Journal of King Saud University – Computer and Information Sciences 34 (2022) 7920–7939

Contents lists available at ScienceDirect

Journal of King Saud University – Computer and Information Sciences

journal homepage: www.sciencedirect.com

Neural networks generative models for time series

Federico Gatta [a], Fabio Giampaolo [a], Edoardo Prezioso [a], Gang Mei [b], Salvatore Cuomo [a], Francesco Piccialli [a,*]

[a] Department of Mathematics and Applications 'R. Caccioppoli', University of Naples Federico II, Italy
[b] School of Engineering and Technology, China University of Geosciences, Beijing, China

ARTICLE INFO

Article history:
Received 1 June 2022
Revised 5 July 2022
Accepted 13 July 2022
Available online 16 July 2022

Keywords:
Time series
Generative adversarial networks
Healthcare
Industry 4.0
Deep learning

ABSTRACT

Nowadays, time series are a widely-exploited methodology to describe phenomena belonging to different fields. In fact, electrical consumption can be explained, from a data analysis perspective, with a time series, as for healthcare, financial index, air pollution or parking occupancy rate. Applying time series to different areas of interest has contributed to the exponential rise in interest by both practitioners and academics. On the other side, especially regarding static data, a new trend is acquiring even more relevance in the data analysis community, namely neural network generative approaches. Generative approaches aim to generate new, fake samples given a dataset of real data by implicitly learning the probability distribution underlining data. In this way, several tasks can be addressed, such as data augmentation, class imbalance, anomaly detection or privacy. However, even if this topic is relatively well-established in the literature related to static data regarding time series, the debate is still open. This paper contributes to this debate by comparing four neural network-based generative approaches for time series belonging to the state-of-the-art methodologies in literature. The comparison has been carried out on five public and private datasets and on different time granularities, with a total number of 13 experimental scenario. Our work aims to provide a wide overview of the performances of the compared methodologies when working in different conditions like seasonality, strong autoregressive components and long or short sequences.

Contents

* Corresponding author at: Department of Mathematics and Applications 'R. Caccioppoli', University of Naples Federico II, via Cinthia, 80126, Italy.
E-mail address: francesco.piccialli@unina.it (F. Piccialli).

Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

https://doi.org/10.1016/j.jksuci.2022.07.010
1319-1578/

1. Introduction

One of the most relevant research topics in Artificial Intelligence (AI) is time series processing. A time series is a collection of observations constantly repeated through time. In particular, we focus on multivariate time series, which can be thought of as sequences of vectors whose components are related. Thanks to their capability to describe variables' temporal evolution, time series are currently used to deal with a variety of phenomena belonging to different domains, such as Finance (Arroyo et al., 2011), Management (Machiwal and Jha, 2006) and Healthcare (Muhammad et al., 2021), so they attract a considerable number of both academics and practitioners.

On the other hand, one of the most recent AI tasks gaining even more interest is the generation of fake samples. In more detail, some researchers are concerned about developing generative models that can create new, fake samples starting from a dataset of real observations. In particular, fake sample generation can be useful for various purposes, such as data augmentation, anomaly detection or privacy issues. Consequently, a quite large literature has been developed in the field of static data, and several works have been proposed to summarize and compare generative approaches for static data, see for example (Oussidi and Elhassouny, 2018; Harshvardhan et al., 2020).

Motivation In contrast to static data, research on generative models for time series is still in an embryonic stage and, although different methods have been successfully proposed to achieve different tasks, such as Che et al. (2018), Shen et al. (2018) and Debnath et al. (2021), there are just a few works which attempt to summarize, compare and give a research direction for time series generative models. In fact, when dealing with time series, an additional issue is to properly model the variables' temporal evolution under observation. Furthermore, another big deal is the evaluation of the fake samples. That is, while for static data, in particular for images, it is roughly simpler to evaluate the quality of generated samples simply by visual inspection, with time series, this task is sometimes impossible (see, for example, the example of time series we work within Section 4). Furthermore, a good metric should also consider the time factor, so it is not correct to exploit the same evaluation criteria as for static data.

Deep Learning Generative Models for Time Series In this work, we focus on deep learning generative models for time series. Actually, existing approaches for time series generation can be divided into two families, namely statistical methods and deep learning methods, as pointed out in Iwana and Uchida (2021). The methods in the former class explicitly model the probability distribution of time series, i.e. their dynamic. So, usually, these methods make strong prior assumptions about the conditional distribution of the time series. Thus, they are less flexible than neural network-based models. In fact, these models approximate the probability distribution, usually without making any prior assumption on its form. Furthermore, thanks to their property to be universal approximators, if properly trained neural networks should be able to replicate any kind of distribution, as stated in Lu and Lu (2020).

Task In this perspective, this paper focuses on the generation of time series. That is, given a dataset made up of real-time series $\mathcal{X} = \{X_j\}_{j \in \mathcal{J}}$, we want to build a dataset made up of fake time series $\mathcal{F} = \{F_j\}_{j \in \mathcal{J}}$. Hereby, we exploit neural network approaches which aim to approximate the original probability distribution p_{data} with a new, replicable distribution p_{fake}, learned from a well-known distribution p_z. In particular, several neural network-based approaches for time series generation have been compared in various experiments. As models, we exploit some methods belonging to the widely used class of *Generative Adversarial Networks* (GAN), namely the *Recurrent Conditional GAN* (RCGAN), the *Time GAN* (TimeGAN) and the *Conditional Sig-Wasserstein GAN* (SigCWGAN). Furthermore, also the *Generative Moment Matching Networks* (GMMN) is used as an example of the non-GAN approach. These techniques have been chosen among those with the highest impact on the research and whose Python code is publicly available online. All of them are neural network-based. In particular, the most commonly used architecture is the *Recurrent Neural Network* (RNN) and its main variation, the *Long Short-Term Memory* (LSTM). Moreover, also the *AutoRegressive-Feedforward Neural Network* (AR-FNN) is quite extensively used. As for the dataset exploited for the comparison, we use 3 public datasets and 2 private, as described in Section 4. The datasets we work with are concerned with traffic flow, air pollutants, electricity consumption, financial index and industrial production. They cover wide records: time series exhibit seasonality, time series with a strong autoregressive part or without manifest patterns. Furthermore, several experiments on different time horizons and granularities are carried out for each dataset, with a total number of 13 experiments.

Main Contribution Our paper addresses some open questions which have been highlighted in previous works related to this topic, see for example (Wen et al., 2020; Brophy et al., 2021). In particular:

- we provide an experimental comparison of neural network generative models for time series without being constrained to a specific task;
- we discuss the benefit of using an ensemble of methodologies for the generation of the fake samples;
- we discuss and employ a huge number of metrics to evaluate task-free generative models for time series;
- we analyze the behaviour of the models when working on the same dataset but with different time horizons and granularities.

Paper Organisation The following of this paper is organized in this way. Section 2 discusses related works which help to contextualize our contribution to the literature. Section 3 briefly describes the generative models exploited for the comparison. Section 4 presents the experimental setup, the dataset and the metric used. Section 5 exhibits and discusses the results obtained. Section 6 contains a comment about the current and future state-of-the-art of deep generative models for time series. Finally, Section 7 concludes this work.

2. Related work

In this section, we review some of the most important works in the field of neural network generative models to contextualize our work in the existing literature and justify the need for our paper. In particular, the Section is composed of three parts: in the first one, we review some application papers which show the potentiality of generative approaches in the solution of data analysis problems; the second part is concerned with other surveys and comparative papers which are similar to ours; in the third part, we state the novelty of our work, and we explicitly highlight how it overcomes the limitations of the other similar papers.

2.1. Generative approaches in practical application

This work proposes a comparison of generative approaches for time series. This type of model is gaining even more interest due to its wide application range for data analysis problems. In our opinion, the most relevant tasks are data augmentation, anomaly detection and privacy.

Among the methodologies and frameworks developed by the researcher community, it is noteworthy to cite (Srinivasan and Knottenbelt, 2022). They introduce a new kind of time series-based GAN, namely the Time-series Transformer GAN, which combine adversarial training with the cutting-edge idea of transformer networks. In this way, the authors claim to be able to overcome most of the existing GAN-based approaches. Experiments on 5 publicly available datasets, both real and simulated, have been carried out. However, no code has been released for this model. Instead, the proposal of Sumiya et al. (2019) is more focused on noise reduction for medical signals. In particular, the authors propose the Noise Reduction GAN, a CNN GAN with a penalization term in the Generator loss which forces the generated sample to be closer to the original data than the standard GAN. Another work focused on medical signals is Hazra and Byun (2020). In this case, the Synthetic biomedical Signals GAN is proposed. It combines CNN in the Discriminator and RNN in the Generator. Another interesting work is Jeha et al. (2021) which proposes PSA-GAN, a GAN-based approach that exploits self-attention mechanisms to identify temporal patterns in the series. Furthermore, PSA-GAN claims to be particularly efficient in modeling long time series thanks to progressive training which allows to approximate time series at different granularities. Sun et al. (2020) propose the Decision-Aware Time-series Conditional GAN by modifying the Multi-Wasserstein loss to take into account decision-awareness. The framework is specifically designed for financial decisions. In Wang et al. (2019), the Sequentially Coupled GAN architecture is proposed. Specifically designed for medical applications, it consists of two generators and one discriminator. The two discriminators are sequential, that is the output of the first one is used as input by the second one and aims to generate both the state of a fake patient and the suggested treatment. Li et al. (2021) propose the Time Series Augmentation GAN to perform data augmentation in the context of time series classification. Their proposal attempts to solve the saturation problem which can affect the Generator with a self-adaptive recovering strategy. In this way, a noticeable improvement in accuracy is achieved. Instead, Liu et al. (2021) faces up with the problem of time series missing values imputation. Their proposal, namely GlowImp, combines both Variational AutoEncoder (VAE) and Wasserstein GAN to alleviate the computational cost of the procedure while providing valid imputation on public datasets.

Regarding data augmentation, interesting work is Tu et al. (2018), where an LSTM-based AutoEncoder (AE) is exploited to augment time series related to skeleton data in the context of the smart house. Different deep learning classifiers are trained with and without augmented data, and the results show the contribution given by the proposed methodology in improving the overall accuracy. Moreover, as for the GAN, several works have been published, showing a rising interest in literature on this topic. In particular, several application domains have been investigated, such as business (Deng et al., 2022), geophysics (Wang et al., 2021), medical diagnosis (Kiyasseh et al., 2020; Altaheri et al., 2021) and industry 4.0 (Li et al., 2021). Finally, it is noteworthy to cite (Shiota et al., 2018), which exploits GMMN to perform data augmentation on voice datasets.

As for the anomaly detection, we report (Zhu et al., 2019) where both convolutional and LSTM layers are employed in a GAN-style architecture to study the occurrence of anomalous values in ECG and New York taxi dataset. The experimental stage confirms that the proposed approach is able to provide more accurate precision than standard methods. Instead, Li et al. (2018) exploits a GAN architecture with the recurrent network to address the anomaly detection task in a multivariate setting. In particular, the proposal's effectiveness is proved by a careful experimental stage on a dataset related to cyber-attacks on a water treatment system. Regarding other neural network approaches to this task, an example is provided by Bovenzi et al. (2022), which proposes an AE based model for anomaly detection in the context of cyber-security. In this view, a new sample is considered malign if its reconstruction error exceeds a certain threshold.

Regarding privacy applications, there is still a scarcity of work. Maybe, this can be one of the most important analysis directions for future works. However, we highlight (Xu et al., 2021), which exploits a neural network-based approach to generate a synthetic network traffic dataset which, accordingly to the results of the paper, can be efficiently used to train cyber-security classification algorithms while preserving the privacy of users. Furthermore, addresses privacy issues due to current regulations by exploiting a neural network-based approach which combines GAN with Transformer architecture, namely the *Time-series Transformer Generative Adversarial Network*, to accurately mimic the conditional distribution of the time series. Their methodology is then compared with other state-of-the-art methodologies to assess the worthiness of the proposal. Finally, Xie et al. (2018) proposes a modified GAN to address differential

privacy by adding noise to the gradient in the back-propagation procedure.

2.2. Survey and comparative works

Generative approaches related to static and time series data have been investigated for years. In particular, several notable works have been proposed in the last years to analyze and compare different approaches. For example, Brophy et al. (2021) provides a survey about GAN approaches for time series. Although this survey is comprehensive of almost all state-of-the-art approaches and it deeply discusses the possible application of generative models, it does not provide a practical comparison between the different presented methodologies. Actually, in its conclusion, it emphasizes the need for such a type of comparison. In fact, to the best of our knowledge, no previous literature has been developed regarding practical comparisons of time series generative methods. The only similar contributions have been found in data augmentation for time series. The main difference between these approaches is that, while data augmentation is focused on the expansion of a base dataset finalized to a specific task (often to reduce the class imbalance in a classification framework) generative approaches are focused on the creation of a completely new dataset by approximating the probability distribution of the original data, either explicitly (statistical methods) or implicitly (neural network methods). It is also clear that, while generative approaches can be exploited for data augmentation, the opposite is not true.

Regarding data augmentation comparative works, Guennec et al. (2016) proposes a comparison of time series data augmentation techniques with the task to fed data into a convolutional classifier. Another work is Um et al. (2017), where different approaches for time series augmentation are compared in the context of wearable sensors. In particular, the generated series are exploited jointly with the real ones to train a convolutional classifier for Parkinson's. In this view, data augmentation shows to be able to enhance the model's predictive capability. Furthermore, the importance of applying ensemble techniques to improve the quality of generated data is emphasized.

An extensive study on reviewing and evaluating time series data augmentation approaches is provided in Iwana and Uchida (2021). In more detail, this paper provides a careful explanation of the most relevant works related to time series augmentation and deals with generative approaches. Furthermore, it is provided with a comparative study of augmentation methodologies for classification tasks over 128 publicly available datasets. Similar to the work mentioned above, there is Wen et al. (2020) that highlights the weaknesses of data augmentation approaches in dealing with the temporal dimension, which is intrinsic in time series data. Furthermore, it is essential to underline that the conclusion of this work provides a critical overview of state-of-the-art research, and they give five guidelines for future works. We underline that our work aims to address two of these points, which are the analysis of generative models based on neural networks and the study of ensemble methods for time series data generation.

2.3. Why this paper?

Our work aims to be the first contribution to the practical comparison of neural network generative approaches for time series. In particular, we want to provide a task-free comparison of the examined approaches by considering almost only metrics related to the statistical distribution of fake data, its distance to that of original samples and its capability to be used as a basis to reconstruct the original information. In order to provide a wide overview of the performances of some state-of-the-art methodologies, we carry out experiments on different datasets and time granularities.

To the best of our knowledge, our work is the first attempt in this direction. All the previous similar works we have found in the literature are mainly focused on other tasks. In more detail, regarding (Guennec et al., 2016), several differences occur. Firstly, their work is concerned with data augmentation, which is a slightly different task than time series generation, as pointed out above. Secondly, it does not take into account neural network-based models, nor GMMN or GAN-based ones. Furthermore, while in our comparison experiments with the same data but at different time granularities are carried out, it is not true for Guennec et al. (2016). Moreover, that work is focused on a specific task, which is the classification of time series. Instead, our work aims to be task-free and more concerned with the statistical properties of fake and real samples. Regarding Um et al. (2017), it exhibits all the limitations of the previous work. Furthermore, another limitation is represented by the number of experiments carried out, as it takes into account just one dataset.

As for Iwana and Uchida (2021), as already pointed out it is made up of two parts: a survey and an experimental stage. In particular, in the second part, the generative models are totally neglected. In contrast, our paper is specifically focused on the experimental comparison of the generative models. About Wen et al. (2020), it proposes different comparisons for different tasks. However, no deep generative models are used for their comparison. At the same time, no attention is paid to the ensemble strategy, and no different time granularities for the same dataset are taken into account. Finally, although Brophy et al. (2021) is strongly related to deep learning-based generative approaches for time series, and despite the great attention paid to describing all the possible application fields, the experimental part is missing.

3. Generative models

In this section, we discuss the models exploited for comparison. We have searched for neural network-based generative approaches whose Python code is publicly available online. In particular, the starting point for the source code is provided by the GitHub repositoryhttps://github.com/SigCGANs/Conditional-Sig-Wasserstein-GANs. Then we have slightly modified some code to allow for better customization of hyperparameters. Among the algorithms exploited in this work, we can identify two subsets: from one side, there are GAN approaches, namely the TimeGAN, Recurrent Conditional GAN and Conditional Sig-Wasserstein GAN; on the other side, there is an example of general neural network approaches, namely the Generative Moment Matching Networks. Table 1 shows the main characteristics and differences between the proposed models.

After defining the generative problem faced up, we briefly discuss the theoretical aspects of the considered algorithms.

3.1. The generative problem

The task this work deals with is the comparison of generative approaches for time series. More specifically, let's assume we have a set of real time series data $\mathcal{X} = \{X_j\}_{j\in\mathcal{J}}$ with $\mathcal{J} = \{1, \cdots, J\}$. Each X_j is assumed to be a multivariate time series $X_j = \{\mathbf{x}_{j,t}\}_{t\in T}$, where the time set $T = \{1, \ldots, N\}$ is common to all the considered time series and each observation is a vector $\mathbf{x}_{j,t} \in \mathbb{R}^d$ with $d \in \mathbb{N}$. We assume the real data follow from an underlying distribution p_{data}.

The generative model task is to learn a probability distribution p_{fake} which approximates p_{data}. Then, p_{fake} is used to generate a new dataset of fake time series, namely $\mathcal{F}$. For simplicity, we generate as much fake time series as the number of original ones, so $\mathcal{F} = \{F_j\}_{j\in\mathcal{J}}$ with $F_j = \{\mathbf{f}_{j,t}\}_{t\in T}$ and $\mathbf{f}_{j,t} \in \mathbb{R}^d$.

Table 1
Main differences between the approaches considered for the comparison.

	Gan Based	Dim Reduction	Stat-Based Loss	RNN vs AR-FNN
SigCWGAN	Yes	No	No	RNN
RCGAN	Yes	Yes	No	RNN
TimeGAN	Yes	No	Yes	AR-FNN
GMMN	No	Yes	Yes	AR-FNN

We point out that, at least for neural network-based generative approaches, the generation of the new distribution p_{fake} is done by mapping random samples $\mathcal{Z} = \{\mathbf{z}_j\}_{j\in\mathcal{J}}$ extracted by a well-known distribution p_z (usually uniform or normal) into fake samples which are compared to the original data in order to enhance the quality of the generative process. How this goal is accomplished by each algorithm is discussed in more detail in the following subsections.

3.2. Generative adversarial networks

GANs have been introduced by Goodfellow et al. (2014) in 2014. The main idea behind GAN is to train two neural networks one against the other. In more detail, the Generator (G) has to create fake samples starting from a random vector $\mathbf{z}$ sampled from a well-known distribution p_z; the Discriminator (D) has to distinguish real X from fake $G(\mathbf{z})$ samples, i.e. it has to provide the probability $D(X)/D(G(\mathbf{z}))$ that a real/fake sample belongs to the real dataset. In such a way, G is forced to learn the distribution of real data p_{data}, i.e., it learns to map the well-known distribution into the target one. Actually, what G really learns is an approximation p_{fake} of the original distribution. The two networks can be viewed as the players of a min–max game, whose value function is:

$$V(D,G) = \mathbb{E}_{x\sim p_{data}}[logD(X)] + \mathbb{E}_{z\sim p_z}[log(1-D(G(\mathbf{z})))] \quad (1)$$

Clearly, D tries to maximize $V(D,G)$, while G aims to minimize it, giving rise to the following min–max problem:

$$\min_G \max_D V(D,G) \quad (2)$$

The gradient, computed after D iteration, is then backpropagated to both D and G, in order to update weights.

In more detail, assuming the batch is made up of J real samples $\mathcal{X} = \{X_j\}_{j\in\mathcal{J}}$ with $\mathcal{J} = \{1,\ldots,J\}$ and J fake samples $\mathcal{F} = \{F_j\}_{j\in\mathcal{J}}$ with $F_j = G(\mathbf{z}_t)$ and $\mathcal{Z} = \{\mathbf{z}_j\}_{j\in\mathcal{J}}$ are the random vectors sampled from p_z, the two loss functions are: Fig. 1.

$$\mathcal{L}_D = -\frac{1}{2J}\left[\sum_{t=1}^{J}\log D(X_j) + \sum_{t=1}^{J}\log(1-D(F_j))\right] \quad (3)$$

$$\mathcal{L}_G = \frac{1}{J}\sum_{t=1}^{J}\log(1-D(F_j)) \quad (4)$$

The overall scheme of adversarial training is summarized in Fig. 2.

Actually, Eq. 2 requires that D is trained at each step of G. As it is computationally too expensive, it is preferred to alternate a certain number k of D steps for each G step, in order to improve convergence speed. The convergence of the overall algorithm is still guaranteed by theoretical results, as shown in the original paper Goodfellow et al. (2014). Finally, it is noteworthy to remark that, although the original paper is concerned about static data, several extensions have been proposed to address the time series problem. In the following, we describe those used in our experiments.

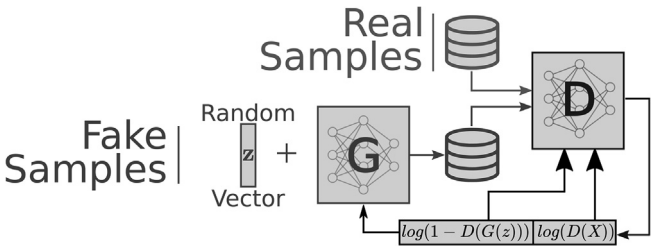

Fig. 1. Generative Adversarial Network architecture. The Generator (G) creates fake samples starting from random noise. Then, the Discriminator (D) has to distinguish between real and fake samples and its error is backpropagated to train both D and G.

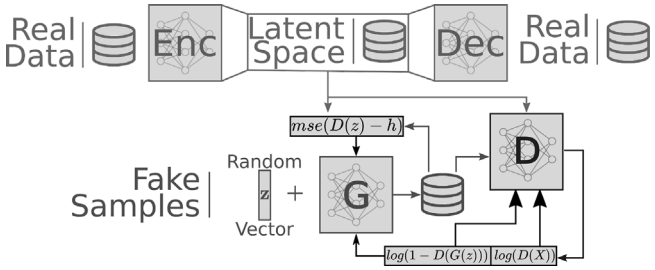

Fig. 2. Time GAN scheme. An AutoEncoder and a GAN work jointly. Firstly, the AutoEncoder is trained. Then, it is frozen and the GAN is applied to the latent space extracted from the Encoder, with an additional supervised loss. Finally, after the training, the fake samples are generated by applying the Decoder to the Generator.

3.3. Recurrent conditional GAN

RCGAN (Esteban et al., 2017) is one of the first GAN-based architectures specifically designed to handle time series. The main scheme is the same as in the standard GAN: from one side, there is the Generator which creates new fake samples starting from a random vector; on the other side, there is the Discriminator, which has to distinguish between real and synthetic samples. In this case, as RCGAN was specifically designed to handle time series, both the Generator and Discriminator are made up of RNN. In particular, they are constructed starting from the most widely used type of RNN, namely the LSTM. So, each element in the batch is a multivariate time series and the G output is itself a sequence. Instead, D is trained to discriminate among real and synthetic time series, that is starting from an input sequence it returns as output a number in the $[0,1]$ interval which represents the probability that the considered series belongs to the real dataset. As for the loss functions exploited by G and D, they are the same cross entropy-based ones as in the standard GAN.

Despite its simplicity, RCGAN has been shown to provide excellent performances in different scenarios and under several evaluation metrics. In particular, it has been tested with synthetic data and with medical signals coming from the Intensive Care Unit. The results show the reliability of samples generated by RCGAN and their usefulness in real scenarios.

3.4. Time GAN

TimeGAN (Yoon et al., 2019) is a GAN model constructed for time series which exploits AE to reduce problem dimensionality. For simplicity, in the following, we omit the sample index j. In particular, TimeGAN focuses on the approximation of the conditional distribution $p_{data}(\mathbf{x}_t|\mathbf{x}_{1:t-1})$. To achieve this task, TimeGAN is made up of two pieces: from one side, there is the AE, and on the other side, there is the GAN which operates on the low-dimensional data representation provided by the latent space. Furthermore, an

enhancement in the approximation of conditional distribution is achieved by also exploiting a supervised loss which measures the distance between the real latent space obtained from data and the fake latent space generated by G.

To formalize the architecture, we use the same notation as before for real data and input noise. Furthermore, the latent space is indicated with $\{\mathbf{h}_t\}_{t\in T}$, the AE output is $\{\hat{\mathbf{x}}_t\}_{t\in T}$, the encoder function is $e = e(\mathbf{h}_{t-1}, \mathbf{x}_t)$ and the decoder function is $d = d(\mathbf{h}_t)$. So, the AE structure is summarized in this way: $\mathbf{h}_t = e(\mathbf{h}_{t-1}, \mathbf{x}_t)$ and $\hat{\mathbf{x}}_t = d(\mathbf{h}_t)$. As for G and D, they operate at the latent space level. So, the generator is defined as $\tilde{\mathbf{h}}_t = G\left(\tilde{\mathbf{h}}_{t-1}, \mathbf{z}_t\right)$ and the fake samples can be obtained as $\mathbf{f}_t = d\left(\tilde{\mathbf{h}}_t\right)$. Instead, the discriminator is slightly more complex as it is based on bidirectional recurrent layers, so $D = D\left(\overrightarrow{\mathbf{u}}_t, \overleftarrow{\mathbf{u}}_t\right)$ where $\overrightarrow{\mathbf{u}}_t$ represents the forward direction of the discriminator, so $\overrightarrow{\mathbf{u}}_t = \overrightarrow{c}\left(\mathbf{h}_t, \overrightarrow{\mathbf{u}}_{t-1}\right)$ and $\overleftarrow{\mathbf{u}}_t$ is the backward direction, that is $\overleftarrow{\mathbf{u}}_t = \overleftarrow{c}\left(\mathbf{h}_t, \overleftarrow{\mathbf{u}}_{t+1}\right)$. In the above formulae, $\overrightarrow{c}, \overleftarrow{c}$ denote the forward and backward recurrent functions, and the same formulation can be obtained by using fake latent space instead of the real one.

3.5. Conditional Sig-Wasserstein GAN

SigCWGAN (Ni et al., 2020) is a GAN-based generative approach for time series based on the Conditional Signature Wasserstein-1 metric (C-Sig-W_1). As for the architecture, the generator exploits an AR-FNN, in order to better mimic the conditional distribution of real data. Instead, the Discriminator exploits the C-Sig-W_1 to measure the distance between the real and generated distribution. To achieve this task, the starting point is the expected signature S of a random process, which can be thought of as the corresponding of the moment generating function for random variables. The signature is exploited in order to efficiently handle non-stationary time series that is, time series whose statistical properties evolve through time. In more detail, let $X = \{\mathbf{x}_t\}_{t\in T}$ with $\mathbf{x}_t \in \mathbb{R}^d$ be a multivariate time series. We can define the tensor algebra space $T(\mathbb{R}^d) = \oplus_{k=0}^{\infty}(\mathbb{R}^d)^{\otimes k}$, that is $T(\mathbb{R}^d)$ is the direct sum of all the spaces of tensors of order k defined on $\mathbb{R}^d$ by varying the order k. By convention, $(\mathbb{R}^d)^{\otimes 0}$ is defined to be the real space $\mathbb{R}$. Now, the signature of the path X is defined as the element

$$S(X_T) = \left(1, X_T^1, \cdots, X_T^k, \cdots\right) \in T(\mathbb{R}^d) \tag{5}$$

in the space $T(\mathbb{R}^d)$ such that:

$$X_T^k = \int_{t_1<\cdots<t_k} dX_{t_1} \otimes \cdots \otimes dX_{t_k} \tag{6}$$

Observe that, as the order k increases, we have a finer partition of the time set T which provides information ever more local about the behavior of the time series.

Actually, in practical applications, in place of $S(X_T)$, it is considered its truncated version

$$S_M(X_T) = \left(1, X_T^1, \cdots, X_T^M\right) \tag{7}$$

also known as the truncated signature of X of degree M.

Thus, assuming $\mathbf{x}_{t+1}$ is conditionally generated under the probability distribution p_{data} or p_{fake}, it is possible to define the C-Sig-W_1 distance between the two distributions in this way:

$$C-Sig-W_1^M(p_{data}, p_{fake}) = \|\mathbb{E}_{\mathbb{p}_d}[S_M(\mathbf{x}_t)|\mathbf{x}_{1:t-1}] - \mathbb{E}_{\mathbb{p}_f}[S_M(\mathbf{x}_t)|\mathbf{x}_{1:t-1}]\|_{l_2} \tag{8}$$

Note that the conditional expectations are estimated via the Monte Carlo method. It is noteworthy to observe that, by minimizing the C-Sig-W_1 between p_{data} and p_{fake} (and in particular their conditional expectations), SigCWGAN is enforcing the generated time series to have the same statistical properties as the original ones.

3.6. Generative moment matching networks

GMMN (Li et al., 2015) is a neural network generative approach designed to overcome the difficulties related to the min–max GAN training. To do this, a specifically designed loss function, namely the *Maximum Mean Discrepancy* (MMD) is exploited to catch all orders of statistics, thus perfectly matching p_{data} and p_{fake} (in fact, it has been shown that MMD = 0 implies the asymptotic convergence of the two distributions). Most importantly, MMD is differentiable (if its kernel is differentiable), so it can be successfully used within the back-propagation framework.

Formally, after defining a proper kernel $k = k(\mathbf{x}, \mathbf{g})$, by exploiting the same notation above we can write the loss function as:

$$\mathcal{L} = \frac{1}{J^2}\sum_{i=1}^{J}\sum_{j=1}^{J}k(X_i, X_j) - \frac{2}{J^2}\sum_{i=1}^{J}\sum_{j=1}^{J}k(X_i, F_j) + \frac{1}{J^2}\sum_{i=1}^{J}\sum_{j=1}^{J}k(F_i, F_j) \tag{9}$$

In practical applications, a good choice could be the Gaussian kernel, defined as:

$$k(X_i, X_j) = exp\left(-\frac{1}{2\sigma}|X_i - X_j|^2\right) \tag{10}$$

In this way, by minimizing the MMD, GMMN is able to match all moments of the two distributions. As for the neural network itself, it is made up of sequential AR-FNN layers which are designed to map a uniform distribution into p_{data}.

Besides this first, simple formulation of GMMN, another version has been proposed to combine the strong points of both GMMN and AE. In this framework, the idea is to first train the AE on the real data, thus obtaining the latent space representation of the real samples. Then, the generative method is applied to generate the latent space, which is later passed through the Decoder in order to obtain the fake samples.

4. Description of the experimental setting

In this section, we provide all the detail about the experimental stage. Firstly, we describe the implementation of each generative technique. Then, we show the datasets used for the comparison. Finally, we discuss the metrics used for the comparison. The results of the experiments are shown and discussed in the next section.

4.1. Implementation details

The procedure adopted for the comparison is almost the same for each experiment carried out. Firstly, the data are scaled in the $[0, 1]$ range; then, two cases have been considered. If the samples in the dataset are obtained as sliding windows of a unique, long time series, then the scaling is performed before windows creation. Instead, if the samples belong to different time series, each one is scaled independently from the others.

After the scaling, the generative models are applied to generate fake samples. In particular, each method has a vector of hyperparameters to be tuned. We carry out 10 experiments for each method using the same number of random hyperparameters vectors. The hyperparameter space in which we search is almost the same for each experiment. In particular, it is described in Table 2.

Table 2
Hyperparameter space. The hyperparameter vectors in the experiments are sampled within this space. **Common** represents the hyperparameters common to all models. In particular, **batch** is for the batch size, **n_hidden** represents the number of hidden layers in the networks, **hidden_dim** indicates the dimension of the hidden layers. **mc_size** represents the number of monte carlo simulations, **depth_p/f** is the truncated order for the signature of the real/fake path, **scale_p/f** is used in the computation of the truncated signature and **lr** is for the learning rate. Furthermore, **lr_G** is the learning rate specifically of the Generator, **lr_D** is the learning rate only for the Discriminator and **step_D_G** is the number of Discriminator steps for each Generator step.

Hyperparameters Table				
Common	**batch**	**n_hidden**	**hidden_dim**	
	4–16	2–3	40–100	
SigCWGAN	**mc_size**	**depth_p/f**	**scale_p/f**	**lr**
	600–1200	2–3	0.15–0.35	5e−4–0.05
RCGAN	**lr_G**	**lr_D**	**step_D_G**	
	5e−5–2e−4	5e−5–4e−4	1–4	
TimeGAN	**lr_G**	**lr_D**	**step_D_G**	
	5e−5–2e−4	5e−5–4e−4	1–4	
GMMN	**lr**			
	5e−5–3e−4			

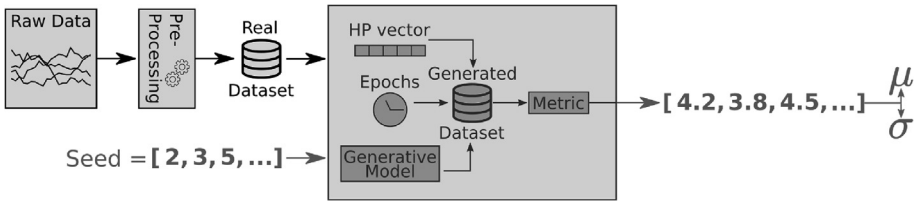

Fig. 3. Comparison pipeline. Firstly, the dataset is preprocessed, and the real dataset is extracted. Then, fixed the hyperparameters vector and the number of epochs, the generative model is applied several times by varying the seed. Then, for each iteration, the metrics are computed. So, for each metric, there is a vector of values (one for each seed). Finally, for each metric, the mean and the standard deviation of the values vector are obtained, and the wcs is computed.

More detailed information about the hyperparameter vectors exploited in each experiment is provided in the Supplementary Material.

Moreover, we analyze the samples generated at different epochs, namely after $[100, 200, 500, 750, 1000, 1500, 2500, 5000]$ epochs. It can be useful to understand how the approximated solution p_{fake} found by the generative models evolves over time.

Furthermore, to assess the stability of the methods, we perform 25 launches by using, like seeds, the first 25 prime numbers. Then, after computing the metrics for each generated dataset, their mean μ and standard deviation σ across seeds are computed, and the *worst-case scenario* (wcs) is obtained. The wcs can be thought of as the bound of the confidence interval in the situation opposite to the best one. Formally, it is defined as $\mu + \sigma$ for those metrics where a low value is desirable and as $\mu - \sigma$ for the other metrics.

Finally, the execution times are recorded both for training and for fake sample generation. The comparison pipeline is summarized in Fig. 3.

As for the hardware exploited, we work on three different workstations:

- Intel(R) Core(TM) i9-9900 K CPU @ 3.60 GHz, with 128 GiB RAM, and GeForce RTX 2080 ti is used for Noval Weekly, Electricity BiMonthly and Finance Quarterly experiments.
- Intel(R) Core(TM) i9-9900 K CPU @ 3.60 GHz, with 128 GiB RAM, and GeForce RTX 3080 is exploited for Beijing 4Yearly, Electricity Quarterly and Finance Yearly.
- AMD Ryzen 5 3600 6-Core Processor, with 32 GiB RAM, and GeForce RTX 3070 is used for all the other experiments.

The different workstations exploited in the experimental stage affect the computational times needed by the algorithms. However, as only one workstation is used in each experiment, this does not have a direct impact on the comparison.

4.2. The datasets

In the experimental stage, we exploit five datasets for the comparison of generative models for time series. Three datasets are publicly available, while the other two are private. The considered datasets exhibit different peculiarities, such as solid seasonalities or predominance of the autoregressive part, to give a wide overview of the performances of generative models in different scenarios. Table 3 summarizes the dimension of the dataset in each experiment, while further details about the considered datasets are provided below. Graphical examples for every experiment are reported in the Supplementary Material. The datasets publicly available are the following ones:

- **Beijing**.[1] The dataset describes air pollutants in 12 different points in Beijing. Each point is treated as a sample of multivariate time series. Each time series is made up of 6 variables referred to as PM2.5, PM10, SO2, NO2, CO and O3, respectively. Three different lengths are considered for the time windows, namely monthly, yearly and 4-years. In the first case, we exploit daily observations made up by averaging the hourly values in the raw dataset and the considered period is January 2016. In the second case, we use weekly time granularity and we consider 2016 as the reference period. In the last case, the time series are made up of aggregated monthly values and they cover the period between 1^{st} March 2013 to 28^{th} February 2017. Furthermore, there are some missing values. We impute these values with the previous value. An example of this series in the monthly case is provided in Fig. 4.

[1] Zhang et al. (2017).

Table 3
Summary of the datasets used for the experiments and their dimension. For each dataset, the table shows all the exploited shapes. Furthermore, below each dataset and each considered shape, the dimension of the input tensor is reported. In more detail, the first dimension is equal to the number of available observations, the second one to the length of each time series and the last represents the number of variables.

Beijing	Monthly	Yearly	4Yearly
	(12, 31, 6)	(12, 53, 6)	(12, 48, 6)
Electricity	Monthly	BiMonthly	Quarterly
	(36, 31, 30)	(18, 62, 30)	(12, 93, 30)
Finance	Monthly	Quarterly	Yearly
	(12, 21, 5)	(12, 62, 5)	(12, 52, 5)
Noval	BiWeekly	Weekly	
	(31, 6, 14)	(67, 6, 14)	
ToIT	Daily	Weekly	
	(84, 24, 6)	(12, 168, 6)	

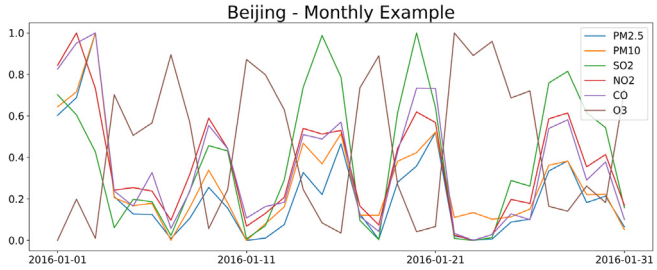

Fig. 4. Sample of monthly series in the Beijing dataset.

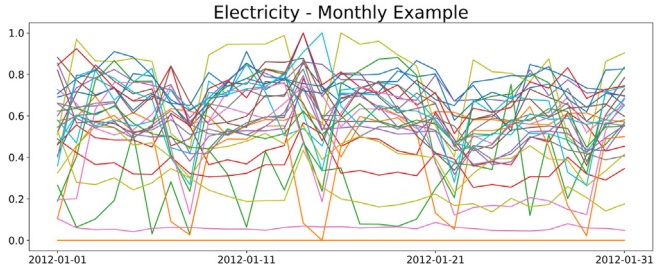

Fig. 5. Example of Electricity monthly multivariate time series.

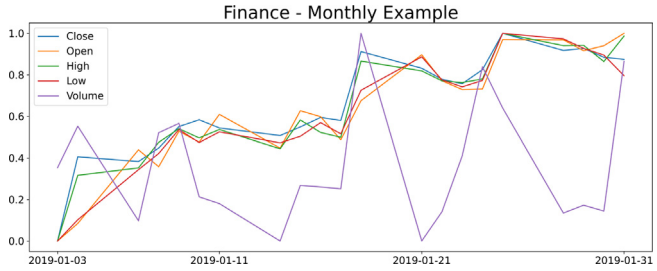

Fig. 6. Example of monthly series is the Finance dataset. Each sequence corresponds to a stock index.

- **Electricity**.[2] This dataset is made up of the observations regarding daily power consumption, expressed in kW, from several consumers. We take into account only the first 30 customers, which are regarded to be the variables of the multivariate time series. Furthermore, the considered period covers from 1^{st} January 2012 to 31^{st} December 2014 and the data are aggregated into daily samples. The samples are generated by working with sliding time windows. We carry out three different experiments on this dataset by changing the length of the windows. In more detail, we generate both monthly, bi-monthly and quarterly time series, with a number of timesteps equal to 31, 62 and 93, respectively. 0-padding is applied for the months with less than 31 days. Furthermore, time windows have strides equal to their length, so there is no overlap between different samples. Fig. 5 shows an example of multivariate time series in the monthly case.

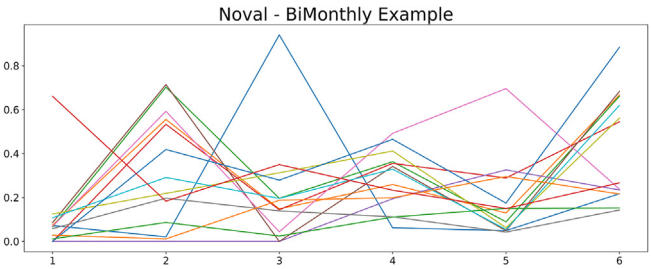

Fig. 7. Multivariate time series in Noval dataset with biweekly aggregation.

- **Finance**.[3] The dataset represents 12 financial indexes of the most important European stock markets. In particular, we exploit the following data: [*AEX, ATX, BEL20, CAC40, DAX, EuroStoxx50, IBEX35, OMX Copenhagen25, OMX Stockholm30, PSI20, SMI, WIG20*]. Each index can be viewed as a multivariate time series with 5 attributes: *Open* price,*Close, High, Low* and the exchanges *Volume*. In this case, the data are scaled series by series that is, each index has its own scaler. We carry out experiments on two different time granularities and three sequences length: monthly and quarterly series are constructed by exploiting daily data in the periods from 3^{rd} January 2019 to 31^{st} January 2019 and to 29^{th} March 2019 respectively; yearly series are obtained during 2019 by using weekly data. In Fig. 6 an example related to monthly series is provided.

The private datasets are listed below.

- **Noval**. The dataset contains information related to the consumption of semifinished products by a window fixtures Italian factory in the period between 30^{th} March 2020 and 07^{th} June 2021. There are 14 materials, which are the variables of the problem. The consumption is aggregated into weekly and biweekly data and sequences of 6 timesteps each one are created, with overlapping. Fig. 7 provides an example of time series in this dataset with biweekly aggregation.
- **ToIT**. The dataset is related to the occupancy rate of parking in the central districts of Caserta and Naples, Italy. The occupancy rate is defined as the ratio between occupied parking and available parking, so it is a number in the interval $[0, 1]$. The observations are hourly recorded by specific sensors at 6 different points, which are considered the variables of the multivariate time series. The total time series, which covers the period from 8^{th} December 2019 to 29^{th} February 2020, is split into daily and weekly windows, without overlapping. So, the daily dataset contains 84 sequences of 24 observations each and the weekly dataset is made up of 12 sequences containing 168 elements. Fig. 8 contains an example referred to the weekly dataset, where it is possible to observe strong daily seasonality.

To summarize, by using the five datasets above mentioned and by varying the length and time granularity of the considered data, a

[2] https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014.

[3] https://www.investing.com/.

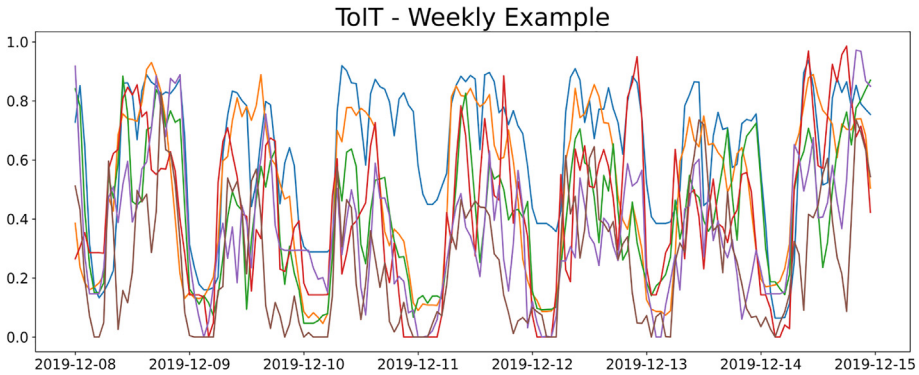

Fig. 8. Weekly time series made up of hourly observations regarding the ToIT dataset.

total of 13 experiments have been carried out for this comparison. Among them, different situations occur. For example, the experiments related to finance exhibit trends and autoregressive patterns. Instead, Beijing experiments and ToIT weekly show more or less strong seasonalities. Furthermore, also in Electricity, there are some seasonality patterns, even if they are less visible, often due to drops in the consumption during the weekend. Moreover, the experiment ToIT daily has the peculiarity that all its samples have a standard, underlying, bell shape. Finally, in the experiments related to the Noval dataset, no visible patterns seem to occur.

4.3. Metrics

As already pointed out in previous sections, the debate regarding metrics for time series generative models is still open and no universally approved metrics have been defined. Furthermore, it seems that each proposal is able to catch just a perspective of the problem and no overall quantities have been proposed in the literature. In this subsection, we provide an overview of the measures used for the comparison. In particular, the metrics exploited in this work can be divided into two macro-categories and four micro-categories. The macro-categories are *Distribution* and *Innovation*. The former is made up of those metrics which catch particular aspects of the two distributions, namely p_{data} and p_{fake}, to understand how close they are. In this context, as we are looking for generated samples which are as representative of the original data as possible, we aim to find low distance values.

However, looking at the metrics in the above-mentioned class, it is straightforward to see that a perfect copy of the original samples would give the best possible results. So, we introduce another category of metrics which are designed to highlight how much the fake samples are distant from the original ones, that is how remote the generated samples are in a certain space with respect to the original ones. In this way, we aim to avoid duplicates of original data. Finally, it is noteworthy to observe that, at least for neural network-based approaches, as the fake samples are generated from random noises, they should be different from simple copies of the original ones.

Following the above distinction, we briefly describe the metrics:

- **Innovation**

  Dynamic Time Warping Nearest Neighbour (NN). For each observation in the fake dataset, we compute the Dynamic Time Warping (DTW) distance from each multivariate time series in the real dataset. Then, the average distance from the nearest neighbour is returned as output. In particular, we search for high values of NN, which should ensure the generated samples are far away from the given data.

- **Statistical Properties**

  – *Kullback–Leibler Divergence (KL).* We compute the Kullback–Leibler Divergence (KL) (Kullback and Leibler, 1951) to estimate the distance between the real data distribution and the generated one. Accordingly, low values for KL are better than high values.
  – *Cross-Correlation (CC).* As in Ni et al. (2020), we evaluate the average difference between cross-correlation in real and fake data. In more detail, for each multivariate time series in the real dataset, we compute the tensor of cross-correlation matrices among variables. Then, we average the tensors obtained from each observation and we repeat the process for the generated time series. Finally, we estimate the l_1 norm of the difference between the real and generated tensors to estimate how the cross-correlation of fake data is close to that of the real data. So, we aim to find low values for CC.
  – *Histogram (Hi).* In a way similar to CC, we also estimate the difference between histograms. In particular, for each variable of the time series, we compute the histogram, so for each time series, there is a matrix whose rows are the histograms of each variable. Finally, the matrices are averaged in the real and generated datasets and the l_1 norm of their difference is used as a proxy for the distance between real and fake data.

- **Distribution**

  – *Discriminator with K-Nearest Neighbours and Principal Component Analysis (KP)* This metric exploits the accuracy obtained by K-Nearest Neighbours (KNN) algorithm in discriminating real from fake samples. The features used by the KNN are the projection of sequences on the principal components extracted via Principal Component Analysis (PCA). The accuracy is obtained via Cross Validation (CV). Furthermore, the dataset is balanced as the number of generated samples is equal to that of real ones. In particular, as our aim is to assess how indistinguishable are real and fake data, we measure the proximity of final accuracy to 0.5, that is we consider the absolute value of the difference $accuracy - \frac{1}{2}$. In this view, low values of KP are preferred.
  – *Discriminator with KNN and t-Distributed Stochastic Neighbor Embedding (Kt)* The aim of this metric is the same as KP. However, in this case, the features fed into the classifier are

extracted via t-Distributed Stochastic Neighbor Embedding (tSNE) (Van and Hinton, 2008). Also for Kt, we hope to find low values.
– *Discriminator with Random Forest and PCA (RP)* This metric evaluates the indistinguishable of real and fake samples by exploiting the Random Forest (RF) (Breiman, 2001) classifier. The procedure is the same as above explained.
– *Discriminator with RF and tSNE (Rt)* This measure assesses the proximity of generated data to real one by exploiting the RF classifier fed with features obtained from the time series via tSNE.
– *Discriminator with KNN and DTW (KD)* This metric directly works with raw multivariate time series by exploiting the DTW as distance measures. The algorithm used for the classification is the KNN. The other details are the same as above.

- **Prediction**

– *Train Synthetic Test Real KNN (TK)* As in Esteban et al. (2017), the aim of this metric is to evaluate how the generated samples can replace real data in the training of predictive algorithms. To do this, we exploit the fake time series to train the KNN regressor and then we evaluate its performance in the test set made up of real samples. We work variable by variable, averaging the results obtained from each univariate series. The loss function used to evaluate regressor performance is the mean squared error. Accordingly, low values of TK are preferred.
– *Train Synthetic Test Real RF (TR)* This measure works exactly as the previous one, with the only difference that an RF regressor is used instead of KNN.

We would like to clarify that our choice of metrics is linked to the needing to take into account several points of view related to the generation process. In other words, since we are interested in providing a complete summary of the time series generated by the considered models, we aim to use a huge amount of metrics. So, we have considered almost all the most frequently used metrics and we have added new ones which could contribute to enhancing the understanding of the generative processes.

5. Experimental results

In this section, we show and comments on the results obtained in the experimental stage. Firstly, we describe the aggregation criterion exploited and we show some graphical example of generated series. Then, we discuss the ensembling strategy. Then, we discuss the results obtained and the computational time. Finally, we provide an insight into the temporal evolution of the generated samples.

5.1. Results aggregation

As already stated in the previous Section, several experiments have been carried out by changing the hyperparameters vector, the number of epochs and the seed. In order to assess the stability of the methodologies, the results obtained among seeds are aggregated by computing the mean, standard deviation and wcs for each metric.

Furthermore, to provide a better representation of the obtained results, in this stage also hyperparameters and epochs are aggregated by taking into account only the best result. That is, instead of considering 10 hyperparameters and 8 epochs (i.e., 80 launches for each algorithm in each experiment), we take into account only the best combination of hyperparameters and epoch. To find the best combination, we look only at the wcs and we work in two steps. In the first step, for each metric, we compute the min and max values obtained among the 80 launches. In the second step, for each launch, we compute an overall aggregated metric by summing the value of each metric scaled with a MinMax scaler. In particular, the NN is added with its negative value, as it is the only metric which is preferable to be high. Finally, the combination which exhibits the minimum sum of scaled wcs is considered to be the optimal one. Fig. 9 summarizes the procedure.

5.2. Graphical examples

We now provide some graphical example of generated series. In particular, Fig. 10 shows the series generated by the four models in the monthly dataset. All of them have a similar behavior, comparable with that of the original series.

Instead, Fig. 11 shows an example for the Electricity monthly generated series.

In Fig. 12 it is possible to observe the generated series in the monthly Finance dataset. In this case, it is interesting to observe the trend in the generated time series. In fact, all except one are increasing series. This is because all the series in this dataset are extrapolated during the bull market, with a rise in the value of the considered indexes. The only series with another behavior is maybe related to the volume, so it does not exhibit particular patterns.

Fig. 13 shows the series generated in the Noval BiWeekly dataset. In this case, no visible patterns are immediately recognizable.

Finally, Fig. 14 is related to examples of weekly series in the ToIT dataset. In this case, the quality of the generated samples is represented by the number of waves in the series. Recall that this dataset contains information related to the parking occupancy rate. On a weekly basis, we expect to find 7 waves, each one corresponding to a particular day. In fact, the generated series exhibit such behavior.

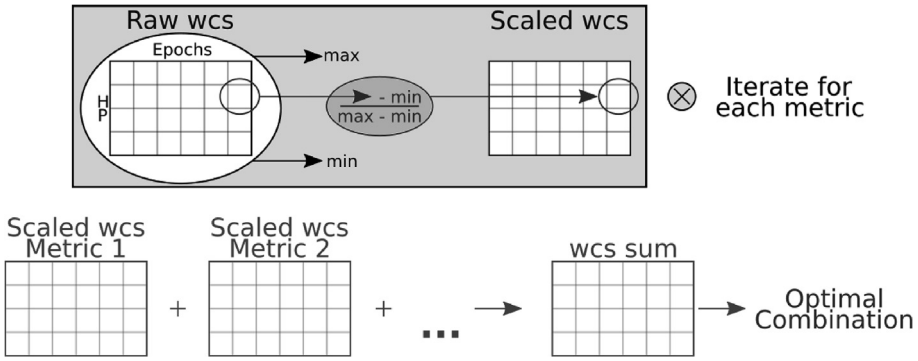

Fig. 9. Aggregation of the results. The metrics among different hyperparameters vectors and epochs are first scaled in the $[0, 1]$ range, then averaged.

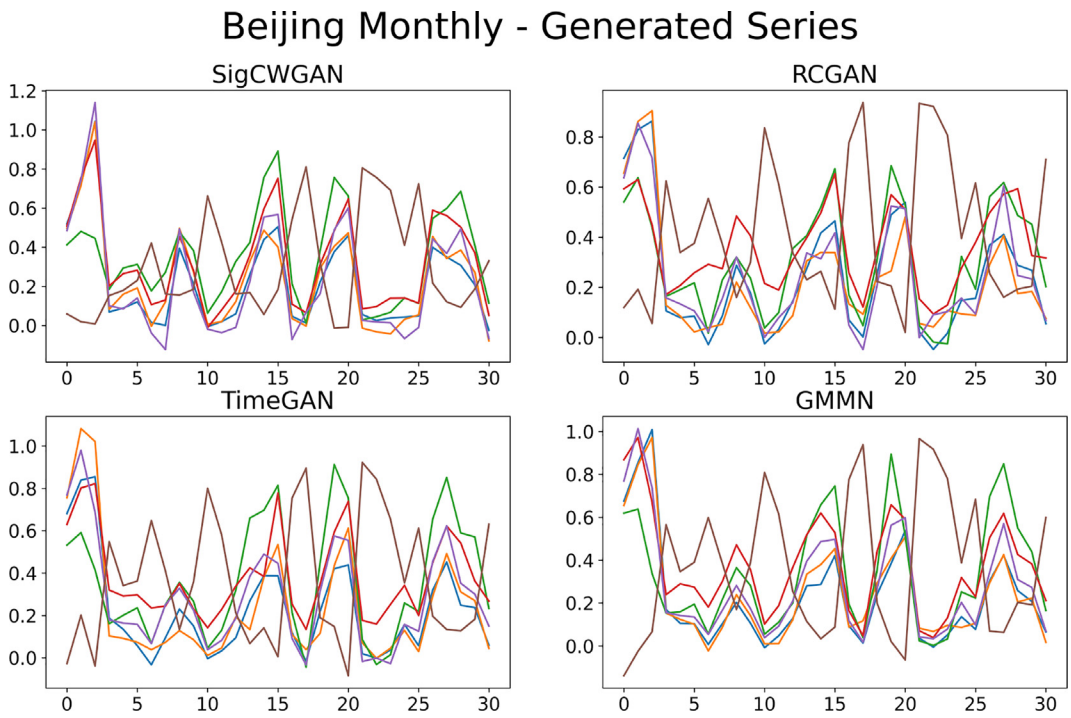

Fig. 10. Beijing Monthly – Example of fake series generated by the four generators.

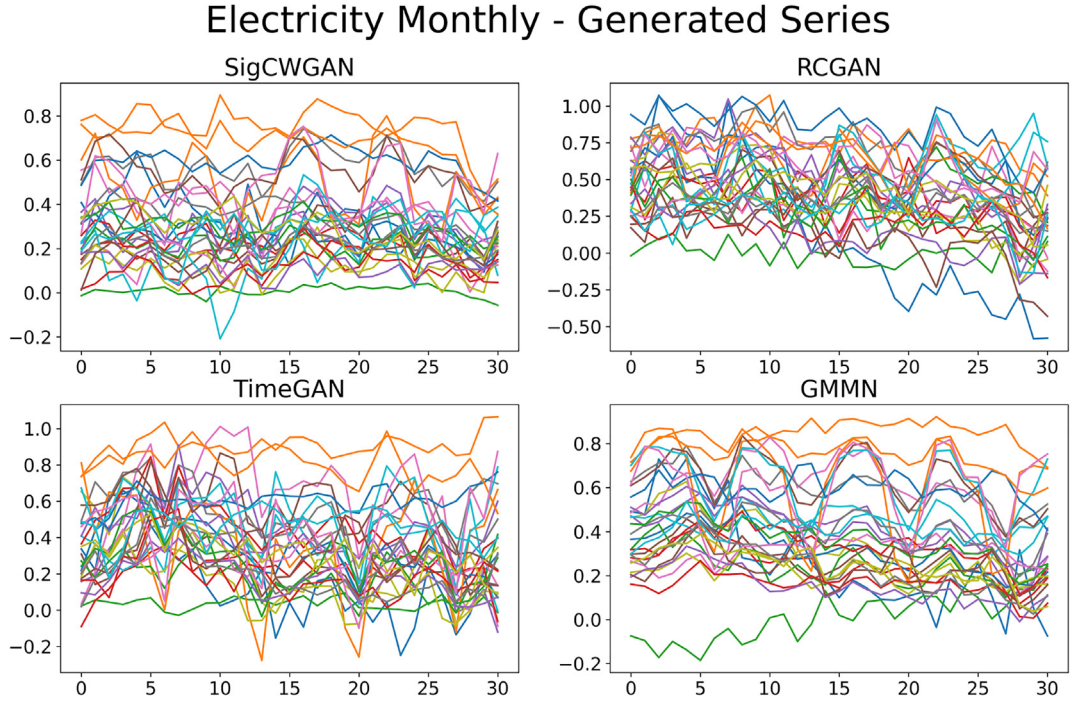

Fig. 11. Electricity Monthly – Example of fake series generated by the four generators.

5.3. Ensemble

Several recent studies (see, for example, Um et al. (2017) and Wen et al. (2020)) start to show the valuable contribution given by ensemble methods for data augmentation. We want to give a further contribution in this area by constructing simple ensembles of generative models and evaluating their performance by comparing them with the base models. In particular, for each experiment, the ensemble dataset is built by using an equal number of samples belonging to the optimal combination of each of the four algorithms exploited for the comparison. The results obtained are shown in the rest of this Section.

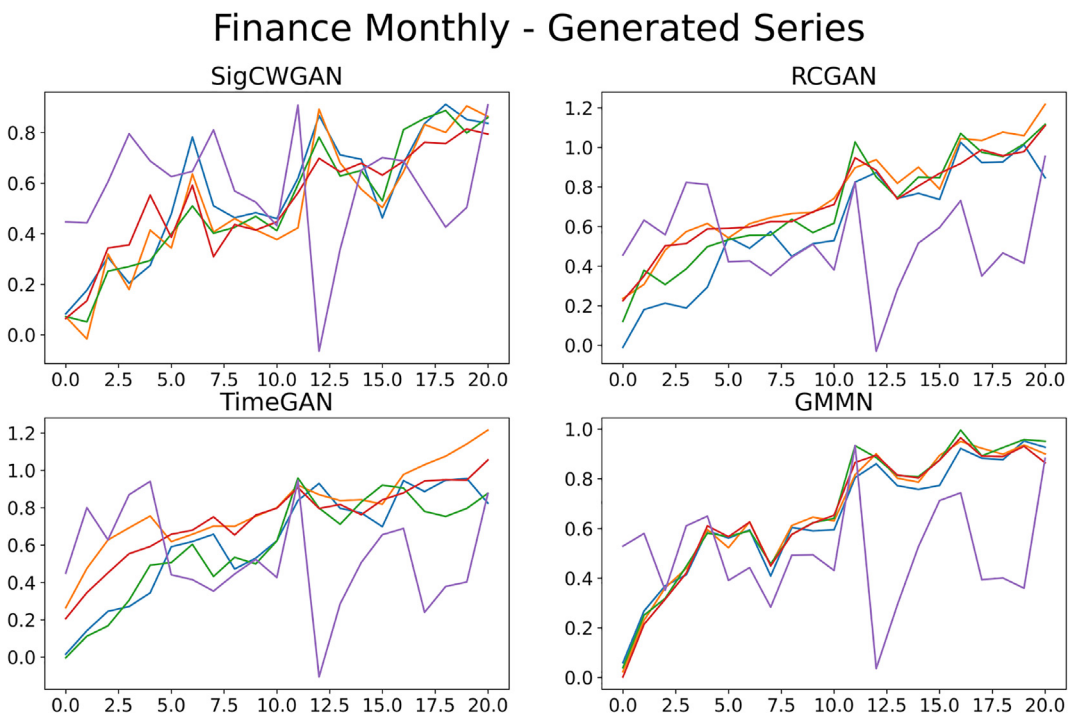

Fig. 12. Finance Monthly – Example of fake series generated by the four generators.

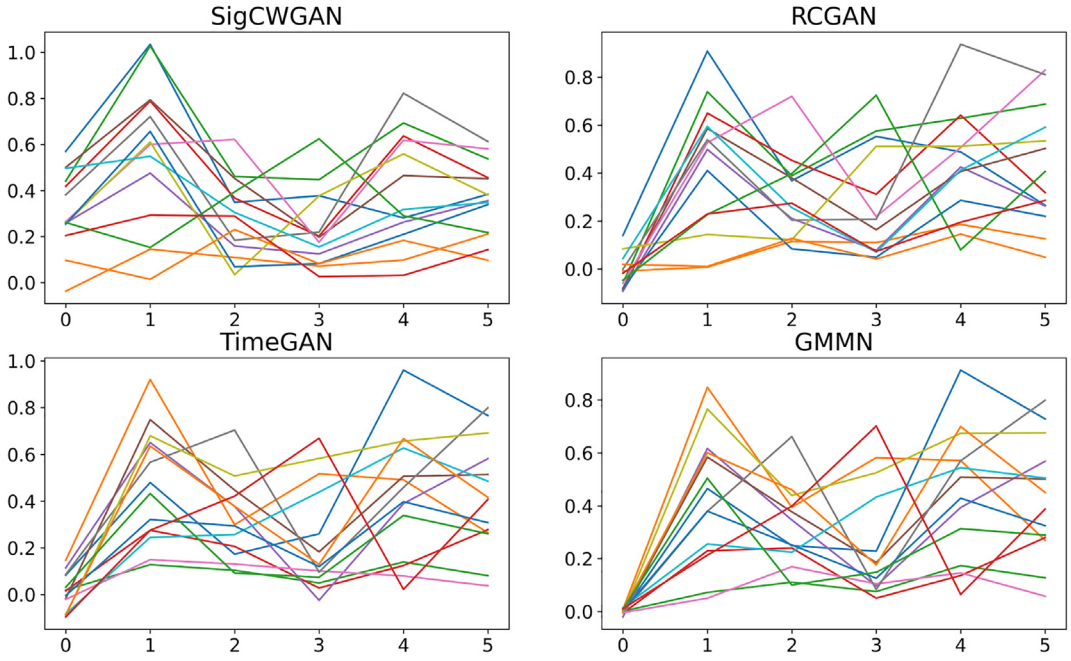

Fig. 13. Noval BiWeekly – Example of fake series generated by the four generators.

5.4. Results discussion

The full results tables are provided in the Supplementary Material. Here, we report a summary of the results obtained. Tables 4–8 summarize the results obtained for each dataset. To aggregate the metrics and to provide these summary tables, we work in two steps: firstly, we scale each metric in each experiment in the $[0, 1]$ interval; secondly, we average the results obtained by the same algorithm among the different experiments with the same dataset and the different metrics of the same type.

The results for the Beijing dataset are summarized in Table 4. As for the monthly experiment, there is a clear superiority of Ensemble, both in μ and wcs metrics. In particular, the KL metric is lower than one-half of the other competitors. Also, GMMN shows quite good performances, even if not at the Ensemble level. Furthermore, despite a great error in CC metric, also SigCWGAN provides note-

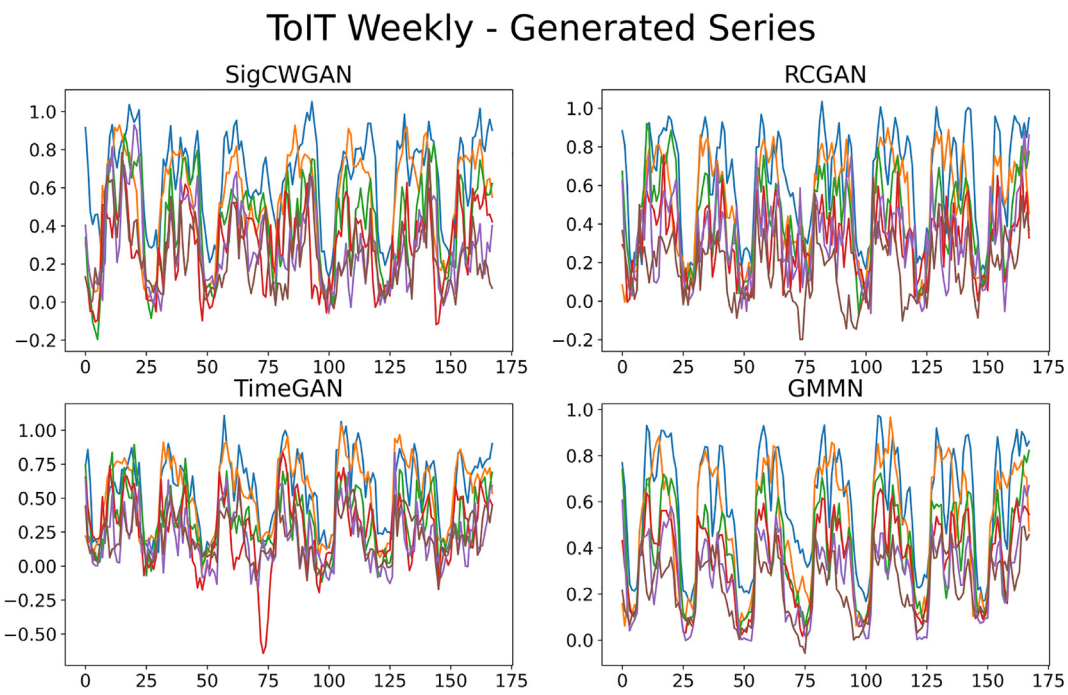

Fig. 14. ToIT Weekly – Example of fake series generated by the four generators.

Table 4
Beijing – Summary of the results obtained in the comparison. For each experiment with the Beijing dataset, the metrics (wcs) are scaled in the $[0, 1]$ interval and then those belonging to the same time are averaged. The best result for each metric is reported in bold, while the second one is underlined.

Beijing	Innovation	Stat_Props	Distribution	Prediction
SigCWGAN	0.667	0.935	0.911	1.0
RCGAN	**0.668**	0.493	0.967	0.25
TimeGAN	0.644	0.427	0.926	0.236
GMMN	0.321	0.364	0.594	0.075
Ensemble	0.486	**0.0**	**0.415**	**0.011**

Table 5
Electricity – Summary of the results obtained in the comparison.

Electricity	Innovation	Stat_Props	Distribution	Prediction
SigCWGAN	0.593	0.71	0.9	0.839
RCGAN	**0.891**	0.677	0.687	0.708
TimeGAN	0.82	0.58	0.58	0.54
GMMN	0.0	0.446	**0.284**	0.029
Ensemble	0.292	**0.207**	0.46	**0.028**

Table 6
Finance – Summary of the results obtained in the comparison.

Finance	Innovation	Stat_Props	Distribution	Prediction
SigCWGAN	**1.0**	0.893	0.767	0.909
RCGAN	0.316	0.371	0.876	0.517
TimeGAN	0.357	0.318	0.832	0.785
GMMN	0.0	0.329	0.753	0.236
Ensemble	0.272	**0.001**	**0.479**	**0.026**

Table 7
Noval – Summary of the results obtained in the comparison.

Noval	Innovation	Stat_Props	Distribution	Prediction
SigCWGAN	0.812	1.0	1.0	1.0
RCGAN	0.255	0.13	**0.079**	0.171
TimeGAN	0.346	0.198	0.107	0.388
GMMN	0.0	0.076	0.109	**0.088**
Ensemble	**0.913**	**0.043**	0.512	0.312

worthy fake samples, without any doubt the best among GAN-based methods. In contrast, RCGAN obtains the worst performances. Regarding the yearly experiment, it also shows a predominance of Ensemble, which keeps an extraordinary low value for KL. The second-best model is the GMMN, even if it is far away from the Ensemble. In the context of GAN methods, SigCWGAN seems to be the best, while the performances of RCGAN and TimeGAN are very close to each other. Furthermore, SigCWGAN achieves the best Innovation score. Also by considering the 4Yearly experiment, we find the Ensemble to be the best method, with very low KL. Behind it, there are SigCWGAN and GMMN, with very close performances. Furthermore, TimeGAN exhibits better performances than RCGAN. Also, SIgCWGAN is still the best method regarding Innovation.

Regarding the Electricity dataset (summarized in Table 5) and Monthly experiment, in this case, GMMN shows to be the best method. Good results are achieved also by Ensemble and TimeGAN, even if their KL metric is infinity. Moreover, the SigCWGAN performances are quite bad, worst than the other methods. As for the BiMonthly experiment, the Ensemble shows the best results, followed by GMMN. Particularly bad are the KL value obtained by GMMN and SigCWGAN, which, in contrast, exhibits the best Innovation value. Among GAN models, TimeGAN shows slightly better performances. Regarding the Quarterly experiment, Ensemble is still the best method, even if GMMN obtains very close performances. Also, SigCWGAN and GMMN suffer big error on KL, while Ensemble's value for this metric is incredibly low. Among the

Table 8
ToIT – Summary of the results obtained in the comparison.

ToIT	Innovation	Stat_Props	Distribution	Prediction
SigCWGAN	0.412	0.565	0.793	0.473
RCGAN	0.969	0.462	0.628	1.0
TimeGAN	**0.981**	0.404	0.628	0.807
GMMN	0.0	0.474	0.517	0.228
Ensemble	0.191	**0.001**	**0.255**	**0.0**

GANs, SigCWGAN and TimeGAN are the best, with very close values.

In the Finance dataset, whose results are summarized in Table 6, Ensemble is always the best algorithm. In the Monthly dataset, it obtain the best results, followed very closely by GMMN and SigCWGAN. Among the GANs methods, SigCWGAN seems to be the best, even thanks to its Innovation value which is the highest one. Furthermore, the GMMN result in KL is very unsatisfying. Regarding the Quarterly experiment, Ensemble is still the best method, far away from all the other ones, which exhibit performances very close to each other. Only regarding the Innovation, SigCWGAN achieves a very good result. Instead, its performance regarding CC is certainly the worst one. As for the Yearly experiment, Ensemble shows the best performances, followed by SigCWGAN and GMMN. In particular, the Innovation score for SigCWGAN is without any doubt the best one, but its KL and CC are the worst ones. Instead, RCGAN and TimeGAN have similar performances which are not affected by such outlier results.

As for the Noval dataset, Table 7 summarizes the results. In the Weekly experiment, there is a predominance of GMMN which achieves performances significantly better than all the other competitors. Behind it, there are RCGAN and TimeGAN with very close values. Furthermore, bad performances are achieved by SigCWGAN, especially regarding KL, even if its Innovation is the best one. A similar situation is obtained in the BiWeekly experiment, with the only difference that the predominance of GMMN vanishes and there is no best algorithm as all of them, except SigCWGAN, achieve very close values. Furthermore, not only SigCWGAN but also GMMN KL is abnormally high.

The results concerning the ToIT dataset are reported in Table 8. Regarding the Daily experiment, there is a strong predominance of Ensemble, followed far away by TimeGAN and RCGAN, which show performances very close to each other. Looking KL metric, there are two noteworthy observations: from one side, Ensemble achieves a very low value; on the other side, SigCWGAN result is abnormally high. Also, the results obtained by SigCWGAN on the other metrics are very bad. As for the Weekly experiment, Ensemble and SigCWGAN achieve the best results and their performances are very close to each other. In particular, both of them obtain very low values for KL.

In general, we can make several observations about the behaviour of the proposed generative models. Firstly, it seems that SigCWGAN often achieves the best Innovation score, but it also exhibits frequent big negative values. On the other side, the Ensemble obtains almost every bad Innovation value, but it is the most stable one, with low values on almost all the metrics. Furthermore, very often the results are significantly lower than the other competitors, showing its capability to overcome the performances obtained by the simple methods. In other words, the results obtained by the Ensemble prove that, by mixing the samples generated by different models, we are able to obtain a dataset capable of accurately mimicking the behaviour and the statistical properties of the original data. Finally, it is noteworthy to emphasize that also GMMN often obtains good performances and sometimes it is able to overcome GAN-based approaches.

5.5. Bounded experiments

As already stated, in all our experiments the real samples are scaled in the $[0, 1]$ interval. However, generated series often contain values outside this range. As our purpose is to generate fake samples starting from original data, in a certain way, in our experiments we work in-the-sample and not out-of-sample. This means we are authorized to bound the generated series in the same interval as the original data. This could enhance the generative power of the models that is, it can be useful for making the approximate distribution p_{fake} closer to the original p_{data}. However, although it is fair from a theoretical point of view, in practical applications it could jeopardize privacy, by creating a new dataset within the same boundaries as the previous one. So, we evaluate the proposed algorithms in two versions: firstly, we evaluate the generated samples as they are; then, we evaluate the bounded version of the generated samples. In this way, we aim to provide a deeper overview of the generative power reached by each generative model.

The tables containing all the results are provided in the Supplementary Material. Instead, Table 9 contains a summary of the performances achieved by the generative models, aggregated in the same way as described before. We can appreciate how there are no big differences in the relative performances of the algorithm. Indeed, there is neither a big difference in the absolute performances, as proved by Table 10 which shows the percentage increment of metrics when shifting to the bounded version of fake samples.

5.6. Computational time

Now, we report a graphical analysis of the computational time, both for training the generative method and for generating the fake samples. The computational time for each algorithm is computed starting from the best hyperparameters vector (as described in the previous subsection) and by training across 5000 epochs. The results obtained across the 25 seeds are averaged. The numerical

Table 9
Summary of the results obtained in the bounded experiments. For each experiment within a fixed dataset, the metrics (wcs) are scaled in the $[0, 1]$ interval and then those belonging to the same type are averaged. The best result for each metric is reported in bold, while the second one is underlined.

		Sig	RCGAN	Time	GMMN	Ens
Beijing	**Innovation**	**0.992**	0.937	0.835	0.0	0.173
	Stat_Props	0.88	0.767	0.65	0.653	**0.0**
	Distribution	0.98	0.953	0.889	0.6	**0.421**
	Prediction	1.0	0.56	0.506	0.207	**0.026**
Electricity	**Innovation**	0.601	**0.894**	0.704	0.0	0.223
	Stat_Props	0.634	0.748	0.658	0.511	**0.323**
	Distribution	0.896	0.684	0.501	**0.318**	0.445
	Prediction	0.836	0.744	0.532	0.293	**0.016**
Finance	**Innovation**	**1.0**	0.353	0.521	0.0	0.342
	Stat_Props	0.753	0.526	0.444	0.436	**0.0**
	Distribution	0.896	0.814	0.802	0.694	**0.453**
	Prediction	0.999	0.437	0.734	0.244	**0.041**
Noval	**Innovation**	0.454	0.368	0.219	0.0	**1.0**
	Stat_Props	0.899	0.252	0.189	0.166	**0.008**
	Distribution	0.974	**0.078**	0.214	0.162	0.593
	Prediction	1.0	0.338	0.222	**0.154**	0.34
ToIT	**Innovation**	0.361	**1.0**	0.926	0.0	0.139
	Stat_Props	0.594	0.461	0.355	0.472	**0.001**
	Distribution	0.776	0.616	0.604	0.509	**0.241**
	Prediction	0.353	1.0	0.861	0.255	**0.0**

Table 10
Percentage increment of metrics values when bounding the fake samples in the $[0, 1]$ interval. Each result is obtained as the average percentage increment among all the metrics and granularities of a given dataset and metric type.

		Sig %	RCGAN %	Time %	GMMN %	Ens %
Beijing	**Innovation**	0	0	0	0	7
	Stat_Props	0	0	0	0	5
	Distribution	0	0	0	0	0
	Prediction	0	0	0	0	3
Electricity	**Innovation**	0	0	0	0	−2
	Stat_Props	0	nan	nan	0	nan
	Distribution	0	0	0	0	0
	Prediction	0	0	0	0	−1
Finance	**Innovation**	0	0	0	0	2
	Stat_Props	0	0	0	0	8
	Distribution	0	0	0	0	0
	Prediction	0	0	0	0	0
Noval	**Innovation**	0	0	0	0	1
	Stat_Props	−1	0	0	0	8
	Distribution	0	0	0	0	0
	Prediction	0	0	0	0	−2
TolT	**Innovation**	0	0	0	0	2
	Stat_Props	0	0	0	0	0
	Distribution	0	0	0	0	0
	Prediction	0	0	0	0	0

results are reported in the Supplementary Material. Instead, Figs. 15 and 16 show a graphical representation of the computational time.

The results of the 13 experiments are plotted together. The bars referred to the same experiment are clustered together and they are normalized by dividing the true value by the maximum in the considered experiment. In such a way, all the experiments are in the same range, the figures are more easily readable and the comparison between algorithms is straightforward. The maximum time for each experiment is reported on the x-axis label.

The first consideration is that SigCWGAN is the most expensive algorithm. This is probably due to the computation of the path signature which, according to the exploited hyperparameters, requires the solution of linear systems of dimension in the scale of d^2 and d^3.

In contrast, TimeGAN often exhibits, among GAN-based methods, the lowest computational time. This is justifiable by taking into account that it works in the low-dimensional latent space obtained from the AE, so reducing problem dimensionality it is able to reduce also computational costs. However, also training the AE has a cost, which in some experiment is not fully compensated by a significant drop in the execution time leading to higher computational time than the RCGAN.

Finally, we can appreciate the low computational time for GMMN, which is often the cheapest model. This is probably due to the structure of GMMN, which does not involve adversarial training and so strongly reduces the cost of each epoch. However, in the generation step, this advantage disappears (in fact, only the Generator plays a role in the generation step, while the Discriminator is not involved) and no strong patterns are noticeable.

5.7. Temporal evolution of the solution

In the Supplementary Material, we show the plots referred to the number of epochs during the training versus the evaluation

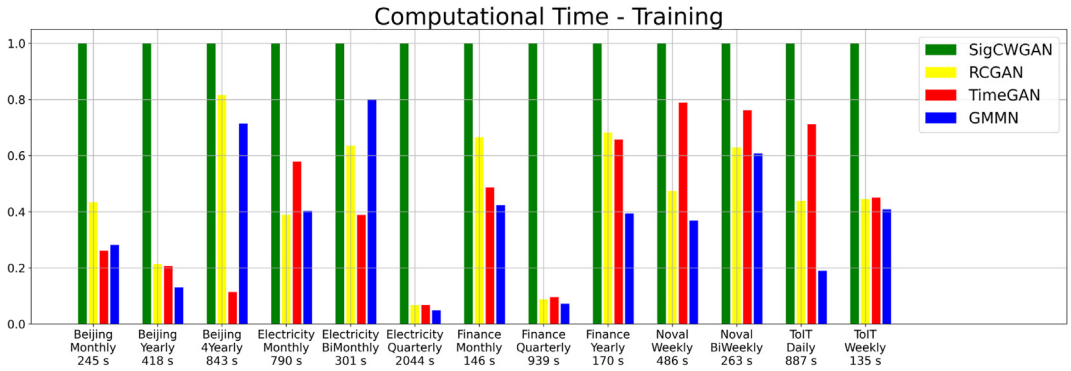

Fig. 15. Comparison of the computational time for training the generative model. Each cluster represents one of the 13 experiments we carried out. Each bar represents an algorithm, namely the green bar (the first of each cluster) is for SigCWGAN; the second bar of each cluster i.e., the yellow one, is for RCGAN; the third, red bar is for TimeGAN; the last, blue bar represents GMMN. Within each experiment, the bars are scaled by dividing the true value by the maximum time of the cluster. In this way, all the experiments are on the same scale, which makes the plot easily understandable. For clarity purposes, the maximum time within each cluster is reported in the x-axis label, under the name of the experiment. In this way, the reader can have an idea of the time required in each situation.

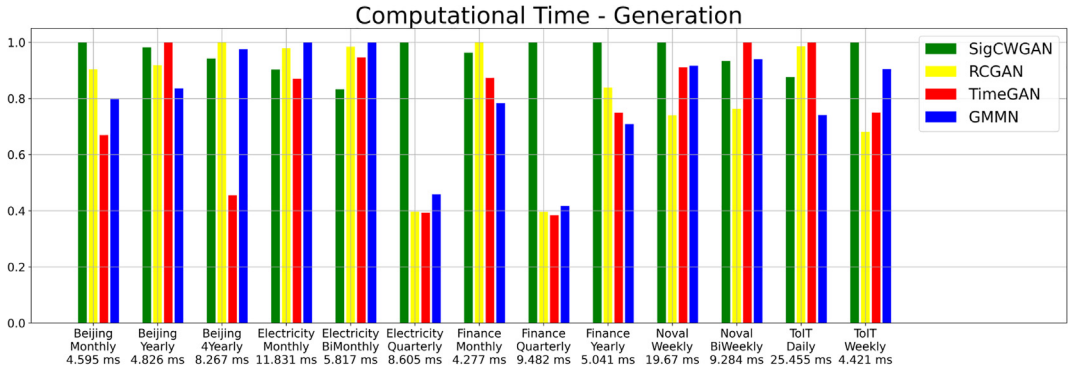

Fig. 16. The computational time required for the generation of fake samples.

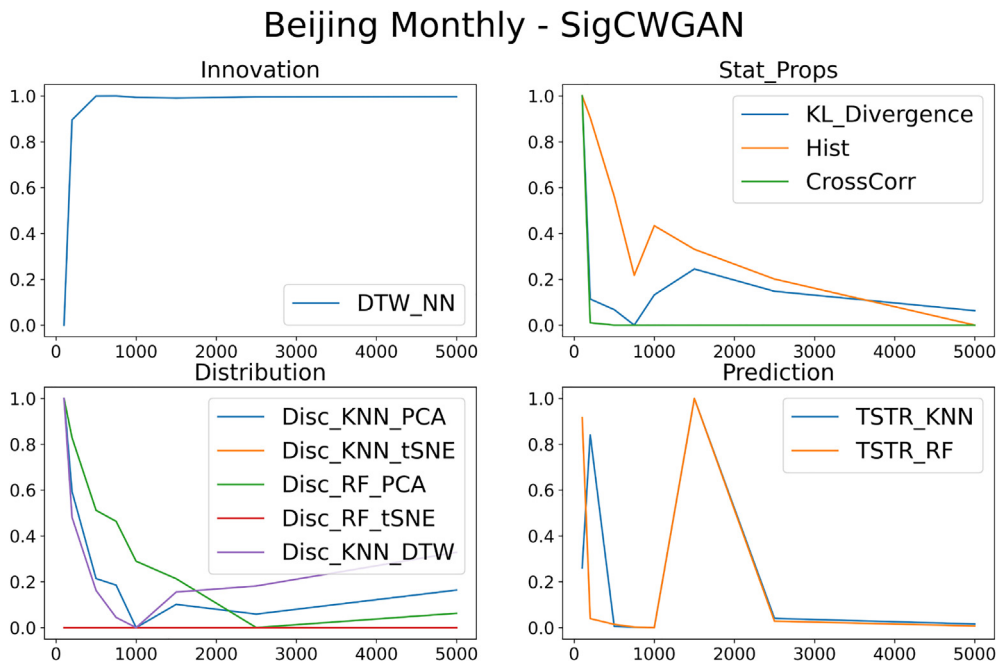

Fig. 17. Evaluation metrics of SigCWGAN for the Beijing dataset in the monthly experiment. The metrics are grouped according to their type (top left for Innovation, top right for Statistical Properties, bottom left for Distribution and bottom right for Prediction). From the Figure, it is possible to appreciate how SigCWGAN performances degrade if the training is on a too big number of epochs.

metrics. The plots are concerned with the best hyperparameters vector for each experiment, as described in the previous Subsection. The metrics are grouped according to their type, in order to provide a clearer plot. There are no noticeable patterns. The only relevant thing to observe is that the performances of SigCWGAN seem to worst as the number of epochs increases, as shown in Fig. 17 which provides an example referred to the Beijing dataset in the Monthly experiment. Similar plots are obtained also for the other experiments, as reported in the Supplementary Material. This suggests that SigCWGAN usually requires a smaller number of epochs for proper training, unlike the other algorithms which often need more epochs.

A final summary of the temporal evolution of the solution is reported in the Tables 11–14. From these tables, it is possible to observe the behavior of SigCWGAN, which often exhibits the best values after a low number of epochs.

6. What's next

After this huge experimental stage, we have to discuss about the current state-of-the-art in deep generative models for time series and about what we can expect in the near future.

6.1. State of the art

Firstly, we would like to provide a short snapshot of the current state-of-the-art in generative approaches for time series. Currently,

Table 11
Temporal evolution of the results related to the Innovation metric. The results are aggregated as explained above.

Innovation		100	200	500	750	1000	1500	2500	5000
Beijing	**SigCWGAN**	0.335	0.796	0.757	0.4	0.698	0.693	0.663	0.61
	RCGAN	1.0	0.638	0.527	0.318	0.142	0.017	0.029	0.003
	TimeGAN	0.912	0.495	0.722	0.184	0.147	0.05	0.048	0.0
	GMMN	1.0	0.537	0.141	0.086	0.066	0.041	0.014	0.001
Electricity	**SigCWGAN**	0.974	0.795	0.414	0.404	0.354	0.366	0.264	0.295
	RCGAN	0.52	0.81	0.368	0.357	0.27	0.119	0.024	0.007
	TimeGAN	0.892	0.668	0.257	0.338	0.258	0.247	0.179	0.075
	GMMN	1.0	0.812	0.539	0.413	0.321	0.236	0.139	0.0
Finance	**SigCWGAN**	0.378	0.313	0.448	0.611	0.725	0.69	0.653	0.618
	RCGAN	0.952	0.889	0.536	0.401	0.588	0.555	0.624	0.603
	TimeGAN	0.615	0.578	0.589	0.284	0.376	0.302	0.4	0.317
	GMMN	1.0	0.562	0.201	0.129	0.094	0.042	0.165	0.588
Noval	**SigCWGAN**	1.0	0.767	0.208	0.14	0.026	0.002	0.027	0.178
	RCGAN	0.72	1.0	0.675	0.382	0.273	0.132	0.084	0.0
	TimeGAN	0.766	0.622	0.583	0.582	0.591	0.243	0.071	0.0
	GMMN	1.0	0.562	0.16	0.141	0.119	0.078	0.07	0.0
ToIT	**SigCWGAN**	0.672	0.32	0.646	0.689	0.543	0.534	0.495	0.503
	RCGAN	0.5	0.693	0.768	0.815	0.67	0.578	0.416	0.313
	TimeGAN	0.307	0.342	0.777	0.927	0.551	0.612	0.336	0.083
	GMMN	1.0	0.761	0.435	0.295	0.256	0.163	0.076	0.0

Table 12
The table reports the temporal evolution of the metrics in the Statistical Properties group.

Stat_Props		100	200	500	750	1000	1500	2500	5000
Beijing	**SigCWGAN**	1.0	0.358	0.148	0.257	0.156	0.1	0.088	0.056
	RCGAN	0.925	0.546	0.193	0.193	0.098	0.049	0.032	0.001
	TimeGAN	0.972	0.499	0.333	0.132	0.084	0.048	0.037	0.004
	GMMN	0.946	0.413	0.303	0.23	0.205	0.179	0.119	0.015
Electricity	**SigCWGAN**	0.804	0.33	0.266	0.205	0.194	0.218	0.276	0.337
	RCGAN	0.745	0.696	0.582	0.425	0.363	0.265	0.242	0.23
	TimeGAN	0.907	0.616	0.454	0.395	0.324	0.283	0.292	0.235
	GMMN	0.876	0.602	0.465	0.273	0.318	0.391	0.335	0.152
Finance	**SigCWGAN**	0.703	0.203	0.329	0.468	0.53	0.352	0.468	0.519
	RCGAN	0.654	0.496	0.406	0.33	0.187	0.073	0.149	0.042
	TimeGAN	0.66	0.631	0.332	0.185	0.203	0.259	0.097	0.031
	GMMN	0.882	0.561	0.58	0.491	0.256	0.278	0.286	0.081
Noval	**SigCWGAN**	0.341	0.181	0.301	0.335	0.407	0.491	0.584	0.696
	RCGAN	1.0	0.46	0.242	0.143	0.126	0.066	0.068	0.019
	TimeGAN	0.881	0.619	0.2	0.215	0.157	0.067	0.024	0.008
	GMMN	0.903	0.665	0.264	0.193	0.132	0.033	0.013	0.058
ToIT	**SigCWGAN**	1.0	0.606	0.237	0.229	0.166	0.133	0.039	0.005
	RCGAN	0.937	0.487	0.391	0.226	0.235	0.223	0.043	0.039
	TimeGAN	0.878	0.697	0.505	0.431	0.423	0.245	0.104	0.005
	GMMN	0.813	0.611	0.437	0.439	0.456	0.327	0.231	0.22

Table 13
Behavior of the Distribution metrics with respect to the number of training epochs.

Distribution		100	200	500	750	1000	1500	2500	5000
Beijing	**SigCWGAN**	0.933	0.781	0.58	0.579	0.541	0.562	0.527	0.55
	RCGAN	0.967	0.982	0.893	0.833	0.918	0.899	0.596	0.4
	TimeGAN	0.986	0.981	0.82	0.943	0.93	0.774	0.563	0.404
	GMMN	0.993	0.989	0.909	0.79	0.738	0.63	0.514	0.401
Electricity	**SigCWGAN**	0.946	0.737	0.522	0.526	0.262	0.435	0.272	0.365
	RCGAN	0.944	0.892	0.739	0.725	0.696	0.448	0.303	0.27
	TimeGAN	0.966	0.656	0.612	0.654	0.532	0.491	0.329	0.278
	GMMN	0.95	0.716	0.597	0.509	0.523	0.443	0.184	0.275
Finance	**SigCWGAN**	0.999	0.838	0.682	0.59	0.531	0.461	0.436	0.406
	RCGAN	0.976	0.981	0.724	0.702	0.753	0.629	0.586	0.424
	TimeGAN	0.966	0.998	0.868	0.789	0.719	0.503	0.486	0.476
	GMMN	0.988	0.955	0.897	0.886	0.799	0.719	0.656	0.403
Noval	**SigCWGAN**	0.793	0.473	0.402	0.298	0.468	0.338	0.398	0.294
	RCGAN	0.858	0.642	0.403	0.189	0.22	0.064	0.242	0.155
	TimeGAN	0.995	0.904	0.565	0.4	0.354	0.208	0.143	0.039
	GMMN	0.89	0.664	0.331	0.198	0.127	0.092	0.123	0.129
ToIT	**SigCWGAN**	0.749	0.603	0.738	0.696	0.659	0.56	0.517	0.513
	RCGAN	0.906	0.777	0.759	0.775	0.723	0.615	0.346	0.284
	TimeGAN	0.773	0.925	0.789	0.749	0.611	0.604	0.488	0.225
	GMMN	0.959	0.926	0.758	0.626	0.541	0.373	0.4	0.265

the literature about deep generative models for time series is decisively less developed than that related to static data. However, in the last years, this field has gained new attention as several works have been published and different methodologies have been proposed. However, most of these works are strongly task-dependent and integrated into particular frameworks. In other words, a huge number of the existing deep generative approaches for time series are part of complex pipelines and so they are not available as standalone models. Furthermore, even among the pure generative models, often the code has not been released from the authors and so it is not publicly available. Clearly, this is a strong limitation for the research and application of the existing methodologies to real-world problems.

6.2. Challenges

Currently, generative models for time series generation exhibit several challenges which need to be addressed. Maybe the major issue is linked to the evaluation metric. In fact, no standard criteria exist to evaluate the goodness of fake samples. Even in our comparison, we exploit a number of metrics. In fact, there aren't standard, global, universally accepted metrics but just a plethora of local-based criteria which are designed to work for a specific task by catching just a single aspect of the new dataset. For example, the metrics based on statistical properties just describe the behavior related to a particular property, for example, the histogram, without taking into account other properties (such as the cross-correlation) or other aspects of the new dataset (such as the usability of learned fake patterns in the real data).

Another big challenge when working with deep generative models for time series is the absence of a universally accepted distance function between time series. Actually, this is a problem that is present in different aspects of time series study, such as clustering. In general, it is one of the main reasons behind the fact that not all the techniques used when working with static data can be used also for time series.

Table 14
Results, in terms of Prediction metrics, as the number of training epochs increases.

Prediction		100	200	500	750	1000	1500	2500	5000
Beijing	**SigCWGAN**	0.863	0.428	0.138	0.137	0.078	0.391	0.031	0.004
	RCGAN	1.0	0.568	0.338	0.247	0.136	0.064	0.04	0.0
	TimeGAN	1.0	0.614	0.561	0.229	0.16	0.089	0.059	0.0
	GMMN	1.0	0.419	0.207	0.14	0.109	0.088	0.05	0.0
Electricity	**SigCWGAN**	0.994	0.79	0.487	0.294	0.208	0.119	0.03	0.021
	RCGAN	0.742	0.962	0.595	0.471	0.224	0.113	0.051	0.0
	TimeGAN	0.777	0.771	0.817	0.395	0.315	0.266	0.139	0.0
	GMMN	1.0	0.658	0.418	0.327	0.291	0.203	0.078	0.0
Finance	**SigCWGAN**	0.999	0.78	0.354	0.227	0.212	0.148	0.024	0.007
	RCGAN	0.874	0.61	0.436	0.208	0.165	0.071	0.095	0.0
	TimeGAN	0.71	0.688	0.335	0.179	0.269	0.146	0.209	0.009
	GMMN	1.0	0.484	0.259	0.158	0.102	0.041	0.038	0.005
Noval	**SigCWGAN**	0.916	0.741	0.637	0.225	0.091	0.06	0.007	0.109
	RCGAN	0.75	0.777	0.847	0.361	0.235	0.135	0.126	0.0
	TimeGAN	0.805	0.624	0.699	0.858	0.761	0.22	0.057	0.085
	GMMN	0.56	0.851	0.665	0.523	0.445	0.168	0.0	0.114
ToIT	**SigCWGAN**	1.0	0.659	0.323	0.245	0.172	0.104	0.021	0.0
	RCGAN	1.0	0.564	0.325	0.166	0.177	0.084	0.019	0.04
	TimeGAN	0.927	0.841	0.554	0.534	0.387	0.256	0.169	0.0
	GMMN	1.0	0.607	0.289	0.14	0.121	0.084	0.044	0.0

Finally, another challenge in the generation of time series is due to GAN instability. In fact, the majority of generative approaches for time series are based on GAN. Unfortunately, this type of approach exhibit a significant instability in the training process, which can expose the models to issues such as non-convergence or vanishing gradients.

6.3. Future directions

In the future, several new contributions are expected in different fields. For example, a deeper analysis of ensemble techniques need to be discussed. In this work, we use a simple ensemble made up of an equal number of fake samples generated by the different methods. Although its simplicity, this strategy shows excellent result and often it overcomes the results obtained by its components. So, it can be noteworthy to further investigate this field by searching for more convenient ensembling or hybrid models strategies.

Another direction that can be interesting is related to the study and implementation of other types of models for generation. In other words, as already pointed out, the majority of the existing strategies relies on adversarial training. However, other approaches like Transformers are thought of as being able to give a contribution to the generative task, while alleviating GAN issues.

Finally, also the study and further analysis of the metrics for fake series evaluation could be of primary importance in the near future. As already pointed out, almost all the current metrics are extremely task-dependent. So, designing new criteria could be of great assistance for researcher, so it is noteworthy to further investigation.

7. Conclusion

This paper analyses the contribution given by different state-of-the-art neural network-based techniques for time series generation. In contrast to static data, time series generation is a more challenging task still little studied in the literature. Nonetheless, this task has several practical applications, such as data augmentation or anomaly detection, making it a noteworthy field of study. In particular, dealing with the temporal dimension raises several difficulties in designing optimal generator architecture. Furthermore, the evaluation of fake samples is an open issue that adds a complexity layer to the problem. In fact, no globally accepted metrics have been proposed. All the proposed measures are extremely task-dependent, and they provide only a local interpretation of the generation quality. Even if in this work we summarize and categorize a considerable number of different metrics trying to offer different perspectives of the fake dataset, no new metrics able to provide global information have been proposed. This is, without any doubt, one of the most important challenges for future works on time series generation.

Among the neural network-based models for time series generation proposed in the literature, the most significant part is concerned with adversarial training, even if other architectures have been proposed, particularly GMMN. Almost all these models handle time series by exploiting particular neural architectures which are able to catch temporal patterns in the data. The most commonly used are RNN, LSTM and AR-FNN. Furthermore, also few attempts to exploit ensemble methodologies have been made. We also focus on the ensemble by creating a simple, equally-balanced one in this work. The results obtained in the experimental part show this technique can make an important contribution, and it can significantly improve the performances of generative models. This is, at least in our opinion, the most interesting finding and the most attractive direction for further investigation. In fact, although a simple ensemble has been proposed in this work, more complex ones can be built in order to enhance the results obtained by giving more weight to the best-performing models.

To assess the quality of the generative models proposed, several experiments have been carried out by exploiting different datasets and time granularities. In this way, we aim to catch different aspects of the generative models which can help in an optimal choice for practical applications. For example, it turns out the SigCWGAN is, in general, the most innovative one, that is, the one whose fake samples are the most distant from the original ones. We also find that GMMN is often capable of excellent generations, even if often it is affected by strong negative performances. Finally, the ensemble provides the most impressive fake samples, being able not only to average but also to overcome the results of the single generators strongly.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work is also supported by the project "4I: mixed reality, machine learning, gamification and educational for Industry", Prog. n. F/190130/01-03/X44, Fondo per la Crescita Sostenibile – Sportello "Fabbrica Intelligente" PON I & C 2014–2020, CUP: B66G21000040005 COR: 4641138. This work is also supported by the Natural Science Foundation of China, Grant No.: 11602235.

Appendix A. List of Notations

- AE → AutoEncoder
- AI → Artificial Intelligence
- AR-FNN → AutoRegressive-Feedforward Neural Network
- C-Sig-W_1 → Conditional Signature Wasserstein-1
- CC → Cross-Correlation
- CV → Cross Validation
- D → Discriminator
- DTW → Dynamic Time Warping
- NN → DTW Nearest Neighbour
- G → Generator
- GAN → Generative Adversarial Networks
- GMMN → Generative Moment Matching Networks
- Hi → Histogram
- KD → Discriminator with KNN and DTW
- KL → Kullback–Leibler Divergence
- KNN → K-Nearest Neighbours
- KP → Discriminator with KNN and PCA
- Kt → Discriminator with KNN and tSNE
- LSTM → Long Short-Term Memory
- MMD → Maximum Mean Discrepancy
- PCA → Principal Component Analysis
- RCGAN → Recurrent Conditional GAN
- RF → Random Forest
- RNN → Recurrent Neural Network
- RP → Discriminator with RF and PCA
- Rt → Discriminator with RF and tSNE
- SigCWGAN → Conditional Sig-Wasserstein GAN
- TimeGAN → Time GAN
- TK → Train Synthetic Test Real KNN
- TR → Train Synthetic Test Real RF
- tSNE → t-Distributed Stochastic Neighbor Embedding
- wcs → worst-case scenario

Appendix B. Supplementary data

Supplementary data associated with this article can be found, in the online version, athttps://doi.org/10.1016/j.jksuci.2022.07.010.

References

Altaheri, H., Muhammad, G., Alsulaiman, M., Amin, S.U., Altuwaijri, G.A., Abdul, W., Bencherif, M.A., Faisal, M., 2021. Deep learning techniques for classification of electroencephalogram (eeg) motor imagery (mi) signals: a review. Neural Comput. Appl., 1–42

Arroyo, J., Espínola, R., Maté, C., 2011. Different approaches to forecast interval time series: a comparison in finance. Comput. Econ. 37 (2), 169–191.

Bovenzi, G., Foggia, A., Santella, S., Testa, A., Persico, V., Pescapé, A., 2022. Data poisoning attacks against autoencoder-based anomaly detection models: a robustness analysis. In: 2022 IEEE International Conference on Communications (ICC).

Breiman, L., 2001. Random forests. Mach. Learn. 45 (1), 5–32.

Brophy, E., Wang, Z., She, Q., Ward, T., 2021. Generative adversarial networks in time series: A survey and taxonomy. arXiv preprint arXiv:2107.11098.

Che, Z., Purushotham, S., Li, G., Jiang, B., Liu, Y., 2018. Hierarchical deep generative models for multi-rate multivariate time series. In: International Conference on Machine Learning. PMLR, pp. 784–793.

Debnath, A., Waghmare, G., Wadhwa, H., Asthana, S., Arora, A., 2021. Exploring generative data augmentation in multivariate time series forecasting: Opportunities and challenges. Solar-Energy 137, 52–560.

Deng, G., Han, C., Dreossi, T., Lee, C., Matteson, D.S., 2022. Ib-gan: A unified approach for multivariate time series classification under class imbalance. In: Proceedings of the 2022 SIAM International Conference on Data Mining (SDM). SIAM, pp. 217–225.

Esteban, C., Hyland, S.L., Rätsch, G., 2017. Real-valued (medical) time series generation with recurrent conditional gans. arXiv preprint arXiv:1706.02633.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y., 2014. Generative adversarial nets. Adv. Neural Inf. Process. Syst. 27.

Harshvardhan, G., Gourisaria, M.K., Pandey, M., Rautaray, S.S., 2020. A comprehensive survey and analysis of generative models in machine learning. Comput. Sci. Rev. 38, 100285.

Hazra, D., Byun, Y.-C., 2020. Synsiggan: Generative adversarial networks for synthetic biomedical signal generation. Biology 9 (12), 441.

Iwana, B.K., Uchida, S., 2021. An empirical survey of data augmentation for time series classification with neural networks. Plos one 16, (7) e0254841.

Jeha, P., Bohlke-Schneider, M., Mercado, P., Kapoor, S., Nirwan, R.S., Flunkert, V., Gasthaus, J., Januschowski, T., 2021. Psa-gan: Progressive self attention gans for synthetic time series. In: International Conference on Learning Representations.

Kiyasseh, D., Tadesse, G.A., Thwaites, L., Zhu, T., Clifton, D., et al., 2020. Plethaugment: Gan-based ppg augmentation for medical diagnosis in low-resource settings. IEEE J. Biomed. Health Inf. 24 (11), 3226–3235.

Kullback, S., Leibler, R.A., 1951. On information and sufficiency. Ann. Math. Stat. 22 (1), 79–86.

Le Guennec, A., Malinowski, S., Tavenard, R., 2016. Data augmentation for time series classification using convolutional neural networks. In: ECML/PKDD workshop on advanced analytics and learning on temporal data.

Li, D., Chen, D., Goh, J., Ng, S.-K., 2018. Anomaly detection with generative adversarial networks for multivariate time series. arXiv preprint arXiv:1809.04758.

Li, Y., Shi, Z., Liu, C., Tian, W., Kong, Z., Williams, C.B., 2021. Augmented time regularized generative adversarial network (atr-gan) for data augmentation in online process anomaly detection. IEEE Trans. Autom. Sci. Eng.

Li, Y., Swersky, K., Zemel, R., 2015. Generative moment matching networks. In: International conference on machine learning. PMLR, pp. 1718–1727.

Li, Z., Ma, C., Shi, X., Zhang, D., Li, W., Wu, L., 2021. Tsa-gan: A robust generative adversarial networks for time series augmentation. In: 2021 International Joint Conference on Neural Networks (IJCNN). IEEE, pp. 1–8.

Liu, C., Zhou, H., Sun, Z., Cui, G., 2021. Glowimp: combining glow and gan for multivariate time series imputation. In: International Conference on Algorithms and Architectures for Parallel Processing. Springer, pp. 50–64.

Lu, Y., Lu, J., 2020. A universal approximation theorem of deep neural networks for expressing probability distributions. Advances in neural information processing systems 33, 3094–3105.

Machiwal, D., Jha, M.K., 2006. Time series analysis of hydrologic data for water resources planning and management: a review. J. Hydrol. Hydromech. 54 (3), 237–257.

Muhammad, G., Alshehri, F., Karray, F., El Saddik, A., Alsulaiman, M., Falk, T.H., 2021. A comprehensive survey on multimodal medical signals fusion for smart healthcare systems. Inf. Fusion 76, 355–375.

Ni, H., Szpruch, L., Wiese, M., Liao, S., Xiao, B., 2020. Conditional sig-wasserstein gans for time series generation. arXiv preprint arXiv:2006.05421.

Oussidi, A., Elhassouny, A., 2018. Deep generative models: Survey. In: 2018 International Conference on Intelligent Systems and Computer Vision (ISCV). IEEE, pp. 1–8.

Shen, Z., Zhang, Y., Lu, J., Xu, J., Xiao, G., 2018. Seriesnet: a generative time series forecasting model. In: 2018 International Joint Conference on Neural Networks (IJCNN). IEEE, pp. 1–8.

Shiota, S., Takamichi, S., Matsui, T., 2018. Data augmentation with moment-matching networks for i-vector based speaker verification. In: 2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC). IEEE, pp. 345–349.

Srinivasan, P., Knottenbelt, W.J., 2022. Time-series transformer generative adversarial networks. arXiv preprint arXiv:2205.11164.

Sumiya, Y., Horie, K., Shiokawa, H., Kitagawa, H., 2019. Nr-gan: Noise reduction gan for mice electroencephalogram signals. In: Proceedings of the 2019 4th International Conference on Biomedical Imaging, Signal Processing, pp. 94–101.

Sun, H., Deng, Z., Chen, H., Parkes, D.C., 2020. Decision-aware conditional gans for time series data. arXiv preprint arXiv:2009.12682.

Tu, J., Liu, H., Meng, F., Liu, M., Ding, R., 2018. Spatial-temporal data augmentation based on lstm autoencoder network for skeleton-based human action recognition. In: 2018 25th IEEE International Conference on Image Processing (ICIP). IEEE, pp. 3478–3482.

Um, T.T., Pfister, F.M., Pichler, D., Endo, S., Lang, M., Hirche, S., Fietzek, U., Kulić, D., 2017. Data augmentation of wearable sensor data for parkinson's disease monitoring using convolutional neural networks. In: Proceedings of the 19th ACM international conference on multimodal interaction, pp. 216–220.

Van der Maaten, L., Hinton, G., 2008. Visualizing data using t-sne. J. Mach. Learn. Res. 9 (11).
Wang, L., Zhang, W., He, X., 2019. Continuous patient-centric sequence generation via sequentially coupled adversarial learning. In: International Conference on Database Systems for Advanced Applications. Springer, pp. 36–52.
Wang, T., Trugman, D., Lin, Y., 2021. Seismogen: Seismic waveform synthesis using gan with application to seismic data augmentation. J. Geophys. Res.: Solid Earth 126 (4). e2020JB020077.
Wen, Q., Sun, L., Yang, F., Song, X., Gao, J., Wang, X., Xu, H., 2020. Time series data augmentation for deep learning: A survey. arXiv preprint arXiv:2002.12478.
Xie, L., Lin, K., Wang, S., Wang, F., Zhou, J., 2018. Differentially private generative adversarial network. arXiv preprint arXiv:1802.06739.
Xu, S., Marwah, M., Arlitt, M., Ramakrishnan, N., 2021. Stan: Synthetic network traffic generation with generative neural models. In: International Workshop on Deployable Machine Learning for Security Defense. Springer, pp. 3–29.
Yoon, J., Jarrett, D., Van der Schaar, M., 2019. Time-series generative adversarial networks. In: Advances in Neural Information Processing Systems 32.
Zhang, S., Guo, B., Dong, A., He, J., Xu, Z., Chen, S.X., 2017. Cautionary tales on air-quality improvement in beijing. Proc. R. Soc. A 473 (2205), 20170457.
Zhu, G., Zhao, H., Liu, H., Sun, H., 2019. A novel lstm-gan algorithm for time series anomaly detection. In: 2019 Prognostics and System Health Management Conference (PHM-Qingdao). IEEE, pp. 1–6.