Applied Intelligence
https://doi.org/10.1007/s10489-022-03742-7

Controlling non-stationarity and periodicities in time series generation using conditional invertible neural networks

Benedikt Heidrich[1] · **Marian Turowski**[1] · **Kaleb Phipps**[1] · **Kai Schmieder**[1,2] · **Wolfgang Süß**[1] · **Ralf Mikut**[1] · **Veit Hagenmeyer**[1]

Accepted: 7 May 2022

Abstract

Generated synthetic time series aim to be both realistic by mirroring the characteristics of real-world time series and useful by including characteristics that are useful for subsequent applications, such as forecasting and missing value imputation. To generate such realistic and useful time series, we require generation methods capable of controlling the non-stationarity and periodicities of the generated time series. However, existing approaches do not consider such explicit control. Therefore, in the present paper, we present a novel approach to control non-stationarity and periodicities with calendar and statistical information when generating time series. We first define the requirements for methods to generate time series with non-stationarity and periodicities, which we show are not fulfilled by existing generation methods. Second, we formally describe the novel approach for controlling non-stationarity and periodicities in generated time series. Thirdly, we introduce an exemplary implementation of this approach using a conditional Invertible Neural Network (cINN). We evaluate this cINN empirically in experiments with real-world data sets and compare it to state-of-the-art time series generation methods. Our experiments show that the evaluated cINN can generate time series with controlled periodicities and non-stationarity, and it also generally outperforms the selected benchmarks.

Keywords Time series generation · Generation methods · Synthetic time series · Non-stationarity · Periodicities · Conditional invertible neural networks

1 Introduction

Using real-world time series is associated with issues such as lack of sufficient data, privacy concerns, or anomalies in various applications. Therefore, real-world time series are often replaced or complemented with synthetic time series to address these issues. For this reason, the generation of synthetic time series aims to provide realistic and useful time series. Generated time series are realistic if they have characteristics similar to real-world time series, such as seasonality and trend [13]. For example, an electricity consumption time series typically has reoccurring daily, weekly, and yearly patterns [10]. However, a realistic generated time series is not necessarily useful for all tasks. For example, to analyse the future growth of a startup company, it may be necessary to generate time series with seasonal fluctuations and a trend when given only a small sample of stationary input data.[1] Therefore, to generate a useful time series, one must be able to influence or even design time series' characteristics during generation independently of the available input data.

Generating realistic and useful time series thus requires generation methods to control the generation of time series further. Firstly, generation methods have to control the non-stationarity of the generated time series to incorporate effects that change the value of the time series at different times. Examples of non-stationary time series are daily stock prices, the monthly beer production, or the annual number of strikes [13]. Secondly, these methods must be able to control the periodicities in the generated time series to represent regularly occurring patterns such as the patterns mentioned above in electricity consumption time series.

Despite generally promising results (e.g. [5, 19, 22, 23]), existing time series generation methods do not explicitly control the non-stationarity and periodicities of

✉ Benedikt Heidrich
benedikt.heidrich@kit.edu

Extended author information available on the last page of the article.

[1] We further elaborate realistic and useful time series in Appendix A.

Published online: 03 August 2022

the generated time series. Instead, these approaches learn a mapping from the latent space, which lacks temporal information, to the realisation space. As a result, all newly sampled data follow the same distribution, resulting in stationary time series. Furthermore, the periodicities in the generated time series are limited to those learned from the training sample.

In the present paper, we thus present a novel approach to control non-stationarity and periodicities with calendar and statistical information when generating time series. For this, we make the following contributions: Firstly, we define the requirements for generation methods to generate time series with non-stationarity and periodicities, which we show is not fulfilled by existing generation methods. Secondly, we formally describe the novel approach for controlling non-stationarity and periodicities in generated time series. Thirdly, we introduce an exemplary implementation of this approach using a conditional Invertible Neural Network (cINN) that preprocesses calendar and statistical information as conditional input with a conditioning network.

To evaluate the proposed cINN, we empirically examine its capabilities to generate time series with controlled non-stationarity and periodicities in experiments with real-world data sets. We also compare the general quality of the time series generated with the cINN to that of state-of-the-art time series generation methods and perform an ablation study to analyse the effects of the conditional information.

2 Related work

The present paper introduces a novel approach for generating realistic and useful time series by controlling non-stationarity and periodicities. In this section, we thus describe how recent approaches to generate realistic and useful time series consider the temporal structure.

The first group of approaches considers the temporal structure independent from a specific domain: Xu et al. [22], for example, propose COT-GAN. It considers causality in the generation process by utilising the causal optimal transport theory. With this, the output of a point of time depends only on inputs up to that point of time. Yoon et al. [23] also focus on the temporal dynamics of generated time series: By using a supervised loss, the proposed TimeGAN can better capture temporal dynamics. In addition to temporal structures, MuseGAN by Dong et al. [6] considers the interplay of different instruments. For this purpose, they integrate multiple generators that focus on different music characteristics. To generate time series with irregular steps, Ramponi et al. [20] present the T-CGAN that uses timestamps to learn the relationship between data and time.

The second group of approaches additionally uses domain-specific information to consider temporal structures. Esteban et al. [8] introduce the Recurrent Conditional GAN (RCGAN) for generating realistic synthetic medical data. Thanks to conditioning utilising specific labels, RCGAN can create time series that can also be used for training models without imposing privacy concerns. For performing high-quality speech synthesis, Prenger et al. [19] combine the melspectograms presented in WaveNet [17] with an Invertible Neural Network. Furthermore, to generate electrical consumption time series for scheduling and energy management, Lan et al. [16] condition their Wasserstein GAN on temporal and consumer information.

Altogether, existing works consider the temporal structure and domain-specific information when generating time series. However, to the best of our knowledge, no work exists that successfully generates realistic and useful time series by controlling non-stationarity and periodicities.

3 Problem formulation

To generate realistic and useful time series, generation methods must be able to control non-stationarity and periodicities during generation. This section firstly formalises the related requirements before highlighting why current generation methods cannot fulfil them.

3.1 Requirements of non-stationarity and periodicities in time series

This section briefly formalises the requirements of non-stationarity and periodicities in time series with calendar information based on phenomena observed in real-world time series.

Requirement 1: non-stationarity Realistic time series can include phenomena such as trends or seasonality. Such characteristics can be represented as components of a time series. In time series analysis, a time series $\mathcal{X}_t$ is typically decomposed into a seasonal component S_t, a trend component T_t, and a random (or remainder) component R_t, i.e. $\mathcal{X}_t = S_t + T_t + R_t$[2] [13].

Since these components can cause a time series to be non-stationary, we detail non-stationary time series. We regard non-stationary time series as realisations of a *non-stationary* stochastic process $\{X_t\}$. However, we must consider stationary stochastic processes due to the lack of a formal definition of non-stationary stochastic processes.

[2] A multiplicative decomposition $\mathcal{X}_t = S_t \times T_t \times R_t$ is also typical [13]. Regardless of considering the additive or multiplicative decomposition, we do not imply the independence of their components in the following.

For time series, it is sufficient to focus on the properties of *weakly-stationary* stochastic processes. According to Hyndman and Athanasopoulos [13], a weakly stationary stochastic process $\{X_t\}$ has the following properties:

1. $\mu = \mathbb{E}[X_t] = \mathbb{E}[X_{t+\tau}],\ \forall t \in [1, L], \forall \tau \in \mathbb{N}$,
2. $\sigma^2 = \text{var}[X_t] = \mathbb{E}[(X_t - \mu)(X_t - \mu)'],\ \forall t \in [1, L]$,
3. $\Gamma(k) = \text{cov}(X_t, X_{t-k}) = \mathbb{E}[(X_t - \mu)(X_{t-k} - \mu)'],\ \forall t \in [1, L], \forall k \in \mathbb{N}$.

A time series is non-stationary if at least one of these properties is violated. That is, either the mean $\mu(t)$, the variance $\sigma^2(t)$, or the autocovariance $\Gamma(k, t)$ vary over time and thus are time-dependent.

Requirement 2: periodicities Realistic time series can also contain the phenomenon of periodicities. In a trend-free time series without a random component $\mathcal{X}_t = x_1, x_2, \ldots x_L,\ t \in [1, L]$, a periodicity with period η can be defined by $x_{t+\eta} = x_t,\ \forall t \in [1, L]$.

Since the random component R_t causes the time series to have constant unpredictable fluctuations, this definition is too strict. Therefore, we utilise the reoccurring autocovariance structure $\Gamma(\eta)$ between time series points separated by the period η (see [13]) to define a periodicity, i.e.

$$\Gamma(\eta) \approx \Gamma(2 \cdot \eta) \approx \ldots \approx \Gamma(\mathcal{P} \cdot \eta),\ \mathcal{P} \in \mathbb{N}. \tag{1}$$

Furthermore, we expect a noticeably different autocovariance between time series observations separated by κ, whereby κ is not a multiple of the period η, i.e.

$$|\Gamma(\eta) - \Gamma(\kappa)| \gg 0 : \kappa \neq \mathcal{P} \cdot \eta,\ \mathcal{P} \in \mathbb{N}. \tag{2}$$

In the case of trend-free time series, observations separated by the period η are additionally similar to each other, i.e.

$$x_t \approx x_{t+\eta} \approx x_{t+2\cdot\eta} \approx \ldots \approx x_{t+\mathcal{P}\cdot\eta},\ \mathcal{P} \in \mathbb{N}. \tag{3}$$

Therefore, a time series includes periodicities if a reoccurring autocovariance structure $\Gamma(\eta)$ is present.

3.2 Shortcomings of generation methods

This section explains why current generation methods cannot fulfil the requirements of non-stationarity and periodicities. We first explain the principles of generation methods before describing how to apply them to time series generation. We then point out shortcomings of these methods concerning non-stationarity and periodicities in generated time series.

Principles of generation methods Generation methods such as GANs [11], VAEs [15], and INNs [14] focus on describing a probability distribution P_X of a random variable $X : \Omega \rightarrow \mathbb{X}$, with Ω being a general probability space and $\mathbb{X}$ the realisation space. The underlying assumption is that the observed data $\mathbf{x} \in \mathbb{X}$ are realisations of the random variable $X \sim P_X$. Since P_X is often an intractable distribution, generation methods indirectly model the joint distribution $P_{X,Z}$ of X and a *latent* random variable $Z : \Omega \rightarrow \mathbb{Z}$ in the latent space $\mathbb{Z}$. Given the joint and latent distributions, P_X can be expressed as

$$P_X = \int P_{X|Z} P_Z \, dZ, \tag{4}$$

where $P_{X|Z}$ is the likelihood and P_Z the prior. If $P_{X|Z}$ and P_Z are tractable, this expression allows an exact calculation of P_X without knowledge of the intractable distribution P_X.

To be able to make use of (4), generation methods learn mappings for the generative process. Given an intractable distribution P_X in the realisation space, VAEs and INNs learn an encoding $f(X; \theta_1)$ from the sample distribution P_X to the distribution $P_{Z|X}$, where $P_{Z|X}$ is the probability distribution in the latent space given X and θ_1 are the trainable parameters. Thereby, regularisation is applied to ensure that $P_{Z|X}$ is a good approximation of P_Z and that P_Z is a tractable distribution in the latent space. Given P_Z, generation methods then learn a second mapping $g(Z; \theta_2)$ from P_Z to $P_{X|Z}$, where θ_2 are the trainable parameters.[3] Based on the learned probability distribution $P_{X|Z}$ and the known tractable distribution P_Z, generation methods finally apply (4) to determine an approximation of the sample distribution P_X.

Generating time series Unfortunately, time series are not realisations of a probability distribution P_X but of a time-dependent stochastic process $\{X_t\}$. Therefore, the aforementioned underlying assumption of generation methods does not hold. To still apply the principles of generation methods, one must account for the time-dependency of time series. One possibility to consider this time-dependency is to split a realised time series sample $\mathbf{x} \in \mathbb{X}$ into, for example, N sequential segments $\mathbf{x} = (\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^N)$ of arbitrary length, because a stochastic process is defined as a series of random variables.[4] Analogously, one can aggregate multiple generated time series segments $\hat{\mathbf{x}}^i,\ i \in [1, N]$ to include time-dependency in a generated time series longer than one segment. To generate these time series segments $\hat{\mathbf{x}}^i,\ i \in [1, N]$, one draws multiple samples $\mathbf{z}^i,\ i \in [1, N]$ from P_Z and uses the mapping $g(\mathbf{z}^i; \theta_2) = \hat{\mathbf{x}}^i$.

Shortcoming 1: non-stationarity in generated time series In the generative process, the used samples $\mathbf{z}^i,\ i \in [1, N]$ from the latent space are realisations of the random variable Z

[3] Note that, unlike VAEs and INNs, GANs learn the mapping g directly via the *generator*. However, this does not affect the problem definition.

[4] While we refer to a time series segment i with x^i, we use a subscript to denote an entry j of the segment i, i.e. x^i_j.

with the known distribution P_Z. Similarly, the generated time series segments $\hat{\mathbf{x}}^i$, $i \in [1, N]$ are realisations of the random variable transformation $g(Z; \theta_2)$. According to the so-called law of the unconscious statistician (LOTUS) [21], this transformed random variable $g(Z; \theta_2)$ has the expected value

$$\mathbb{E}[g(Z; \theta_2)] = \int g(\mathbf{z}; \theta_2) P_Z \, d\mathbf{z}. \tag{5}$$

Assuming that the mapping $g(Z; \theta_2)$ only depends on the random variable Z and the fixed learned parameters θ_2, all generated time series segments also have the same expected value, i.e.

$$\mathbb{E}[\hat{\mathbf{x}}^i] = \mathbb{E}[g(\mathbf{z}^i; \theta)], \quad \forall\, i \in [1, N]. \tag{6}$$

The same argument applies to the variance. Since the variance of a random variable X is defined as $\sigma_X^2 = \mathbb{E}[X^2] - (\mathbb{E}[X])^2$, one can again use LOTUS [21] to show that the variance is also the same for all generated time series segments (for details see Appendix B).

The equal variance across all generated time series segments has implications for the autocovariance. Since all generated time series segments are realisations of the same random variable transformation $g(Z; \theta_2)$, the autocovariance that is defined for two random variables simplifies to the variance, i.e.

$$\text{cov}(g(Z; \theta_2), g(Z; \theta_2)) = \sigma^2_{g(Z;\theta_2)}, \tag{7}$$

which, as previously shown, is the same for all generated time series segments.

Altogether, existing generation methods cannot vary the statistical properties of generated time series segments. Therefore, these methods cannot control non-stationarity in generated time series[5] and do not fulfil the previously defined Requirement 1.

Shortcoming 2: periodicities in generated time series As previously shown, the autocovariance structure is the same for all generated time series segments in a generative process. As a result, existing generation methods cannot create reoccurring and different autocovariance structures. Therefore, these methods cannot control periodicities in generated time series and thus do not fulfil the previously defined Requirement 2.

[5]Existing methods, such as a recurrent neural network with teacher forcing, can generate non-stationary time series segments of arbitrary length if trained on non-stationary data sets. However, such methods can only reproduce the non-stationarity available in the training data, and this non-stationarity cannot be controlled. Additionally, all of the time series segments generated by such a method contain only the non-stationarities identical to those in the training data. Therefore, it is impossible to control the non-stationarity across multiple time series segments to generate time series with desired non-stationarities.

4 Controlling non-stationarity and periodicities in generated time series

This section presents a novel approach for controlling non-stationarity and periodicities in generated time series. Firstly, we formally describe our approach to time series generation that fulfils Requirements 1 and 2. To show the practical viability of our approach, we then introduce a conditional Invertible Neural Network (cINN) as an exemplary implementation.

4.1 Formal solution

This section formally describes the novel approach for generating time series with controlled non-stationarity and periodicities whilst overcoming the previously presented Shortcomings 1 and 2 of existing generation methods. We firstly explain an assumption to guarantee the existence of time series segments with non-stationarity and periodicities. Afterwards, we detail how our approach uses and combines calendar and statistical information to control non-stationarity and periodicities in generated time series.

Existence guarantee To guarantee the existence of time series segments with controlled non-stationarity and periodicities, we assume the encoding $f(X; \theta_1)$ to be a bijective mapping, where $g(Z; \theta_2)$ is the inverse function of $f(X; \theta_1)$, i.e. $f^{-1}(\cdot; \theta) := g(\cdot; \theta)$ and $\theta = \theta_1 = \theta_2$. This mapping guarantees that the image of the concatenation $\text{Im}((f \circ f^{-1})(\mathbb{X})) = \mathbb{X}$ includes the entire realisation space $\mathbb{X}$. Therefore, for all possible samples from the latent space, a corresponding time series segment in the realisation space exists, i.e.

$$\forall \mathbf{z} \sim P_Z \; \exists \hat{\mathbf{x}} : f^{-1}(\mathbf{z}; \theta) = \hat{\mathbf{x}}. \tag{8}$$

Besides guaranteeing the existence of a corresponding time series segment for all possible samples, this bijective mapping allows us to include additional inputs in the mapping. With these inputs, we can vary the properties of each generated time series segment:

Calendar information The first additional input is calendar information such as the hour, day of week, month, or year. This information is implicitly present in a time series as a realisation of a stochastic process $\{X_t\}$ but is currently not considered by generation methods. To include this information, we use the calendar information $\mathbf{d}$ as an additional input to our mapping, i.e.

$$f : \mathbb{X} \to \mathbb{Z}, \; \mathbf{x}^i \mapsto f(\mathbf{x}^i; \mathbf{d}, \theta) = \mathbf{z}^i. \tag{9}$$

Considering calendar information enables us to generate time series segments with varying calendar information, even though these segments are generated from samples

$\mathbf{z}^i$ that are realisations of the same random variable Z. However, solely including calendar information does not allow us to vary the statistical properties of each generated time series segment.

Statistical information To vary the statistical properties of each generated time series segment, we consider statistical information such as mean and variance as a second additional input. Therefore, we add statistical information $\mathbf{s}$ to our mapping, i.e.

$$f : \mathbb{X} \to \mathbb{Z}, \ \mathbf{x}^i \mapsto f(\mathbf{x}^i; \mathbf{d}, \mathbf{s}, \theta) = \mathbf{z}^i. \tag{10}$$

Combining calendar and statistical information Based on calendar and statistical information as inputs, we are able to generate time series segments with varying statistical properties dependent on the calendar information. For example, as a result, the mean of the transformed random variable $f^{-1}(Z; \mathbf{d}, \mathbf{s}, \theta)$ is dependent on calendar and statistical information, i.e.

$$\mathbb{E}[f^{-1}(Z; \mathbf{d}, \mathbf{s}, \theta)] = \int f^{-1}(\mathbf{z}; \mathbf{d}, \mathbf{s}, \theta) P_Z \, d\mathbf{z}. \tag{11}$$

Similarly, the variance and the autocovariance also depend on the calendar and statistical information. Since the calendar and statistical information are included as additional inputs to the mapping, we can effectively control the statistical properties of the generated time series segments for the calendar information. This interplay between calendar and statistical information enables us to include and control non-stationarities and periodicities in the generated time series. Therefore, combining calendar and statistical information as additional inputs allows us to fulfil Requirements 1 and 2 mentioned above.

4.2 Exemplary implementation

This section presents the exemplary cINN-based implementation of our novel approach for generating time series with controlled non-stationarity and periodicities. After a brief overview of its architecture, we describe its training and generative process.

Architecture To realise the previously defined bijective mapping that considers calendar and statistical information, we use a conditional Invertible Neural Network (cINN). To implement the exemplary cINN, we use FrEIA[6] and PyTorch [18]. As shown in Fig. 1, the used cINN comprises 15 subsequent invertible coupling layers, one conditioning network q, and their trainable parameters θ.[7]

As coupling layers, we use the conditional affine coupling layer proposed by Ardizzone et al. [2], which extends RealNVP by Dinh et al. [4]. Each coupling layer contains two subnetworks. The architecture of the subnetworks is shown in Table 1. As inputs, each coupling layer takes the output of the previous coupling layer and the conditional information $\mathbf{c}$. This conditional information is the calendar information $\mathbf{d}$ and the statistical information $\mathbf{s}$ mentioned above, both encoded by a separate conditioning network q.

The architecture of the conditioning network q is also described in Table 2. The input of q are calendar and statistical information. The calendar information $\mathbf{d}$ are information for all the time stamps for which a value should be generated. This information includes the hour of the day encoded as a sine function $\sin(\pi \cdot \text{hour}/23)$ and cosine function $\cos(\pi \cdot \text{hour}/23)$, the month of the year as sine function $\sin(\pi \cdot \text{month}/11)$ and cosine function $\cos(\pi \cdot \text{month}/11)$, and the weekend as a Boolean. The statistical information $\mathbf{s}$ for the training is the mean of the time-series sample, i.e. $\hat{\mu}^i = \mathbb{E}[\hat{\mathbf{x}}^i]$. In the generation, the mean is the desired mean of the generated sample.

Training To train the used cINN, we extend the training proposed by Ardizzone et al. [2] with a statistical loss. The training is generally based on a maximum likelihood optimisation using the *change of variable formula*

$$P_X(\mathbf{x}; \mathbf{c}, \theta) = P_Z(f(\mathbf{x}; \mathbf{c}, \theta)) \left| \det \frac{\partial f}{\partial \mathbf{x}} \right| \tag{12}$$

with the Jacobian matrix $\partial f / \partial \mathbf{x}$ [2]. To implement this optimisation, we select the standard normal distribution as the latent distribution, choose Gaussian priors, and apply Bayes' theorem. The result is the maximum likelihood loss, i.e.

$$\mathcal{L}_{\text{ml}} = \mathbb{E}^i \left[\frac{\| f(\mathbf{x}^i; \mathbf{c}^i, \theta) \|_2^2}{2} - \log | J^i | \right] + \lambda \| \theta \|_2^2, \tag{13}$$

where J^i is the Jacobian corresponding to the i-th sample [2]. In addition to $\mathcal{L}_{\text{ml}}$ and as an extension of Ardizzone et al. [2], we also minimise the difference between the desired mean and the mean of the generated time series segment, i.e.

$$\mathcal{L}_s = \sqrt{(\mathbf{s}_\mu - \frac{1}{n} \sum_j \hat{\mathbf{x}}_j)^2}, \tag{14}$$

where $\hat{\mathbf{x}}_j$ is an entry of the generated time series segment and $\mathbf{s}_\mu$ is the desired mean in the statistical information. Therefore, the overall loss used to train the cINN is

$$\mathcal{L} = \mathcal{L}_{\text{ml}} + \lambda \mathcal{L}_s, \tag{15}$$

where λ is a hyperparameter weighting the influence of the statistical loss.

We train the cINN for 200 epochs with this defined loss using the ADAM optimiser.

[6] https://github.com/VLL-HD/FrEIA

[7] https://github.com/KIT-IAI/ControllableTimeSeriesGeneration

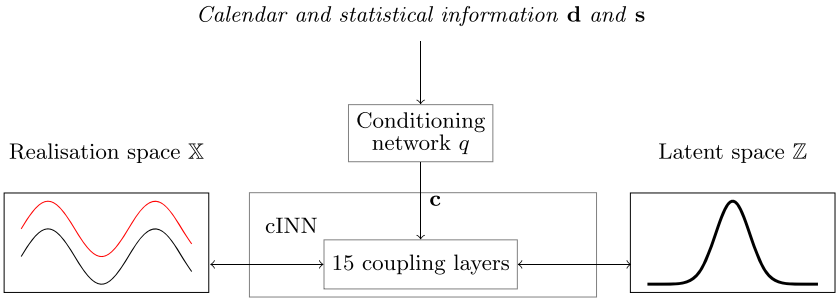


Fig. 1 Our cINN realises a bijective mapping from the realisation space $\mathbb{X}$ to the latent space $\mathbb{Z}$. It comprises 15 subsequent coupling layers and one conditioning network q. The conditioning network processes the calendar information $\mathbf{d}$ and statistical information $\mathbf{s}$ to provide the conditional input $\mathbf{c}$ for each coupling layer

Generative process The cINN learns a mapping from the realisation space to the latent space in the described training. Since this mapping is bijective, we can use the inverse direction as a generative process: To generate a time series segment $\hat{\mathbf{x}}^i$, we firstly draw a sample $\mathbf{z}^i$ from the random variable Z, choose the desired calendar and statistical information $\mathbf{d}^i$ and $\mathbf{s}^i$, and apply the conditioning network to encode $\mathbf{c}^i = q(\mathbf{d}^i, \mathbf{s}^i)$. With these inputs, we use the trained cINN in the inverse direction and obtain a time series segment $\hat{\mathbf{x}}^i = f^{-1}(\mathbf{z}^i; \mathbf{c}^i, \theta)$.

To create a time series longer than one segment, we utilise the calendar information previously included in the mapping to aggregate the generated time series segments $\hat{\mathbf{x}}^i$. More specifically, we take advantage of the fact that the calendar information $\mathbf{d^i}$ of adjacent time series segments overlap: For adjacent segments, similar and related calendar information form the input of the conditional network. This input ensures that the sample distribution is conditioned on similar and related calendar information. This way, generated time series segments $\hat{\mathbf{x}}^i$ with adjacent calendar information $\mathbf{d^i}$ are related, and we can calculate the median over all entries of a certain time t with

$$\hat{x}_t = \text{Median}(\{\hat{\mathbf{x}}^i_j \mid \mathbf{d}^i_j \rightarrow t\}) \, \forall t \in [1, L], \tag{16}$$

where the condition $\mathbf{d^i_j} \rightarrow t$ ensures that only entries of the time series segments $\hat{\mathbf{x}}^i$ with the same time t are aggregated.

5 Experiments

This section empirically evaluates the cINN mentioned above as an exemplary implementation of our proposed approach for generating time series with controlled non-stationarity and periodicities. After introducing the used data sets, we demonstrate how the cINN generates time series with controlled non-stationarity and periodicities. Finally, we compare the cINN to state-of-the-art time series generation benchmarks to assess the general quality of the generated time series. At this point, we also perform an ablation study to determine the influence of conditional information and the statistical loss.

Table 1 Implementation details of the cINN regarding the used subnetwork

	Subnetwork
1	Dense neurons: 16; activation: tanh
2	Dense neurons: Segment length; activation: linear

5.1 Data sets

In order to comprehensively evaluate our proposed approach and the exemplary implementation, we aim to select data sets from different domains. Since data sets commonly used to evaluate time series generation methods are not publicly available or do not contain calendar information (e.g. [8, 22, 23]), we select three publicly available time series data sets. The selected data sets[8] all contain univariate time series with calendar information but differ in their temporal resolution, variance, and periodicities:

Energy The first data set has an hourly temporal resolution, a low variance, and contains daily, weekly, and yearly periodicities. It consists of the electricity consumption of one client from the UCI Electricity Load Dataset [7].

Information The second data set has a daily temporal resolution, a medium variance, and contains no periodicities. It comprises the number of daily views of the Wikipedia page "Hitchhiker's Guide to the Galaxy (video game)" from the Web Traffic Time Series Forecasting Dataset.[9]

Mobility The third data set also has an hourly temporal resolution, a high variance, and contains daily and yearly periodicities. It contains the hourly records of rented bikes from the UCI Bikesharing Dataset [7, 9].

5.2 Generated time series with controlled non-stationarity and periodicities

To demonstrate how the cINN creates time series with controlled non-stationarity and periodicities as described

[8] Appendix C provides further information on these data sets and their preprocessing.

[9] https://www.kaggle.com/c/web-traffic-time-series-forecasting/data

Table 2 Implementation details of the cINN regarding the used conditioning network q

	Conditioning network q
1	Dense neurons: 2; activation: tanh
2	Dense neurons: 1; activation: linear

in Requirements 1 and 2, we first generate a time series with controlled non-stationarity and second a time series with controlled periodicities for each data set. We first define the calendar and statistical information used in the generation for each time series. Afterwards, we evaluate the generated time series by visually inspecting them and calculating statistics corresponding to the requirements of non-stationarity and periodicities, respectively.

Controlled non-stationarity To demonstrate controlled non-stationarity, we define calendar and statistical information for the cINN as follows. To determine the calendar information, we choose the years from 2011 to 2013 for the three data sets. As statistical information for the energy and information data sets, we specify a mean with a linear trend starting from 75 and ending at 125 and a yearly sinusoidal periodicity with an amplitude of 15. For the mobility data set, we specify a mean with a linear trend starting from 100 and ending at 250 and a yearly sinusoidal periodicity with an amplitude of 25.

The time series generated based on the selected calendar and statistical information are shown in Fig. 2 for the three data sets. For all data sets, the generated time series accurately reflect the specified mean, including the trend and the sinusoidal shape, while retaining the previously described original characteristics of the respective data set concerning variance and periodicities.

To further examine the generated time series according to the previously defined requirement of non-stationarity, we determine corresponding statistics. On four three-month cut-outs, we compare the mean, the variance, and the autocovariance with a fixed lag of half a year for two generated time series of each data set: one with a constant mean as statistical information and one with a controlled mean as statistical information to control the non-stationarity.

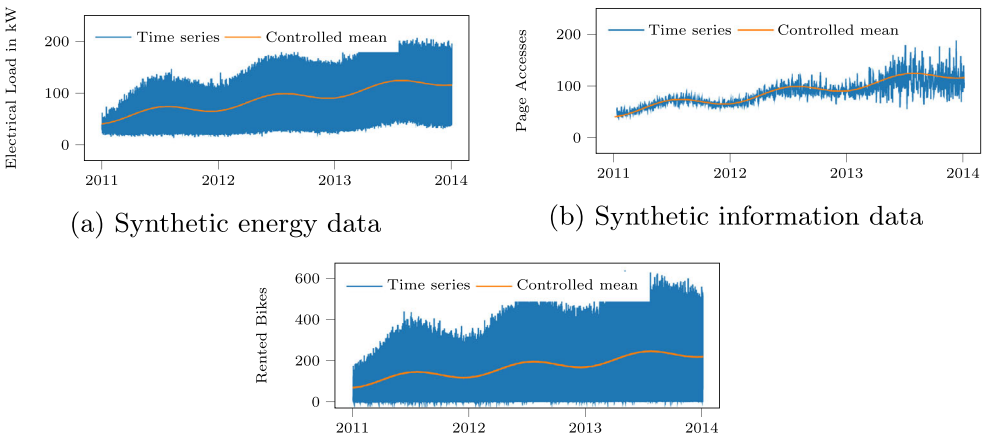
For both generated time series of each data set, Table 3 presents the mean, the variance, and the autocovariance of the four cut-outs. For all data sets, the calculated statistics of the generated time series with constant mean are similar for the considered cut-outs. In contrast, the calculated statistics of the generated time series with controlled non-stationarity are different for the considered cut-outs of each data set and thus are all time dependent. This shows that the generated time series is non-stationary according to Requirement 1.

Controlled periodicities To demonstrate controlled periodicities, we also define calendar and statistical information for the cINN. To determine the calendar information, we again choose the years from 2011 to 2013. As statistical information for the three data sets, we specify a constant mean of 150 and a yearly sinusoidal periodicity with an amplitude of 50.

For the three data sets, Fig. 3 shows the resulting generated time series and the used mean. For all data sets, the generated time series follows the periodicity defined by the mean and retains the variance and periodicities of the respective data set.

To further analyse the generated time series according to the previously defined requirement of periodicities, we examine the autocovariance structure of the generated time series. More specifically, for the generated time series of each data set, we calculate the autocovariance of the first three months and the rest of the time series.

Figure 4 shows the yearly and the daily autocovariance of the generated time series of each data set. Note that no daily autocovariance structure exists for the information



(a) Synthetic energy data

(b) Synthetic information data

(c) Synthetic mobility data

Fig. 2 To demonstrate controlled non-stationarity, a time series is generated using defined calendar and statistical information. For the energy, the information, and the mobility data sets, the generated time series is shown in blue and the used controlled mean in orange

Table 3 Statistics according to Requirement 1, i.e. mean, variance, and autocovariance, for the generated time series with constant mean as statistical information and with a controlled mean as statistical information to control the non-stationarity

		Energy			Information			Mobility		
	Cut-out	μ	σ^2	cov	μ	σ^2	cov	μ	σ^2	cov
Constant Mean	03/11 - 07/11	71	1048	1005	76	37	–1	182	183	16107
	12/11 - 02/12	71	1032	990	76	37	–4	183	182	16174
	03/12 - 07/12	71	1044	994	76	36	5	182	182	16039
	12/12 - 02/13	71	1022	979	76	40	1	182	182	16026
Controlled Mean	03/11 - 07/11	65	858	761	65	48	–7	191	143	12919
	12/11 - 02/12	64	791	1107	65	24	2	134	235	14754
	03/12 - 07/12	91	1667	1582	89	99	8	222	174	18062
	12/12 - 02/13	90	1594	1626	89	88	28	166	262	19846

data set because this data set has a daily resolution and thus no daily periodicities. The autocovariance has a daily and yearly reoccurring structure for all generated time series. According to Requirement 2, the generated time series thus contains periodicities. These periodicities are more regular for the energy and mobility data sets than for the information data set.

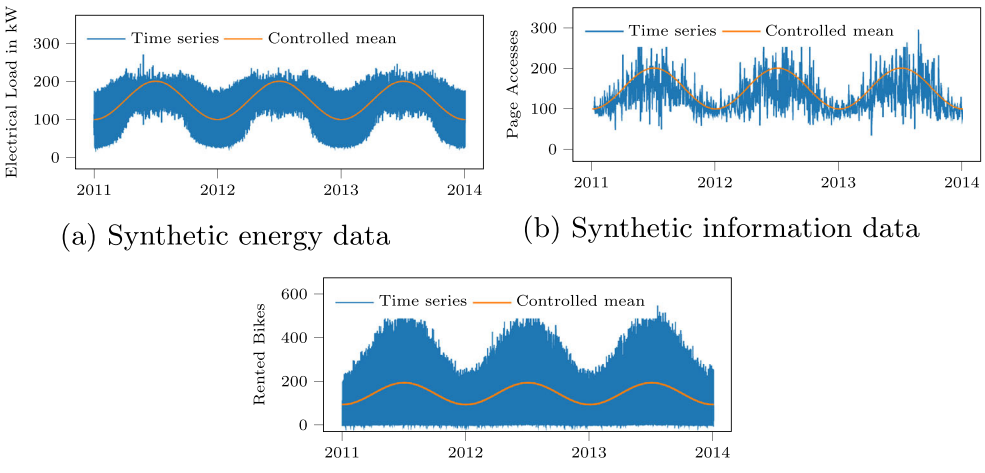
5.3 Quality of generated time series

To assess the general quality of the generated time series, we compare the cINN to state-of-the-art benchmarks on the three selected data sets and perform an ablation study regarding the influence of the conditional information and the statistical loss. We first introduce the three used evaluation metrics and the six benchmarks before presenting the benchmarking and the ablation study results. For this evaluation, we run each generation method three times. The cINN uses the calendar information and a calculated rolling mean of the respective data set for the generation.

5.3.1 Metrics

To assess the quality of the generated time series segments, we use three metrics. Firstly, we apply the train-synthetic-test-real evaluation [8] to obtain a predictive score. The predictive score measures the usefulness of the generated time series. Secondly, we make use of a discriminator to obtain a discriminative score. With the discriminative score, we examine the distinguishability of the generated and the original time series [23]. Thirdly, we measure the training time of the generation methods to assess their computational cost. In the following, we detail all metrics.

Predictive score For the train-on-synthetic-test-on-real evaluation [8], we train a predictive model on the generated

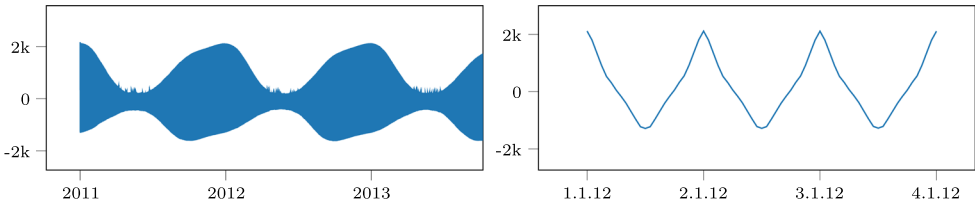


(a) Synthetic energy data

(b) Synthetic information data

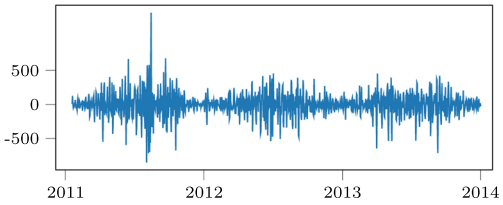
(c) Synthetic mobility data

Fig. 3 To demonstrate controlled periodicities, a time series is generated using defined calendar and statistical information. For the energy, the information, and the mobility data sets, the generated time series is shown in blue and the used controlled mean in orange

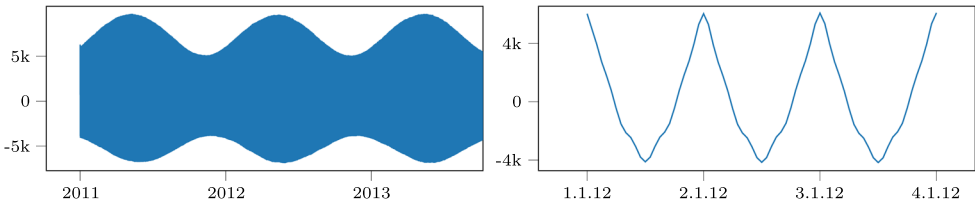


(a) Yearly autocovariance of synthetic energy data

(b) Daily autocovariance of synthetic energy data



(c) Yearly autocovariance of synthetic information data



(d) Yearly autocovariance of synthetic mobility data

(e) Daily autocovariance of synthetic mobility data

Fig. 4 The yearly and daily autocovariance structure according to Requirement 2 of the generated time series with controlled periodicities based on the three data sets. The autocovariance is calculated between the first three months and all other three months segments of the time series. Note that the information data has a daily resolution and thus we only calculate the yearly autocovariance

time series and test the model on the original time series to obtain a predictive score. As the predictive score, we use the mean absolute error (MAE), the mean absolute scaled error (MASE), and the root mean squared error (RMSE).

The architecture of the predictive model is a three-layered fully connected neural network with ten hidden neurons. The model is designed to work with time series segments of 24 hours. Thereby, the first 23 hours are used to forecast the last value. We use a ReLU activation function for the hidden layers and a linear activation function for the output layer.

To implement the predictive model, we use pyWATTS[10] [12] with Keras [3]. We train the implemented predictive model for 100 epochs and apply early stopping during the training process. To obtain more robust results, we train the predictive model five times on the generated time series of each data set.

Discriminative score For the evaluation by the discriminator [23], we merge the generated and the original time series. We label the generated time series with 0 and the original time series with 1. Afterwards, the merged data set is split into a training (70%) and test set (30%). The discriminative model is then trained on the training set and the discriminative score is calculated on the test set. We use $|\ \text{Accuracy} - 0.5\ |$ as discriminative score, where Accuracy refers to the performance of the discriminative model on the test set.

The architecture of the discriminative model is a three-layered fully connected neural network. The network uses tanh as an activation function for the hidden layers and softmax for the output layer.

To implement the discriminative model, we also use pyWATTS [12] with Keras [3]. We train the implemented discriminative model on the CPU for ten epochs using the ADAM optimiser and the binary cross-entropy loss. We run the training five times on each generated time series to obtain more robust results.

Computational cost To evaluate the computational cost, we measure the training time of the evaluated generation methods in seconds. For this, we measure the time required for training of all generation methods three times and calculate the respective average to obtain robust results. For comparable results, we perform the training on the same hardware. As hardware, we use an off-the-shelf Lenovo Thinkpad T490 laptop with an Intel i7-8565U processor and 16 GB of RAM.

[10] https://github.com/KIT-IAI/pyWATTS

5.3.2 Benchmarks

As benchmarks, we select six generation methods. As state-of-the-art benchmark generation methods, we consider COT-GAN [22], RGAN and RCGAN [8], and TimeGAN [23]. As baseline generation methods, we additionally consider a simple GAN and a simple VAE.

COT-GAN For the implementation of COT-GAN [22], we use the publicly available source code.[11] We only adapt the data loading functionality to apply COT-GAN to our data.

RGAN and RCGAN For the implementation of RGAN and RCGAN [8], we use the publicly available source code.[12] For our experiments, we only adapt the setting file `test.txt` and the `data_utils.py` to load our data and ensure the training of RGAN and RCGAN on our data is robust. For RCGAN, we use the same calendar and statistical information as the implemented cINN.

TimeGAN To implement TimeGAN [23], we use the publicly available source code.[13] Compared to the source code, we adapt the data loader of TimeGAN to apply it to our data sets.

GAN To implement the GAN, we use Keras [3] and Tensorflow [1]. Table 4 provides details on the generator and the discriminator of the implemented GAN. The dimension of the random noise input of the implemented generator is 32. We train the GAN for 200 epochs and use the binary cross-entropy as the loss function for the discriminator.

VAE For the implementation of the VAE, we use Keras [3] and Tensorflow [1]. Table 5 details the encoder and the decoder of the implemented VAE. The latent dimension of the VAE is 2. When training the model, we use the Kullback-Leibler distance and the RMSE as the loss functions. We train the VAE for 2000 epochs and apply an early stopping with a patience of 10 epochs.

5.3.3 Benchmarking results

With the cINN and the six selected benchmark methods, we generate time series segments for the three selected data sets. We compare the related predictive score, discriminative score, and training time in the following.

[11] https://github.com/tianlinxu312/cot-gan

[12] https://github.com/ratschlab/RGAN

[13] https://github.com/jsyoon0823/TimeGAN

Table 4 Implementation details of the GAN used as benchmark. Note that the stride is 1 for each convolutional layer

	Generator	Discriminator
1	Dense neurons: 64	Conv1d filters: 32; kernel size: 3
2	Dense neurons: 64	Conv1d filters: 16; kernel size: 3
3	Dense neurons: 64	Conv1d filters: 8; kernel size: 3
4		Dense neurons: 1

Predictive score For all three data sets, the average, minimum, and maximum predictive scores of the cINN and the six benchmark methods are shown in Table 6. We also report a predictive model trained on the original time series as original data in the table, which corresponds to a train-on-real-test-on-real evaluation.

The cINN outperforms all benchmark methods except COT-GAN on the three selected data sets. While the cINN performs better than COT-GAN on the mobility data set, it is on par with COT-GAN on the energy and information data sets. The cINN is also on par with the GAN for the information data set. Moreover, the cINN is almost on par with the predictive model trained on the original time series. Considering the MASE, we also observe that the performance of the generation methods depends on the data set. More specifically, most generation methods obtain the best relative predictive score –measured by the MASE– on the mobility data set. However, most generation methods achieve the worst predictive score on the mobility data set compared to the original data.

Discriminative score The average, minimum, and maximum discriminative scores of the cINN and the six benchmark methods on the three data sets are shown in Table 7. We also observe that the cINN outperforms all benchmark methods except COT-GAN for the discriminative score. However, while the cINN performs better than COT-GAN on the information and mobility data sets, it performs slightly worse on the energy data set. Moreover, we observe that the discriminative score of each generation method is more similar across the different data sets than the predictive score.

Computational cost The computational cost of the cINN and the six benchmark methods in terms of the average training time is presented in Table 8. Overall, we observe that the simple generation methods, namely the GAN and the VAE, have the lowest training times. However, we observe that the cINN has the lowest training times when only considering state-of-the-art generation methods. Note

Table 5 Implementation details of the VAE used as benchmark. Note that the stride is 1 for each convolutional layer

	Encoder	Decoder
1	Conv1D filters: 12; Kernel size: 2	Dense neurons: 24
2	Conv1D filters: 64; Kernel size: 2	Dense neurons: 96
3	Conv1D filters: 64; Kernel size: 3	Reshape shape: horizon $\times$ 4
4	Dense neurons: 128	Conv1DTranspose: filters: 32; kernel size: 3
5	Dense for μ neurons: 2	Conv1DTranspose: filters: 16; kernel size: 4
6	Dense for σ^2 neurons: 2	Conv1DTranspose: filters: 8; kernel size: 5
7		Conv1D filters: 1; kernel size: 2

that the training times of the generation methods on the information data set are shorter due to the smaller length of the related time series segments.

5.3.4 Ablation study

To determine the influence of the conditional information comprising calendar and statistical information as well as the statistical loss defined in (14), we perform an ablation study for the predictive and the discriminative scores. Based on the three data sets, we compare the cINN using calendar and statistical information and the statistical loss (cINN) to a cINN using only the calendar and statistical information (cINN Stats + Cal). Additionally, we compare cINNs using only statistical information (cINN Stats), calendar information (cINN Cal), and no information (INN).

Table 6 The average, minimum, and maximum predictive scores of the cINN and the six benchmark methods on the energy, information, and mobility data sets. For comparison, a predictive model trained on the original time series is additionally reported as original data. Lower the better

	Method	Energy	Information	Mobility
MAE	cINN	12 (11 - 13)	14 (13 - 15)	47 (42 - 58)
	COT-GAN	11 (11 - 12)	14 (13 - 16)	70 (65 - 74)
	RGAN	17 (15 - 20)	100 (16 - 251)	176 (129 - 257)
	RCGAN	18 (12 - 30)	85 (75 - 91)	162 (101 - 268)
	TimeGAN	16 (12 - 50)	16 (14 - 18)	82 (74 - 91)
	GAN	26 (18 - 32)	14 (13 - 16)	174 (118 - 240)
	VAE	32 (23 - 47)	30 (16 - 58)	196 (115 - 338)
	Original Data	12 (11 - 12)	14 (13 - 14)	40 (34 - 48)
MASE	cINN	0.94 (0.87 - 1.03)	0.83 (0.78 - 0.91)	0.43 (0.38 - 0.53)
	COT-GAN	0.88 (0.85 - 0.92)	0.84 (0.78 - 0.97)	0.64 (0.60 - 0.68)
	RGAN	1.33 (1.14 - 1.55)	6.01 (0.93 - 15.05)	1.63 (1.19 - 2.37)
	RCGAN	1.42 (0.93 - 2.32)	5.08 (4.53 - 5.48)	1.79 (1.05 - 3.09)
	TimeGAN	1.24 (0.95 - 3.89)	0.95 (0.83 - 1.10)	0.76 (0.68 - 0.84)
	GAN	2.01 (1.36 - 2.46)	0.85 (0.78 - 0.94)	1.59 (1.08 - 2.20)
	VAE	2.49 (1.76 - 3.61)	1.81 (0.96 - 3.47)	1.79 (1.05 - 3.09)
	Original data	0.90 (0.85 - 0.96)	0.81 (0.78 - 0.85)	0.37 (0.31 - 0.44)
RMSE	cINN	17 (16 - 18)	18 (17 - 20)	68 (61 - 81)
	COT-GAN	16 (16 - 17)	18 (17 - 22)	108 (103 - 113)
	RGAN	23 (21 - 27)	104 (19 - 254)	221 (163 - 306)
	RCGAN	24 (16 - 37)	88 (79 - 95)	217 (138 - 348)
	TimeGAN	22 (17 - 64)	21 (18 - 24)	132 (120 - 148)
	GAN	34 (23 - 42)	19 (17 - 21)	226 (161 - 303)
	VAE	42 (30 - 59)	35 (20 - 62)	261 (172 - 429)
	Original Data	16 (15 - 17)	17 (17 - 18)	58 (52 - 69)

Table 7 The average, minimum, and maximum discriminative scores of the cINN and the six benchmark methods on the energy, information, and mobility data sets. Lower the better

Method	Energy	Information	Mobility
cINN	0.05 (0.03 - 0.09)	0.02 (0.00 - 0.06)	0.03 (0.01 - 0.05)
COT-GAN	0.03 (0.00 - 0.08)	0.03 (0.00 - 0.08)	0.10 (0.07 - 0.12)
RGAN	0.30 (0.16 - 0.42)	0.33 (0.06 - 0.49)	0.32 (0.30 - 0.35)
RCGAN	0.21 (0.11 - 0.31)	0.38 (0.11 - 0.48)	0.24 (0.16 - 0.33)
TimeGAN	0.13 (0.04 - 0.21)	0.06 (0.00 - 0.15)	0.08 (0.04 - 0.13)
GAN	0.46 (0.44 - 0.46)	0.23 (0.04 - 0.38)	0.39 (0.28 - 0.46)
VAE	0.33 (0.24 - 0.38)	0.09 (0.00 - 0.17)	0.31 (0.30 - 0.33)

Predictive score The predictive scores of the different cINNs for the three data sets are shown in Table 9. We observe that the considered cINNs generally perform similarly on the three data sets. While the INN barely achieves the best performance on the energy data set and the cINN using only calendar information achieves the best results on the mobility data set, all cINNs are on par on the information data set.

Discriminative score The discriminative scores of the different cINNs for the three data sets are presented in Table 10. Our observation is that the considered cINNs perform similarly on the energy and information data set. However, the cINNs using statistical information perform better on the mobility data set than the cINNs that do not consider this information.

6 Discussion

This section discusses the previously reported results of the experiments, limitations, and benefits of the cINN as the exemplary implementation of the proposed approach.

In the experiments, we observe that, based on defined conditional information, the cINN can generate time series with controlled non-stationarity and periodicities while retaining the characteristics of the original data set. Furthermore, the cINN outperforms or is on par with the selected benchmark generation methods regarding the predictive and discriminative scores. Also, the cINN requires the lowest training time of the considered state-of-the-art generation methods, probably due to the cINN's lower number of parameters or non-recurrent architecture. Additionally, in the ablation study, we observe that considering calendar and statistical information as conditional information only partly influences the predictive and discriminative score. From these observations, we conclude that the cINN, as an exemplary implementation of the proposed approach, can control the non-stationarity and the periodicities of generated high-quality time series with arbitrary length.

Table 8 The average training time in seconds of the cINN and the six benchmark methods for the three selected data sets

Method	Energy	Information	Mobility
cINN	1130	45	1129
COT-GAN	3586	1991	3841
RGAN	6801	293	3325
RCGAN	1422	307	1379
TimeGAN	18086	5783	23334
GAN	524	358	499
VAE	48	4	39

Despite these promising results, we note that the performed experiments may have limitations. One limitation could be that the selected data sets only have an hourly or daily resolution and only contain moderate variations. Another limitation might be that we do not evaluate how the described method for aggregating generated time series segments affects the performance of our approach. Furthermore, we only use univariate time series in our evaluation, even though our approach can be extended to multivariate time series.

Concerning the proposed approach, one limitation could also be the required bijective mapping realised by the cINN in the exemplary implementation. While using the bijective mapping guarantees the existence of all generated time series segments, we assume that it is not a necessary requirement for the proposed approach. Therefore, the proposed approach should also be effective with other generative mappings without the guaranteed existence of the generated time series. These mappings could approximate the inverse function, as in VAEs, or be trained by a discriminator, as in GANs. Extending our method to these generative mappings could lead to a more general framework to control non-stationarity and periodicities when generating time series. Another limitation could be that the proposed approach requires calendar information.

Table 9 Ablation study comparing different cINNs with respect to the average, minimum, and maximum predictive score. Lower the better

	Method	Energy	Information	Mobility
MAE	cINN	12 (11 - 13)	14 (13 - 15)	47 (42 - 58)
	cINN Stats + Cal	12 (11 - 15)	14 (13 - 15)	46 (41 - 50)
	cINN Stats	13 (12 - 13)	14 (13 - 15)	50 (44 - 54)
	cINN Cal	12 (12 - 13)	14 (13 - 17)	42 (39 - 48)
	INN	11 (10 - 13)	14 (13 - 19)	46 (41 - 50)
MASE	cINN	0.94 (0.87 - 1.03)	0.83 (0.78 - 0.91)	0.43 (0.38 - 0.53)
	cINN Stats + Cal	0.94 (0.88 - 1.18)	0.83 (0.79 - 0.89)	0.46 (0.39 - 0.54)
	cINN Stats	0.98 (0.92 - 1.03)	0.84 (0.80 - 0.89)	0.46 (0.40 - 0.49)
	cINN Cal	0.95 (0.90 - 1.01)	0.84 (0.79 - 1.00)	0.39 (0.36 - 0.44)
	INN	0.87 (0.79 - 0.99)	0.84 (0.77 - 1.11)	0.42 (0.37 - 0.46)
RMSE	cINN	17 (16 - 18)	18 (17 - 20)	68 (61 - 81)
	cINN Stats + Cal	17 (16 - 20)	18 (17 - 19)	75 (65 - 87)
	cINN Stats	17 (16 - 18)	18 (17 - 20)	75 (66 - 80)
	cINN Cal	17 (16 - 17)	18 (17 - 22)	62 (58 - 70)
	INN	16 (15 - 17)	18 (17 - 24)	67 (61 - 71)

Although this calendar information is present in a wide range of real-world time series, some time series, such as audio time series, do not contain such information. In addition to calendar information, the proposed approach also considers statistical information to generate time series with desired properties. In real-world applications, these desired properties may only be partially known and thus may need to be approximated.

However, given calendar and statistical information, the proposed approach enables controlling non-stationarity and periodicities in generated time series in many real-world applications, especially where real-world time series are non-existent, only partly available, not usable due to privacy concerns, or expensive to measure. In these cases, the proposed approach allows the generation of specific and diverse scenarios with non-stationarity and periodicities in time series. These scenarios could then be used to investigate unusual phenomena and various applications such as forecasting and imputation. Hence, our approach noticeably extends the capabilities of existing time series generation methods and offers new opportunities for purposeful time series generation and analysis.

7 Conclusion

The present paper presents a novel approach to control non-stationarity and periodicities with calendar and statistical information when generating time series. For this purpose, we first define the requirements for generation methods to generate time series with non-stationarity and periodicities, which we show is not fulfilled by existing generation methods. Secondly, we formally describe the novel approach for controlling non-stationarity and periodicities in generated time series. Thirdly, we introduce an exemplary implementation of this approach using a conditional Invertible

Table 10 Ablation study comparing different cINNs with respect to the average, minimum, and maximum discriminative score. Lower the better

Method	Energy	Information	Mobility
cINN	0.05 (0.03 - 0.09)	0.01 (0.00 - 0.04)	0.04 (0.00 - 0.05)
cINN Stats + Cal	0.03 (0.02 - 0.06)	0.02 (0.00 - 0.06)	0.03 (0.01 - 0.05)
cINN Stats	0.05 (0.02 - 0.07)	0.03 (0.00 - 0.06)	0.04 (0.01 - 0.06)
cINN Cal	0.05 (0.02 - 0.07)	0.04 (0.00 - 0.08)	0.10 (0.07 - 0.12)
INN	0.04 (0.01 - 0.06)	0.02 (0.00 - 0.05)	0.14 (0.11 - 0.15)

Neural Network (cINN) that preprocesses calendar and statistical information as conditional input with a conditioning network.

To evaluate the proposed cINN, we examine its capabilities to generate time series with controlled non-stationarity and periodicities in experiments with real-world data sets. We also compare the general quality of its generated time series to state-of-the-art benchmark generation methods and perform an ablation study to analyse the effects of the conditional information. The presented experiments show that the cINN can generate time series with controlled non-stationarity and periodicities while retaining the characteristics of the original data set. Furthermore, the cINN outperforms or is on par with the selected benchmark generation methods regarding the predictive and discriminative scores. The cINN also requires the lowest training time of the considered state-of-the-art generation methods.

Future work could relax the assumption of a bijective mapping by applying the proposed approach to other generative models to control non-stationarity and periodicities during time series generation. This way, relaxing this assumption could enable a more general framework to control non-stationarity and periodicities during time series generation. Moreover, future work could extend the proposed approach to multivariate time series and time series without calendar information. Similarly, future work could focus on identifying additional controllable properties of time series and incorporating them into the proposed approach.

Appendix A: Examples of realistic and useful time series

We introduce two possible use cases to clarify realistic and useful time series.

Mobility Use Case: Assuming that we have a dataset similar to the Mobility data set in our paper, a bike-sharing startup has just opened up a branch in a new city and has a few months of available data. If the managers want to develop a business plan for the next years, it will not be easy with their data. In this exemplary case, the data only comprises one season and the initial period where demand may differ from when the company is fully established. In order to accurately model the demand for the coming years, synthetic data that includes realistic seasonal trends and growth is required. Therefore, the managers would need to control the stationarity and trends in the synthetic data to perform any reasonable analysis.

Energy Use Case: Often, energy systems behave normally, i.e. we have a steady demand with clear daily, weakly, and yearly patterns. However, normal behaviour is not the scenario that leads to grid instabilities and other related problems. Instead, these problems are often caused by unexpected peaks in demand or outages in renewable energy generation. Therefore, many algorithms are being developed to predict these peaks and outages to provide early warning systems to grid operators. Unfortunately, these scenarios do not occur regularly, and, therefore, there is a lack of real-world data that can be used for training purposes. Generating synthetic data manipulated to represent such seldom situations (for example, anomalies and concept drift) is vital to assist other researchers in developing algorithms and forecasting methods to cope with such situations.

Both use cases have in common that they require realistic time series, i.e. time series that have characteristics similar to real-world data. Furthermore, both use cases benefit from useful time series that can be controlled according to the use case-specific problem that needs to be solved (e.g. business plan development or dealing with seldom events in energy forecasting).

Appendix B: Shortcomings of generation methods

This section further points out the shortcomings of generation methods concerning non-stationarity and periodicities in generated time series. In addition to Section 3.2 of the main paper, where we already show that all generated time series segments have the same expected value and autocovariance structure, this section focuses on the variance of the generated time series segments.

Variance As stated in the main paper, the variance of a random variable X is defined by

$$\sigma_X^2 = \mathbb{E}[(X-\mu)^2] = \mathbb{E}[X^2] - (\mathbb{E}[X])^2. \tag{17}$$

Given this definition of the variance, we can again apply the so-called law of the unconscious statistician (LOTUS) [21] to obtain an expression for the variance of the transformed random variable $g(Z,\theta_2)$, i.e.

$$\begin{aligned}\sigma_{g(Z;\theta_2)}^2 &= \mathbb{E}[g(Z;\theta_2)^2] - (\mathbb{E}[g(Z;\theta_2)])^2 \\ &= \int g(\mathbf{z};\theta_2)^2 P_Z \, d\mathbf{z} - (\mathbb{E}[g(Z;\theta_2)])^2,\end{aligned} \tag{18}$$

where θ_2 are the fixed trainable parameters. Since the transformed random variable $g(Z,\theta_2)$ only depends on the random variable Z and the fixed trainable parameters θ_2, all

Table 11 Characteristics of the time series contained in the three selected data sets energy, information, and mobility with regard to their temporal resolution, length, variance, and physical constraint

Data set	Temporal resolution	Length	Variance	Physical constraint
Energy	Hourly	17545	251.26 kW	Non-negative numbers
Information	Daily	550	920.94 # daily views	Non-negative numbers
Mobility	Hourly	16638	33132.16 # rented bikes	Non-negative numbers

generated time series segments $\hat{\mathbf{x}}^i,\ i \in [1, N]$ thus have the same variance, i.e.

$$\begin{aligned}\sigma^2_{\hat{\mathbf{x}}^i} &= \mathrm{var}[\hat{\mathbf{x}}^i] = \mathrm{var}[g(\mathbf{z}^i; \theta_2)] \\ &= \mathbb{E}[(g(\mathbf{z}^i; \theta_2) - \mu)^2],\ \forall i \in [1, N],\end{aligned} \tag{19}$$

where $\mathbf{z}^i,\ i \in [1, N]$ are realisations of the random variable Z used in the generative process.

Appendix C: Data sets and preprocessing

This section first lists and characterises the three selected data sets from the application fields energy, information, and mobility. Secondly, we detail each data set's selection and preprocessing of time series. Lastly, we give information on the rights to use the data and data privacy.

C.1 Data sets

We select the following three publicly available data sets for our experiments:

- **Energy** UCI Electricity Load Dataset [7]: https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014
- **Information** Google's Web Traffic Time Series Forecasting Dataset: https://www.kaggle.com/c/web-traffic-time-series-forecasting/data
- **Mobility** UCI Bikesharing Dataset [7, 9]: https://archive.ics.uci.edu/ml/datasets/bike+sharing+dataset

For all three data sets, Table 11 provides an overview of the contained time series regarding their temporal resolution, length, variance, periodicity, and physical constraint.

C.2 Selection and preprocessing of time series

We use time series without missing values and with a daily or hourly temporal resolution for our experiments. Therefore, we select a time series from each data set and prepare the selected time series as follows:

- **Energy** We resample the data set to a sample rate of one hour and select the time series *MT_158* from the data set.
- **Information** From the data set, we select the time series *Hitchhiker's Guide to the Galaxy (video game)*.
- **Mobility** We create the time index by merging the columns *dteday* and *hr*. Afterwards, we linearly interpolate the missing values before selecting the time series *cnt* from the data set.

Furthermore, we scale all selected time series using a Standard-Scaler to make the training more stable. Based on the selected and scaled time series, we create samples containing values of one day for the energy and the mobility data set and one week for the information data set as inputs for the cINN and the benchmarks. The resulting samples thus contain 24 values for the energy and the mobility data sets and seven values for the information data set. For all data sets, we took the first year for training.

C.3 Data use rights and data privacy

The associated license allows the reuse of the selected energy and mobility data sets, and the associated donation policy implies consent. Regarding the information data set, the competition guidelines of Kaggle[14] as data host limit the use of the data set to the competition itself. Nevertheless, the data set is commonly used in scientific publications. However, concerning the information data set, we cannot give further information on the consent because Google as a data provider, does not provide any information on this aspect.

Regarding privacy, none of the selected data sets contains personally identifiable information because the energy data set is pseudonymised, and the information and mobility data sets are aggregated.

Acknowledgements Benedikt Heidrich and Marian Turowski were funded by the Helmholtz Association's Initiative and Networking Fund through Helmholtz AI. Kai Schmieder was funded by the Helmholtz Association's Initiative and Networking Fund through Helmholtz Metadata Collaboration. Kaleb Phipps was funded by the German Research Foundation (DFG) Research Training Group 2153 "Energy Status Data: Informatics Methods for its Collection, Analysis and Exploitation". Veit Hagenmeyer was funded by the Helmholtz Association under the Program "Energy System Design". We thank Nicole Ludwig for her valuable input during the preparation of the manuscript.

[14] https://www.kaggle.com/c/web-traffic-time-series-forecasting/rules

Author Contributions **Benedikt Heidrich:** Conceptualisation, Methodology, Software, Investigation, Writing - Original Draft, Visualisation **Marian Turowski:** Conceptualisation, Methodology, Investigation, Writing - Original Draft **Kaleb Phipps:** Conceptualisation, Methodology, Investigation, Writing - Original Draft **Kai Schmieder:** Conceptualisation, Methodology, Investigation, Writing - Original Draft **Wolfgang Süß:** Writing - Review & Editing, Supervision **Ralf Mikut:** Writing - Review & Editing, Supervision **Veit Hagenmeyer:** Funding acquisition, Writing - Review & Editing, Supervision

Funding Open Access funding enabled and organized by Projekt DEAL.

Declarations

Competing interests The authors have no competing interests to declare that are relevant to the content of this article.

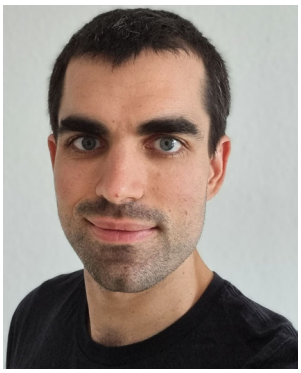
Data Availability The datasets analysed during the current study are available in the UCI repository, https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014. The datasets generated during the current study can be replicated with the code made available in our GitHub repository, https://github.com/KITIAI/ControllableTimeSeriesGeneration.

References

1. Abadi M, Agarwal A, Barham P et al (2015) TensorFlow: large-scale machine learning on heterogeneous systems. https://www.tensorflow.org/
2. Ardizzone L, Lüth C, Kruse J et al (2019) Guided image generation with conditional invertible neural networks. arXiv:1907.02392
3. Chollet F et al (2015) Keras. https://keras.io
4. Dinh L, Sohl-Dickstein J, Bengio S (2017) Density estimation using Real NVP. In: 5th International conference on learning representations, ICLR 2017 - conference track proceedings. arXiv:1605.08803
5. Donahue C, McAuley J, Puckette M (2019) Adversarial audio synthesis. In: 7th International conference on learning representations, ICLR 2019. arXiv:1802.04208
6. Dong HW, Hsiao WY, Yang LC et al (2018) MuseGAN: multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. In: The thirty-second AAAI conference on artificial intelligence (AAAI-18), pp 34–41. https://ojs.aaai.org/index.php/AAAI/article/view/11312
7. Dua D, Graff C (2019) UCI machine learning repository. http://archive.ics.uci.edu/ml
8. Esteban C, Hyland SL, Rätsch G (2017) Real-valued (medical) time series generation with recurrent conditional GANs. arXiv:1706.0.2633
9. Fanaee TH, Gama J (2014) Event labeling combining ensemble detectors and background knowledge. Progress Artif Intell 2:113–127. https://doi.org/10.1007/s13748-013-0040-3
10. Ge L, Liao W, Wang S et al (2020) Modeling daily load profiles of distribution network for scenario generation using flow-based generative network. IEEE Access 8:77,587–77,597. https://doi.org/10.1109/ACCESS.2020.2989350
11. Goodfellow I, Pouget-Abadie J, Mirza M et al (2014) Generative adversarial nets. In: Ghahramani Z, Welling M, Cortes C et al (eds) Advances in neural information processing systems, pp 4089–4099. https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf
12. Heidrich B, Bartschat A, Turowski M et al (2021) pywatts: Python workflow automation tool for time series. arXiv:2106.10157
13. Hyndman R, Athanasopoulos G (2018) Forecasting: principles and practice, 2nd edn. OTexts: Melbourne, Australia. OTexts.com/fpp2. accessed on 24.05.2021
14. Kingma DP, Dhariwal P (2018) Glow: generative flow with invertible 1x1 convolutions. In: Bengio S, Wallach H, Larochelle H et al (eds) Advances in neural information processing systems, pp 10,215–10,224. https://proceedings.neurips.cc/paper/2018/file/d139db6a236200b21cc7f752979132d0-Paper.pdf
15. Kingma DP, Welling M (2014) Auto-encoding variational Bayes. arXiv:1312.6114
16. Lan J, Guo Q, Sun H (2018) Demand side data generating based on conditional generative adversarial networks. Energy Procedia 152:1188–1193. https://doi.org/10.1016/j.egypro.2018.09.157
17. van den Oord A, Dieleman S, Zen H et al (2016) WaveNet: a generative model for raw audio. arXiv:1609.03499
18. Paszke A, Gross S, Massa F et al (2019) Pytorch: an imperative style, high-performance deep learning library. In: Wallach H, Larochelle H, Beygelzimer A et al (eds) Advances in neural information processing systems, pp 8024–8035. https://proceedings.neurips.cc/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf
19. Prenger R, Valle R, Catanzaro B (2019) WaveGlow: a flow-based generative network for speech synthesis. In: 2019 IEEE International conference on acoustics, speech and signal processing (ICASSP), pp 3617–3621. https://doi.org/10.1109/ICASSP.2019.8683143
20. Ramponi G, Protopapas P, Brambilla M et al (2018) T-CGAN: conditional generative adversarial network for data augmentation in noisy time series with irregular sampling. arXiv:1811.08295
21. Ross SM (2010) Introduction to probability models, 10th edn. Academic Press. https://doi.org/10.1016/C2009-0-30640-6
22. Xu T, Wenliang LK, Munn M et al (2020) COT-GAN: generating sequential data via causal optimal transport. In: Larochelle H, Ranzato M, Hadsell R et al (eds) Advances in neural information processing systems, pp 8798–8809. https://papers.nips.cc/paper/2020/file/641d77dd5271fca28764612a028d9c8e-Paper.pdf
23. Yoon J, Jarrett D, van der Schaar M (2019) Time-series generative adversarial networks. In: Wallach H, Larochelle H, Beygelzimer A et al (eds) Advances in neural information processing systems, pp 5508–5518. https://proceedings.neurips.cc/paper/2019/file/c9efe5f26cd17ba6216bbe2a7d26d490-Paper.pdf

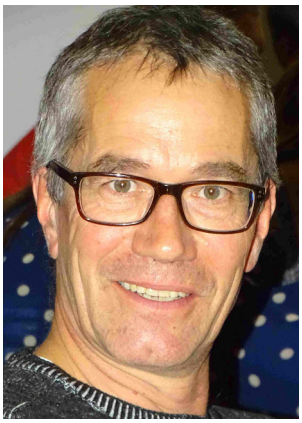
Benedikt Heidrich completed a Master of Science degree in informatics in 2019 with the Karlsruhe Institute of Technology, German. Currently, he is working towards a PhD in Informatics at the Karlsruhe Institute of Technology. His research focuses on using deep generative models in the energy systems and coping with concept drift in energy time series forecasting.

Marian Turowski received the M.Sc. degree in Industrial Engineering and Management from the Karlsruhe Institute of Technology, Karlsruhe, Germany in 2017. He is currently pursuing his Ph.D. at the Institute for Automation and Applied Informatics with the Karlsruhe Institute of Technology. His research interests comprise anomaly detection and handling as well as forecasting in energy time series.



Kaleb Phipps completed a double Master of Science degree in industrial engineering and management in 2019 with the Karlsruhe Institute of Technology, Germany and Linköping University, Sweden. Currently, he is working towards a PhD in computer science at the Karlsruhe Institute of Technology. His research focuses on quantifying the uncertainty in energy systems with probabilistic forecasts for renewable generation and demand, concentrating on non-parametric machine learning methods.

Kai Schmieder Before joining the Fraunhofer Institute for Industrial Engineering IAO as a research assistant, he was with the Institute for Automation and Applied Informatics within the Karlsruhe Institute of Technology and supported the Helmholtz Metadata Collaboration in the field of research data management for energy research.



Wolfgang Süß received a diploma in mathematics in 1988 from the University of Karlsruhe and a Ph.D. degree in computer science in 1993 from the University of Koblenz-Landau. At Karlsruhe Institute of Technology (KIT) he is leader of the research group IT-methods and -components for energy data management. He currently coordinates the Hub Energy of the Helmholtz Metadata Collaboration (HMC).

Ralf Mikut received the Diploma degree in automatic control from the University of Technology, Dresden, Germany, in 1994, and the Ph.D. and Habilitation degrees in mechanical engineering from the University of Karlsruhe, Karlsruhe, Germany, in 1999 and 2007, respectively. Since 2011, he has been an Adjunct Professor at the Faculty of Mechanical Engineering and the Head of the Research Group "Automated Image and Data Analysis," Institute for Automation and Applied Informatics, Karlsruhe Institute of Technology, Eggenstein-Leopoldshafen, Germany. His current research interests include machine learning, image processing, life science applications, and smart grids.

Veit Hagenmeyer received the Ph.D. degree from Universite Paris XI, Paris, France, in 2002. He is currently a Professor of energy informatics with the Faculty of Computer Science, and the Director of the Institute for Automation and Applied Informatics, Karlsruhe Institute of Technology, Karlsruhe, Germany. His research interests include modeling, optimization and control of sector-integrated energy systems, machine-learning based forecasting of uncertain demand and production in energy systems mainly driven by renewables, and integrated.

Affiliations

Benedikt Heidrich[1] · Marian Turowski[1] · Kaleb Phipps[1] · Kai Schmieder[1,2] · Wolfgang Süß[1] · Ralf Mikut[1] · Veit Hagenmeyer[1]

Marian Turowski
marian.turowski@kit.edu

Kaleb Phipps
kaleb.phipps@kit.edu

Kai Schmieder
kai.schmieder@iao.fraunhofer.de

Wolfgang Süß
wolfgang.suess@kit.edu

Ralf Mikut
ralf.mikut@kit.edu

Veit Hagenmeyer
veit.hagenmeyer@kit.edu

[1] Karlsruhe Institute of Technology, Institute for Automation and Applied Informatics, Hermann-von-Helmholtz-Platz 1, 76344 Eggenstein-Leopoldshafen, Baden-Württemberg, Germany

[2] Research and Innovation Center for Cognitive Service Systems (KODIS), Fraunhofer Institute for Industrial Engineering IAO, Bildungscampus 9, 74076 Heilbronn, Baden-Württemberg, Germany