Generation of Realistic Synthetic Financial Time-Series

MIHAI DOGARIU, LIVIU-DANIEL ŞTEFAN, and BOGDAN ANDREI BOTEANU, Politehnica University of Bucharest, Romania
CLAUDIU LAMBA and BOMI KIM, Big Data & AI Lab, Hana Institute of Technology, Hana TI, South Korea
BOGDAN IONESCU, Politehnica University of Bucharest, Romania

Financial markets have always been a point of interest for automated systems. Due to their complex nature, financial algorithms and fintech frameworks require vast amounts of data to accurately respond to market fluctuations. This data availability is tied to the daily market evolution so it is impossible to accelerate its acquisition. In this paper, we discuss several solutions for augmenting financial datasets via synthesizing realistic time-series with the help of generative models. This problem is complex since financial time series present very specific properties, e.g., fat-tail distribution, cross-correlation between different stocks, specific autocorrelation, cluster volatility etc. In particular, we propose solutions for capturing cross-correlations between different stocks and for transitioning from fixed to variable length time-series without resorting to sequence modeling networks, and adapt various network architectures, e.g., fully connected and convolutional GANs, variational autoencoders, and generative moment matching networks. Finally, we tackle the problem of evaluating the quality of synthetic financial time-series. We introduce qualitative and quantitative metrics, along with a portfolio trend prediction framework which validates our generative models' performance. We carry out experiments on real-world financial data extracted from the US stock market proving the benefits of these techniques.

CCS Concepts: • **Mathematics of computing → Time series analysis**; • **Applied computing → Economics**; • **Computing methodologies → Adversarial learning**; **Unsupervised learning**.

Additional Key Words and Phrases: time-series, synthetic financial data generation, financial data prediction, generative models, fintech

ACM Reference Format:
Mihai Dogariu, Liviu-Daniel Ştefan, Bogdan Andrei Boteanu, Claudiu Lamba, Bomi Kim, and Bogdan Ionescu. 2018. Generation of Realistic Synthetic Financial Time-Series. *ACM Trans. Multimedia Comput. Commun. Appl.* 37, 4, Article 111 (August 2018), 30 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Financial markets are a field which, if acted upon correctly, can bring a major financial gain. This has attracted attention both from individual traders and from researchers. The latter focused on fitting mathematical models to the market's behavior, trying to reach accurate automatic prediction of financial events. Historically, researchers opted for one of two major directions: (i) statistical

This paper is the result of research funded by Hana TI.
Authors' addresses: Mihai Dogariu, mihai.dogariu@upb.ro; Liviu-Daniel Ştefan; Bogdan Andrei Boteanu, Politehnica University of Bucharest, Bucharest, Romania; Claudiu Lamba, lambac@hanafn.com; Bomi Kim, bomik@tepper.cmu.edu, Big Data & AI Lab, Hana Institute of Technology, Hana TI, Seoul, South Korea; Bogdan Ionescu, bogdan.ionescu@upb.ro, Politehnica University of Bucharest, Bucharest, Romania.

1551-6857/2018/8-ART111 $15.00
https://doi.org/10.1145/1122445.1122456

models, e.g., ARCH with its variants [5, 16, 17, 24, 31, 47, 52, 65, 66, 75, 85], and (ii) agent-based models [7–9, 51, 60, 81]. Despite all these efforts, it is still too complex to perfectly capture the underlying properties of financial time series with such mathematical tools [78].

One possible solution to overcoming this drawback is to resort to state-of-the-art deep learning methods [33]. This idea is also strongly backed by the current exponential progress that has been done in other signal modeling domains such as speech recognition [3, 68, 88], speaker recognition [67, 77], medical image classification [1, 23], action recognition [13, 35, 40] or image enhancement [10, 21, 37]. This makes deep neural networks perfect candidates for tackling the financial data modeling problem as well, since financial time-series are time-varying signals with very specific properties. However, these come at the cost of needing significantly larger amounts of data for training robust models than their hand-crafted feature-based learning counterparts. In these domains, such as multimedia and vision, the content available for research is growing at an exponential pace [49, 56]. Moreover, when there is a need for a very specific type of data, researchers can create these resources by themselves by scraping the Internet and using crowd sourcing annotation tools or even fully automatic annotations, e.g., for face and object detection [4] or for concept detection [70].

In contrast, even though financial market indices are public and accessible to everybody, it is a thorough and long-lasting gathering of this data that is missing. Quite often, if there are such datasets available, they are kept behind a pay-wall, discouraging open-source research and reproducibility. Complete and curated datasets can be bought from specialised providers, such as *Bloomberg L.P.*[1]. This restriction forced researchers to resort to building their own lightweight versions of the datasets. There are works that evaluated their research on the Google stocks [84], Shanghai Composite Index, International Business Machine (IBM) Index, Microsoft Corporation (MSFT) Index, Ping An Insurance Company of China (PAICC) Index [86], CSI 300 Index, Nifty 50 Index, Hang Seng Index, Nikkei 225 Index, DJIA Index [58] and S&P 500 Index [58, 86]. The most common approach remains, however, to evaluate on the entire S&P 500 dataset[2] [48, 78, 82]. The financial markets depend, by their nature, on the daily evolution of worldwide events and there is no available mechanism to accelerate the acquisition of such data. Thus, it is of great interest to be able to synthesize new financial data which resembles real stock markets, on the spot, to train the models.

Deep learning methods have achieved great success in realistic data generation and out of all the proposed network models, Generative Adversarial Networks (GANs) [26] offered the most spectacular results. This has been proven in several fields, GANs having the ability to generate realistic faces [42, 71], perform image-to-image translation [36], generate scenes [2], raw audio waveforms [15] or realistic text sequences [87]. Different approaches involving variational embeddings have also proved successful [18, 38, 63]. In this paper we tackle this issue and investigate various solutions to generating realistic synthetic time-series as well as providing effective tools for assessing their quality.

The applications related to financial time-series generation include, but are not limited to: 1) portfolio management — portfolio managers (stockbrokers and hedge funds) extract meaningful data related to the companies in which they should invest on short/medium/long term, e.g., daily values prediction and trend pattern analysis. 2) metadata management — extracting valuable information from a statistically relevant number of samples each day, e.g., understanding data seasonality and predicting sales patterns for various companies; 3) forecasting models — all previously mentioned

[1]https://www.bloomberg.com/professional
[2]https://www.spglobal.com/ratings/en

applications boil down to building accurate forecasting models used for stock market prediction. This is, in essence, the main goal of any financial application.

The remainder of the article is organized as follows. In Section 2 we discuss the existing approaches in the literature, Section 3 provides a description of the financial time series. The proposed solutions are presented in Section 4, Section 5 discusses data processing, and the experimental setup and results are presented in Section 6. Finally, Section 7 provides an in-depth discussion about the entire set of results and discusses future work.

2 RELATED WORKS

Recently, the financial field has also started focusing on the use of generative models for time series generation. There are two main directions that researchers are following.

One approach is to focus on predicting the next sample(s) in a sequence based on the available recent history. This can be modeled as a regression problem, trying to perform the prediction $p(x_{t+1}, ..., x_{t+n}|x_1, ..., x_t)$, where $x_1, ..., x_t$ are the historical values from moment 1 up to moment t and $x_{t+1}, ..., x_{t+n}$ are the future n samples that are being predicted. Most of the time, $n = 1$ is considered to be enough, since we are interested in more accurate predictions in the immediate future, rather than fuzzy predictions in the distant future. This is mainly due to the possibility of unexpected events occurring which may strongly and quickly alter the market's behavior making it unnecessary to attempt predictions far into the future. Several works have been carried out in this direction. For instance, Zhou et al. [89] propose a generic framework employing Long Short-Term Memory (LSTM) and convolutional neural networks for adversarial training to forecast high-frequency stock market. Wiese et al. [82] use temporal convolutional networks and manage to capture longer-ranging dependencies such as the presence of volatility clusters. A more general approach is TimeGAN [84], a framework that learns an embedding space jointly optimized with both supervised and adversarial objectives. The authors test their method on four different types of data: multivariate sinusoidal sequences, stock prices, appliances energy consumption prediction, and events. Kim et al. [44] tackle financial time series prediction with stacked LSTM, attention networks, and weighted attention networks.

Another approach is to generate a fixed number of consecutive samples at a time, in a dataset augmentation paradigm. This procedure requires to extract windows of a given length L_w when creating the training dataset for the generative model. Koshiyama et al. [48] propose a cGAN model for trading strategies calibration and aggregation. Choosing the correct cGAN model is, however, very biased and does not rely on a particular scientific explanation. Similarly, Fu et al. [20] use cGANs for financial time series generation. The conditions can be both categorical and continuous variables containing different kinds of auxiliary information. Another example is the framework in [78], where the authors propose several mixtures of multilayer perceptrons and convolutional neural networks to generate fixed-length time series. However, the authors only evaluate their approach from a qualitative point of view, leaving the results to subjective interpretation.

Synthesized samples are usually validated by a stock market prediction algorithm. There are several papers in the literature covering this field, since it is the most straight-forward approach [19, 22, 43, 53, 76]. Another problem, where it is needed to predict which companies (out of a portfolio) will be the best performing ones is known as stock market trend prediction and has also been covered in the literature [29, 34, 57, 72, 80, 83]. It also represents the main application for our financial time series generation problem.

In this paper, we propose a financial time-series generation framework. Data pre-processing and post-processing play an important role in achieving the best results. Qualitative, quantitative evaluation and prediction of stock market movement of our models validates our approach, as our proposed system achieves better performance when the training dataset is augmented with

synthetically generated samples. Contrary to other existing works, we decided to focus more on exploring the boundaries of this problem and therefore, investigated several solutions rather than proposing a single one. Previous experiments show that different network architectures come each with their own pros and cons, and finding a candidate that performs best in all situations, e.g., capturing all the statistical properties of the real time-series, may be an ill-posed task as one doesn't look for an identical replica of the initial data. Researchers will usually find solutions for different usage situations of the data where variable independent properties of the data should be captured, but very seldom they will focus on meeting all the properties at once.

The contributions of our work, beyond the state of the art are as follows: (i) we provide solutions for data pre- and post- processing that allow increasing the performance of the models, (ii) we extend the range of generative models and evaluation metrics for financial time-series addressed by the current state of the art, (iii) we propose a solution for capturing cross-correlation between different stocks during training, (iv) we propose a solution for transitioning from fixed to variable length time-series without resorting to sequence modeling networks (e.g., LSTM, RNN, GRU etc.), (v) we explore a great wealth of advanced generative architectures and provide several solutions for different scenarios, (vi) we validate the proposed approach by performing stocks trend prediction and prove that synthetic samples help improve prediction accuracy. This paper builds upon our previous work [12] where fixed-length financial time-series were generated with GANs. The main new features of this work consist of investigating more classes of generative models, proposing several additional qualitative metrics, proposing a new batch feeding mechanism to capture cross-correlation of the real stocks and implementing a complex trend prediction setup used to evaluate the goodness of the synthetic samples, as well as providing extensive testing. We compare our results with our previous work [12] and with the ones of Takahashi et al. [78] since the principle they are following is the closest to our work.

Overall, this is an incipient domain which started to gain traction recently, which is visible from the amount of publications and few algorithm resources available. Even though they are scarce, it is worth mentioning that they are very recent, proving that this field receives growing attention. In this context, we consider our work exploratory, providing a deeper understanding of the financial time series generation problem.

3 FINANCIAL TIME-SERIES

Financial data represents information about the state and progress of a company's financial assets. A company's financial time series represents the chronological evolution of several indicators. To create good prediction models it is necessary to train on sufficiently large and diverse datasets. Such a dataset must contain not only a great number of companies (in order to offer good generalization perspectives), but also a large number of samples for each company, i.e., to span a long time period (in order to capture as many different moments in trading history as possible). However, this resource is not always freely available and, when it is, it does not provide enough data for more complex models. It is therefore extremely helpful to have a system that can synthetically generate training data for models to become more profitable.

In our work, we use the daily values of the closing price (C) — the price that the stock reached at the end of the trading day. All entities involved in the trading of stocks use the closing price as a reference point to monitor a company's performance over one day. Moreover, the trend that the closing price is following is more important than its magnitude since stock prices can have various ranges between companies. We take as an example the evolution of Apple stocks from March 20th to 21st 2019, when the closing price rose from \$187.43 to \$194.34, resulting in an absolute difference of \$6.91. Similarly, Amazon closing price increased from \$1897.83 to \$1904.28 from June 26th to 27th 2019, resulting in an absolute difference of \$6.45. These two differences are comparable in

magnitude, but proportional to their corresponding closing prices, they are different by one order of magnitude. Therefore, we focus on ratios of closing prices, rather than absolute closing prices. In particular, we investigate log returns, i.e., the logarithm of ratios of closing prices from consecutive days, given by:

$$r_i = \ln \frac{C_i}{C_{i-1}}, \tag{1}$$

where C_i represents the closing price of day i and r_i the log return closing price of day i. This ratio is useful because it reduces not only the intra-variation of the time series, but also acts as a normalization between different companies' stocks, as displayed in Figure 1. We can see there that two stocks that do not have the same behavior and whose magnitudes of the closing prices belong to different ranges can be successfully encoded under the same range by applying the log return transformation.

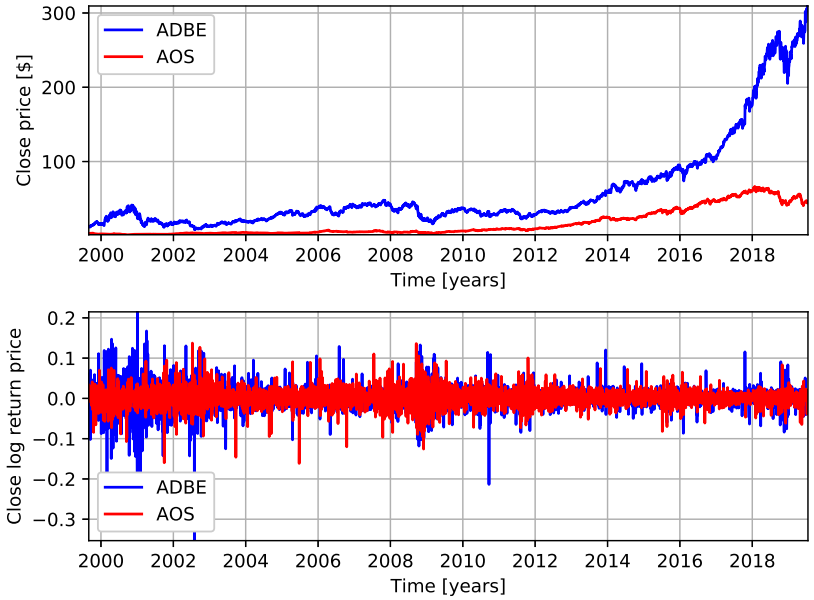


Fig. 1. Log return influence on the intra and inter variations of the closing prices for two companies (A. O. Smith Corp — AOS and Adobe Inc. — ADBE); top — closing prices evolution over 20 years, bottom — log return of the same prices.

Financial time-series are different from other data in the sense that it is necessary to wait for an entire day to extract one new sample (if the granularity is at day level), given that international stock markets update closing prices at the end of the trading day. This strengthens the necessity to have a good data generator for this type of data.

The microstructure of the financial market gives the financial time series several properties and shapes [6, 11, 82]. It is known that these time series are more peaked than normal distribution and exhibit a fat-tailed behavior, meaning that extreme values (both high and low) are more probable than in normal distributions. Also, large changes of prices tend to cluster together, an effect called volatility clustering and can be observed in Figure 1, where large/small changes are followed by large/small changes, respectively. This volatility is negatively correlated with the return process

and is called leverage effect. Lastly, empirical asset returns are uncorrelated for any value of the lag larger than one, but not independent. Generative models face the major challenge of having to cover all these properties of the financial time-series.

4 GENERATIVE MODELS

The first step of our proposed solution consists of generating a fixed-length 1-dimensional array of samples with the help of several generative models. Since there is no standardised architecture agreed upon in the literature that would solve this problem, we performed an in-depth study over a vast number of generative architectures. Our aim was not to come up with a novel model architecture since this domain is still young and there has not been any work conducted to prove one model's superiority over the others. Instead, we adapted to our framework many architectures that have been successfully applied in other multimedia domains (mostly image generation). We investigated 3 major classes of generative models: Generative Adversarial Networks [26] (GANs), Variational Autoencoders (VAEs) [45, 73] and Generative Moment Matching Networks (GMMNs) [14, 54]. For each such class we explore various architectures and training setups in search of the model that manages to best capture the financial market's characteristics. We explored the influence of the number of neurons, layers, activations and types of layers by proposing several models that would cover a range as diverse as possible and were surprised that choosing one model over the others made an important difference. Moreover, as supported by our previous work [12] and the work of Takahashi et al. [78], we concluded that small variations in the model architecture, such as batch normalization, for example, make a crucial difference by turning an otherwise good working model into an unusable generator. The target is to generate a fixed-length vector of log return Close prices. In the following, we describe each of the architectures that were implemented. The length of the synthesized 1D array has been set to 250 for all models, the equivalent of an entire working year in finance. Please note that all architectures are presented in their optimized versions, achieved after in-depth ablation studies.

4.1 Generative Adversarial Networks

GANs have been successfully used in other tasks such as image generation with outstanding results. As pointed out in [25], if both the generator and the discriminator have enough capacity and at each step of the training process the discriminator is allowed to reach its optimum given the generator, then the generator model's probability distribution will converge to that of the training data under the classical GAN optimization function. As mentioned in Section 3, financial time-series follow specific probability distributions, so it was our motivation to fit this exact distribution with the help of GANs. We experimented two types of GANs: fully connected and fully convolutional. The fully connected setup aims to take into consideration the effect that each value from the 250 samples long 1D array has on the outcome. The fully convolutional approach is more oriented towards the effect that short groups of consecutive values captured by the receptive field of the convolution process (which also have the highest correlation since they refer to consecutive business days), have on the outcome. Time series are essentially 1-dimensional arrays that hold a different value for each time step. Therefore, our architectures have been redesigned for the 1D case.

4.1.1 Fully connected GANs - MLP. We implemented four vanilla MLP architectures with different number of layers and neurons per layer, denoted MLP_1 to MLP_4. Each of the four models has several distinct attributes, namely:

- MLP_1: a $20-40-80-160-L_w$ generator network and a $L_w-80-40-20-10-1$ discriminator structure.
- MLP_2: is similar to MLP_1, but without dropout and batch normalization layers;

- MLP_3: has the same layer organization structure as MLP_2, but with different number of neurons on each layer;
- MLP_4: is a shallow version of MLP_3, with different number of neurons in the discriminator's layers.

4.1.2 Fully convolutional GANs - FCGAN. We implemented five architectures for fully convolutional GANs, which were derived from the well known DCGAN [71]. All models follow a $100 - 96 - 48 - 24 - 12 - 1 - fc(L_w)$ structure in the generator and a $L_w - 1 - 12 - 24 - 48 - 96 - 1 - fc(1)$ structure in the discriminator, where each value represents the number of feature maps in the respective layer and L_w represents the length of the synthesized 1D array. The dimension of the normal distributed random noise vector that was used as input for the generator is on the first layer's position, i.e., 100.

The five architectures are as follows:

- $FCGAN_1$: the generator contains 5 layers of 1D transpose convolutions, each of which is followed by a batch normalization layer and a ReLU activation, except for the last layer, which does not connect to batch normalization. The discriminator has a mirrored structure, with convolutional layers replacing the transpose convolutions;
- $FCGAN_2$: is a shallower version of $FCGAN_1$ that requires flattening layers to adapt to the output dimension;
- $FCGAN_3$: is the same as $FCGAN_1$, but without any batch normalization.
- $FCGAN_4$: is the same as $FCGAN_2$, but without any batch normalization.
- *snFCGAN*: spectral normalization GAN. This is the same layer organization as in $FCGAN_1$, with the only difference that we replaced batch normalization layers with spectral normalization layers [64].

4.1.3 Matching statistic moments. Motivated by the fact that we would like to generate samples whose statistic moments match those of the real data, we adopted a weighted loss function between the Maximum Mean Discrepancy (MMD) [27, 28] loss and the classical generator loss function for both fully connected and fully convolutional GANs. Thus, the objective functions to optimize alternatively during one training iteration become:

$$\mathcal{L}_D = \log D(x) + \log(1 - D(G(z))) \tag{2}$$

$$\mathcal{L}_G = (1 - \alpha)\log(1 - D(G(z))) + \alpha\mathcal{L}_{MMD^2}, \tag{3}$$

where $\mathcal{L}_D$ and $\mathcal{L}_G$ are the discriminator and generator losses, respectively, $D(x)$ and $G(x)$ are the discriminator and generator outputs, respectively, x are real financial data, z are the latent noise vectors that are transformed by the generator into synthetic samples, α is a weighting factor and $\mathcal{L}_{MMD^2}$ is the Maximum Mean Discrepancy computed between the generated samples and the real ones, expressed as:

$$\mathcal{L}_{MMD^2} = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{i'=1}^{N} k(x_i, x_{i'}) - \frac{2}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M} k(x_i, y_j) + \frac{1}{M^2}\sum_{j=1}^{M}\sum_{j'=1}^{M} k(y_j, y_{j'}), \tag{4}$$

where x and y represent samples from two different sets $X = \{x_i\}_{i=1}^{N}$ and $Y = \{y_j\}_{j=1}^{M}$, with N and M being the sample set's dimensions. These samples are drawn from two different distributions P_X and P_Y. MMD gives an estimate of the distance between the two distributions. If this loss becomes 0, then $P_X = P_Y$. In our case, we compute the MMD between real and generated samples on each iteration of the generator's training, over one full batch. Therefore, $N = M = n$, where n is the batch size. Then, $k(x, x') = exp(-\frac{1}{2\sigma}|x - x'|^2)$ represents the Gaussian kernel with σ being the

bandwidth parameter. Leveraging this algorithm, we can get an explicit feature map by using a Taylor series expansion with an infinite number of terms which, in theory, covers all orders of statistics. The rationale behind adding the MMD is that we wanted the generator to have two objectives: generating good enough samples that fool the discriminator while at the same time matching the statistical moments of the training distribution. Since it is not clear beforehand which of the two terms has more importance, we added a weighting parameter that was varied (between 0 – original GAN formulation and 1 – generator trained solely on MMD loss) throughout the training procedure.

4.1.4 Wasserstein training. For each of the aforementioned models we applied two different training frameworks. One is the vanilla GAN setup [26] and the other is the Wasserstein GAN setup [2], with gradient penalty. The change to be done here was only in the way the training was performed and not in the model architectures. In the Wasserstein models' case, for each epoch we trained the Discriminator for 5 iterations, then the Generator for 1 iteration.

4.2 Variational Autoencoders

We built two VAE models in close connection to the GAN architectures described above, one based on MLP_2 and the other on $FCGAN_3$. The discriminator's structure was copied in the encoder and the generator's structure was copied in the decoder. Regarding the rationale of these specific architectures, we found these two approaches (one for MLP and one for FCGAN) to offer the best stability in the long-run training so we continued with them only in the extended testing phase that is introduced in the next sections. As also explained in Section 3, there are other properties of the financial time-series which are not easily quantifiable (such as volatility clusters, auto and cross correlation for specific lag values etc.). It was our intuition that these properties may be hidden in a lower, fundamental, dimension of the data which led to the choice of VAEs. Moreover, VAEs have a more direct way of training which can be assessed numerically with the help of the RMSE. The input of the VAE, x, is encoded into the mean and variance vectors. Random noise ϵ is drawn from the Gaussian normal distribution $\mathcal{N}(0, 1)$, multiplied with the variance and the result is added to the mean, forming $z \sim \mathcal{N}(\mu, \sigma^2)$, which is decoded into the output x'. In both cases, the encoder of the VAE would replicate the layer structure of the discriminator from its GAN correspondent and the decoder would follow the generator layout. We chose the bottleneck for both models to be of size 20.

The cost function that we minimize during the VAEs' training is as follows:

$$\mathcal{L}_{VAE} = \alpha \mathcal{L}_{reconstruction} + (1 - \alpha)\mathcal{L}_{regression} \tag{5}$$

$$= \alpha MSE(x, \hat{x}) - (1 - \alpha) D_{KL}(q(z|x)||p_{model}(z)), \tag{6}$$

where α is a weighting factor, MSE is the mean square error, D_{KL} is the Kullback-Leibler divergence, $\hat{x}$ is the reconstruction of the input, $q(z|x)$ models the encoder network and $p_{model}(z)$ the decoder network. Additionally, the weighting factor helps us analyze the impact of each of the two losses. For $\alpha = \frac{1}{2}$ we obtain the original VAE loss formulation. Choosing this loss function was motivated by the fact that for this application it is more useful to capture the overall data distribution rather than reconstruct the input samples, especially since we treated the entire generation process predominantly from a statistical point of view. More so, our intuition was that if we focused more on the reconstruction than on the regression part, we would end up with averaged versions of the input, which would probably help the prediction part due to the smoother nature of the data, but would definitely fail in the subjective metrics described in Section 6.1.

4.3 Generative Moment Matching Networks

The key idea of GMMNs is the use of a statistical hypothesis testing framework, namely the maximum mean discrepancy (MMD). Minimizing this discrepancy is equivalent to replicating the statistical moments. That is, if the samples generated by the model follow a distribution whose moments match those of the training data distribution, then the two distributions (empirical and model) are bound to be similar.

The setup for successfully creating a GMMN involves first training an autoencoder on a given dataset. Next, the encoder is used to transform the input data into the latent code space. The generator is then trained to sample data from the latent code distribution which, in turn, will be transformed by the autoencoder's decoder into new samples. Autoencoders, by themselves, have a discrete latent code distribution, which makes them unusable for generation. GMMNs, however, infer a continuos data distribution over the latent code space. One disasdvantage of this method is that it requires large batch size in order to have the moment estimation average over a statistically significant number of samples. This setup is also known as GMMN+AE. The autoencoder is trained to encode the training samples, X, into a latent vector representation, z. Afterwards, the GMMN is trained to map the noise vector n to the latent vector distribution. The GMMN ensures that the model's latent code distribution and that of the training data are similar by minimizing the MMD. This means that we can generate new samples from the continuous latent code distribution which will be transformed, by the autoencoder's decoder, into new samples, X'.

The use of MMD in the GANs and the GMMN+AE is slightly different. For GANs, we use it as a weighted term in the generator's cost function in order to force it to also focus on matching the statistics of the training data. For GMMN+AE, we use it to train a noise-driven generator that outputs values in the autoencoder's latent code (bottleneck) distribution by matching the central moment statistics of this space, thus adding the generating capabilities to the autoencoder. Similarly to the VAEs, we use the same two encoder-decoder setups, MLP_2 and $FCGAN_3$. The moment matching network is an MLP with the $n_z - 40 - 80 - 120 - 180 - n_{out}$ structure. n_z represents the dimension of the noise vector, set to 100. n_{out} is the dimension of the autoencoder's bottleneck and depends on the encoder-decoder architecture.

5 DATA

Financial data is different from most other types of multimedia data in the sense that it possesses several distinct properties, as discussed in Section 3. Furthermore, each country has its own stock market with specific companies and specific behaviours. This means that we will see differences in how the stock market evolves in different countries. This discrepancy affects financial algorithms and makes it difficult to perform a fair comparison between models trained on different stock markets. However, it has become the norm to train, validate and compare results on the S&P500 dataset. Financial data is publicly available for each listed company, but gathering data from all companies under a single dataset, aligning them from a temporal point of view and pruning them is currently an effort that resides behind a pay-wall.

5.1 Dataset Creation

We train our generative models on the S&P dataset provided by Hana Institute of Technology. This dataset consists of 1,506 companies with daily closing prices records from January 1st 2000 to March 31st 2020. Compared to the commonly used S&P500 dataset, it contains more companies (1,506 vs 500), but our dataset spans a shorter time interval (start date is January 1st 2000 vs 31st March 1964). We address only the most recent 20 years of data because they are closer to current market behaviour than older stocks. We chose the granularity of the data to be at a daily level

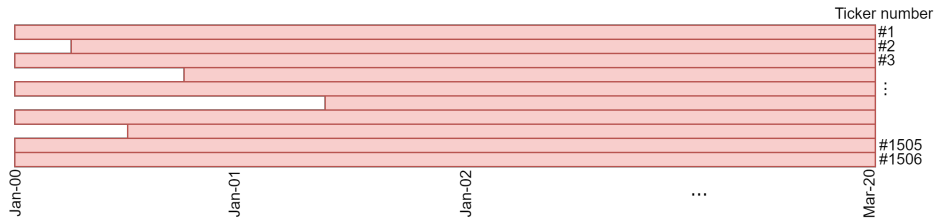


Fig. 2. Visual representation of the S&P dataset layout in time.

after running preliminary experiments that showed that a finer granularity, e.g., hour-based, would bring significant noise to the monitored statistics, whereas a coarser granularity, e.g., week-based, would overlook important variations that took place during the workweek.

In this dataset, the financial time series are available for companies which entered the stock market (were listed) at different times, so we expect them to start at different moments on the time axis. However, all companies were still active on the stock market at the moment when the data was gathered, meaning that these time series last until the same day. In other words, the dataset consists of time series with different starting dates, but the same end-date. We illustrate in Figure 2 how this time series are laid out in time. This introduces a certain bias, since there is no available data related to companies that disappeared from the stock market at previous moments (due to bankruptcy, fusion or being bought by other companies).

Our generative models restrict us from using varying length inputs, therefore we need to feed them fixed size data. The solution that we found for this problem was to split each available time series into segments of a fixed number of samples using a sliding window mechanism. Starting from the earliest position from our dataset we began extracting segments of 250 samples for each company (or ticker, as depicted in Figure 2) in order to prepare the data for processing. This amount is equivalent to one working year's worth of samples and is a reasonable choice since it allows capturing possible seasonality (events happening once per season/year) and it is not long enough for the market to change drastically between the start and end point. When fixing such a window it is possible to encounter tickers that have an incomplete set of samples due to the fact that they appeared on the stock market sometime during the captured window time interval. This problem can be dealt with in two possible ways. One can pad the incomplete segments with 0 until they reach the full extent of the window or remove them completely. We chose to drop these segments completely in order to avoid altering the dataset by adding hard coded values. We formally represent missing values from these incomplete segments as 'NaN'. Consequently, if the starting position of a segment is 'NaN' then the segment can be deemed as incomplete. Thus, we define:

$$W_i = \{[x_{i,j}, x_{i+1,j}, ..., x_{i+L_w-1,j}] \mid x_{i,j} \neq \text{'NaN'}, \forall j \in [1, 1506]\}, \tag{7}$$

to be the set of all segments starting at day i extracted from all companies among the 1,506 that are listed on the stock market at moment i. L_w represents the length of the window, which we set to 250, as previously mentioned. Formally, we consider the start of the dataset (January 1$^{\text{st}}$ 2000) to be the day with index $i = 1$.

We process the rest of the dataset in a sliding window fashion, with a step of 30 samples, equivalent to 6 working weeks, and add them to our training data set. We denote the obtained training dataset as:

$$D = \bigcup_{\substack{i=1 \\ \text{step } 30}} W_i \tag{8}$$

All data was transformed to log returns, as explained in equation 1. The closer we come to the present with the sliding window, the more companies will present full segments over the captured period due to the previously discussed reasons.

5.2 Dataset Preparation

The success of the time series generation process depends on the dataset preparation. Thus, an important contribution of this paper lies in the way we addressed this step, as explained next. Our final dataset, D, contains more than 200k 1-dimensional entries each of length L_w. An important issue that arises here is how to draw samples from this set. In our previous work [12], we considered D to be a homogeneous mixture of windows and randomly sampled batches of data from it . This led to partly realistic synthetic samples, but there was no cross-correlation present between the generated samples. This is mainly due to the fact that the generative models can encounter cross-correlated input (values from the same time frame for different companies) only by accident. The same approach was carried out by [78], [48] and [20] with window lengths of 8,192, 252 and 230 samples, respectively.

Cross-correlation is important in the stock market domain because companies activate in a limited number of sectors. Thus, news that may impact a given sector will directly impact all companies belonging to that industry and indirectly impact connected industries. For example, if there is a global shortage of Silicium, then companies which activate in the extraction industry will be directly affected and suffer a decrease in prices. Then, companies which use this resource for their products will also be negatively impacted (e.g. glass industry, electronic devices industry etc.) and from here on there is a chain reaction up to a certain point. This means that stocks belonging to the aforementioned sectors will behave similarly when confronted with a powerful external factor (political, social, economic etc.).

In order to feed cross-correlated samples to the models and, inherently teach the generator to synthesize cross-correlated data, we suggest the following processing. Instead of scrambling all the extracted windows inside a large dataset, we kept each set of segments W_i in their original form and formed batches out of each such subset . This means that we process each subset as an entire batch. We perform shuffling inside each subset and between subsets but we do not mix entries belonging to different subsets. We are aware that this imposes the batch size of different lengths and it also forces large batch sizes (up to 1,506), but since each window is only 250 samples long, this does not pose any problem (maximum amount of values that are fed at one iteration is 1506×250 values which is less than the equivalent of one HD image). Even if this is only an implementation issue, we discovered that it greatly helps in encoding latent connections between different stocks. It is very helpful to have such a mechanism because usually the entire stock market responds in approximately the same manner to strong external stimuli, e.g., economic crisis, presidential elections, pandemic outbreaks etc.

5.3 Regime Splits

One interesting aspect about the stock market dataset is that it is generally "growing". This means that the values of the closing prices are increasing on the long term. In log return terms (see Equation 1) it means that the positive values outweigh the negative ones. In this context, we need to determine whether the short-term regime of a time series is up-trending or down-trending for each day. We define the regime for day i as:

$$r_i = \begin{cases} \text{up-trending}, & \text{if } \mu_{\times i} - \mu_{\times\_rolling_i} \geq 0 \\ \text{down-trending}, & \text{if } \mu_{\times i} - \mu_{\times\_rolling_i} < 0 \end{cases} \tag{9}$$

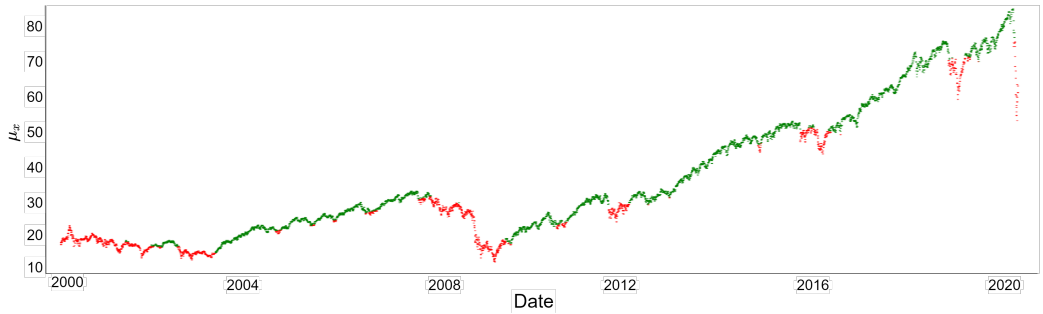


Fig. 3. Dataset division between up-trending (green - 68.07%) and down-trending (red - 31.93%) regimes.

Here, $\mu_{\times_i}$ and $\mu_{\times\_rolling_i}$ are the cross-sectional mean and the cross-sectional rolling mean, respectively, for day i. If $K_i = \{x_{i,j} \mid x_{i,j} \neq \text{'NaN'}, \forall j \in [1, 1, 506]\}$ is the set of all available samples on day i:

$$\mu_{\times_i} = \frac{1}{|K_i|} \sum_{k \in K_i} k \tag{10}$$

$$\mu_{\times\_rolling_i} = \frac{1}{L_{regime}} \sum_{l=i-L_{regime}}^{i} \mu_{\times_i} \tag{11}$$

In other words, we define $\mu_{\times_i}$ as the mean over all closing prices from day i and $\mu_{\times\_rolling_i}$ as the mean of $\mu_{\times_i}$ over the previous L_{regime} days. L_{regime} is constant, representing the length of the window on which the regime is computed. In setting its value, we examined values from the set $\{30, 50, 100, 150, 200, 230, 250\}$. Since there was no significant difference between these setups we decided to keep consistency with previously mentioned rolling windows so we set $L_{regime} = L_w = 250$. Also, for $\mu_{\times\_rolling_i}$ we applied a triangular rolling window. Again, this had no significant impact over applying a regular rolling window. We show in Figure 3 how the market is distributed between up-trending and down-trending.

Performing this split based on regimes results in labelling 68.07% of the days as being up-trending and the rest of 31.93% as down-trending. The difference between the two regimes is significant enough to take into consideration the fact that adding data belonging to only one of the two regimes in the training process might introduce some noise in the generative models' outcome. Therefore, we decided to implement 2 strategies for training each of our models. The first one was to train each model with the complete dataset D. The second one was to compute the regimes for each day on the original dataset, split the dataset according to the two regimes and then apply the windowing mechanism described in Equations 7 and 8 on each of the two regimes. This means that for each architecture setup we trained 3 models: a complete one, trained on the entirety of the dataset without regime splits, an up-trending one, trained only on up-trending days and a down-trending one, trained only on down-trending days. We denote these 3 versions as 'complete', 'up' and 'down', respectively.

5.4 Synthetic Time Series Formation

Once we have the generators trained to output a fixed length time series, we need to combine them such that we obtain arbitrary size time series. In the case of the 'complete' models we simply generate several batches of fixed-length and concatenate them together until reaching the desired length. For the 'mixed' regimes approach, however, we apply the following procedure. We rely on

the fact that the up-trending and down-trending regimes come in bursts of 20 to 120 (statistically determined) consecutive samples. Moreover, we know the final quota of each of the two regimes. Therefore, we sample segments of random length between 20 and 120 samples with a 68% probability of them coming from the 'up' model generator and 32% of them coming from the 'down' model. We concatenate these segments until we reach the desired time series length. Also, adjusting the batch size for the generator is equivalent to setting the number of stocks that we want to generate for a given period, i.e. the financial universe size.

6 EVALUATION PROCEDURE

While the evaluation of classification and retrieval systems is a well known problem and many metrics have been validated through time [61], the evaluation of synthetic generated data is still an open issue and an entire research domain by itself [25, 79], especially in the financial field. Several attempts have been made to assess the goodness of synthesized images, e.g., Inception Score [74] and Fréchet Inception Distance [32] but transforming financial time series to images behaves poorly when evaluated by a neural network trained on natural images. We therefore analyze and propose a series of metrics that were inspired by signal processing problems. Most of these metrics are still experimental regarding how accurate they can describe the performance of financial data synthesis, but they can still give an idea on a hierarchy between different models. We approach the evaluation at three levels: (i) qualitatively, (ii) quantitatively, and via a (iii) predictive accuracy test, presented as follows.

6.1 Qualitative analysis

Lucic et al. [59] argue that it is necessary to report a summary of distribution of results, rather than a single best result achieved. To capture as much information as possible, we randomly select one batch of real data and one of synthesized data during each epoch and we compare several metrics. Graphical examples of how these metrics were assessed are available in the appendix. Since many architectures and variations have to be compared, a complete qualitative analysis is virtually impossible. To overcome this setback, we propose to explore the following properties:

- **Central moments** — one way to determine whether two distributions are alike is to examine their statistical central moments. If the samples generated from two distributions have similar behavior in terms of mean, variance , skew and kurtosis, it is a strong indicator that the two distributions might be similar.
- **Autocorrelation** — a well-known property of financial time series is that they do not posses a linear predictability, meaning that the autocorrelation of the returns is a diminishing function.
- **Heavy-tailed distribution** — financial time series are known to exhibit a heavy-tailed behavior, i.e., their distribution presents a higher probability than a normal distribution of sampling very high and very low values. This translates in having a taller peak than normal distributions and thinner middle.
- **Volatility clustering** — another well-known property is the volatility clustering, meaning that changes (either high or low) come grouped in clusters. In other words, small changes tend to be followed by small changes and high changes tend to be followed by high changes, respectively. This can be assessed by examining the autocorrelation plot of non-linear transformations of returns such as squared returns or absolute returns where no significant autocorrelation can be observed.
- **Cumulative sum** — plotting the cumulative sum of the returns of any company should yield a non-monotonous curve with varying shapes. The cumulative sum for any company, at time

i is defined as $c_i = \sum_{k=0}^{i} r_i$, where r_i is the log return from equation 1. We are interested in the general form of these graphs and not in specific values.

- **Trend ratios** — we define the trend ratio as:

$$tro_i = \frac{trend_i}{n_i}, \tag{12}$$

where $trend_i = \frac{c_i}{c_{i-L_t}} - 1$ is the stock's trend and $n_i = \sum_{k=i-L_t}^{i} \left| \frac{c_k}{c_{k-1}} - 1 \right|$ the noise corresponding to the same stock. Here, L_t is the trend lookback window which we set to 20. Again, we are interested only in the general form of these graphs.

6.2 Quantitative analysis

Quantitative analysis is generally quite difficult to perform on generative models, especially in the context of this relatively new issue of financial data generation. To provide a solution, we adapt metrics that are generally used in the information theory field as well as metrics designed for generative models but in other fields, such as image and speech processing, namely:

- **Kullback-Leibler divergence** [50] — this is a measure of how one probability distribution is different from another. Since it is not a symmetric result, we set the real data distribution as reference and computed the metric for all synthetic data distributions. This measures the capability of our models to assign high probability to most realistic points [25]. For continuous probabilities it is defined as:

$$D_{\mathrm{KL}}(P||Q) = \int_{-\infty}^{\infty} p(x) \log\left(\frac{p(x)}{q(x)}\right) dx, \tag{13}$$

where P and Q are distributions of 2 continuous random variables and p and q denote the probability densities of P and Q, respectively. We are interested in the lowest values that we can attain.

- **Jensen-Shanon divergence** [55] — compared to the Kullback-Leibler divergence, the Jensen-Shanon divergence is symmetric, and it is a measure of similarity between two probability distributions. It is defined as:

$$JSD(P||Q) = \frac{1}{2} D_{\mathrm{KL}}(P||M) + \frac{1}{2} D_{\mathrm{KL}}(Q||M), \tag{14}$$

where $M = \frac{1}{2}(P + Q)$ is the average of the two distributions, P and Q. Again, we are interested in the lowest values.

- **Kolmogorov-Smirnov test statistics** [62] — this is a method of telling if two samples belong to the same distribution or not. We run this on two randomly selected batches, one of real data and one of synthesized data and report the K-S statistics. A low value means that we cannot reject the hypothesis that the two instances are from the same data distribution. With the increase in the number of samples for each instance we expect a more precise statistic.
- **Earth Mover's distance** [41] — the first Wasserstein distance between two 1D distributions. It can be seen as the minimum amount of "work" required to transform distribution u into distribution v, where "work" is measured as the amount of distribution weight that must be moved, multiplied by the distance it has to be moved. It is defined as:

$$l_1(u, v) = \inf_{\pi \in \Gamma(u,v)} \int_{\mathbb{R}\times\mathbb{R}} |x - y| \mathrm{d}\pi(x, y), \tag{15}$$

where $\Gamma(u, v)$ is the set of (probability) distributions on $\mathbb{R} \times \mathbb{R}$ whose marginals are u and v on the first and second factors respectively.

All of these metrics are in accordance with the optimization criterion of the generative models and we found them to be the most adequate for our setup from an information theoretic point of view.

6.3 Predictive accuracy test

To assess the quality of the synthetic data, we propose to predict stock market movement by framing the prediction as a classification problem, discerning whether stocks are up-trending or down-trending at a predefined time point, as the movement information is inherently generated from price. The goal is to provide profitable buy and sell action signals. In this context, this section presents a deep learning approach for financial time series prediction, involving four stages: (i) statistical clustering of stocks based on their normalized returns, (ii) statistical labelling of stocks in up-trending or down-trending, (iii) denoising and reducing the dimensionality of representations using a stacked autoencoder model, and (iv) training predictive models to generate the one-step-ahead output. In the following, we explain each block in detail.

6.3.1 Statistical Clustering for Industry Classification. We have selected an unsupervised learning algorithm in the detriment of the industry classifications such as GICS[3], NAICS[4], Finviz[5], etc., (which are independent of the pricing data) because the synthetic data is unknown to the aforementioned classification schemes, therefore, we are required to use an automatic algorithm to cluster the real and synthetic stock universe. In this context, we used the protocol in [39] by applying the k-means clustering algorithm to cluster the entire universe of stocks according to how close the normalized returns are to the cross-sectional means of the parent clusters. Let N be the number of observations, d the trading days, and R_{is} the daily stock returns, $i = 1, ..., N$, and $s = 1, ..., d$. We cluster the normalized returns R_{is}, where $\hat{R}_{is} = \frac{R_{is}}{\sigma_i u_i}$, $u_i = \frac{\sigma_i}{v}$ and $v = exp(Median(ln(\sigma_i)) - 3MAD(ln(\sigma_i)))$. For all $u_i < 1, u_i \equiv 1$, and Median$(\cdot)$ and MAD$(\cdot)$ are cross-sectional. The standard deviation is computed with a loopback of 100 days, the clusterization is set to 15 clusters, performed with a loopback of 1000 days, and a stride of 30.

6.3.2 Statistical Labeling. To predict stock trend, we formulate the task as a classification problem by classifying stocks into up-trending and down-trending for each period in the training set and for each statistical cluster defined previously. Let P_{is} be the time series of stock prices, where $i = 1, \ldots, N$ labels the stocks, and $s = 1, \ldots, d$ labels the trading days, a time series from day s will be assigned with a corresponding label, denoted L_{is}, according to the value of P_{is} compared to the median of the cluster it belongs to. If P_{is} is greater or equal to the median, then $L_{is} = 1$, otherwise $L_{is} = 0$.

6.3.3 Denoising and dimensionality reduction. Because of the huge number of immediate market movements and trade noise, financial data has a complicated structure of irregularities and roughness. The noise in financial data generally shows strong tailedness, which means that the underlying time series data has a lot of sharp breaks every once in a while. Ignoring these anomalies might lead to erroneous data mining and statistical modeling results. As a result, in order to unveil more meaningful representations, we propose to denoise and reduce the dimensionality of the data using a stacked autoencoder structure (SAE) by layering a succession of single-layer autoencoders (AEs). In this regard, the input daily log returns are mapped into the first hidden vector using the single-layer autoencoder. The reconstruction layer of the first single-layer autoencoder is discarded after training, and the hidden layer is passed as the input layer of the succeeding AEs. By trial and error, the bottleneck's size is fixed at 16, and the depth is set to 4. The denoising is obtained

[3]https://www.msci.com/gics
[4]https://www.census.gov/naics/
[5]https://finviz.com/

by encoding and then decoding the input. If we reconstruct the time series using the bottleneck features, we will reduce the outliers and get a smoothed input estimate to predict the future stock prices. The dimensionality reduction is obtained by encoding the input and using the bottleneck features to predict future stock prices.

6.3.4 Prediction. Three variants of DNN have been implemented and tested, started with universal approximators such as a 4-layer perceptrons (MLP), and moving forward with a 1D ResNet50 variant [30], and finally a bidirectional Long Short Term Memory (BiLSTM) network with the goal of avoiding the long-term dependency problem of time-series data. We run our simulations over 18 years of data (from 2000 up to 2020, with the first 3 years being used solely for the statistical clustering), using two protocols i) train and test on real data, and ii) train on a mix of real and synthetic data and test strictly on real data. Both scenarios involved using a split protocol of past 7 years' worth of data for training and the following year for testing, in a rolling window manner, until we pass through the whole dataset. We use the first protocol to build a baseline approach. The bidirectional LSTM achieved the best results when trained on real data, and tested on real data so we report it as our baseline. This approach will further be compared with the second protocol, where the same model is trained on the mix of real and synthetic datasets to asses whether the synthetic data improves the prediction of up-trending and down-trending stocks or not. To the best of our knowledge, our work is the first to perform trend prediction on such a long time span and for so many companies.

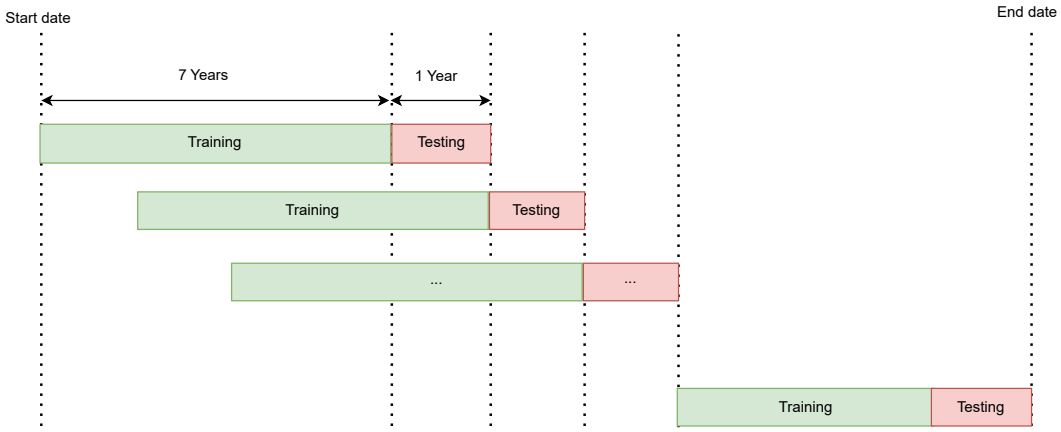


Fig. 4. Continuous dataset arrangement for training and testing during the entire sample period.

7 RESULTS AND DISCUSSION

Each generative model took 6±1 hours to train for 100 epochs on an NVIDIA QUADRO M4000, using the PyTorch[69] framework and we examined the results of a total of approximately 1200 models (excluding the preliminary stages where we established the final architectures and training setups). The pre-processing part took about 0.5 hours for each experiment, but since it was the same for all models, we saved the state of the system after pre-processing was computed and used it from that point on for each model. Regarding the prediction setup, the pre-processing took roughly 20 minutes and the entire training about 2 hours on the same hardware. As there are very few other approaches tailored for this type of time series data generation in the literature, we compare our proposed architectures to the ones in [78] and with our previous work [12]. Due to the vast

number of experiments that we conducted, presenting all the results is physically impossible (we explored the outcome after more than 100 different epochs for more than 650 network models), which led us to adopting the following best-performer selection process. We made snapshots of each network setting whenever it would encounter a new best value for any of the proposed quantitative metrics. Afterwards, we manually inspected all the qualitative metrics of these snapshots. Empiric results show that among the proposed metrics, the Jensen-Shannon divergence is the best indicator to which model has a better overall performance, so we present the results for the snapshots that achieved the best JSD for each model. Also, between vanilla and Wasserstein training setups, preliminary results and previous work [12] suggest that the Wasserstein GAN with gradient penalty was superior in all aspects so we continued solely with this framework for the GAN setups. By examining the results synthesized in Table 1 we draw the following conclusions.

7.1 Training Results

Regarding the models' training procedure there are several aspects that are worth mentioning. The majority of models that produced viable results were trained with a learning rate of 1e-04. We carried out experiments with the following learning rates: 1e-03, 5e-04, 1e-04, 1e-05. Choosing between mixed or complete models does not have a major influence on the trend prediction accuracy. Both techniques offer similar results, thus validating our mixing technique. Only 2 cases ended up with considerable differences: FCGAN_2 (the complete model outperforms the mixed model by 0.29%) and MLP_1 (the mixed model outperforms the complete model by 0.17%). All other model pairs have small differences (<0.1%). In all cases, however, the lower performing models still bring a meaningful improvement to the ground truth dataset and help the prediction models in achieving higher accuracy.

Batch normalization layers do not not hurt the model training anymore. A possible explanation for this is that in the stock market all prices across one day usually follow the same trend. If a major event happens, it is likely to affect the entire market in the same direction for all stocks. Therefore, log returns of different companies do not have significantly different values for a given day. Consequently, applying batch normalization on cross-sectional batches of data is likely to adapt well on individual samples (since they will all have similar mean and variance). This does not happen in the setup proposed in [12] and [78], where batch normalization layers lead to the notorious mode collapse. Another argument that batch normalization is not an issue anymore comes from the fact that models such as MLP_1 and FCGAN_1, which contain batch normalization layers, achieved valid results, as opposed to the works in [12, 78] where any model containing batch normalization layers would fail.

Lastly, balancing the different losses (the α parameter in Equations 3 and 5) for each model one way or another does not have a meaningful impact on the result. Each of our models was trained under 5 different setups depending on the values for α: 0, 0.3, 0.5, 0.7 and 1.

7.2 Metric Results

We can identify several metrics that especially emphasize bad models, which is useful in reducing search time for good candidates. For example, the Jensen-Shannon divergence is a very strong indicator for bad models. Namely, high values of JSD mean that the model does not perform well. Good values, on the other hand, generally indicate good models. However, this is not flawless, since this metric can be confusing especially when the model collapses to a single sample. This sample was already computed as a solver for minimizing the JSD in the generative model's cost function, therefore obtaining low JSD values, but having bad overall characteristics. This is the case for the the sn_FCGAN complete model, which despite achieving a JSD value of 0.0666 (among the smallest ones), generates the same sample, irrespective of the noise vector that is used to drive

Table 1. Qualitative and quantitative results. The * in the Regime column indicates that the respective model resulted in a mode collapse. Qualitative metrics that have been met by the model are indicated by 'yes', whereas missing them by 'no'.

Model	Regime	Mean	Var	Skew	Kurtosis	Autocorr	Heavy tail	Cluster Volatility	Cum Sum	Trend Ratio	JS	KL	K-S	EM
MLP_1	complete	yes	yes	yes	no	yes	no	no	yes	yes	0.4511	237.8	0.2759	0.1254
MLP_1	up	no	no	yes	no	no	no	no	yes	yes	0.575	698.4	0.3546	0.1053
MLP_1	down	no	no	yes	no	yes	no	no	yes	yes	0.4782	394	0.302	0.1103
MLP_1 [12, 78]	N/A	no	no	yes	no	no	no	no	N/A	N/A	0.5757	721.5	0.5757	0.114
MLP_2	complete*	no	no	no	no	no	no	no	no	no	0.4845	276.9	0.3204	0.1196
MLP_2	up*	yes	no	no	no	no	no	no	no	no	0.6263	961.6	0.4019	0.1424
MLP_2	down*	yes	no	no	no	no	no	no	no	no	0.6841	677.4	0.4184	0.4029
MLP_2 [12, 78]	N/A	yes	yes	yes	no	yes	yes	no	N/A	N/A	0.0897	29.81	0.0301	0.0028
MLP_3	complete*	yes	no	no	no	no	no	no	no	no	0.5836	621.4	0.3774	0.2012
MLP_3	up*	yes	no	no	no	no	no	no	no	no	0.7216	1359	0.445	0.3872
MLP_3	down*	no	no	no	no	no	no	no	no	no	0.6552	845.8	0.4115	0.413
MLP_3 [12]	N/A	yes	yes	yes	no	no	yes	no	N/A	N/A	0.1235	128	0.0465	0.0088
WMLP_3 [12]	N/A	yes	yes	yes	yes	yes	yes	no	N/A	N/A	0.1031	39.08	0.0323	0.003
MLP_4	complete	yes	yes	yes	no	yes	yes	no	yes	yes	0.1225	35.69	0.07936	0.00697
MLP_4	up	yes	yes	no	no	yes	yes	no	yes	yes	0.2692	186.4	0.1641	0.01232
MLP_4	down	yes	no	no	no	yes	no	no	yes	yes	0.6354	536.4	0.3845	0.281
MLP_4 [12]	N/A	yes	yes	yes	no	no	no	no	N/A	N/A	0.2095	99.89	0.131	0.0106
FCGAN_1	complete	yes	no	no	no	yes	no	yes	no	yes	0.2947	140.1	0.1818	0.06226
FCGAN_1	up	yes	yes	yes	no	no	no	no	yes	yes	0.4541	395.2	0.2745	0.07718
FCGAN_1	down	yes	yes	no	no	yes	no	no	yes	yes	0.4595	287.7	0.3001	0.2058
FCGAN_1 [12, 78]	N/A	no	no	no	no	no	no	no	N/A	N/A	0.2315	197.5	0.1709	0.0115
WFCGAN_1 [12]	N/A	yes	yes	yes	no	yes	yes	no	N/A	N/A	0.057	18.23	0.0359	0..0018
FCGAN_2	complete	no	yes	yes	no	no	yes	no	yes	yes	0.1592	42.44	0.1385	0.01779
FCGAN_2	up	no	no	no	no	no	no	no	no	no	0.3776	256.9	0.2438	0.06838
FCGAN_2	down	no	no	yes	no	yes	no	no	yes	yes	0.5569	381.8	0.3451	0.2074
FCGAN_2 [12, 78]	N/A	yes	yes	yes	yes	no	yes	no	N/A	N/A	0.0454	13.26	0.0178	0.0011
WFCGAN_2 [12]	N/A	yes	yes	no	no	no	yes	yes	N/A	N/A	0.0825	26.06	0.0387	0.0034
FCGAN_3	complete*	yes	no	no	no	no	yes	no	no	no	0.1198	83.27	0.05527	0.00156
FCGAN_3	up*	yes	no	no	no	no	yes	no	no	no	0.1576	267.4	0.07755	0.00376
FCGAN_3	down*	yes	no	no	no	no	yes	no	no	no	0.2508	92.44	0.1136	0.08977
FCGAN_3 [12]	N/A	no	no	no	no	no	no	no	N/A	N/A	0.5341	673.8	0.4007	0.1026
FCGAN_4	complete	yes	yes	no	no	no	yes	no	yes	yes	0.1324	82.87	0.0525	0.002
FCGAN_4	up	yes	no	yes	yes	no	yes	no	yes	yes	0.1739	92.61	0.1051	0.0056
FCGAN_4	down	yes	yes	no	no	no	yes	no	no	no	0.1294	46.47	0.0902	0.0107
sn_FCGAN	complete*	yes	yes	yes	no	no	yes	no	no	no	0.0666	5.613	0.0383	0.0039
sn_FCGAN	up*	yes	yes	no	no	no	no	no	no	no	0.4882	719.4	0.3216	0.0525
sn_FCGAN	down*	no	no	no	no	no	no	no	no	no	0.2985	140.1	0.2476	0..0475
sn_FCGAN [12]	N/A	no	no	no	no	no	yes	no	N/A	N/A	0.0953	23.53	0.0408	0.0031
FCGANmc [12]	N/A	yes	yes	yes	yes	no	yes	no	N/A	N/A	0.1101	78.16	0.0797	0.0032
GMMN_AE_FC	complete*	no	no	no	no	no	no	no	no	no	0.1211	155.9	0.03031	0.00153
GMMN_AE_FC	up*	no	no	no	no	no	no	no	no	no	0.1926	436.9	0.1069	0.00358
GMMN_AE_FC	down*	no	no	no	no	no	no	no	no	no	0.2783	985.3	0.141	0.00951
GMMN_AE_MLP	complete*	no	no	no	no	no	no	no	no	no	0.3671	1349	0.1937	0.0165
GMMN_AE_MLP	up*	no	no	no	no	no	no	no	no	no	0.2576	1417	0.1237	0.0052
GMMN_AE_MLP	down*	no	no	no	no	no	no	no	no	no	0.2278	822.4	0.0941	0.0072
VAE_FC	complete	yes	yes	yes	yes	yes	yes	no	yes	yes	0.0929	14.26	0.0553	0.00566
VAE_FC	up	yes	yes	yes	yes	yes	yes	no	yes	yes	0.07917	42.5	0.0273	0.00156
VAE_FC	down	yes	yes	yes	yes	yes	yes	no	yes	yes	0.09543	30.97	0.0343	0.00357
VAE_MLP	complete	yes	yes	yes	yes	no	yes	no	yes	yes	0.1058	54.4	0.07054	0.00312
VAE_MLP	up	yes	no	yes	yes	yes	yes	no	yes	yes	0.06591	21.67	0.03615	0.00182
VAE_MLP	down	yes	no	yes	no	no	yes	no	yes	yes	0.09136	19.79	0.04364	0.00331

the generator. The same thing happens to the GMMN_AE family. All of them present lower JSD values than MLP_1, for example, but behave poorly in absolutely all qualitative aspects. Another indicator of bad models is the cumulative sum. Samples that follow the exact same path indicate that the generator model outputs the same values at each inference, therefore it resulted in a mode collapse. However, this indicator must be visually analyzed in order to draw conclusions as there is no straightforward way of numerically assessing this property. Finding a way of quantitatively

characterizing the cumulative sum difference between two or more time series could result in a very good exclusion criterion.

We noticed that models that generate samples whose probability density function (PDF) matches the real samples' probability density function tend to offer good results. This information is integrated in the 'Heavy tail' property. Oppositely, non-overlapping PDFs indicate bad models. This can be assessed by examining the heavy-tail property, which also incorporates the PDF of the generated samples. We also noticed that models that managed to fit the 4^{th} central moment, i.e., kurtosis, behave well on many levels. This can constitute a ranking criterion in future developments and was met only by FCGAN_2 and the VAE family. As it is also pointed out in [12, 78], cluster volatility is difficult to meet. In our experiments, only FCGAN_1 managed to meet this property. Fitting the autocorrelation property, however, improved as compared to our previous work [12]. With our new approach, 12/39 (30.76%) models managed to capture autocorrelation, while previous models [12] had a lower rate: 3/12 (25%). We believe that the proposed dataset preparation technique helped in capturing autocorrelation by feeding cross-correlated samples at each iteration. Based on the number of properties that our models succesfully managed to capture and on the quantitative metrics that were obtained, we compiled a shortlist of best performing models belonging to each major class that was implemented: MLP_4 complete, FCGAN_4 up, sn_FCGAN complete, GMMN_AE_FC complete and VAE_FC complete. Out of these, the clear winner is VAE_FC complete, which outperforms every other model in almost every aspect.

7.3 Prediction Results

Trained strictly on real data, and tested on real data, the best performer was the BiLSTM network achieving a score of 50.04%, becoming the baseline reference for our research. We further augmented the training data set with synthetic data obtained with each of the models presented in Table 2 and tested the baseline algorithm on real data to asses whether the synthetic data add value to our research. We report the mean accuracy obtained over the 10 train-test split pairs presented in Section 6.3 as well as the maximum accuracy over all evaluation periods. The motivation is that the mean accuracy offers a general appreciation of how well such a model is adapted to the entire period, whereas the maximum accuracy finds the best performing training + testing periods. One could argue that after finding the maximum accuracy it would be worth freezing the said model and use it to predict the market performance on the entire validation dataset. The maximum accuracies are on average 0.11% higher than the mean accuracies showing an important performance variation, confirming the volatility nature of stock markets. The baseline value was obtained by training the prediction algorithm with real data only and tested on real data.

One important note regarding prediction accuracy values is that most papers in the literature report accuracies in the 50%-65% range: Feng et al. [19] obtained accuracies of 57.2% and 53.05% on datasets containing 88 and 50 stocks, spanning 2 and 9 years, respectively; Hu et al. [34] reached an accuracy of 47.8% on 2527 stocks, over the course of 3 years; Kinlay[46] tested 1 million different prediction models and obtained an accuracy of 51.5% on 10 stocks, spanning 10 years; Liu [57] obtained an accuracy of 66.93% on 473 stocks, spanning 12 years; finally, Wiese et al. [82] report a prediction accuracy of 58.23% on 88 stocks, over 2 years. We can see a large variety in the results and this is mostly due to the way the experiments were conducted as there is no universal consensus regarding the training and testing dataset, which makes prediction results difficult to compare. Our financial partners clearly expressed that achieving 52% true prediction accuracy for this application is nearly impossible and that would lead to immense financial profits, so we set this as a gold standard. Contrary to the enumerated approaches, we report the average results obtained on 1,506 companies over 20 years. This is a significantly larger dataset than anything reported in the literature. Moreover, we did not make any selection as to what periods to report. This is

Table 2. Augmented dataset evaluation accuracies.

Model	Mean accuracy	Max accuracy
MLP_1_complete	50.12%	50.26%
MLP_1_mixed	50.30%	50.58%
MLP_2_complete	50.41%	50.56%
MLP_2_mixed	50.40%	50.48%
MLP_3_complete	50.38%	50.43%
MLP_3_mixed	50.40%	50.50%
MLP_4_complete	50.34%	50.45%
MLP_4_mixed	50.27%	50.38%
FCGAN_1_complete	50.27%	50.40%
FCGAN_1_mixed	50.35%	50.45%
FCGAN_2_complete	50.40%	50.54%
FCGAN_2_mixed	50.11%	50.27%
FCGAN_3_complete	50.37%	50.43%
FCGAN_3_mixed	50.28%	50.37%
FCGAN_4_complete	50.39%	50.46%
FCGAN_4_mixed	50.32%	50.41%
sn_FCGAN_complete	50.31%	50.48%
sn_FCGAN_mixed	50.39%	50.44%
GMMN_AE_FC_complete	50.40%	50.44%
GMMN_AE_FC_mixed	50.42%	50.52%
GMMN_AE_MLP_complete	50.40%	50.48%
GMMN_AE_MLP_mixed	50.41%	50.57%
VAE_FC_complete	50.31%	50.41%
VAE_FC_mixed	50.29%	50.42%
VAE_MLP_complete	50.18%	50.28%
VAE_MLP_mixed	50.22%	50.32%
Baseline	50.04%	50.08%

important, since the year 2008 introduces a strong disturbance in the algorithms' performance due to the financial crisis, when most patterns were broken and almost all companies suffered important losses. Finally, our trend prediction algorithm is focused more on fairness (following the correct steps such that no information from the future is leaked into the training set) than on achieving the best performance, since this is not our goal, because subtle errors can occur very often and lead to unrealistically high prediction results.

Looking at the previously compiled shortlist of models we can see the following absolute accuracy improvements over the baseline: MLP_4 complete (+0.30%), FCGAN_4 up (+0.28%), sn_FCGAN complete (+0.27%), GMMN_AE_FC complete (+0.36%) and VAE_FC complete (+0.27%). Given that these accuracies are computed as an average over 1506 companies' performances, and that the baseline prediction accuracy is 50.04%, we can conclude that the proposed generative models achieved their intended purpose of boosting the trend prediction accuracy.

7.4 Takeaway Findings

Closely examining each model allowed us to identify several key aspects:

- Under the GAN formulation, Wasserstein training (with gradient penalty) outperforms its vanilla counterparts. As mentioned before, all preliminary experiments indicated this aspect.
- There are several models that converged to generating a single sample, irrespective of the noise vector used to drive the generators. These models are marked with * in Table 1 and among them are the spectral normalization GANs and GMMNs. This indicates that these 2 types of generative networks are not well suited for this problem.
- Variational models reached convergence much faster than all other models. On average, it took them 20 epochs to reach the best state, whereas other models took 59 epochs.
- Variational models offered the best overall results. We ranked the models based on each individual quantitative metric, we averaged these ranks and performed a final ranking based on this average. 4 out of the first 5 ranks were occupied by variational models, which is coherent with our manual analysis.
- Using any of the proposed models for dataset augmentation helps in achieving better prediction accuracy with the proposed prediction framework. Even though the accuracy increase is not spectacular and the the prediction framework is not optimal, it is enough to prove that our proposed solutions achieve their goal. Concretely, a financial time-series regime prediction model achieves better results if the training dataset is augmented with synthetically generated samples.

7.5 Open Challenges

Given the fact that this research field is still at its early stages, we encountered several issues that have not been specifically approached in the literature, nor have they been mentioned by researchers in their prospective future works so far. We have compiled a list of the challenges that remain open up to this point:

- **Optimal architecture.** There is no consensus in the current state of the art as to what network architecture works best for generating financial time series. Researchers tried several models but have failed in finding one type that outperforms all others.
- **Cost function.** Finding the right cost function to optimize the generation process is another key aspect that requires further investigation. Since a concrete evaluation metric is missing, it is difficult to design a proper cost function, not knowing what the end goal of the learning procedure is. This makes the choice of an optimization function a random process.
- **Standardized evaluation.** Most papers in this field report their results on different datasets and under different evaluation setups. Without having a common denominator it is extremely difficult to assess which algorithm performs better. Moreover, it is still debatable what evaluation metric should be used in order to assess the goodness of the synthetic data. This last point is a common problem for generative models, also encountered in other fields, such as computer vision.

8 CONCLUSIONS

In this paper we proposed a complex framework for generating realistic financial time-series. We proposed a new way of extracting batches of data from the training set, adapted to the particularity of financial time-series. We investigated 3 major classes of generative models with various model composition, setups, hyperparameters, training frameworks and data regimes. We examined different qualitative and quantitative metrics and tested the dataset augmentation ability on real data, under a complex prediction scenario.

Based on our results, we strongly believe it is necessary to perform exhaustive tests on a large number of network models in order to find the optimal setup. This study involved a large number of iterations in order to single out the best combinations that can provide a performance boost to fintech algorithms. Our entire progress was validated at each step by experts from our financial partner, offering valuable insights when the nature of the processed data cast a shade of ambiguity on the obtained results.

Our current work suggests that even though good results can be achieved with various generative models, it is far more probable to find a good setup under the variational autoencoder framework. This manages to satisfy both qualitative and quantitative constraints, improves accuracy when used for augmenting the training dataset on the prediction task, converges faster than all other models and it managed to generate realistic samples to the point of being difficult to tell apart from real data by financial experts.

Finally, similar to the image generation field, we stress the need of finding a metric or validation framework that can harness both objective and subjective properties under a single quantifiable value. This should completely characterise the goodness of the generated samples and serve as an optimisation criterion. Our future work will focus on this specific part, since it was among the most difficult obstacles that we encountered during the development stage.

ACKNOWLEDGMENTS

The work of Mihai Dogariu, Liviu-Daniel Ștefan and Bogdan Ionescu was partly funded under project AI4Media "A European Excellence Centre for Media, Society and Democracy", grant #951911, H2020 ICT-48-2020.

REFERENCES

[1] Parnian Afshar, Arash Mohammadi, Konstantinos N Plataniotis, Anastasia Oikonomou, and Habib Benali. 2019. From handcrafted to deep-learning-based cancer radiomics: challenges and opportunities. IEEE Signal Processing Magazine 36, 4 (2019), 132–160.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein generative adversarial networks. In International conference on machine learning. PMLR, 214–223.

[3] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. 2020. wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations. Advances in Neural Information Processing Systems 33 (2020).

[4] Ankan Bansal, Karan Sikka, Gaurav Sharma, Rama Chellappa, and Ajay Divakaran. 2018. Zero-shot object detection. In Proceedings of the European Conference on Computer Vision (ECCV). 384–400.

[5] Tim Bollerslev. 1986. Generalized autoregressive conditional heteroskedasticity. Journal of econometrics 31, 3 (1986), 307–327.

[6] Anirban Chakraborti, Ioane Muni Toke, Marco Patriarca, and Frédéric Abergel. 2011. Econophysics review: I. Empirical facts. Quantitative Finance 11, 7 (2011), 991–1012.

[7] Damien Challet and Matteo Marsili. 2003. Criticality and market efficiency in a simple realistic model of the stock market. Physical Review E 68, 3 (2003), 036132.

[8] Damien Challet and Y-C Zhang. 1997. Emergence of cooperation and organization in an evolutionary game. Physica A: Statistical Mechanics and its Applications 246, 3-4 (1997), 407–418.

[9] Jun-Jie Chen, Bo Zheng, and Lei Tan. 2013. Agent-based model with asymmetric trading and herding for complex financial systems. PloS one 8, 11 (2013), e79531.

[10] Yu-Sheng Chen, Yu-Ching Wang, Man-Hsin Kao, and Yung-Yu Chuang. 2018. Deep photo enhancer: Unpaired learning for image enhancement from photographs with gans. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 6306–6314.

[11] R. Cont. 2001. Empirical properties of asset returns: stylized facts and statistical issues. Quantitative Finance 1, 2 (2001), 223–236. https://doi.org/10.1080/713665670

[12] Mihai Dogariu, Liviu-Daniel Ştefan, Bogdan-Andrei Boteanu, Claudiu Lamba, and Bogdan Ionescu. 2021. Towards Realistic Financial Time Series Generationvia Generative Adversarial Learning. In Proceedings of the 29th European Signal Processing Conference (EUSIPCO).

[13] Haodong Duan, Yue Zhao, Yuanjun Xiong, Wentao Liu, and Dahua Lin. 2020. Omni-Sourced Webly-Supervised Learning for Video Recognition. In Computer Vision – ECCV 2020, Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (Eds.). Springer International Publishing, Cham, 670–688.
[14] Gintare Karolina Dziugaite, Daniel M Roy, and Zoubin Ghahramani. 2015. Training generative neural networks via maximum mean discrepancy optimization. In Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence. 258–267.
[15] Jesse Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts. 2018. GANSynth: Adversarial Neural Audio Synthesis. In International Conference on Learning Representations.
[16] Robert F Engle. 1982. Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. Econometrica: Journal of the econometric society (1982), 987–1007.
[17] Robert F Engle and Victor K Ng. 1993. Measuring and testing the impact of news on volatility. The journal of finance 48, 5 (1993), 1749–1778.
[18] Patrick Esser, Ekaterina Sutter, and Björn Ommer. 2018. A variational u-net for conditional appearance and shape generation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 8857–8866.
[19] Fuli Feng, Huimin Chen, Xiangnan He, Ji Ding, Maosong Sun, and Tat-Seng Chua. 2019. Enhancing Stock Movement Prediction with Adversarial Training. In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19. International Joint Conferences on Artificial Intelligence Organization, 5843–5849. https://doi.org/10.24963/ijcai.2019/810
[20] Rao Fu, Jie Chen, Shutian Zeng, Yiping Zhuang, and Agus Sudjianto. 2020. Time Series Simulation by Conditional Generative Adversarial Net. International Journal of Mechanical and Industrial Engineering 14, 6 (2020), 463–476.
[21] Michaël Gharbi, Jiawen Chen, Jonathan T Barron, Samuel W Hasinoff, and Frédo Durand. 2017. Deep bilateral learning for real-time image enhancement. ACM Transactions on Graphics (TOG) 36, 4 (2017), 1–12.
[22] Pushpendu Ghosh, Ariel Neufeld, and Jajati Keshari Sahoo. 2020. Forecasting directional movements of stock prices for intraday trading using LSTM and random forests. arXiv preprint arXiv:2004.10178 (2020).
[23] Eli Gibson, Wenqi Li, Carole Sudre, Lucas Fidon, Dzhoshkun I Shakir, Guotai Wang, Zach Eaton-Rosen, Robert Gray, Tom Doel, Yipeng Hu, et al. 2018. NiftyNet: a deep-learning platform for medical imaging. Computer methods and programs in biomedicine 158 (2018), 113–122.
[24] Lawrence R Glosten, Ravi Jagannathan, and David E Runkle. 1993. On the relation between the expected value and the volatility of the nominal excess return on stocks. The journal of finance 48, 5 (1993), 1779–1801.
[25] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. Deep Learning. MIT Press. http://www.deeplearningbook.org.
[26] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In Advances in neural information processing systems. 2672–2680.
[27] Arthur Gretton, Karsten Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex Smola. 2006. A kernel method for the two-sample-problem. Advances in neural information processing systems 19 (2006), 513–520.
[28] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. 2012. A kernel two-sample test. The Journal of Machine Learning Research 13, 1 (2012), 723–773.
[29] Yaping Hao and Qiang Gao. 2020. Predicting the trend of stock market index using the hybrid neural network based on multiple time scale feature learning. Applied Sciences 10, 11 (2020), 3961.
[30] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition. 770–778.
[31] Ludger Hentschel. 1995. All in the family nesting symmetric and asymmetric garch models. Journal of Financial Economics 39, 1 (1995), 71–104.
[32] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In Advances in neural information processing systems. 6626–6637.
[33] M Hiransha, E Ab Gopalakrishnan, Vijay Krishna Menon, and KP Soman. 2018. NSE stock market prediction using deep-learning models. Procedia computer science 132 (2018), 1351–1362.
[34] Ziniu Hu, Weiqing Liu, Jiang Bian, Xuanzhe Liu, and Tie-Yan Liu. 2018. Listening to chaotic whispers: A deep learning framework for news-oriented stock trend prediction. In Proceedings of the eleventh ACM international conference on web search and data mining. 261–269.
[35] Noureldien Hussein, Efstratios Gavves, and Arnold W.M. Smeulders. 2019. Timeception for Complex Action Recognition. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).
[36] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. 2017. Image-To-Image Translation With Conditional Adversarial Networks. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).
[37] Yifan Jiang, Xinyu Gong, Ding Liu, Yu Cheng, Chen Fang, Xiaohui Shen, Jianchao Yang, Pan Zhou, and Zhangyang Wang. 2021. Enlightengan: Deep light enhancement without paired supervision. IEEE Transactions on Image Processing 30

(2021), 2340–2349.
[38] Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. 2017. Variational deep embedding: an unsupervised and generative approach to clustering. In Proceedings of the 26th International Joint Conference on Artificial Intelligence. 1965–1972.
[39] Zura Kakushadze and Willie Yu. 2016. Statistical industry classification. Journal of Risk & Control 3, 1 (2016), 17–65.
[40] M Esat Kalfaoglu, Sinan Kalkan, and A Aydin Alatan. 2020. Late temporal modeling in 3d cnn architectures with bert for action recognition. In European Conference on Computer Vision. Springer, 731–747.
[41] Leonid V Kantorovich. 1960. Mathematical methods of organizing and planning production. Management science 6, 4 (1960), 366–422.
[42] Tero Karras, Samuli Laine, and Timo Aila. 2019. A style-based generator architecture for generative adversarial networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 4401–4410.
[43] Raehyun Kim, Chan Ho So, Minbyul Jeong, Sanghoon Lee, Jinkyu Kim, and Jaewoo Kang. 2019. Hats: A hierarchical graph attention network for stock movement prediction. arXiv preprint arXiv:1908.07999 (2019).
[44] Sangyeon Kim and Myungjoo Kang. 2019. Financial series prediction using Attention LSTM. arXiv preprint arXiv:1902.10877 (2019).
[45] Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. In International Conference on Learning Representation (ICLR).
[46] Jonathan Kinlay. 2011. CAN MACHINE LEARNING TECHNIQUES BE USED TO PREDICT MARKET DIRECTION?-THE 1,000,000 MODEL TEST.
[47] Claudia Klüppelberg, Alexander Lindner, and Ross Maller. 2004. A continuous-time GARCH process driven by a Lévy process: stationarity and second-order behaviour. Journal of Applied Probability (2004), 601–622.
[48] Adriano Koshiyama, Nick Firoozye, and Philip Treleaven. 2021. Generative adversarial networks for financial trading strategies fine-tuning and combination. Quantitative Finance 21, 5 (2021), 797–813.
[49] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems. 1097–1105.
[50] Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. The annals of mathematical statistics 22, 1 (1951), 79–86.
[51] Blake LeBaron. 2006. Agent-based computational finance. Handbook of computational economics 2 (2006), 1187–1233.
[52] Dong Li, Xingfa Zhang, Ke Zhu, and Shiqing Ling. 2018. The ZD-GARCH model: A new way to study heteroscedasticity. Journal of Econometrics 202, 1 (2018), 1–17.
[53] Xinyi Li, Yinchuan Li, Hongyang Yang, Liuqing Yang, and Xiao-Yang Liu. 2019. DP-LSTM: Differential privacy-inspired LSTM for stock prediction using financial news. arXiv preprint arXiv:1912.10806 (2019).
[54] Yujia Li, Kevin Swersky, and Rich Zemel. 2015. Generative moment matching networks. In International Conference on Machine Learning. PMLR, 1718–1727.
[55] Jianhua Lin. 1991. Divergence measures based on the Shannon entropy. IEEE Transactions on Information theory 37, 1 (1991), 145–151.
[56] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In European conference on computer vision. Springer, 740–755.
[57] Huicheng Liu. 2018. Leveraging financial news for stock trend prediction with attention-based recurrent neural network. arXiv preprint arXiv:1811.06173 (2018).
[58] Jialin Liu, Fei Chao, Yu-Chen Lin, and Chih-Min Lin. 2019. Stock Prices Prediction using Deep Learning Models. (2019). arXiv:1909.12227
[59] Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. 2018. Are gans created equal? a large-scale study. In Advances in neural information processing systems. 700–709.
[60] Thomas Lux and Michele Marchesi. 1999. Scaling and criticality in a stochastic multi-agent model of a financial market. Nature 397, 6719 (1999), 498–500.
[61] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. Introduction to information retrieval. Cambridge university press.
[62] Frank J Massey Jr. 1951. The Kolmogorov-Smirnov test for goodness of fit. Journal of the American statistical Association 46, 253 (1951), 68–78.
[63] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. 2017. Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks. In International Conference on Machine Learning. PMLR, 2391–2400.
[64] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. 2018. Spectral Normalization for Generative Adversarial Networks. In International Conference on Learning Representations.

[65] Daniel B Nelson and Charles Q Cao. 1992. Inequality constraints in the univariate GARCH model. Journal of Business & Economic Statistics 10, 2 (1992), 229–235.
[66] Philipp Otto, Wolfgang Schmid, and Robert Garthoff. 2018. Generalised spatial and spatiotemporal autoregressive conditional heteroscedasticity. Spatial Statistics 26 (2018), 125–145.
[67] Manuel Pariente, Samuele Cornell, Antoine Deleforge, and Emmanuel Vincent. 2020. Filterbank design for end-to-end speech separation. In ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 6364–6368.
[68] Daniel S Park, Yu Zhang, Ye Jia, Wei Han, Chung-Cheng Chiu, Bo Li, Yonghui Wu, and Quoc V Le. 2020. Improved Noisy Student Training for Automatic Speech Recognition. Proc. Interspeech 2020 (2020), 2817–2821.
[69] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems 32 (2019), 8026–8037.
[70] Obioma Pelka, Christoph Friedrich, Alba García Seco de Herrera, and Henning Müller. 2020. Overview of the ImageCLEFmed 2020 Concept Prediction Task: Medical Image Understanding. In CLEF2020 working notes (CEUR Workshop Proceedings, Vol. 2696).
[71] Alec Radford, Luke Metz, and Soumith Chintala. 2016. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. In 4th International Conference on Learning Representations, ICLR, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1511.06434
[72] Andrea Picasso Ratto, Simone Merello, Luca Oneto, Yukun Ma, Lorenzo Malandri, and Erik Cambria. 2018. Ensemble of technical analysis and machine learning for market trend prediction. In 2018 IEEE symposium series on computational intelligence (ssci). IEEE, 2090–2096.
[73] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In International conference on machine learning. PMLR, 1278–1286.
[74] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved techniques for training GANs. In Proceedings of the 30th International Conference on Neural Information Processing Systems. 2234–2242.
[75] Enrique Sentana. 1995. Quadratic ARCH models. The Review of Economic Studies 62, 4 (1995), 639–661.
[76] Lior Sidi. 2020. Improving S&P stock prediction with time series stock similarity. arXiv preprint arXiv:2002.05784 (2020).
[77] David Snyder, Daniel Garcia-Romero, Gregory Sell, Alan McCree, Daniel Povey, and Sanjeev Khudanpur. 2019. Speaker recognition for multi-speaker conversations using x-vectors. In ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 5796–5800.
[78] Shuntaro Takahashi, Yu Chen, and Kumiko Tanaka-Ishii. 2019. Modeling financial time-series with generative adversarial networks. Physica A: Statistical Mechanics and its Applications 527 (2019), 121261.
[79] L Theis, A van den Oord, and M Bethge. 2016. A note on the evaluation of generative models. In International Conference on Learning Representations (ICLR 2016). 1–10.
[80] Arun Upadhyay, Gautam Bandyopadhyay, and Avijan Dutta. 2012. Forecasting stock performance in indian market using multinomial logistic regression. Journal of Business Studies Quarterly 3, 3 (2012), 16.
[81] Michael P Wellman and Elaine Wah. 2017. Strategic agent-based modeling of financial markets. RSF: The Russell Sage Foundation Journal of the Social Sciences 3, 1 (2017), 104–119.
[82] Magnus Wiese, Robert Knobloch, Ralf Korn, and Peter Kretschmer. 2020. Quant gans: Deep generation of financial time series. Quantitative Finance 20, 9 (2020), 1419–1440.
[83] Yumo Xu and Shay B Cohen. 2018. Stock movement prediction from tweets and historical prices. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 1970–1979.
[84] Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. 2019. Time-series Generative Adversarial Networks. In Advances in Neural Information Processing Systems. 5509–5519.
[85] Jean-Michel Zakoian. 1994. Threshold heteroskedastic models. Journal of Economic Dynamics and control 18, 5 (1994), 931–955.
[86] Kang Zhang, Guoqiang Zhong, Junyu Dong, Shengke Wang, and Yong Wang. 2019. Stock market prediction based on generative adversarial network. Procedia computer science 147 (2019), 400–406.
[87] Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. 2017. Adversarial feature matching for text generation. In Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR. org, 4006–4015.
[88] Yu Zhang, James Qin, Daniel S Park, Wei Han, Chung-Cheng Chiu, Ruoming Pang, Quoc V Le, and Yonghui Wu. 2020. Pushing the Limits of Semi-Supervised Learning for Automatic Speech Recognition. In Neural Information Processing Systems Workshop on Self-Supervised Learning for Speech and Audio Processing Workshop.

[89] Xingyu Zhou, Zhisong Pan, Guyu Hu, Siqi Tang, and Cheng Zhao. 2018. Stock market prediction on high-frequency data using generative adversarial nets. Mathematical Problems in Engineering 2018 (2018).

A QUALITATIVE METRICS ASSESSMENT

For the qualitative metrics evaluation of the synthetically generated samples we resorted to manual assessment of the goodness of the synthesized samples. The procedure was as follows: we saved a snapshot of all the employed metrics at the end of each training epoch for any given model. We then manually went through the logged data and decided whether the qualitative metrics are similar to the ones presented by the real data or not. This has been done by visual inspection of the similarity between the plotted information for both types of data (real and synthetic). We show below several plots for the different metrics that we evaluated. For the sake of completeness we display the metrics for the real samples, and two examples for the synthetic samples: one where the generating model behaves well and one where it fails. All metrics are computed on the log return values.

A.1 Central moments statistics

Synthetic financial time-series should have the first 4 central moments values similarly spread to real samples. We searched for the models that had the 4 central moments in the same value range, but also similarly spread on the graph. This information could not be captured numerically, therefore we resorted to manual assessment. Figure 5 captures such an example.

A.2 Autocorrelation

Figure 6 displays an example of box plot of the autocorrelation function for different time lags. These box plots were obtained over an entire batch for both real and syntehtic samples. Financial time-series should have diminishing values of the autocorrelation starting with lags greater than 1 and the spread should be similar for real and synthetic cases.

A.3 Heavy tail distribution

We plot in Figure 8 the probability distribution function overlap of real and synthetically generated samples. We extract this information from randomly chosen batches of the corresponding data at the end of each epoch. When the two distributions do not overlap, it is a clear indicator that the model does not behave well.

A.4 Cluster volatility

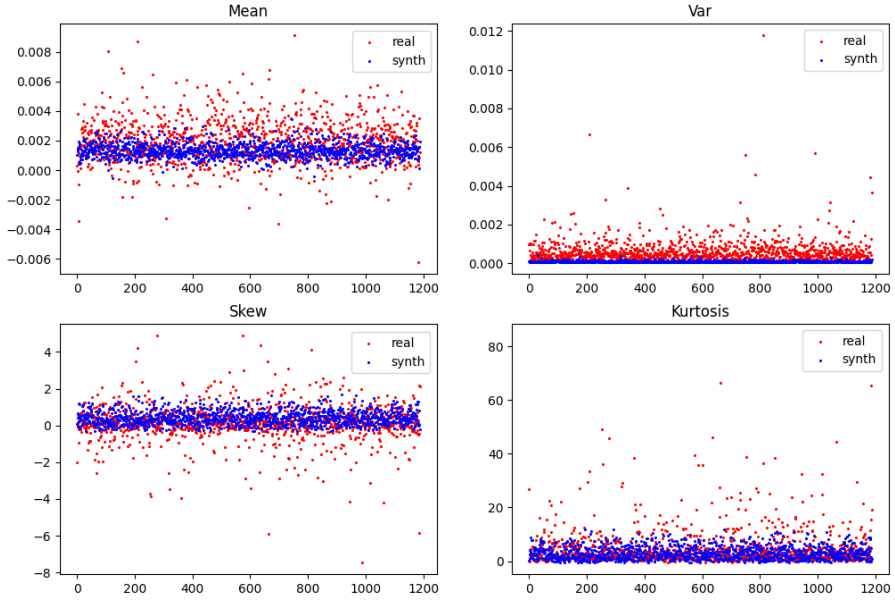
The plots for assessing whether the cluster volatility characteristic is met or not are similar to the ones for the autocorrelation. The reason behind this is that cluster volatility is evaluated based on the autocorrelation of the absolute or squared values. In their case, the plot should display no significant autocorrelation.

A.5 Cumulative sum

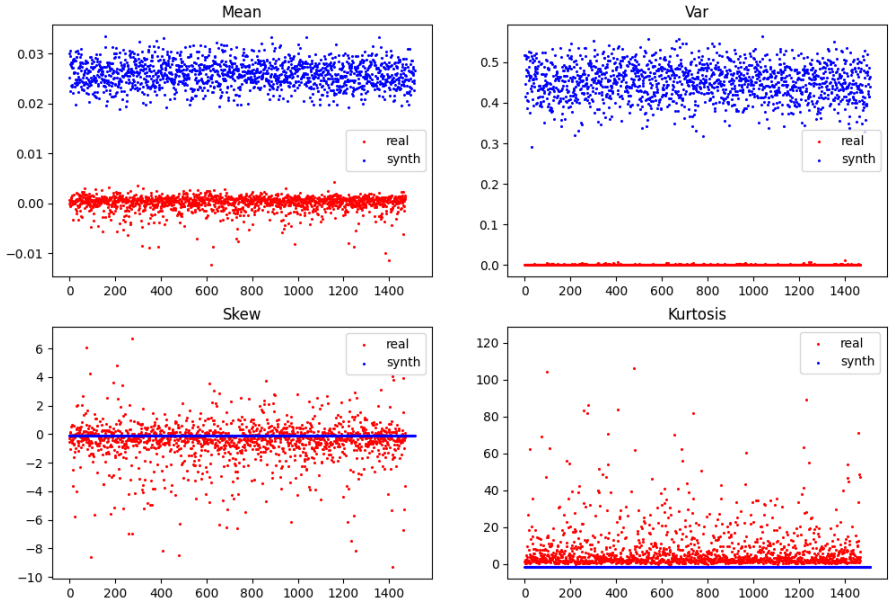
The cumulative sum is plotted under the form of a line plot in Figure 7. Real samples display distinct forms of these lines for each company. Inflexion points usually occur at different times for different companies and only companies that have direct links (either they are owned by the same mother company or have very similar activity) can behave similarly but, even then, only for short periods of time (5-10 consecutive samples).

A.6 Trend ratio

These are also displayed under the form of line plots and behave similarly to the cumulative sum. The same explanation that applies for the cumulative sum graph is valid here as well.

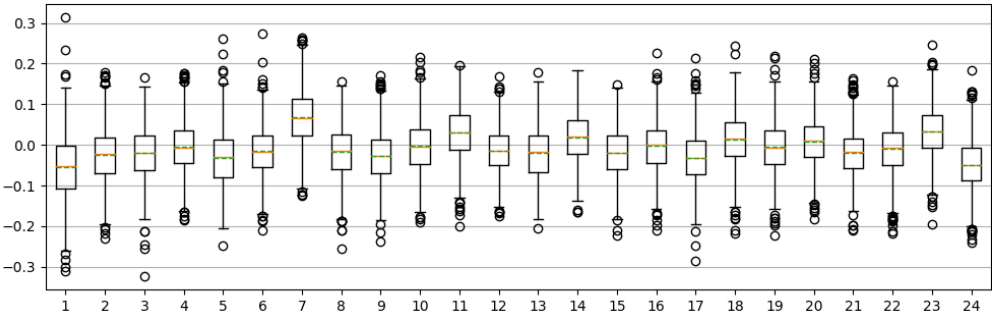


(a) Synthetic samples were obtained with a good generation model.

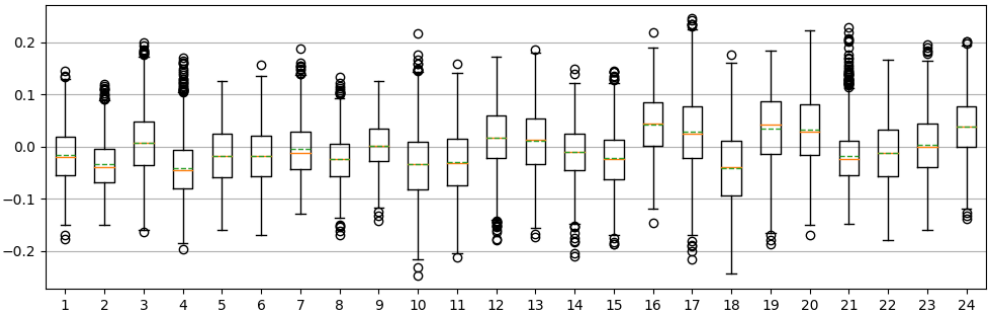


(b) Synthetic samples were obtained with a bad generation model.

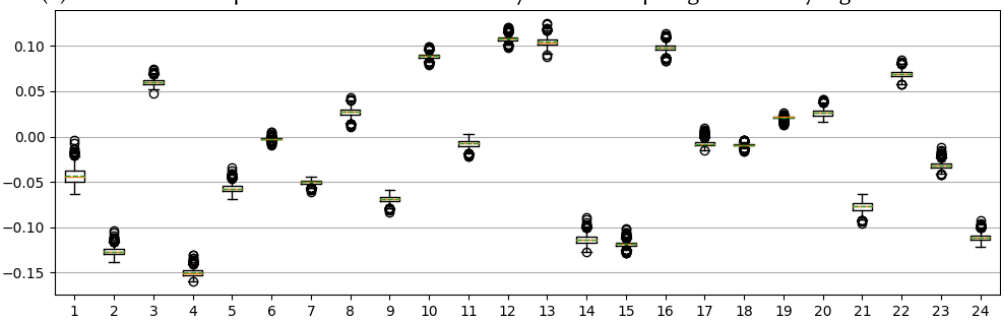
Fig. 5. Real (red) vs synthetic (blue) samples central moments distribution.



(a) Autocorrelation plot obtained on a batch of real samples.

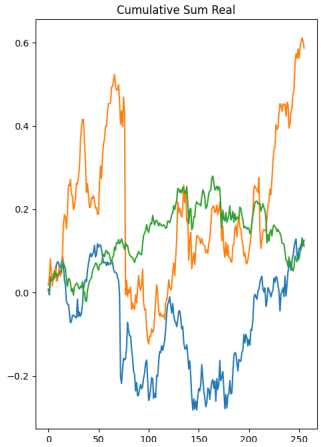


(b) Autocorrelation plot obtained on a batch of synthetic samples generated by a good model.

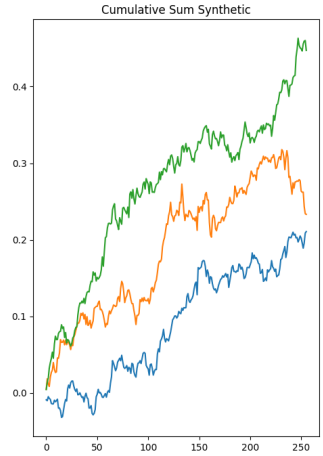


(c) Autocorrelation plot obtained on a batch of synthetic samples generated by a bad model.

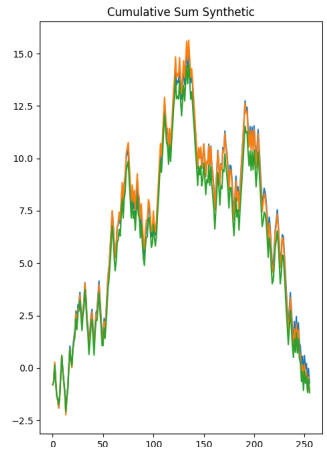
Fig. 6. Autocorrelation box plots computed on an entire batch of samples, for different lags.



(a) Cumulative sum plot obtained on a batch of real samples.

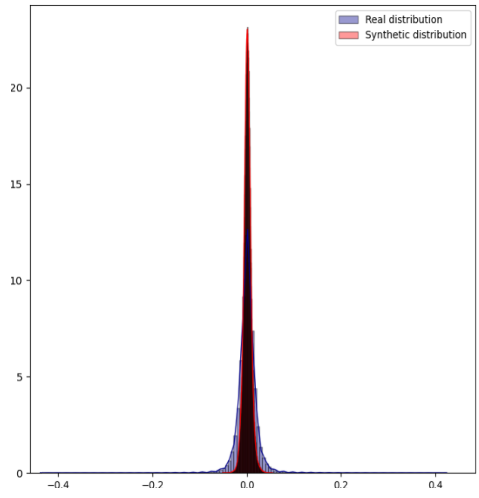


(b) Cumulative sum plot obtained on a batch of synthetic samples generated by a good model.

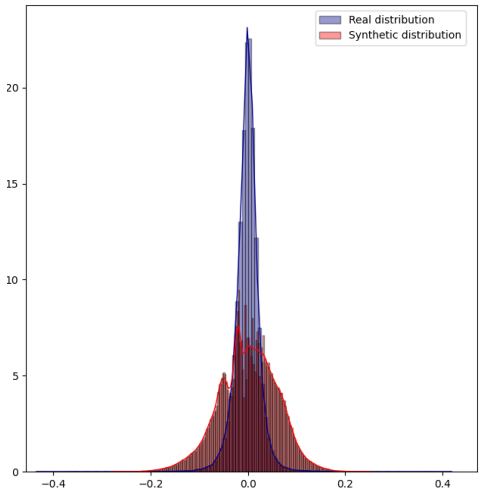


(c) Cumulative sum plot obtained on a batch of synthetic samples generated by a bad model.

Fig. 7. Cumulative sum plots computed on an entire batch of samples.



(a) Probability density functions of real samples and synthetic samples generated by a good model.



(b) Probability density functions of real samples and synthetic samples generated by a bad model.

Fig. 8. Probability density function plots computed on an entire batch of samples.