A. Experimental details

A.1. General notes

Sample code A straightforward pedagogical implementation has been made available as an example in the `torchsde` library (Li, 2020).

Software We used PyTorch (Paszke et al., 2019) as an autodifferentiable framework. We used the `torchsde` library (Li, 2020) to solve SDEs. We used the Signatory library (Kidger & Lyons, 2021) to calculate the signatures used in the MMD metric. We used the `torchcde` library (Kidger, 2020) for its interpolation schemes, and to solve the neural CDEs used in the classification and prediction metrics. We used the `torchdiffeq` library (Chen, 2018) to solve the neural ODEs used in the classification and prediction metrics, and for the ODE components of the Latent ODE and CTFP models.

Computing infrastruture Training was performed on computers using Ubuntu 18.04 LTS, across a mix of five GeForce RTX 2080 Ti and two Quadro GP100; each experiment was performed on a single GPU. Training time varied by problem; none took more than a week per experiment.

SDE solvers The SDEs used the midpoint method. Recall that the target time series data was regularly sampled and linearly interpolated to make a path. We took the SDE solver to take a single step between each output data point.

ODE solvers All ODEs were solved using the midpoint method. (For consistency with the solvers used to train the SDE, for example to ensure that the discriminator's action on real data is similar to the discriminator's action on generated data.)

CDE solvers The CDEs of the classification and prediction models were solved by reducing to ODEs as in Kidger et al. (2020).

Normalisation All data was normalised to have zero mean and unit variance.

Architectures To recap, the neural SDE has generator initial condition ζ_θ, generator drift μ_θ, generator diffusion σ_θ, discriminator initial condition ξ_ϕ, discriminator drift f_ϕ, and discriminator diffusion g_ϕ. All of these are parameterised as neural networks.

Meanwhile Latent ODEs have an ODE-RNN encoder (with a neural network vector field) and a neural ODE decoder (with a neural network vector field). The CTFP has an ODE-RNN encoder (with a neural network vector field) and a continuous normalising flow (Chen et al., 2018; Grathwohl et al., 2019) (with a neural network vector field) Additionally Deng et al. (2020) condition the normalising flow on the time evolution of a neural ODE of some latent state, which requires another neural network vector field.

Hyperparameters were selected according to informal hyperparameter optimisation across all models.

For the stocks, air quality, and weights datasets (but not the time-dependent Ornstein–Uhlenbeck process, which was done separately and is described below), every neural network was parameterised as a feedforward network with 2 hidden layers, width 64, and softplus activations. The drift, diffusion and vector fields, for every model, all additionally had a tanh nonlinearity as their final operation. As described in the main text we found that this improved the performance of every model.

The neural SDE's generator has hidden state of size x and the discriminator has hidden state is of size h. These were both taken as $x = h = 96$. Note that this is larger than the width of each hidden layer within the neural networks, so that the first operation within each neural network is a map from $\mathbb{R}^{96} \to \mathbb{R}^{64}$. Somewhat anecdotally, we found that taking the state to be larger than the hidden width was beneficial for model performance.[2]

The Latent ODE likewise has evolving hidden state, which was also taken to be of size 96.

The Latent ODE samples noise from a normally distributed initial condition, which we took to have 40 dimensions. The CTFP samples noise from a Brownian motion, which as a continuous normalising flow has dimension equal to the number of dimensions of target distribution.

The neural SDE samples noise from both a normally distributed initial condition and a Brownian motion. For the stocks, air quality and weights datasets (but not the time-dependent Ornstein–Uhlenbeck process, which was done separately and is described below), we took the initial condition to have 40 dimensions. The number of dimensions of the Brownian motion was dataset dependent, see below.

The CTFP included a latent context vector as described in Deng et al. (2020). This was taken to have 40 dimensions.

Optimisers The CTFP and Latent ODE were both trained with Adam (Kingma & Ba, 2015) with a learning rate of 4×10^{-5}. The generator and discriminator of the neural SDE were trained with Adadelta with a learning rate

[2]This has some loose theoretical justification: a signature is a linear differential equation with very large state, and it is a universal approximator. (See Kidger et al. (2020, Appendix B) and references within – this is a classical fact within rough analysis.) That is to say, it is a simple vector field with a large state, rather than a complicated vector field with a small state.

of 4×10^{-5}. The learning rates were chosen by starting at 4×10^{-4} (arbitrarily) and reducing until good performance was achieved. (In particular seeking to avoid oscillatory behaviour in training of the neural SDE.)

Training For the stocks, air quality and weights datasets (but not the time-dependent Ornstein–Uhlenbeck process, which was done separately and is described below), every model was trained for 100 epochs. The discriminator of the neural SDE received five training steps for every step with which the generator was trained, as is usual; the number of epochs given at 100 is for the generator, for a fair comparison to the other models.

Batch sizes were picked based on what was the largest possible batch size that GPU memory allowed for; these vary by problem and are given below.

Classifier and predictor The classifier was taken to be a neural CDE with hidden state of size 32, and whose vector field was parameterised as a feedforward neural network with 2 hidden layers of width 32, with softplus activations and final tanh activation.

The predictor was taken to be a neural CDE/neural ODE encoder/decoder pair. Both had a hidden state of size 32, and vector fields parameterised as feedforward neural network with 2 hidden layers of width 32, with softplus activations and final tanh activation. 32 dimensions were used at the encoder/decoder interface.

The learning rate used was 10^{-4} for both models, for every dataset and generative model considered, with the one exception of CTFP on Beijing Air Quality, where we observed divergent training of the classifier; the learning rate was reduced to 10^{-5} for this case only.

In all cases they were trained for 50 epochs using Adam, with early stopping if the model failed to improve its training loss over 20 epochs.

The classifier took an 80%/20% train/test split of the dataset given by combining the underlying dataset and model-generated samples of equal size.

A.2. Time-dependent Ornstein–Uhlenbeck process

Each sample is of length 64. The batch size was 1024. The learning rate was 10^{-3} The neural SDE was trained for 6000 steps. (Not epochs.) L^2 weight decay with scaling 0.01 was applied. Weight averaging (over both generator and discriminator) was performed over the final 5500 steps.

The Brownian motion from which noise was sampled has 3 dimensions. The initial noise was sampled over 5 dimensions. The evolving hidden states were taken to have size 32; the the width of each MLP was taken to be 16; each such MLP had a single hidden layer.

A.3. Stocks

Each sample is of length 100.

The batch size was 2048 for every model.

For the neural SDE, the discriminator received 1 epoch of training before the main training (of both generator and discriminator simultaneously) commenced. The weight averaging (over both generator and discriminator) was over every training epoch. The Brownian motion from which noise was sampled had 3 dimensions.

The prediction metric was based on using the first 80% of the input to predict the last 20%.

A.4. Beijing Air Quality

Each sample is of length 24.

The data was normalised to have zero mean and unit variance.

The batch size was 1024 for every model.

For the neural SDE, the discriminator received 10 epochs of training before the main training (of both generator and discriminator simultaneously) commenced. The weight averaging (over both generator and discriminator) was over the final 40 epochs of training. (We realised that this was an obvious improvement over averaging every epoch, as was done for the previous two experiments.) The Brownian motion from which noise was sampled had 10 dimensions.

The prediction metric was based on using the first 50% of the input to predict the last 50%. (An accidental change from the 80%/20% split used in the other experiments; this was kept as it is fair, as it is the same for all models on this dataset.)

A.5. Weights

Each sample is of length 100. Each sample corresponds to the trajectory of a single scalar weight, epoch-by-epoch, as a small convolutional model is trained on MNIST for 50 epochs. Every weight from the network is used, and treated as a separate sample. This is repeated 10 times. If P is the number of parameters in the convolutional network, then the overall size of the dataset is now $(samples = 10P, length = 100, channels = 1)$.

The batch size was 4096 for the neural SDE and latent ODE. This was reduced to 1024 for the CTFP, which we found to be a very memory intensive model on this problem.

For the neural SDE, the discriminator received 10 epochs of training before the main training (of both generator and discriminator simultaneously) commenced. The weight averaging (over both generator and discriminator) was over every training epoch. The Brownian motion from which noise was sampled had 3 dimensions.

The prediction metric was based on using the first 80% of the input to predict the last 20%.