sensors

MDPI

Systematic Review

Survey of Time Series Data Generation in IoT

Chaochen Hu [1,2], Zihan Sun [1,2], Chao Li [1,2,*], Yong Zhang [1,2,*] and Chunxiao Xing [1,2]

[1] Beijing National Research Center for Information Science and Technology, Tsinghua University, Beijing 100084, China; hcc20@mails.tsinghua.edu.cn (C.H.); sunzh22@mails.tsinghua.edu.cn (Z.S.); xingcx@tsinghua.edu.cn (C.X.)
[2] Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
* Correspondence: li-chao@tsinghua.edu.cn (C.L.); zhangyong05@tsinghua.edu.cn (Y.Z.); Tel.: +86-10-62788788 (C.L. & Y.Z.)

Abstract: Nowadays, with the rapid growth of the internet of things (IoT), massive amounts of time series data are being generated. Time series data play an important role in scientific and technological research for conducting experiments and studies to obtain solid and convincing results. However, due to privacy restrictions, limited access to time series data is always an obstacle. Moreover, the limited available open source data are often not suitable because of a small quantity and insufficient dimensionality and complexity. Therefore, time series data generation has become an imperative and promising solution. In this paper, we provide an overview of classical and state-of-the-art time series data generation methods in IoT. We classify the time series data generation methods into four major categories: rule-based methods, simulation-model-based methods, traditional machine-learning-based methods, and deep-learning-based methods. For each category, we first illustrate its characteristics and then describe the principles and mechanisms of the methods. Finally, we summarize the challenges and future directions of time series data generation in IoT. The systematic classification and evaluation will be a valuable reference for researchers in the time series data generation field.

Keywords: time series; data generation; categorization; IoT

check for updates

Citation: Hu, C.; Sun, Z.; Li, C.; Zhang, Y.; Xing, C. Survey of Time Series Data Generation in IoT. *Sensors* **2023**, *23*, 6976. https://doi.org/10.3390/s23156976

Received: 3 July 2023
Revised: 24 July 2023
Accepted: 26 July 2023
Published: 5 August 2023

1. Introduction

A time series is a form of data that records events or quantities occurring over time, usually indexed by timestamps. They can be sampled periodically or irregularly and cover various fields such as sensor readings, financial market quotes, weather forecasts, etc. Mathematically, a time series comprises a series of ordered data points, where each data point represents a value or state at a certain point in time. A time series can be expressed as a sequence, denoted as $(t_1, x_1), (t_2, x_2), \ldots, (t_n, x_n)$, where t_i represents the timestamp, which can be discrete or continuous, of the ith data point and x_i represents the value of the ith data point. A time series usually contains certain time correlations and regular characteristics of the sequence itself, such as periodicity, trend, seasonality, etc. These characteristics can be mined and analyzed through time series analysis, signal processing, machine learning and other methods.

Time series data generation refers to the use of specific methods and technologies to generate time series data that conform to specific rules or patterns, which can effectively solve these problems. As shown in Figure 1, a time series data generation method $TSG = (S, T, G)$ contains a model selector S, a model trainer T, and a generator G. The selector S takes prior knowledge $\mathcal{P}$ as input and selects a generative model M. The trainer T takes real time series data $\mathcal{T}$, metadata $\mathcal{M}$, and the selected model M as input, and obtains a trained model with parameters $M_\theta = T(\mathcal{T}, \mathcal{M}, M)$. The generator G, finally, generates new time series data $\mathcal{O}$ using the trained model M_θ and control information $\mathcal{C}$. The graphics made up of dashed lines represent optional components.

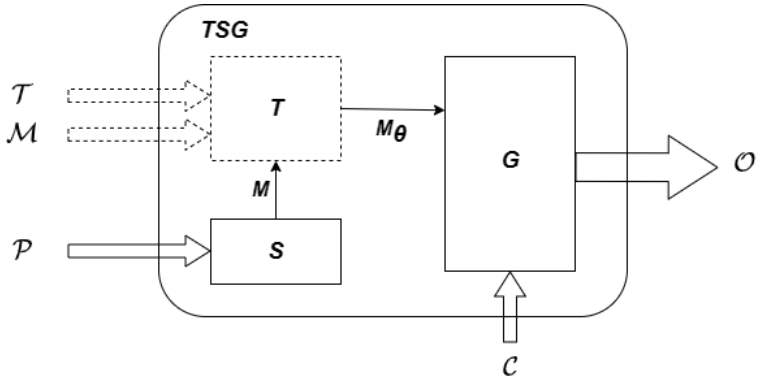

Figure 1. A formal model of time series data generation methods.

With the development of IoT technology [1] more and more sensors are being deployed in various fields, ranging from industrial manufacturing and transportation to medical care. This development results in the constant generation of a large amount of time series data. These time series data can come from various fields, including architecture [2,3], meteorology [4], finance [5], transportation [6], medical treatment [7], biomedical signals [8], environmental pollution [9], earthquake geology [10], etc. They reflect various phenomena and events that change over time. At the same time, the complexity of time series data is also increasing. The massive volume and complexity bring greater challenges to data analysis. To address this, various technologies are employed to manage [11], store [12], process, and analyze [13] time series data aiming to extract useful information from the data.

Time series data plays an important role in scientific and technological research. Through the analysis of time series data, it is possible to reveal the underlying patterns and laws in the data, discover the correlation and periodicity between events, and then deeply understand the nature and mechanism of the event itself, providing strong support for research in related disciplines. Specifically, a deep understanding of time trends [14], periodicity [15,16], correlation [17,18], etc., can be gained and valuable information can be further extracted, such as anomaly detection [19–21], classification [22–24], clustering [25,26], etc. These studies require a large amount of time series data for experiments to test the effectiveness and practicality of different algorithms and techniques, optimize the parameters and structure of algorithms, evaluate the performance and accuracy of different techniques, and train machine learning models.

However, there are two challenges in acquiring massive time series data. First, publicly obtainable time series data is limited because it may contain sensitive or confident information. For example, data from sensors may leak information such as location and temperature [27,28]. Second, due to the diversity of data sources, the instability of data collection, noise, etc., the quality of time series data is often relatively low, so it needs to be cleaned and verified. Low quality may bring various problems. For the problem of an unbalanced data distribution, some data sets may have unbalanced distribution problems, which will lead to a decrease in the performance of the training model. For the data diversity problem, datasets in some domains may lack diversity to cover all scenarios and situations.

Time series data generation can solve the two challenges above. First, by generating synthetic data, privacy is preserved and data sharing and analysis is allowed. At the same time, it is possible to expand the size of the dataset. Second, the data quality issues can be reduced or eliminated, resulting in more accurate and high-quality data. At the same time, the coverage of the data sets can be expanded, making the data more balanced and diverse, and improving the performance of the model. As a result, time series analysts, time series researchers, and time series processing system and database testing engineers will benefit from time series generation.

Researchers in many fields have proposed a variety of time series data generation methods in their respective fields such as biology [8], database benchmark [29,30], electricity [31], energy [32–39], environment [40–42], finance [5,43], medicine [7], music [44], networks [45], remote sensing [46–49] and sensors [50]. Despite the abundance of research on time series generation, a comprehensive survey that systematically classifies and evaluates the previous work is lacking. Researchers may find it hard to select appropriate generation methods for different scenarios. This survey aims to bridge this gap.

The sources of articles that are taken into consideration are top journals such as those published by IEEE, Springer, Elsevier, etc., and proceedings of top conferences such as AAAI, ICDM, NeurIPS, VLDB, etc. The keywords used for the search strategy are "time series generation", "temporal data generation", "time series prediction", "sequence generation", "series GAN", and "series VAE". We selected the articles related to time series generation and took their common references into consideration. The commonly cited articles and the new articles with sufficient novelty are selected for our survey.

Based on the underlying algorithms and models used by the existing time series data generation methods, this paper divides these methods into four categories: rule-based methods [51–53], simulation-model-based methods [37,45], traditional machine-learning-based methods [29,32,39] and deep-learning-based methods [7,30,44,54–63]. Rule-based methods use a set of rules or constraints to specify the properties and structure of the data to be generated. A simulation-model-based method refers to the establishment of simulation models to simulate various situations in real systems or events, thereby generating time series data. Traditional machine-learning-based methods typically utilize classic machine learning algorithms to generate time series data. These algorithms leverage existing time series data as the training datasets, learn the characteristics and patterns of the time series data, train the generative models, and use these models to generate new time series data. Deep-learning-based methods rely on deep neural network models to generate time series data. They typically use models such as generative adversarial networks (GANs) [64] or variational autoencoders (VAEs) [65] to generate data with specific temporal dependencies.

The key contributions of this paper are summarized as follows:

1. The time series data generation methods are classified into four categories based on their underlying algorithms and models.
2. The characteristics, mechanisms and application scenarios of each category are illustrated.
3. The challenges and future directions of time series data generation are summarized.

This paper first introduces the four categories of time series data generation methods with an analysis of their pros and cons in Section 2, then introduces the methods of each category in detail from Sections 3–6. After that, this paper summarizes the challenges and future directions of time series data generation in Section 7. Finally, the conclusions are presented in Section 8.

2. Four Categories of Time Series Data Generation Methods

Based on the underlying algorithms and models used by time series data generation methods, they can be divided into the following four categories. These methods have several features, of which the need for real data and domain-specific knowledge describe the difficulty of training/constructing the underlying models; the authenticity and complexity of generation results describe the quality of generated time series data; the controllability of results is an important feature describing how much the users can take control of the methods to obtain the desired results. These features of the four categories of methods are summarized in Table 1.

Rule-based methods: These use a set of rules or constraints to specify the properties and structures of the data to be generated. These rules can be based on properties of the data such as data type, data range, data density, etc. In addition, rules can also be based on relationships between data, such as correlations and dependencies, and so on. These rules are used to specify the required data attributes, ensuring that the generated data will conform to these rules and restrictions. These methods do not need to rely on large

amounts of historical data or training models; the generated data may be relatively simple and unrealistic.

Simulation-model-based methods: These approaches use computer simulation techniques to generate time series data based on the modeling of actual scenarios or systems. For example, a fluid dynamics model or a mechanical model may be used to generate corresponding time series data. These methods can generate more realistic data, simulate the behavior of complex systems, and produce different data by changing model parameters, but they require a large amount of domain knowledge and model parameters. Additionally, the amount of calculations is relatively large.

Traditional machine-learning-based methods: These methods are based on traditional machine learning algorithms, which utilize existing time series data to train the models, and then generate new time series data using the models. For example, time series data can be generated using algorithms such as linear regression, support vector machines, or random forests. These methods take into account the influence of historical data, but require parameter adjustment and model training.

Deep-learning-based methods: These methods are based on deep learning algorithms and use deep learning models such as recurrent neural networks (RNNs) or convolutional neural networks (CNNs) to generate time series data. For example, models such as GANs or VAEs can be used to generate time series data. These methods can generate more complex data and take into account the impact of longer time spans, but require a large amount of training data and computing resources.

Table 1. Comparison of time series data generation methods.

Method	References	The Need for Real Data	The Need for Domain-Specific Knowledge	Controllability of Results	Authenticity of Results	Simulating Complex IoT System Behavior
Rule-based methods	[51–53]	×	×	Strong	Weak	Weak
Simulation-model-based methods	[37,45]	✓	✓	Strong	Depends on the model	Depends on the model
Traditional machine-learning-based methods	[29,32,39]	✓	×	Weak	Depends on the model	Strong
Deep-learning-based methods	[7,30,44,54–63]	✓	×	Weak	Strong	Strong

As can be seen from the features shown in Table 1, rule-based methods can be used for scenarios in which the distribution of the generated data does not depend on the real data; simulation-model-based methods can be used for scenarios in which the distribution of the generated data is the same as a known stochastic process in the real world; traditional machine-learning-based and deep-learning-based methods can be used for scenarios in which the distribution of the generated data is unknown beforehand and can be learned from the real data.

3. Rule-Based Methods

This section describes rule-based methods. Section 3.1 introduces three common time series models. Section 3.2 introduces three rule-based methods to generate data. The first two methods are proposed for the data generation of traditional relational databases, while the third solution uses the MAR model, which provides users with a wealth of adjustable parameters.

The characteristics of rule-based methods are as follows:

1. Simplicity and speed: Rule-based methods are simple, only need to define rules and parameters, and use random number generation. Therefore, the generation speed is fast, which can meet some scenarios with high real-time requirements.
2. Strong controllability: The properties of generating time series data can be easily controlled and adjusted using rule-based methods. By modifying the rules and parameters, data that meet specific needs can be easily generated.
3. Low reliance on historical data: Rule-based methods do not require a large amount of historical data to generate new data. This makes them useful in situations where historical data are scarce.
4. Lack of authenticity: Since the generated data are based on fixed rules and parameters, without considering the actual system behavior, they may be different from the actual data and lack authenticity.
5. Inability to simulate complex system behavior: These methods are generally unable to simulate complex system behavior, because the actual system behavior is often very complex and cannot be well described and simulated by simple rules and parameters.

3.1. Common Time Series Models

3.1.1. Autoregressive (AR) Model [66]

An AR model of order p, denoted as $AR(p)$, can be defined as:

$$x_t = \sum_{i=1}^{p} \phi_i x_{t-i} + \epsilon_t \tag{1}$$

where $\phi_1, ..., \phi_p$ are the parameters of the model, and ϵ_t is a white noise, whose samples are regarded as a sequence of serially uncorrelated random variables with zero mean and finite variance, thus providing the randomness.

3.1.2. Moving-Average (MA) Model [67]

An MA model of order q, denoted as $MA(q)$, can be defined as:

$$x_t = \mu + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} + \epsilon_t \tag{2}$$

where μ is the mean of the series, the $\theta_1, ..., \theta_q$ are the parameters, and $\epsilon_t, \epsilon_{t-1}, ..., \epsilon_{t-q}$ are white noise error terms.

3.1.3. Autoregressive Moving-Average (ARMA) Model [68]

An ARMA model with p autoregressive terms and q moving-average terms is denoted as $ARMA(p, q)$. This model contains the $AR(p)$ and$MA(q)$ models:

$$x_t = \epsilon_t + \sum_{i=1}^{p} \phi_i x_{t-i} + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} \tag{3}$$

3.1.4. Autoregressive Integrated Moving-Average (ARIMA) Model [69]

An ARIMA model is a generalization of an ARMA model. An ARIMA model of AR order p, MA order q, and a degree of differencing d, denoted as $ARIMA(p, d, q)$, can be defined as:

$$(1 - \sum_{i=1}^{p} \phi_i L^i)(1 - L)^d x_t = (1 + \sum_{j=1}^{1} \theta_j L^j)\epsilon_t \tag{4}$$

where L is the lag operator, $L^i x_t = x_{t-i}$.

3.2. Rule-Based Methods

3.2.1. FDG [51]

Obtaining comprehensive real data can be difficult, and without a flexible data generation framework capable of modeling various rich data distributions, real data may not be available at all, or it may not be comprehensive enough to thoroughly evaluate the system under consideration. This work proposes a flexible database generation framework, introduces a data generation language (DGL), uses iterators as basic units to form data tuple generation streams, and applies it to generate databases with complex composite distributions and inter-table dependencies.

3.2.2. SRDG [52]

SRDG is a general-purpose relational data generation tool designed for database testing. It supports the definition of relationships within and between tables, and users can specify some simple data characteristics. The data generation algorithm is based on a graph model, in which tables are represented as nodes and foreign-key constraints as edges. The generation algorithm is a depth-first traversal which begins at non-referenced nodes and then examines all out-bound edges, generating data according to the types of the edges.

3.2.3. GRATIS [53]

Generating time series (GRATIS) is a time series data generator that utilizes the mixture autoregressive (MAR) model [70]. The generator is designed for testing various time series analysis methods and provides diverse parameters to efficiently generate new time series data with controllable features. A K-component MAR model, which is actually a finite mixture of K Gaussian AR models, can be defined as:

$$F(x_t|\mathcal{F}_{-t}) = \sum_{k=1}^{K} \alpha_k \Phi\left(\frac{x_t - \phi_{k0} - \phi_{k1}x_{t-1} - ... - \phi_{kp_k}x_{t-p_k}}{\sigma_k}\right) \quad (5)$$

where $F(x_t|\mathcal{F}_{-t})$ is the conditional cumulative distribution of x_t given the past information $F_{-t} \subseteq \{x_{t-1}, ..., x_{t-p_k}\}$, $\Phi(\cdot)$ is the cumulative distribution function of the standard normal distribution, $x_t - \phi_{k0} - \phi_{k1}x_{t-1} - ... - \phi_{kp_k}x_{t-p_k}$ is the autoregressive term in each mixing component, $\sigma_k > 0$ is the standard error, $\sum_{k=1}^{K} \alpha_k = 1$, and $\alpha_k > 0$ for $k = 1, 2, ..., K$.

To generate diverse time series data instances, it uses distributions instead of fixed values for the parameters in the underlying models. It also adopts a genetic algorithm to tune the MAR model parameters to generate time series data with target features extracted from real time series data.

4. Simulation-Model-Based Methods

This section introduces two methods for time series data generation by constructing simulation models. These two methods select specific simulation models for the workload of the cloud data center and wind speed.

The characteristics of simulation-model-based methods are as follows:

1. Ability to simulate actual system behavior: Compared with the rule-based methods, simulation-model-based methods can simulate the behavior of actual systems more accurately, because the simulation models can analyze and model the actual system behavior and simulate the dynamic evolution of the system.
2. Interpretability of generated data: Through the analysis and adjustment of the simulation model, the reasons and rules of the generated data can be well explained, making the generated data more reliable and interpretable.
3. Reliance on model accuracy: The accuracy of the generated data is closely related to the accuracy of the model. If the accuracy of the model is not high, errors and deviations may also exist in the generated data.

4. Slow data generation speed: Compared with the rule-based random generation methods, simulation-model-based methods require model building, so the generation speed is relatively slow and cannot meet the high real-time requirements of the scene.
5. Limited data quantity: Due to the need of model building, a certain amount of historical data are required for training and adjustment of the model. Therefore, when the amount of data is insufficient, the accuracy and reliability of the generated data will be affected.

Kultok et al. [45] proposed a model-based method to create synthetic workload trajectories for cloud data centers. It randomly samples from existing time series data, selects some alternative distributions, and calculates the parameters of these distributions using the maximum likelihood method. After that, the Anderson–Darling test is used to select a suitable distribution from the alternative distributions, and the initial data set is randomly sampled from the selected distribution. Then, it runs an iterative process to rearrange the initial data set. In each iteration, the current series is shuffled, and if it has a smaller mean square error (MSE) than the real time series data, the outcome is selected as the current series. The process continues until the current series reaches an MSE that is less than a threshold.

Bokde et al. [37] proposed two generation methods for synthesizing wind speed time series data. Both of the methods first sample the initial data from a Weibull distribution. The Weibull distribution [71] is parameterized by x, k and its probability density function is

$$f(x;\lambda,k) = \begin{cases} \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-(x/\lambda)^k} & x \geq 0 \\ 0 & x < 0 \end{cases} \quad (6)$$

The methods then rearrange them to make their autocorrelation properties close to the real data, which is the difference between these two methods. The first method arranges the data so that the order of the generated series data is the same as the order of the existing data. In contrast, the second method calculates the parameters of the Weibull distribution based on the existing data, samples the distribution to obtain a batch of data, and selects values whose distances to the real data are less than a threshold in order. The process is repeated until a time series of a given length is generated.

5. Traditional Machine-Learning-Based Methods

This section introduces traditional machine-learning-based methods. Section 5.1 introduces the traditional Markov model. Then, Section 5.2 presents three time series data generation methods based on variants of the Markov model.

The characteristics of the methods of generating time series data based on traditional machine learning are as follows:

1. Ability to process data with more complex patterns: Traditional machine-learning-based methods can process data with more complex patterns, enabling them to better capture the complex relationships between data and generate more realistic data.
2. Fast data generation: The methods have fast data generation speeds, so they can produce large amounts of data in a short period of time.
3. Rely on model accuracy: The accuracy of the generated data is closely related to the accuracy of the model. If the accuracy of the model is not high, errors and deviations may exist in the generated data.
4. Sufficient historical data required: Due to the need to train the model, a certain amount of historical data are necessary for the training and adjustment of the model. Therefore, when the amount of data is insufficient, the accuracy and reliability of the generated data will be affected.

5.1. Markov Model

5.1.1. Discrete-Time Markov Chain [72]

A discrete-time Markov chain is a stochastic process that can be parameterized by empirically estimating transition probabilities between discrete and finite states. The state at time step i is a random variable X_i and the probability of moving to the next state depends only on the present state but not on the previous states:

$$Pr(X_{i+1} = x_{i+1}|X_1 = x_1, ..., X_i = x_i) = Pr(X_{i+1} = x_{i+1}|X_i = x_i) \quad (7)$$

This probability is called the transition probability from state x_i to x_{i+1}. All of the possibilities between the states form a transition matrix.

5.1.2. Hidden Markov Model (HMM) [73]

A hidden Markov model is a statistical Markov model that consists of a hidden state X and an observable state Y. Formally, let X_i and Y_i be discrete-time stochastic processes and $i \geq 1$. X_n is a Markov process whose behavior is not directly observable. The state of Y at step i is determined only by the state of X at step i:

$$P(Y_i = y_i|X_1 = x_1, ..., X_i = x_i) = P(Y_i = y_i|X_i = x_i) \quad (8)$$

This probability is called the emission probability.

5.2. Traditional Machine-Learning-Based Methods

IoTAbench [29] is a benchmark toolkit designed for IoT big data scenarios. It contains a Markov chain-based synthetic data generator for smart meter data. The generator can learn the statistical properties from real time series data. To capture the dependence on several contextual features such as time of day, weather, etc., and incorporate them into the model, the generator augments the Markov chain model by adding additional inputs. It uses maximum likelihood estimation to estimate the transitional probability matrix from the empirical data. It also employs Laplace smoothing, which increases the count for each transition by one, to address the sparse problem of the transitional probability matrix.

Shamshad et al. [32] proposed a method which uses a probability transition matrix of first-order and second-order Markov chains to synthesize new data from existing wind speed data. Each state in the Markov chain represents a wind speed range.

Li et al. [39] proposed a method which uses the Gaussian mixture model hidden Markov model to generate medium- and long-term wind power generation data. The method uses the expectation-maximum (EM) [74] algorithm to estimate the parameters of the model, and then randomly samples from the initial state probability distribution to generate an initial hidden state. It generates a random number according to the uniform distribution on the $(0, 1)$ interval, and finds a hidden state that conforms to the state probability transition matrix according to the random numbers as the hidden state at next time step. To generate time series data from the hidden states, it converts the hidden states into the arguments of the Gaussian mixture model and samples the time series data from the model.

6. Deep-Learning-Based Methods

This section describes deep-learning-based methods. Section 6.1 introduces GANs, and lists several methods that use GANs to generate time series data. Most of these methods use the combination of a GAN and an RNN. Some of the methods also support conditional input. The features of GAN-based methods are summarized in Table 2. Afterwards, Section 6.2 introduces VAEs and time series data generation methods based on them.

The methods of generating time series data based on deep learning have the following features:

1. Learning higher-level features: The deep generative models can learn higher-level features and can automatically capture nonlinear and complex relationships in the data, thereby generating more realistic and complex data.
2. Generating more diverse data: The methods can not only generate data with similar characteristics, but also generate more diverse data, which allows patterns of data to be shown from various angles. This enables a better data generalization ability.
3. Generating more realistic data: The deep generative models can learn the high-order statistical features of data, thereby generating more realistic data, and the differences between the data generated by the model and the real data are becoming smaller and smaller.
4. Difficulty in training: Compared with traditional machine learning models, the training processes of deep generative models are more complicated and require more computing resources and time.
5. High data volume: Deep generation models require large amounts of data for training. If the amount of training data is insufficient, the generalization ability of the model and the accuracy of the generated data will be affected.

Table 2. Comparison of GAN-based methods.

Method	Reference	Support Condition/Metadata	Has Embedding	Other Features
C-RNN-GAN	[44]	×	×	
RCGAN	[7]	✓	×	”Train on synthetic data, test on real data“ strategy
SeqGAN	[54]	×	×	Modeling the generator as a RL policy
T-CGAN	[55]	✓(timestamp only)	×	Support for input data with missing value
TimeGAN	[56]	✓	✓	Combination of supervised and unsupervised loss
DoppelGANger	[57]	✓	×	Separating the generation and discrimination of metadata from time series data
COT-GAN	[58]	×	×	Utilization of causal optimal transport theory
TS-Benchmark	[30]	✓	×	Generating pieces of time series data and splicing them together
RTSGAN	[59]	✓	✓	Fixed length vector in the latent space; support for input data with missing value

6.1. GAN-Based Methods

Generative Adversarial Network [64]:

A generative adversarial network (GAN) is a deep learning model designed by Ian J. Goodfellow et al. Given a training set, this technique learns to generate new data with the same statistics as the training set. It consists of two adversarial parts: a generative model G that captures the data distribution and a discriminative model D that estimates

the probability that a sample came from the training data rather than G. D is trained to maximize the probability of assigning the correct label, while G is trained to minimize the probability of being discriminated. The adversarial parts play a two-player minimax game with value function $V(G, D)$:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[logD(x)] + \mathbb{E}_{z \sim p_z(z)}[log(1 - D(G(z)))] \quad (9)$$

where x is the real data subject to distribution $p_{data}(x)$ and z is an input noise variable of G subject to distribution $p_z(z)$.

6.1.1. C-RNN-GAN [44]

Continuous RNN-GAN (C-RNN-GAN) is a recurrent neural network architecture that is trained with adversarial training to model the whole joint probability of a sequence, and to be able to generate sequences of data. The recurrent network used in the discriminator is long short-term memory (LSTM) [75]. The model is evaluated by learning the generating distribution behind classical music so the signal at every data point is modeled with four real-valued scalars: tone length, frequency, intensity, and time spent since the previous tone.

6.1.2. RCGAN [7]

The recurrent conditional generative network (RCGAN) utilizes a GAN where the generator and discriminator are substituted by recurrent neural networks. The generator of RCGAN accepts a random seed and auxiliary condition input at each step, and the discriminator accepts the output of the generator and the auxiliary condition as input. LSTM was chosen as the implementation of RNN. The maximum mean discrepancy (MMD) was used to evaluate the authenticity of the data generated by the algorithm. This work also proposes a "train on synthetic data, test on real data" (TSTR) approach to evaluate generative algorithms. The process involves training a classifier using the data generated by the algorithm and testing the performance of the classifier on real data to represent the performance of the generated algorithm.

6.1.3. SeqGAN [54]

SeqGAN models the data generator as a stochastic policy in reinforcement learning (RL) [76] and bypasses the generator differentiation problem by directly performing gradient policy updates. The RL reward signal that comes from the GAN discriminator is evaluated on a complete sequence and passed back to the intermediate-state action steps using a Monte Carlo search [77]. SeqGAN first pre-trains the generator G_θ, parameterized by θ using maximum likelihood estimation, and uses its output to pre-train the discriminator D_ϕ, parameterized by ϕ via minimizing the cross-entropy. It then starts adversarial training iteratively. In each iteration, G_θ first generates a sequence $Y_{1:T} = (y_1, ..., y_T)$ and computes an action value $Q(a = y_t, s = Y_{1:t-1})$ for each y_t using the following equation:

$$Q_{D_\phi}^{G_\theta}(a = y_t, s = Y_{1:t-1}) = \begin{cases} \frac{1}{N}\sum_{n=1}^{N} D_\phi(Y_{1:T}^n), Y_{1:T}^n \in MC^{G_\beta}(Y_{1:t}; N) & \text{for } t < T \\ D_\phi(Y_{1:t}) & \text{for } t = T \end{cases} \quad (10)$$

where $MC^{G_\beta}(Y_{1:t}; N)$ is a Monte Carlo search with a roll-out policy G_β to sample the unknown last $T - t$ tokens. After computing the action value, it updates generator parameters with the following gradient:

$$\nabla_\theta J(\theta) \simeq \sum_{t=1}^{T} \mathbb{E}_{y_t \sim G_\theta(y_t|Y_{1:t-1})}[\nabla_\theta log G_\theta(y_t|Y_{1:t-1}) \cdot Q(y_t, Y_{1:t-1})] \quad (11)$$

Then, it uses the current G_θ to generate negative examples, combines them with given positive examples S, and trains D_ϕ. SeqGAN repeats the iterations until it converges. It adopts LSTM and CNN as the implementation for the generator and discriminator, respectively.

6.1.4. T-CGAN [55]

The time conditional generative adversarial network (T-CGAN) is based on conditional generative adversarial networks (CGANs) [78], where the generator is implemented by a deconvolutional neural network and the discriminator is implemented by a convolutional neural network (CNN) [79]. Both the generator and the discriminator are conditioned on the sampling timestamps. This method is primarily used to augment data for time series with irregular sampling. The objective function of this model is:

$$\min_{G}\max_{D} V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[logD(x|t)] + \mathbb{E}_{z\sim p_z(z)}[log(1-D(G(z|t)))] \quad (12)$$

where $t =< t_1, \ldots, t_n >$ is a sorted vector of timestamps sampled at random from a space T.

The generator consists of four deconvolution layers with ReLU activation functions and batch normalization at each layer except for the last one. The discriminator is composed of two layers of convolution, each followed by a max-pooling layer and at the end there is a fully connected layer.

6.1.5. TimeGAN [56]

The time series generative adversarial network (TimeGAN) is the first to combine the flexibility of the unsupervised GAN framework with the control afforded by supervised training in autoregressive models. It adopts the original unsupervised adversarial loss as well as a stepwise supervised loss using the real data as supervision, thereby explicitly encouraging the model to capture the stepwise conditional distributions in the data. Moreover, it utilizes an embedding network to provide a reversible mapping between features and latent representations, thereby reducing the high-dimensionality of the adversarial learning space. It divides the features of time series data into static features $S \in \mathcal{S}$ and temporal features $X \in \mathcal{X}$. TimeGAN consists of four network components: embedding functions $e_{\mathcal{S}} : \mathcal{S} \to \mathcal{H}_{\mathcal{S}}$, $e_{\mathcal{X}} : \mathcal{H}_{\mathcal{S}} \times \mathcal{H}_{\mathcal{X}} \times \mathcal{X} \to \mathcal{H}_{\mathcal{X}}$, recovery functions $r_{\mathcal{S}} : \mathcal{H}_{\mathcal{S}} \to \mathcal{S}$,$r_{\mathcal{X}} : \mathcal{H}_{\mathcal{X}} \to \mathcal{H}_{\mathcal{S}} \times \mathcal{H}_{\mathcal{X}} \times \mathcal{X}$, sequence generators $g_{\mathcal{S}} : \mathcal{Z}_{\mathcal{S}} \to \mathcal{H}_{\mathcal{S}}$, $g_{\mathcal{X}} : \mathcal{H}_{\mathcal{S}} \times \mathcal{H}_{\mathcal{X}} \times \mathcal{Z}_{\mathcal{X}} \to \mathcal{H}_{\mathcal{X}}$, and sequence discriminators. The embedding network provides the latent space, the adversarial network operates within this space, and the latent dynamics of both real and synthetic data are synchronized through a supervised loss. Therefore, the object function contains three parts.

Reconstruction loss:

$$\mathcal{L}_R = \mathbb{E}_{s,x_{1:T}\sim p}\left[\parallel s - \tilde{s} \parallel_2 + \sum_t \parallel x_t - \tilde{x}_t \parallel_2 \right] \quad (13)$$

where $\tilde{s} = r_{\mathcal{S}}(e_{\mathcal{S}}(s))$, $\tilde{x}_t = r_{\mathcal{X}}(e_{\mathcal{X}}(x_t))$. This loss is the difference between the actual data and the data encoded and recovered from the embedding space.

Unsupervised loss:

$$\mathcal{L}_U = \mathbb{E}_{s,x_{1:T}\sim p}\left[\log y_S + \sum_t \log y_t\right] + \mathbb{E}_{s,x_{1:T}\sim \hat{p}}\left[\log(1-\hat{y}_S) + \sum_t \log(1-\hat{y}_t)\right] \quad (14)$$

This is the loss in the traditional GAN model.

Supervised loss:

$$\mathcal{L}_S = \mathbb{E}_{s,x_{1:T}\sim p}\left[\sum_t \parallel h_t - g_{\mathcal{X}}(h_{\mathcal{S}}, h_{t-1}, z_t) \parallel_2 \right] \quad (15)$$

where $h_{\mathcal{S}} = e_{\mathcal{S}}(s)$ and $h_t = e_{\mathcal{X}}(h_{\mathcal{S}}, h_{t-1}, x_t)$. This loss is the difference between the actual next-step latent vector and synthetic next-step latent vector.

Let θ_e, θ_r, θ_g, and θ_d, respectively, denote the parameters of the embedding, recovery, generator, and discriminator networks. The encoder and decoder are trained on both the

reconstruction and supervised losses: $\min_{\theta_e,\theta_r}(\lambda\mathcal{L}_S + \mathcal{L}_R)$. The generator and discriminator are trained on supervised and unsupervised losses: $\min_{\theta_g}(\eta\mathcal{L}_S + \max_{\theta_d})\mathcal{L}_U$.

6.1.6. DoppelGANger [57]

DoppelGANger aims to generate time series data with high fidelity by capturing the complex correlations between measurements and metadata, maintaining long-term correlations and preventing mode collapse. To capture the correlations between metadata and measurements, DoppelGANger first generates metadata using an MLP generator, then it generates measurements using an RNN that takes the metadata as input. It also adopts an auxiliary discriminator which discriminates only on metadata to prevent the discriminator from handling a sample of high dimension. The losses from the two discriminators are combined by a weighting parameter α: $\min_{G}\max_{D_1,D_2}\mathcal{L}_1(G,D_1) + \alpha\mathcal{L}_2(G,D_2)$, where $\mathcal{L}_i, i \in \{1,2\}$ is the Wasserstein loss of the original and the auxiliary discriminator, respectively. To maintain long-term correlations, it adopts LSTM and generates a batch of data with consecutive timestamps at each pass. To address the mode collapse problem for the measurements, it normalizes each time series signal individually, and stores the min/max as "fake" metadata.

This work also identifies the fundamental challenges with both classical notions of privacy and recent advances to improve the privacy properties of GANs, and suggests a potential roadmap for addressing these challenges.

6.1.7. COT-GAN [58]

Causal optimal transport generative adversarial network (COT-GAN) builds on optimal transport (OT) [80] theory, and constrains the transport plans to respect causality: the probability mass moved to the target sequence at time t can only depend on the source sequence up to time t. It uses Sinkhorn divergence with the causal constraint in OT theory to calculate the distance between synthetic and real time series data and makes it an item of the adversarial objective function. This work proposes a mixed Sinkhorn divergence which processes two batches at once to solve the convergence problem in the previous algorithm at the level of mini-batches. The discriminator consists of two separate neural networks parameterized using φ: $h_{\varphi_1} := (h^j_{\varphi_1})^J_{j=1}$, $M_{\varphi_2} := (M^j_{\varphi_2})^J_{j=1}$. The adversarial objective function of COT-GAN is:

$$\widehat{\mathcal{W}}^{mix,L}_{c^K_\varphi,\epsilon}(\hat{x},\hat{x}^{'},\hat{y}_\theta,\hat{y}^{'}_\theta) - \lambda pM_{\varphi_2}(\hat{x}) \tag{16}$$

where $\hat{x}$ and $\hat{x}^{'}$ are empirical measures corresponding to two samples of the dataset, and $\hat{y}_\theta$ and $\hat{y}^{'}_\theta$ are the ones corresponding to two samples from the generator. λ is a positive constant and $pM_{\varphi_2}(\hat{x})$ is the martingale penalization for M_{φ_2}.

Item c^K_φ is a cost function whose output at time t depends on the input only up to time t:

$$c^K_\varphi(x,y) := c(x,y) + \sum_{j=1}^{J}\sum_{t=1}^{T-1} h^j_{\varphi_1,t}(y)\Delta_{t+1}M^j_{\varphi_2}(x) \tag{17}$$

Item $\widehat{\mathcal{W}}^{mix,L}_{c^K_\varphi,\epsilon}(\hat{x},\hat{x}^{'},\hat{y}_\theta,\hat{y}^{'}_\theta)$ is the mixed Sinkhorn divergence:

$$\widehat{\mathcal{W}}^{mix,L}_{c^K_\varphi,\epsilon}(\hat{x},\hat{x}^{'},\hat{y}_\theta,\hat{y}^{'}_\theta) = \mathcal{W}_{c,\epsilon}(\hat{x},\hat{y}_\theta) + \mathcal{W}_{c,\epsilon}(\hat{x}^{'},\hat{y}^{'}_\theta) - \mathcal{W}_{c,\epsilon}(\hat{x},\hat{x}^{'}) - \mathcal{W}_{c,\epsilon}(\hat{y}_\theta,\hat{y}^{'}_\theta) \tag{18}$$

where $\mathcal{W}_{c,\epsilon}(x,y)$ is the Wasserstein distance.

6.1.8. TS-Benchmark [30]

TS-Benchmark is a benchmark of time series database. It contains a deep convolutional generative adversarial network (DCGAN)-based data generation model to generate large volumes of time series data from some real time series data. It first creates seed fragments from real time series data, and then generates synthetic fragments from real seeds using DCGAN. The generated fragments are connected to each other to generate a longer time series. The connectivity of a sequence a to another sequence b is defined as

$$s(a,b) = \frac{1}{\sqrt{\sum_{i=1}^{l}(a_{ti} - b_{hi})^2}} \tag{19}$$

where a_t denotes the tail of sequence a, whose length is l, and b_h denotes the head of sequence b, whose length is l too. A directed graph can be built by creating edges from a segment to others whose connectivity with the segment is greater than a threshold. The weight of a directed edge from segment a to segment b is

$$w(a,b) = \frac{s(a,b) - \bar{s}}{\sum_{b' \in N_a} s(a,b') - \bar{s}} \tag{20}$$

where $N_a = \{b | s(a,b) > \bar{s}\}$ and $\bar{s}$ is the previously mentioned threshold. Given an initial seed sequence a, a subsequent sequence b is generated by a random walk with probability $w(a,b)$ on the directed graph. To connect these two sequences smoothly, a_t and b_h of the two adjacent sequences are spliced using a fitting function $c_i = (1-\sigma_i)a_i + \sigma_i b_i$, where $i = [-\frac{l}{2}, ..., \frac{l}{2})$ is the index of overlaps between sequences and σ is the sigmoid function $\sigma_i = \frac{1}{1+2^{-i}}$.

6.1.9. RTSGAN [59]

The real-world time series generative adversarial network (RTSGAN) consists of an encoder–decoder module and a GAN. It first learns an encoder–decoder module which provides a mapping between a time series data instance and a fixed-dimension latent vector. Subsequently, it learns a generation module to generate vectors in the same latent space. The encoder, which takes dynamic and global features transformed into $[0,1]$ as its input, is composed of an N-layer gated recurrent unit (GRU), a pooling layer, and a fully connected layer using LeakyReLU as the activation function. The decoder first reconstructs the global features via a fully connected layer, and then reconstructs the dynamic features via a GRU. The overall loss function of the encoder–decoder module is a linear combination of reconstruction loss for global features and dynamics features. The generation module uses an improved version of WGAN in which the generator aims to minimize the 1-Wasserstein distance between the real data distribution and synthetic data distribution with the help of an iteratively trained 1-Lipschitz discriminator.

Furthermore, this work proposes RTSGAN-M to address the value missing problem in time series data. RTSGAN-M adopts an observation embedding to enrich the information at each time step, and a decide-and-generate decoder which first determines the time and missing patterns of the next step and then generates the corresponding feature values based on both local and global dependencies.

6.2. VAE-Based Methods

6.2.1. Variational Autoencoder

Autoencoder is an unsupervised algorithm used for feature extraction or data dimensionality reduction. It consists of an encoder and a decoder. The input features x are abstracted into intermediate variables y by the encoder, and then mapped back to the original data space $\bar{x}$ by the decoder, aiming to reconstruct the original data as accurately as possible. The purpose of an autoencoder is to extract abstract features y, and its learning process minimizes the loss function $L(x, \bar{x})$. The mean squared error function

can be used: $L(x, \bar{x}) = \sum_{i=1}^{n} ||x_i - \bar{x}_i||^2 = \sum_{i=1}^{n} ||x_i - d(e(x_i))||^2$. Where i represents the ith sample, and $x_i \in \mathbb{R}^n$. Autoencoder can go from raw data x to abstract features y, which can achieve tasks such as data dimensionality reduction, denoising, compression, and feature extraction.

The autoencoder can reconstruct the intermediate variable y to $\bar{x}$. The variational autoencoder attempts to infer and learn the distribution of intermediate variable y, and generates data by sampling from y. It is a generative model. The variational autoencoder assumes that the posterior probability $q_\phi(y|x)$ follows a multi-dimensional mixture normal distribution, using two networks to estimate the mean and variance of hidden state $z^{(i)}$ corresponding to each sample. By regularizing the loss function with a regularization term, it ensures that $q_\phi(y|x)$ conforms to a standard normal distribution, ensuring the generative ability of the model.

6.2.2. FSTS [60]

FSTS (few-shot learning for time series data generation) is a method for generating time series data based on autoencoders. Firstly, a small amount of data is used to pre-train the autoencoder. The encoder maps input data into the hidden space, $x_h = E(x_{in})$, and then the decoder restores it back to the original space, $x_{out} = D(x_h)$, by minimizing mean square error between input and output data during training. Through pre-training, the autoencoder models the hidden space well enough for generating sufficient amounts of hidden space data from which large volumes of generated samples can be obtained using decoder restoration mechanisms. Although this method applies an autoencoder technique in data generation with good results achieved, it takes no specific design targeted at time series data and also fails to address problems related to regularity within the latent space.

6.2.3. VRNN [61]

Variational RNN (VRNN) combines the methods of VAE and RNN by using latent-space random variables to represent time sequences, which explores for the first time the combination of VAE and RNN for generating sequence data. The process of generating data begins with calculating the prior probability based on the previous hidden state as historical information, then generating data by sampling from the prior probability, updating the hidden state, and calculating the posterior probability. The objective function of VRNN is basically consistent with that of the original VAE.

$$\mathbb{E}_{q(z_{\leq T}|x_{\leq T})}\left[\sum_{t=1}^{T}(-\mathrm{KL}(q(z_t \mid x_{\leq t}, z_{<t})\|p(z_t \mid x_{<t}, z_{<t})) + \log p(x_t \mid z_{\leq t}, x_{<t}))\right]. \quad (21)$$

6.2.4. SRNN [62]

SRNN builds upon the basis of VRNN and combines RNN and SSM, which are commonly used methods in time series modeling. RNN is known for its strong non-linear fitting ability, but its hidden state is deterministic. On the other hand, SSM's random state transition is more suitable for uncertainty modeling, but the inference process is usually simple. By integrating RNN and SSM, SRNN ensures that the random state does not affect the deterministic state during the generation process. During the inference process, SRNN incorporates a reverse RNN to capture future information.

$$F_i(\theta, \phi) = E_{q_\phi}\left[\log p_\theta(x_{1:T}|z_{1:T}, \widetilde{d}_{1:T})\right] - KL\left(q_\phi(z_{1:T}|\widetilde{d}_{1:T}, x_{1:T}, z_0) \,\|\, p_\theta(z_{1:T}|\widetilde{d}_{1:T}, z_0)\right) \quad (22)$$

6.2.5. DSAE [63]

Compared to traditional autoencoders, the DSAE model introduces hidden variables that are invariant and variant over time, which can better handle temporal information in sequence data. Specifically, the DSAE model divides the hidden space into two parts: variant and invariant over time. The variable that varies with time represents the trend of data on

the time axis, while the invariant variable represents features that remain unchanged on the time axis. This hierarchical architecture can distinguish between latent time-related features and those independent of time, helping to capture both static and dynamic characteristics of sequence data and further improving the model's generative ability.

7. Challenges and Future Directions

Time series data generation methods still face the following challenges:

1. High dimensionality of time series: Time series data usually have high dimensionality, which increases the difficulty of training the time series data generation model. Generative models need to have sufficient memory capacity while avoiding overfitting to handle noise and discontinuities in the generation processes.
2. Long-term dependencies: Time series data usually have long-term dependencies, that is, previous data points may have an impact on the generation of subsequent data points. Many traditional generative models do not handle such long-term dependencies well, and for some time series data, such dependencies can be very important.
3. Insufficient data samples: In some fields, such as healthcare and finance, the data may be very limited and rarely labeled. Therefore, how to efficiently train generative models to generate high-quality time series data remains a challenging problem.

Different categories of methods face different challenges according to their properties. Learning-based methods mostly face all of the challenges, while other methods face challenges 1 and 2.

Future development directions of time series data generation methods may include the following aspects:

1. More efficient model design: At present, many time series data generation models have been proposed, but most of them require long training times and large amounts of computing resources. Future directions may include designing more efficient models to reduce the training time and computational cost.
2. Better long-term dependency modeling: Time series data usually exhibit long-term dependencies, where previous data points may have an impact on the generation of subsequent data points. Future directions may include designing better generative models that can efficiently handle long-term dependencies and maintain data continuity and smoothness during generation.
3. Time series data generation applications based on deep learning and advanced technologies: With the development of deep learning and other advanced technologies, future development directions may include the development of more time series data generation applications, such as speech synthesis, music generation, video generation, machine translation, natural language generation, etc. These applications can be combined with other technologies, such as automation, augmented reality, and virtual reality, etc., to create more intelligent, natural, and real time series data generation applications.

8. Conclusions

This paper introduces four categories of time series data generation methods in IoT, including rule-based methods, simulation-model-based methods, traditional machine-learning-based methods, and deep-learning-based methods. Among them, the rule-based methods are simple, fast, and highly controllable, but the generated results lack authenticity and cannot simulate complex system behavior so they can be used for scenarios in which the distribution of the generated data does not need to correspond to real data. The simulation-model-based methods can simulate the actual system behavior and are interpretable, but the data generation is limited by the specific situation in a particular field so they can be used for scenarios in which the distribution of the generated data is the same as a known stochastic process in the real world. The traditional machine-learning-based methods can handle more complex patterns of data and generate data faster, but the quality of data depends on the accuracy of the model and the quality of historical data. The deep-learning-

based methods can learn higher-level features and generate more diverse and real data, but they require more computing resources, a larger volume of data, and longer processing time. These two categories of methods can be used for scenarios in which the distribution of the generated data is unknown beforehand and can be learned from real data.

Nowadays, time series data generation methods still face challenges in generating high-dimensional and long time series data. In addition, time series data generation methods need to solve problems such as insufficient data samples. Therefore, future research in time series data generation should focus on improving model efficiency, enhancing long-term dependency modeling, and following up the development of new deep learning technologies. We believe that the systematic classification and evaluation presented in this survey will be a valuable reference for researchers in the time series data generation field.

There are still several limitations of this survey. First, the classification is rough and lacks detailed classification, such as application-based classification. The scope of this survey does not include methods for time series data generation in a broad sense, such as text generation. The potential usages of big models such as GPT for time series data generation are also not discussed. These limitations will be the guidance for future studies.

Author Contributions: Conceptualization, C.H., C.L. and Y.Z.; investigation, C.H. and Z.S.; writing—original draft preparation, Z.S. and C.H.; writing—review and editing, Z.S., Y.Z. and C.L.; visualization, Z.S.; supervision, C.X.; project administration, Y.Z.; funding acquisition, Y.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Social Science Fund of China (22&ZD141).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Li, S.; Xu, L.D.; Zhao, S. The internet of things: A survey. *Inf. Syst. Front.* **2015**, *17*, 243–259.
2. Mobaraki, B.; Pascual, F.J.C.; Garcia, A.M.; Mascaraque, M.Á.M.; Vázquez, B.F.; Alonso, C. Studying the impacts of test condition and nonoptimal positioning of the sensors on the accuracy of the in-situ U-value measurement. *Heliyon* **2023**, *9*, 17282. [CrossRef]
3. Mobaraki, B.; Castilla Pascual, F.J.; Lozano-Galant, F.; Lozano-Galant, J.A.; Porras Soriano, R. In situ U-value measurement of building envelopes through continuous low-cost monitoring. *Case Stud. Therm. Eng.* **2023**, *43*, 102778. [CrossRef]
4. Coxon, G.; Addor, N.; Bloomfield, J.P.; Freer, J.; Fry, M.; Hannaford, J.; Howden, N.J.; Lane, R.; Lewis, M.; Robinson, E.L.; et al. CAMELS-GB: Hydrometeorological time series and landscape attributes for 671 catchments in Great Britain. *Earth Syst. Sci. Data* **2020**, *12*, 2459–2483.
5. Sezer, O.B.; Gudelek, M.U.; Ozbayoglu, A.M. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Appl. Soft Comput.* **2020**, *90*, 106181.
6. Feyrer, J. Trade and income—Exploiting time series in geography. *Am. Econ. J. Appl. Econ.* **2019**, *11*, 1–35. [CrossRef]
7. Esteban, C.; Hyland, S.L.; Rätsch, G. Real-valued (medical) time series generation with recurrent conditional gans. *arXiv* **2017**, arXiv:1706.02633.
8. Haradal, S.; Hayashi, H.; Uchida, S. Biosignal data augmentation based on generative adversarial networks. In Proceedings of the 2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Honolulu, HI, USA, 18–21 July 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 368–371.
9. Li, H.; Xu, X.L.; Dai, D.W.; Huang, Z.Y.; Ma, Z.; Guan, Y.J. Air pollution and temperature are associated with increased COVID-19 incidence: A time series study. *Int. J. Infect. Dis.* **2020**, *97*, 278–282. [CrossRef]
10. Liu, F.; Elliott, J.; Craig, T.; Hooper, A.; Wright, T. Improving the resolving power of InSAR for earthquakes using time series: A case study in Iran. *Geophys. Res. Lett.* **2021**, *48*, e2021GL093043. [CrossRef]
11. Jensen, S.K.; Pedersen, T.B.; Thomsen, C. Time series management systems: A survey. *IEEE Trans. Knowl. Data Eng.* **2017**, *29*, 2581–2600. [CrossRef]
12. Wang, C.; Huang, X.; Qiao, J.; Jiang, T.; Rui, L.; Zhang, J.; Kang, R.; Feinauer, J.; McGrail, K.A.; Wang, P.; et al. Apache iotdb: Time-series database for internet of things. *Proc. Vldb Endow.* **2020**, *13*, 2901–2904. [CrossRef]
13. Ghaderpour, E.; Pagiatakis, S.D.; Hassan, Q.K. A survey on change detection and time series analysis with applications. *Appl. Sci.* **2021**, *11*, 6141. [CrossRef]

14. Mudelsee, M. Trend analysis of climate time series: A review of methods. *Earth-Sci. Rev.* **2019**, *190*, 310–322.
15. Feng, C.; Liu, Y.; Zhao, H. Periodic measures and Wasserstein distance for analysing periodicity of time series datasets. *Commun. Nonlinear Sci. Numer. Simul.* **2023**, *120*, 107166. [CrossRef]
16. Puech, T.; Boussard, M.; D'Amato, A.; Millerand, G. A fully automated periodicity detection in time series. In Proceedings of the Advanced Analytics and Learning on Temporal Data: 4th ECML PKDD Workshop, AALTD 2019, Würzburg, Germany, 20 September 2019; Revised Selected Papers 4; Springer: Berlin/Heidelberg, Germany, 2020; pp. 43–54.
17. Zhou, S.; Wang, X.; Zhou, W.; Zhang, C. Recognition of the scale-free interval for calculating the correlation dimension using machine learning from chaotic time series. *Phys. Stat. Mech. Its Appl.* **2022**, *588*, 126563. [CrossRef]
18. Edelmann, D.; Fokianos, K.; Pitsillou, M. An updated literature review of distance correlation and its applications to time series. *Int. Stat. Rev.* **2019**, *87*, 237–262. [CrossRef]
19. Park, M.H.; Chakraborty, S.; Vuong, Q.D.; Noh, D.H.; Lee, J.W.; Lee, J.U.; Choi, J.H.; Lee, W.J. Anomaly Detection Based on Time Series Data of Hydraulic Accumulator. *Sensors* **2022**, *22*, 9428. [CrossRef]
20. Kim, B.; Alawami, M.A.; Kim, E.; Oh, S.; Park, J.; Kim, H. A Comparative Study of Time Series Anomaly Detection Models for Industrial Control Systems. *Sensors* **2023**, *23*, 1310. [CrossRef]
21. Wang, C.; Xing, S.; Gao, R.; Yan, L.; Xiong, N.; Wang, R. Disentangled Dynamic Deviation Transformer Networks for Multivariate Time Series Anomaly Detection. *Sensors* **2023**, *23*, 1104. [CrossRef]
22. Karim, F.; Majumdar, S.; Darabi, H.; Chen, S. LSTM fully convolutional networks for time series classification. *IEEE Access* **2017**, *6*, 1662–1669. [CrossRef]
23. Wang, Z.; Yan, W.; Oates, T. Time series classification from scratch with deep neural networks: A strong baseline. In Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, USA, 14–19 May 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 1578–1585.
24. Ismail Fawaz, H.; Forestier, G.; Weber, J.; Idoumghar, L.; Muller, P.A. Deep learning for time series classification: A review. *Data Min. Knowl. Discov.* **2019**, *33*, 917–963. [CrossRef]
25. Bandara, K.; Bergmeir, C.; Smyl, S. Forecasting across time series databases using recurrent neural networks on groups of similar series: A clustering approach. *Expert Syst. Appl.* **2020**, *140*, 112896. [CrossRef]
26. Maharaj, E.A.; D'Urso, P.; Caiado, J. *Time Series Clustering and Classification*; CRC Press: Boca Raton, FL, USA, 2019.
27. Lin, H.; Bergmann, N.W. IoT privacy and security challenges for smart home environments. *Information* **2016**, *7*, 44. [CrossRef]
28. Tawalbeh, L.; Muheidat, F.; Tawalbeh, M.; Quwaider, M. IoT Privacy and security: Challenges and solutions. *Appl. Sci.* **2020**, *10*, 4102. [CrossRef]
29. Arlitt, M.; Marwah, M.; Bellala, G.; Shah, A.; Healey, J.; Vandiver, B. Iotabench: An internet of things analytics benchmark. In Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering, Austin TX, USA, 28 January–4 February 2015; pp. 133–144.
30. Hao, Y.; Qin, X.; Chen, Y.; Li, Y.; Sun, X.; Tao, Y.; Zhang, X.; Du, X. Ts-benchmark: A benchmark for time series databases. In Proceedings of the 2021 IEEE 37th International Conference on Data Engineering (ICDE), Chania, Greece, 19–22 April 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 588–599.
31. Zhang, C.; Kuppannagari, S.R.; Kannan, R.; Prasanna, V.K. Generative adversarial network for synthetic time series data generation in smart grids. In Proceedings of the 2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm), Aalborg, Denmark, 29–31 October 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–6.
32. Shamshad, A.; Bawadi, M.; Hussin, W.W.; Majid, T.A.; Sanusi, S. First and second order Markov chain models for synthetic generation of wind speed time series. *Energy* **2005**, *30*, 693–708. [CrossRef]
33. Chen, P.; Pedersen, T.; Bak-Jensen, B.; Chen, Z. ARIMA-based time series model of stochastic wind power generation. *IEEE Trans. Power Syst.* **2009**, *25*, 667–676. [CrossRef]
34. Shi, J.; Guo, J.; Zheng, S. Evaluation of hybrid forecasting approaches for wind speed and power generation time series. *Renew. Sustain. Energy Rev.* **2012**, *16*, 3471–3480. [CrossRef]
35. Kardakos, E.G.; Alexiadis, M.C.; Vagropoulos, S.I.; Simoglou, C.K.; Biskas, P.N.; Bakirtzis, A.G. Application of time series and artificial neural network models in short-term forecasting of PV power generation. In Proceedings of the 2013 48th International Universities' Power Engineering Conference (UPEC), Dublin, Ireland, 2–5 September 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 1–6.
36. Bright, J.; Smith, C.; Taylor, P.; Crook, R. Stochastic generation of synthetic minutely irradiance time series derived from mean hourly weather observation data. *Sol. Energy* **2015**, *115*, 229–242. [CrossRef]
37. Bokde, N.D.; Feijoo, A.; Al-Ansari, N.; Yaseen, Z.M. A comparison between reconstruction methods for generation of synthetic time series applied to wind speed simulation. *IEEE Access* **2019**, *7*, 135386–135398. [CrossRef]
38. Talbot, P.W.; Rabiti, C.; Alfonsi, A.; Krome, C.; Kunz, M.R.; Epiney, A.; Wang, C.; Mandelli, D. Correlated synthetic time series generation for energy system simulations using Fourier and ARMA signal processing. *Int. J. Energy Res.* **2020**, *44*, 8144–8155. [CrossRef]
39. Li, Y.; Hu, B.; Niu, T.; Gao, S.; Yan, J.; Xie, K.; Ren, Z. GMM-HMM-based medium-and long-term multi-wind farm correlated power output time series generation method. *IEEE Access* **2021**, *9*, 90255–90267. [CrossRef]

40. Bogárdi, J.J.; Duckstein, L.; Rumambo, O.H. Practical generation of synthetic rainfall event time series in a semi-arid climatic zone. *J. Hydrol.* **1988**, *103*, 357–373. [CrossRef]
41. Smakhtin, V.Y. Generation of natural daily flow time-series in regulated rivers using a non-linear spatial interpolation technique. *Regul. Rivers Res. Manag. Int. J. Devoted River Res. Manag.* **1999**, *15*, 311–323. [CrossRef]
42. Efstratiadis, A.; Dialynas, Y.G.; Kozanis, S.; Koutsoyiannis, D. A multivariate stochastic model for the generation of synthetic time series at multiple time scales reproducing long-term persistence. *Environ. Model. Softw.* **2014**, *62*, 139–152. [CrossRef]
43. Wiese, M.; Knobloch, R.; Korn, R.; Kretschmer, P. Quant GANs: Deep generation of financial time series. *Quant. Financ.* **2020**, *20*, 1419–1440. [CrossRef]
44. Mogren, O. C-RNN-GAN: Continuous recurrent neural networks with adversarial training. *arXiv* **2016**, arXiv:1611.09904.
45. Koltuk, F.; Schmidt, E.G. A novel method for the synthetic generation of non-iid workloads for cloud data centers. In Proceedings of the 2020 IEEE Symposium on Computers and Communications (ISCC), Rennes, France, 7–10 July 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–6.
46. Manunta, M.; De Luca, C.; Zinno, I.; Casu, F.; Manzo, M.; Bonano, M.; Fusco, A.; Pepe, A.; Onorato, G.; Berardino, P.; et al. The parallel SBAS approach for Sentinel-1 interferometric wide swath deformation time-series generation: Algorithm description and products quality assessment. *IEEE Trans. Geosci. Remote. Sens.* **2019**, *57*, 6259–6281. [CrossRef]
47. Chuvieco, E.; Englefield, P.; Trishchenko, A.P.; Luo, Y. Generation of long time series of burn area maps of the boreal forest from NOAA–AVHRR composite data. *Remote. Sens. Environ.* **2008**, *112*, 2381–2396. [CrossRef]
48. Hilker, T.; Wulder, M.A.; Coops, N.C.; Seitz, N.; White, J.C.; Gao, F.; Masek, J.G.; Stenhouse, G. Generation of dense time series synthetic Landsat data through data blending with MODIS using a spatial and temporal adaptive reflectance fusion model. *Remote. Sens. Environ.* **2009**, *113*, 1988–1999. [CrossRef]
49. Bonano, M.; Manunta, M.; Marsella, M.; Lanari, R. Long-term ERS/ENVISAT deformation time-series generation at full spatial resolution via the extended SBAS technique. *Int. J. Remote. Sens.* **2012**, *33*, 4756–4783. [CrossRef]
50. Alzantot, M.; Chakraborty, S.; Srivastava, M. Sensegen: A deep learning architecture for synthetic sensor data generation. In Proceedings of the 2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), Kona, HI, USA, 13–17 March 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 188–193.
51. Bruno, N.; Chaudhuri, S. Flexible database generators. In Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, 30 August–2 September 2005; pp. 1097–1107.
52. Houkjær, K.; Torp, K.; Wind, R. Simple and realistic data generation. In Proceedings of the 32nd International Conference on Very Large Data Bases, Seoul, Republic of Korea, 12–15 September 2006; pp. 1243–1246.
53. Kang, Y.; Hyndman, R.J.; Li, F. GRATIS: GeneRAting TIme Series with diverse and controllable characteristics. *Stat. Anal. Data Mining ASA Data Sci. J.* **2020**, *13*, 354–376. [CrossRef]
54. Yu, L.; Zhang, W.; Wang, J.; Yu, Y. Seqgan: Sequence generative adversarial nets with policy gradient. In Proceedings of the AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017; Volume 31.
55. Ramponi, G.; Protopapas, P.; Brambilla, M.; Janssen, R. T-cgan: Conditional generative adversarial network for data augmentation in noisy time series with irregular sampling. *arXiv* **2018**, arXiv:1811.08295.
56. Yoon, J.; Jarrett, D.; Van der Schaar, M. Time-series generative adversarial networks. *Adv. Neural Inf. Process. Syst.* **2019**, *32*.
57. Lin, Z.; Jain, A.; Wang, C.; Fanti, G.; Sekar, V. Using gans for sharing networked time series data: Challenges, initial promise, and open questions. In Proceedings of the ACM Internet Measurement Conference, Virtual Event, 27–29 October 2020; pp. 464–483.
58. Xu, T.; Wenliang, L.K.; Munn, M.; Acciaio, B. Cot-gan: Generating sequential data via causal optimal transport. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 8798–8809.
59. Pei, H.; Ren, K.; Yang, Y.; Liu, C.; Qin, T.; Li, D. Towards generating real-world time series data. In Proceedings of the 2021 IEEE International Conference on Data Mining (ICDM), Auckland, New Zealand, 7–10 December 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 469–478.
60. Zheng, Y.; Zhang, Z.; Cui, R. Few-Shot Learning for Time Series Data Generation Based on Distribution Calibration. In Proceedings of the Web Information Systems and Applications: 18th International Conference, WISA 2021, Kaifeng, China, 24–26 September 2021; Proceedings 18. Springer: Berlin/Heidelberg, Germany, 2021; pp. 198–206.
61. Chung, J.; Kastner, K.; Dinh, L.; Goel, K.; Courville, A.C.; Bengio, Y. A recurrent latent variable model for sequential data. *Adv. Neural Inf. Process. Syst.* **2015**, *28*. Available online: https://github.com/jych/nips2015_vrnn (accessed on 1 July 2023).
62. Fraccaro, M.; Sønderby, S.K.; Paquet, U.; Winther, O. Sequential neural models with stochastic layers. *Adv. Neural Inf. Process. Syst.* **2016**, *29*. [CrossRef]
63. Li, Y.; Mandt, S. Disentangled sequential autoencoder. *arXiv* **2018**, arXiv:1803.02991.
64. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial networks. *Commun. Acm* **2020**, *63*, 139–144. [CrossRef]
65. Doersch, C. Tutorial on variational autoencoders. *arXiv* **2016**, arXiv:1606.05908.
66. Lewis, R.; Reinsel, G.C. Prediction of multivariate time series by autoregressive model fitting. *J. Multivar. Anal.* **1985**, *16*, 393–411. [CrossRef]
67. Durbin, J. Efficient estimation of parameters in moving-average models. *Biometrika* **1959**, *46*, 306–316. [CrossRef]
68. Kashyap, R.L. Optimal choice of AR and MA parts in autoregressive moving average models. *IEEE Trans. Pattern Anal. Mach. Intell.* **1982**, *PAMI-4*, 99–104. [CrossRef] [PubMed]

69. Nelson, B.K. Time series analysis using autoregressive integrated moving average (ARIMA) models. *Acad. Emerg. Med.* **1998**, *5*, 739–744. [CrossRef]
70. Wong, C.S.; Li, W.K. On a mixture autoregressive model. *J. R. Stat. Soc. Ser. (Stat. Methodol.)* **2000**, *62*, 95–115. [CrossRef]
71. Rinne, H. *The Weibull Distribution: A Handbook*; CRC Press: Boca Raton, FL, USA, 2008.
72. Norris, J.R. *Markov Chains*; Number 2; Cambridge University Press: Cambridge, UK, 1998.
73. Rabiner, L.; Juang, B. An introduction to hidden Markov models. *IEEE Assp Mag.* **1986**, *3*, 4–16. [CrossRef]
74. Xuan, G.; Zhang, W.; Chai, P. EM algorithms of Gaussian mixture model and hidden Markov model. In Proceedings of the 2001 International Conference on Image Processing (Cat. No. 01CH37205), Thessaloniki, Greece, 7–10 October 2001; IEEE: Piscataway, NJ, USA, 2001; Volume 1, pp. 145–148.
75. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]
76. Kaelbling, L.P.; Littman, M.L.; Moore, A.W. Reinforcement learning: A survey. *J. Artif. Intell. Res.* **1996**, *4*, 237–285. [CrossRef]
77. Browne, C.B.; Powley, E.; Whitehouse, D.; Lucas, S.M.; Cowling, P.I.; Rohlfshagen, P.; Tavener, S.; Perez, D.; Samothrakis, S.; Colton, S. A survey of monte carlo tree search methods. *IEEE Trans. Comput. Intell. Games* **2012**, *4*, 1–43. [CrossRef]
78. Mirza, M.; Osindero, S. Conditional generative adversarial nets. *arXiv* **2014**, arXiv:1411.1784.
79. O'Shea, K.; Nash, R. An introduction to convolutional neural networks. *arXiv* **2015**, arXiv:1511.08458.
80. Villani, C. *Optimal Transport: Old and New*; Springer: Berlin/Heidelberg, Germany, 2009; Volume 338.