	6	500	249.0
DuckDuckGeese	1	2	16	100	140.9
ERing	2	3	8	1000	16.7
EigenWorms	3	3	12	100	250.1
Epilepsy	2	3	8	1000	42.8
EthanolConcentration	2	4	24	1000	454.2
FaceDetection	1	4	8	1000	1816.2
FingerMovements	1	2	4	100	30.8
HandMovementDirection	2	2	None	50	66.3
Handwriting	6	2	32	1000	280.3
Heartbeat	1	4	None	50	45.1
InsectWingbeat	1	3	45	1000	5367.5
JapaneseVowels	2	3	6	1000	95.4
LSST	4	2	60	1000	1590.5
Libras	6	2	None	100	28.4
MotorImagery	1	3	24	50	347.1
NATOPS	2	3	32	1000	37.8
PEMS-SF	1	3	80	1000	252.3
PenDigits	3	2	80	1000	302.3
PhonemeSpectra	2	4	45	1000	2188.7
RacketSports	3	2	None	500	13.9
SelfRegulationSCP1	3	2	None	100	186.6
SelfRegulationSCP2	3	2	6	50	138.1
SpokenArabicDigits	2	3	45	1000	1204.0
StandWalkJump	1	3	2	50	101.5
UWaveGestureLibrary	2	2	60	500	21.8