Supplementary for SigGPDE: Scaling Sparse Gaussian Processes on Sequential Data

A. Additional Proof

In this section we prove Thm. 4.1 from the main paper which yields an efficient algorithm to compute the gradients of the signature kernel with respect to its input paths. We recall Thm. 4.1 below.

Theorem 4.1. *For any $\gamma \in \mathcal{X}$ the directional derivative $k_\gamma(X,Y)$ of the signature kernel along the path γ satisfies the following relation*

$$k_\gamma(X,Y) = \int_0^T \int_0^T U(s,t)\widetilde{U}(T-s,T-t)(\dot{\gamma}_s^T \dot{Y}_t)dsdt$$

where $\widetilde{U}(s,t) = k(\overleftarrow{X}|_{[0,s]}, \overleftarrow{Y}|_{[0,t]})$ and where $\overleftarrow{X}, \overleftarrow{Y}$ are respectively the paths X, Y reversed in time.

Before proving Thm. 4.1 we need the following important lemma.

Lemma A.1. *For any two paths continuous paths of bounded variation $X, Y \in \mathcal{X}$ the signature kernel satisfies the following relation*

$$k(X,Y) = k(\overleftarrow{X}, \overleftarrow{Y}) \tag{1}$$

where $\overleftarrow{X}, \overleftarrow{Y}$ are the respectively X, Y reversed in time.

Proof. It is a standard result in rough path theory (see for example (Lyons et al., 2007)) that $S(\overleftarrow{X}) = S(X)^{-1}$, where the inverse is taken in the set of grouplike elements, which is a group. The operator on grouplike elements $g : S(X) \mapsto S(X)^{-1}$ reverses the order of the letters in each word and multiplies the result by -1 if the length of the word is odd. Expanding out $k(\overleftarrow{X}, \overleftarrow{Y})$ coordinate-wise it is easy to see that the two -1's for words of odd length cancel as multiplied together, therefore the expansion of $k(X,Y)$ matches the one of $k(\overleftarrow{X}, \overleftarrow{Y})$. $\square$

Recall the notation for the signature kernel and its directional derivative used in the statement of Thm. 4.1:

$$U(s,t) := k\left(X|_{[0,s]}, Y|_{[0,t]}\right)$$
$$U_\gamma(s,t) := k_\gamma\left(X|_{[0,s]}, Y|_{[0,t]}\right)$$

Proof of Theorem 4.1. Let $\gamma : [0,T] \to \mathbb{R}^d$ be a continuous path of bounded variation along which we wish to differentiate k. Let's assume that for any $s,t \in [0,T]$ there exists a function $A_{s,t} : [0,T] \times [0,T] \to \mathbb{R}$ such that

$$U_\gamma(s,t) = \int_0^s \int_0^t A_{s,t}(u,v)U(u,v)(\dot{\gamma}_u^T \dot{Y}_v)dudv \tag{2}$$

Differentiating U_γ with respect to s and t we get

$$\frac{\partial^2 U_\gamma}{\partial s \partial t} = \int_0^s \int_0^t \frac{\partial^2 A_{s,t}(u,v)}{\partial s \partial t} U(u,v)(\dot{\gamma}_u^T \dot{Y}_v)dudv + A_{s,t}(s,t)U(s,t)(\dot{\gamma}_s^T \dot{Y}_t) \tag{3}$$

By eq. (41) in the main paper we know that the directional derivative of the signature kernel along the path γ solves the following PDE

$$\frac{\partial^2 U_\gamma}{\partial s \partial t} = (\dot{X}_s^T \dot{Y}_t)U_\gamma(s,t) + (\dot{\gamma}_s^T \dot{Y}_t)U(s,t) \tag{4}$$

Equating eq. (3) and eq. (4) we deduce that $A_{s,t}(s,t) = 1$ and

$$\begin{aligned}&\int_0^s \int_0^t \frac{\partial^2 A_{s,t}(u,v)}{\partial s \partial t} U(u,v)(\dot{\gamma}_u^T \dot{Y}_v)dudv \\ &= U_\gamma(s,t)(\dot{X}_s^T \dot{Y}_t) \\ &= (\dot{X}_s^T \dot{Y}_t) \int_0^s \int_0^t A_{s,t}(u,v)U(u,v)(\dot{\gamma}_u^T \dot{Y}_v)dudv\end{aligned}$$

Which implies that

$$\frac{\partial^2 A_{s,t}(u,v)}{\partial s \partial t} = (\dot{X}_s^T \dot{Y}_t)A_{s,t}(u,v) \tag{5}$$

Or equivalently, by integrating with respect to s and t

$$A_{s,t}(u,v) = 1 + \int_u^s \int_v^t A_{s',t'}(u,v)(\dot{X}_{s'}^T \dot{Y}_{t'})ds'dt' \tag{6}$$

Hence

$$A_{T,T}(u,v) = \langle S(X)_{[u,T]}, S(Y)_{[v,T]} \rangle \tag{7}$$
$$= k(\overleftarrow{X}|_{[0,T-u]}, \overleftarrow{Y}|_{[0,T-v]}) \tag{8}$$

where the last equality is a consequence of Lemma A.1. Pluging back this result into eq. (2) concludes the proof. $\square$

B. Additional Experimental Details

In this section we describe the experimental setup for Sec. 6. All experiments were conducted on NVIDIA Tesla P100 GPUs.

B.1. Data collection process

The classification tasks of Sections 6.1 and 6.2 were performed on two datasets (PenDigits, RightWhaleCalls) from the UCR & UEA time series classification repository.[1] For the large scale classification experiment of Sec. 6.3 we used a dataset of 1M satellite time series (STS).[2] Lastly, the climatic data (WeatherForecast) for rainfall prediction task in Sec. 6.4 was downloaded from the Max Planck Institute for Biogeochemistry website.[3]

Data pre-processing included the following two steps. As explained in Sec. 3.1, we first add a monotonically increasing coordinate to all multivariate time series that we call "time", which effectively augments by one the number of channels. This is a standard procedure employed within signature based methods (Toth & Oberhauser, 2020; Chevyrev & Kormilitzin, 2016). Then, we standard scale the time series using tslearn library (Tavenard et al., 2020). This is particularly important for the WeatherForecast dataset where channels have different scales. Additional processing steps have been performed for two datasets (RightWhaleCalls, WeatherForecast) which we treat separately next.

A standard data transformation to tackle classification tasks on audio signals consists in computing their *spectrograms*. We follow this procedure for the RightWhaleCalls dataset which contains univariate highly-oscillatory time series of length $2\,000$. We used the scipy Python library to do so. The spectrogram is commonly represented as a graph with one axis representing time, the other axis representing the frequency, and the color intensity representing the amplitude of a particular frequency at a particular time. In this paper, we consider the spectrogram as a multivariate time series, where each channel represents the change in amplitude of a particular frequency over time. Furthermore, exploiting the fact that frequencies in whale call signals are typically between 50 and 300Hz, we only consider frequencies which fall within this range. As a result we obtain 28-dimensional time series each of length 30. We then apply the pre-processing steps described above.

To create the WeatherForecast dataset we used the recordings of various climatic variables in two weather stations located in Germany from 2004 to 2020. The outliers were filtered out, and we used the recordings of 7 variables (depicted on Fig. 2) over 6 hours in order to predict whether it would rain by more than 1mm over the next hour. There is one recording every 10min resulting in input time series of length $\ell = 36$. Since there were much fewer positive cases (raining) than negative cases (not raining), we dropped at random a fraction of the data, such that the ratio of positive/negative examples is brought down to 3.

[1] `https://timeseriesclassification.com`

[2] `https://cloudstor.aarnet.edu.au/plus/index.php/s/pRLVtQyNhxDdCoM`

[3] `https://www.bgc-jena.mpg.de/wetter/weather_data.html`

B.2. Training procedure

The datasets for classification of sequential digits (PenDigits), audio signals (RightWhaleCalls), and satellite time series (STS) come with a predefined test-train split. In order to report standard deviations on our results we subsampled 20% (PenDigits,RightWhaleCalls) or 2% (STS) of the training set to form a validation set.

The training was equally split into 3 different phases. During the first phase, only the variational parameters are trained. For the second phase, both the variational parameters and the hyperparameters of the kernel are trained. During the last phase the variational parameters are trained on the full training set (the validation data being merged back). Overall, the hyperparameters are fixed for two-third of the iterations. SigGPDE and the GPSig-IT/IS baselines have the same set of hyperparameters, which correspond to the scaling factors for each channel for the ARD parametrization of the signature kernel Sec. 3.3. Those were initialized with the same value for all models. The inducing tensors for GPSig-IT and inducing sequences for GPSig-IS were initialized following the procedure outlined in (Toth & Oberhauser, 2020). We recall that for SigGPDE there is no such parameters to initialize. As recommended in (Toth & Oberhauser, 2020), we use a truncation level of $n = 4$ for their signature kernel algorithm (GPSig-IT/IS).

The minibatch size is either 50 (PenDigits, RightWhaleCalls) or 200 (STS). We used the Nadam optimizer (Dozat, 2016) with learning rate 10^{-3}. In the main paper we report the time per iteration which corresponds to one minibatch.

C. Additional Algorithmic Details

In this section we start by outlining the space and time complexities of the algorithms underlying SigGPDE. Then, we explain how we have developed a dedicated CUDA TensorFlow operator for GPU acceleration to speed-up the computation of the signature kernel and its gradients.

C.1. Complexity analysis

The main algorithms underpinning SigGPDE consist in computing three different covariance matrices to evaluate the ELBO. These are the covariance matrix between the inducing variables $\mathbf{u}$ (denoted by $C_{\mathbf{uu}}$), between the marginal $\mathbf{f}$ and the inducing variables (denoted by $C_{\mathbf{fu}}$), and finally the covariance matrix of $\mathbf{f}$ (its diagonal is denoted by $\mathrm{diag}(C_{\mathbf{ff}})$). In Tables 1 and 2 we compare the time and space complexities for the corresponding SigGPDE algorithms to those of GPSig-IT/IS.

In the SigGPDE sparse variational inference framework, $C_{\mathbf{uu}}$ is diagonal which lowers both the memory and computational costs (see first line Tables 1 and 2). Besides there is no need to compute the Cholesky decomposition of $C_{\mathbf{uu}}$ to invert it (see last line Table 1). Lastly, in SigGPDE the inducing variables do not depend on any variational parameter (see last line Table 2).

Operation	SigGPDE (ours)	GPSig-IT	GPSig-IS
$C_{\mathbf{uu}}$	$\mathcal{O}(1)$	$\mathcal{O}(n^2M^2d)$	$\mathcal{O}((n+d)M^2\tilde{\ell}^2)$
$C_{\mathbf{fu}}$	$\mathcal{O}(\tilde{N}M\ell)$	$\mathcal{O}(n^2\tilde{N}M\ell d)$	$\mathcal{O}((n+d)\tilde{N}M\tilde{\ell}\ell)$
$\mathrm{diag}(C_{\mathbf{ff}})$	$\mathcal{O}(d\tilde{N}\ell^2)$	$\mathcal{O}((n+d)\tilde{N}\ell^2)$	$\mathcal{O}((n+d)\tilde{N}\ell^2)$
Lin. Alg.	$\mathcal{O}(\tilde{N}M^2)$	$\mathcal{O}(\tilde{N}M^2+M^3)$	$\mathcal{O}(\tilde{N}M^2+M^3)$

Table 1: Comparison of time complexities. M is the number of inducing variables, $\tilde{N}$ the batch size, d the number of channels in the time series, ℓ the length of the sequences, n the truncation level (for GPSig-IT and GPSig-IS) and $\tilde{\ell}$ the length of the inducing sequences.

Operation	SigGPDE (ours)	GPSig-IT	GPSig-IS
$C_{\mathbf{uu}}$	N/A	$\mathcal{O}(n^2M^2)$	$\mathcal{O}(M^2\tilde{\ell}^2)$
$C_{\mathbf{fu}}$	$\mathcal{O}(\tilde{N}M\ell)$	$\mathcal{O}(n^2\tilde{N}M\ell)$	$\mathcal{O}(\tilde{N}M\ell\tilde{\ell})$
$\mathrm{diag}(C_{\mathbf{ff}})$	$\mathcal{O}(\tilde{N}\ell^2)$	$\mathcal{O}(\tilde{N}\ell^2)$	$\mathcal{O}(\tilde{N}\ell^2)$
$\mathbf{z}$	N/A	$\mathcal{O}(n^2Md)$	$\mathcal{O}(M\tilde{\ell}d)$

Table 2: Comparison of space complexities, separated by algorithm to compute each covariance matrix. The last line accounts for the storage of the inducing tensors and inducing sequences in GPSig-IT and GPSig-IS.

C.2. Computing the signature kernel and its gradients

Recall that the signature kernel solves the following PDE,

$$\frac{\partial^2 U}{\partial s \partial t} = (\dot{X}_s^T \dot{Y}_t)U \qquad U(0,\cdot)=1,\ U(\cdot,0)=1 \tag{9}$$

therefore each kernel evaluation amounts to a call to a PDE solver. Using a straightforward implementation of a finite-difference PDE solver which consists in applying an update of the form

$$U(s_i,t_j) = g(U(s_{i-1},t_{j-1}), U(s_i,t_{j-1}), U(s_{i-1},t_j)),$$

in row or column order, the time complexity for N kernel evaluations for time series with d channels of length ℓ is $\mathcal{O}(dN\ell^2)$. Indeed there is no data dependencies between each of the N kernel evaluations, hence we can solve each PDE in parallel. But, this does not reduce the quadratic complexity with respect to the length ℓ. However, it is possible to parallelize the PDE solver by observing that instead of solving the PDE in row or column order, we can update the antidiagonals of the solution grid. As illustrated

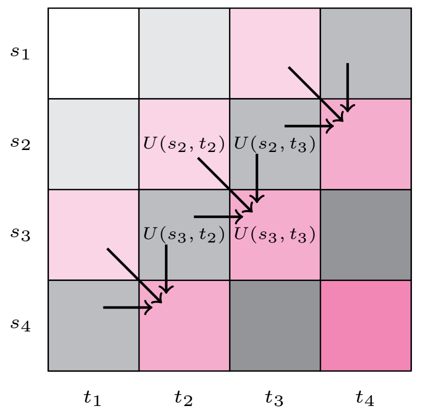

Figure 1: Parallelization of the finite-difference scheme. Each cell on an antidiagonal can be computed in parallel, provided the previous antidiagonals have been computed.

on Fig. 1, each cell on an antidiagonal can be updated with in parallel as there is no data dependency between them. Therefore, we propose a CUDA implementation where N collections of $2\ell - 1$ threads (the number of cells on the biggest antidiagonal) running in parallel can simultaneosly update an antidiagonal of the solution grids.

To compute the gradients, we use the result from Thm. 4.1. During the forward pass we solve the PDEs defined by the input time series using the CUDA operator described above. For the backward pass, we first solve the PDEs with the input time series reversed in time, by calling the same CUDA operator. Second, we compute the gradients using simple vectorized TensorFlow operations.

References

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

Dozat, T. Incorporating nesterov momentum into adam. 2016.

Lyons, T. J., Caruana, M., and Lévy, T. *Differential equations driven by rough paths*. Springer, 2007.

Tavenard, R., Faouzi, J., Vandewiele, G., Divo, F., Androz, G., Holtz, C., Payne, M., Yurchak, R., Rußwurm, M., Kolar, K., and Woods, E. Tslearn, a machine learning toolkit for time series data. *Journal of Machine Learning Research*, 21(118):1–6, 2020. URL `http://jmlr.org/papers/v21/20-091.html`.

Toth, C. and Oberhauser, H. Bayesian learning from sequential data using gaussian processes with signature covariances. In *International Conference on Machine Learning*, pp. 9548–9560. PMLR, 2020.