Appendix

A The signature transform

For completeness, we will give an elementary introduction to the *signature transform*, already defined in Section 2.1, and we will briefly see many of the properties that make it such a convenient transformation in machine learning.

A.1 Paths of finite p-variation

Definition A.1 (p-variation)**.** *Let $p \geq 1$ be a real number, and $\|\cdot\|$ any norm on $\mathbb{R}^d$. Let $x : [0, T] \to \mathbb{R}^d$ be a continuous path. The p-variation of x is defined as*

$$\|x\|_p = \left[\sup_{\mathcal{D}} \sum_{i=0}^{n-1} ||x_{t_{i+1}} - x_{t_i}||^p \right]^{1/p},$$

where the supremum is taken over the set of partitions of $[0, T]$, denoted as $\mathcal{D}$. The space of d-dimensional continuous paths of finite p-variation will be denoted as $\mathcal{V}^p([0, T]; \mathbb{R}^d)$.

We will equip $\mathcal{V}^p([0, T]; \mathbb{R}^d)$ with the following norm.

Definition A.2 (p-variation norm)**.** *The p-variation norm of a path $x \in \mathcal{V}^p([0, T]; \mathbb{R}^d)$ is defined as*

$$\|x\|_{p-var} = \|x\|_p + \sup_{t \in [0,T]} \|x_t\|,$$

where $\|\cdot\|$ is any norm in $\mathbb{R}^d$.

For simplicity, we will only work with paths of finite 1-variation, which are called of bounded variation. The reason for this is that in practice we will always be given a discrete stream of data, which can be converted into a continuous path by performing some interpolation scheme. The most common one is linear interpolation (with the resulting paths being of bounded variation), since then the signature is very easy to compute. This is what the Signatory package (Kidger & Lyons, 2021) does, providing differentiable computations of the signature on the GPU.

Moreover, to work with signatures of general paths of finite p-variation, we would need to introduce many new concepts of rough path theory, which is beyond the scope of this paper. However, we remark that all the properties of the signature that we will explain also hold (in some sense) for general continuous paths of finite p-variation.

A.2 The tensor algebra

Definition A.3 (Tensor algebra of $\mathbb{R}^d$)**.** *Let $(\mathbb{R}^d)^{\otimes k}$ denote the kth tensor power of $\mathbb{R}^d$. By convention, $(\mathbb{R}^d)^{\otimes 0} = \mathbb{R}$. We define the extended tensor algebra of $\mathbb{R}^d$ as*

$$T((\mathbb{R}^d)) = \{\boldsymbol{a} = (a_0, a_1, ...) \mid \forall k \geq 0, a_k \in (\mathbb{R}^d)^{\otimes k}\}.$$

We also define the truncated tensor algebra of order N of $\mathbb{R}^d$, which is a linear subspace of $T((\mathbb{R}^d))$, as

$$T^{(N)}(\mathbb{R}^d) = \{\boldsymbol{a} = (a_0, a_1, ..., a_N) \mid \forall k : 0 \leq k \leq N, a_k \in (\mathbb{R}^d)^{\otimes k}\}.$$

Note that $T^{(N)}(\mathbb{R}^d)$ is a real vector space of dimension

$$\sum_{k=0}^{N} d^k = \begin{cases} N+1 & \text{if } d = 1 \\ \frac{d^{N+1}-1}{d-1} & \text{if } d > 1. \end{cases}$$

We will equip $T((\mathbb{R}^d))$ with an admissible norm, defined as follows.

Definition A.4. *We say that the extended tensor algebra $T((\mathbb{R}^d))$ is endowed with an admissible norm $\|\cdot\|$ if the following conditions hold:*

1. *For each $k \geq 1$, the symmetric group $\mathcal{S}_k$ acts by isometry on $(\mathbb{R}^d)^{\otimes k}$, i.e.*

$$\|\sigma v\| = \|v\|, \quad \forall v \in (\mathbb{R}^d)^{\otimes k}, \ \forall \sigma \in \mathcal{S}_k.$$

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

$$\|v \otimes w\| \leq \|v\| \, \|w\|, \quad \forall v \in (\mathbb{R}^d)^{\otimes n}, \ w \in (\mathbb{R}^d)^{\otimes m}.$$

A.3 Properties of the signature of a path

In this section we go through some of the main properties of the signature of a path, defined in Definition 2.1.

One of its most important properties is its uniqueness up to time-reparametrizations and translations. Specifically, we have the following result (Hambly and Lyons, 2010).

Lemma A.1. *Let $\mathcal{A}$ be a set of continuous paths with bounded variation such that all paths have the same initial value and have at least one monotone coordinate. Then the signature of a path in $\mathcal{A}$ completely determines it in $\mathcal{A}$.*

In order to fulfill the requirements of the last lemma, the two most common injective transformations that are applied to the original path are:

1. **Basepoint augmentation:** concatenation of x_t with the linear path defined in $[-1, 0]$ with initial value $0 \in \mathbb{R}^d$ and final value x_0, resulting in

$$x_t^* = \begin{cases} (t+1)x_0 & t \in [-1, 0), \\ x_t & t \in [0, T]. \end{cases}$$

2. **Time augmentation:** append time as an extra channel, $\hat{x}_t^* = (t, x_t^*)$.

In conclusion, after applying a couple of simple transformations, the signature provides a perfect codification for any continuous path of bounded variation. We will denote the space obtained when applying this two transformations to the paths in $\mathcal{V}^1([0, T]; \mathbb{R}^d)$ as $\mathcal{A}^1([0, T]; \mathbb{R}^d)$.

As we already explain in Section 2.1, in practice one is not able to compute all the levels of the signature of a path. This is the reason for defining the truncated signature, which we did in Definition 2.2.

It turns out that selecting the first levels of the signature is a good approximation, since they decay in size factorially.

Proposition A.1 (Lyons et al., 2004, Proposition 2.2)**.** *Let $x \in \mathcal{V}^1([0, T]; \mathbb{R}^d)$. Let $\mathrm{sig}_k(x)$ denote the kth level of the signature of x. Then, for any $k \in \mathbb{N}$ we have that*

$$\|\mathrm{sig}_k(x)\| \leq \frac{\|x\|_{1-var}^k}{k!}.$$

A direct consequence is that one is able to approximate the signature arbitrarily well by the sequence of truncated signatures.

Corollary A.1. *Let $x \in \mathcal{V}^1([0, T]; \mathbb{R}^d)$. Then, for every $\epsilon > 0$, there exists an $N \in \mathbb{N}$ such that*

$$\left\|\mathrm{sig}(x) - \mathrm{sig}^N(x)\right\| \leq \epsilon.$$

Moreover, if we restrict ourselves to a compact subset $K \subset \mathcal{V}^1([0, T]; \mathbb{R}^d)$, then the convergence is uniform.

Another very important feature of the signature is the universal non-linearity property, which states that the signature provides a basis for the space of continuous functions on path space.

Theorem A.2 (Universal non-linearity, Arribas, 2018, Theorem 4.2)**.** *Let $K \subset \mathcal{A}^1([0, T]; \mathbb{R}^d)$ be a compact subset. Then*

$$\bigcup_{N \in \mathbb{N}} \left\{ x \mapsto \ell(\mathrm{sig}^N(x)) \mid \ell : T^{(N)}(\mathbb{R}^{d+1}) \to \mathbb{R}^v \text{ is linear} \right\}$$

is uniformly dense in the space of continuous maps from K to $\mathbb{R}^v$, denoted by $\mathcal{C}(K; \mathbb{R}^v)$.

A.4 The expected signature of a stochastic process

The last property that we will see states that the expected signature of a stochastic process that has compact support completely determines its distribution.

Theorem A.3 (Fawcett, 2003). *Let $\{X_t\}_{t\in[0,T]}$ and $\{Y_t\}_{t\in[0,T]}$ be two stochastic processes with compact support $K \subset \mathcal{A}^1([0,T];\mathbb{R}^d)$. Then, we have that*

1. *The expected signatures are well defined, $\mathbb{E}_{x\sim X}[\mathrm{sig}(x)]$, $\mathbb{E}_{y\sim Y}[\mathrm{sig}(y)] < \infty$.*
2. *If $\mathbb{E}_{x\sim X}[\mathrm{sig}(x)] = \mathbb{E}_{y\sim Y}[\mathrm{sig}(y)]$, then X and Y are equal in the distribution sense. In other words, they have the same law.*

B The Signature-Wasserstein-1 metric

We first recall the definition of the Wasserstein-1 distance between distributions defined on the same measurable space.

Definition B.1 (Wasserstein-1 distance). *Let $\Pi(\mu,\nu)$ denote the set of all joint distributions on the measurable space $(\mathcal{X}\times\mathcal{X}, \mathcal{F}\otimes\mathcal{F})$ whose marginals are respectively μ and ν. Then the Wasserstein-1 distance is defined as*

$$W_1(\mu,\nu) := \inf_{\gamma\in\Pi(\mu,\nu)} \mathbb{E}_{(x,y)\sim\gamma}[||x-y||_1]. \tag{3}$$

In practice the infimum in (3) is highly intractable. Luckily for us, the Kantorovich-Rubinstein duality (Villani, 2009) states that the Wasserstein-1 distance can be calculated as

$$W_1(\mu,\nu) = \sup_{\|F\|_L\leq 1} \Big\{\mathbb{E}_{x\sim\mu}[F(x)] - \mathbb{E}_{x\sim\nu}[F(x)]\Big\}, \tag{4}$$

where the supremum is over all the 1-Lipschitz functions $F : \mathcal{X}\to\mathbb{R}$. The most common approach is to approximate F by a parameterized family of functions that are 1-Lipschitz, like in the Wasserstein-GAN setting (Arjovsky et al., 2017), which we briefly explain now.

Let Z be a random variable taking values on a space $\mathcal{Z}$. Let $G_\theta : \mathcal{Z}\to\mathcal{X}$ be a map parameterized by θ. Consider the pushforward distribution in $\mathcal{X}$ that is given by $G_\theta(Z)$, which we denote as $\mathbb{P}_\theta$, and let $\mathbb{P}_r$ denote the distribution of the data. Then, our goal is to minimize an approximation of the Wasserstein-1 distance between $\mathbb{P}_\theta$ and $\mathbb{P}_r$.

In the Wasserstein GAN approach, the optimization problem that one tries to solve is

$$\min_{\theta\in\Theta}\max_{\psi\in\Psi} \Big\{\mathbb{E}_{x\sim\mathbb{P}_r}[F_\psi(x)] - \mathbb{E}_{z\sim Z}[F_\psi(G_\theta(z))]\Big\},$$

where $\{F_\psi\}_{\psi\in\Psi}$ is a family of 1-Lipschitz functions. The training algorithm consists in performing k steps on the parameters on the critic F_ψ, and once a good enough approximation of the Wasserstein-1 distance has been reached, perform one step on the parameters of the

generator G_θ. However, this method is very unstable and sensitive to the choice of the hyperparameters.

Remember that, in our case, we are interested in approximating distributions in path space, such that $\mathcal{X} = \mathcal{A}^1([0,T];\mathbb{R}^d)$. Let $K \subseteq \mathcal{X}$ denote the union of the supports of μ and ν. By definition of (4), there exists a sequence $F_n : K \to \mathbb{R}$ of functions that are 1-Lipschitz such that they attain the supremum $W_1(\mu,\nu)$.

If we assume that $K \subset \mathcal{X}$ is a compact subset, by the universal non-linearity property of the signature we have that, $\forall \epsilon > 0$ and for each F_n there exists a linear functional $\ell_n : T((\mathbb{R}^{d+1})) \to \mathbb{R}$ such that

$$\sup_{x \in K} |F_n(x) - \ell_n(\text{sig}(x))| \leq \epsilon.$$

For this reason, Ni et al., 2020 and Ni et al., 2021 propose the use of the sig-W_1 metric, defined as

$$\begin{aligned} \text{sig-}W_1(\mu,\nu) &= \sup_{\|\ell\|_L \leq 1} \Big\{ \mathbb{E}_{x\sim\mu}[\ell(\text{sig}(x))] - \mathbb{E}_{x\sim\nu}[\ell(\text{sig}(x))] \Big\} \\ &= \sup_{\|\ell\|_L \leq 1} \ell\Big(\mathbb{E}_{x\sim\mu}[\text{sig}(x)] - \mathbb{E}_{x\sim\nu}[\text{sig}(x)] \Big), \end{aligned} \tag{5}$$

where the expected signature is well defined, since we assumed that both measures have a compact support. Moreover, by using the truncated signatures and Corollary A.1 we can approximate (5) by

$$\text{sig}^N\text{-}W_1(\mu,\nu) = \sup_{\|\ell\|_L \leq 1} \ell\Big(\mathbb{E}_{x\sim\mu}[\text{sig}^N(x)] - \mathbb{E}_{x\sim\nu}[\text{sig}^N(x)] \Big), \tag{6}$$

where now the linear functionals ℓ are defined in the truncated tensor algebra $T^{(N)}(\mathbb{R}^{d+1})$. Since the domain space is finite dimensional, we have that the Lipschitz constant of a linear functional $\ell : T^{(N)}(\mathbb{R}^{d+1}) \to \mathbb{R}$ is simply defined as the norm of its coefficients as an euclidean vector.

It turns out that when we choose this to be the l_2 norm, we have that the optimization problem (6) admits a closed-form solution (Ni et al., 2020, Lemma A.3), with the solution being

$$\text{sig}^N\text{-}W_1(\mu,\nu) = \left\| \mathbb{E}_{x\sim\mu}[\text{sig}^N(x)] - \mathbb{E}_{x\sim\nu}[\text{sig}^N(x)] \right\|_2,$$

which is called the truncated Signature-Wasserstein-1 metric of order N. In the GAN setting, we can approximate the expected signatures of the model and the data by performing a Monte Carlo simulation, i.e. by computing the empirical averages of the signatures.

C The Conditional Signature-Wasserstein GAN approach

What if we have some knowledge that we want to use to condition this distribution? Let $X = \{X_t\}_{t\in[0,T_x]}$ and $Y = \{Y_t\}_{t\in[0,T_y]}$ be two stochastic processes taking values on

$$\mathcal{X} = \mathcal{A}^1([0,T_x];\mathbb{R}^{d_x}), \qquad \mathcal{Y} = \mathcal{A}^1([0,T_y];\mathbb{R}^{d_y}).$$

Let $p(X,Y)$ denote their joint distribution. Then, given any known trajectory $x \in \mathcal{X}$, we are interested in approximating the conditional distribution $p(Y|X=x)$, which we will compactly denote by $Y|x$.

In practice, we cannot approximate the expected signature of $Y|x$ by a Monte Carlo procedure, since for every x we are usually given only one sample. Therefore we must come up with a different method to estimate $\mathbb{E}_{y\sim Y|x}[\mathrm{sig}(y)]$.

Just like in supervised learning[3], we can formulate this problem as trying to approximate a map

$$\begin{array}{rccc} f\colon & \mathcal{X} & \to & \mathcal{P}(\mathcal{Y}) \\ & x & \mapsto & Y|x, \end{array} \tag{7}$$

where $\mathcal{P}(\mathcal{Y})$ is the space of all stochastic processes taking values on $\mathcal{Y}$. If we restrict ourselves to the family of stochastic processes with compact support, by Theorem A.3 and Lemma A.1 there will exist a unique function $\hat{f} : T((\mathbb{R}^{d_x+1})) \to T((\mathbb{R}^{d_y+1}))$ such that

$$\hat{f}(\mathrm{sig}(x)) = \mathbb{E}_{y\sim Y|x}[\mathrm{sig}(y)], \tag{8}$$

where we wrote $f(x) = Y|x$. If we assume that the domain of f is a compact subset, we can use the uniform convergence of the truncated signatures to approximate (8) by

$$\begin{array}{rccc} \hat{f}_{N,M}\colon & K \subset T^{(N)}(\mathbb{R}^{d_x+1}) & \to & T^{(M)}(\mathbb{R}^{d_y+1}) \\ & \mathrm{sig}^N(x) & \mapsto & E_{y\sim Y|x}[\mathrm{sig}^M(y)]. \end{array}$$

The image space of $\hat{f}_{N,M}$ is real and has finite dimension, so one can think of it as an euclidean space. Therefore we can use the universal non-linearity of the signature to approximate $\hat{f}_{N,M}$ by a linear functional $\ell_{N,M}$,

$$\begin{array}{rccc} \ell_{N,M}\colon & K \subset T^{(N)}(\mathbb{R}^{d_x+1}) & \to & T^{(M)}(\mathbb{R}^{d_y+1}) \\ & \mathrm{sig}^N(x) & \mapsto & E_{y\sim Y|x}[\mathrm{sig}^M(y)]. \end{array} \tag{9}$$

In conclusion, we have reduced a non-linear modelling problem between two continuous and infinite dimensional spaces in (7) to a linear problem between two discrete and finite dimensional spaces. Moreover, the function $\ell_{N,M}$ represents a conditional expectation, and so we can use the classical linear regression framework to approximate it.

[3]The difference with respect to supervised learning is that in this case we are interested in the conditional distribution, and not its expectation.

This is, given some data $\{x^{(i)}, y^{(i)}\}_{i=1}^{n}$, we will assume that

$$\text{sig}^M(y^{(i)}) = W \cdot \text{sig}^N(x^{(i)}) + \epsilon_i,$$

where

- $\text{sig}^M(y^{(i)})$ can be viewed as an $\frac{(d_y+1)^M-1}{d_y}$ dimensional real vector.
- $\text{sig}^N(x^{(i)})$ can be viewed as an $\frac{(d_x+1)^N-1}{d_x}$ dimensional real vector.
- W is a real matrix of dimension $\frac{(d_y+1)^M-1}{d_y} \times \frac{(d_x+1)^N-1}{d_x}$.
- $\epsilon_i \sim \mathcal{N}(0, \sigma^2 I)$ is i.i.d. Gaussian noise, with I denoting the identity matrix.

and use any standard libraries to compute the weight matrix W.

In practice, we will proceed as follows. Let Z be a random variable taking values on a space $\mathcal{Z}$. Let $G_\theta : \mathcal{Z} \times \mathcal{X} \to \mathcal{Y}$ be a map parameterized by θ. Then, for a known $x \in \mathcal{X}$, we will consider the pushforward conditional distribution in $\mathcal{Y}$ that is given by $G_\theta(Z, x)$. Just like in the unconditional case, the expected signature of $G_\theta(Z, x)$ is estimated through a Monte Carlo procedure.

In order to estimate the expected signature of $Y|x$ from the set of given data, we simply perform the linear regression explained earlier, from which we obtain an estimate of (9), that we denote by $\hat{\ell}$. Then we can obtain an approximation of $\mathbb{E}_{y \sim Y|x}[\text{sig}(y)]$ by

$$\hat{\mathbb{E}}_{y \sim Y|x}[\text{sig}^M(y)] = \hat{\ell}(\text{sig}^N(x)).$$

D Experimental Details

The experimental details that were used in Section 4, such as the model architectures and the selected hyperparameters, can be found in the following pages.

D.1 Architectures for the resources cost analysis

LSTM Conditional WGAN The LSTMs of the conditional generator had 5 hidden layers of width 32, with 5 dimensional random noise, and had 77,089 learnable parameters. The LSTMs of the critic also had 5 hidden layers of width 32, with 76,609 learnable parameters.

LSTM Conditional Sig-WGAN The architecture of the generator was the same as the one we just described for the LSTM Conditional WGAN generator. The signature transform was truncated at depth 5 and 4, for the input and output paths, respectively. We also applied

the cumulative sum transformation[4]. Overall, the dimension of both truncated signatures was 363 and 120, respectively.

Neural SDE Conditional Sig-WGAN The number of hyperparameters in a conditional Neural SDE is slightly larger. Since it is a little bit hard to keep track of all of them, we will mention them along the corresponding notation we used in Definition 3.1.

The size d_z of the Neural SDE was 92. The initial random noise size d_v was 16 and the network ξ_θ^2 was formed by one hidden layer of size 32. The network ξ_θ^1 was simply a linear function. The initial condition of the SDE was formed by a random part k of size 8 and a deterministic part $d_z - k$ of size 84. The diffusion g_θ was chosen to be of general type. The dimension of the Wiener process d_w was 10. Both the drift f_θ and diffusion g_θ had a hidden layer of width 32. The only activation function we used was the hyperbolic tangent, tanh. The total number of learnable parameters was 79,273.

For the Neural SDE the signatures in the SigCWGAN algorithm were of the same order and with the same transformations as the LSTM Conditional Sig-WGAN model. The transformation $\phi(x)$ of the input paths x was set to be the signature transform, with the same truncation order and transformation the ones used in the SigCWGAN algorithm.

D.2 Architectures and hyperparameters for the experiments

In this section we will detail the architectures and hyperparameters that were used for each experiment in Section 4.2, which were selected by performing an informal grid search.

First we will list the ones that were common to all the datasets.

D.2.1 Common hyperparameters

The optimizers that were used were as follows: for the LSTM CWGAN, following Arjovsky et al., 2017 we used RMSprop. For the models trained with the SigCWGAN algorithm, we used Adam. The learning rate was always set to 10^{-3}.

In all the univariate time series experiments the signature transform was truncated at depth 5 and 4, for the input and output paths, respectively. For the bivariate time series experiment, the signature was truncated at depth 4 for both the input and output paths.

We also applied the cumulative sum transformation, and we normalized each dimension so that it had zero mean and unit variance. The transformation $\phi(x)$ of the Neural SDEs was set to be the signature transform, with the same truncation order and transformation the ones used in the SigCWGAN algorithm.

[4]Although the truncated signatures approximate arbitrarily well the signature transform, this does not mean that some crucial information might be lost by not computing higher levels. In order to mitigate this, usually some transformations are applied to the given stream, see Morrill et al., 2020.

D.2.2 AR(p) dataset

The training set was formed by 14,900 pairs of time series data (x, y). The validation set was formed by 2,400 samples. The test set was formed by 12,400 samples. We normalized the data with the mean and standard deviation of the input streams in the training set.

The architectures and other hyperparameters for each model were as follows.

LSTM WGAN The LSTMs in the generator had 5 layers with hidden size equal to 32, with 5 dimensional random noise. The LSTMs in the critic had 4 layers with hidden size 32. The number of parameters of the generator was 77,089, while the number of parameters of the critic was 59,713.

LSTM Sig-WGAN The generator was the same as in the LSTM WGAN model.

Neural SDE Sig-WGAN The size d_z of the Neural SDE was 48. The initial random noise size d_v was 16 and the network ξ_θ^2 was formed by one hidden layer of size 32. The network ξ_θ^1 was simply a linear function.

The initial condition of the SDE was formed by a random part k of size 16 and a deterministic part $d_z - k$ of size 32. The diffusion g_θ was chosen to be of diagonal type, and therefore the dimension of the Wiener process d_w was the same as the size of the solution, 48. Both the drift f_θ and diffusion g_θ had one hidden layer of width 84. The only activation function we used was the hyperbolic tangent, tanh. Following Kidger, Foster, Li, Oberhauser, et al., 2021, we also applied a final tanh to the vector fields. The total number of learnable parameters was 29,161.

For the LSTM WGAN and the Neural SDE Sig-WGAN models, the batch size was set to 528. Due to memory constrains, for the LSTM Sig-WGAN the batch size was set to 228.

D.2.3 Seattle Weather dataset

The train set was formed by 15,122 pairs of time series data (x, y). The validation set was formed by 3,781 samples. The test set was formed by 6,468 samples. The test set was extracted from a different time period than the train and validation sets (out-of-time samples). We normalized the data with the mean and standard deviation of the input streams in the training set.

The architectures and other hyperparameters for each model were as follows.

LSTM WGAN The LSTMs in the generator had 5 layers with hidden size equal to 32, with 5 dimensional random noise. The LSTMs in the critic had 4 layers with hidden size 36. The number of parameters of the generator was 77,089, while the number of parameters of the critic was 75,241.

LSTM Sig-WGAN The generator was the same as in the LSTM WGAN model.

Neural SDE Sig-WGAN The architecture was the same as the one we detailed in Section D.2.2, with the exception that we set the hidden size of the SDE to $d_z = 64$. Just like in the previous experiment we applied a final tanh to the vector fields of the SDE. The total

number of parameters was 35,113.

For the Neural SDE Sig-WGAN models, the batch size was set to 724. For the LSTM WGAN, it was set to 528. Due to memory constrains, for the LSTM Sig-WGAN the batch size was set to 228.

D.2.4 Forex dataset

The train set was formed by 15,000 pairs of time series data (x, y). The validation set was formed by 5,000 samples. The test set was formed by 10,000 samples. The test set was extracted from a different time period than the train and validation sets (out-of-time samples). As it is usually done in this kind of datasets, we applied the logarithm transformation to the data. After that, we normalized the data with the mean and standard deviation of the input streams in the training set.

The architectures and other hyperparameters for each model were as follows.

LSTM WGAN The architecture of the generator was the same as the one we detailed in the Seattle Weather dataset experiment. However, in this case the architecture of both LSTMs of the critic was set to hidden size equal to 32 and a number of layers equal to 5. The number of parameters of the critic was 234,113.

LSTM Sig-WGAN The generator was the same as in the LSTM WGAN model.

Neural SDE Sig-WGAN The architecture was the same as the one we detailed in the Seattle Weather dataset experiment.

For the LSTM WGAN and the Neural SDE Sig-WGAN, the batch size was set to 528. Due to memory constrains, for the LSTM Sig-WGAN the batch size was set to 128.

D.2.5 IBEX35 dataset

The train set was formed by 4,296 pairs of time series data (x, y). The validation set was formed by 1,074 samples. The test set was formed by 1,944 samples. The test set was extracted from a different time period than the train and validation sets (out-of-time samples). We normalized the data with the mean and standard deviation of the input streams in the training set.

The architectures and other hyperparameters for each model were as follows.

LSTM WGAN The LSTMs in the generator had 2 layers with hidden size equal to 64, with 5 dimensional random noise. The LSTMs in the critic had 4 layers with hidden size 32. The number of parameters of the generator was 101,953, while the number of parameters of the critic was 59,713.

LSTM Sig-WGAN The generator was the same as in the LSTM WGAN model.

Neural SDE Sig-WGAN The size d_z of the Neural SDE was 120. The initial random noise size d_v was 16 and the network ξ_θ^2 was formed by one hidden layer of size 48. The network ξ_θ^1 was simply a linear function.

The initial condition of the SDE was formed by a random part k of size 56 and a deterministic part $d_z - k$ of size 64. The diffusion g_θ was chosen to be of diagonal type, and therefore the dimension of the Wiener process d_w was the same as the size of the solution, 64. Both the drift f_θ and diffusion g_θ had one hidden layer of width 96. The only activation function we used was the hyperbolic tangent, tanh. Just like in the previous experiments we applied a final tanh to the vector fields of the SDE. The total number of learnable parameters was 73,489.

For the LSTM WGAN and the Neural SDE Sig-WGAN, the batch size was set to 1024. Due to memory constrains, for the LSTM Sig-WGAN the batch size was set to 528.

D.2.6 IBEX35 & CAC40 dataset

The train set was formed by 4,262 pairs of time series data (x, y). The validation set was formed by 1,066 samples. The test set was formed by 1,231 samples. The test set was extracted from a different time period than the train and validation sets (out-of-time samples). We normalized the data with the mean and standard deviation of the input streams in the training set.

The architectures and other hyperparameters for each model were as follows.

LSTM WGAN The LSTMs in the generator had 2 layers with hidden size equal to 64, with 5 dimensional random noise. The LSTMs in the critic had 4 layers with hidden size 32. The number of parameters of the generator was 102,274, while the number of parameters of the critic was 59,969.

LSTM Sig-WGAN The generator was the same as in the LSTM WGAN model.

Neural SDE Sig-WGAN The size d_z of the Neural SDE was 80. The initial random noise size d_v was 16 and the network ξ_θ^2 was formed by one hidden layer of size 24. The network ξ_θ^1 was simply a linear function.

The initial condition of the SDE was formed by a random part k of size 30 and a deterministic part $d_z - k$ of size 50. The diffusion g_θ was chosen to be of diagonal type, and therefore the dimension of the Wiener process d_w was the same as the size of the solution, 80. Both the drift f_θ and diffusion g_θ had one hidden layer of width 56. The only activation function we used was the hyperbolic tangent, tanh. Just like in the previous experiments we applied a final tanh to the vector fields of the SDE. The total number of learnable parameters was 58,481.

For the LSTM WGAN, the batch size was set to 1024. For the Neural SDE Sig-WGAN, it was set to 528. Due to memory constrains, for the LSTM Sig-WGAN the batch size was set to 400.

E Computing infrastructure

All experiments were run on a computer that had Windows 11 Home as the operative system, equipped with an AMD Ryzen 7 5800X 8-Core Processor, 16GB of RAM and an Nvidia GeForce RTX 3060 with 12GB of memory.

The main python libraries that were used are listed below:

- Pytorch 1.9.0+cu111 as the main deep learning framework (Paszke et al., 2019).
- Signatory 1.2.6, which provides differentiable computations of the signature on the GPU (Kidger & Lyons, 2021).
- torchsde 0.2.5, which provides stochastic differential equation (SDE) solvers with GPU support and efficient backpropagation (Li, 2020).

We highlight that, in order to use the Signatory package, one needs to have a Pytorch version that is no older than 1.9.0.