J. Plasma Phys. (2022), *vol.* 88, 895880502
Published by Cambridge University Press

doi:10.1017/S0022377822000769

Data augmentation for disruption prediction via robust surrogate models

Katharina Rath[1,2,†], **David Rügamer**[1,3], **Bernd Bischl**[1], **Udo von Toussaint**[2], **Cristina Rea**[4], **Andrew Maris**[4], **Robert Granetz**[4] and **Christopher G. Albert**[5]

[1]Department of Statistics, Ludwig-Maximilians-Universität München, Germany

[2]Max-Planck-Institut für Plasmaphysik, Garching, Germany

[3]Institute of Statistics, RWTH Aachen University, Germany

[4]Plasma Science and Fusion Center, Massachusetts Institute of Technology, Cambridge, MA, USA

[5]Fusion@OEAW, Institute of Theoretical and Computational Physics, Graz University of Technology, Austria

(Received 18 May 2022; revised 7 August 2022; accepted 8 August 2022)

The goal of this work is to generate large statistically representative data sets to train machine learning models for disruption prediction provided by data from few existing discharges. Such a comprehensive training database is important to achieve satisfying and reliable prediction results in artificial neural network classifiers. Here, we aim for a robust augmentation of the training database for multivariate time series data using Student t process regression. We apply Student t process regression in a state space formulation via Bayesian filtering to tackle challenges imposed by outliers and noise in the training data set and to reduce the computational complexity. Thus, the method can also be used if the time resolution is high. We use an uncorrelated model for each dimension and impose correlations afterwards via colouring transformations. We demonstrate the efficacy of our approach on plasma diagnostics data of three different disruption classes from the DIII-D tokamak. To evaluate if the distribution of the generated data is similar to the training data, we additionally perform statistical analyses using methods from time series analysis, descriptive statistics and classic machine learning clustering algorithms.

Key words: fusion plasma, plasma instabilities

1. Introduction

Disruptions pose serious challenges to the operation and design of tokamaks. Due to rapidly growing instabilities, thermal and magnetic energy is rapidly lost during a disruption, the magnetic confinement of the plasma is destroyed and energy is deposited into the confining vessel, potentially causing serious damages. Hence, to maintain a reliable fusion operation, disruption mitigation mechanisms should be triggered with sufficient warning time prior to the disruption. Recent advances on real-time disruption

[†] Email address for correspondence: katharina.rath@ipp.mpg.de

prediction have been made using machine learning (Berkery *et al.* 2017; Rea & Granetz 2018; Kates-Harbeck, Svyatkovskiy & Tang 2019; Pau *et al.* 2019; Rea *et al.* 2019, 2020; Aymerich *et al.* 2022). Disruption prediction is a challenging task for various reasons. One of them is the imbalanced data situation; for some disruption classes, only a few measurements are available, making it difficult to obtain robust results. This is challenging, especially when working with neural networks, as they require a large training data set in order to give satisfying results and to avoid overfitting (see e.g. Aggarwal 2018). However, generating such an amount of training data from additional discharges is expensive and also potentially harmful for the reactor. Particularly with regard to future reactors such as ITER or SPARC, a sufficient data set will not be available at the time these reactors start operating.

Data augmentation is one possibility to balance the training data set by creating rare disruption events and thereby improving the prediction performance of machine learning models. The aim of data augmentation is to produce an arbitrarily large number of artificial samples that have the same statistical properties as the original small data set. Especially in the context of image classification, data augmentation is a widely used technique to improve the prediction accuracy and avoid overfitting (Shorten & Khoshgoftaar 2019). Commonly used methods are random transformation-based approaches, such as cropping or flipping. However, these methods are not expedient for the task at hand, as time dependencies and the causal structure of physical signals are destroyed by such transformations (Iwana & Uchida 2021; Wen *et al.* 2021). More elaborate methods for multivariate time series generation using neural networks (Yoon, Jarrett & van der Schaar 2019) require substantially more samples per class than usually available for disruption prediction. Other advanced data augmentation methods are based on decomposition into trend, seasonal/periodic signal and noise (Cleveland *et al.* 1990; Wen *et al.* 2019) or involve statistical modelling of the dynamics using, e.g. mixture autoregressive models (Kang, Hyndman & Li 2020).

Here, we tackle the above-mentioned challenges by relying on a non-parametric Bayesian approach to design the multivariate surrogate model based on Student t process regression (Shah, Wilson & Ghahramani 2014; Roth *et al.* 2017) to generate additional data. This model is closely related to the more commonly used Gaussian process regression (Williams & Rasmussen 1996). One drawback of standard Gaussian processes regression is the assumption of Gaussian noise, which is inaccurate due to outliers in the present application case. This results in unreliable uncertainty estimates. There have been attempts to make Gaussian process regression robust against outliers by using a Student t distributed noise model and relying on approximate inference (Neal 1997; Vanhatalo, Jylanki & Vehtari 2009). However, our approach rather builds on Student t processes with an analytic inference scheme (Shah *et al.* 2014) that also allows a heavy tailed noise distribution and gives robust results even for noisy data corrupted by outliers.

Another challenge imposed by high-resolution time series data is the computational complexity of multivariate Gaussian or Student t process regression of $O(N^3)$, where $N = DT$ is the number of training data points given by the product of dimensions D and time steps T of the multivariate time series. For typical values of $N > 1000$, traditional regression requires too much computing time. We instead use the state space formulation of a Student t process as a linear time invariant stochastic differential equation, which can be solved using a corresponding filter and smoother (Solin & Särkkä 2015). In the case of a Gaussian process, the analogous approach is the well-known Kalman filter and Rauch–Tung–Striebel (RTS) smoother (Särkkä 2013; Särkkä & Solin 2019). This ansatz reduces the computational complexity to $O(N)$, making it also suitable for high-resolution time series.

https://doi.org/10.1017/S0022377822000769 Published online by Cambridge University Press

Here, we are working with a multi-output state space model to generate multivariate time series. We first assume that dimensions of the multivariate time series are not correlated. This is done to avoid the requirement of optimizing all hyperparameters at the same time, which is practically unfeasible due to the limited amount of available data. To still account for signal interdependencies, we then induce correlations and cross-correlations via colouring transformations in a post-processing step.

To balance the training data set, we use several local surrogate models to generate data coming from different disruption classes. From a small set – usually less than 10 discharges – of multivariate time series with D measurement signals coming from one disruption class with similar operating conditions, we estimate the posterior distribution. We then sample from the trained model in order to generate similar data that enlarge the training database. To evaluate if the generated samples are from the same distribution as the training data, we use several methods from time series analysis, descriptive statistics and clustering algorithms to show that generated and training samples are almost indistinguishable.

2. Methods

2.1. *Student t processes*

Student t processes (TPs) are a generalization of the widely used Gaussian processes (GPs) (Williams & Rasmussen 1996; Shah *et al.* 2014). TPs allow for a heavy tailed noise distribution (estimated by an additional hyperparameter $\nu > 2$) and therefore put less weight on outliers compared with GPs (Shah *et al.* 2014; Roth *et al.* 2017). This is illustrated in figure 1 for a test case of synthetic data corrupted by outliers. As in GP regression, we consider a set of N training observations $\mathcal{D} = \{(t_i, y_i)\}_{i=1}^{T}$ of scalar function values $y_i = f(t_i)$ plus measurement noise at training points t_i with $i = 0, 1, \ldots, T$ (in our case, time). We model these data points using a TP with zero mean and covariance function $k(t, t')$,

$$f(t) \sim \mathcal{TP}(0, k(t, t'), \nu). \tag{2.1}$$

Similar to the GP, a kernel function $k(t, t')$ quantifies the covariance between values of f at times (t, t') and yields an $N \times N$ covariance matrix $\boldsymbol{K}$ with components $K_{ij} = k(t_i, t_j)$ for the random vector of all observed y_i. Kernel hyperparameters determine further details, e.g. a length scale l quantifies how fast correlations vanish with increasing distance in t. The additional hyperparameter $\nu > 2$ corresponds to the degrees of freedom that specify the noise distribution. The predicted distribution of a scalar output $f(t_*)$ at test point t_* is given in closed form by

$$\mathbb{E}[f(t_*)] = \boldsymbol{k}_*^{\top} \boldsymbol{K}_y^{-1} \boldsymbol{y}, \tag{2.2}$$

$$\mathbb{V}[f(t_*)] = \frac{\nu - 2 + \boldsymbol{y}^{\top} \boldsymbol{K}_y^{-1} \boldsymbol{y}}{\nu - 2 + N} (k_{**} - \boldsymbol{k}_*^{\top} \boldsymbol{K}_y^{-1} \boldsymbol{k}_*), \tag{2.3}$$

where $\boldsymbol{K}_y = \boldsymbol{K} + \sigma_n^2 \boldsymbol{I}$ is the measurement noise parametrized by the noise variance σ_n^2. Here, $\boldsymbol{k}_*$ is an N-dimensional vector with the ith entry being $k(t_*, t_i)$; $k_{**} = k(t_*, t_*)$ describes the covariance between training and test data and the variance at the test point t_*. In contrast to GP regression, the posterior variance $\mathbb{V}[f(t_*)]$ of the prediction explicitly depends on training observations by taking data variability into account and results in more reliable uncertainty estimates. An analogous expression to (2.3) is obtained for the covariance matrix between predictions at multiple t_* (Shah *et al.* 2014).

https://doi.org/10.1017/S0022377822000769 Published online by Cambridge University Press

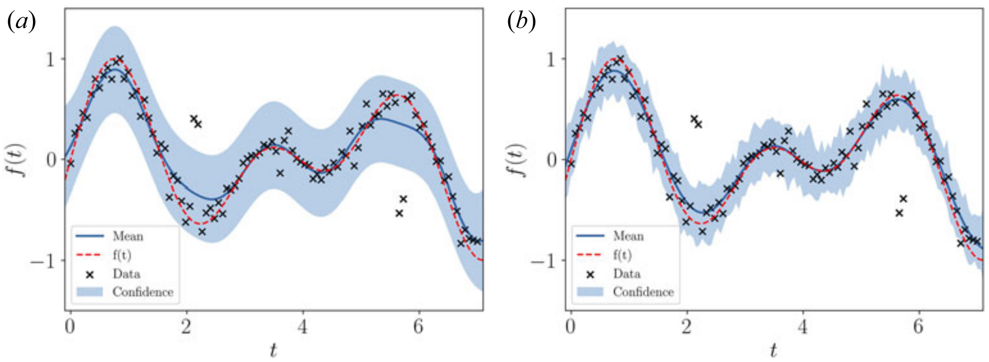

FIGURE 1. Predicted mean and 95 % confidence band with (*a*) GP and (*b*) TP trained on $N = 100$ training data points following $f(t) = \sin(2t)\cos(0.4t)$ corrupted by Gaussian noise $0.1\mathcal{N}(0, 1)$, with several outliers.

2.2. *State space formulation*

As in GP regression, the computational complexity increases with $O(N^3)$, as an inversion of the covariance matrix via Cholesky factorization is necessary to train TPs (Williams & Rasmussen 1996). This makes GP and also TP regression unfavourable for high-resolution time series data. However, as shown by Solin & Särkkä (2015), the TP regression problem can be reformulated as an mth-order linear time invariant stochastic differential equation (SDE)

$$\frac{\mathrm{d}\hat{\boldsymbol{f}}(t)}{\mathrm{d}t} = \boldsymbol{F}\hat{\boldsymbol{f}}(t) + \boldsymbol{L}\boldsymbol{w}(t), \tag{2.4}$$

$$f(t_i) = \boldsymbol{H}\hat{\boldsymbol{f}}(t_i), \tag{2.5}$$

where $\hat{\boldsymbol{f}}(t) = (f(t), \mathrm{d}f(t)/\mathrm{d}t, \ldots, \mathrm{d}^{m-1}f(t)/\mathrm{d}t^{m-1})^\top$, the feedback matrix $\boldsymbol{F}$ and noise effect matrix $\boldsymbol{L}$ are derived from the underlying TP, $\boldsymbol{H} = (1, 0, \ldots, 0)$ is the measurement or observation matrix and $\boldsymbol{w}(t)$ is a vector of white noise processes with spectral density $\gamma\boldsymbol{Q}$, where γ is a scaling factor (Solin & Särkkä 2015).

To solve this SDE for discrete points in time by estimating the posterior distribution $p(\hat{\boldsymbol{y}}_{0:T}|\boldsymbol{y}_{1:T})$ of the latent state $\hat{\boldsymbol{y}}_{0:T}$ given noisy observations $\boldsymbol{y}_{1:T}$, we use the corresponding Student t filter and smoother as outlined in Solin & Särkkä (2015). Here, the posterior is estimated by using marginal distributions: (i) filtering distribution $p(\hat{\boldsymbol{y}}_t|\boldsymbol{y}_{1:t})$ given by the update step in Algorithm 1, (ii) prediction distribution $p(\hat{\boldsymbol{y}}_{t+k}|\boldsymbol{y}_{1:t})$ given by the prediction step in Algorithm 1 for k steps after the current time step t and (iii) smoothing distributions $p(\hat{\boldsymbol{y}}_t|\boldsymbol{y}_{1:T})$ for $t < T$ given by Algorithm 2 (Särkkä 2013). The initial distribution is determined by the prior state mean given by the measurements at $t = 0$ and prior state covariance $\boldsymbol{P}_0$ given by the stationary covariance (Solin & Särkkä 2015). The augmented states $\mathrm{d}f/\mathrm{d}t$ that are not measured and noise are initialized with 0.

For example, the state space formulation of the Matérn 3/2 kernel is given by the following expressions for feedback, noise effect matrix and spectral density (Särkkä & Solin 2019):

$$\boldsymbol{F} = \begin{pmatrix} 0 & 1 & 0 \\ -\lambda^2 & -2\lambda & 0 \\ 0 & 0 & -\infty \end{pmatrix}, \quad \boldsymbol{P}_0 = \begin{pmatrix} \sigma^2 & 0 & 0 \\ 0 & \sigma^2\lambda^2 & 0 \\ 0 & 0 & \sigma_n^2 \end{pmatrix}, \quad \boldsymbol{H} = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}, \quad \boldsymbol{L} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \tag{2.6a–d}$$

where $\lambda = \sqrt{3}/l$. Hyperparameters l, σ^2, σ_n^2 and ν needed in the Student t filter algorithm are estimated by minimizing the negative log likelihood (Solin & Särkkä 2015). The log likelihood is sequentially calculated using the Student t filter (Algorithm 1). When the hyperparameters are optimized, the predictive distribution is first calculated via Algorithm 1 and then smoothed using Algorithm 2. In order to include the noise model with σ_n^2 corresponding to $\boldsymbol{K}_y = \boldsymbol{K} + \sigma_n^2 \boldsymbol{I}$ in traditional TP regression, the SDE is directly augmented by the entangled noise model. As the model is not only augmented with the noise model, but also with the first derivative of the target function we want to predict, we can immediately infer $\mathrm{d}f(t)/\mathrm{d}t$ from the given observations y.

Here, the task at hand concerns multivariate time series $\boldsymbol{Y}$ with multiple measurements n with D dimensions where the ith row is $\boldsymbol{y}_i = \boldsymbol{f}(t_i)$ at every time step t_i. To facilitate the training of the model, we consider an uncorrelated model, such that the associated random processes are not correlated. In traditional GP/TP regression, this corresponds to a multi-output model with a block-diagonal covariance matrix. The multi-output state space model to estimate $p(\hat{\boldsymbol{Y}}_{0:T}|\boldsymbol{Y}_{1:T})$ is built by stacking the univariate SDE models resulting in a block-diagonal structure for feedback and covariance matrices. Then, the dynamics of $\boldsymbol{y}_i$ is independent. We sample uncorrelated multivariate time series from this model and apply colouring transformations in a following post-processing step to account for correlations (§ 2.4). Each dimension has its own set of hyperparameters in order to grasp the dynamics that happen on different time scales. The measurement covariance matrix $\boldsymbol{R}$ (Algorithm 1) is estimated using the covariance of n measurements for each dimension at every time step.

2.3. *Student t sampler*

To sample from the estimated posterior distribution, we employ a Student t sampler, which is a modified version of the sampling technique presented by Durbin & Koopman (2002). First, we draw a t distributed random sequence $\hat{\boldsymbol{X}}_{0:T} = \hat{\boldsymbol{x}}_{i,0:T}$ from the prior estimated by the trained Student t model. These sequences are initialized by $\mathcal{T}(0, \boldsymbol{P}_0)$ and then filtered using Algorithm 1 and smoothed via Algorithm 2, which yields $\mathbb{E}(\hat{\boldsymbol{X}}_{0:T}|\boldsymbol{Y}^+_{1:T})$ where $\boldsymbol{Y}^+_{1:T} = \boldsymbol{H}\hat{\boldsymbol{X}}_{0:T}$, with the stacked measurement matrix $\boldsymbol{H} = (\mathsf{I}, 0, 0)$ that extracts only the first component of $\hat{x}_t$ in every time step t. Here, $\boldsymbol{Y}^+_{1:T}$ are data associated with the filtered and smoothed sequence $\hat{\boldsymbol{X}}_{0:T}$ given by (A2). Finally, to obtain a random sequence $\bar{\boldsymbol{Y}}_{0:T} = \bar{\boldsymbol{y}}_{i,0:T} \sim p(\hat{\boldsymbol{Y}}_{0:T}|\boldsymbol{Y}_{1:T})$, we combine

$$\bar{\boldsymbol{Y}}_{1:T} = \boldsymbol{H}(\mathbb{E}(\hat{\boldsymbol{Y}}_{0:T}|\boldsymbol{Y}_{1:T}) + \hat{\boldsymbol{X}}_{0:T} - \mathbb{E}(\hat{\boldsymbol{X}}_{0:T}|\boldsymbol{Y}^+_{1:T})), \tag{2.7}$$

where H extracts the first component of $\hat{\boldsymbol{y}}_t$ in every time step t. This procedure gives a D-dimensional multivariate time series for T time steps.

2.4. *Post-processing*

Given the trained model, we sample data $\bar{\boldsymbol{Y}}_{1:T}$ from the estimated posterior, where rows are dimensions $\bar{y}_i$ and columns are time steps; $\bar{\boldsymbol{Y}}_{1:T}$ can be split into a mean given by the smoothing distribution and deviations due to the sampling. Correlations between dimensions D of the generated data are not reproduced correctly with the uncorrelated model. However, with three different post-processing methods of increasing complexity compared in the results, we aim to handle correlations.

We thus want to inscribe the average covariance $\boldsymbol{\Sigma}$ over all samples empirically observed in the training data $\boldsymbol{Y}_{1:T}$ into the generated data $\bar{\boldsymbol{Y}}_{1:T}$. However, the covariance matrix $\bar{\boldsymbol{\Sigma}}$ of $\bar{\boldsymbol{Y}}_{1:T}$ has small non-zero off-diagonal elements. Therefore, we first perform a

https://doi.org/10.1017/S0022377822000769 Published online by Cambridge University Press

Zero Components Analysis (ZCA) whitening (also known as Mahalanobis) transformation (see e.g. Kessy, Lewin & Strimmer 2018):

$$\boldsymbol{Z} = \bar{\boldsymbol{\Sigma}}^{-1/2}\bar{\boldsymbol{Y}}. \tag{2.8}$$

The transformed data $\boldsymbol{Z}$ have a diagonal covariance matrix $\boldsymbol{\Lambda}_Z$, with unit variances on the diagonal. We then colour the generated data via a colouring transformation (Kessy *et al.* 2018)

$$\tilde{\boldsymbol{Y}} = \boldsymbol{\Sigma}^{1/2}\boldsymbol{Z} = \boldsymbol{\Sigma}^{1/2}\bar{\boldsymbol{\Sigma}}^{-1/2}\bar{\boldsymbol{Y}}, \tag{2.9}$$

obtaining data $\tilde{\boldsymbol{Y}}$, which now have the same (temporally local) covariance as the training data $\boldsymbol{Y}$.

Another possibility is to directly take the distribution of the training data covariance matrix $\boldsymbol{\Sigma}$ over samples into account by using samples from a corresponding multivariate Gaussian distribution as data covariance matrices. This generates variation in the covariance of the generated data, especially if there are local differences between the samples. However, on average for a large enough sample size, we recover the training data covariance matrix $\boldsymbol{\Sigma}$.

To also take time-lagged correlations into account, we must adjust not only covariances but also cross-covariances in our generated data. Therefore, we use the cross-covariance matrix given by

$$\bar{\boldsymbol{\Sigma}}_{c,rs}(t_1, t_2) = \mathbb{E}[(\bar{y}_{r,t_1} - \mu_{r,t_1})(\bar{y}_{s,t_2} - \mu_{s,t_2})], \tag{2.10}$$

where the expected value $\mathbb{E}[\cdot]$ is estimated by averaging over all combinations of lags $t_1 - t_2$ in addition to the sample mean. Here, $\mu_{i,t}$ is the expected value of $\bar{y}_{i,t}$. To decorrelate and colour the data in the way described above, we formally use a global covariance matrix $\boldsymbol{\Sigma}_g$ of size $DT \times DT$ involving correlations both over time and across dimensions of the multivariate time series. The global covariance matrix is a periodic block matrix given by

$$\boldsymbol{\Sigma}_{g,(t_1D+r)(t_2D+s)} = \boldsymbol{\Sigma}_{c,rs}(t_1, t_2) \tag{2.11}$$

for the cross-covariance $\boldsymbol{\Sigma}_c$ with lag. The generated data is coloured using the global covariance matrix:

$$\tilde{\boldsymbol{Y}} = \boldsymbol{\Sigma}_g^{1/2}\boldsymbol{Z} = \boldsymbol{\Sigma}_g^{1/2}\overline{\boldsymbol{\Sigma}_g}^{-1/2}\bar{\boldsymbol{Y}}. \tag{2.12}$$

This incorporates the empirical cross-covariance for all time lags and between all dimensions D of the generated data.

3. Evaluation of generated data

As the generated data serve as augmented training data for later analyses, statistical properties of the original training data should be reflected in the generated data. Therefore, we perform statistical tests to check if training and generated share key statistical properties.

3.1. *Distribution and Wasserstein distance*

To measure the distance between the distribution of the training and the generated data, we use the Wasserstein-1 metric (Villani 2008)

$$W_1(P, V) = \inf_{\gamma \in \Gamma(P,V)} \int_{\mathbb{R}\times\mathbb{R}} |x - y| \,\mathrm{d}\gamma(x, y), \tag{3.1}$$

where $\Gamma(P, V)$ denotes the set of all probability distributions on $\mathbb{R} \times \mathbb{R}$, with P, V being its marginals. The minimizer γ of (3.1) denotes the optimal transport plan to transport

https://doi.org/10.1017/S0022377822000769 Published online by Cambridge University Press

P to V. We compare each signal separately and average the corresponding Wasserstein distances. Although the problem concerns time series data, we discard all time information and only consider the global distribution of the data due to the small amount of available training data samples.

3.2. *Maximum mean discrepancy two-sample test*

In addition to the Wasserstein distance, we perform the kernel two-sample test (Gretton *et al.* 2012) for each signal (again discarding time information). The null hypothesis we want to test is that both n training data $\boldsymbol{y}_{i,1:T}$ and m generated data samples $\tilde{\boldsymbol{y}}_{i,1:T}$ follow the same distribution P. We use the maximum mean discrepancy (MMD) test statistic via a kernel g

$$\mathrm{MMD}^2 = \frac{1}{n(n-1)} \sum_{i,j=1}^{n} g(\boldsymbol{y}_{i,1:T}, \boldsymbol{y}_{j,1:T}) + \frac{1}{m(m-1)} \sum_{i,j=1}^{m} g(\tilde{\boldsymbol{y}}_{i,1:T}, \tilde{\boldsymbol{y}}_{i,1:T}) \\ - \frac{2}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} g(\boldsymbol{y}_{i,1:T}, \tilde{\boldsymbol{y}}_{i,1:T}), \tag{3.2}$$

where $g(x, y) = \exp(-||x-y||^2/(2\sigma^2))$ with $\sigma = \mathrm{Median}(|\Upsilon_i - \Upsilon_j|)/2$ and Υ is the combined sample of $\boldsymbol{y}_{i,1:T}$ and $\tilde{\boldsymbol{y}}_{i,1:T}$. To estimate a threshold for the acceptance of the null hypothesis for a given confidence level, bootstrapping is performed via mixing samples $\boldsymbol{y}_{i,1:T}$ and $\tilde{\boldsymbol{y}}_{i,1:T}$, which generates a distribution with 10 000 samples that satisfies the null hypothesis. Finally, we can estimate a p-value for the MMD of the generated data distributions.

3.3. *Auto- and cross-correlation*

To evaluate if the generated data reflect the temporal dependencies of the training data, we calculate auto- and cross-correlations ρ_{rs} for training and generated data by normalizing the cross-covariance $\boldsymbol{\Sigma}_c$ in (2.10) by $1/(\sigma_{r,t_1}\sigma_{s,t_2})$. Here, $\sigma_{s,t}$ is the standard deviation of $\tilde{y}_{s,t}$. If $r = s$, this diagnostic becomes the auto-correlation – see, e.g. Park (2017). For $t_1 = t_2$, the local correlation matrix follows. We evaluate the mean squared error (MSE) to the auto- and cross-correlation of the training data. Evidently, the global colouring transformation (2.10) produces a perfect match in this diagnostic.

3.4. *Power spectral density*

All frequencies that are present in the training data set should also appear in the generated data. This can be evaluated using the power spectral density (PSD), which provides an estimate of power distribution across the frequency of a signal. We evaluate the mean squared error between the PSD of the training data and generated data.

3.5. *Embedding via kernel principal component analysis*

We apply two-dimensional (2-D) kernel principal component analysis (PCA) on the training data with flattened temporal dimension and project the generated data onto the first two principal components of the training data to evaluate the embedding and visualize if both training and generated data lie on the same submanifold (Schölkopf, Smola & Müller 1998). In all test cases, a polynomial kernel of degree 3 with optimized kernel coefficient (minimization of the reconstruction error) is used.

The distance between the embedded distributions of training and generated data is measured by using the sliced Wasserstein distance that takes advantage of the very efficient

https://doi.org/10.1017/S0022377822000769 Published online by Cambridge University Press

calculation of 1-D Wasserstein distances (Bonneel *et al.* 2015; Flamary *et al.* 2021). The multivariate distribution is sliced and randomly projected on a 1-D subspace, and the corresponding 1-D Wasserstein distances are averaged to obtain an estimation for the multivariate distribution. With an increasing number of projections, the sliced Wasserstein distance converges. Here, we use 10^3 projections to estimate the distance W_{emb} between the embedded distributions.

3.6. *Multivariate functional PCA*

For the evaluation of the correctly represented temporal evolution of the generated data, we apply multivariate functional principal component analysis (mfPCA) on the training data and project the generated data onto the eigenbasis of the training data (Happ & Greven 2018). Then, we reconstruct both training and generated data with the same eigenbasis and evaluate the variance of the residuals.

3.7. *Dynamic time warping*

For time series comparison, dynamic time warping (DTW) is widely used to measure the similarity between two temporal sequences $\boldsymbol{y}_{i,1:T}$ and $\tilde{\boldsymbol{y}}_{j,1:T}$ (Berndt & Clifford 1994). This metric is formulated as an optimization problem

$$\mathrm{DTW}(\boldsymbol{y}_{i,1:T}, \tilde{\boldsymbol{y}}_{j,1:T}) = \min_{\gamma} \sqrt{\sum_{(i,j)\in\gamma} d(\boldsymbol{y}_i, \tilde{\boldsymbol{y}}_j)^2}, \tag{3.3}$$

where γ is the alignment path such that the Euclidean distance between $\boldsymbol{y}_{i,1:T}$ and $\tilde{\boldsymbol{y}}_{j,1:T}$ is minimal. Hence, DTW gives the distance between two time series with the best temporal alignment. We compare each training data sample with each generated data sample and use the mean to compare different post-processing methods.

3.8. *Self-organizing maps on time series*

Finally, we apply time series clustering based on DTW self-organizing maps (SOMs) on both the training and generated data (Vettigli 2018). If the generated data are a potentially useful extension of the training data, the clustering should show similar results. Therefore, we compute a clustering model on the training data and use the trained model to predict cluster labels of both the training and generated data. From the predicted labels, we evaluate the $F1$ score (harmonic mean of precision and recall) (Murphy 2022) with the ground truth.

4. Numerical experiments

We evaluate the performance of the proposed model using disruption data from several discharges from the DIII-D tokamak taken from the 2016 experimental campaign. These disruptions were already included in previously published papers on data-driven applications in fusion (Montes *et al.* 2021).

We cluster the available data sets depending on the similarity of the conditions and on the occurring instability. Here, we use the model to augment five signals of the training data set (referred to as β_n, the normalized β given by $\beta_n = \beta a B_T / I_p$, where β is the ratio of plasma pressure to magnetic pressure, B_T is the toroidal magnetic field, a the minor radius and I_p the plasma current; normalized internal inductance li; plasma elongation κ; safety factor q_{95}; Greenwald fraction n/n_G) for different disruptions: (i) disruptions due to locked modes (LMs) in high β, low torque plasmas with $n = 1$ resonant magnetic perturbations (RMPs) applied (shots 166463, 166464, 166465, 166466, 166468, 166469), (ii) disruptions

https://doi.org/10.1017/S0022377822000769 Published online by Cambridge University Press

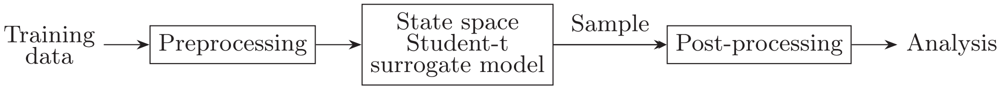

FIGURE 2. Data processing flow.

due to LMs during an RMP edge localized mode (ELM) suppression experiment applied to an ITER-like plasma shape (shots 166452, 166454, 166457, 166460) and (iii) density accumulation events during detachment studies of helium plasmas (shots 166933, 166934, 166937).

For each disruption class, the model is trained on these few available training samples. The choice of signals is influenced by the use case of augmenting the training database for a neural network for disruption prediction, but in general, the method is extendable to any number and any kind of signals.

Following the flow shown in figure 2, preprocessing is performed on the training data. As we are primarily interested in the behaviour close to a disruption, we align the samples according to their end time and only consider the stable flat-top phase. Additionally, all data are rescaled via min–max scaling to a range of $[-0.5, 0.5]$. This stabilizes the optimization of the hyperparameters in the Student t filter algorithm, as the input to the optimizer is of order 1. Missing data points are interpolated linearly. All discharges are sampled every 25 ms. Then, we set up the state space Student t surrogate model. In all experiments, a Matérn 3/2 kernel as in (2.6*a–d*) is used. We train the surrogate model by optimizing its hyperparameters by minimizing the negative log likelihood using the Scipy implementation of L-BFGS-B (Virtanen *et al.* 2020), and resulting values for all experiments can be found in Appendix B, table 4. Each signal has its own set of hyperparameters in order to be able to handle the dynamics that happen on different time scales. Subsequently, we apply the Student t filter and smoother (Algorithms 1 and 2) with optimized hyperparameters to our data. From the estimated distribution, we draw 1000 samples from the posterior using the Student t sampler and perform the colouring transformations in the post-processing. Finally, after rescaling the samples to the original data range, we evaluate the generated data sets by using the defined metrics. In general, the generation of the time series samples is of $O(N)$, but some of the metrics used to evaluate the generate data are not. Therefore, we limited the number of samples in the given analysis to 1000.

5. Results and analysis

For each disruption class, we draw 1000 samples from the posterior estimated by the trained model and compare four available post-processing methods: (I) uncorrelated model (here, no post-processing is performed), (II) colouring transformation with the empirical covariance matrix, (III) colouring transformation with the empirical cross-covariance matrix to account for lagged correlations and (IV) colouring transformation with the sampled covariance matrix. The results for test cases (i) and (ii) are presented in Appendix in C.1 and C.2.

In figure 3, a visual comparison is given between training data and generated data for the colouring transformation with empirical cross-covariance matrix, together with the estimated mean and 95 % confidence intervals for the disruption data from DIII-D for test case (i). The model is able to capture the general trend given by the training data and can also reproduce outliers. In general, the generated data fit the distribution of the training data.

We continue with a thorough statistical analysis, which allows a ranking of the different post-processing methods following the metrics outlined in § 3. The results are given in

https://doi.org/10.1017/S0022377822000769 Published online by Cambridge University Press

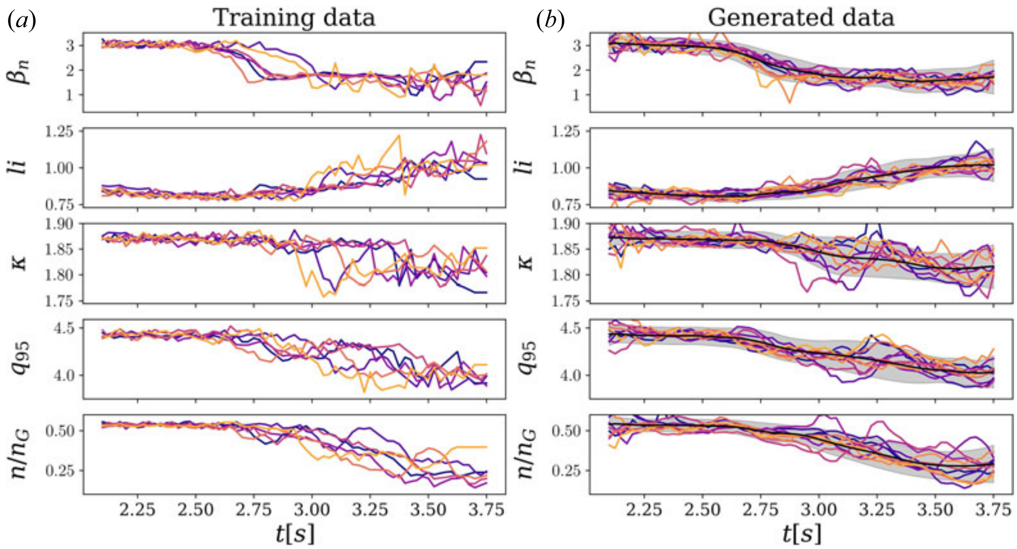

FIGURE 3. (*a*) Training data and (*b*) 10 generated data sets from the state space Student t surrogate model together with the estimated mean (black solid line) and 95 % confidence (grey shaded region) for test case (i). Different colours correspond to different shots of training data and different samples of the generated data, respectively.

Metric	Uncorrelated	Emp. cov	Emp. crosscov	Sample cov
W_1	0.035 ± 0.013	0.035 ± 0.011	0.038 ± 0.012	$\mathbf{0.036 \pm 0.01}$
MMD p-value	0.68 ± 0.35	0.82 ± 0.18	$\mathbf{0.92 \pm 0.06}$	0.85 ± 0.13
MSE ρ_{rs}	0.019 ± 0.009	0.018 ± 0.009	$\mathbf{0.0011 \pm 0.0004}$	0.017 ± 0.009
W_{emb}	$\mathbf{0.0592 \pm 0.0006}$	0.0682 ± 0.0006	0.0766 ± 0.0007	0.0682 ± 0.006
MSE PSD $[10^{-6}]$	5 ± 4	4 ± 3	$\mathbf{3 \pm 3}$	4 ± 3
DTW	0.8 ± 0.5	0.8 ± 0.4	$\mathbf{0.7 \pm 0.3}$	0.85 ± 0.4
MSE mfPCA	0.137	0.015	**0.011**	0.022

TABLE 1. Post-processing method comparison for test case (i). Mean and standard deviation over five dimensions and $N = 1000$ samples generated from the trained model for statistical metrics described in § 3. Best values are highlighted in bold.

table 1 for test case (i). Other experiments give similar results, as indicated in Appendix in C.1 (table 5), and C.2 (table 7) for test cases (ii) and (iii), respectively.

To put the calculated metrics into context, we identify nearby non-disruptive shots coming from the same specific campaign with similar operating conditions for test case (ii). Then, we evaluate the Wasserstein distance between nearby non-disruptive and disruptive discharges to compare the obtained Wasserstein distances for the generated data for this disruption class. For test case (ii), we identify five nearby non-disruptive shots 166433, 166434, 166442, 166444, 166455 and found $W_1 = 0.31 \pm 0.12$ between non-disruptive and disruptive discharges. Additionally, the 2-D kernel PCA embedding of nearby non-disruptive and disruptive discharges evaluated by the estimation of the 2-D sliced Wasserstein distance is estimated. We observe $W_{\mathrm{emb}} = 0.74 \pm 0.01$ for test case (ii).

https://doi.org/10.1017/S0022377822000769 Published online by Cambridge University Press

Test case	Uncorrelated	Emp. cov	Emp. crosscov	Sample cov
(i) stable	0.03 ± 0.01	0.03 ± 0.01	0.04 ± 0.01	0.03 ± 0.01
(i) unstable	0.04 ± 0.01	0.04 ± 0.01	0.04 ± 0.01	0.04 ± 0.01
(ii) stable	0.02 ± 0.02	0.02 ± 0.01	0.02 ± 0.01	0.02 ± 0.01
(ii) unstable	0.07 ± 0.02	0.07 ± 0.02	0.08 ± 0.02	0.08 ± 0.02
(iii) stable	0.08 ± 0.08	0.03 ± 0.02	0.02 ± 0.01	0.03 ± 0.02
(iii) unstable	0.07 ± 0.08	0.03 ± 0.03	0.02 ± 0.02	0.04 ± 0.03

TABLE 2. Post-processing method comparison for disruption data from DIII-D. Mean and standard deviation of the Wasserstein metric between training and generated data for stable and unstable phases of the disruptive discharges. The Wasserstein metric is averaged over five dimensions and $N = 1000$ samples generated from the trained model.

The achieved Wasserstein distance between training and generated data for this disruption class is significantly smaller in all post-processing methods, as given in table 5. The same holds for the Wasserstein distance of the 2-D kernel PCA embedding. This is promising, as it implies that the augmented data are much more similar to disruptive discharges within their proper class than to non-disruptive discharges from the same campaign in these measures.

For test cases (i) and (iii), non-disruptive discharges from those specific campaigns are not available. Therefore, we investigate the distributions in stable and unstable phases of the training and generated disruptive discharges in more detail. Using the average time stamp of the manually labelled training data, this information about the stable and unstable phase was propagated to label the generated data. Then we calculate the Wasserstein distance averaged over all features between training and generated data for both phases separately. The obtained results for all test cases are given in table 2. For comparison, we also estimate the Wasserstein distances between stable and unstable phases and found $W_1 = 0.36 \pm 0.07$ for test case (i), $W_1 = 0.37 \pm 0.08$ for test case (ii) and $W_1 = 0.24 \pm 0.1$ for test case (iii). The obtained distances between training and generated data within the different phases lie sufficiently below the distances between stable and unstable parts of the discharges.

The superiority of the post-processing with the empirical cross-covariance is apparent in figure 4, where the auto- (on the diagonal) and cross-covariance for all estimated signals are shown. As we are inscribing the empirical cross-covariance into the uncorrelated generated data from the model, the cross-covariance fits exactly, and the cross-covariances lie on top of each other. When using either the empirical covariance or the sample covariance, only the cross-covariance at lag 0 matches the cross-covariance of the training data. Both post-processing methods give on average the same cross-covariance for 1000 generated samples. Additionally, the difference in covariance at lag 0 is shown in figure 5.

Figure 6 displays the kernel density of the 2-D kernel PCA embedding of the generated data in the eigenspace of the training data. All four methods generate data that lie on the same submanifold as the training data. However, when cross-covariances are included, the shape of the training data is better reproduced. In test case (i) shown in figure 6, one of the three extrema is not reproduced by the generated data. By evaluating the embedding for different combinations of input signals, a likely explanation is that β_n causes this extremum. The reason why the generated data are not able to reproduce this extremum in the eigenspace is due to the multi-modality of the distribution around the drop in β_n in the range 2.75–3.00 s. This is also one limitation of the presented model as it is not

https://doi.org/10.1017/S0022377822000769 Published online by Cambridge University Press

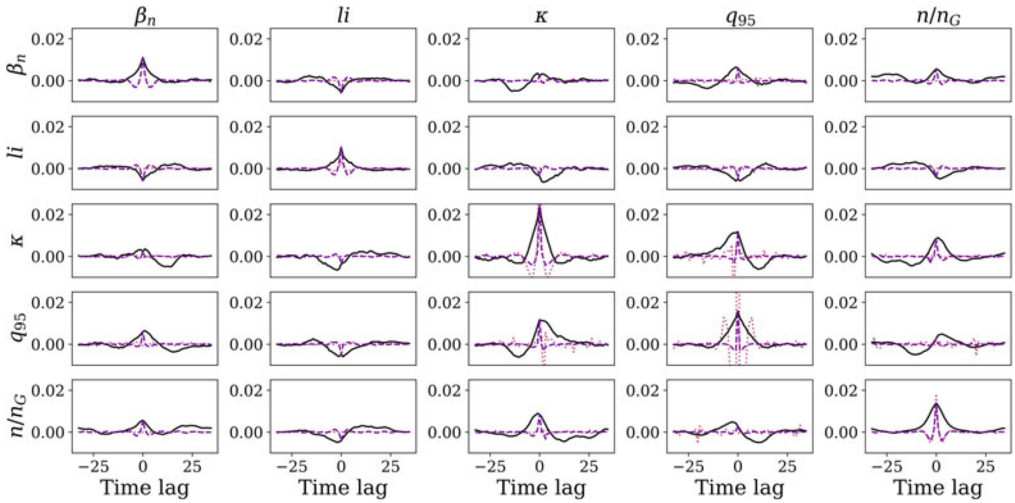

FIGURE 4. Comparison of the cross-covariance in the training and generated data with cross-covariance (solid lines on top of each other, numerical error of order 10^{-16}), covariance or sampled covariance post-processing (dashed lines) and uncorrelated model (dotted line) for test case (i).

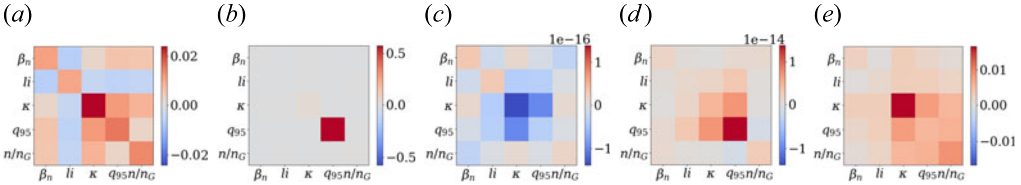

FIGURE 5. Comparison of the covariance of training data (*a*) and the difference from the generated data (*b*) with uncorrelated model, (*c*) empirical covariance, (*d*) cross-covariance and (*e*) sampled covariance post-processing for test case (i). Note the different scaling in the colour scale.

able to represent multi-modality of a cluster correctly. One possibility is to further refine the considered clusters to augment the data base (in the extreme case, down to one single discharge). In general, the number of available training data samples is very limited, as we are working with manually labelled disruptive data from DIII-D. Therefore, the results here only give an idea of whether the features apparent in the training data are also apparent in the generated data.

Besides the Wasserstein distance, DTW is difficult to interpret without context. Again, we calculate the metric between nearby non-disruptive and disruptive discharges for test case (ii) and obtain DTW $= 2.9 \pm 1.6$. The large error is due to averaging over all signals. Overall, the distances between the generated and training data for this disruption class lie below the distance between nearby non-disruptive and disruptive discharges for this test case. In test cases (i) and (iii), where non-disruptive data from the same campaigns are not available, DTW distances between generated and training data with included correlations are of the same order as in test case (ii).

The training data were also reconstructed using the multivariate functional PCA with 5 components. We observe the following reconstruction mean squared errors for test case (i) 0.006, (ii) 0.003 and (iii) 0.008. We use the first five eigenfunctions of the training data

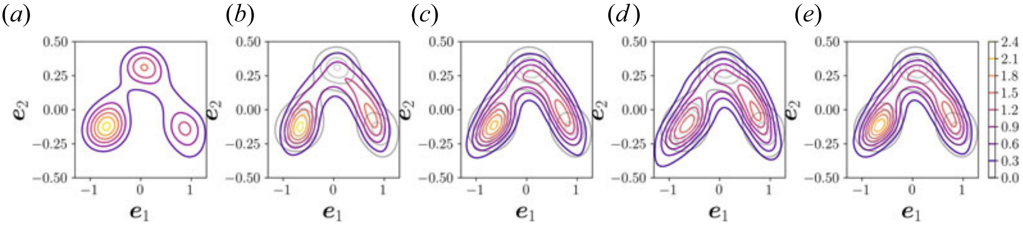

FIGURE 6. Kernel density estimation of the 2-D kernel PCA embedding of the (*a*) training data and generated data via (*b*) uncorrelated model, (*c*) empirical covariance, (*d*) cross-covariance and (*e*) sampled covariance post-processing for test case (i). The embedded training data are shown in grey in all plots. The colour scale representing the density is the same in all plots.

Train	Test	Training	Uncorrelated	Emp. cov	Emp. crosscov	Sample cov
original	generated	0.75	0.74	0.75	0.74	0.78
generated	original	0.88	0.86	0.90	0.90	0.89
mix	mix	0.89	0.89	0.89	0.89	0.89

TABLE 3. The $F1$ score for DTW SOM clustering of different post-processing methods for test case (i).

as a basis to project the generated data of each test case. The reconstruction error of the generated data with included correlations in the post-processing is still of the same order.

Finally, we use SOMs for time series clustering to evaluate if the label prediction works similarly well for the generated data. Here, we only use three classes, as the training data look quite similar for different signals. The results for three different experiments are given in table 3. Between the four post-processing methods, no significant difference is evident. The clustering algorithm performs as well on all methods as on the original training data.

6. Conclusion and outlook

We applied Student *t* process regression in a state space formulation to introduce robust data augmentation for multivariate time series. The state space formulation reduces the computational complexity and is thus suitable for high-resolution time series. We used the model to learn the distribution of time series coming from a given disruption class. From the estimated posterior, time series were generated to augment the training database. To evaluate if the original and generated data share key statistical properties, multiple statistical analyses and classic machine learning clustering algorithms have been carried out. We found that, within the scope of the used metrics, the generated time series resemble the training data to a sufficient extent. An important limitation of the method is multi-modality in the training data set which a Student *t* process cannot reproduce. In this case, the training data sets can be further split.

When the method is applied to augment the training database for the neural network disruption predictor, a thorough analysis of the existing (labelled) training database is necessary to decide which disruption classes are not available in sufficient quantity. For each of those classes, we will train the surrogate model and then be able to generate data to balance the data set. Subsequently, the performance of the neural network trained with the augmented training database will be evaluated. Due to the broad range of

https://doi.org/10.1017/S0022377822000769 Published online by Cambridge University Press

evaluation metrics, we are optimistic that the generated data will improve and robustify the performance.

Another perspective regards disruption prediction of future devices, where little data will be available to train machine learning-based approaches. In this case, the surrogate model could be used and updated, as more data are being collected and can therefore update machine learning-driven models.

To improve the proposed method, the integration of correlations and cross-correlations on the level of a multivariate surrogate model instead of the colouring in post-processing will be investigated in future work (Boyle & Frean 2004; Vandenberg-Rodes & Shahbaba 2015). Another possible extension of the current method could also take spatial information of profiles into account (Wilkinson *et al.* 2020).

However, the approach developed here is sufficiently generic to be used for data augmentation in a broad range of applications, e.g. time series in climate research.

Acknowledgements

Editor William Dorland thanks the referees for their advice in evaluating this article.

Funding

The present contribution is supported by the Helmholtz Association of German Research Centers under the joint research school HIDSS-0006 ‘Munich School for Data Science – MUDS’ (K.R.) and the MIT-Germany Lockheed Martin Seed Fund (K.R., C.R., A.M., R.G., U.v.T.). This work has been carried out within the framework of the EUROfusion Consortium, funded by the European Union via the Euratom Research and Training Programme (Grant Agreement No 101052200 – EUROfusion). Views and opinions expressed are, however, those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the European Commission can be held responsible for them (C.A.). This work was supported by the Federal Ministry of Education and Research (BMBF) of Germany by Grant No. 01IS18036A (D.R., B.B.). This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Fusion Energy Sciences, using the DIII-D National Fusion Facility, a DOE Office of Science user facility, under Award(s) DE- SC0014264, and DE-FC02-04ER54698 (C.R., A.M., R.G.).

Disclaimer

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process or service by trade name, trademark, manufacturer or otherwise does not necessarily constitute or imply its endorsement, recommendation or favouring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

Declaration of interests

The authors report no conflict of interest.

https://doi.org/10.1017/S0022377822000769 Published online by Cambridge University Press

Appendix A. Algorithm

Algorithm 1 Multivariate Student-t filter (Solin & Särkkä 2015)

```
Init:
    $\hat{\boldsymbol{y}}_{0|0} = \boldsymbol{y}_0, \quad \boldsymbol{P}_{0|0} = \boldsymbol{P}_0, \quad \boldsymbol{\nu}_0 = \boldsymbol{\nu}, \quad \gamma_0 = \boldsymbol{I}_D$    (A 1)
for $t = 1, 2, \ldots, T$ do
    Filter prediction:
        $\hat{\boldsymbol{y}}_{t|t-1} = \boldsymbol{A}_{t-1}\hat{\boldsymbol{y}}_{t-1}$    (A 2)
        $\boldsymbol{P}_{t|t-1} = \boldsymbol{A}_{t-1}\boldsymbol{P}_{t-1}\boldsymbol{A}_{t-1}^{\top} + \gamma_{t-1}\boldsymbol{Q}_{t-1},$    (A 3)
    where $\boldsymbol{A}_t = \exp(\boldsymbol{F}\Delta t)$ and $\boldsymbol{Q}_t = \boldsymbol{P}_0 - \boldsymbol{A}_t\boldsymbol{P}_0\boldsymbol{A}_t^{\top}$.
    Filter update (if measurement $\boldsymbol{y}_t$ with mean $\bar{\boldsymbol{y}}_t$ is available):
        $\boldsymbol{v}_t = \bar{\boldsymbol{y}}_t - \boldsymbol{H}_t\hat{\boldsymbol{y}}_t$    (A 4)
        $\boldsymbol{S}_t = \boldsymbol{H}_t\boldsymbol{P}_{t|t-1}\boldsymbol{H}_t^{\top} + \boldsymbol{R}$    (A 5)
        $\gamma_t = \frac{\gamma_{t-1}}{\nu_t - 2}(\nu_{t-1} - 2 + \boldsymbol{v}_t\boldsymbol{S}_t^{-1}\boldsymbol{v}_t)$    (A 6)
        $\boldsymbol{K}_t = \boldsymbol{P}_{t|t-1}\boldsymbol{H}_t^{\top}\boldsymbol{S}_t^{-1}$    (A 7)
        $\hat{\boldsymbol{y}}_{t|t} = \hat{\boldsymbol{y}}_{t|t-1} + \boldsymbol{K}_t\boldsymbol{v}_t$    (A 8)
        $\boldsymbol{P}_{t|t} = \frac{\gamma_t}{\gamma_{t-1}}(\boldsymbol{P}_{t|t-1} - \boldsymbol{K}_t\boldsymbol{S}_t\boldsymbol{K}_t^{\top})$    (A 9)
end for
```

Algorithm 2 Multivariate Student-t smoother (Solin & Särkkä 2015)

```
Init:
    $$\hat{y}_T = \hat{y}_{T|t}, \quad \boldsymbol{P}_T = \boldsymbol{P}_{T|t} \tag{A 10}$$
for $t = T-1, T-2, \ldots, 1$ do
    Smoother prediction:
        $$\hat{y}_{t+1|t} = \boldsymbol{A}_t \hat{y}_{t|t} \tag{A 11}$$
        $$\boldsymbol{P}_{t+1|t} = \boldsymbol{A}_t \boldsymbol{P}_{t|t} \boldsymbol{A}_t^\top + \gamma_t \boldsymbol{Q}_t \tag{A 12}$$
    Smoother update:
        $$\boldsymbol{G}_t = \boldsymbol{P}_{t|t} \boldsymbol{A}_t^\top \boldsymbol{P}_{t+1|t}^{-1} \tag{A 13}$$
        $$\hat{y}_{t|T} = \hat{y}_{t|t} + \boldsymbol{G}_t(\hat{y}_{t+1|T} - \hat{y}_{t+1|t}) \tag{A 14}$$
        $$\boldsymbol{P}_{t|T} = \frac{\gamma_T}{\gamma_t}(\boldsymbol{P}_{t|t} - \boldsymbol{G}_t \boldsymbol{P}_{t+1|T} \boldsymbol{G}_t^\top) + \boldsymbol{G}_t \boldsymbol{P}_{t+1|T} \boldsymbol{G}_t^\top \tag{A 15}$$
end for
```

Appendix B. Hyperparameters

For the different test cases, we used the hyperparameters given in table 4.

Test case	hyp	β_n	li	κ	q_{95}	n/n_G
(i)	ν	2.19	2.58	2.15	2.21	2.1
	σ_n^2	0.024	0.01	0.056	0.029	0.02
	σ^2	1.74	1.76	1.96	1.60	1.60
	l	19.6	28.3	20.3	20.1	15.7
(ii)	ν	3.4	2.57	2.36	2.49	2.7
	σ_n^2	0.023	0.032	0.036	0.033	0.022
	σ^2	1.65	0.53	1.36	1.87	1.91
	l	17.7	19.8	9.63	19.1	16.8
(iii)	ν	2.14	2.12	2.01	2.71	2.55
	σ_n^2	0.163	0.044	0.493	0.016	0.011
	σ^2	0.62	1.73	1.24	1.21	0.58
	l	17.6	11.5	4.5	12.8	10.0

TABLE 4. Optimized hyperparameters for the state space Student t surrogate model for all test cases.

https://doi.org/10.1017/S0022377822000769 Published online by Cambridge University Press

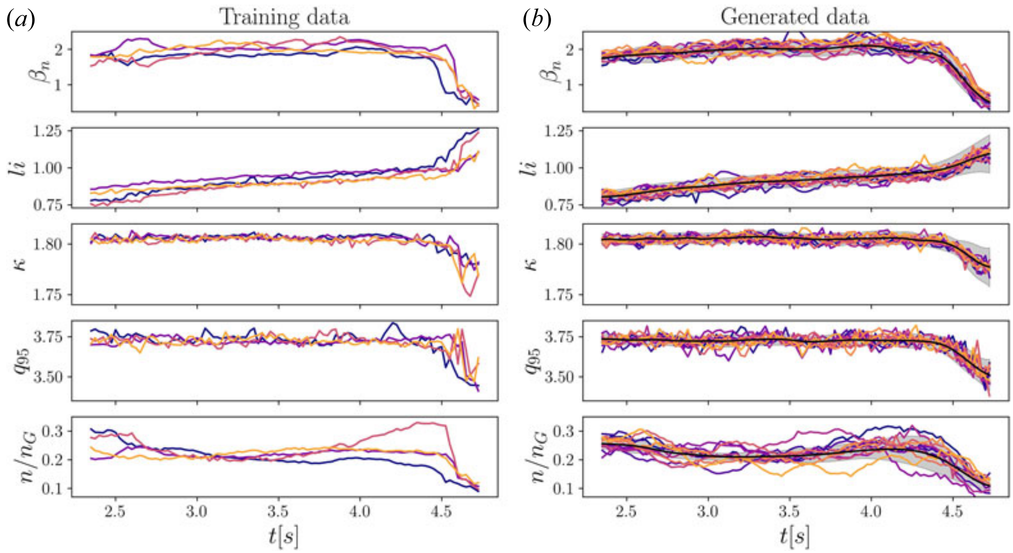

FIGURE 7. (*a*) Training data and (*b*) 10 generated data sets from the state space Student *t* surrogate model together with the estimated mean (black solid line) and 95 % confidence (grey shaded region) for test case (ii). Different colours correspond to different shots of training data and different samples of the generated data, respectively.

Metric	Uncorrelated	Emp. cov	Emp. crosscov	Sample cov
W_1	0.027 ± 0.013	0.020 ± 0.006	0.022 ± 0.007	$\mathbf{0.020 \pm 0.006}$
MMD p-value	0.617 ± 0.335	0.869 ± 0.153	$\mathbf{0.885 \pm 0.107}$	0.876 ± 0.15
MSE ρ_{rs}	0.013 ± 0.017	0.013 ± 0.017	$\mathbf{0.005 \pm 0.005}$	0.014 ± 0.017
W_{emb}	0.0458 ± 0.0003	0.0496 ± 0.0004	$\mathbf{0.0466 \pm 0.0004}$	0.0539 ± 0.0004
MSE PSD $[10^{-6}]$	7 ± 9	3 ± 4	$\mathbf{1 \pm 2}$	3 ± 4
DTW	0.86 ± 0.37	0.78 ± 0.32	$\mathbf{0.65 \pm 0.31}$	0.83 ± 0.39
MSE mfPCA	0.011	0.009	**0.007**	0.011

TABLE 5. Post-processing method comparison for test case (ii). Mean and standard deviation over five dimensions and $N = 1000$ samples generated from the trained model for statistical metrics described in § 3. Best values are highlighted in bold.

Appendix C. Results for other test cases

In the following sections, the results for test cases (ii) and (iii) are presented.

C.1. *Test case (ii): disruption due to MHD instability during RMP ELM control*

A visual comparison of the training and the generated data for test case (ii) is shown in figure 7. Here, the disruption occurs due to magnetohydrodynamic (MHD) instability induced by RMPs applied to control ELMs (shots 166452, 166454, 166457, 166460). The results of the statistical analysis are given in table 5 and are of the same order as for test case (i). Figures 8 and 9 show the cross-covariance and the covariance of the training and generated data. Figure 10 displays the kernel density of 2-D PCA embedding

https://doi.org/10.1017/S0022377822000769 Published online by Cambridge University Press

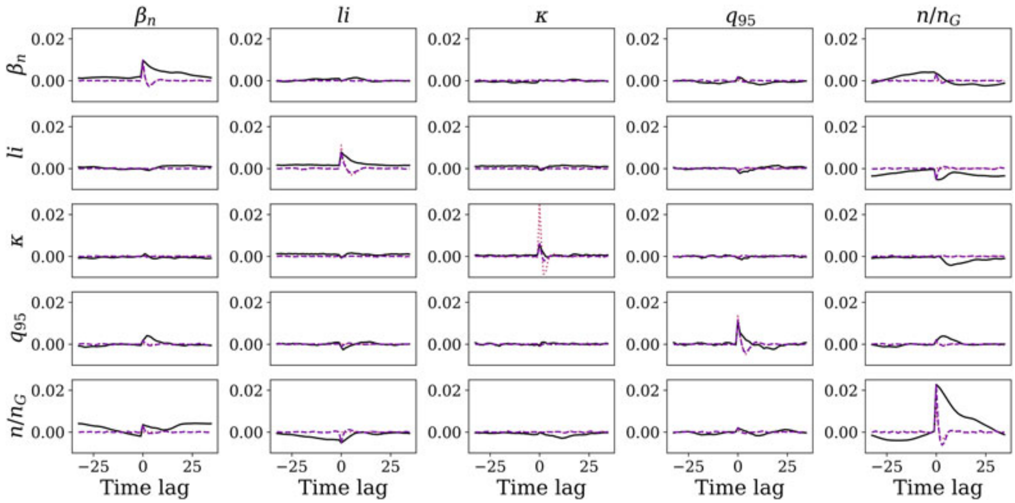

FIGURE 8. Comparison of cross-covariance of training data and generated data with cross-covariance (solid lines on top of each other, numerical error of order 10^{-16}), covariance or sampled covariance (dashed lines) post-processing and uncorrelated model (dotted line) for test case (ii).

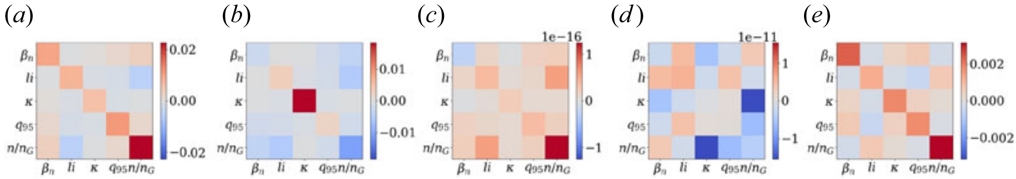

FIGURE 9. Comparison of covariance of training data (*a*) and difference of generated data (*b*) with uncorrelated model, (*c*) empirical covariance, (*d*) cross-covariance and (*e*) sampled covariance post-processing for test case (ii). Note the different scaling in the colour scale.

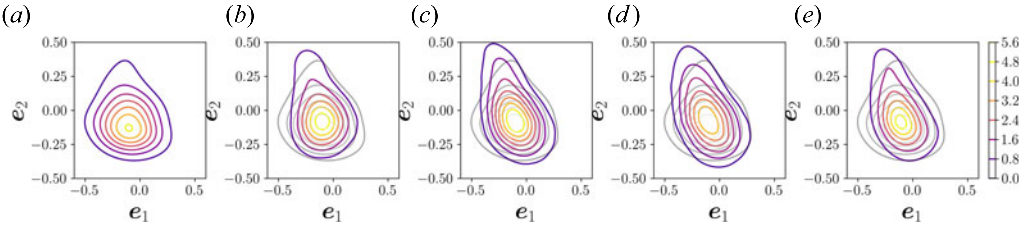

FIGURE 10. Kernel density estimation of the 2-D kernel PCA embedding of the (*a*) training data and generated data via (*b*) uncorrelated model, (*c*) empirical covariance, (*d*) cross-covariance and (*e*) sampled covariance post-processing for test case (ii). The embedded training data are shown in grey in all plots. The colour scale representing the density is the same in all plots.

of the generated data. Again, the results show that the generated data lives on the same submanifold for all four post-processing methods. In table 6, the $F1$ score for DTW SOM clustering is given.

Train	Test	Training	Uncorrelated	Emp. cov	Emp. crosscov	Sample cov
original	generated	1.0	1.0	1.0	0.94	1.0
generated	original	1.0	0.91	1.0	1.0	1.0
mix	mix	1.0	1.0	1.0	0.96	1.0

TABLE 6. The $F1$ score for DTW SOM clustering of different post-processing methods for test case (ii).

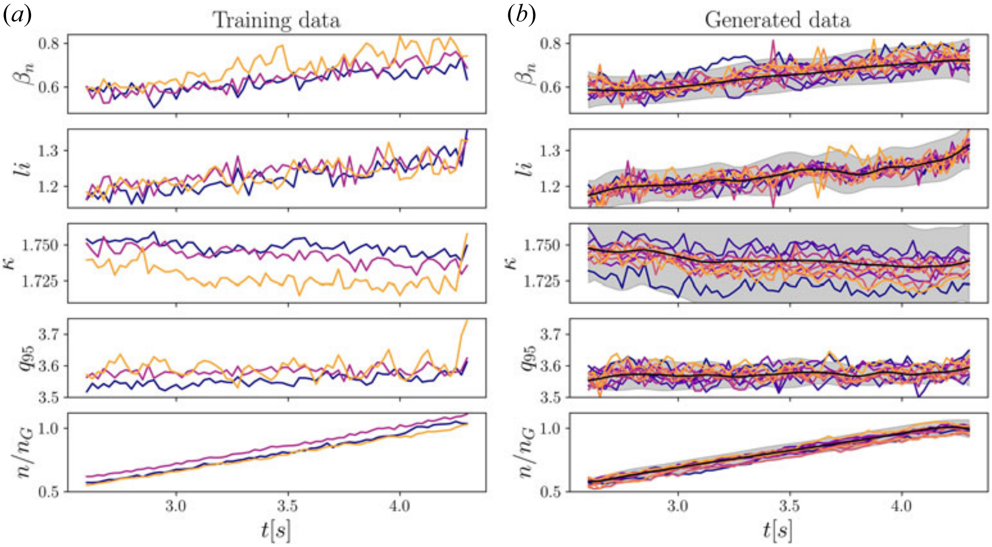

FIGURE 11. (a) Training data and (b) 10 generated data sets from the state space Student t surrogate model together with the estimated mean (black solid line) and 95 % confidence (grey shaded region) for test case (iii). Different colours correspond to different shots of training data and different samples of the generated data, respectively.

Metric	Uncorrelated	Emp. cov	Emp. crosscov	Sample cov
W_1	0.071 ± 0.088	0.030 ± 0.026	$\mathbf{0.025 \pm 0.019}$	0.03 ± 0.028
MMD p-value	0.43 ± 0.32	0.86 ± 0.17	0.84 ± 0.09	$\mathbf{0.87 \pm 0.13}$
MSE ρ_{rs}	0.0083 ± 0.0075	0.0076 ± 0.007	$\mathbf{0.0019 \pm 0.0019}$	0.0070 ± 0.007
W_{emb}	0.1808 ± 0.0034	0.0621 ± 0.0011	$\mathbf{0.0517 \pm 0.0008}$	0.0656 ± 0.0012
MSE PSD $[10^{-6}]$	240 ± 33	11 ± 19	$\mathbf{0.8 \pm 0.5}$	79 ± 13
DTW	1.7 ± 2.2	0.9 ± 0.6	$\mathbf{0.8 \pm 0.6}$	0.9 ± 0.6
MSE mfPCA	0.138	0.021	**0.010**	0.025

TABLE 7. Post-processing method comparison for test case (iii). Mean and standard deviation over five dimensions and $N = 1000$ samples generated from the trained model for statistical metrics described in § 3. Best values are highlighted in bold.

https://doi.org/10.1017/S0022377822000769 Published online by Cambridge University Press

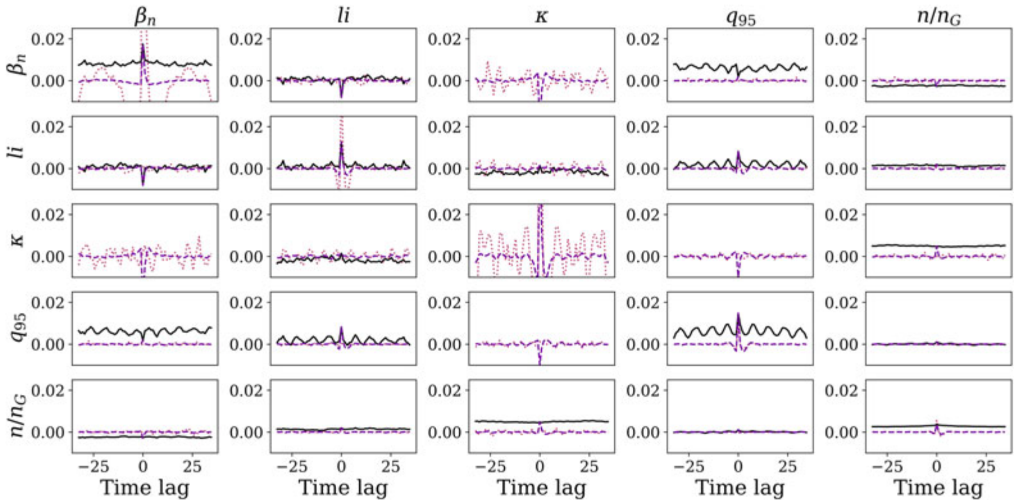

FIGURE 12. Comparison of cross-covariance of training data and generated data with cross-covariance (solid lines on top of each other, numerical error of order 10^{-16}), covariance or sampled covariance (dashed lines) post-processing and uncorrelated model (dotted line) for test case (iii).

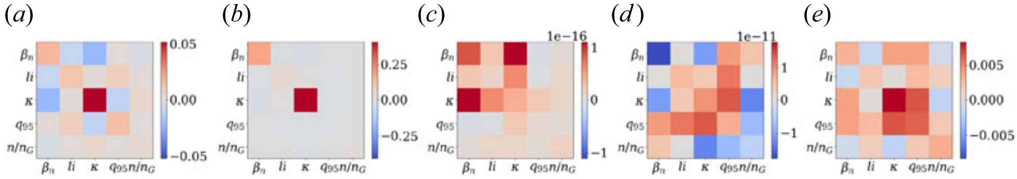

FIGURE 13. Comparison of covariance of training data (*a*) and difference of generated data (*b*) with uncorrelated model, (*c*) empirical covariance, (*d*) cross-covariance and (*e*) sampled covariance post-processing for test case (iii). Note the different scaling in the colour scale.

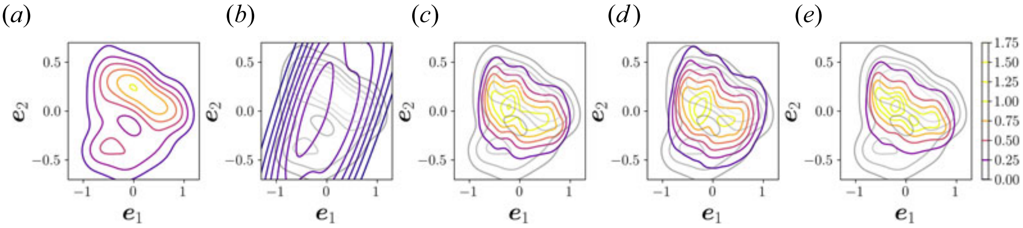

FIGURE 14. Kernel density estimation of the 2-D kernel PCA embedding of the (*a*) training data and generated data via (*b*) uncorrelated model, (*c*) empirical covariance, (*d*) cross-covariance and (*e*) sampled covariance post-processing for test case (iii). The embedded training data are shown in grey in all plots. The colour scale representing the density is the same in all plots.

C.2. *Test case (iii): density accumulation*

For the third test case with a disruption occurring due to density accumulation (shots 166933, 166934, 166937), the visual comparison is given in figure 11 followed by the results of the statistical analysis in table 7. The cross-covariance and covariance are

https://doi.org/10.1017/S0022377822000769 Published online by Cambridge University Press

Train	Test	Training	Uncorrelated	Emp. cov	Emp. crosscov	Sample cov
original	generated	0.81	0.98	1.0	0.85	0.99
generated	original	0.99	0.92	0.93	0.93	0.93
mix	mix	0.96	0.96	0.96	0.94	0.97

TABLE 8. The $F1$ score for DTW SOM clustering of different post-processing methods for test case (iii).

displayed in figures 12 and 13, respectively. The embedding is shown in figure 14. Here, the skew of the embedding caused by the broad distribution of κ is not perfectly reproduced by the generated data. However, the results should be regarded with caution as only 3 training data samples are available in this test case. This presents also a limit to this metric. However, when looking at samples of the generated data shown in figure 11, this broad range present in the training data is still well reproduced by the generated data. In this test case, the uncorrelated model performs worst as correlations are not reproduced. The results for generated data with included correlations are again of the same order of magnitude as for test cases (i) and (ii). The results obtained for the $F1$ score for DTW SOM clustering are given in table 8.

REFERENCES

AGGARWAL, C.C. 2018 *Neural Networks and Deep Learning*. Springer.

AYMERICH, E., SIAS, G., PISANO, F., CANNAS, B., CARCANGIU, S., SOZZI, C., STUART, C., CARVALHO, P.J., FANNI, A. & JET CONTRIBUTORS 2022 Disruption prediction at JET through deep convolutional neural networks using spatiotemporal information from plasma profiles. *Nucl. Fusion* **62** (6), 066005.

BERKERY, J.W., SABBAGH, S.A., BELL, R.E., GERHARDT, S.P. & LEBLANC, B.P. 2017 A reduced resistive wall mode kinetic stability model for disruption forecasting. *Phys. Plasmas* **24** (5), 056103.

BERNDT, D.J. & CLIFFORD, J. 1994 Using dynamic time warping to find patterns in time series. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining, AAAIWS'94*, pp. 359–370. AAAI.

BONNEEL, N., RABIN, J., PEYRÉ, G. & PFISTER, H. 2015 Sliced and radon wasserstein barycenters of measures. *J. Math. Imag. Vis.* **1** (51), 22–45.

BOYLE, P. & FREAN, M. 2004 Dependent Gaussian processes. In *Advances in Neural Information Processing Systems* (ed. L. Saul, Y. Weiss & L. Bottou), vol. 17. MIT.

CLEVELAND, R.B., CLEVELAND, W.S., MCRAE, J.E. & TERPENNING, I. 1990 STL: a seasonal-trend decomposition procedure based on loess (with discussion). *J. Off. Stat.* **6**, 3–73.

DURBIN, J. & KOOPMAN, S.J. 2002 A simple and efficient simulation smoother for state space time series analysis. *Biometrika* **89** (3), 603–615.

FLAMARY, R., COURTY, N., GRAMFORT, A., ALAYA, M.Z., BOISBUNON, A., CHAMBON, S., CHAPEL, L., CORENFLOS, A., FATRAS, K.F., NEMO, G., *et al.* 2021 POT: Python optimal transport. *J. Mach. Learn. Res.* **22** (78), 1–8.

GRETTON, A., BORGWARDT, K.M., RASCH, M.J., SCHÖLKOPF, B. & SMOLA, A. 2012 A kernel two-sample test. *J. Mach. Learn. Res.* **13**, 723–773.

HAPP, C. & GREVEN, S. 2018 Multivariate functional principal component analysis for data observed on different (dimensional) domains. *J. Am. Stat. Assoc.* **113** (522), 649–659.

IWANA, B.K. & UCHIDA, S. 2021 An empirical survey of data augmentation for time series classification with neural networks. *PLoS ONE* **16** (7), 1–32.

KANG, Y., HYNDMAN, R.J. & LI, F. 2020 Gratis: Generating time series with diverse and controllable characteristics. *Stat. Anal. Data Min.* **13** (4), 354–376.

https://doi.org/10.1017/S0022377822000769 Published online by Cambridge University Press

KATES-HARBECK, J., SVYATKOVSKIY, A. & TANG, W. 2019 Predicting disruptive instabilities in controlled fusion plasmas through deep learning. *Nature* **568** (7753), 526–531.

KESSY, A., LEWIN, A. & STRIMMER, K. 2018 Optimal whitening and decorrelation. *Am. Stat.* **72** (4), 309–314.

MONTES, K.J., REA, C., TINGUELY, R.A., SWEENEY, R., ZHU, J. & GRANETZ, R.S. 2021 A semi-supervised machine learning detector for physics events in tokamak discharges. *Nucl. Fusion* **61** (2), 026022.

MURPHY, K.P. 2022 *Probabilistic Machine Learning: An Introduction*. MIT.

NEAL, R.M. 1997 Monte Carlo Implementation of Gaussian Process Models for Bayesian Regression and Classification. (Technical report, University of Toronto, Department of Statistics). University of Toronto.

PARK, K.I. 2017 *Fundamentals of Probability and Stochastic Processes with Applications to Communications*. Springer.

PAU, A., FANNI, A., CARCANGIU, S., CANNAS, B., SIAS, G., MURARI, A. & RIMINI, F. 2019 A machine learning approach based on generative topographic mapping for disruption prevention and avoidance at JET. *Nucl. Fusion* **59** (10), 106017.

REA, C. & GRANETZ, R.S. 2018 Exploratory machine learning studies for disruption prediction using large databases on DIII-D. *Fusion Sci. Technol.* **74** (1–2), 89–100.

REA, C., MONTES, K.J., ERICKSON, K.G., GRANETZ, R.S. & TINGUELY, R.A. 2019 A real-time machine learning-based disruption predictor in DIII-D. *Nucl. Fusion* **59** (9), 096016.

REA, C., MONTES, K.J., PAU, A., GRANETZ, R.S. & SAUTER, O. 2020 Progress toward interpretable machine learning–based disruption predictors across tokamaks. *Fusion Sci. Technol.* **76** (8), 912–924.

ROTH, M., ARDESHIRI, T., ÖZKAN, E. & GUSTAFSSON, F. 2017 Robust Bayesian filtering and smoothing using student's *t* distribution. CoRR abs/1703.02428, arXiv:1703.02428.

SÄRKKÄ, S. 2013 *Bayesian Filtering and Smoothing*. Institute of Mathematical Statistics Textbooks. Cambridge University Press.

SÄRKKÄ, S. & SOLIN, A. 2019 *Applied Stochastic Differential Equations*. Institute of Mathematical Statistics Textbooks. Cambridge University Press.

SCHÖLKOPF, B., SMOLA, A. & MÜLLER, K.-R. 1998 Nonlinear component analysis as a kernel eigenvalue problem. *Neural Comput.* **10** (5), 1299–1319.

SHAH, A., WILSON, A.G. & GHAHRAMANI, Z. 2014 Student-*t* processes as alternatives to Gaussian processes. *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics* **PMLR: 33**, 877–885.

SHORTEN, C. & KHOSHGOFTAAR, T.M. 2019 A survey on image data augmentation for deep learning. *J. Big Data* **6**, 1–48.

SOLIN, A. & SÄRKKÄ S. 2015 State space methods for efficient inference in student- process regression. *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics* ***PMLR:*** **38**, 885–893.

VANDENBERG-RODES, A. & SHAHBABA, B. 2015 Dependent matérn processes for multivariate time series. https://arxiv.org/abs/1502.03466.

VANHATALO, J., JYLÄNKI, P. & VEHTARI, A. 2009 Gaussian process regression with student-*t* likelihood. In *Advances in Neural Information Processing Systems* (ed. Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams & A. Culotta), vol. 22. Curran Associates, Inc.

VETTIGLI, G. 2018 Minisom: minimalistic and numpy-based implementation of the self organizing map. https://github.com/JustGlowing/minisom/.

VILLANI, C. 2008 *Optimal Transport: Old and New*. Grundlehren der mathematischen Wissenschaften. Springer.

VIRTANEN, P., GOMMERS, R., OLIPHANT, T.E., HABERLAND, M., REDDY, T., COURNAPEAU, D., BUROVSKI, E., PETERSON, P., WECKESSER, W., BRIGHT, J., *et al.* 2020 SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nat. Meth.* **17**, 261–272.

WEN, Q., GAO, J., SONG, X., SUN, L., XU, H. & ZHU, S. 2019 RobustSTL: a robust seasonal-trend decomposition algorithm for long time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 5409–5416.

https://doi.org/10.1017/S0022377822000769 Published online by Cambridge University Press

WEN, Q., SUN, L., YANG, F., SONG, X., GAO, J., WANG, X. & XU, H. 2021 Time series data augmentation for deep learning: a survey. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization.

WILKINSON, W.J., CHANG, P.E., ANDERSEN, M.R. & SOLIN, A. 2020 State space expectation propagation: efficient inference schemes for temporal Gaussian processes. *Proceedings of the 37th International Conference on Machine Learning* **PMLR: 119**, 10270–10281.

WILLIAMS, C.K.I. & RASMUSSEN, C.E. 1996 Gaussian processes for regression. In *Advances in Neural Information Processing Systems*, vol. 8, pp. 514–520. MIT.

YOON, J., JARRETT, D. & VAN DER SCHAAR, M. 2019 Time-series generative adversarial networks. In *Advances in Neural Information Processing Systems* (ed. H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox & R. Garnett), vol. 32. Curran Associates, Inc.

https://doi.org/10.1017/S0022377822000769 Published online by Cambridge University Press