Synthetic Financial Time Series Generation with Regime Clustering

Kirill Zakharov *, Elizaveta Stavinova, and Alexander Boukhanovsky

National Center for Cognitive Research, ITMO University, Saint Petersburg 199034, Russia;
Email: stavinova@itmo.ru (E.S.), avbukhanovskii@itmo.ru (A.B.)
*Correspondence: kazakharov@itmo.ru (K.Z.)

***Abstract*—Methods for synthetic data generation are extremely valuable nowadays since they allow researchers and practitioners to develop and test their models without the risk and cost associated with using real data. In this paper, we propose a method for the generation of synthetic financial time series. The method adopts time series regimes clustering to perform generative models training on the data from each cluster separately. Also, we suggest the modification of Quantum Generative Adversarial Networks (QuantGAN) architecture that is able to produce synthetic data with frequency characteristics closer to the corresponding real-world time series ones. Our experiments show that (1) synthetic financial time series can be effectively generated by our method; (2) the distribution characteristics of synthetic time series generated by the method are closer to the initial ones in comparison with Fourier Flows and QuantGAN; (3) training the forecasting model on the synthetics generated by the proposed method (Fourier Flows model is used within it) can reduce the forecasting error on the real-world series.**

***Keywords*—regime clustering, Generative Adversarial Networks (GAN), normalising flows, time series generation, synthetic time series**

I. Introduction

The task of financial time series forecasting is demanded for the last decades [1]—the industry is interested in the accurate predictions of financial indicators such as volatility [2], stock prices [3] or stock exchange rate [4]. Despite that these questions have been studied for a long time, there are still several difficulties the researchers are faced with, such as data sharing restrictions or the lack of historical data [5]. Considering that modern forecasting models frequently rely on machine learning [6], the challenges listed above could lead to a model of the undesired quality due to the insufficient training data. One of the promising approaches to cope with this problem is the usage of the synthetic data. In particular, one can bypass the data sharing restrictions by sharing not the initial data, but the generative model trained on it. Moreover, a trained generative model can be used to produce sufficient amount of data replacing or supplementing historical data. Recent studies report that the synthetic data can be used to augment the historical data, that results in improving of machine learning models performance [7, 8].

In this paper, we face the task of synthetic time series generation on the basis of a particular real-world financial time series. Financial time series reflects the dynamics of resources redistribution in the economy. These dynamics are usually expressed via the sequences of different financial indicators values. The specificity of this data can be described as follows [9]: financial time series exhibit a multiscale and evolving nature. They contain weekly and seasonal patterns, as well as trends, and are influenced by crises, calendar-specific events such as holidays, and consumer behaviour patterns associated with them. Thus, the financial time series are non-stationary, non-periodical, with erratic transitions between states affected by many factors. As a result, inferring an explicit probabilistic financial time series model is a difficult task, that, in turn, makes it necessary to apply Machine Learning (ML) based models for synthetic financial time series generation. However, in this work, we train ML-based generative models not on the whole initial time series (as is usually performed), but divide the initial time series into regimes, cluster them according to their characteristics and train several generative models on the data of each cluster. This is performed to cope with the complicated financial time series nature. To summarize, the impact of this study is as follows:

- We propose a new method for synthetic financial time series generation based on the clustering of time series regimes;
- We use Fourier Flows (FF) [10] and Quantum Generative Adversarial Networks (QuantGAN)[11] generative models within our method (they are called CLustering for Fourier Flows (CLFF) and CLustering for GAN (CLGAN), respectively, throughout the study) and, also, propose the modification of QuantGAN that apply within the method, too (we call it CLustering and Supervisor for GAN (CLSGAN));
- We show in our experimental study of the proposed method that it is efficient in generating synthetic financial time series and moreover the quality of its realisations by three generative models (CLFF, CLGAN and CLSGAN) are higher than the quality of FF and QuantGAN according to several criteria.

Manuscript received June 4, 2023; revised June 26, 2023; accepted July 13, 2023; published December 14, 2023.

The code, data and experimental results are given on GitHub (https://github.com/AlgoMathITMO/CLSGAN).

This paper is organized as follows: Section II is devoted to a discussion of existing research on the topic; Section III describes the proposed method for synthetic financial time series generation in details; Section IV is about the experiments performed using the proposed method; Section V is related to the results of the mentioned experiments and their discussion; Section VI contains the conclusions, as well as research limitations and directions for further research.

II. Literature Review

Since the task regarded in this paper relates to such topics as financial time series, time series clustering, synthetic time series generation and synthetic data quality assessment, further we provide a review covering all the topics. Before we begin, it is worth noting that the subject of time series analysis and forecasting is not new and has been extensively studied in various fields, including medical [12] and financial data [13, 14]. However, there are still several challenges in this area, which will be discussed below.

A. Financial Time Series Challenges

There are specific problems, that researchers face with during modelling financial time series. Firstly, the daily returns exhibit a heavy-tailed distribution and have more peaks in comparison to the initial distribution of time series [15]. The second issue is volatility clustering, when regimes are changing with high volatility and low volatility periods. Due to the low probability of high volatility periods existence within time series values, the models are not able to precisely learn the initial dynamic [11]. The next issue is the absence of autocorrelation of returns [9], that means that returns have less dependence on previous values with time scale.

B. Time Series Clustering

A typical approach for time series clustering is usually tries to divide a set of time series into groups with similar properties [16, 17]. Some methods exploit the distance-based approaches for clustering [18], the others are based on neural network models [19]. Also, there is a group of methods that cluster time series using the regimes features [20, 21]. But in this work, we face a different task as we need to divide time series into regimes firstly, and then to cluster the obtained regimes. To the best of our knowledge, there are no works dedicated to regimes clustering within one time series yet.

C. Synthetic Time Series Generation

We propose the following classification of methods for synthetic time series generation (and provide the description of the most common models in each group). The first group consists of Generative Adversarial Network-based (GAN-based) models: Wasserstein Generative Adversarial Network (WGAN) [22] (based on Wasserstein GAN for images generation), Time-series Generative Adversarial Network (TimeGAN) [23] (uses recurrent neural networks), QuantGAN [11] (temporal convolution networks, overperforms TimeGAN), Time Series Generative Adversarial Network (TSGAN) [24] and Unseen Transition Suss GAN (UTSGAN) [25] (exploit spectrogram of time series, large training time). Another group of methods is based on variational autoencoders. They have fast sampling mechanism and short training time but demonstrate the results of the low quality [26]. The third group consists of methods based on normalising flows such as Fourier Flows [10] (the best results in comparison with [22, 23]). The last group consists of transformers: for example, Transformer-based Time-Series Generative Adversarial Network (TTS-GAN) [27]. It shows a good performance, especially on a big data, but has a huge training time.

D. Synthetic Data Quality Assessment

Synthetic data quality assessment Synthetic data quality assessment can be conducted by the several ways. The first one is to estimate the similarity of synthetic and initial data distributions, that can be done via computation of divergence (e.g., Kullback-Leibler or Jensen-Shannon divergences [27, 28]), comparison of statistical characteristics (e.g., mean, standard deviation, skewness, or kurtosis [29]) and empirical distribution functions (with the help of Kolmogorov- Smirnov test or χ^2-test [30]). To assess quality in the case of synthetic time series generation, a second method is to analyze the autocorrelations [22] and frequency components [10] of both the synthetic and initial time series. Finally, there is a possibility to assess the synthetic data quality using the error obtained in the result of applying the forecasting model trained on a synthetic data to the corresponding real-world time series values [23].

III. Materials and Methods

A. Pipeline

Consider an initial time series $X = (x_1, x_2, \dots, x_N)^T \in \mathbb{R}^{N\times 1}$, that is defined on an interval $[t_1, t_N]$, $t_1 < t_N \in \mathbb{R}^+$. Time series X is non-stationary reflecting erratic changes between different states of the corresponding process (no constraints on autocorrelation function of X are imposed). Every time series can be divided into several segments, where the time series behaviour differs from its behaviour in other segments. These segments are called regimes, while the time points between the regimes are called change points. More formally, we denote change points by $\tau = (\tau_0, \tau_1, \tau_2, \dots, \tau_l)$, where $\tau_0 = 0$ is the initial point and l is the number of regimes. Then, the regime $\mathcal{R}(\tau_i), i \in \{1, \dots, l\}$ of the time series in the interval $[\tau_{i-1}, \tau_i)$ is defined as:

$$R(\tau_i) = X_{[\tau_{i-1}:\tau_i)} \tag{1}$$

Let us also mention that the time series log return is $log\left(\frac{X_t}{X_{t-1}}\right)$.

Note that the pipeline of the proposed method is illustrated in Fig. 1. Let us provide a more detailed description of the pipeline below. First, we start with change points detection and regimes allocation according to the obtained points (red blocks in Fig. 1). Then for every regime we calculate the vector of its characteristics (the red dashed box on the pipeline). For this purpose, we define the operators $\chi^{(i)}: \mathbb{R}^{|\mathcal{R}(\tau_i)|} \to \mathbb{R}^{d_\chi}$, where $|\mathcal{R}(\tau_i)|$ is the number of elements in the regime. Operators map the regime to the vector of its characteristics of the dimension d_χ:

$$\chi^{(i)}\mathcal{R}(\tau_i) = y^{(i)} = \left(y_1^{(i)}, y_2^{(i)}, \ldots, y_{d_\chi}^{(i)}\right). \tag{2}$$

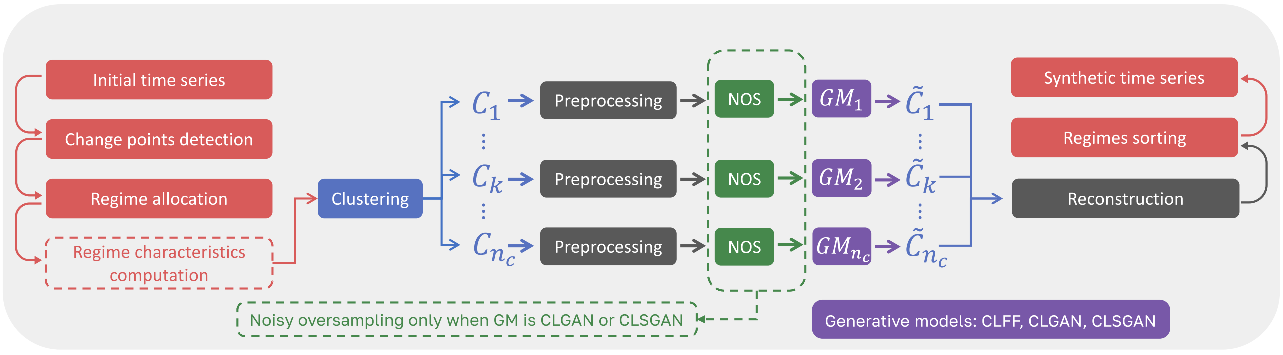


Figure 1. The pipeline of the proposed method.

Then we apply the respective operators on every regime within a time series and get the vectors $y^{(i)}$, $i \in \{1, \ldots, l\}$. After that, we cluster the regimes according to their characteristics $y^{(i)}$, $i \in \{1, \ldots, l\}$ (the number of clusters is predefined and equals n_c; blue box in Fig. 1). The obtained clusters are denoted as $C_1, C_2, \ldots, C_{n_c}$. Further, each cluster is inputted into the grey block of the pipeline, which corresponds to preprocessing procedures.

Moreover, there is an optional green dashed block in Fig. 1, which should be applied only when the model for synthetic data generation is GAN-based. In this case, every cluster is extended via noisy oversampling.

Once the clusters are preprocessed, generative models (GMs) $GM_1, \ldots, GM_{n_c}$ are trained on the data from the corresponding clusters $C_1, C_2, \ldots, C_{n_c}$ along with the noise drawn from the standard Gaussian distribution $\mathcal{N}(\mathbf{0}, I)$, where $\mathbf{0}$ is a zero vector and I is an identity matrix (purple blocks in Fig. 1). The output of the generative models is the synthetic clusters $\tilde{C}_1, \ldots, \tilde{C}_{n_c}$.

The generated synthetic clusters $\tilde{C}_1, \ldots, \tilde{C}_{n_c}$ are fed to the grey block of the pipeline corresponding to the reconstruction process, where the procedures inverse to the procedures from the preprocessing block are performed. Further, we sort the regimes from the synthetic clusters in order corresponding to the order of the regimes in the initial time series. As a result, we obtain a synthetic time series (red blocks on the right side of the pipeline).

B. *Generative Models*

Now we describe the generative models that are applied to the regimes clusters $C_1, C_2, \ldots, C_{n_c}$ according to the pipeline (purple block in Fig. 1) to obtain a set of synthetic clusters $\tilde{C}_1, \tilde{C}_2, \ldots, \tilde{C}_{n_c}$.

1) *CLustering for Fourier Flows (CLFF)*

The first proposed generative model is CLFF, which is a Fourier Flows model [10] trained on the time series data belonging to a particular cluster. The choice of the model is determined by its fast sampling, short training time and generated synthetics quality [31].

2) *CLustering for GAN (CLGAN)*

The second proposed generative model is CLGAN with a QuantGAN architecture [11], that is trained on the data from a certain regimes cluster. In Ref. [11], the generator and the discriminator are based on temporal convolution networks (TCNs). In our work, we use the TCN architecture similar to [11], but with some modifications, namely, the dropout mechanism for stable learning in each TCN block. This block is represented as:

$$\varphi^{(i)}\left(X, \zeta^{(i)}\right) = (d_2 \circ \psi_2 \circ f_2 \circ d_1 \circ \psi_1 \circ f_1)(X), \tag{3}$$

where $\circ$ is the composition operator, f_j is the convolution layer, ψ_j is the PReLU activation function, d_j is the dropout, $j \in \{1,2\}$ is the layer index, i is the block index, $\zeta^{(i)}$ is the set of parameters:

$$\zeta^{(i)} = \{d_I, d_H, d_O, N_K, N_D, N_P, N_S, dr, lkr\}, \tag{4}$$

where d_I is the input dimension of convolution layers, d_H is the hidden dimension of convolution layers, d_O is the output dimension of convolution layers, N_K is the kernel size, N_D is the dilation parameter value, N_P is the padding parameter value, N_S is the stride parameter value, dr is the dropout rate, lkr is the rate of LeakyReLU which is applied to the output of the skip connections procedure. Also, we apply batch normalisation after each TCN.

3) *CLustering and Supervisor for GAN (CLSGAN)*

One of the possible problems that GAN-based models can face is too diverse generated data which results in a large gaps after the regimes reconstruction. To overcome the aforementioned problem we propose the modification of CLGAN that allows to make the generation process more stable and to obtain synthetic time series that are close to the real-world ones. The modification contains the new control elements by analogy with TimeGAN [23]: the authors apply the supervisor in the latent space after the encoder to detect the dynamics of the real data. In turn, CLSGAN has the following data transformation scheme (represented in Fig. 2): the generator G takes the noise

$\mathcal{N}(\mathbf{0}, I)$ as an input and produces the synthetic data $\hat{X}$, that serves as an input to the supervisor S. The supervisor is a TCN with the loss function defined as $MSE(\tilde{X}, X) + \mathbb{E}[D_2(\log(\tilde{X}))]$, where $\tilde{X}$ is the output of S. The supervisor aim is to detect the initial time series dynamics and to approximate the synthetic time series dynamics to the initial one. Two discriminators D_1 and D_2 are used to train the generator G: D_1 tries to distinguish the initial data X from the generated data $\hat{X}$, while D_2 tries to distinguish X from the supervised data $\tilde{X}$.

Thus, the generator G is trained by the joint loss $\alpha\mathbb{E}[D_1(\log(\hat{X}))] + \beta\mathbb{E}[D_2(\log(\tilde{X}))]$, where $\alpha + \beta = 1$. Thereby, the supervisor does not allow GAN to generate the time series whose dynamics differs a lot from the initial time series dynamics.

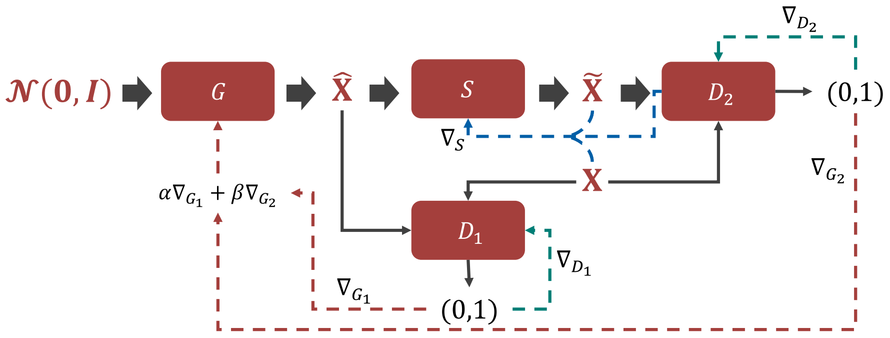


Figure 2. Scheme of data transformation (solid lines) and training process (dashed lines) in the CLSGAN model.

IV. Experimental Study

A. Data Description

For our experiments we use financial time series, in particular, stock prices available via open access (https://www.kaggle.com/datasets/borismarjanovic/price-volume-data-for-all-us-stocks-etfs). The data presented in CSV format with following features: Date, Open, High, Low, Close, Volume, OpenInt. We choose the Close price feature and stocks such as ZEUS, GEN, and FISI for the further experiments. We choose these time series because of their periods of high volatility with fast changes on low volatility periods. Thus, we can properly study the benefits of the proposed method based on volatility clustering using the mentioned data. The lengths of time series are 3201, 2645, 3200, respectively.

Train sets for the GAN-based generative models are formed in the following way: each cluster $C_i, i \in \{1, \dots, n_c\}$ is divided into $N - T + 1$ series of the length T by the sliding window method. In the experiments we take $T = 127$. Then, the batches of size 80 are formed using the obtained time series. In the case of CLFF generative model, the whole cluster is used as a training set.

B. Pipeline Implementation Details

The following section contain some details about the implementation of the pipeline presented in Fig. 1. In our work we use the Pruned Exact Linear Time (PELT) method [32] for change points detection, which allows to detect an unspecified number of change points. We apply it with the following specification: the penalty value is 1 and the minimum length of detected regime is $T + 1 = 128$.

As for the dimension of the regime characteristics vector (2), it is chosen to be 8 and consists of the following components:

- sample mean (calculated on logarithmic normalised via MinMax transformation regime data);
- standard deviation (calculated on initial regime data);
- skewness (calculated on initial regime data);
- kurtosis (calculated on initial regime data);
- minimum (calculated on logarithmic standardise regime data);
- maximum (calculated on logarithmic standardise regime data);
- mean of spectral density squared absolute values (calculated on twice differentiated regime data):

$$S_X = \mathbb{E}_f[|S(f)|^2], \tag{5}$$

where

$$S(f) = \sum_{k=-\infty}^{+\infty} r_X[k]e^{-2\pi kf}, \tag{6}$$

and $r_X[k]$—autocorrelation function of time series X with lag k;

- Kolmogorov-Smirnov test statistic with respect to standard normal distribution (calculated on differentiated regime data).

As for the regimes clustering method, we use the agglomerative clustering algorithm with the Ward distance between clusters and euclidean metric between objects inside clusters.

C. Data Preprocessing

Since clusters are formed from regimes that located at different positions in the original series, the series within a cluster are not continuous. Therefore, in the preprocessing procedure (see Fig. 1), we smooth out the clusters based on the changes between different regimes. Further we will denote this transformation as Δ. Consider a regime

$\mathcal{R}(\tau_i), i \in \{1, \dots, l-1\}$ with the last value $\mathcal{R}_{-1}(\tau_i)$; $\mathcal{R}_0(\tau_{i+1})$ is the first value of the next regime. Denote the difference between them as $\Delta_i = \mathcal{R}_{-1}(\tau_i) - \mathcal{R}_0(\tau_{i+1})$.

The transformation of the regime $\mathcal{R}(\tau_{i+1})$ values is performed according to the following rules:

$$\begin{aligned} &\text{if } \Delta_i \geq 0 \Rightarrow \mathcal{R}(\tau_{i+1}) = \mathcal{R}(\tau_{i+1}) + \Delta, \\ &\text{if } \Delta_i < 0 \Rightarrow \mathcal{R}(\tau_{i+1}) = \mathcal{R}(\tau_{i+1}) - |\Delta|. \end{aligned} \quad (7)$$

After the Δ transformation the regimes are further preprocessed via applying log returns, standardisation and Lambert transformation.

In the case of CLFF generative model the Δ transformation with the opposite signs is applied to the synthetic clusters $\tilde{C}_1, \dots, \tilde{C}_{n_c}$, as well as inverse transformations of log returns, standardisation and Lambert transform within the data *reconstruction* procedure.

In the case of GAN-based models the reconstruction procedure differs from the above described. This is caused by the fact that these models produce more diverse synthetic data and there is a possibility that after the reconstruction the differences between the regimes in the synthetic time series will vary a lot comparing to the initial data. To cope with it, we use four additional transformations described in Algorithm 1 (the difference between regimes in initial time series is denoted by Δ_i^r).

Algorithm 1. Additional Δ transform for GANs

```
Require: $\Delta_i, \Delta_i^r$
1  if $(\Delta_i \geq 0)$ & $(sign(\Delta_i) = sign(\Delta_i^r))$ then
2      if $|\Delta_i| \geq |\Delta_i^r|$ then
3          $\tilde{\mathcal{R}}(\tau_{i+1}) = \tilde{\mathcal{R}}(\tau_{i+1}) + (|\Delta_i| - |\Delta_i^r|)$
4      else if $|\Delta_i| < |\Delta_i^r|$ then
5          $\tilde{\mathcal{R}}(\tau_{i+1}) = \tilde{\mathcal{R}}(\tau_{i+1}) - (|\Delta_i^r| - |\Delta_i|)$
6      end if
7  else if $(\Delta_i \geq 0)$ & $(sign(\Delta_i) \neq sign(\Delta_i^r))$ then
8          $\tilde{\mathcal{R}}(\tau_{i+1}) = \tilde{\mathcal{R}}(\tau_{i+1}) + (|\Delta_i| + |\Delta_i^r|)$
9  else if $(\Delta_i < 0)$ & $(sign(\Delta_i) = sign(\Delta_i^r))$ then
10     if $|\Delta_i| \geq |\Delta_i^r|$ then
11         $\tilde{\mathcal{R}}(\tau_{i+1}) = \tilde{\mathcal{R}}(\tau_{i+1}) - (|\Delta_i| - |\Delta_i^r|)$
12     else if $|\Delta_i| < |\Delta_i^r|$ then
13         $\tilde{\mathcal{R}}(\tau_{i+1}) = \tilde{\mathcal{R}}(\tau_{i+1}) + (|\Delta_i^r| - |\Delta_i|)$
14     end if
15 else if $(\Delta_i < 0)$ & $(sign(\Delta_i) \neq sign(\Delta_i^r))$ then
16         $\tilde{\mathcal{R}}(\tau_{i+1}) = \tilde{\mathcal{R}}(\tau_{i+1}) - (|\Delta_i| + |\Delta_i^r|)$
17 end if
```

D. Evaluation of Synthetic Time Series Quality

We evaluate the models performance using the following criteria (motivated by the financial time series specific properties):

- extreme points coincidence in initial and synthetic time series;
- fading autocorrelation function of the synthetic time series with behaviour similar to the initial one;
- closeness of time series distributions at different scales (in particular, on daily and monthly basis);
- synthetic and initial time series distributions similarity in terms of sample characteristics (**Skewness**, **Kurtosis**), Jensen-Shannon divergence ($\mathbf{D_{JS}}$), sum of spectral density squared absolute values (5) ($\mathbf{S_\chi}$) and two-sample Kolmogorov-Smirnov statistic, that shows the maximum distance between two empirical distribution functions ($\mathbf{KS}^*$);
- synthetic data quality from the point of its usefulness in the initial time series forecasting tasks (below described in details).

The last criteria of synthetic data quality is connected with the time series forecasting task. For this experiment we divided the synthetic and the corresponding real-world time series into train and test sets according to the standard cross validation procedure for time series. Namely, the time series are divided into 31 disjoint parts of the length $L = 60$. The forecasting model (from fbprophet Python library: https://facebook.github.io/prophet/docs/quick_start.html) is trained on the synthetic time series part of length $k \cdot L$, where $k \in \{1, \dots, 30\}$ is the step number and the model is tested on the real-world time series part of length L. On every step we append the train sample by L values.

E. Generative Models Implementation Details

For *CLFF* model we use the following hyperparameters: 800 epochs, 10 flows, learning rate equals to 10^{-4}.

As it was said earlier, for *CLGAN* and *CLSGAN* we use the architecture where generator and discriminator based on TCNs. The number of temporal convolution blocks H inside the TCN depends on the receptive field size (the detailed description of this dependence is given in [11]). In this work we consider the value for receptive field size equal 127 (thus, $T = 127$). There is a disadvantage of using clustering approach in the case of GAN-based models since clusters can have a small number of elements. This can result in insufficient data to train the GANs of the desired quality. If the number of elements in the preprocessed cluster $C_k, k \in \{1, \dots, n_c\}$ is less than $H \cdot T$, then *noisy oversampling* is applied to it, that is we add noise with standard normal distribution to cluster values. The number of noisy series is equal to s. It is chosen as $\lceil \frac{H \cdot T}{|C_k|} \rceil$, where $|C_k|$ is the number of elements in the cluster C_k. After the noisy oversampling we have s time series belonging to the cluster C_k with the same properties as the initial one but with noisy values. Then we stack them in one series $\hat{C}_k$ and feed it to QuantGAN.

TABLE I. CLGAN and CLSGAN Architecture Details

ζ	d_I	d_H	d_O	N_K	N_D	N_P	N_S	dr	lkr
$\zeta^{(1)}$	1	80	80	1	1	0	1	0.1	0.002
$\zeta^{(i)}$	80	80	80	2	2^{i-2}	2^{i-2}	1	0.1	0.002

For the experiments we employ the model with the values of parameters from $\zeta^{(i)}$ for TCN presented in Table I, where $i \in \{2, \dots, 7\}$. Last layer of TCN is 1×1 convolution [11] with $d_I = 80, d_H = 80, d_O = 1, N_K = 1, N_D = 1$. Note that in CLSGAN Supervisor's TCN has $d_H = 60$. As for the generators, they get random noise sampled from multivariate standard normal distribution of the dimension 3.

TABLE II. HYPERPARAMETERS FOR CLGAN AND CLSGAN

Hyperparameters	**CLGAN**	**CLSGAN**
M	23	20
initial learning rate	0.0009	0.0009 (and 0.001 for Supervisor)
exponential learning rate scheduler	gamma=0.96	gamma=0.96
generator training	every 5 iterations	every 5 iterations
supervisor training	–	every 5 iterations
batch size	80	80
sequence length	127	127
number of clusters	2–3	3–4
clip value for discriminator	0.01	0.01

For the training of CLGAN and CLSGAN we use the hyperparameters listed in Table II (also, in experiments we take $\alpha = 0.8$, $\beta = 0.2$ for training the generator in CLSGAN). As it can be seen, we use learning rate scheduler for faster convergence at the start and we train generator (in both models) and supervisor (in CLSGAN) once every 5th iteration of discriminator. Number of epochs is selected dynamically related on the number of elements in cluster. Let $M \in \mathbb{N}$, $M > 2$ be the maximum acceptable value for the number of epochs. We choose index $i_k \in \{1, \ldots, M-1\}$ as the smallest value for which holds $|C_k| \leq \frac{i_k N}{M-1}$. Then the number of epochs for cluster C_k is equal to $i_k + 1$.

V. RESULTS AND DISCUSSION

In this section we show the advantages of the proposed generative models (CLFF, CLGAN, CLSGAN) by comparing the results to the chosen existing baselines (FF [10], QuantGAN [11]).

In Fig. 3, we can see the obtained Q-Q plots of the extremum points in synthetic and the corresponding initial time series (GEN, ZEUS, FISI). These plots are constructed by finding the points of time series local extremum in windows of the length 40, then computing the quantiles with the step 0.01 for the sample of time series extremum points and plotting the obtained quantiles values. Due to the specific behaviour of financial time series, the generative model should be able to produce extremum points that are close to the initial ones. As it can be seen, CLFF model repeats the real time series more precise than others. GAN-based approaches detect the extrema worse than FF-based models (especially QuantGAN, that tends to underestimate the real data in the case of GEN and ZEUS; in the case of FISI this model produces the extremum points that far from the real ones). Contrariwise, CLSGAN tends to overestimate the real data in all three cases, while the CLGAN shows the closest to the initial data performance in terms of extrema.

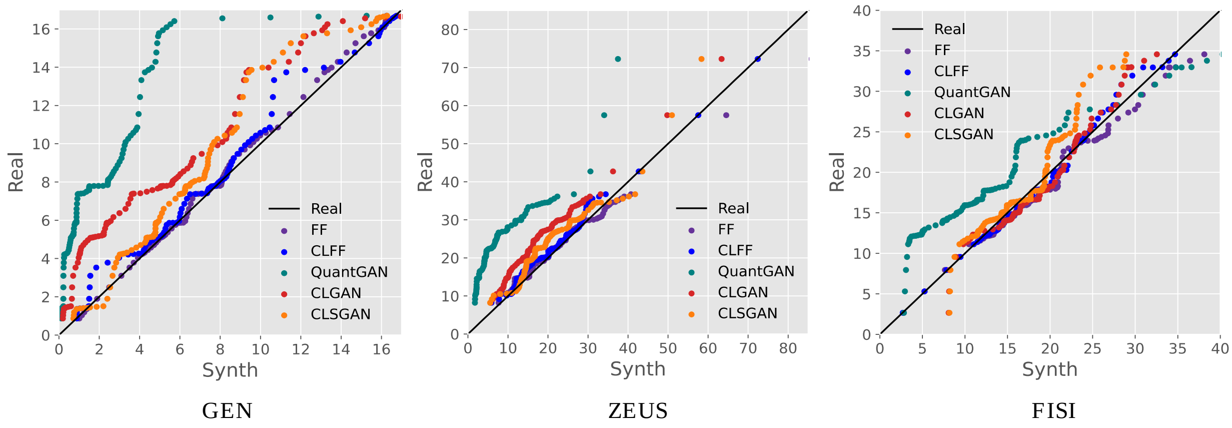


Figure 3. Q-Q plots (quantiles with step 0.01) of local extremum points (computed in windows of the length 40) in synthetic and corresponding initial time series.

In Fig. 4, the autocorrelation function values for each time series and generative model are shown. All models produce fading autocorrelation functions, that is an important characteristic of financial time series. One can note that the dependence of autocorrelation on lags is almost identical in the case of the initial time series and the time series generated by CLFF. In the case of CLGAN and CLSGAN models, the main properties of autocorrelation function behaviour are detected, but are reproduced with different amplitudes.

In Fig. 5, the box plots of obtained MSE values distributions (see Evaluation metrics section for the mechanism of their computation) are shown. The median error value corresponding to CLFF model is closer to the value corresponding to the real-world time series in comparison to Fourier Flows model; also, CLFF's error interquartile range has similar size to the real-world one. In case of ZEUS and FISI (time series with rough dynamics) QuantGAN shows the worst results in terms of MSE. Vice versa, CLGAN and CLSGAN result in smaller values of MSE, thus, these methods are more helpful in approximating the behaviour of the initial time series. Note that training the forecasting model on synthetic generated by CLFF results in reduction of MSE for all tested time series.

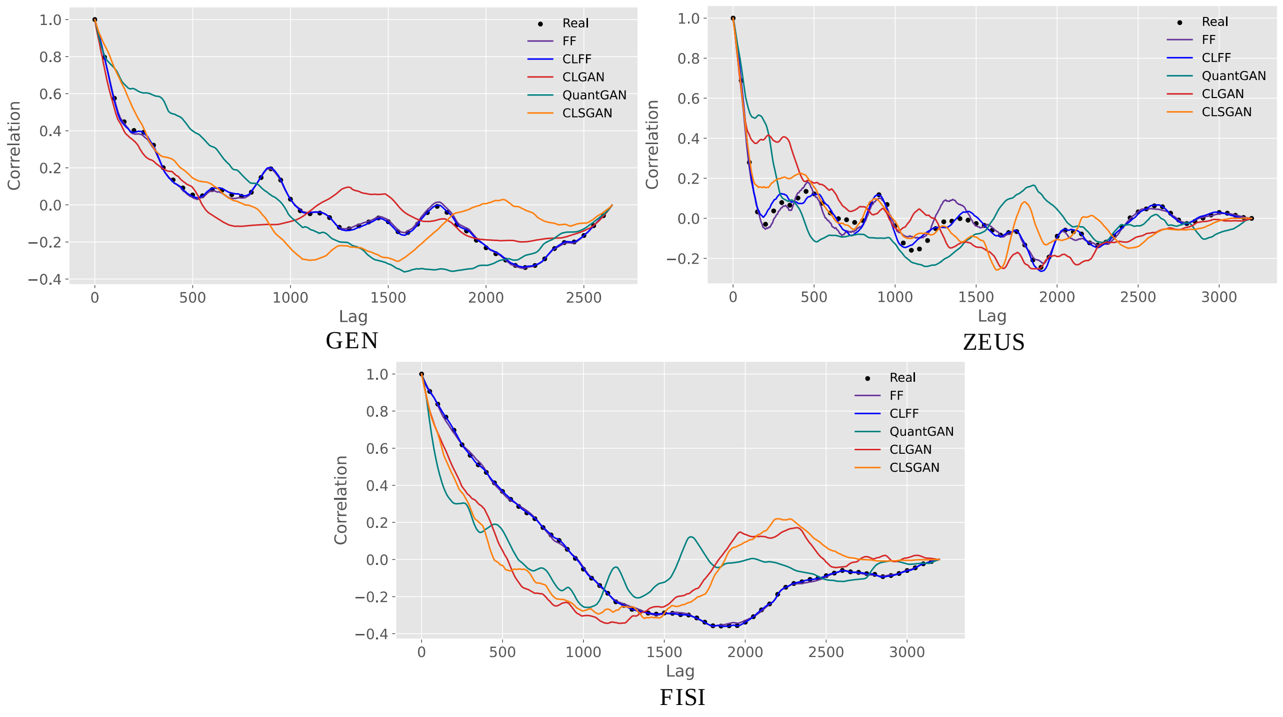


Figure 4. Autocorrelation functions for the initial and synthetic time series.

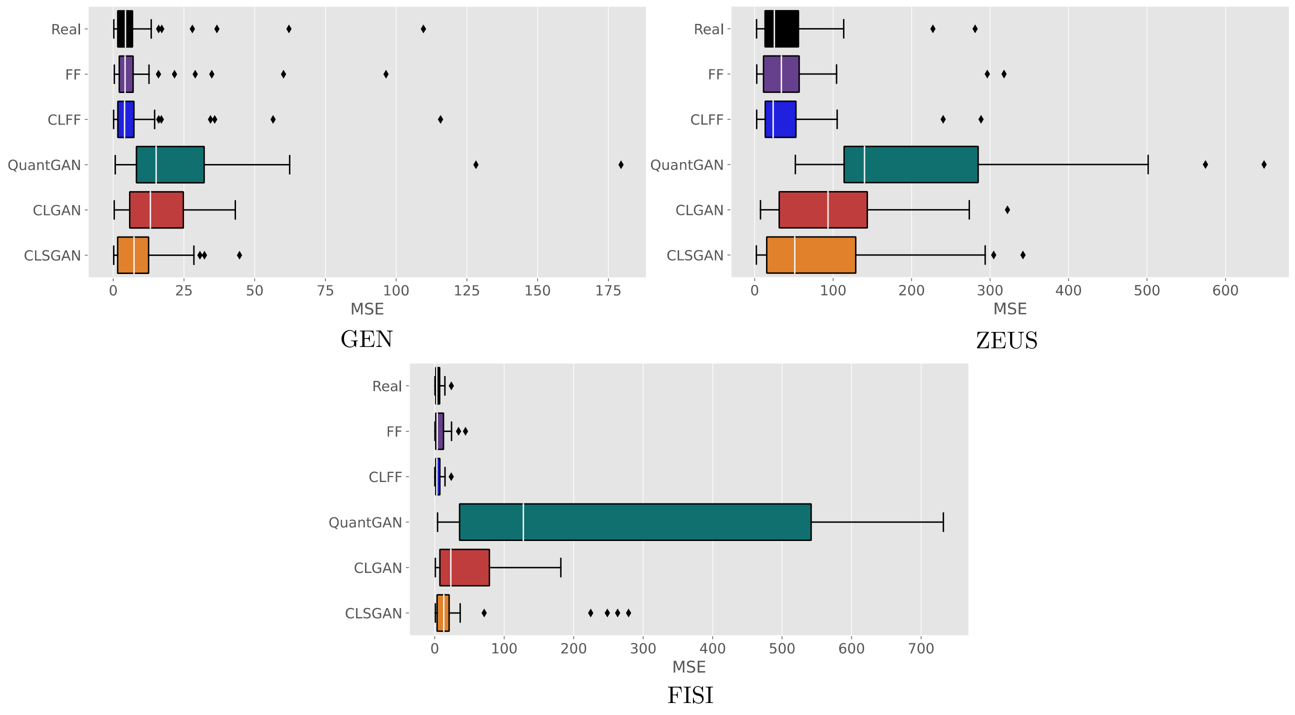
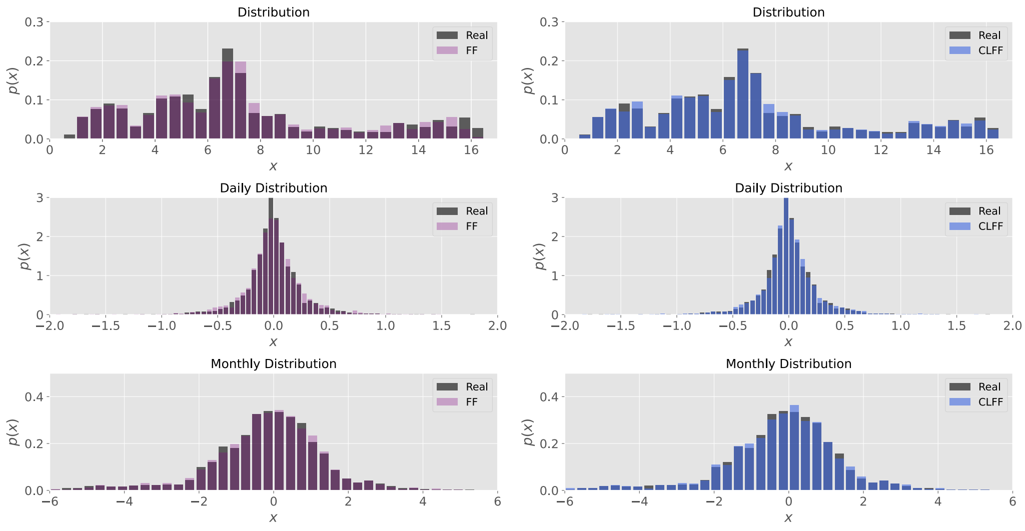
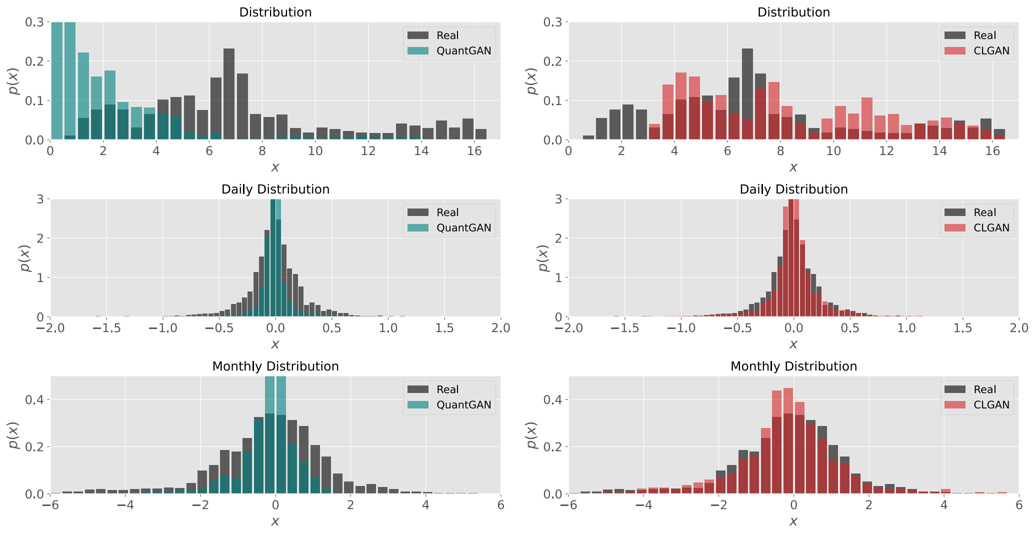


Figure 5. Box plots for MSE values obtained after training the Prophet model on the corresponding synthetic time series and testing it on the real-world time series (repeated 30 times procedure of cross-validation).

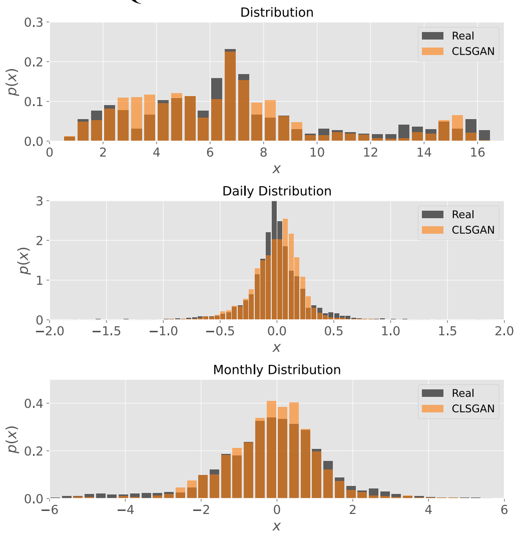
Fig. 6 shows the distributions: the original one and the distribution of revenues in daily (differentiated time series) and monthly (differentiated with time lag of 20 days time series) scales. When dealing with financial time series, it is important to pay attention to daily and monthly distributions of revenues, as the original one can be rather noisy. The CLFF, CLGAN, and CLSGAN models are capable of accurately modelling distributional properties that exist in the real data. However, due to space limitations, we only present the results for GEN in the paper. Models are able to detect the heavy tails and have more precise values in daily and monthly distributions comparing to QuantGAN produced data.



FF and CLFF



QuantGAN and CLGAN



CLSGAN

Figure 6. The original time series distribution, the differentiated time series distribution (daily scale) and the differentiated with lag of 20 days time series distribution (monthly scale without weekends) of the initial and synthetic data on GEN time series.

In Table III, the values of computed distributional characteristics for synthetic and the corresponding real-world time series are presented. To compute every characteristic value we generated 40 synthetic instances for every real-world time series and averaged the obtained values. Regarding the skewness and kurtosis characteristics, we also include the confidence interval bounds in brackets. We can observe absolute dominating of CLFF model in comparison with the baselines as well as with the proposed generative models (the best obtained values for Fourier Flows based models are highlighted with the blue colour). Also, we independently compare the characteristics values of GAN-based approaches (the closest to the real-world time series values are highlighted with the orange colour). Our proposed method with clustering outperforms QuantGAN among all chosen characteristics except the kurtosis for GEN and FISI (note that QuantGAN which outperforms approaches with clustering in FISI has a large confidence interval, thus, the generation process is unstable). CLSGAN in all three data cases approximates the initial frequency S_{χ} closer than QuantGAN due to Supervisor which can help to generate more stable synthetic time series.

TABLE III. SYNTHETIC AND REAL-WORLD TIME SERIES DISTRIBUTION CHARACTERISTICS (COMPUTED BY 40 REALISATIONS OF SYNTHETIC TIME SERIES). FOR SKEWNESS AND KURTOSIS CHARACTERISTICS CONFIDENCE INTERVAL BOUNDS ARE GIVEN IN BRACKETS

Data	Methods	Skewness	Kurtosis	D_{JS}	S_{χ}	KS^{*}
GEN	Real	**0.747**	**−0.135**	**0.000**	**2.004**	**0.000**
	FF	0.595 (±0.05)	−0.280 (±0.08)	0.077	2.350	0.159
	CLFF	**0.733** (±0.01)	**−0.152** (±0.01)	**0.017**	**2.001**	**0.029**
	QuantGAN	0.211 (±0.13)	**−0.393** (±0.18)	0.241	2.287	0.843
	CLGAN	**0.371** (±0.21)	−0.576 (±0.29)	0.162	2.256	**0.456**
	CLSGAN	0.336 (±0.31)	−0.552 (±0.41)	**0.141**	**1.995**	0.501
ZEUS	Real	**1.750**	**6.268**	**0.000**	**1.951**	**0.000**
	FF	0.996 (±0.15)	2.732 (±0.75)	0.048	2.434	0.347
	CLFF	**1.681** (±0.04)	**5.673** (±0.19)	**0.027**	**2.014**	**0.121**
	QuantGAN	0.548 (±0.21)	0.159 (±0.69)	0.171	2.084	0.426
	CLGAN	0.123 (±0.33)	−0.213 (±0.51)	0.192	2.107	0.282
	CLSGAN	**0.590** (±0.31)	**0.701** (±0.76)	**0.169**	**2.052**	**0.211**
FISI	Real	**0.735**	**0.198**	**0.000**	**2.072**	**0.000**
	FF	0.868 (±0.03)	0.601 (±0.09)	0.050	2.614	0.221
	CLFF	**0.739** (±0.01)	**0.238** (±0.02)	**0.020**	**2.102**	**0.039**
	QuantGAN	0.563 (±0.22)	**0.151** (±0.65)	0.180	2.157	0.467
	CLGAN	0.252 (±0.11)	−0.473 (±0.25)	0.185	1.954	0.354
	CLSGAN	**0.778** (±0.19)	−0.401 (±0.29)	**0.157**	**2.015**	**0.335**

Thus, FF and CLFF methods can be used for the accurate generation because they estimate explicit likelihood and show the closest metric values to the real-world time series ones. Additional experiments on varying the hyperparameters (such as number of flows or epochs) in these models demonstrate the results of the low quality as synthetics start to differ from the initial data significantly. Therefore, to generate diverse data, one should use methods that have more freedom in parameters, such as GANs. The proposed approaches show that the regime clustering can help to detect rough dynamic of the financial time series.

VI. CONCLUSION

We have proposed the new method for synthetic financial time series generation, which combines generative models with regime clustering. For the generative models within our method, we used the Fourier Flows and QuantGAN approaches. The proposed method is advisable to use in situations where inferring an explicit probabilistic time series model is difficult or even impossible, such as with financial time series. Due to regimes clustering, our method can deal with multiscale nature of time series and generate a data containing a diversity of patterns presented in the initial data: the distribution characteristics of synthetic data produced by the method are closer to the corresponding values of the initial time series characteristics in comparison with FF and QuantGAN generative models. If one is interested in an accurate and stable generation of the synthetic data, then the CLFF generative model should be used, as it shows the closest to the initial characteristics with narrow confidence intervals. GAN-based generative models provide generation of a more diverse synthetic data.

As for the computation time on CPU (M1 with 8GB RAM), the CLFF (69 s) method works even faster that FF (76.2 s) in the same conditions, due to the training the models separately on clusters data (all measurements are carried out on GEN time series with parameters values described earlier). On the contrary, CLGAN (1218 s) and CLSGAN (1520.4 s) are more computationally expensive in comparison with QuantGAN (687 s), but the experimental study has shown that CLGAN and CLSGAN outperform QuantGAN in terms of generated synthetics quality. We plan to improve the CLGAN and CLSGAN architectures in terms of their generation process time in the future.

The developed method can be applied to the tasks of historical data supplementation for training a ML model of a desired quality or historical data replacement in case of data sharing restrictions. Further elaboration of this research is possible in such directions as constructing a neural network for regimes clustering, that can be able to find more complex relationships between regimes, or method's modification for working with multivariate time series.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Conceptualisation: K.Z., E.S. and A.B.; methodology: K.Z.; data curation: K.Z.; writing—original draft: K.Z. and E.S.; writing—review and editing: E.S. and A.B.; all authors had approved the final version.

Funding

This research is financially supported by the Russian Science Foundation, Agreement 17-71-30029 (https://rscf.ru/en/project/17-71-30029/), with co-financing of Bank Saint Petersburg.

References

[1] O. B. Sezer, M. U. Gudelek, and A. M. Ozbayoglu, "Financial time series forecasting with deep learning: A systematic literature review: 2005–2019," *Applied Soft Computing*, vol. 90, 106181, 2020.

[2] R. Bhowmik and S. Wang, "Stock market volatility and return analysis: A systematic literature review," *Entropy*, vol. 22, no. 5, 522, 2020.

[3] W. Lu, J. Li, J. Wang *et al.*, "A CNN-BiLSTM-AM method for stock price prediction," *Neural Computing and Applications*, vol. 33, pp. 4741–4753, 2021.

[4] A. Parot, K. Michell, and W. D. Kristjanpoller, "Using artificial neural networks to forecast exchange rate, including VAR-VECM residual analysis and prediction linear combination," *Intelligent Systems in Accounting, Finance and Management*, vol. 26, no. 1, pp. 3–15, 2019.

[5] S.A. Assefa, D. Dervovic, M. Mahfouz *et al.*, "Generating synthetic data in finance: Opportunities, challenges and pitfalls," in *Proc. the First ACM International Conference on AI in Finance*, 2020, pp. 1–8.

[6] B. Krollner, B. J. Vanstone, G. R. Finnie *et al.*, "Financial time series forecasting with machine learning techniques: A survey," in *Proc. 18th European Symposium on Artificial Neural Networks (ESANN 2010)*, 2010, pp. 25–30.

[7] F. D. M. Pardo and R. C. Lopez, "Mitigating overfitting on financial datasets with generative adversarial networks," *The Journal of Financial Data Science*, vol. 2, no. 1, pp. 76–85, 2020.

[8] A. Rusnak, "Conditional synthetic financial time series with generative adversarial networks," Master's thesis, Digital Humanities Laboratory, École Polytechnique Fédérale de Lausanne, 2022.

[9] A. Chakraborti, I. M. Toke, M. Patriarca *et al.*, "Econophysics review: I. empirical facts," *Quantitative Finance*, vol. 11, no. 7, pp. 991–1012, 2011.

[10] A. Alaa, A. J. Chan, and M. Schaar, "Generative time-series modeling with fourier flows," in *Proc. International Conference on Learning Representations*, 2020.

[11] M. Wiese, R. Knobloch, R. Korn *et al.*, "Quant gans: Deep generation of financial time series," *Quantitative Finance*, vol. 20, no. 9, pp. 1419–1440, 2020.

[12] A. Rajak and K. Saxena, "Modeling clinical database using time series based temporal mining," *International Journal of Computer Theory and Engineering*, vol. 2, no. 2, pp. 185–188, 2010.

[13] A. Ganatr and Y. P. Kosta, "Spiking back propagation multilayer neural network design for predicting unpredictable stock market prices with time series analysis," *International Journal of Computer Theory and Engineering*, vol. 2, no. 6, pp. 963–971, 2010.

[14] A. Chitra and S. Uma, "An ensemble model of multiple classifiers for time series prediction," *International Journal of Computer Theory and Engineering*, vol. 2, no. 3, pp. 454–458, 2010.

[15] R. Cont, "Empirical properties of asset returns: Stylized facts and statistical issues," *Quantitative Finance*, vol. 1, no. 2, 223, 2001.

[16] P. D'Urso, L. Giovanni, and R. Massari, "Garch-based robust clustering of time series," *Fuzzy Sets and Systems*, vol. 305, pp. 1–28, 2016.

[17] F. Chamroukhi, A. Samé, P. Aknin *et al.*, "Model-based clustering with hidden Markov model regression for time series with regime changes," in *Proc. the 2011 International Joint Conference on Neural Networks*, IEEE, 2011, pp. 2814–2821.

[18] T. W. Liao, "Clustering of time series data: A survey," *Pattern Recognition*, vol. 38, no. 11, pp. 1857–1874, 2005.

[19] A. Cherif, H. Cardot, and R. Bon, "Som time series clustering and prediction with recurrent neural networks," *Neurocomputing*, vol. 74, no. 11, pp. 1936–1944, 2011.

[20] A. Samé, F. Chamroukhi, G. Govaert *et al.*, "Model-based clustering and segmentation of time series with changes in regime," *Advances in Data Analysis and Classification*, vol. 5, no. 4, pp. 301–321, 2011.

[21] J. Wiljes, A. Majda, and I. Horenko, "An adaptive markov chain monte carlo approach to time series clustering of processes with regime transition behavior," *Multiscale Modeling & Simulation*, vol 11, no. 2, pp. 415–441, 2013.

[22] M. Pfenninger, S. Rikli, and D. N. Bigler, "Wasserstein gan: Deep generation applied on financial time series," *Other Financial Economics eJournal*, 2021.

[23] J. Yoon, D. Jarrett, and M. Schaar, "Time-series generative adversarial networks," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[24] K. E. Smith and A. O. Smith, "Conditional gan for timeseries generation," arXiv preprint, arXiv:2006.16477, 2020.

[25] K. E. Smith and A. O. Smith, "A spectral enabled gan for time series data generation," arXiv preprint, arXiv:2103.01904, 2021.

[26] Y. Yacoby, W. Pan, and F. Doshi-Velez, "Failure modes of variational autoencoders and their effects on downstream tasks," arXiv preprint, arXiv:2007.07124, 2021.

[27] X. Li, V. Metsis, H. Wang *et al.*, "TTS-GAN: A transformer-based time-series generative adversarial network," in *Proc. AIME 2022: the 2022 International Conference on Artificial Intelligence in Medicine*, 2022, pp. 133–143.

[28] D. M. Mateos, L. E. Riveaud, and P. W. Lamberti, "Detecting dynamical changes in time series by using the jensen shannon divergence," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 27, no. 8, 083118, 2017.

[29] A. Marani, A. Jamali, and M. Nehdi, "Predicting ultra-high-performance concrete compressive strength using tabular generative adversarial networks," *Materials*, vol. 13, pp. 1–24, 2020.

[30] G. K. Kanji, *100 Statistical Tests*, Sage Publications Ltd., 2006.

[31] J. Jeon, J. Kim, H. Song *et al.*, "GT-GAN: General purpose time series synthesis with generative adversarial networks," in *Proc. NeurIPS 2022, Advances in Neural Information Processing Systems*, 2022.

[32] R. Killick, P. Fearnhead, and I. A. Eckley, "Optimal detection of changepoints with a linear computational cost," *Journal of the American Statistical Association*, vol. 107, no. 500, pp. 1590–1598, 2012.