UNIVERSITÉ DE FRIBOURG
UNIVERSITÄT FREIBURG

eXascale Infolab

UNIVERSITY OF FRIBOURG

BACHELOR THESIS

Comparison of Synthetic Time Series Data Generation Techniques

Author:
Jonas Fontana

Supervisor:
Abdelouahab Khelifati,
Dr. Mourad Khayati,
Prof. Dr. Philippe
Cudré-Mauroux

September 07, 2021

eXascale Infolab
Department of Informatics

Boulevard de Pérolles 90 • 1700 Fribourg • Switzerland
phone +41 (26) 300 84 65 • diuf-secr@unifr.ch • www3.unifr.ch/inf

Abstract

Jonas Fontana

Comparison of Synthetic Time Series Data Generation Techniques

With the explosion of time series data, numerous studies have focused on developing tools to analyze them. However, in order to train, improve and evaluate these methods, a lot of data is necessary. Large datasets are often very homogeneous, and in addition for many domain-specific application, very few if any data is available. To this end, various synthetic time series augmentation techniques have been proposed. Those techniques rely on different principles and aim to achieve different goals when generating new data, which makes it difficult to choose the best technique for a given use-case situation.

In this thesis, we study six augmentation techniques: Anomalies Injection, DBA, Autoregressive models, GAN, InfoGAN and TimeGAN. We empirically evaluate these techniques using datasets that exhibit different characteristics (for example, some of them have very evident cyclical patterns, some others contain anomalies). In addition, we introduce a new a tool to automate the evaluation, parameterize the techniques, and visualize the generated time series. From this we will conclude that in most cases machine learning methods are in many cases the best option, however simple methods (such as DBA and Anomalies Injection) and methods based on autoregressive models are also a valid option in some situations.

Keywords: Time Series, Data Augmentation, UCR, Comparation

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

In the last decade, there has been an explosion of time series data. From financial field to rivers and sea levels, but also medical data and speech analysis: time series are everywhere. Every time a sequence of values is recorded in time order they create a time series. Consequently, a big number of time series analysis methods are appearing for tasks such as classification, forecasting, anomaly detection, and more. However, in order to train, improve and evaluate these methods, a lot of data is necessary. Paradoxically, large datasets are often very homogeneous, which makes it challenging to train a model. In addition, for many domain-specific applications very few if any data is available. For this reason, new time series augmentation methods are created as well.

In this thesis, we organize the different methods and propose a way to evaluate them. We also present a tool to facilitate this task and show an example of how to use it with different datasets and with different metrics.

1.2 Problem Definition

In the last years, many time series generation methods have been proposed. These methods rely on different techniques, and the results differ widely ([Wen et al., 2020]). Therefore, one question that arises spontaneously is: which method is the best? Answering this question is not as obvious as it might seem, as it is first necessary to determine what characterizes a good generation. How should the generated data be? Should it be similar to the original data? Or should it rather be different, but sharing some characteristics? Should one instead focus on the preservation of some features, and if yes, which?

What makes it challenging to answer those questions is that it highly depends on the task we have. When generating data to improve classification, we want to have synthetic data similar to the original one. Instead, if our goal is to improve an anomaly detection model, we need to have more variegated data. The goal of the generation depends therefore on the use we want to make of the generated data. Furthermore, a method satisfying our requirements for one dataset may not do so for another. The choice of the best technique for a given situation also depends on the properties of the input dataset (presence of a seasonal pattern, anomalies, etc.).

In this work, we analyze different techniques to generate synthetic data, as well as to analyze this generation in multiple ways (preservation of features, metrics score, visual comparison). To do this, we will first discuss the main classes used at the moment (1.3 Related Work). Then, in (2.1 Time Series Properties and Features)

and (2.2 Time Series Evaluation Metrics) we will present a way to evaluate time series, and in (3 Generation Algorithms) we will show some implementations for each of the discussed classes. The results of the analysis performed are presented in (4 Experiments). Finally, in (5 Generation Tool) we will also present a tool we created to facilitate the use and the comparison of the techniques considered.

1.3 Existing Work

In the past years, multiple algorithms for time series generation and augmentation have been implemented. The difference between generation and augmentation methods is that the latter starts from an initial dataset and create new data to enlarge it (in this work we will focus on these). Although the ultimate goal is the same for all of them, namely the creation of synthetic data to improve analysis models, they are very different from each other. We can distinguish 3 main categories: Simple/Basic approaches, Model-based approaches, and Machine Learning approaches.

The easiest approach consists in taking an input dataset and generating new data by modifying/perturbing it. This can go from a simple operation like injecting noise (for example Gaussian noise, spikes, or trend), to others like window cropping ([Le Guennec, Malinowski, and Tavenard, 2016]) and Dynamic Time Warping ([Forestier et al., 2017]). In general, we distinguish two categories: Time Domain transformations and Frequency Domain transformations. Time Domain transformations are the most common, and as the name says they operate on the time aspect of time series. Most of them operate directly on the original time series, however not all of them. An example of this second case is given in [Gao et al., 2020], where operations on labels are used to improve anomaly detection. On the other hand, Frequency Domain transformations work on the frequency aspect of time series. An example is given again by [Gao et al., 2020].

A more elaborated approach to generate time series consists in building a model from existing data, then generate new data starting from this model ([Chen et al., 2010], [Talbot et al., 2019]). For example, the work of [Kang, Hyndman, and Li, 2020] focuses on creating a mixture of autoregressive models to generate time series covering the entire feature space. More basic examples consist of simple Autoregressive Models, where each point is considered to be determined by the previous points. Once the weight of each previous point is determined, this model can be used to estimate the future values. An article about this is presented from [Brownlee, 2020].

A third category is constituted by (machine) learning methods. These methods use ML models to learn existing data and then generate similar ones. An example is provided by [Laptev, 2018], where Variational Auto Encoders (VAEs) and Generative Adversarial Network (GANs) are used to generate new data to improve anomaly detection. VAEs compress the data to a latent space and then reconstruct it. Randomly sampling from this latent space will then allow to reconstruct synthetic data. GANs, on the other hand, are composed of two models competing against each other, one creating synthetic data and the other trying to distinguish original data from synthetic ones. Other works to better adapt GANs to time series include [Chen et al., 2016] and [Yoon, Jarrett, and Schaar, 2019].

Chapter 2

Background

In this section, we will briefly discuss the major techniques to evaluate time series. We will introduce a list of time series features and metrics based on these features, which we will use to evaluate the generated time series. We will explain how they work, and discuss why we consider them important.

2.1 Time Series Properties and Features

Time series can be described and classified based on their characteristics. For example, a time series representing the temperature recorded each day might have a "cyclical form" (higher during summer and lower in winter), while at the same time an increasing tendency (due to climate change). Then it might have a lot of "spikes" (days with extraordinary high/low values) or, on the contrary, the values might all be within a small range. All this and many other characteristics can be mathematically described by a list of features, which are in part summarized in this chapter.

2.1.1 Spectral Entropy

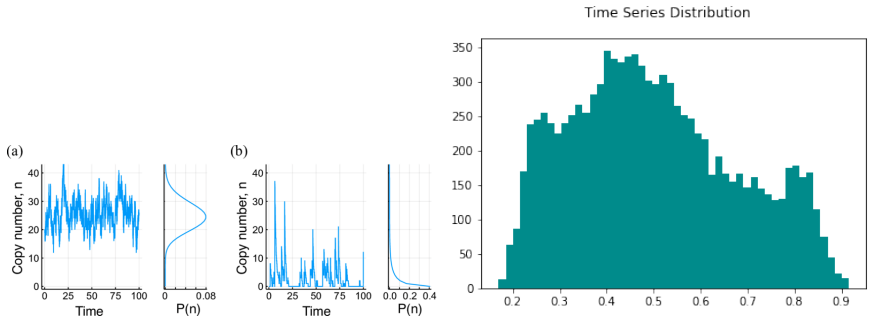
In Information Theory, Shannon's Entropy is the amount of information contained in a variable ([Vajapeyam, 2014]). An intuitive way to see this is: if I had to store the information given by the variable, how much storage would I need? For example, if we need to store the result of a non-biased coin toss, one bit would be enough. This is because there are only 2 possible results (head and coin), so for example we could set the bit to 0 in the first case and to 1 in the second. Similarly, if we had 8 variables instead of 2, we would need 3 bits.
But if the variables would not appear with the same frequency? For example, suppose we have 3 variables a, b and c. If a comes 9 times out of 10, we could decide to just store a 0 for a. If the variable is a b, then we would store 10, and 11 if it is a c. In this way, 90% of the time we would only need 1 bit, and 10% of the time 2 bits. Therefore, on average we need 1.1 bits.
The formula of Shannon's Entropy, therefore, sums all the values with the logarithm of their frequency (as the log contains a fraction, the value will be negative. For this reason the result is multiplied by -1).

$$E = -\sum_{i=1}^{n} p_i \cdot log(p_i)$$

Spectral Entropy is a normalized version of Shannon's Entropy (between 0 and 1). We can use it to measure the disorder between a time series, and its forecastability. The smaller is the Spectral Entropy, the higher the frequency of some values, and therefore it is easier to "guess" the next value.

2.1.2 Distribution

Another important characteristic of a time series is the way in which its values are distributed. If the values are distributed in a normal way (Gaussian distribution), the mean and variance of this distribution are enough to fully describe it. However, most of the time it is not so. To show the distribution of a time series we can therefore partition the possible values in sets and count how many points are in each group. It is then possible to plot the results using a histogram, as shown in Figure 2.1a and Figure 2.1b.



(A) Example of a time series and the corresponding distribution functions[1]

(B) Example of time series distribution

FIGURE 2.1: Time Series distribution

2.1.3 Autocorrelation

Given 2 variables (in this case 2 time series), the correlation between them is the measure of similarity they have. It measure how they vary together. The correlation function is defined as

$$Corr(x, y) = \frac{Cov(x, y)}{\sigma(x)\sigma(y)}$$

where $Cov(x, y)$ is the covariance of x and y, and $\sigma(x)$ is the standard deviation of x. It can have a value between -1 (perfect negative correlation) and +1 (perfect positive correlation).

As the name says, the autocorrelation of a time series is its correlation with itself (shifted by n steps). The autocorrelation function measures its correlation with a copy of itself shifted by 1 step, by 2 steps, by 3 steps, etc. Figure 2.2 shows an example of 2 autocorrelation functions

2.2 Time Series Evaluation Metrics

When describing time series on their own, features are very practical. However, when comparing multiple time series, we have an additional tool: metrics. Metrics uses the features to describe the relationship between two (or more) time series. In our case, this is especially useful to evaluate generated data against the original one.

2.2.1 Mutual Information

A first important metric used to measure similarity between two sets of data is the Mutual Information ([Dionisio, Menezes, and Mendes, 2004]). Given two variables

[1] [Ham, Brackston, and Stumpf, 2019]

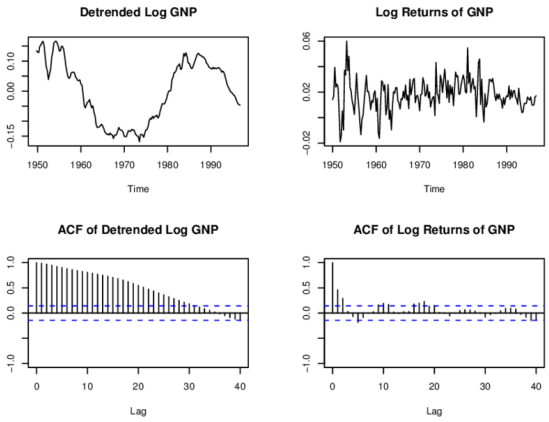


FIGURE 2.2: Example of autocorrelation function ([Paige and Trindade, 2010])

v_1 and v_2, the Mutual Information measures the amount of information we can obtain about v_2 by looking at v_1 (and the other way around, as it is symmetric).

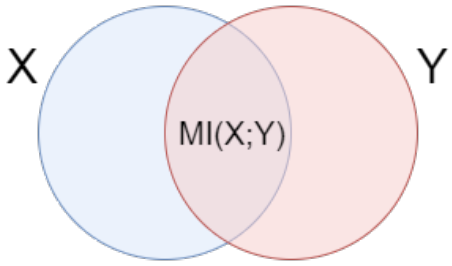


FIGURE 2.3: Mutual information ([*Mutual Information diagram*])

In this case, given two time series s1 and s2 (the original one and the synthetic one), Mutual Information can be used to measure the similarity between them, thus evaluating the generation algorithm. The Mutual Information (MI) between time series X and Y can be computed as

$$MI(X,Y) = \sum_{i=1}^{|X|}\sum_{j=1}^{|Y|} p(i,j)log\left(\frac{p(i,j)}{p(i)p(j)}\right)$$

where $p(i,j) = \left|X_i \cap Y_j\right| / N$ is the probability that an object picked at random falls into both classes X_i and Y_j ([*Scikit-learn: Clustering*]), as illustrated by Fig. 2.3.

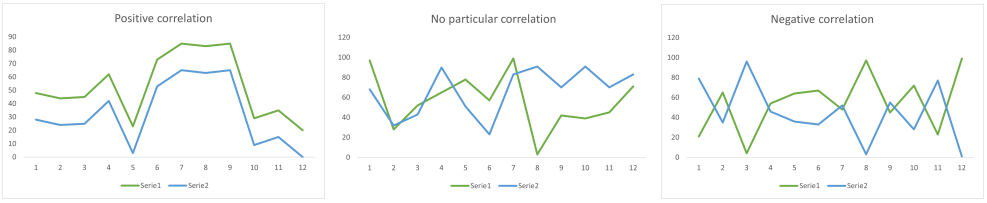
2.2.2 Correlation

As introduced in 2.1.3, correlation is the measure of similarity between two variables (in this case, between two time series). It is computed with:

$$Corr(x,y) = \frac{Cov(x,y)}{\sigma(x)\sigma(y)}$$

where $Cov(x,y)$ is the covariance of x and y, and $\sigma(x)$ is the standard deviation of x. Its value ranges from +1 to -1, where +1 means that x and y vary together, -1 that

they vary perfectly oppositely, and 0 that they vary independently from each other. Fig. 2.4 shows three examples of correlation.



(A) Perfect positive correlation (+1) (B) Very small correlation (~0.01) (C) Perfect negative correlation (-1)

FIGURE 2.4: Examples of correlation

2.2.3 Root Mean Squared Error (RMSE)

Another important metric to evaluate a generation model is the Root Mean Squared Error (or Deviation), shown in Fig. 2.5. Given a model and a time series, the RMSE sums the squares of the differences between each point of the time series and the model. Squaring them ensures that the values are positive (otherwise positive and negative errors would compensate each other), and gives a higher weight to bigger errors. Taking the root at the end ensures the result to have the same scale of the data.

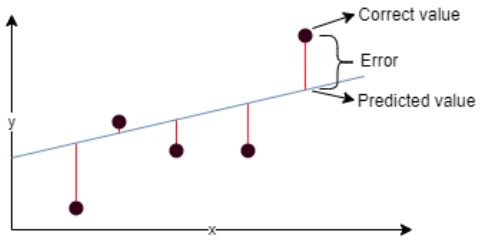


FIGURE 2.5: Root Mean Squared Error

Chapter 3

Time Series Generation Methods

In this section, for each approach mentioned in 1.3 we will present some concrete techniques. These are the techniques that will then be used to generate synthetic data in 4.

3.1 Anomalies Injection

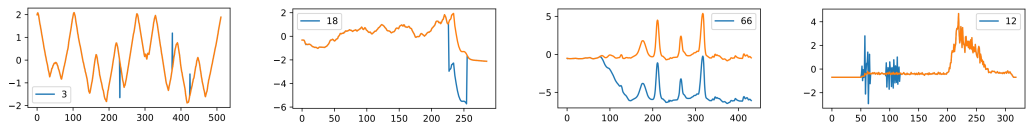
A first basic approach is the anomalies injection. This belongs to the data perturbation class. Given an initial time series, it is modified by randomly injecting some anomalies in order to obtain new data. For example, [*Agots*] provides four type of anomalies: "extreme" anomalies, "shift" anomalies, "trend" anomalies, and "variance" anomalies.

Extreme anomaly: a point is modified to have a much bigger/smaller value than the original one, thus resulting in a spike when the time series is plotted. An example is given in Figure 3.1 (A).

Shift anomaly: all the records in a given interval are shifted by a given value, which is equal in every point. The result is that a part of the time series is shifted up or down. An example is given in Figure 3.1 (B).

Trend anomaly: a trend is inserted at a given point in the time series, meaning that an increasing (or decreasing) sequence of values is added to a portion of the time series. For example, a time series [1,1,1,1,1,1,1,1,1,1] might become [1,1,1,2,3,4,4,4,4,4]. Notice that after the trend part ([2,3,4]), all the values are modified in order to continue "directly" from the last point (in this example, they are all increased by 3). An example is given in Figure 3.1 (C).

Variance anomaly: the variance of a random interval is augmented. Visually, this results in something similar to a "vibration". An example is given in Figure 3.1 (D).



(A) Example of 3 **extreme** anomalies (B) Example of a **shift** anomaly (C) Example of a **trend** anomaly (D) Example of two **variance** anomalies

FIGURE 3.1: Anomalies examples

Algorithm 1 Pseudocode of AnomaliesInjection

Input: Original dataset
Optional inputs: Nb. anomalies, anomalies probabilities, anomalies max length
cp ← copy of original dataset
for Nb. anomalies required **do**
    Randomly choose:
        TS target ▷ Time Series in which to add the anomaly
        Anomaly type
        Starting point
        Length
    Add anomaly to selected TS in cp
Output: cp (Copy of original dataset with anomalies)

3.2 Dynamic Time Warping Barycenter Averaging (DBA)

Another basic approach to generate new data from an initial dataset is DBA. With DBA, new time series are creating by averaging some of the existing ones using (an advanced version of) Dynamic Type Warping, a measure of similarity between time series ([Volny, Novak, and Zezula, 2012]). Figure 3.2 gives a visual explanation of DTW.

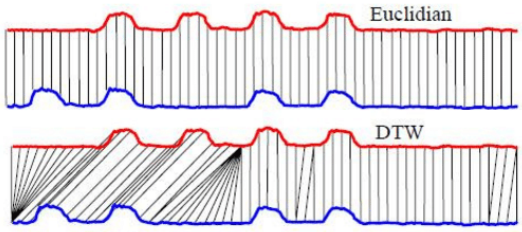


FIGURE 3.2: Difference between Euclidean and DTW distance ([Volny, Novak, and Zezula, 2012])

With this algorithm, a new time series is created as the weighted average of the already existing time series. The only parametrization is in how the weights are distributed. A first approach is to distribute them randomly across all the available time series. Theoretically, with this approach it would be possible to generate an infinite number of new time series. However, more advanced approaches exist. According to [Forestier et al., 2017], the best results are obtained when a "main" time series receives most of the weight, and the remaining is distributed across the other series according to an exponential function based on their distance. The farther they are from this main time series, the smaller the weight they receive. Formally, the weight for time series i is given by

$$w_i = e^{ln(0.5)\cdot\frac{DTW(T_i,T^*)}{d^*_{NN}}} \tag{3.1}$$

where T^* is the "main" time series selected and d^*_{NN} is the distance between T^* and its nearest neighbor. Once all the weights have been distributed, the weighted average of N time series under DTW is the time series that minimizes

$$arg\,min\,\overline{T} := \sum_{i=1}^{N} w_i DTW^2(\overline{T}, T_i) \tag{3.2}$$

Fig. 3.3 shows a set of time series (left) and a new time series generated with DBA starting from this set (right).

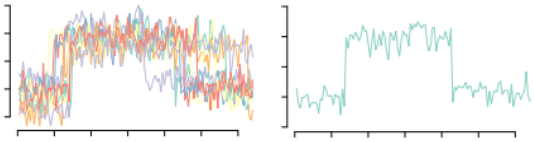


FIGURE 3.3: Example of data generated with DBA ([Forestier et al., 2017])

Algorithm 2 Pseudocode of DBA

Input: Original dataset
for Nb. new time series required **do**
  Randomly select main time series T*
  Compute DTW distance to nearest neighbour
  Distribute weights w_i ▷ Using 3.1
  Create new time series $\overline{T}$ ▷ Using 3.2
Output: set of new time series

3.3 Data generation with autoregressive models

The basic model-base approach is the Autoregressive models. This model assumes that a value y at time t depends on previous values at times t-1, t-2, ..., t-p, plus a noise ε_t. Formally,

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-1} + ... + \phi_p y_{t-p} + \varepsilon_t$$

where c is a constant. Based on the existing data, the model sets a coefficient ϕ for each of the p previous values. Once the model is constructed, it can then be used to generate new time series, one point by one.

AR models can be extended to more sophisticated models, for example ARMA (Autoregressive Moving Average), ARIMA (Autoregressive Integrated Moving Average), or SARIMA (Seasonal Autoregressive Integrated Moving Average). Despite the increasing number of components, they are all based on the idea that past values determine the current one.

Algorithm 3 Pseudocode of AR

Input: Original dataset, window size n
for time series t in input dataset **do**
  Find n coefficients ϕ_i for time series t using regression
  for length of time series to generate **do**
    Generate new point using coefficients ϕ
  Keep only new points
Output: set of new time series

3.4 Generative Adversarial Networks (GANs)

Generative Adversarial Networks were introduced in 2014 by Ian Goodfellow in [Goodfellow et al., 2014] and have been described by Yann LeCun as "the most interesting idea in the last 10 years in Machine Learning"([*Generative Adversarial Network (GAN)*]). It is an architecture to train a Generative model, constituted from two submodels: a "generator", in charge of generating synthetic data starting from a random input vector, and a "discriminator", whose task is to distinguish real data (i.e. from the original source) from synthetic data. As the name suggests, the two models compete one against the other. The generator is a generative model which tries to "foolish" the discriminator, maximizing the probability for it to make a mistake in the classification. To do this, it receives a vector randomly drawn from a Gaussian distribution (the seed of the generation), and generates a new sample of the data. On the other side, the discriminator is a classification model, and it learns to distinguish real data from fake ones. It receives a sample (either from the original dataset or from the generator output) and does a binary prediction (real or fake). Formally, the generator (G) and the discriminator (D) play a minimax game with value function $V(D, G)$:

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

The training succeeds when the discriminator is wrong about 50% of the time. At this point, the generator should be able to create synthetic data that is indistinguishable from the real one. Fig. 3.4a shows a scheme of a GAN model.

Algorithm 4 Pseudocode of GAN

Input: Original dataset
for Nb. epochs **do**
  $r \leftarrow$ random sample from Gaussian distribution
  Generator:
    Receive r
    Generate fake data d_{gen}
  Discriminator:
    Receive d_{gen} **or** sample from original data
    Predict class (real or fake)

3.4.1 TimeGAN

GANs were not created specifically for time series. Indeed, they are mostly used with images, and therefore are not particularly good when it comes to properties specific to time series.

For this reason, [Yoon, Jarrett, and Schaar, 2019] developed an alternative called TimeGAN. The goal of TimeGAN is to capture not only the properties within each time point, but also across time ([Yoon, Jarrett, and Schaar, 2019]). In other words, it tries to consider *temporal dynamics*. To do this, it uses four components (instead of 2): an embedding function, a recovery function, a sequence generator, and a sequence discriminator. Training the autoencoding components (first two) jointly with the adversarial components (last two), the model should be able to simultaneously learns

[1] [Goodfellow et al., 2014]

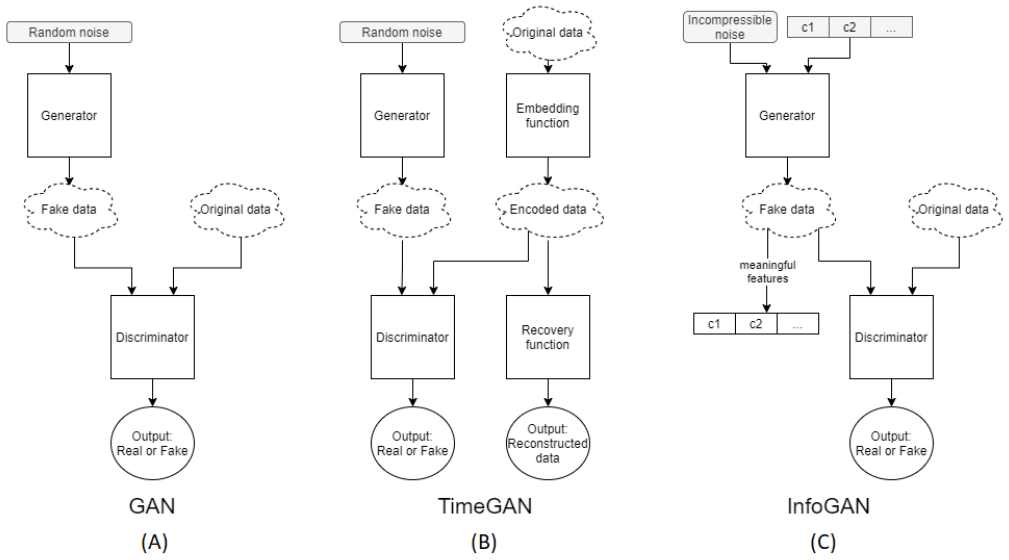


FIGURE 3.4: Sketch of GAN, TimeGAN and InfoGAN architectures

to encode features, generate representations, and iterate across time ([Yoon, Jarrett, and Schaar, 2019]). In this way, the adversarial components act on the latent space provided by the embedding function, learning the underlying temporal dynamics. Fig. 3.4b shows a scheme of a TimeGAN model.

3.4.2 InfoGAN

Another evolution of GAN is InfoGAN, presented by [Chen et al., 2016]. The idea behind InfoGAN is to encourage the model to learn *meaningful interpretation* ([Chen et al., 2016]) through mutual information maximization and unsupervised disentangled representation. According to the authors, the underlying problem is that GANs receive no restrictions on how to use the input vector, and might therefore learn a very entangled representation. However, "many domains naturally decompose into a set of semantically meaningful factors"([Goodfellow et al., 2014]). For example, the image of a face can be represented through the eyes color, eyes distance, hairs length, etc. To encourage this representation, InfoGAN decomposes the input vector into two parts: a first component z, which will be the source of the *incompressible noise*, and c, whose structure will enforce learning of the *meaningful features*. The Generator thus becomes $G(z, c)$. To ensure that it will not just ignore the c part, InfoGAN requires $G(z, c)$ to have a high mutual information (see 2.2.1) with c. Fig. 3.4c shows a scheme of an InfoGAN model.

Chapter 4

Experiments

In this chapter, we will evaluate the data generated with the techniques mentioned in 3. Firstly, we will summarize the specifications of the machine we use. After that, we will present the datasets used in the experiment, as well as the metrics considered. In a second part, we will then analyze the results obtained, and at the end of the chapter we will derive a recommendation on which class of techniques to use in which situation.

4.1 Setup

In this section we will present the setup for our experiment. In particular, we will indicate the machine used, the dataset considered, and the evaluation methods. The algorithms used have already been presented in 3 Time Series Generation Methods.

4.1.1 Machine

For the experimental part, we chose to run our code on a linux server with the following specifications:

Processor	Intel(R) Core(TM) i7-4770 CPU @ 3.40GHz 1 physical processor; 4 cores; 8 threads
Machine Type	Mini Tower
Operating System	Ubuntu 20.04.2 LTS
Kernel	Linux 5.4.0-67-generic (x86_64)

4.1.2 Datasets

In this section, we will shortly describe the datasets used to compare the different generation techniques. These have been chosen to have different characteristics, for example a greater or smaller number of records, a more o less strong seasonal pattern, presence or absence of anomalies/spikes, and uniformity or variation of the time series.

BeetleFly (from UCR archive)

The BeetleFly dataset is a 1-D mapping of Beetle's and Fly's images uploaded to the UCR archive by J. Hills and A. Bagnall. It represents the distance of the outline from the center of the images. The result is a dataset of time series with a cyclical pattern and a mean of 0. More about this dataset can be found at [*Dataset: BeetleFly*].

Coffee (from UCR archive)

The Coffee dataset comes from the paper [Briandet, Kemsley, and Wilson, 1996] about food spectrographs, and has been uploaded to the UCR archive by K. Kemsley and A. Bagnall. Most of the series in the dataset overlaps, with only some minor variation. No particular trend or cyclical pattern emerges. More about this dataset can be found at [*Dataset: Coffee*].

Ham (from UCR archive)

The Ham dataset is another dataset about food spectrographs (same as 4.1.2 Coffee), uploaded to the UCR archive by K. Kemsley and A. Bagnall. It represents Spanish and French dry-cured hams. The result is a dataset with time series containing up to 4/5 spikes each. More about this dataset can be found at [*Dataset: Ham*].

Lighting7 (from UCR archive)

The Lighting7 dataset contains records of the FORTE satellite about electromagnetic events associated with lightning. It was uploaded to the UCR dataset by D. Eads. Similar to the 4.1.2 Ham dataset, most of the values are in a small range, with occasionally some spikes (but rarely more than 1 per time series). More about this dataset can be found at [*Dataset: Lighting7*].

Alabama_weather

The Alabama_weather dataset contains longer and more variegated time series w.r.t. the considered UCR datasets. The time series in the datasets show no particular trend or seasonality. Some contain noise, others much less. To keep the generation time reasonable, only a subset has been considered (limiting the number of time series and their length).

Currency2

As the name says, the Currency2 dataset contains a list of currencies values. Most of the time series in this dataset have a similar value range, but there are some exceptions. They do not have any evident cyclical pattern or common trend, and they tend not to change too abruptly.

Alabama_weather_3072

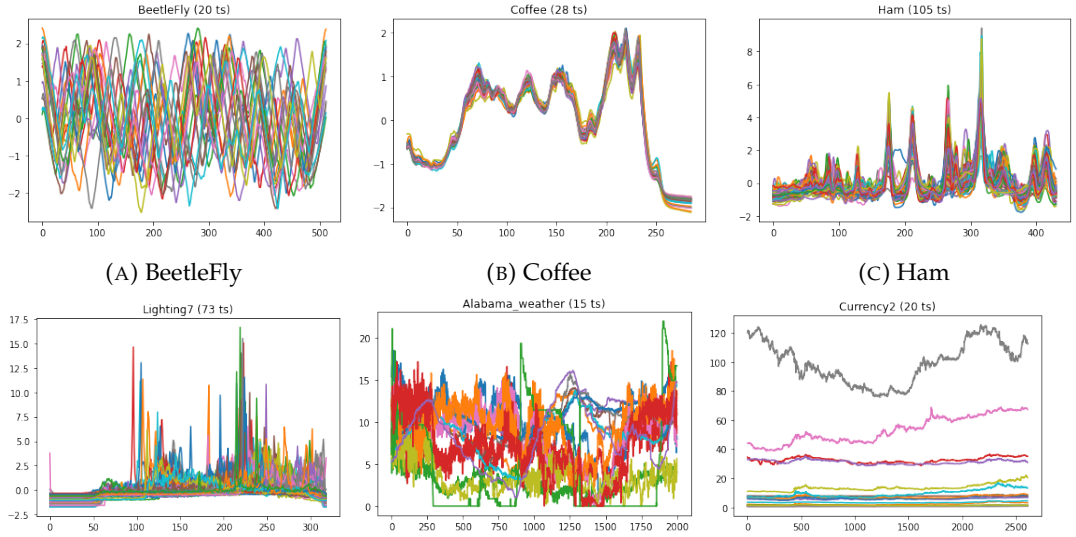
The major drawback of the BasicGAN implementation found is that it requires a dataset with a specific structure, meaning 3072 time series of length 3072. To obtain it, we manually created a dataset shifting a 3072-points window over a time series of length 6143, with a step of 1. As consequence, the resulting dataset is extremely uniform, but this allows to consider BasicGAN technique as well. The original time series is from the Alabama_weather dataset.

Summary

Table 4.1 summarize the datasets used. Fig. 4.1 shows them, while Fig. 4.2 shows the only the first 3 time series of each dataset.

	Nb. of time series	Length	Caracteristics
BeetleFly	20	512	Similar pattern for all time series Cyclic
Coffee	28	286	Very uniform
Ham	105	431	Spikes at the same point for each time series
Ligthing7	73	319	Few anomalies, in different poin for each time series
Alabama_weather	15	2000	Longer time series with more noise
Currency2	20	2610	Some time series with trend, most not. Only small variations
Alabama_weather_3072	3072	3072	Identical time series, shifted by 1 in time

TABLE 4.1: Summary of the considered datasets



(A) BeetleFly (B) Coffee (C) Ham

(D) Lighting7 (E) Alabama_ weather (F) Currency2

FIGURE 4.1: Datasets plots

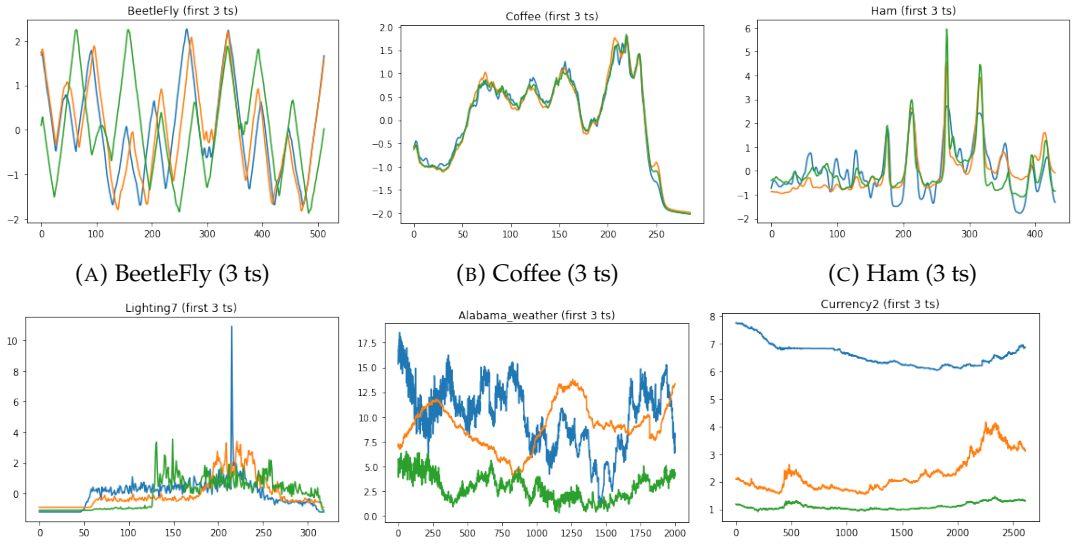
4.1.3 Metrics

The first thing to do in order to evaluate the generation algorithms is to choose *how*. Since our goal is to perform a general analysis (and not task-specific), we have decided to select some features of the original time series and observe whether or not they are maintained in the generated ones. In addition, we have also selected some metrics, to discover the relation between the original time series and the generated ones. Below is the list of what we decided to consider:

Mean. Although this is arguably the least interesting property we consider, it is a good way to understand if a method generates data on the same scale as the real one. If not, it might be necessary to process the generated data before being able to use it. It is therefore important to know it.

Variance. This is a relatively simple way to compare the distribution of the new time series with that of the original ones. Alone it would not be enough, but it is still an important property to check.

Spectral Entropy (see 2.1.1). Applying this to both the original data and the generated one, it allows us to see which method better preserves the amount of



(A) BeetleFly (3 ts) (B) Coffee (3 ts) (C) Ham (3 ts)

(D) Lighting7 (3 ts) (E) Alabama_ weather (3 ts) (F) Currency2 (3 ts)

FIGURE 4.2: Datasets plots (3 time series each)

information in the time series. If the generated data has a much lower Spectral Entropy than the original data, it might not be optimal for tasks such as forecasting.

Mutual Information (see 2.2.1) shared between the original data and the synthetic one. This is an important evaluation metric and, in contrast to the previous features, it compares directly the new data with the original one (instead of extracting properties from both and compare these).

Correlation (see 2.2.2) between original time series and the synthetic one. A very high or very small correlation mean that the generated data has a similar trend to the original one (i.e. almost the same or the opposite values with only small variations), and therefore will not be so useful. At the same time, a very small correlation indicate that the generated data is completely different from the original one, and therefore a model trained with this data could perform poorly on the real one.

Distribution (see 2.1.2). The distribution is a very important property of time series, and it is important to evaluate whether or not the generating algorithms are capable of preserving it.

Autocorrelation (see 2.1.3). The particularity of time series is that time is important. In particular, values at a given time might strongly depend on values at previous times. Autocorrelation measures this dependence, and it is useful to evaluate how much the time-dependencies are preserved.

4.2 Results

In this section, we will group the metrics listed in 4.1.3 into logical classes and for each of them we will analyze the results across all the datasets and all the generation methods.

Statistical metrics:. This group includes the mean and the variance. They are classic statistical measures that give a first insight into the considered data.

Distribution-based metrics:. This group includes the correlation with the original dataset, the distribution, and the auto-correlation. They are very important to

understand if the generated data preserves the distribution of the values in the original dataset, and, if yes, whether or not it does it generating completely new data or just by copying the original one.

Information-based metrics:. This group includes entropy and mutual information. They are both based on the information theory, and will analyze the amount of information preserved during the generation.

4.2.1 Statistical metrics

From Fig. 4.3 and Fig. 4.4 we can notice that the basic methods (DBA and AnomaliesInjection) generate data with very similar mean and variance w.r.t. the original one, for all the datasets considered. The fact that simple methods preserve statistical metrics is not a surprise, since they use the original time series as the starting point and only slightly modify them.

The performance or AR depends on the dataset considered. It is able to preserve the statistical metrics with BeetleFly and Coffee, but has worst results for example with Ham and Lighting7. This is due to the fact that the latter contains anomalies (spikes), and the auto-regressive model is not able to handle them.

Among the GAN-based methods, InfoGAN is the one that better preserves the statistical metrics during the generation. However, same as for AR, it has more difficulties when handling datasets with anomalies. The other GAN-based methods do not preserve the statistical metrics. This is due to the fact that the aim of GANs is not to preserve the statistical metrics, but rather focus on other properties such as the information contained in the data.

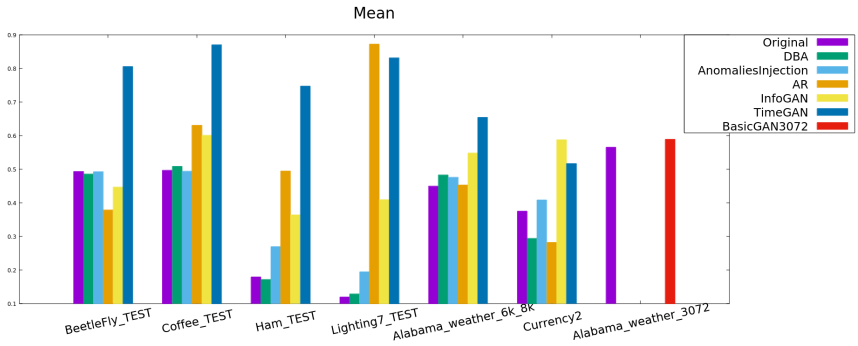


FIGURE 4.3: Normalized mean of the considered datasets

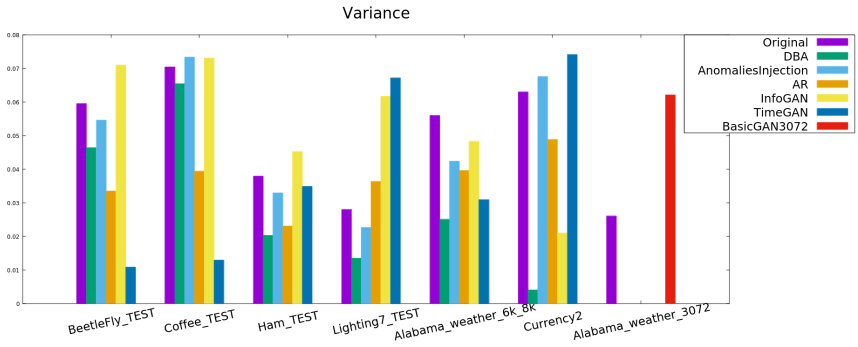


FIGURE 4.4: Normalized variance of the considered datasets

4.2.2 Distribution-based metrics

The results for Correlation are shown in Fig. 4.5, while those for Distribution and Autocorrelation are shown respectively in Fig. 4.6-4.11. and Fig. 4.12-4.14.

Again, the first thing we can notice is that simple methods (DBA and Anomalies Injection) have a very high correlation and similar autocorrelation function and distribution (even though Anomalies Injection has in the generated data some extreme values, which is completely predictable given how the method work). This shows once again that the generated data is only a minor modification of the existing data.

For AR, the correlation is very low. We can also notice that the generated values are much more concentrated in a smaller range. The method is not able to cover all the patterns of the original data. The autocorrelation function is in general similar to the autocorrelation function of the original data for the first part. However, it loses precision after a while. This is again due to the fact that after a certain point the new values are constructed based on synthetic values, increasing the error.

The correlation of the GAN-based methods is in general low, especially for TimeGAN and BasicGAN. The autocorrelation function is similar to the one of the original data when the dataset is large enough for the GAN model to capture the temporal evolution in the data (see Fig. 4.13 and Fig. 4.14). When instead the dataset is too small, GANs methods are not able to model it properly, as we can see from Fig. 4.12.

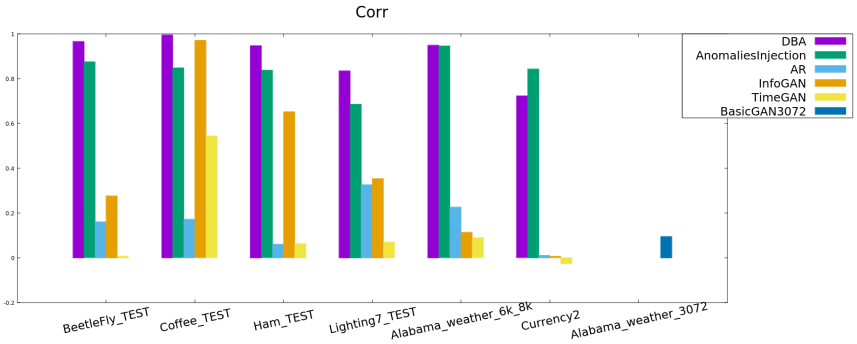


FIGURE 4.5: Correlation of the considered datasets w.r.t. the original data

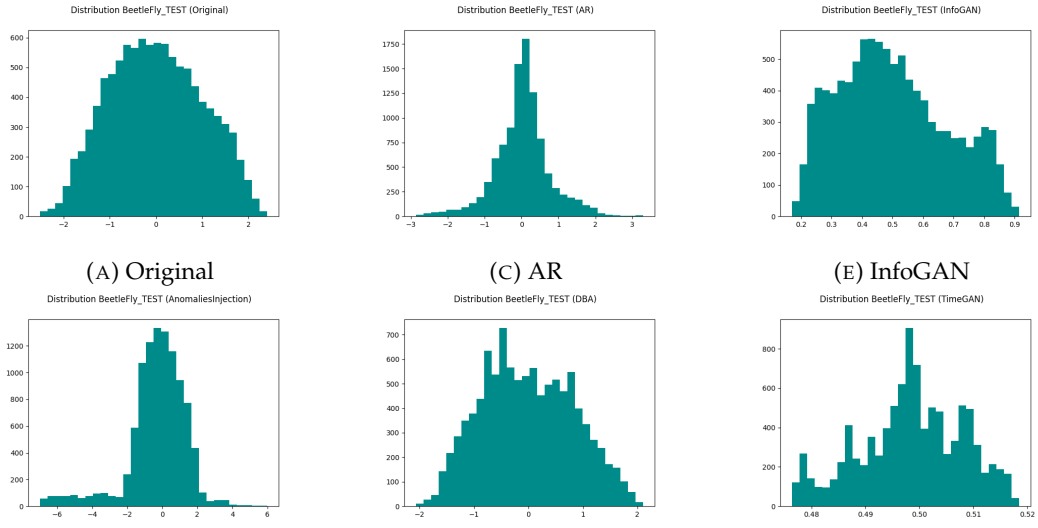
4.2.3 Information-based metrics

The results for Entropy and Mutual Information are shown in Fig. 4.15 and Fig. 4.16.

We can see that, with minor exceptions with the longer datasets, simple methods (DBA and Anomalies Injection) generate data with similar entropy to the original one, and share a high mutual information with it. This result was expected, since the generated data is only a perturbation of the original one.

AR also generates data with similar entropy to the original one, but has a smaller mutual information especially in the last three datasets. This is probably due to the fact that Lighting7 contains a lot of spikes (and our AR model is not complex enough to consider them), and the other two datasets are much longer. Since AR only consider the last points, after a certain threshold the new values are generated based on other synthetic values, thus losing precision.

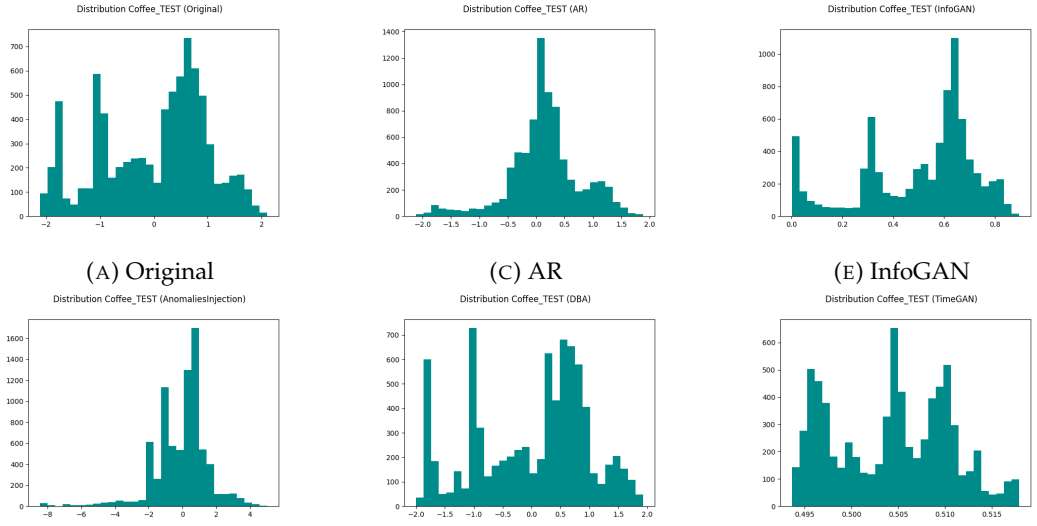
For GAN-based methods, we can immediately see that the entropy of the data generated with TimeGAN and BasicGAN is much higher. However, for TimeGAN the mutual information is similar to those with the other methods. We can deduce



(A) Original (C) AR (E) InfoGAN

(B) AnomaliesInjection (D) DBA (F) TimeGAN

FIGURE 4.6: Distribution of BeetleFly_TEST



(A) Original (C) AR (E) InfoGAN

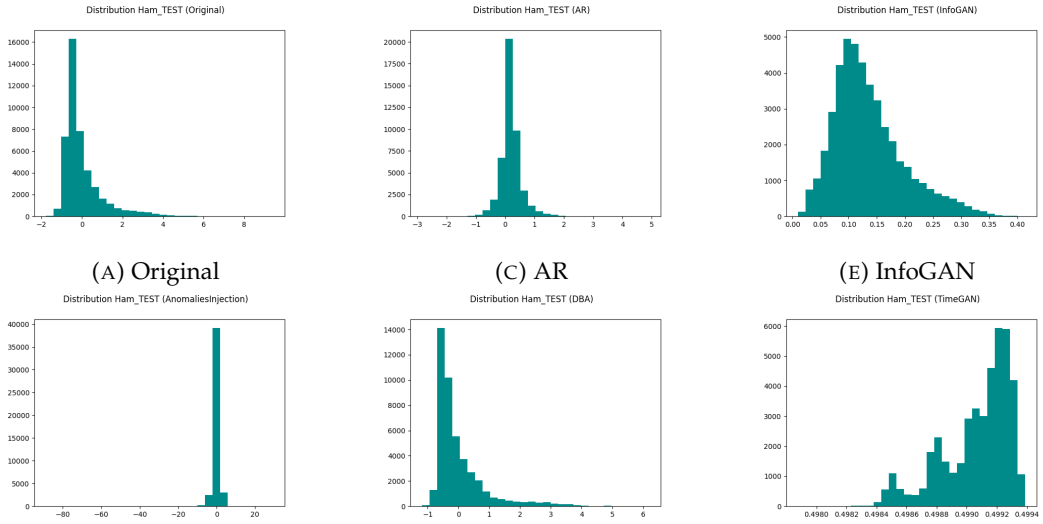
(B) AnomaliesInjection (D) DBA (F) TimeGAN

FIGURE 4.7: Distribution of Coffee_TEST

that the data generated with this method is less similar (i.e. has a higher distance) to the original one than those generated with other methods, but despite this, it shares a lot of information with it. InfoGAN has very similar results w.r.t. mutual information, but slightly better preservation of the entropy in the datasets without spikes.

4.2.4 Runtime

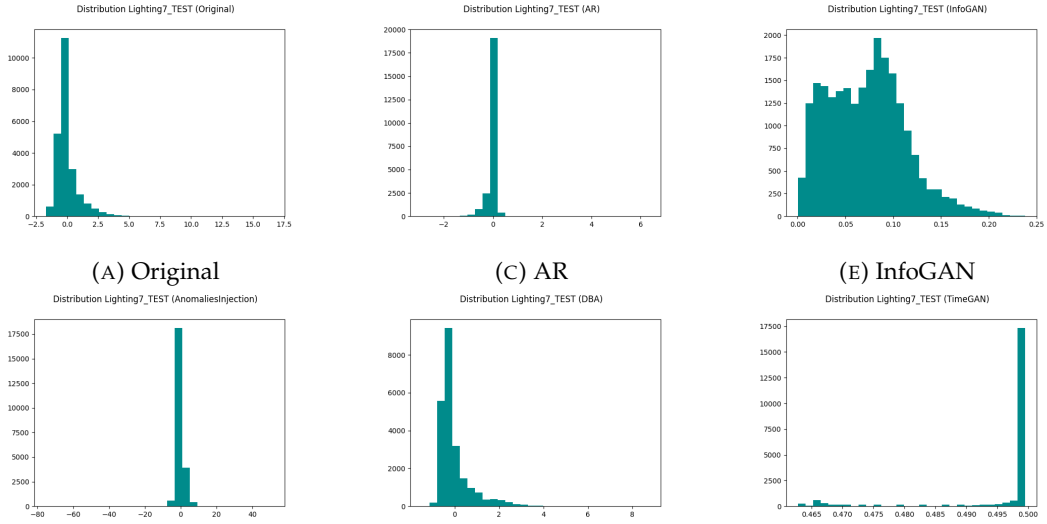
Below are presented the times (in seconds) needed to generate synthetic data with every method and for each of the 6 main datasets considered. The time presented only considers the generation (and training for InfoGAN and TimeGAN), but not the time needed to extract the metrics, plot the results, save the data, or any other task. In addition, it is to mention that the values presented are the results of a single



(A) Original (C) AR (E) InfoGAN

(B) AnomaliesInjection (D) DBA (F) TimeGAN

FIGURE 4.8: Distribution of Ham_TEST



(A) Original (C) AR (E) InfoGAN

(B) AnomaliesInjection (D) DBA (F) TimeGAN

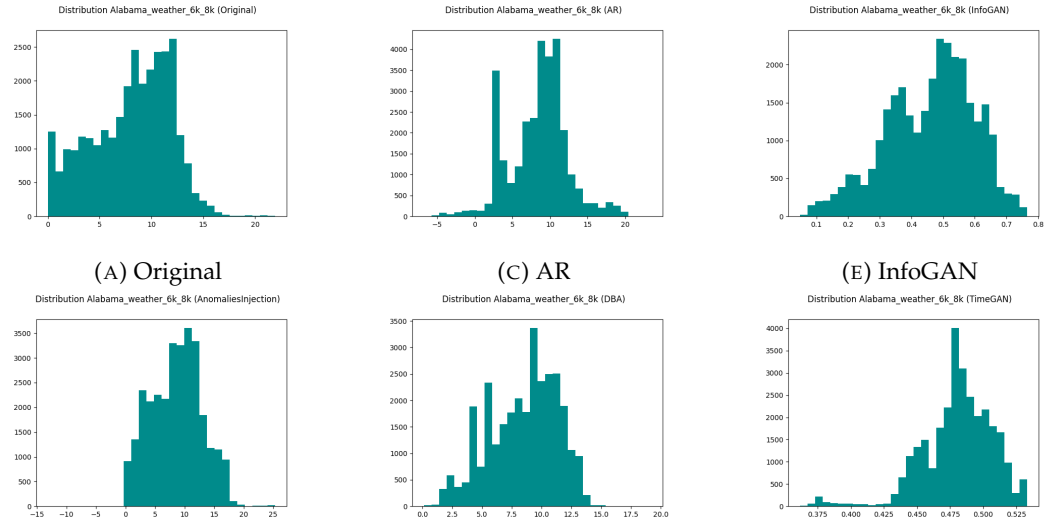
FIGURE 4.9: Distribution of Lighting7_TEST

experiment. Although it would be more correct to do multiple runs for each value and consider the mean, in this case the exact result is not particularly important. What is investigated here is rather the order of growth of each algorithm, and the relation between them.

Runtime (s)	BeetleFly (20x512)	Coffee (28x286)	Ham (105x431)	Ligthing7 (73x319)	Alabama_weather (15x2000)	Currency2 (20x2610)
AnomaliesInjection	3.4792	3.7648	13.2981	10.4487	3.9154	2.4432
AR	8.6032	4.1136	26.3448	3.3944	57.1649	18.2239
DBA	5.1887	4.0671	104.6299	33.9407	54.898	143.2891
InfoGAN (200 eps)	222.7651	232.73	1381.9785	400.3487	2576.8138	2659.7723
TimeGAN (200 eps)	301.3007	343.3313	319.0164	288.4910	266.4494	269.8655

TABLE 4.2: Runtime of the different methods

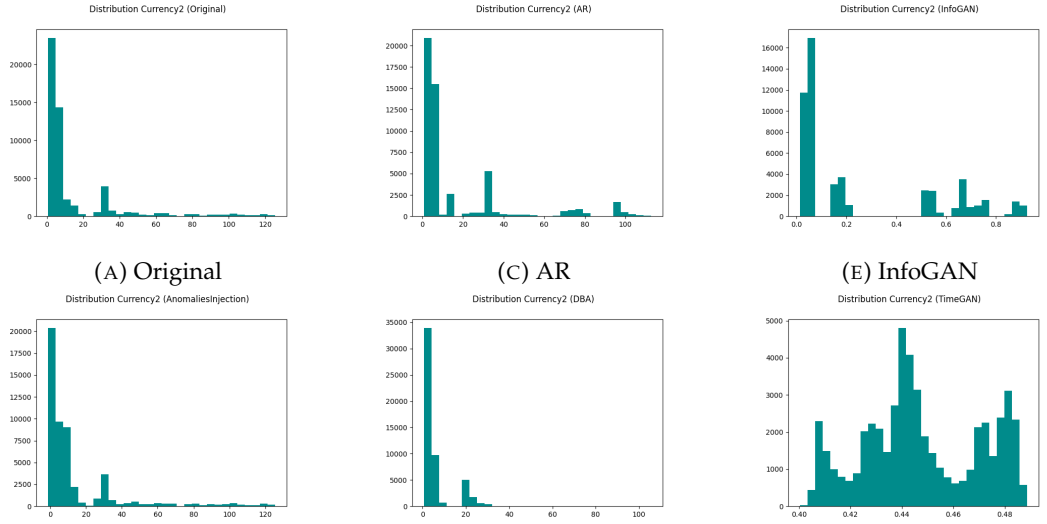
From table 4.2 we can see that AnomaliesInjection, AR and DBA require very few time compared to GAN-based methods. In addition, the required time is less influenced by the size of the input dataset.



(A) Original (C) AR (E) InfoGAN

(B) AnomaliesInjection (D) DBA (F) TimeGAN

FIGURE 4.10: Distribution of Alabama_weather_6k_8k



(A) Original (C) AR (E) InfoGAN

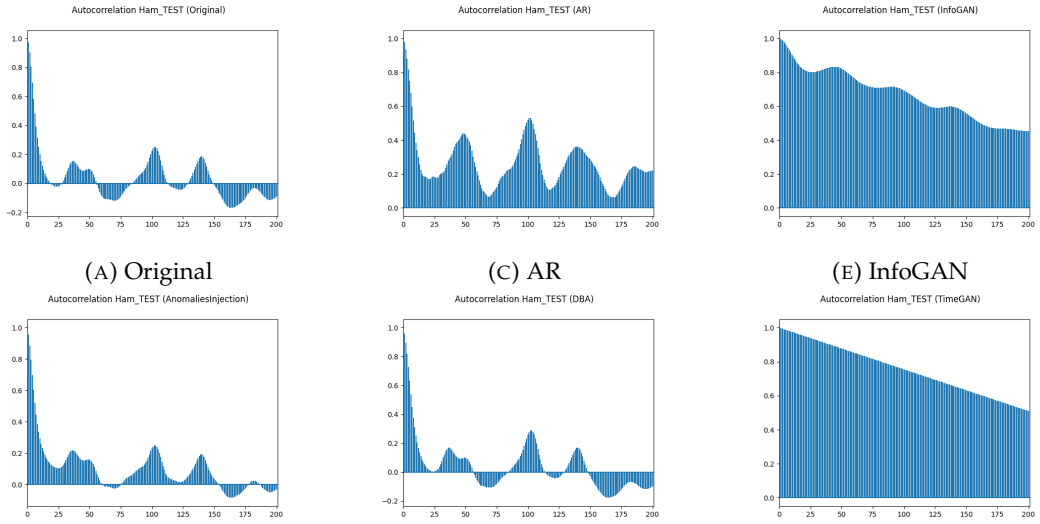
(B) AnomaliesInjection (D) DBA (F) TimeGAN

FIGURE 4.11: Distribution of Currency2

Generation with InfoGAN requires much longer, and the required time is heavily conditional on the size of the input dataset. For small datasets, the time required by TimeGAN is similar to that of InfoGAN. However, for TimeGAN the time remains constant with the growth of the input dataset, leading to a much faster generation with bigger inputs.

4.3 Recommendations

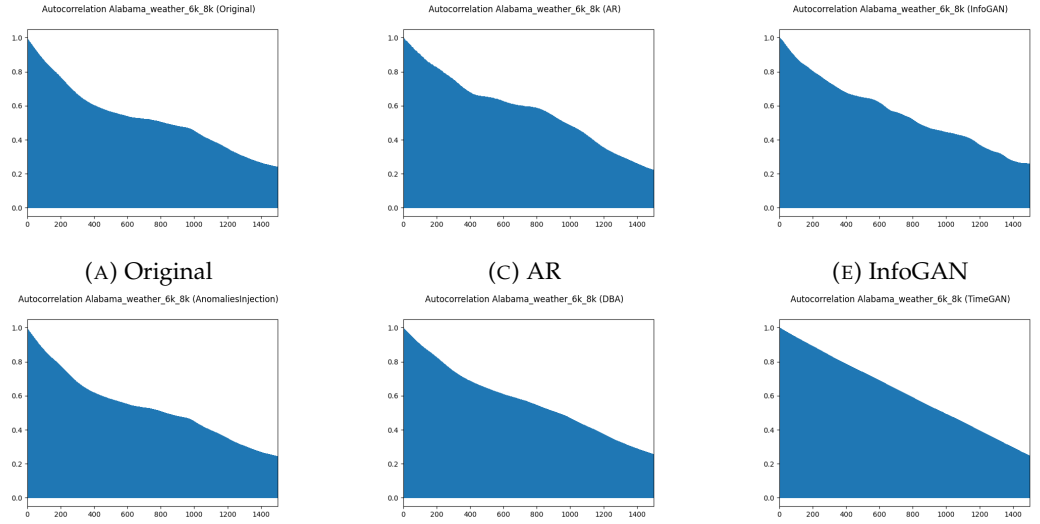
In this section, we try to summarize which class of algorithms is better in which situation, based on the results obtained. In particular, we will consider three variants: the similarity of the generated data, the size, and the presence of anomalies. In the first case, we consider a generated time series to be similar to the original one when



(A) Original (C) AR (E) InfoGAN

(B) AnomaliesInjection (D) DBA (F) TimeGAN

FIGURE 4.12: Autocorrelation of Ham_TEST



(A) Original (C) AR (E) InfoGAN

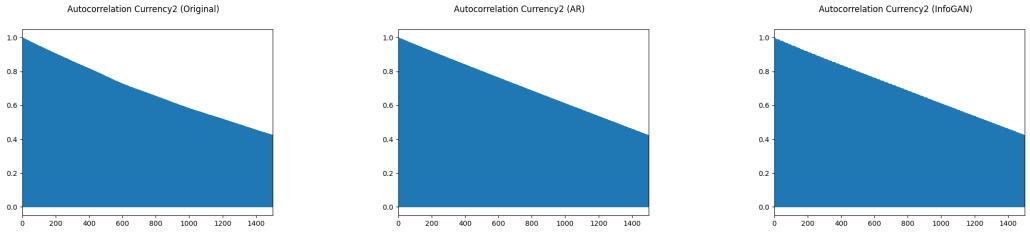
(B) AnomaliesInjection (D) DBA (F) TimeGAN

FIGURE 4.13: Autocorrelation of Alabama_weather_6k_8k

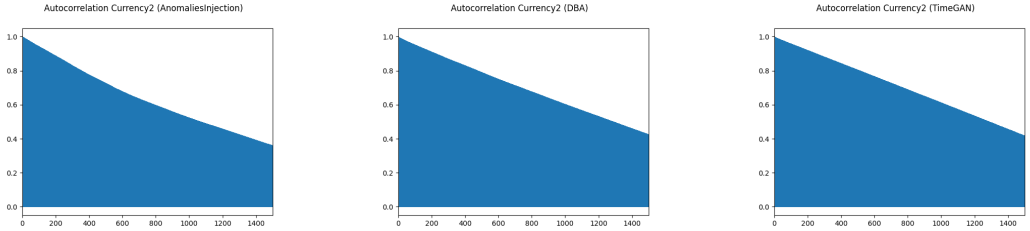
it consists of almost the same values, with only some minor changes. More formally, two similar time series have a high correlation and a small RMSE (Root Mean Squared Error, not analyzed here). The second variable depends on the size of the required output. We distinguish two cases: when it is enough to double the initial (original) input, and when instead we need to generate a bigger dataset. Lastly, we distinguish input datasets with anomalies (e.g. 4.1d) from datasets without anomalies (e.g. 4.1b).

The classes of algorithms considered are those just analyzed, meaning simple methods (including DBA and Anomalies Injection), autoregressive methods (AR), and ML methods (e.g. GANs).

The recommendations derived are illustrated in Fig. 4.17.



(A) Original (C) AR (E) InfoGAN



(B) AnomaliesInjection (D) DBA (F) TimeGAN

FIGURE 4.14: Autocorrelation of Currency2

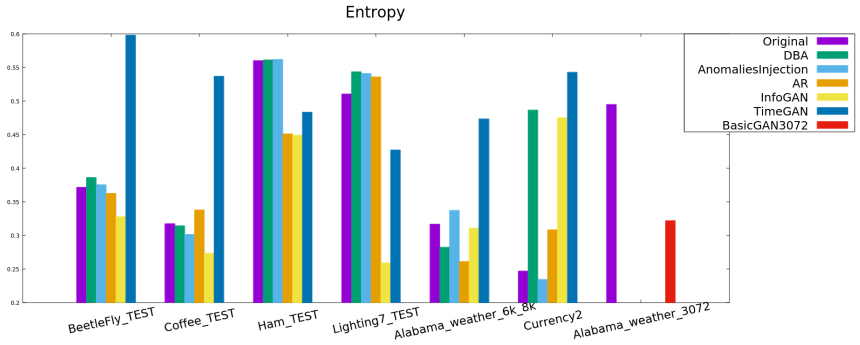


FIGURE 4.15: Entropy of the considered datasets

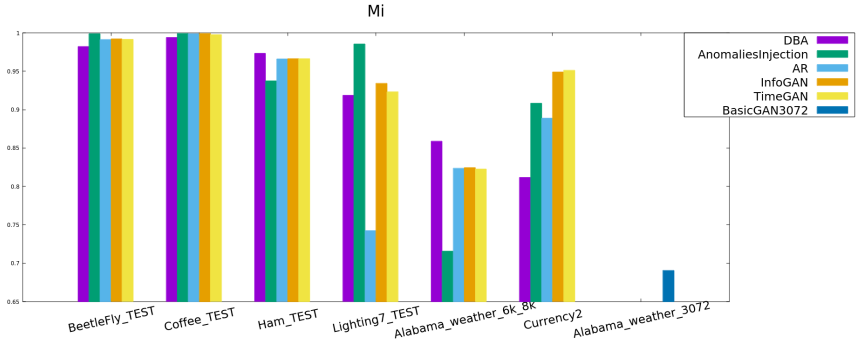


FIGURE 4.16: Mutual information of the considered datasets w.r.t. the original data

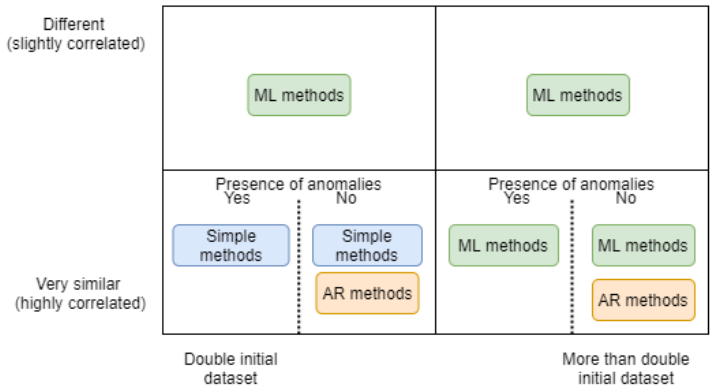


FIGURE 4.17: Recommendations

Chapter 5

Augmentation Framework

To automate the experimental part of this work, and to facilitate its replicability, we have implemented a tool that brings together the selected algorithms[1]. It has been created for Ubuntu 20.4, and it is mainly written in python 3.6. The tool allows to generate new time series starting from any dataset with the right format (.csv file with time series as columns). The datasets considered in this work are included with the code, however a user can add its own.

It is possible to choose between 6 generation techniques: AnomaliesInjection, AR, DBA, Basic GAN, InfoGAN, and TimeGAN. The tool takes care of hiding the differences in their implementation (different programming languages, input format, etc.) and standardizes the output format. In addition, upon request, it extracts some metrics both from the new and the original data in order to evaluate the generation.

5.1 Implementation

As mentioned, the sources for the different techniques are very heterogeneous. Some of them are written in python, others in java. Some have a ready-to-work implementation, while others are just a library, or even an article, and need additional work to be used. In addition, some implementation use different version of the same library (in particular tensorflow), or require different input format. TimeGAN and InfoGAN also produces sub-samples of time series, which needs to be reconstructed into full time series to be compared with the other techniques.

Our contribution with this tool is to hide all these differences and make it possible to use the different techniques in a uniform and seamless way. Table 5.1 summarize these differences.

[1]The code and the datasets are publicly available at https://github.com/Fontanjo/TS_generation_benchmark

Technique	Source	Last version data	Language	Limitations
Anomalies Injection	*Agots*	2019	Python	It is a library
DBA	Forestier et al., 2017	2017	Java	Requires different input format
AR	Brownlee, 2020	2020	Python	It is an article, no implementation
Basic GAN	*dbiir/TS-Benchmark*	2021	Python	Requires specific input shape
TimeGAN	*TimeGAN*	2021	Python	Requires tensorflow==1.15.0 Provides no output, only visual analysis Produces sub-samples
InfoGAN	*tsgen*	2021	Python	Requires tensorflow==1.4.0 Produces sub-samplese

TABLE 5.1: Summary of the different sources used

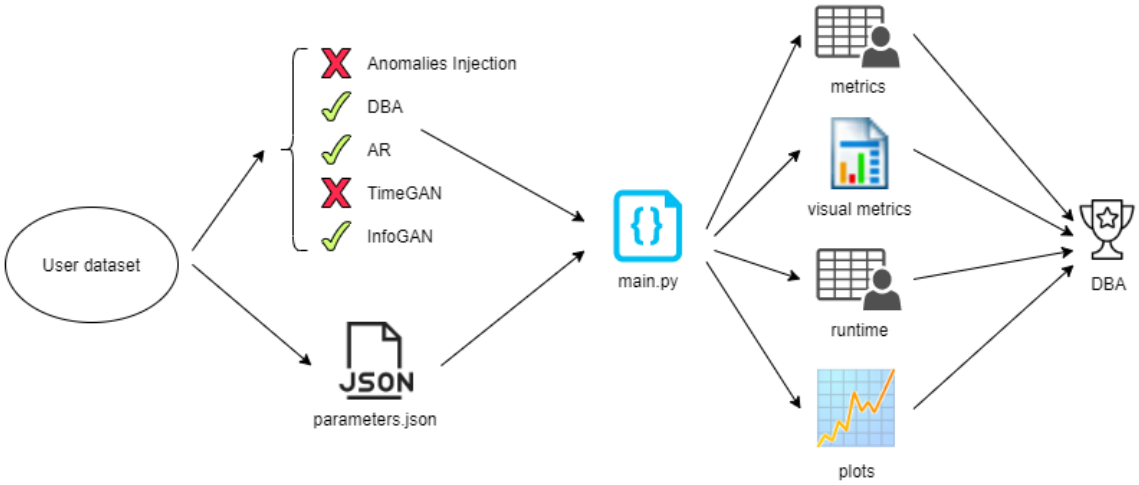


FIGURE 5.1: Use example: searching for the best algorithm combining the 3 scenarios

5.2 Scenarios

The main goal of the generation tool is to simplify the generation of synthetic time series using different algorithms, as well as visualizing and analyzing the results. Figure 5.1 shows how to use the tool to choose the best algorithm for a given situation, and below is a list with a more specific description of the main use scenarios:
Scenario 1: Visualize datasets. The first thing a user might be interested in is to visualize a dataset. In this scenario, the user can add its own dataset to the framework and use the provided tools to visualize it. In addition, he can visualize the distribution of the dataset, as well as the distribution of each single time series in it. The same thing holds for autocorrelation, for which it is possible to generate a single plot for the entire dataset or a separate plot for each time series.
Scenario 2: Analyze datasets. In this scenario, a user studies a dataset based on a set of properties and metrics. It is possible to extract the mean, the spectral entropy, and all the properties described in 2.1. It is also possible to specify two datasets and compute metrics s.a. mutual information and correlation between them. All these results are stored in a csv file, and they can be plotted with gnuplot using another provided script.
Scenario 3: Augment datasets. In this scenario, the user has a dataset he wants to augment. He can add this dataset to the framework, then select the algorithm he wants to use and customize the parameters. For example, he can specify the number of epochs for ML-based methods, or frequency and type of anomalies for anomalies injection. It is also possible to specify the size of the generated data (number of time series and their length).

Chapter 6

Conclusion

In this work we presented a tool to evaluate different algorithms to generate synthetic time series. To do this, we presented some of the state-of-the-art methods to generate new time series and show some of the most diffused implementations. We then discussed how to evaluate the quality of generated data through time series properties and metrics. With these concepts, we used our tool to generate data for 6 different datasets and 6 generating methods, and analyzed the results. Finally, we presented some conclusions and suggestions that could be drawn from the results.

A natural future work on this tool is the introduction of new generating algorithms. For example, more sophisticated auto-regressive models, other anomalies injection techniques, or VAEs-based algorithms. In addition, there are many properties and metrics to evaluate time series that could be included in the code, for example DTW-distance. Finally, other things that would be useful to add are techniques to visually evaluate the generated data, such as PCA or t-SNE.

Another direction to continue this work is to evaluate the data generated on a time series analysis task. For example, evaluate whether the synthetic data created help improve the accuracy of a model for anomaly detection, classification or prediction.

Bibliography

Briandet, Romain, E. Katherine Kemsley, and Reginald H. Wilson (1996). "Discrimination of Arabica and Robusta in instant coffee by Fourier transform infrared spectroscopy and chemometrics". In: *Journal of Agricultural and Food Chemistry* 44.1, pp. 170–174. DOI: 10.1021/jf950305a. URL: https://hal.archives-ouvertes.fr/hal-01606904.

Brownlee, Jason (Aug. 2020). *Autoregression Models for Time Series Forecasting With Python*. URL: https://machinelearningmastery.com/autoregression-models-time-series-forecasting-python/.

Chen, Peiyuan et al. (June 2010). "ARIMA-Based Time Series Model of Stochastic Wind Power Generation". In: *Power Systems, IEEE Transactions on* 25, pp. 667 – 676. DOI: 10.1109/TPWRS.2009.2033277.

Chen, Xi et al. (2016). "InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee et al. Vol. 29. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2016/file/7c9d0b1f96aebd7b5eca8c3edaa19ebb-Paper.pdf.

Dataset: BeetleFly. URL: https://timeseriesclassification.com/description.php?Dataset=BeetleFly.

Dataset: Coffee. URL: https://timeseriesclassification.com/description.php?Dataset=Coffee.

Dataset: Ham. URL: https://timeseriesclassification.com/description.php?Dataset=Ham.

Dataset: Lighting7. URL: https://timeseriesclassification.com/description.php?Dataset=Lightning7.

Dbiir. *dbiir/TS-Benchmark*. URL: https://github.com/dbiir/TS-Benchmark.

Dionisio, Andreia, Rui Menezes, and Diana A. Mendes (2004). "Mutual information: a measure of dependency for nonlinear time series". In: *Physica A: Statistical Mechanics and its Applications* 344.1. Applications of Physics in Financial Analysis 4 (APFA4), pp. 326–329. ISSN: 0378-4371. DOI: https://doi.org/10.1016/j.physa.2004.06.144. URL: https://www.sciencedirect.com/science/article/pii/S0378437104009598.

eXascaleInfolab. *tsgen*. URL: https://github.com/eXascaleInfolab/tsgen.

Forestier, Germain et al. (2017). "Generating Synthetic Time Series to Augment Sparse Datasets". In: *2017 IEEE International Conference on Data Mining (ICDM)*, pp. 865–870. DOI: 10.1109/ICDM.2017.106.

Gao, Jingkun et al. (2020). *RobustTAD: Robust Time Series Anomaly Detection via Decomposition and Convolutional Neural Networks*. arXiv: 2002.09545 [cs.LG].

Generative Adversarial Network (GAN). URL: https://docs.paperspace.com/machine-learning/wiki/generative-adversarial-network-gan.

Goodfellow, Ian J. et al. (2014). *Generative Adversarial Networks*. arXiv: 1406.2661 [stat.ML].

Ham, Lucy, Rowan Brackston, and Michael Stumpf (Apr. 2019). *Extrinsic noise and heavy-tailed laws in gene expression*. DOI: 10.1101/623371.

jsyoon0823. *TimeGAN*. URL: https://github.com/jsyoon0823/TimeGAN.

Kang, Yanfei, Rob J. Hyndman, and Feng Li (May 2020). "GRATIS: GeneRAting TIme Series with diverse and controllable characteristics". In: *Statistical Analysis and Data Mining: The ASA Data Science Journal* 13.4, 354–376. ISSN: 1932-1872. DOI: 10.1002/sam.11461. URL: http://dx.doi.org/10.1002/sam.11461.

KDD-OpenSource. *Agots*. URL: https://github.com/KDD-OpenSource/agots.

Laptev, N. (2018). "AnoGen : Deep Anomaly Generator Nikolay Laptev". In:

Le Guennec, Arthur, Simon Malinowski, and Romain Tavenard (Sept. 2016). "Data Augmentation for Time Series Classification using Convolutional Neural Networks". In: *ECML/PKDD Workshop on Advanced Analytics and Learning on Temporal Data*. Riva Del Garda, Italy. URL: https://halshs.archives-ouvertes.fr/halshs-01357973.

Mutual Information diagram. URL: https://commons.wikimedia.org/wiki/File:Entropy-mutual-information-relative-entropy-relation-diagram.svg.

Paige, Robert and A. Trindade (Jan. 2010). "The Hodrick-Prescott Filter: A special case of penalized spline smoothing". In: *Electronic Journal of Statistics* 4. DOI: 10.1214/10-EJS570.

Scikit-learn: Clustering. URL: https://scikit-learn.org/stable/modules/clustering.html#mutual-info-score.

Talbot, Paul W et al. (June 2019). "Correlated Synthetic Time Series Generation using Fourier and ARMA". In: URL: https://www.osti.gov/biblio/1634101.

Vajapeyam, Sriram (2014). "Understanding Shannon's Entropy metric for Information". In: *CoRR* abs/1405.2061. arXiv: 1405.2061. URL: http://arxiv.org/abs/1405.2061.

Volny, Petr, David Novak, and Pavel Zezula (Apr. 2012). "Employing Subsequence Matching in Audio Data Processing". In:

Wen, Qingsong et al. (Feb. 2020). "Time Series Data Augmentation for Deep Learning: A Survey". In: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*. DOI: 10.24963/ijcai.2021/631.

Yoon, Jinsung, Daniel Jarrett, and Mihaela van der Schaar (2019). "Time-series Generative Adversarial Networks". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2019/file/c9efe5f26cd17ba6216bbe2a7d26d490-Paper.pdf.