Physica D 453 (2023) 133831

Contents lists available at ScienceDirect

Physica D

journal homepage: www.elsevier.com/locate/physd

Modern AI *versus* century-old mathematical models: How far can we go with generative adversarial networks to reproduce stochastic processes?

Pedro Lencastre [a,b,c], Marit Gjersdal [a], Leonardo Rydin Gorjão [d], Anis Yazidi [a,b,c], Pedro G. Lind [a,b,c,*]

[a] Department of Computer Science, OsloMet – Oslo Metropolitan University, P.O. Box 4 St. Olavs plass, N-0130, Oslo, Norway
[b] OsloMet Artificial Intelligence lab, OsloMet, Pilestredet 52, N-0166, Oslo, Norway
[c] NordSTAR - Nordic Center for Sustainable and Trustworthy AI Research, Pilestredet 52, N-0166, Oslo, Norway
[d] Faculty of Science and Technology, Norwegian University of Life Sciences, 1432, Ås, Norway

ARTICLE INFO

Article history:
Received 5 December 2022
Received in revised form 26 April 2023
Accepted 21 June 2023
Available online 28 June 2023
Communicated by R. Kuske

Keywords:
Generative Adversarial Networks
GANs
Markov models
Time-series prediction
Eye-tracking data

ABSTRACT

The usage of generative adversarial networks (GAN)s for synthetic time-series data generation has been gaining popularity in recent years with applications from finance to music composition and processing of textual content. However, beyond their reported success, few comparisons exist with other artificial intelligence (AI) methods or standard mathematical models. Here, we test GANs performance, comparing them with a well-known mathematical model, namely a Markov chain. We implement comparative metrics based on one- and two-point statistics to evaluate the performance of each method. We find that, similarly to other AI approaches, GANs struggle to capture rare events and cross-feature relations and are unable to create synthetic faithful data. GANs are relatively successful in replicating the auto-correlation function, but they still lag significantly behind simple Markov chains. We also provide a qualitative explanation for this limitation of AI approaches.

1. Introduction

With the improvements in computation capabilities and the increasing accessibility of large amounts of data, non-parametric models of time-series have become increasingly popular. It is easy to understand why that is the case: non-parametric models have an almost universal application without the laborious task of understanding and reducing the problem to a core of fundamental quantities to describe a system, often with inaccurate but necessary simplifications.

In this context, artificial neural networks (ANNs) based algorithms are used, e.g. to predict wind speeds [1], wind power output [2], air quality [3], the evolution of the price financial assets [4], the rate of infections during a pandemic outbreak [5] or the number of patients arriving at a hospital's emergency services [6].

While it is obvious that being able to predict the future of statistical quantities is useful, these methods can further be used in data augmentation when the original data is insufficient, or, e.g. in the medical domain, to produce anonymised data that can be shared without strict ethical constraints.

Among these methods, one of the most popular is generative adversarial networks (GANs) [7], which, after their success in faithfully creating realistic images, have recently become popular in time-series replication [8]. GANs consider two "coupled" ANNs playing a zero-sum game. The first ANN is called "generator" and creates synthetic data with the objective to "fool" the second ANN. This latter ANN is called "discriminator" and tries to distinguish if a particular set of data is synthetic or not.

GANs have been applied to biomedical signal data, where they have been used to model the time-evolution of data from electrocardiogram, electroencephalogram, electromyography and photoplethysmography [9]. GANs have also been used in natural language processing [10], generating music [11,12], predicting pedestrian trajectory [13] and in predicting the evolution of financial assets [14]. In the realm of biomedical signal data, one type of time-series with several potential applications is eye-tracking data, which can be used to determine personality traits [15], drug consumption habits [16,17], as well as diagnosing attention-deficit hyperactivity disorder [18,19] and autism [20].

In this paper, we will test the performance of GANs in reproducing stochastic trajectories and compare them with some

* Corresponding author at: Department of Computer Science, OsloMet – Oslo Metropolitan University, P.O. Box 4 St. Olavs plass, N-0130, Oslo, Norway.
E-mail address: pedro.lind@oslomet.no (P.G. Lind).

https://doi.org/10.1016/j.physd.2023.133831

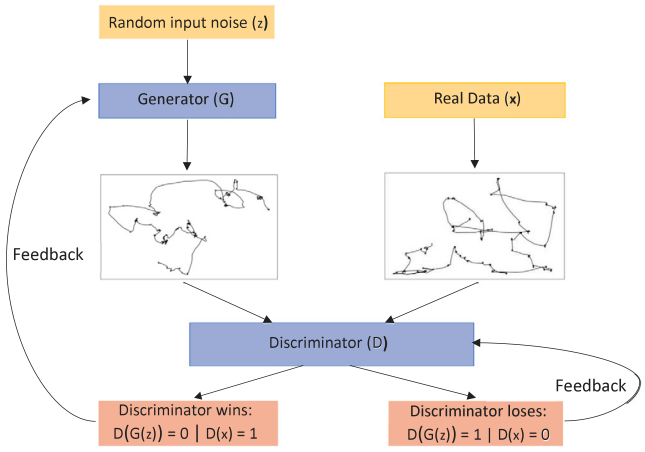

Fig. 1. Representation of a GAN: the generator tries to create data as faithfully as possible, while the discriminator attempts at distinguishing real data from that produced by the generator.

benchmarks from stochastic modelling. Eye-tracking data are particularly appropriate to evaluate GANs' performance for two reasons. Firstly, eye-tracking data reflect a broad panoply of footprints flagging particular health states and features of human behaviour. Secondly, they have very large kurtosis, i.e. they are prone to extreme events, typically due to fast gaze relocations in-between two areas of interest. These extreme events lead to heavy-tailed distribution and ANN-based methods have been shown to struggle to capture these distributions' tails.

Thus, within the family of non-parametric models, we compare GANs' performance with that of Markov models, which are a benchmark to describe stochastic time series [21]. Similarly to AI methods, Markov models can be used to replicate a given set of data without the knowledge of the underlying process. When compared to its AI counterparts, Markov models have some advantages: they can provide a unique time-continuous description of the system, or, alternatively, assert that a given process is not time-homogeneous (stationary) or time-continuous [22]. Furthermore, in a Markov model each parameter of the model has a straightforward meaning and thus an inspection of the model can give us information about the system, something that does not generally happen for ANN-based methods.

We start in Section 2 by describing the technical details in the implementation of different GAN architecture and Markov chains. In Section 3 we describe the data that will be used to test GANs and Markov models. Besides eye-tracking data we will consider synthetic data. Section 4 focuses on comparing the results obtained when using GANs and Markov chains for both synthetic and empirical data and finally, Section 5 concludes the manuscript.

2. Algorithms and methods

2.1. GANs architectures

GANs are a subset of AI algorithms following the structure represented in Fig. 1. The generator is initialised with a random noise vector z_0 as input from which a time-series is generated. This time-series is then analysed by the discriminator, which will get either real data x or data generated by the generator, z, and will try to distinguish between both.

These two networks are trained together in a *min–max* game fashion [23]. The discriminator is trained to maximise its correct labelling of the input as real or fake, while the generator tries to minimise it and tries to "fool" the discriminator. Ideally, this simultaneous adversarial training will eventually lead to the generator learning to create outputs that mimic the statistical properties of the original data: as the discriminator gets more accurate so must the generator in order to fool it.

Table 1
The generator and the discriminator used in each of the GAN architectures addressed in this paper, cf. Fig. 1.

GAN	Generator	Discriminator
RCGAN	LSTM	LSTM
TimeGAN	bidirect. LSTM	bidirect. LSTM
SigCWGAN	AR-FNN	C-Sig-W_1
RCWGAN	AR-FNN	AR-FNN

Mathematically, a GAN implementation considers the following objective function

$$\mathcal{L}(G(z), D(x)) = \mathbb{E}_{x\sim\rho_x}\left[\log D(x)\right] + \mathbb{E}_{z\sim\rho_Z}\left[\log\left(1 - D(G(z))\right)\right], \tag{1}$$

where $D(x) \in [0, 1]$ is the discriminator-assigned probability of x being a real data point. Thus, $\mathbb{E}_{x\sim\rho_x}[\log D(x)]$ is the expected fraction of correct guesses by the discriminator about real data series x, among all trials in which the discriminator is evaluating real data. In this context, G is the function characterising the generator which maps an input noise series z into a series $G(z)$. See Fig. 1. Therefore, $1 - D(G(z))$ represents the probability the discriminator assigns of $G(z)$ correctly being labelled as generated data. The probability distributions of the real data series x, of the input noise z and of the generated series $G(z)$ are represented as ρ_x, ρ_z and ρ_G, respectively.

The task of the generator is to bring the probability distribution ρ_G of the generated series $G(z)$ as close as possible to the distribution ρ_x of real data series x, such that both terms on the right-hand side are decreased. Simultaneously, the discriminator will try to maximise both expected values, one by maximising $D(x)$, and the other by minimising $D(G(z))$. Therefore, having defined the objective function in Eq. (1), the optimisation scheme which trains the GAN model solves the *min–max* problem

$$\min_G \max_D \mathcal{L}(G(z), D(x)). \tag{2}$$

Within the general GAN framework, several architectures are possible with different types of ANNs as generators and discriminators. In this paper, we consider a selection of different architectures, indicated in Table 1 together with the specific types of generators and discriminators. Details on how to train GAN architectures are given in Appendix B.

2.1.1. Recurrent conditional GAN

Recurrent conditional GAN (RCGAN) was one of the first time-series GAN models introduced [24], as a model for replicating multi-valued time-series of medical data, namely the evolution of the health state of patients in emergency care, with the motivation to both predict the patient outcomes and generate faithful synthetic data unconstrained from privacy concerns.

As the name suggests, it uses recurrent neural networks for the generator and discriminator with Long Short-Term Memory (LSTM) cells [25].

2.1.2. TimeGAN

TimeGAN was introduced in 2019 with the aim to outperform several state-of-the-art GANs [26], presenting concrete examples from stock market and energy consumption data. TimeGAN combines a classical GAN architecture with an autoencoder a type of ANN composed of two networks, one that "encodes" a series into

symbols and another one that "decodes" the symbols back into a series [27]. TimeGAN consists of four networks. In the classical GAN part of the algorithm, a recurrent neural network (RNN) is used as a generator and a bidirectional LSTM as a discriminator. Instead of giving the discriminator the original real data as input to classify it as real or fake, it is given the encoded vector of the real data, after being processed by an autoencoder.

TimeGAN uses three loss functions to control the training: Reconstruction loss; supervised loss; and unsupervised loss. The reconstruction loss is calculated using the output of the decoder and is used to calculate the gradients for the encoder and the decoder. The unsupervised loss is based on the discriminator's output and, similarly to the *min–max* loss function of the original GAN Eq. (1), is used for calculating the generator and discriminators gradients. Finally, the supervised loss is calculated based on both the generator's output and the encoder's output of the real data, and is used for calculating the gradients of the generator and the encoder. Some authors implement TimeGAN with the GAN part only without the autoencoder part of the algorithm [28]. We here follow that implementation.

2.1.3. SigCWGAN and RCWGAN

The Conditional Sign-Wasserstein GAN (SigCWGAN) was designed with the purpose to capture temporal dependencies in multi-dimensional time-series data while being able to correctly model the tail of its underlying distribution [28]. To capture this, a new metric is introduced, namely the so-called Signature Wasserstein-1 (C-Sig-W_1), which serves as the discriminator of the GAN. The usage of this metric is supposed to be not only more robust but also computationally less expensive to train than a typical ANN. For further mathematical details about the discriminator the interested reader should consult Ref. [28].

In this architecture, the authors introduce as a generator a three-layer feed-forward neural network residual connections and parametric ReLUs as activation functions, which determines if the output of a neuron activates the next neuron or not. The authors call this ANN, Autoregressive Feed-Forward Neural Network (AR-FNN). The generator has the main aim to capture auto-regressive processes and thus be able to capture the time-dependency of the process. More details can be found in the appendix of the original paper [28].

The authors of SigCWGAN also implemented a GAN where both the generator and discriminator are RNNs with the aforementioned AR-FNN cells. This architecture is called Recurrent Conditional Wasserstein GAN (RCWGAN). In this paper, we will consider this architecture.

The original author's pytorch implementation of the all previously mentioned GAN algorithms can be found in Github [29].

2.2. Markov-chain models for reproducing time-series

Markov models were introduced in 1906 [30] with the general aim to model conditional probabilities and thus being able to provide a description of the time-evolution of a stochastic process. They were implemented firstly in an effort to estimate the probability of finding a vowel in a text, based on the knowledge of the previous letter [21].

By definition, a time-series X_t is said to follow a Markov process if it fulfils the Markov property:

$$\Pr(X_{t=j} = \hat{x}_j \mid X_{t=j-1} = \hat{x}_{j-1}, \dots, X_{t=0} = \hat{x}_0) = \Pr(X_{t=j} = \hat{x}_j \mid X_{t=j-1} = \hat{x}_{j-1}), \tag{3}$$

for all positive integers j, where capital letters mean stochastic variables at different time steps and lowercase letters are the respective values of those variables. Successive values are measured at constant time intervals Δt.

The Markov condition implies that, at all times, any prediction on the future of time series X_t depends only on the current state of the system and not on past states. For that reason, Markov processes are often said to be "memoryless". The Markov property is a simplification that, in the strict sense, does not apply to most natural systems, but it is very convenient because by computing the conditional probability $\Pr(X_{t=j} = \hat{x}_j \mid X_{t=j-1} = \hat{x}_{j-1})$ we can describe the full time-evolution of the system, i.e. two-point statistics contains all the information about the process. More details on how to generate a Markov process in this way are given in Appendix C.

3. Data and evaluation metrics

3.1. Synthetic data

The first set of data analysed in this paper is a synthetically generated Vector Auto-Regressive (VAR) process. In one dimension, a VAR(p) process assumes that the observable at the present time t is defined from its values in the previous p observations apart a small random noise. In our case, the observable is the velocity of the eye-gaze, which means that we consider spatial increments of the positions in both x and y direction. For increments (ΔX) the VAR (p) model reads

$$\Delta X_t = \sum_{n=1}^{p} \phi_n \Delta X_{t-n} + \xi_t(\sigma), \tag{4}$$

where ξ is normally distributed random number with zero mean and σ standard deviation. Consequently, in a VAR(p) process, apart from Gaussian fluctuations, the future increments are defined through a linear combination of the last p increments. Again, the time labelling is done indicating the number of elementary constant time intervals Δt. Here we take $\Delta t = 1$ and generate around 85 thousand points (the same number as the empirical eye-tracking dataset we use), given the initial condition $\Delta X_0 = 0$

In what follows, we consider a VAR(1) process, meaning that future increments are determined by a random number and the increment immediately preceding it. We consider, however, a VAR(1) process on a two-dimensional plane (X and Y), with some correlation between ΔX and ΔY, given by σ_{XY}. Our process is thus defined by the system of equations

$$\begin{aligned} \Delta X_t &= \phi_X \Delta X_{t-1} + \xi_t^{(X)}(\sigma_X, \sigma_{XY}), \\ \Delta Y_t &= \phi_Y \Delta Y_{t-1} + \xi_t^{(Y)}(\sigma_Y, \sigma_{YX}), \end{aligned} \tag{5}$$

where $\xi_t^{(X)}(\sigma_X, \sigma_{XY})$ represents the stochastic fluctuations in the X-dimension, with zero mean, standard deviation σ_X and a correlation σ_{XY} with the stochastic fluctuations in the Y-dimension. We will consider a process purely isotropic, i.e. $\sigma_X = \sigma_Y \equiv \sigma$ and $\phi_X = \phi_Y \equiv \phi$.

The VAR(1) process is one of the most simple synthetic time-series that are correlated in time. It is, by construction, a Markov process, allowing us to test the accuracy of our implemented model, and, for $\phi > 0$, it is suitable to test the commonly mentioned limitation of NN and GAN algorithms in modelling the tails of a distribution. In Fig. 2 (top) a two-dimensional VAR(1) process is represented, with $\sigma = 1$, $\phi = 0.8$ and $\sigma_{XY} = 0.8$, with an inset showing the respective scatter plot of the increments in both X- and Y-directions. From this inset, we observe that indeed ΔX and ΔY are positively correlated, with a clear propensity of the values to lay on the main diagonal. Furthermore, we observe that values on the extremes of ΔX (corr. ΔY) tend to follow also extreme values of ΔX (corr. ΔY), thus illustrating the positive auto-correlation of both coordinates ($\phi = 0.8$). In this paper we will use three types of VAR processes, namely with the values of $p = 1, 2$ and 3.

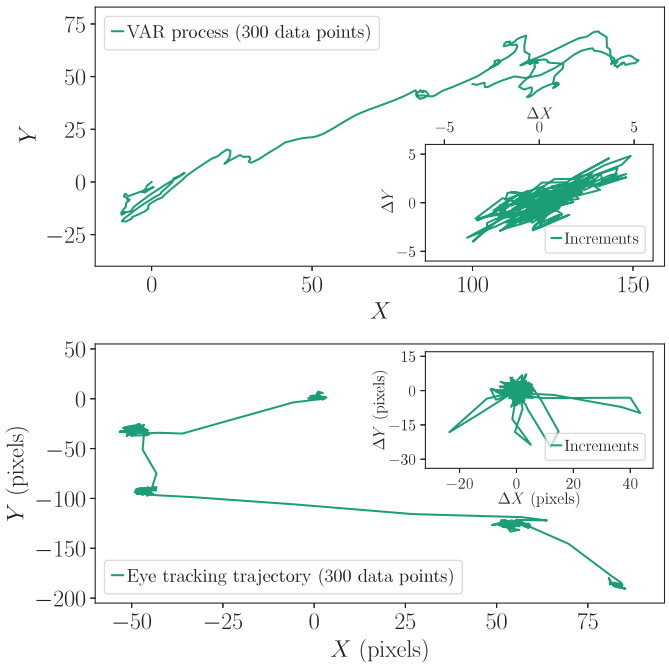

Fig. 2. Top: Representation of a trajectory of two dimensional VAR(1) process (left) and its corresponding increments (right). This time-series was synthetically generated with positive auto-correlation $\phi = 0.8$ and positive feature correlation $\sigma_{XY} = 0.8$. Thus, values are distributed mainly on the diagonal corresponding to the first and third quadrants. Bottom: Representation of a gaze trajectory (left) and its corresponding increments (right). This series was empirically estimated with a modern eye-tracker. Typically, gaze positions are concentrated on a limited area which is alternated with fast relocation trajectories.

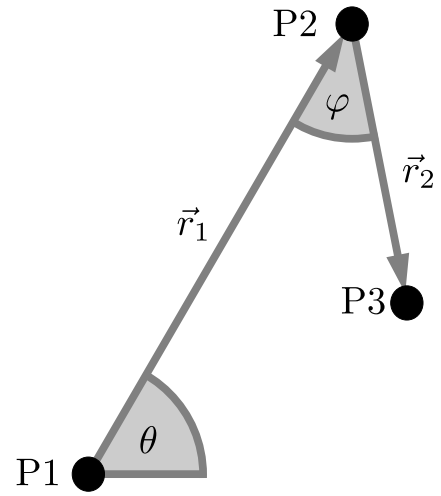

Fig. 3. Illustration of some of the relevant quantities to describe gaze trajectories, namely gaze velocity, gaze direction (θ) and the angle between two consecutive gaze relocations (φ). Here, the two steps $\vec{r}_1$ and $\vec{r}_2$ occurred within constant time-lags Δt. Consequently, the velocity magnitude is given by, e.g. $\|\vec{v}_1\| = \|\vec{r}_1\|/\Delta t$.

3.2. Eye-tracking data

The eye-tracking data was collected at Oslo Metropolitan University with the Eye-link Duo, a state-of-the-art equipment with a maximum frequency of 2000 Hz and a precision of 0.1 degrees of visual angle. Here we have downsampled our data to a frequency of 200 Hz and blinks have been removed, yielding around 85 thousand data-points. Units of X and Y are presented in pixels of the viewing screen and eye-tracking data was extracted while the participant was engaged in trying to find pre-selected targets. For this purpose, images from the book *Where's Wally?* were used. Eight pictures were used, each one for two minutes. It is very unlikely that two minutes are enough to find all the pre-selected targets in a large image and the experiment was set up this way to make sure that a participant is kept engaged throughout the experiment. The data was collected following all ethical requirements approved by the Norwegian Center for Research Data (*Norsk senter for forskningsdata*), with the application with Reference Number 176347.

In Fig. 2 (bottom) we see an illustration of a gaze trajectory, with, in the inset, its corresponding increments. As described in the literature, we observe periods where the eyes are fixated around a point (called fixations) and periods of relocations (called saccades). Indeed we observe some extreme velocity events and the velocity time-series presents an excess kurtosis of $\kappa \sim 50$.

The presence of extreme events is a welcomed feature of gaze trajectories that makes them particularly suitable to study the performance of ANN based methods. It has been widely reported that ANN-based methods struggle to capture extreme events [2] with some models analysed here explicitly stating their ability to overcome this limitation in replicating the tails of a distribution [28]. Real data used to assess ANN performance has a considerably small kurtosis, $5 < \kappa < 15$, and thus includes much less extreme events. There are two important examples of non-Markov behaviour. One is the so-called inhibition of return (IoR) [31], a known mechanism by which, for some time, the human gaze avoids visiting the same area after it has recently left it. The other is screen confinement, by which eye-tracker gaze trajectories are restricted to a screen area, and which introduces non-trivial long-range dependencies on the series of gaze velocities.

Three quantities are important to describe gaze trajectories: the velocity magnitude, $\|v\|$, the angle, θ, of a given velocity with the horizontal axis, and the angle, φ, between two consecutive measurements. All three quantities are illustrated in Fig. 3. In what follows we will access the performance of a model by its ability to replicate the distribution of these three quantities, together with the time evolution of the velocity magnitude $\|v\|$.

3.3. Evaluation metrics

The question of which metrics are preferable in evaluating the performance of a GAN is open to debate [32–34]: most research on the topic is focused on GANs' applications to images.

In Ref. [24] the authors assess the similarity between a sample of synthetically generated data and an empirical original sample, by employing a metric called *maximum mean discrepancy*. With it, the average fluctuations of the values in both synthetic and empirical samples are compared with the difference between samples. The authors also introduce a scheme called *train-on-synthetic-test-on-real* (TSTR). It requires labels for the generated data, as well as a supervised learning model that is able to classify well the original data when trained with it. Then, the synthetic generated data is first used to train the classifier, and finally, the classifier is tested with real data. If it is able to correctly label that data, it means that synthetic fully anonymous data is suited to create models with real-world predictive power. In Ref. [26] the authors use a statistical method for visualising high-dimensional data in a two-dimensional map, called *t-distributed stochastic neighbour embedding*. This method enables one to visualise the distribution of the data. To study the preservation of correlations between different dimensions of the data, principal component analysis is used. While the profile of auto-correlations is not directly assessed, the authors compare conditional probability

Table 2
Moments of the probability distribution function of $|v|$ for each dataset and the corresponding Markov/GAN generated data. We see that, when it comes to modelling the distribution of $|v|$, the Markov model is relatively successful. GAN models, however, consistently underestimate the values of the moments of the distribution. In particular, GAN models have difficulty capturing long tails of the distribution, as can be seen by the reduced values of the kurtosis. The only exception is TimeGAN for the VAR(2) dataset, but notice that, in this case, the value of the standard deviation is largely underestimated. The difficulty in replicating the moments of the $|v|$-distribution is even more pronounced in the empirical case of gaze trajectories which is significantly more complex than a VAR process.

		Mean	Standard deviation	Skewness	Excess of kurtosis
VAR(1)	Data	2.0	1.27	1.1	1.39
	Markov	2.00 ± 10^{-2}	1.30 ± 10^{-2}	$1.10 \pm 3 \cdot 10^{-2}$	1.4 ± 10^{-1}
	RC	$1.00 \pm 4 \cdot 10^{-2}$	$6.8 \cdot 10^{-1} \pm 5 \cdot 10^{-2}$	$1.2 \pm 2 \cdot 10^{-1}$	2 ± 1
	Time	$1.10 \pm 2 \cdot 10^{-2}$	$7.1 \cdot 10^{-1} \pm 1 \cdot 10^{-2}$	$7.8 \cdot 10^{-1} \pm 6 \cdot 10^{-2}$	$3 \cdot 10^{-1} \pm 2 \cdot 10^{-1}$
	SIGCW	$1.20 \pm 2 \cdot 10^{-2}$	$7.2 \cdot 10^{-1} \pm 2 \cdot 10^{-2}$	$1.00 \pm 7 \cdot 10^{-2}$	$1.0 \pm 3 \cdot 10^{-1}$
	RCW	$1.40 \pm 3 \cdot 10^{-2}$	$8.7 \cdot 10^{-1} \pm 2 \cdot 10^{-2}$	$9.3 \cdot 10^{-1} \pm 7 \cdot 10^{-2}$	$8 \cdot 10^{-1} \pm 3 \cdot 10^{-1}$
VAR(2)	Data	1.5	1.0	1.0	1.2
	Markov	$1.500 \pm 4 \cdot 10^{-3}$	$9.80 \cdot 10^{-1} \pm 3 \cdot 10^{-3}$	1.10 ± 10^{-2}	$1.20 \pm 5 \cdot 10^{-2}$
	RC	$8.9 \cdot 10^{-1} \pm 2 \cdot 10^{-2}$	$5.1 \cdot 10^{-1} \pm 1 \cdot 10^{-2}$	$7.2 \cdot 10^{-1} \pm 6 \cdot 10^{-2}$	$6 \cdot 10^{-1} \pm 2 \cdot 10^{-1}$
	Time	$2.80 \pm 4 \cdot 10^{-2}$	$1.8 \cdot 10^{-1} \pm 10^{-2}$	5.0 ± 10^{-1}	$3.00 \pm 9 \cdot 10^{-2}$
	SIGCW	$1.100 \pm 7 \cdot 10^{-3}$	$8.40 \cdot 10^{-1} \pm 6 \cdot 10^{-3}$	$1.00 \pm 2 \cdot 10^{-2}$	$8.7 \cdot 10^{-1} \pm 7 \cdot 10^{-2}$
	RCW	(Diverges)	(Diverges)	(Diverges)	(Diverges)
VAR(3)	Data	2.1	1.4	1.1	1.2
	Markov	$2.200 \pm 9 \cdot 10^{-3}$	$1.400 \pm 8 \cdot 10^{-3}$	$1.00 \pm 1 \cdot 10^{-2}$	$1.20 \pm 7 \cdot 10^{-2}$
	RC	1.80 ± 10^{-2}	$4.4o \cdot 10^{-1} \pm 9 \cdot 10^{-3}$	$-5 \cdot 10^{-1} \pm 5$	$5 \cdot 10^{-1} \pm 10^{-1}$
	Time	(Diverges)	(Diverges)	(Diverges)	(Diverges)
	SIGCW	1.00 ± 10^{-2}	$7.10 \cdot 10^{-1} \pm 8 \cdot 10^{-3}$	$1.10 \pm 3 \cdot 10^{-2}$	1 ± 10^{-1}
	RCW	(Diverges)	(Diverges)	(Diverges)	(Diverges)
Eye-Gaze	Data	3.2	6.2	5.8	$4.1 \cdot 10^{+1}$
	Markov	3.3 ± 10^{-1}	$6.2 \pm 3 \cdot 10^{-1}$	$5.8 \pm 2 \cdot 10^{-1}$	$4.1 \cdot 10^{+1} \pm 3$
	RC	$4.20 \cdot 10^{-1} \pm 5 \cdot 10^{-3}$	$2.90 \cdot 10^{-1} \pm 5 \cdot 10^{-3}$	$1.50 \pm 8 \cdot 10^{-2}$	$2.9 \pm 5 \cdot 10^{-1}$
	Time	$3.70 \cdot 10^{-1} \pm 4 \cdot 10^{-3}$	$2.20 \cdot 10^{-1} \pm 3 \cdot 10^{-3}$	$1.20 \pm 8 \cdot 10^{-2}$	$2.4 \pm 6 \cdot 10^{-1}$
	SIGCW	$1.200 \pm 7 \cdot 10^{-3}$	$7.70 \cdot 10^{-1} \pm 7 \cdot 10^{-3}$	$1.30 \pm 4 \cdot 10^{-2}$	$2.6 \pm 3 \cdot 10^{-1}$
	RCW	$4.50 \cdot 10^{-1} \pm 9 \cdot 10^{-3}$	$3.7 \cdot 10^{-1} \pm 10^{-2}$	$2.0 \pm 2 \cdot 10^{-1}$	6 ± 1

distributions as an indicative measure of the ability to capture time dependencies and make predictions about the future of the series. Finally, in Ref. [28], the authors check the probability density and auto-correlation functions, using the L^1-distance. In multidimensional data, this metric is also used to evaluate differences in feature correlation. The TSTR scheme is also used by the authors to evaluate performance.

In broad terms, one usually wants to check two aspects of the generated data: (i) if the generated distribution is similar to the original data and (ii) if the time-dependency and its possible long time-correlations reproduce those of real data. To estimate how well an algorithm replicates the distribution of the original data, we evaluate the distributions of the velocity magnitude $\|v\|$ and each angle, θ and φ (see Fig. 3).

In order to compare the similarity between distributions, we will use the Jensen–Shannon (JS) divergence, which is a symmetrised version of the Kullback–Leibler (KL) divergence, fulfilling the properties of a distance, in this case, between distributions. The KL divergence is defined by

$$D_{\mathrm{KL}}(\rho_{\mathrm{emp}}||\rho_{\mathrm{syn}}) = \int \rho_{\mathrm{emp}} \log\left(\frac{\rho_{\mathrm{emp}}}{\rho_{\mathrm{syn}}}\right). \tag{6}$$

With this definition, JS divergence is defined as

$$D_{\mathrm{JS}}(\rho_{\mathrm{emp}}||\rho_{\mathrm{syn}}) = \frac{1}{2}\left(D_{\mathrm{KL}}(\rho_{\mathrm{emp}}||\bar{\rho}) + D_{\mathrm{KL}}(\rho_{\mathrm{syn}}||\bar{\rho})\right), \tag{7}$$

where $\bar{\rho} = \frac{1}{2}(\rho_{\mathrm{emp}} + \rho_{\mathrm{syn}})$. The KL divergence is perhaps the most common measure to quantify the similarity between distributions, since minimising the KL divergence leads to a maximum likelihood estimation. The JS divergence remains this important feature and, additionally, has a symmetric property, i.e. $D_{\mathrm{JS}}(\rho_{\mathrm{emp}}||\rho_{\mathrm{syn}}) = D_{\mathrm{JS}}(\rho_{\mathrm{syn}}||\rho_{\mathrm{emp}})$, which in the present case is more intuitive: while the generator tries to approximate the synthetic distributions to the empirical ones, the discriminators try to distinguish the empirical distributions from those generated by the generator.

When it comes to quantifying how well the time-dependency is replicated, we evaluate the auto-correlation of the velocity magnitude $\|v\|$. The auto-correlation is defined as the Pearson correlation for each spatial coordinate, namely (for X)

$$\gamma_X(t_{\mathrm{lag}}) = \frac{\mathrm{E}\left[(X(t)-\overline{X})(X(t-t_{\mathrm{lag}})-\overline{X})\right]}{\mathrm{E}\left[(X(t)-\overline{X})(X(t)-\overline{X})\right]}, \tag{8}$$

where $\overline{X}$ is the average of the variable (spatial coordinates) and t_{lag} is the time-lag for which the auto-correlation is computed and $E[x]$ symbolises the expected value of a stochastic variable x. Notice that, while gaze trajectories are not stationary stochastic processes, both the AI models as well as the Markov model create time-homogeneous trajectories. In that sense, γ_X (and γ_Y separately) evaluates the model's ability to replicate the "average" dynamics of a given time-series.

To assess how well an algorithm replicates the auto-correlation function of each one of these three quantities we will consider the L^2-distance. The L^2-distance between two functions $f_{\mathrm{emp}}(x), f_{\mathrm{syn}}(x)$ is given by

$$d(f_{\mathrm{emp}}, f_{\mathrm{syn}}) = \left(\int_{-\infty}^{\infty} (f_{\mathrm{emp}}(x) - f_{\mathrm{syn}}(x))^2 dx\right)^{\frac{1}{2}}. \tag{9}$$

4. Comparative analysis

The results of this paper are shown in Fig. 4 – together with additional simulation in Appendix A, namely Figs. 6 and 7 – and in Fig. 5, with the results to reproduce eye-gaze trajectories. Table 2 shows the expected value and the uncertainty of the first four moments of the distribution of $|v|$-values, namely mean, variance, skewness and kurtosis, for each case of original data (synthetic, VAR-processes and empirical eye-gaze trajectories, and the corresponding Markov- and GAN-generated data. Table 3 shows the similarity between empirical and modelled data. For the distribution of each one of the three quantities characterising eye-gaze trajectories (cf. Fig. 3) we indicate the

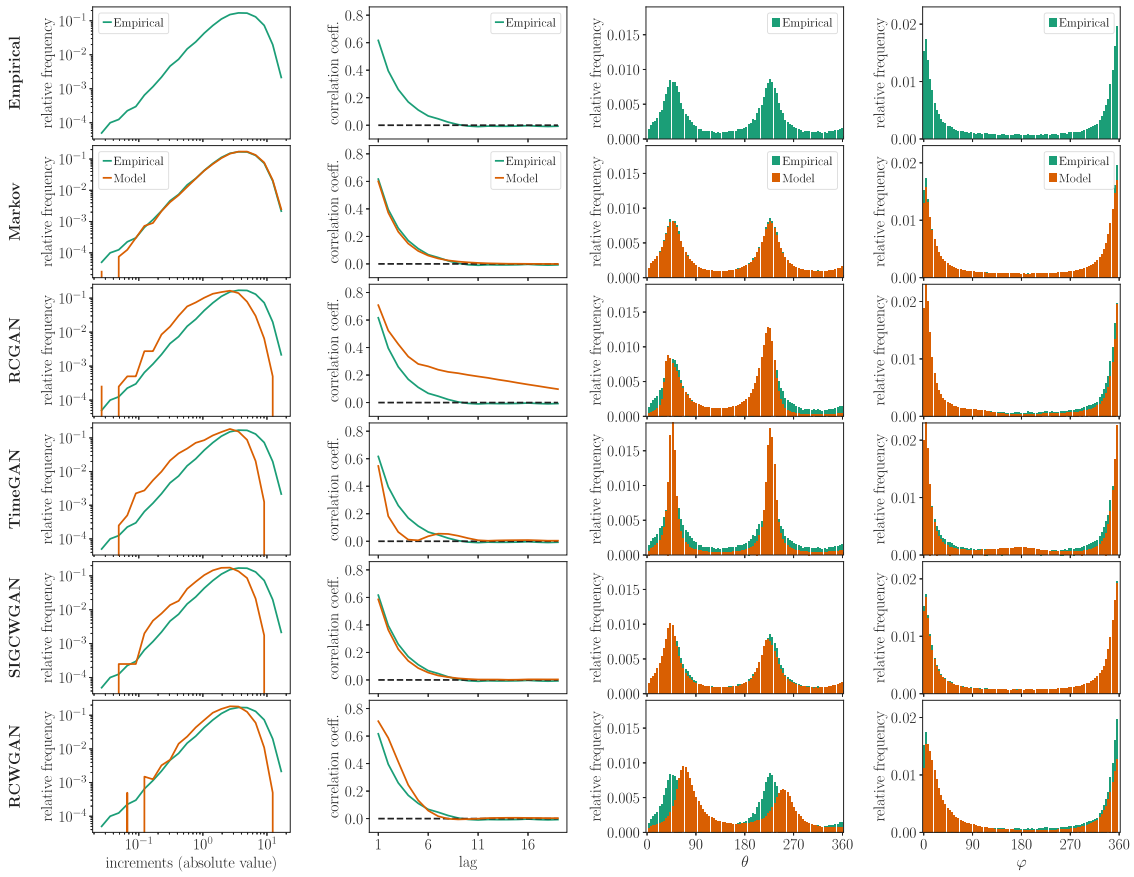

Fig. 4. Results for reproducing the VAR(1) process with a Markov model and time-series GANs. On the first and second rows, the distribution and auto-correlation function of $\|v\|$ are plotted. On the last two rows, the distributions of the angles θ and φ are shown.

Table 3
Algorithms' performance in replicating a VAR process (top) and gaze trajectories (bottom) according to the metrics defined in Eqs. (9) and (7). Firstly, the accuracy in replicating the distribution (first column) and auto-correlation (second column) function of $\|v\|$ are calculated. The distance between the empirical and synthetic distributions of the angles θ and φ are shown in the third and fourth columns respectively. It is possible to see that AI algorithms significantly fail in replicating any distribution of either $\|v\|$, θ or φ. In what the auto-correlation of $\|v\|$ are concerned, SigCWGAN comes close to a Markov process. With the distribution of θ and φ so poorly replicated, however, it is difficult to assume that it correctly captures the time-evolution of the process.

		Dist. $\|v\|$ $[D_{\text{JS}}]$	Aut.corr. $\|v\|$ $[d(f_{\text{emp}}, f_{\text{syn}})]$	Dist. θ $[D_{\text{JS}}]$	Dist. φ $[D_{\text{JS}}]$
VAR(1)	Markov	$\mathbf{6 \cdot 10^{-3}} \pm 2 \cdot 10^{-3}$	$\mathbf{7 \cdot 10^{-2}} \pm 1 \cdot 10^{-2}$	$\mathbf{6 \cdot 10^{-2}} \pm 10^{-2}$	$\mathbf{1.4 \cdot 10^{-1}} \pm 2 \cdot 10^{-2}$
	RCGAN	$3.4 \pm 4 \cdot 10^{-1}$	$7 \cdot 10^{-1} \pm 3 \cdot 10^{-1}$	$3.1 \pm 6 \cdot 10^{-1}$	1.2 ± 10^{-1}
	TimeGAN	2.8 ± 10^{-1}	$3.7 \cdot 10^{-1} \pm 4 \cdot 10^{-2}$	$4.7 \pm 3 \cdot 10^{-1}$	1.5 ± 10^{-1}
	SigCWGAN	2.5 ± 10^{-1}	$1.3 \cdot 10^{-1} \pm 4 \cdot 10^{-2}$	$6 \cdot 10^{-1} \pm 10^{-1}$	$3.2 \cdot 10^{-1} \pm 4 \cdot 10^{-2}$
	RCWGAN	1.0 ± 10^{-1}	$3.0 \cdot 10^{-1} \pm 3 \cdot 10^{-2}$	$6.8 \pm 3 \cdot 10^{-1}$	$2.1 \pm 2 \cdot 10^{-1}$
VAR(2)	Markov	$\mathbf{3.5 \cdot 10^{-3}} \pm 8 \cdot 10^{-4}$	$2.20 \cdot 10^{-1} \pm 6 \cdot 10^{-3}$	$\mathbf{2.8 \cdot 10^{-2}} \pm 4 \cdot 10^{-3}$	$\mathbf{4.5 \cdot 10^{-2}} \pm 5 \cdot 10^{-3}$
	RCGAN	$2.7 \pm 2 \cdot 10^{-1}$	1.6 ± 10^{-1}	$9.8 \pm 3 \cdot 10^{-1}$	$1.7 \pm 2 \cdot 10^{-1}$
	TimeGAN	$2.80 \pm 4 \cdot 10^{-2}$	$1.8 \cdot 10^{-1} \pm 10^{-2}$	$5.00 \pm 2 \cdot 10^{-2}$	$3.00 \pm 8 \cdot 10^{-2}$
	SigCWGAN	$9.3 \cdot 10^{-1} \pm 4 \cdot 10^{-2}$	$\mathbf{4 \cdot 10^{-2}} \pm 10^{-2}$	$10^{1} \pm 10^{-1}$	9.2 ± 10^{-1}
	RCWGAN	(Diverges)	(Diverges)	(Diverges)	(Diverges)
VAR(3)	Markov	$\mathbf{3.1 \cdot 10^{-3}} \pm 8 \cdot 10^{-4}$	$9.2 \cdot 10^{-1} \pm 10^{-2}$	$\mathbf{2.9 \cdot 10^{-2}} \pm 4 \cdot 10^{-3}$	$\mathbf{8.9 \cdot 10^{-2}} \pm 8 \cdot 10^{-3}$
	RCGAN	5.2 ± 10^{-1}	$1.30 \pm 7 \cdot 10^{-2}$	$4 \cdot 10^{1} \pm 1$	$1.20 \cdot 10^{1} \pm 3 \cdot 10^{-1}$
	TimeGAN	(Diverges)	(Diverges)	(Diverges)	(Diverges)
	SigCWGAN	4.2 ± 10^{-1}	$\mathbf{1.8 \cdot 10^{-1}} \pm 5 \cdot 10^{-2}$	$1.60 \cdot 10^{1} \pm 2 \cdot 10^{-1}$	$3.0 \pm 2 \cdot 10^{-1}$
	RCWGAN	(Diverges)	(Diverges)	(Diverges)	(Diverges)
Eye-Gaze	Markov	$\mathbf{1.6 \cdot 10^{-2}} \pm 3 \cdot 10^{-3}$	$\mathbf{3.2 \cdot 10^{-1}} \pm 7 \cdot 10^{-2}$	$\mathbf{3.3 \cdot 10^{-1}} \pm 2 \cdot 10^{-2}$	$\mathbf{4.6 \cdot 10^{-2}} \pm 6 \cdot 10^{-3}$
	RCGAN	$2.600 \pm 3 \cdot 10^{-3}$	$6.7 \cdot 10^{-1} \pm 10^{-2}$	1.1 ± 10^{-1}	$\mathbf{5 \cdot 10^{-2}} \pm 6 \cdot 10^{-2}$
	TimeGAN	$2.600 \pm 3 \cdot 10^{-3}$	$1.100 \pm 5 \cdot 10^{-3}$	1.2 ± 10^{-1}	$4.2 \cdot 10^{-1} \pm 5 \cdot 10^{-2}$
	SigCWGAN	$1.60 \pm 2 \cdot 10^{-2}$	$4.3 \cdot 10^{-1} \pm 2 \cdot 10^{-2}$	$9.3 \cdot 10^{-1} \pm 4 \cdot 10^{-2}$	$1.20 \cdot 10^{1} \pm 10^{-1}$
	RCWGAN	$2.6 \pm 4 \cdot 10^{-1}$	$8.90 \cdot 10^{-1} \pm 8 \cdot 10^{-3}$	$2.2 \pm 2 \cdot 10^{-1}$	$5.1 \pm 2 \cdot 10^{-1}$

numerical value of the Jensen–Shannon divergence between both distributions (cf. Eq. (7)), while for the auto-correlation function of the velocity magnitude we use the L^2-distance (cf. Eq. (9)). For each algorithm, 100 time-series were generated, each with 85 thousand data points, the same number of data points as the original time-series.

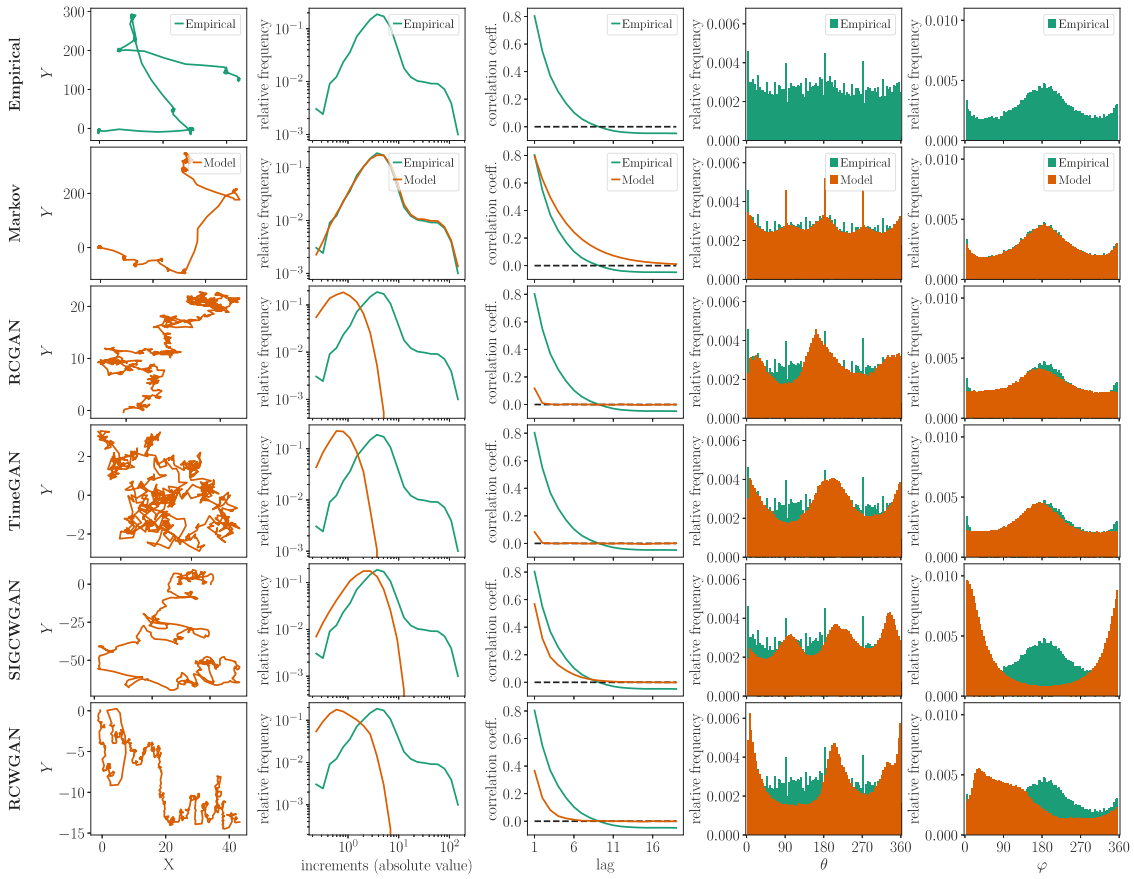

Fig. 5. Results for reproducing eye-gaze trajectories with a Markov model and time-series GANs. In the first column, the first 600 points of a trajectory are represented. In the second column and third columns, the distribution and auto-correlation function of $\|v\|$ are plotted. On the last two rows, the distributions of the angles θ and φ are shown.

4.1. Replicating synthetic data

Starting with the results for the VAR(1) process (Fig. 4), we observe that the distribution of (absolute value of) velocity increments has no heavy tails (first column). Even so, GAN models consistently fail to capture the large values of the distribution. They all perform at the same level, and, when compare with the Markov chain model, the performance is two to three orders of magnitude worse (cf. 3). Still, the RCWGAN architecture shows better results.

As for the auto-correlation functions (second column), we see also that again the Markov model performs better than any GAN architecture. Here, the SigCWGAN seems to retrieve the best results, followed by RCWGAN. Such a fact can be explained by the choice of the generator for these architectures, namely an AR-FNN, which is usually built with the stated aim to capture auto-correlations [28]. Still, contrary to what one would expect, even though these two GAN architectures are relatively good at estimating the (positively valued) auto-correlation function, they fail in simulating the large values of the velocity increment distribution. We see in Fig. 4 that the AI models tested here fail to correctly model the larger values of the VAR increments' distributions, even though their excess kurtosis is insignificant. This happens even when GANs can correctly replicate the auto-correlation function.

The distribution of the angle θ is also best grasped by the Markov model. Again, SIGCWGAN comes close but significantly worse than the Markov model, followed by the RCGAN architecture. RCWGAN shows significant overestimation bias, while in the case of TimeGAN, the distribution of θ is concentrated on 45° and 135°. For the distribution of the angle φ, all algorithms work relatively well. SigCWGAN and the Markov model work at the same level, with Markov performing better, but the results for the SigCWGAN lying within the margin of error. These results are in line with the ones published in Ref. [28] on the same data, with SigCWGAN performing slightly worse and the other GAN models performing slightly better.

Notice that VAR(1) processes are Markov processes by construction, so it is not surprising that a Markov process reproduces so well this process. However, similar results are obtained for VAR processes of higher dimension. In Appendix A we show the results for VAR(p) processes with $p = 2$ and $p = 3$, and there again one observes that Markov processes surpass all the set of GANs architectures. Moreover, as will be discussed in the next subsection, for the eye-gaze trajectories, the results do not differ much from this.

4.2. Replicating empirical data

For empirical data, examples of empirical eye-gaze trajectories, as well as the respective modelled trajectories with the different models, are shown in the first column of Fig. 5.

The distribution of the velocity increment (second column) shows again no heavy tails. However, while the Markov model's distribution fits again considerably well with the empirical distribution, the GAN model's inability to capture the extreme values is even stronger than for VAR processes. SigCWGAN works slightly better than the other architectures.

As for the auto-correlation function, since we do not expect gaze trajectories to obey the Markov condition, it is normal to

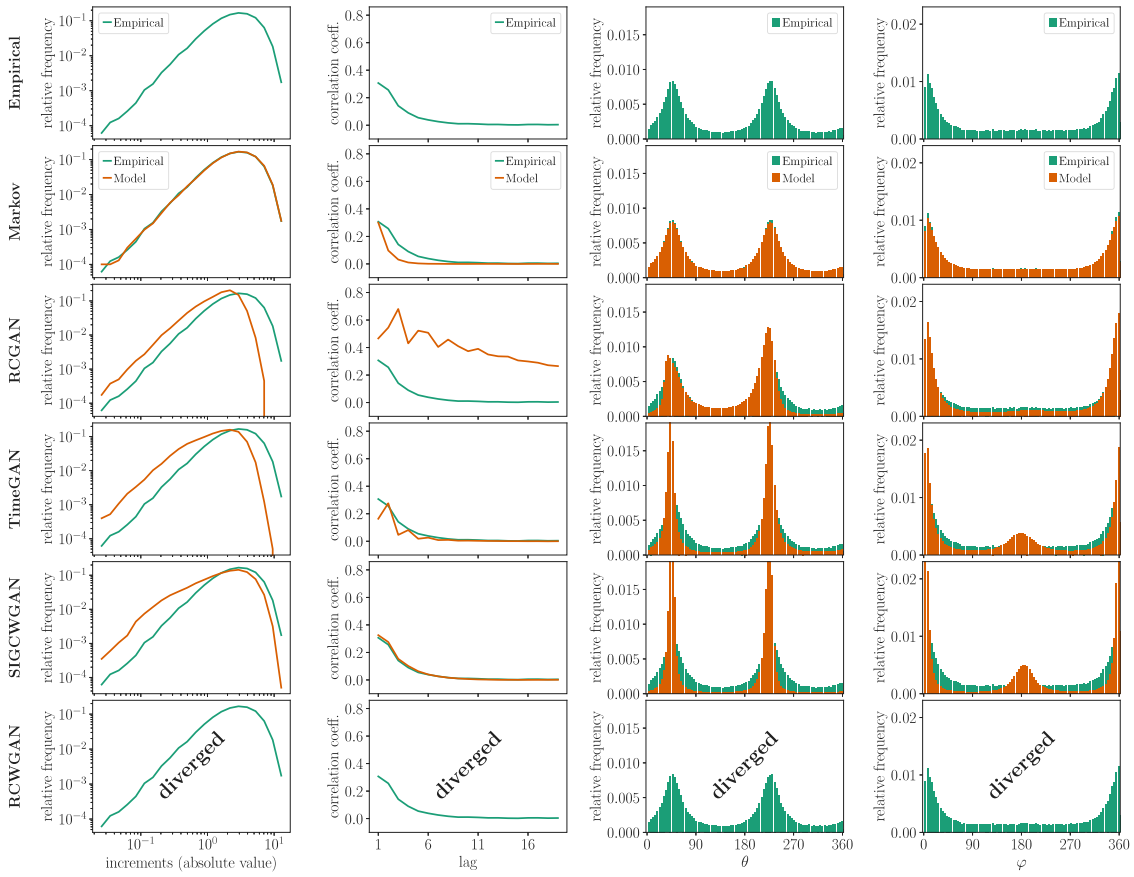

Fig. 6. Results for reproducing the **VAR(2)** process with a Markov model and time-series GANs. For comparison with Fig. 4.

expect a deviation between both functions. The SigCWGAN architecture comes close to the Markov model in modelling the auto-correlation function, while the other architectures are only able to partially capture the positive auto-correlations of the process.

The distribution of angle θ is also well grasped by the Markov model. In particular, the Markov model is able to capture the preferential directions around the 0°, 90°, 180°, 270°. It is however unable to capture the finer details of this metric, smoothing some of the other picks in the distribution. It is possible that larger samples (trajectories) or a different choice of the kernel when computing the numerical distribution would lead to a better resolution. GANs, however, cannot capture this distribution at all. They replicate some of the fluctuations, but with significant biases and miss the preferential angles. For the distribution of the angle φ, again, the Markov model works better than any GAN. However, TimeGAN and RCWGAN are able to capture some distribution around the 180° value. We see that even though SigCWGAN somewhat replicates the auto-correlation function, it does not capture the distribution of φ. Indeed, this time-series must be one rare example that has an angle between two increments around 180° but that, at the same time, has a positive auto-correlation function. It is possible that not being able to replicate the larger values of the distribution, in order to capture the positive value auto-correlation function, this algorithm tends to create trajectories in the same direction.

5. Discussion and conclusions

From the comparative analysis in the previous section, we conclude the Markov model outperforms all GAN architectures considered in this paper. Within the set of GAN architectures, it is possible to see that the SigCWGAN outperforms the rest of the GANS and that it performs at the level of a Markov implementation when it comes to the distribution of φ, but slightly worse for the distribution of θ. The other GANs perform relatively worse and do not come statistically close to the efficacy of a Markov process, particularly when it comes to the distribution of θ.

The fact that one specific GAN architecture can reproduce some of the statistics at the same level as a Markov process is good news for the AI community since it opens the door to their application to more complex processes. However, we provide evidence that simple Markov models are significantly better at modelling the distribution of the process and are themselves not as complex (black-box-like) as an AI method. Indeed, GAN implementations typically use a number of parameters two orders of magnitude larger than Markov models. Moreover, by simply computing a Markov transition matrix one might be able to assert other important features of the natural process (trajectory), e.g. if it is time-continuous or stationary [22].

Such drawbacks of GANs seem to be present in processes with a negligible excess kurtosis, such as VAR(p) processes, as well as 200 Hz free-viewing eye-gaze trajectory, which have typically a very large excess kurtosis. Thus, while we do not claim that non-parametric Markov models always outperform AI algorithms, we show that, when non-parametric models are needed, the implicit assumption of the universal preference of AI methods is not justified. This is done by showing that old-fashion mathematical models outperform their AI counterparts both in a very simple synthetic time-series and in a highly complex empirical one. We thus recommend caution when employing AI to predict time-series without comparing them with simpler methods that humans can easily explain.

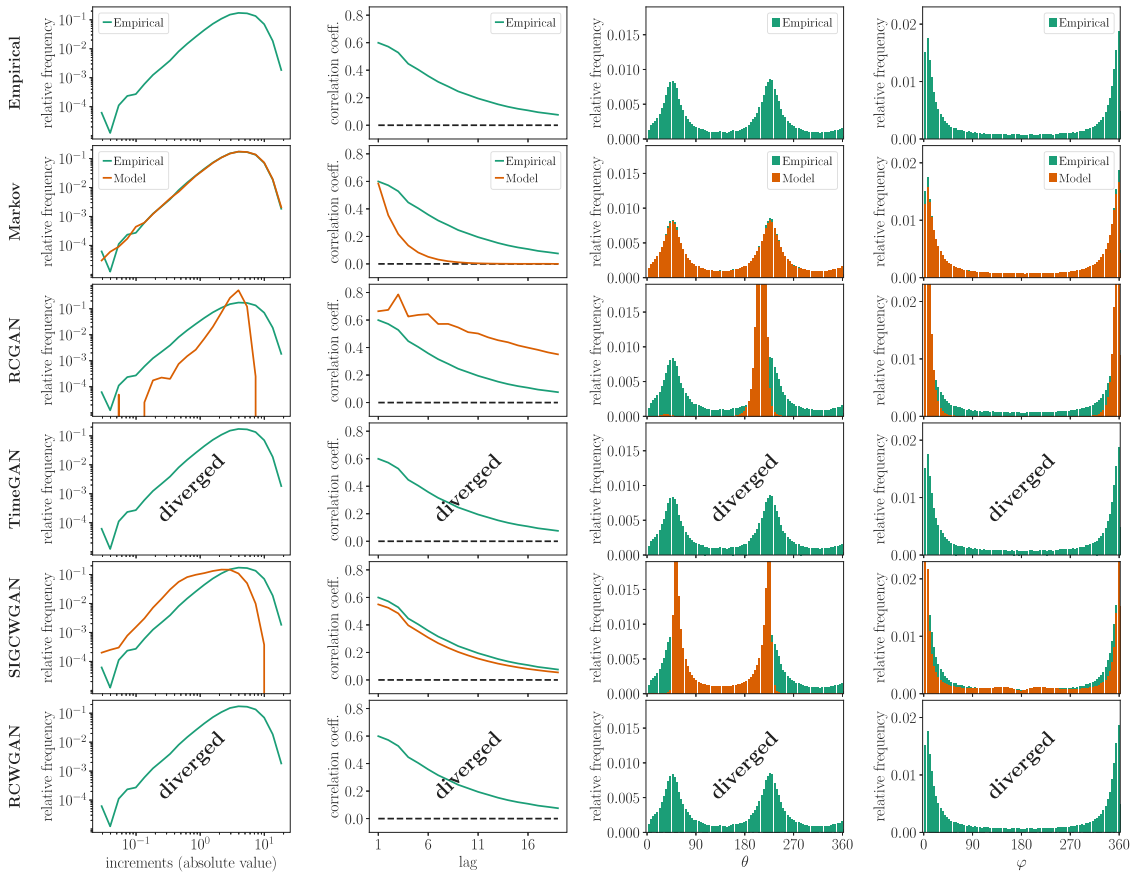

Fig. 7. Results for reproducing the **VAR(3)** process with a Markov model and time-series GANs. For comparison with Figs. 4 and 6.

The evidence uncovered in this paper can be now extended in different ways. On one hand, while the GAN models considered here are some of the most used in AI approaches and the processes consider cover both Gaussian and non-Gaussian features, other GAN architectures and datasets could be considered. In particular, emphasis should be given to architectures aimed at capturing the extreme values of a distribution, as has been the case of SigCWGAN and TimeGAN. Datasets covering other types of processes could also help to a systematic approach to exploring the limits of our findings, namely including jump-diffusion [35] noise and intermittent processes [36].

On the other hand, extensions of the Markov model used here could be investigated, namely models with a Markov length longer than one time lag. Finally, a hybrid approach could be to use a GAN to model the residuals of a Markov process or train GANs to generate Markov matrices (eventually in more dimensions).

We have compared our training with that of previous works on the same data [28] and, in most cases, found very similar training statistics and outcomes when training on the same data.

The GANs tested here are conditional GANs, meaning that they use the previous points to calculate the following points. Thus, the number of points used in calculating the next set of points (typically represented a "p" in the literature). A systematic search of the optimal value of this parameter was performed and the value of $p = 3$ was chosen. Other aspects, such as the size of the network and the batch size were also optimised after comparing the outcomes of several trials.

All in all, despite the claims of GAN's ability to capture the overall time-evolution of a stochastic process this is not verified in complex time-series. Even in algorithms which, according to the authors, were able to capture extreme values of the data, this limitation is quite significant, with the GAN architectures not being able to capture the full spectrum of the auto-correlation function. At face value, the inability of AI methods to outperform a conceptually simple mathematical process is surprising. When these methods are presented, their successes are highlighted and further applications of these methods focus only on the case where they are successful. Nonetheless, they are constrained by the central limit theorem: the typical distribution of the GAN-generated data is the normal distribution. One possibility to overcome this would be to re-design the standard GAN architectures, enabling them to include noise inputs distributed according to α-stable distributions.

Declaration of competing interest

The authors declare that there are no interest to be stated.

Data availability

The authors do not have permission to share data.

Acknowledgements

The authors thank Oslo Metropolitan University for partial financial support, through the Nordic Center for Sustainable and Trustworthy AI Research (NordSTAR).

Appendix A. Results for replicating VAR(2) and VAR(3) processes

While gaze trajectories have a non-Markov behaviour it is also interesting to apply our methodology to synthetic processes that,

contrarily to the VAR(1) process, are also, by definition, non-Markov. These are the VAR(p = 2) processes and the VAR(p = 3) processes as defined in Eq. (4).

In the VAR(2) process we have used $\phi_1 = 0.5$ and $\phi_2 = 0.4$ and in the VAR(3) process we have used $\phi_1 = 0.3$, $\phi_2 = 0.3$ and $\phi_3 = 0.3$. The correlation coefficient the two spacial dimensions was kept at 0.8 and, just like before, for each case we have created 85 thousand points time-series with a sampling time $\Delta t = 1$. Results can be observer in Table 3 and in Figs. 6 and 7.

In this case, we observed that some GAN-generated time-series diverged to infinity. This was the case for the RCWGAN in the VAR(2) case and for the RCWGAN and TimeGAN in the VAR(3) process. We found a similar behaviour with other choices of ϕ with RCGAN diverging at times too. SigCWGAN was not observed to diverge for convergent time-series. Changing the number of epochs in training or the learning rate did not seem to significantly affect some GANs, producing divergent series. Indeed, we observe that SigCWGAN typically replicates the auto-correlation function of the VAR(2) and VAR(3) at an accuracy similar to the case of VAR(1). As expected, this is not the case for a Markov process.

However, given that the ANN that is used in the SigCWGAN (the AR-FNN) is explicitly built with the purpose of mimicking the auto-correlation function of the data and that this architecture fails to capture the distribution of φ, it is reasonable to assume that the algorithm is unable to replicate many significant aspects of the temporal dynamics of the process.

When it comes to the distribution of the increments of a process, we see that, similarly to VAR(1) processes and gaze trajectories, GANs underestimate its moments, specially in the standard deviation. We also see that GANs fail in reproducing the distribution of θ showing that they do not accurately represent the relationship between the two spacial dimensions.

In conclusion, for the case of VAR(2) and VAR(3) we see that one aspect of the temporal dynamics of the process (the auto-correlation function) is better replicated by one GAN, the SigCWGAN, while the other (the distribution of φ) is still far from the results produced by the Markov model. Moreover, GANs still fail to replicate the distribution of the increments and to capture correctly the relationship between the two spacial dimensions of the problem, which can be evaluated by the distribution of θ. Thus, even in the case of a simple synthetic data where the Markov hypothesis is not present, the advantages of GANs are still limited.

Appendix B. About the training of GANs

Training GANs is a challenging task since this is often not a stable process, where each epoch is better than the previous one, and where some pitfalls exist [8]. Two problems are usual, namely the *vanishing gradients* and the *mode collapse*. In the first one, the discriminator becomes so successful that neither the discriminator nor the generator is able to have any learning with successive epochs. In *mode collapse* the generator is able to fool the discriminator with just small different modes ignoring all the others, thus producing time-series that are all very similar among themselves. To solve this, one is typically advised to choose carefully the learning rate of both networks.

In our implementation, we have indeed found that training a GAN for a longer period did not necessarily lead to better results. Moreover, we experienced some instances of mode collapse when trying to model real-time series and had to carefully calibrate learning rates. We found that training the GANs for around 200 epochs would decrease the train and testing error while also avoiding the pitfall of mode collapse. The weights of the ANNs were updated using the Adam optimiser [37].

Appendix C. Implementation of the Markov-chain model

In a Markov model, we can generate a time-series by computing $\Pr(X_{t=n+1} = x_{n+1} \mid X_{t=n} = x_n)$. We estimate this quantity empirically with the help of a Gaussian estimation kernel K as follows:

$$\Pr(X_{t=n+1} = x_{n+1} \mid X_{t=n} = x_n) = \frac{\Pr(X_{t=n+1} = x_{n+1}, X_{t=n} = x_n)}{\Pr(X_{t=n} = x_n)}, \tag{C.1}$$

with

$$\Pr(X_{t=n+1} = x_{n+1}, X_{t=n} = x_n) = \frac{1}{(\hat{N}-1)^2 h^2} \sum_{i=1}^{\hat{N}-1} K\left(\frac{x_{n+1} - \hat{x}_{i+1}}{h}\right) K\left(\frac{x_n - \hat{x}_i}{h}\right), \tag{C.2}$$

$$\Pr(X_{t=n} = x_n) = \frac{1}{h(\hat{N}-1)} \sum_{i=1}^{\hat{N}-1} K\left(\frac{x_n - \hat{x}_i}{h}\right), \tag{C.3}$$

where

$$K\left(\frac{x_n - \hat{x}_i}{h}\right) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x_n - \hat{x}_i}{h}\right)^2\right), \tag{C.4}$$

and h represents the bandwidth of the Gaussian estimation kernel K and it is calculated following Silverman's rule [38]:

$$h = \left(\frac{4\hat{\sigma}^5}{3\hat{N}-3}\right)^{\frac{1}{5}} \approx 1.06\,\hat{\sigma}\,(\hat{N}-1)^{-1/5}, \tag{C.5}$$

where $\hat{\sigma}$ is the standard deviation of $\hat{x}_1 \ldots \hat{x}_n$ and $\hat{N}$ the number of data points in our sample.

When analysing empirical data, $\Pr(X_{t=n+1} \mid X_{t=n})$ can be represented as a matrix $\mathbf{T}$ of dimension $N_s \times N_s$ with entries given by

$$T_{i,j} = \Pr\Big(X_{t=n+1} \subset [k_i, k_{i+1}) \mid X_{t=n} \subset [k_j, k_{j+1})\Big), \tag{C.6}$$

with $i, j \in \mathbb{N}$, $i, j \subset [0, N_s]$ and $k_m > k_n \Leftrightarrow m > n$. Thus, if we observer the state $X_{t=n} \subset [k_j, k_{j+1})$ we can calculate the probability of observing $X_{t=n+1} \subset [k_i, k_{i+1})$. When generating a new time-series, if it is found that $X_{t=n+1} \subset [k_i, k_{i+1})$, we assign a value to $X_{t=n+1}$ from the uniform distribution in the interval $[k_i, k_{i+1})$.

The accuracy of this method depends on three major factors: Firstly, on the validity of the Markov condition (3); Secondly, on the number of states N_s, which is constrained by computational time (scaling approximately with N_s^4 for a two-dimensional process); Thirdly, accuracy is also affected by the amount of data in our sample $\hat{N}$, which impacts the bandwidth calculus h: the larger $\hat{N}$ is, the smaller is the resulting value of h thus increasing the spatial resolution of the model. An implementation of this algorithm using python and numpy can be found in Github [39].

References

[1] A. Flores, H. Tito-Chura, V. Yana-Mamani, Wind speed time series prediction with deep learning and data augmentation, in: Proceedings of SAI Intelligent Systems Conference, Springer, 2021, pp. 330–343, http://dx.doi.org/10.1007/978-3-030-82193-7_22.

[2] P.G. Lind, L. Vera-Tudela, M. Wächter, M. Kühn, J. Peinke, Normal behaviour models for wind turbine vibrations: Comparison of neural networks and a stochastic approach, Energies 10 (12) (2017) 1944, http://dx.doi.org/10.3390/en10121944.

[3] A. Russo, F. Raischel, P. Lind, Air quality prediction using optimal neural networks with stochastic variables, Atmos. Environ. 79 (2013) 822–830, http://dx.doi.org/10.1016/j.atmosenv.2013.07.072.

[4] J. Cao, Z. Li, J. Li, Financial time series forecasting model based on CEEMDAN and LSTM, Phys. A 519 (2019) 127–139, http://dx.doi.org/10.1016/j.physa.2018.11.061.
[5] P. Wang, X. Zheng, G. Ai, D. Liu, B. Zhu, Time series prediction for the epidemic trends of COVID-19 using the improved LSTM deep learning method: Case studies in Russia, Peru and Iran, Chaos Solitons Fractals 140 (2020) 110214, http://dx.doi.org/10.1016/j.chaos.2020.110214.
[6] F. Harrou, A. Dairi, F. Kadri, Y. Sun, Forecasting emergency department overcrowding: A deep learning framework, Chaos Solitons Fractals 139 (2020) 110247, http://dx.doi.org/10.1016/j.chaos.2020.110247.
[7] I.J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial networks, 2014, http://dx.doi.org/10.48550/arXiv.1406.2661, arXiv preprint arXiv:1406.2661.
[8] E. Brophy, Z. Wang, Q. She, T. Ward, Generative adversarial networks in time series: A survey and taxonomy, 2021, http://dx.doi.org/10.48550/arXiv.2107.11098, arXiv preprint arXiv:2107.11098.
[9] D. Hazra, Y.-C. Byun, SynSigGAN: Generative adversarial networks for synthetic biomedical signal generation, Biology 9 (12) (2020) 441, http://dx.doi.org/10.3390/biology9120441.
[10] L. Yu, W. Zhang, J. Wang, Y. Yu, Seqgan: Sequence generative adversarial nets with policy gradient, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 31, 2017, p. 1, URL https://www.aaai.org/Conferences/AAAI/2017/PreliminaryPapers/12-Yu-L-14344.pdf.
[11] O. Mogren, C-RNN-GAN: Continuous recurrent neural networks with adversarial training, 2016, http://dx.doi.org/10.48550/arXiv.1611.09904, arXiv preprint arXiv:1611.09904.
[12] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, Y.-H. Yang, Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018, p. 1, URL https://salu133445.github.io/musegan/pdf/musegan-aaai2018-paper.pdf.
[13] Z. Lv, X. Huang, W. Cao, An improved GAN with transformers for pedestrian trajectory prediction models, Int. J. Intell. Syst. 37 (8) (2021) 4417–4436, http://dx.doi.org/10.1002/int.22724.
[14] M. Wiese, R. Knobloch, R. Korn, P. Kretschmer, Quant GANs: Deep generation of financial time series, Quant. Finance 20 (9) (2020) 1419–1440, http://dx.doi.org/10.1080/14697688.2020.1730426.
[15] S. Berkovsky, R. Taib, I. Koprinska, E. Wang, Y. Zeng, J. Li, S. Kleitman, Detecting personality traits using eye-tracking data, in: Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, 2019, pp. 1–12, http://dx.doi.org/10.1145/3290605.3300451.
[16] M. Steffens, B. Becker, C. Neumann, A. Kasparbauer, I. Meyhöfer, B. Weber, M. Mehta, R. Hurlemann, U. Ettinger, Effects of ketamine on brain function during smooth pursuit eye movements, Hum. Brain Mapp. 37 (11) (2016) 4047–4060, http://dx.doi.org/10.1002/hbm.23294.
[17] P.M. Grace, T. Stanford, M. Gentgall, P.E. Rolan, Utility of saccadic eye movement analysis as an objective biomarker to detect the sedative interaction between opioids and sleep deprivation in opioid-naive and opioid-tolerant populations, J. Psychopharmacol. 24 (11) (2010) 1631–1640, http://dx.doi.org/10.1177/0269881109352704.
[18] H. Chauhan, A. Prasad, J. Shukla, Engagement analysis of ADHD students using visual cues from eye tracker, in: Companion Publication of the 2020 International Conference on Multimodal Interaction, 2020, pp. 27–31, http://dx.doi.org/10.1145/3395035.3425256.
[19] A. Lev, Y. Braw, T. Elbaum, M. Wagner, Y. Rassovsky, Eye tracking during a continuous performance test: Utility for assessing ADHD patients, J. Atten. Disord. 26 (2) (2022) 245–255, http://dx.doi.org/10.1177/1087054720972786.
[20] T. Wadhera, D. Kakkar, Eye tracker: An assistive tool in diagnosis of autism spectrum disorder, in: Emerging Trends in the Diagnosis and Intervention of Neurodevelopmental Disorders, IGI Global, 2019, pp. 125–152, http://dx.doi.org/10.4018/978-1-5225-7004-2.ch007.
[21] B. Hayes, et al., First links in the Markov chain, Am. Sci. 101 (2) (2013) 252, http://dx.doi.org/10.1511/2013.101.92.
[22] P. Lencastre, F. Raischel, T. Rogers, P. Lind, From empirical data to time-inhomogeneous continuous Markov processes, Phys. Rev. E 93 (3) (2016) 032135, http://dx.doi.org/10.1103/PhysRevE.93.032135.
[23] R. Huang, B. Xu, D. Schuurmans, C. Szepesvári, Learning with a strong adversary, 2015, http://dx.doi.org/10.48550/arXiv.1511.03034.
[24] S.L. Hyland, C. Esteban, G. Rätsch, Real-valued (medical) time series generation with recurrent conditional GANs, 2017, http://dx.doi.org/10.48550/arXiv.1706.02633, arXiv preprint arXiv:1706.02633.
[25] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780, http://dx.doi.org/10.1162/neco.1997.9.8.1735.
[26] J. Yoon, D. Jarrett, M. Van der Schaar, Time-series generative adversarial networks, Adv. Neural Inf. Process. Syst. 32 (2019) URL http://papers.neurips.cc/paper/8789-time-series-generative-adversarial-networks.pdf.
[27] D.P. Kingma, M. Welling, et al., An introduction to variational autoencoders, Found. Trends Mach. Learn. 12 (4) (2019) 307–392, http://dx.doi.org/10.1561/2200000056.
[28] H. Ni, L. Szpruch, M. Wiese, S. Liao, B. Xiao, Conditional sig-Wasserstein GANs for time series generation, 2020, http://dx.doi.org/10.48550/arXiv.2006.05421, arXiv preprint arXiv:2006.05421.
[29] H. Ni, L. Szpruch, M. Wiese, S. Liao, B. Xiao, Conditional-Sig-Wasserstein-GANs, 2020, URL https://github.com/SigCGANs/Conditional-Sig-Wasserstein-GANs.
[30] A.A. Markov, Extension of the law of large numbers to dependent quantities, Izv. Fiz.-Matem. Obsch. Kazan Univ. (2nd Ser.) 15 (1) (1906) 135–156.
[31] R.M. Klein, W.J. MacInnes, Inhibition of return is a foraging facilitator in visual search, Psychol. Sci. 10 (4) (1999) 346–352, http://dx.doi.org/10.1111/1467-9280.00166.
[32] A. Borji, Pros and cons of GAN evaluation measures, Comput. Vis. Image Underst. 179 (2019) 41–65, http://dx.doi.org/10.1016/j.cviu.2018.10.009.
[33] A. Borji, Pros and cons of GAN evaluation measures: New developments, Comput. Vis. Image Underst. 215 (2022) 103329, http://dx.doi.org/10.1016/j.cviu.2021.103329.
[34] K. Shmelkov, C. Schmid, K. Alahari, How good is my GAN? in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 213–229, http://dx.doi.org/10.1007/978-3-030-01216-8_14.
[35] L. Rydin Gorjão, D. Witthaut, P.G. Lind, JumpDiff: Non-parametric numerical estimation of jump-diffusion processes, J. Stat. Softw. 15 (2023) 1–22, http://dx.doi.org/10.18637/jss.v105.i04.
[36] P. Lencastre, S. Denysov, A. Yazidi, P.G. Lind, Uncovering Lévy flights and intermittent processes from empirical time series: a theoretical framework to classify, 2023, in prepartion.
[37] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, http://dx.doi.org/10.48550/arXiv.1412.6980, arXiv preprint arXiv:1412.6980.
[38] B.W. Silverman, Density Estimation for Statistics and Data Analysis, first ed., Routledge, New York, 1998, http://dx.doi.org/10.1201/9781315140919.
[39] P. Lencastre, Markov model, 2023, URL https://github.com/134f/Physica-D-Markov-model.