Available online at www.sciencedirect.com
ScienceDirect
Physica A
Physica A: Statistical Mechanics and its Applications 00 (2019) 1–14

Modeling financial time-series with generative adversarial networks

Shuntaro Takahashi[a], Yu Chen[a], Kumiko Tanaka-Ishii[b]

[a]*Graduate School of Frontier Sciences, The University of Tokyo, Japan*
[b]*Research Center for Advanced Science and Technology, The University of Tokyo, Japan*

Abstract

Financial time-series modeling is a challenging problem as it retains various complex statistical properties and the mechanism behind the process is unrevealed to a large extent. In this paper, a deep neural networks based approach, generative adversarial networks (GANs) for financial time-series modeling is presented. GANs learn the properties of data and generate realistic data in a data-driven manner. The GAN model produces a time-series that recovers the statistical properties of financial time-series such as the linear unpredictability, the heavy-tailed price return distribution, volatility clustering, leverage effects, the coarse-fine volatility correlation, and the gain/loss asymmetry.

Keywords: financial market, stylized facts, deep learning, generative adversarial networks

1. Introduction

Building a model of natural and social phenomenon is a fundamental aspect of science. A good model for a certain kind of natural or social phenomenon can reproduce the properties of its observation, and therefore, allows one to make reasonable analysis and prediction of the phenomenon. The construction of such a model is, however, a hard and sometimes challenging task for so-called complex phenomena. Complex phenomena retain unique statistical properties such as the heavy-tailed distribution, the long-range dependence, the asymmetry structure etc. The description of complex phenomena is often non-trivial and the explanation to the properties tends to be inadequate. The dynamics of financial market is a typical example of complex phenomena, from which various empirically universal properties (stylized facts) are observed, but the mechanism behind them is unrevealed to a large extent.

In the studies of financial time-series, there are two major approaches, namely stochastic processes [1, 2] and agent-based models [3, 4] for modeling the financial time-series. Stochastic processes such as the ARCH and the GARCH describe a financial time-series as a series of random variable with temporally dependent parameters. However, it is difficult to recover all the major stylized facts with such explicit mathematical formulations [5]. In contrast, agent-based models describe agents' behavior and interaction with other agents and the environment based on reasonable assumptions. The interaction leads to a complex collective behavior and the emergence of stylized facts. However, it is also difficult to design such agents' behavior and to calibrate a large number of model parameters based on real data.

Considering the difficulties of these modeling approaches, it is desirable to develop an alternative approach that has a high reproducibility of stylized facts and can be built without a number of assumptions. Deep learning, especially deep generative models may provide such a solution. Among deep generative models, generative adversarial networks (GANs) have shown spectacular ability in the generation of data including realistic image [6], audio [7], and natural language text [8, 9]. The applications of GANs have expanded to a wider range of fields such as physics [10, 11], complex networks [12], music [13, 14], medical time-series [15], DNA sequences [16, 17] and finance [18, 19, 20, 21, 22, 23, 24].

In this paper, a deep neural network based financial time-series model, FIN-GAN is proposed. FIN-GAN learns the properties of financial time-series in a data-driven manner. FIN-GAN can capture the temporal structures of financial time-series so as to generate the major stylized facts of price returns, including the linear unpredictability, the fat-tailed distribution, volatility clustering, the leverage effects, the coarse-fine volatility correlation, and the gain/loss asymmetry. While the application of neural networks to financial time-series is topical, it is mainly being tested to predict the price return and the volatility Kaastra and Boyd [25], Saad et al. [26]. We present an alternative direction to apply modern deep neural networks to the modeling problem in finance field.

The rest of paper is organized as follows. In Section 2, major stylized facts of price return are reviewed, which are the criteria for the evaluation of generated time-series. In Section 3, the construction of FIN-GAN is detailed. The algorithm, the network architectures and hyper-parameters are specified. The analysis on FIN-GAN and the generated time-series is carried out in Section 4. Conclusion and future direction are stated in Section 5.

2. Statistical properties of financial time-series

In this section, stylized facts of financial time-series [27, 28] are reviewed, which will be used to evaluate the time-series generated by FIN-GAN. Note that such an approach is also used in the evaluation of agent-based models for stock markets [4, 29]. In this section and the later experiment, the daily record of S&P500 individual firms over 50 years (from January 2nd, 1962 to November 22nd, 2016) is used.

Let p_t be the price of an asset at t, the log return of price is calculated as

$$r_t = \log p_{t+1} - \log p_t. \tag{1}$$

Note that price return $[\frac{p(t+1)-p(t)}{p(t)}]$ can be approximated by Eq.1 if the price change is small enough.

2.1. Linear unpredictability

The first fundamental property of financial time-series is its linear unpredictability. This property is quantified by the diminishing auto-correlation function of price return

$$\frac{E[(r_t - \mu)(r_{t+k} - \mu)]}{\sigma^2} = Corr(r_t, r_{t+k}) \approx 0, \text{for } k \geq 1 \tag{2}$$

where μ and σ are the mean and the standard deviation of the price return. Fig. 1(a) shows the decay of the auto-correlation function of the price return in daily scale. The absence of linear correlation in the price return in daily scale implies that the financial markets are efficient to a certain extent [28].

2.2. Fat-tailed distribution

The probability distribution function of the price returns $P(r)$ has been extensively studied in different financial markets over different time scales [28, 30]. It is empirically known that the probability distribution $P(r)$ consistently has a power-law decay in the tails

$$P(r) \propto r^{-\alpha}. \tag{3}$$

The exponent typically ranges $3 \leq \alpha \leq 5$ and the exponent $\alpha = 4.37$ is obtained for the positive tail of S&P500 data of $1,638,872$ data points in Fig. 1(b). [1]

2.3. Volatility clustering

While the auto-correlation of the price return is absent, there is still an important temporal structure in the financial time-series, namely volatility clustering. Qualitatively speaking, volatility clustering refers to the fact that the large/small price fluctuations tend to cluster together temporally. Quantitatively, volatility clustering is characterized with the power-law decay of the auto-correlation function of the absolute price returns.

$$Corr(|r_t|, |r_{t+k}|) \propto k^{-\beta} \tag{4}$$

The exponent β ranges between 0.1 and 0.5 in S&P 500 data. Fig. 1(c) shows the averaged auto-correlation function of the absolute price return in a log-log scale. It has a slow and power decay up to $k \approx 10^2$ scale. The power-law decay of the auto-correlation function indicates the presence of the long-range temporal dependence in financial markets [29].

[1] The Python package *powerlaw* [31] based on Clauset et al. [32] is used to obtain the power-law exponent.

2.4. Leverage effect

The leverage effects refer to the tendency that past price return has the negative correlation with future volatility [33]. According to Bouchaud et al. [33] and Qiu et al. [34], this statistical property is quantified by the following lead-lag correlation function

$$L(k) = \frac{E[r(t)|r(t+k)|^2 - r(t)|r(t)|^2]}{E[|r(t)|^2]^2}. \tag{5}$$

In contrast to the statistical properties above, this property is market dependent [34]. While the negative correlation (leverage effect) is observed in German DAX, the positive correlation (so-called anti-leverage effect) is detected in Chinese markets. In the case of S&P500, as shown in Fig. 1(d) the negative correlation is found in the averaged market dynamics [35]. Although the lead-lag correlation function $L(k)$ has a relatively large fluctuation, Bouchaud et al. [33] stated that $L(k)$ has a negative value for $1 \leq k \leq 10$ and it follows the exponential decay.

2.5. Coarse-fine volatility correlation

The coarse-fine volatility correlation is a multi-time-scale analysis of volatility. Müller et al. [36] defined the coarse volatility v_c^τ and the fine volatility v_f^τ as

$$\begin{aligned} v_c^\tau(t) &= \left| \sum_{i=1}^{\tau} r_{t-i} \right| \\ v_f^\tau(t) &= \sum_{i=1}^{\tau} |r_{t-i}|. \end{aligned}$$

While the coarse volatility is the absolute value of the price movement in τ days, the fine volatility is the sum of the absolute price return in τ days. In this paper, $\tau = 5$ is chosen as it stands for day-week time-scales. Müller et al. [36] analyzed the temporal interactions of the two different time-scales with the lead-lag correlation function

$$\rho_{cf}^\tau(k) = Corr(v_c^\tau(t+k), v_f^\tau(t)). \tag{6}$$

Müller et al. [36], Rydberg [37], and Gavrishchaka and Ganguli [38] reported that there exists the negative asymmetry of the lead-lag correlation quantified by the difference

$$\Delta\rho_{cf}^\tau(k) = \rho_{cf}^\tau(k) - \rho_{cf}^\tau(-k). \tag{7}$$

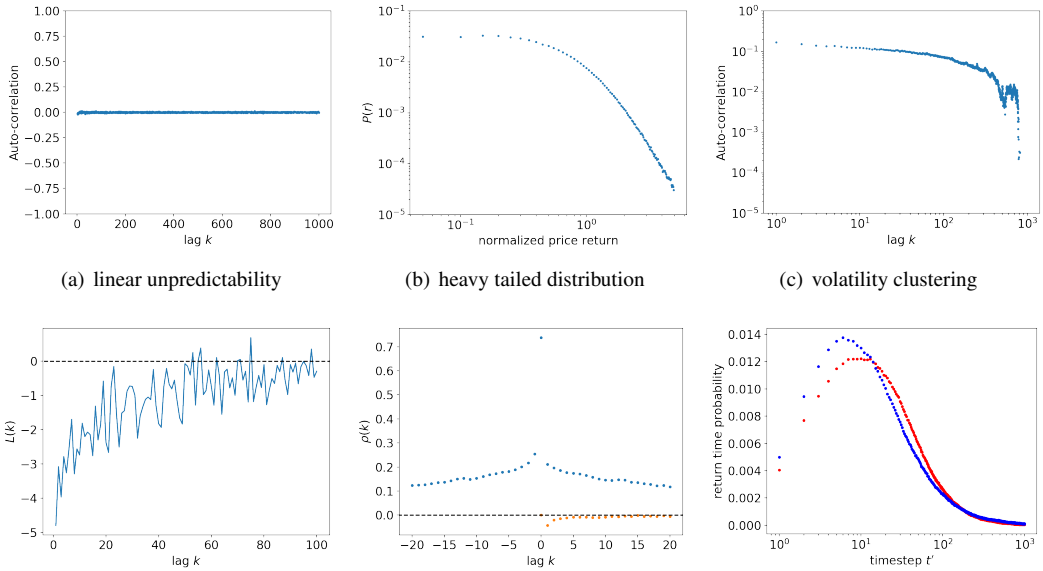
The negativity of $\Delta\rho_{cf}^\tau(k)$ indicates that fine volatility has the power of predicting coarse volatility. Fig. 1(e) shows the coarse-fine volatility correlation ρ_{cf}^τ in blue points and the lead-lag correlation asymmetry $\Delta\rho_{cf}^\tau$ in orange points. The asymmetry is present at $k = 1$ in S&P500 stocks as the value deviates from the zero level indicated by the black dashed line. Note that this statistical property is so noisy that it can not be clearly quantified with a single time-series at the daily scale.

2.6. Gain/loss asymmetry

The gain/loss asymmetry is the observation that the speed of the price fall is faster than that of the price rise. The authors of [39] defined $T_{\text{wait}}^t(\theta)$, which is the number of time-step t' required to reach the price return θ from the time t

$$T_{\text{wait}}^t(\theta) = \begin{cases} \inf\{t' | \log p_{t+t'} - \log p_t >= \theta, t' > 0\} & (\theta > 0) \\ \inf\{t' | \log p_{t+t'} - \log p_t <= \theta, t' > 0\} & (\theta < 0). \end{cases} \tag{8}$$

They analyzed the probability distribution of T_{wait}^t relating to the speed of price movement to reach the certain positive and negative thresholds $\pm\theta$. Fig. 1(f) shows the probability distribution of T_{wait}^t for $\theta = 0.1$ in red and $\theta = -0.1$ in blue for S&P500 stocks. The peak of the positive returns comes after the peak of negative returns, indicating the presence of asymmetry in price up/down. This statistical property is also relatively noisy as the clear functional form [39] can not be seen in a single stock data (See Appendix A).



(a) linear unpredictability (b) heavy tailed distribution (c) volatility clustering

(d) leverage effect (e) coarse-fine volatility correlation (f) gain/loss asymmetry

Figure 1. Stylized facts for the stock prices of S&P 500 firms. Data points are averaged over 486 S&P firms, which are recorded longer than 1,000 days. (a) Linear unpredictability shown as the zero level of the auto-correlation function of the price return is at zero level for all lags. (b) Heavy tailed distribution demonstrated by the power-law decay of the tails of the probability distribution. (c) Volatility clustering expressed by the slow decay of auto-correlation function for the absolute values of the price return. (d) Leverage effects shown as the lead-lag correlation of the price return and thesquared price return. (e) Coarse-fine volatility correlation shown as the lead-lag correlation of the coarse and the fine volatility. (f) Gain/loss asymmetry shown in the difference of the probability distributions for the time-step required to reach the certain price return thresholds, $\theta = 0.1$ in red and $\theta = -0.1$ in blue.

2.7. Models of the statistical properties

Referring to those stochastic models, GARCH is able to reproduce the linear unpredictability and the heavy-tailed distribution. Siven and Lins [40] reported that the exponential GARCH (EGARCH) can even reproduce the leverage effect and the gain/loss asymmetry.

In the early works of agent-based models (including the highly abstracted toy models emerged in the field of econophysics), the heavy-tailed distribution and volatility clustering were the main target to be reproduced. Typically, Lux and Marchesi [4] reported that an agent-based models with fundamental and technical traders could reproduce fat-tailed distribution and volatility clustering. Challet and Marsili [41] proposed grand canonical minority game (GCMG), and observed that GCMG is also capable of reproducing the two major statistical properties. Chen et al. [42] built a microscopic agent-based model, which can reproduce the leverage effects in addition to the heavy-tailed distribution and volatility clustering.

3. Generative Adversarial Networks for Financial Time-Series Model

3.1. Generative Adversarial Networks

GANs are one of deep generative models to generate realistic data by training deep neural networks. GANs consist of two different neural networks, a generator G and a discriminator D. The generator G is responsible for the generation of data, and the discriminator D functions to judge the quality of the generated data and provide feedback to the generator G. These neural networks are optimized under game-theoretic condition: the generator G is optimized to generate data that deceive the discriminator D and the discriminator D is optimized to distinguish the source of the input, namely the generator G or realistic dataset. It is formulated into the following optimization problem

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

GANs are categorized as an end to end approach: the neural networks are directly trained from data so that no explicit assumption is required to build a model. As a simple but powerful framework, GANs have been applied to many types of data generation.

3.2. Applications of GANs to problems in finance

The application of GANs to problems in finance is an emerging topic. Zhou et al. [18] integrated the adversarial learning framework to the stock price prediction. In the model, the price prediction model is optimized to minimize the loss of the discriminator to make a realistic prediction on the price movement. Koshiyama et al. [19] introduced GANs to systematic trading strategies for fine-tuning model and finding combination of trading strategies. Hadad et al. [22] proposed an adversarial learning to disentangle the factors of financial time-series. The model learnt to separate entire market behaviors from specific price movement of stocks. Fiore et al. [20] and Zheng et al. [21] applied GANs to fraud detection problems. Fiore et al. [20] employed GANs to generate realistic fraudulent credit card transactions and use them to train the fraud detection system for improving its accuracy. Zheng et al. [21] proposed and applied a modified version of GANs to the telecom fraud detection. They reported that the adversarial learning contributed to the detection of suspicious transactions with a low misclassification rate. Kumar et al. [23] and Doan et al. [24] proposed to use GANs to generate pseudo customer orders.

3.3. Comparison with other deep approaches

The one most natural approach for time-series modeling is the auto-regressive approach. Deep auto-regressive models such as pixel RNN [43] and WaveNet [44] demonstrated their high capacity in generating audio and sequential image data.

Another popular deep generative model is variational auto-encoders (VAEs) [45]. VAEs learn the latent representation of data by variational inference with the encoder-decoder architecture. VAEs have been applied to computer vision and natural language processing [46] and their frameworks have been extended. Variational recurrent neural networks (VRNNs) [47] are one of the extension of VAE for the sequential data. VRNNs define a temporal latent representation to describe the sequentiality of data. Neural Stochastic Volatility Model (NSVM) [48] employed VRNN

for the volatility prediction. NSVM successfully modeled the stochasticity of financial time-series and outperformed GARCH and other variants in the volatility prediction at a daily scale.

These two categories of models are trained to predict values at each data point (price return). It is natural to apply them to the return/volatility prediction in financial time-series. Modeling stylized facts, however, requires the model to capture the global structures of time-series, which may not be acquired through training for the prediction of each value. GANs architecture trains the generator G using the signals of the discriminator D, which only concerns the properties of the whole time-series. Building auto-regressive models of financial time-series meets insurmountable difficulties [49] since financial time-series is highly unpredictable considering the arbitrage free condition and the market impact effect [50].

3.4. FIN-GAN

FIN-GAN is developed in this study as a deep generative model for financial time-series, which reproduces the stylized facts introduced in Section 2.

3.4.1. Network architecture

Following the general structure of GANs, FIN-GAN has two neural networks, the generator G and the discriminator D, which are responsible for the generation and the evaluation of financial time-series, respectively. The generator G is constructed to take the 100 dimensional Gaussian noise z as the input and to output the $N = 8,192$ steps time-series. N is set to a large value because the length of time-series should be long enough to reliably calculate the statistical properties such as the probability distribution and the correlation functions. The multi-layer perceptron (MLP), convolutional neural networks (CNNs) and the combination of these two neural networks (MLP-CNNs) are tested for the generator G. Note that MLP and CNNs are used in many successful GANs [6, 51] and are the de facto standard in deep learning research [52]. The MLP architecture has four layers of fully connected neural networks with the hyperbolic tangent activation function. The CNNs architecture has six convolutional layers and the leaky ReLU activation function. The MLP-CNN combines these two architectures by the element-wise multiplication of their outputs.

Recurrent neural networks are excluded from this experiment because the computational cost for generation and optimization is infeasible due to the extremely long length of the time-series. MLP-CNNs is also chosen for the discriminator D. The investigation of the effects of the network architecture is focused on the generator rather than the discriminator, because the generation of data is considered as a harder task. In fact, the network architecture of the generator critically affects the quality of the generated time-series in the experiment.

3.4.2. Algorithm and details

The algorithm of FIN-GAN is generally the same to the originally proposed GANs [51]. The discriminator D is optimized to distinguish the source of the fed data (the dataset or the generator) and the generator G is optimized to deceive the discriminator D. See Algorithm 1 for the detail. Adam [53] was chosen as the stochastic gradient algorithm to update the parameters of the generator θ_G and the discriminator θ_D. The learning rate is set to $\alpha_G = 2 \cdot 10^{-4}$ and the parameter $\beta_{1,G} = 0.5$ for the generator and the learning rate $\alpha_D = 1 \cdot 10^{-5}$ the parameter $\beta_{1,D} = 0.1$ for the discriminator. The rest of the hyper-parameters of Adam are set to the originally proposed values. Batch normalization [54] is applied to the MLP and CNNs layers of the generator. The effect of batch normalization is investigated with the MLP generator. A technique called noisy labeling [55] is also used in the implementation, with which noise are added to the labels for D. Uniform distribution from 0.9 to 1.1 and from 0.1 to 0.3 are used for the labels of dataset and generated time-series, respectively. The batch size is set to $m = 24$. The batch of the real data of $8,192$ steps is randomly fetched from the dataset. The most of the hyper-parameters are taken from implementations of GANs for image generation and are unchanged from the default values. Learning rates of Adam optimizers are tuned by hand to maintain the robustness of the training process. Excessively large values of learning rates lead the model to fail to be well trained.

4. Result

In this section, the statistical properties of the generated time-series from FIN-GAN are analyzed and compared with that of the real financial time-series. Specifically, the effect of network architectures of the generator G and batch normalization to the MLP generator to the quality of generated time-series is investigated. The reproducibility of the major stylized facts of the models is summarized in Table. 1.

Algorithm 1 FIN-GAN

Require: : $\{\alpha_G, \beta_{1,G}\}$, learning rate and β_1 parameter of Adam (stochastic gradient algorithm) for G. $\{\alpha_D, \beta_{1,D}\}$, learning rate and β_1 parameter of Adam for D. m, batch size.

Initialize the parameters of the generator θ_G and the discriminator θ_D
while θ_G is not converged **do**
  Sample $\{z^{(i)}\}_{i=1}^m, z^{(i)} \sim \mathcal{N}(0, 1)$ ▷ Sample a batch of prior
  Sample $\{x^{(i)}\}_{i=1}^m$ ▷ Sample a batch of financial time-series
  Sample $\{G(z^{(i)})\}_{i=1}^m$ ▷ Sample a batch of time-series from G with prior
  Assign the label of source $\{y\}_{i=1}^{2m}$.
  $\nabla\theta_D \leftarrow \mathrm{Adam}(\theta_D, \{G(z^{(i)})\}_{i=1}^m + \{x^{(i)}\}_{i=1}^m, \{y\}_{i=1}^{2m})$ ▷ Compute the gradient of D to the target $\{y\}$
  $\theta_D \leftarrow \theta_D - \alpha_D \cdot \nabla\theta_D$ ▷ Update the parameter of the discriminator
  $\nabla\theta_g \leftarrow \mathrm{Adam}(\theta_G, \{z^{(i)}\}_{i=1}^m, \{1\}_{i=1}^m)$ ▷ Compute the gradient of G to deceive D
  $\theta_G \leftarrow \theta_G - \alpha_G \cdot \nabla\theta_G$ ▷ Update the parameter of the generator

4.1. Effect of network architecture

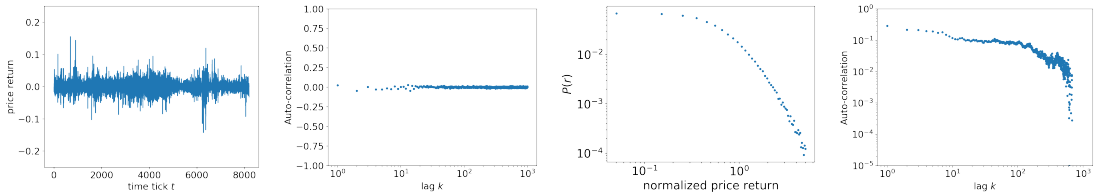
It was first observed that the time-series generated by the MLP generator satisfies all six major statistical properties of financial time-series. Fig. 2 shows one example of time-series generated by the MLP generator and its statistical properties, which are averaged over a batch of sequences. Fig. 2(b) shows the auto-correlation function of the generated price return series. As is comparable to Fig. 1(a), the generated time-series does not have any auto-correlation. Fig. 2(c) shows the positive tail of the price return distribution. The tail clearly follows a power-law decay, and the exponent is $\alpha = 4.68$, which satisfies the condition $3 < \alpha < 5$. The generated time-series also retains volatility clustering as shown in Fig. 2(d). Referring to the leverage effect, the function $L(k)$ is notably negative at small k and it reaches to zero level in Fig. 2(e).

The model also satisfies the asymmetry in coarse-fine volatility correlation as the value of the lead-lag correlation difference $\Delta\rho(k)^{\tau}_{cf}$ at $k = 1$ is negative as shown in Fig. 2(f). The gain/loss asymmetry is also clearly observed in Fig. 2(g). Similar to the MLP generator, the MLP-CNNs generator also outputs time-series that satisfies the major stylized facts as shown in Fig. 3. The model, however, does not successfully reproduce the asymmetry in the coarse-fine volatility correlation. In contrast to these successful generators, the CNNs generator cannot be well optimized to output a time-series close to the real data. In particular, the generated price returns fluctuate at a higher level above zero with a strong auto-correlation.

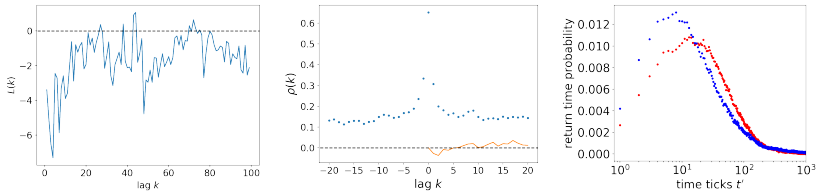
The difference among these three different network architectures can be understood from the generated time-series without training. Fig. 4 shows the time-series generated by the generator G with the MLP, CNNs and MLP-CNNs generator without training. The MLP generator without training outputs a fluctuating time-series around the zero level with relatively large magnitudes as shown in Fig. 4(a). The output of the MLP-CNNs generator Fig. 4(c) similarly fluctuates around the zero level with a smaller variance. Compared with these network architecture, the output of the initial CNNs generator shown in Fig. 4(b) does not consistently fluctuate around the zero level. It can be inferred that the network architecture of the generator G characterizes statistical properties of its initial state and therefore affects the quality of generated time-series after training.

4.2. Effect of batch normalization

It was observed in the experiment that the use of the batch normalization [54], a standard technique in GANs for image generation [6], led the MLP generator to fail in generating the realistic time-series. The layers of the batch normalization rescale the hidden vectors (computed at the middle of the generator) to keep the mean and the variance of the vector consistent during training. With the batch normalization, however, the generated time-series have an extremely large fluctuation and is strongly auto-correlated, which are different from the financial time-series. In the stochastic process modeling of financial time-series, the formulation of the variance is the key to successfully reproduce the statistical properties. The batch normalization, contrary to the naive expectation, is an obstacle for the train process to learn the way to control the variance. This observation suggests that some of common techniques in deep learning would also not be applicable to the modeling of the financial time-series.

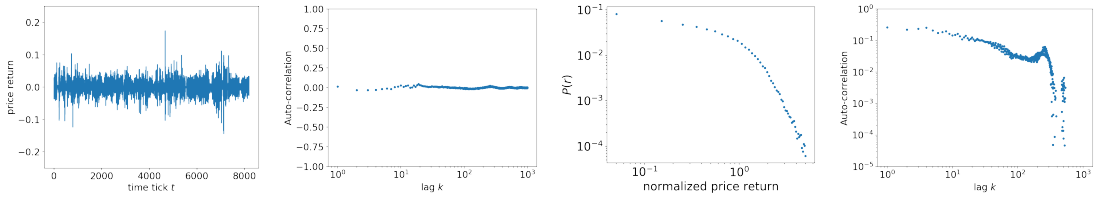


(a) time-series (b) linear unpredictability (c) heavy-tailed distribution (d) volatility clustering

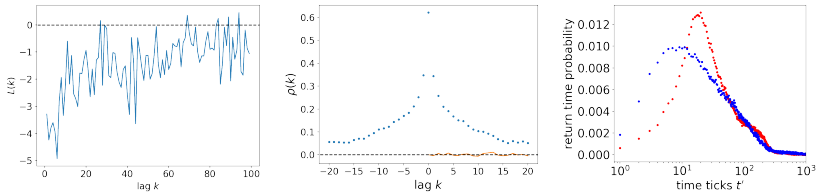


(e) leverage effect (f) coarse-fine volatility correlation (g) gain/loss asymmetry

Figure 2. Time-series generated by MLP generator and its statistical properties. Note that the statistical properties are averaged over 24 sequences of the length of 8, 192. (a) Time-series has the zero mean and the sufficient price return fluctuations. (b) The auto-correlation function for the price return is absent for any $k \geq 1$. (c) The positive tail of the price return probability distribution follows the power-law decay with the exponent $\alpha = 4.68$. (d) The auto-correlation function for the absolute price return remains persistently up to $k \approx 10^3$. (e) Leverage effects are present as the lead-lag correlation $L(k)$ takes negative values for small k and roughly follows the exponential decay. (f) Coarse-fine volatility correlation is reproduced as the value of $\Delta\rho^{\tau}_{cf}(k)$ is negative for $k = 1$. (g) Gain/loss asymmetry is also reproduced as the peak of the distribution for the positive return (red) comes after that of the negative return (blue).

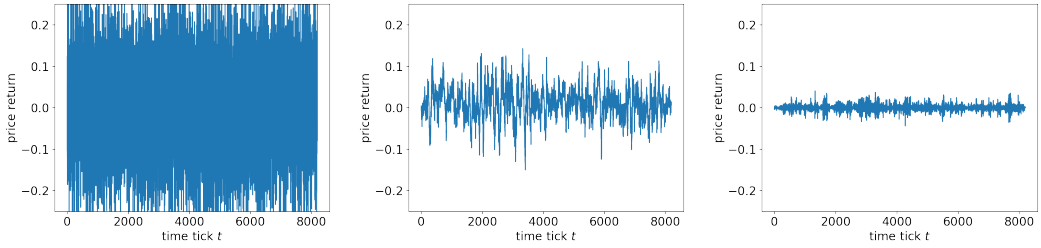


(a) time-series (b) linear unpredictability (c) heavy-tailed distribution (d) volatility clustering



(e) leverage effect (f) coarse-fine volatility correlation (g) gain/loss asymmetry

Figure 3. Time-series generated by MLP-CNNs generator and its statistical properties. The statistical properties are averaged over 24 sequences of the length of 8, 192. (a) Time-series has the zero mean and the sufficient price fluctuation. Except for (f) coarse-fine volatility correlation, stylized facts are reproduced by the MLP-CNNs generator. (b) Linear unpredictability shown as auto-correlation function for the price return, (c) Heavy-tailed distribution of the price return with the power-law exponent $\alpha = 4.04$, (d) Volatility clustering shown as auto-correlation function for the absolute price return, (e) Leverage effects shown as the lead-lag correlation function $L(k)$, and (g) Gain/loss asymmetry shown as the difference between the probability distributions for $T^{t}_{\mathrm{wait}}(\theta)$.



(a) MLP (b) CNNs (c) MLP-CNNs

Figure 4. Generated time-series without training. (a) The MLP generator produces a time-series with relatively large fluctuation. (b) The CNNs architecture generates a fluctuated time-series. However, the time-series already has a strong auto-correlation. (c) The generated time-series of the MLP-CNNs generator has the small fluctuation.

Table 1. Reproducibility of stylized facts with generators. * BN stands for batch normalization. N/A indicates that reproducibility is not reported.

Model	linear unpredictability	heavy-tailed distribution	volatility clustering	leverage effect	coarse-fine volatility correlation	gain/loss asymmetry
GARCH [2]	✓	✓	X	X	X	X
EGARCH [40]	✓	✓	X	✓	N/A	✓
Chen et al. [42]	✓	✓	✓	✓	N/A	N/A
MLP	✓	✓	✓	✓	✓	✓
MLP with BN*	X	X	X	X	X	X
CNN	X	X	X	X	X	X
MLP-CNN	✓	✓	✓	✓	X	✓

4.3. Robustness of training

It was reported that GANs often suffer from difficulties met in the training process [55]. In the case of FIN-GAN, however, the robustness of the training process is generally high. Fig. 5 shows the loss of the MLP generator G and the discriminator D. The losses of these models are limited in a certain range of values, which indicates that the two models are so competitive that the discriminator D sends the meaningful signal to the generator G. The independency of the seeds of the random number generator is also verified. Fig. 6 shows the time-series generated by four different random seeds. The training process and the reproducibility of stylized facts are independent of random seeds. Furthermore, Sub-sequences of the generated time-series are also analyzed to check whether the statistical properties are preserved in parts of the time-series. Fig. 7 shows the statistical properties of the first 4000 time-steps of the generated time-series by the MLP generator. Compared with the whole time-series, the statistical properties are generally preserved except the auto-correlation function of the absolute price return has a faster decay. Yet, the function roughly follows the power-law function up to $k \approx 10^2$. Overall, FIN-GAN has the high robustness in its training process.

5. Conclusion and Future Direction

In this paper, we developed FIN-GAN, GANs for financial time-series modeling. FIN-GAN is capable of generating realistic financial time-series that retains the major stylized facts such as the linear unpredictability, the fat-tailed price return distribution, volatility clustering, the leverage effects, the coarse-fine volatility correlation, and the gain/loss asymmetry. While most of the application of deep neural networks focused on the prediction of the price returns or the volatility, our study is the first proposal to use deep neural networks to build a model of financial time-series satisfying the major stylized facts. This deep neural networks based approach is an 'end-to-end' one, namely the model directly learns the properties of data and requires neither explicit assumptions nor mathematical formulations. This is a notable advantage over the other modeling approaches such as stochastic process modeling and agent-based modeling that requires explicit and non-trivial assumptions for the construction.

The application of GANs is open to the modeling of other complex systems. This paper demonstrated the capacity of GANs to reproduce the fat-tail distributed and long-range dependent time-series, which are discovered in a wide range of complex phenomena with a seemingly high level of unpredictability, which includes natural language, biological

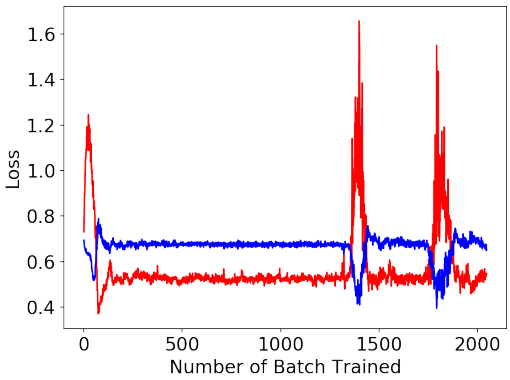
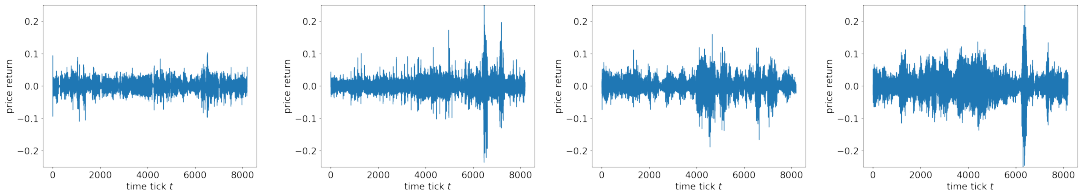
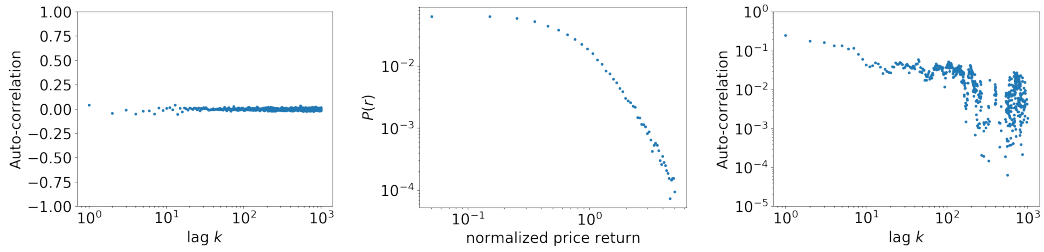


Figure 5. Loss of MLP generator G (red) and discriminator D (blue) in training. The generator G and the discriminator D are so competitive that the losses do not explode or converge to zero. While there are abrupt increases of the generator's loss and the corresponding decreases of the discriminator's loss, the losses recover to the previous state.

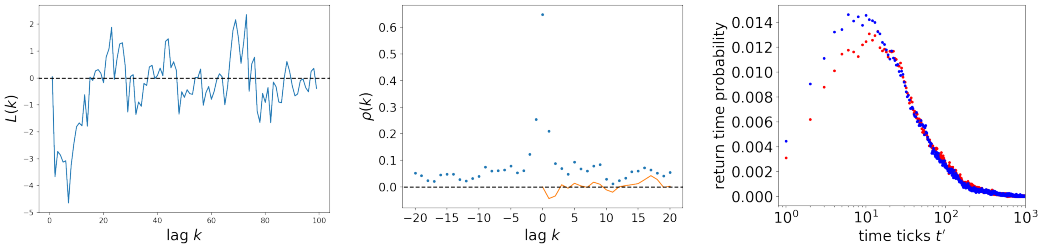


(a) time-series with seed A (b) time-series with seed B (c) time-series with seed C (d) time-series with seed D

Figure 6. Time-series generated by MLP-CNNs generators with different seeds. MLP-CNNs generator is able to produce the realistic time-series regardless of the choice of seed for a random number generator.



(a) linear unpredictability (b) heavy-tailed distribution (c) volatility clustering



(d) leverage effect (e) coarse-fine volatility correlation (f) gain/loss asymmetry

Figure 7. Statistical properties of first 4000 time-steps of time-series generated by MLP generator. The statistical properties are generally preserved even in the part of the generated time-series.

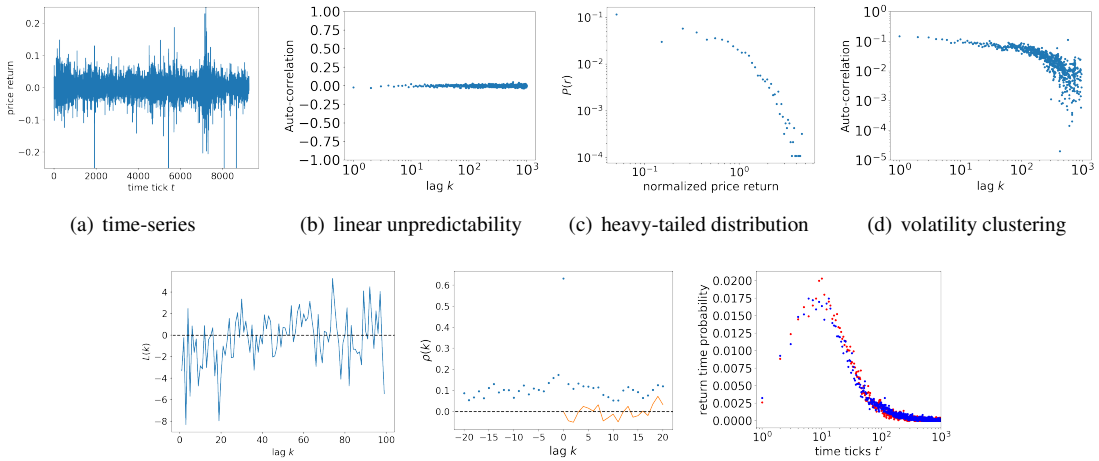
systems, internet traffic etc. In the studies of these complex systems, the construction of either a mathematical or a simulation model was regarded as a major challenge. GANs and potentially other deep generative models may enable us to construct a useful model for various behaviors of complex systems even without a full understanding of the mechanism working behind. Another future work is the exploration of more proper network architectures of the generator G. While the major network architectures (i.e. MLP, CNNs, and MLP-CNNs) are tested, more effective architectures are found in deep learning researches to the specific domains [56, 57, 58]. Possibly, effective architecture for the financial time-series model could be different from the major network architectures.

Acknowledgement

This work has been supported in part by HITE funding. We thank Hiroshi Okuda for his encouraging comments.

Appendix A. Statistical properties of single S&P500 firms

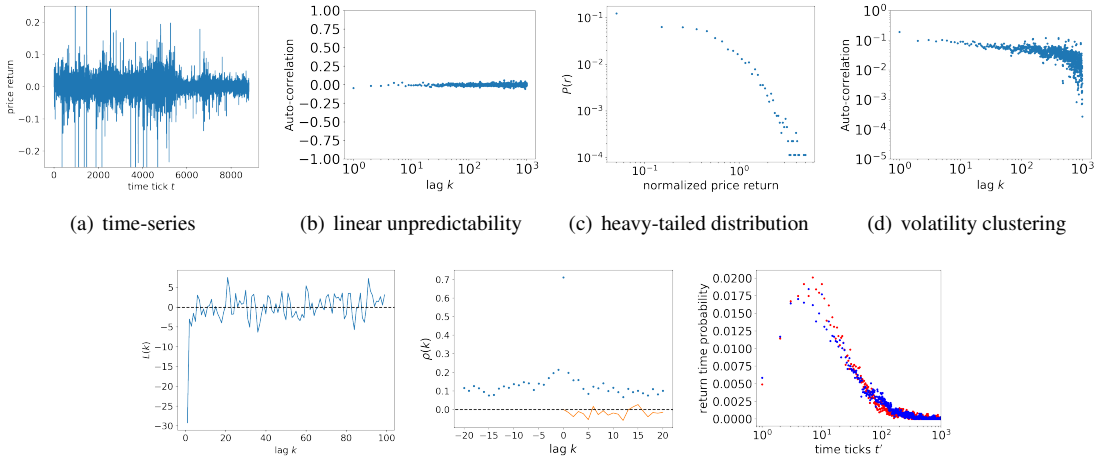
This appendix is a supplementary information to Section. 2. In Section. 2, the major stylized facts are introduced and Fig. 1 showed the averaged statistical properties of S&P 500 firms. Fig. Appendix A1, Fig. Appendix A2, and Fig. Appendix A3 are the statistical properties of the single firms. While the general tendency is confirmed, the data points are noisier and the functional forms of the properties are less clear than those averaged in Fig. 1.



(a) time-series (b) linear unpredictability (c) heavy-tailed distribution (d) volatility clustering

(e) leverage effect (f) coarse-fine volatility correlation (g) gain/loss asymmetry

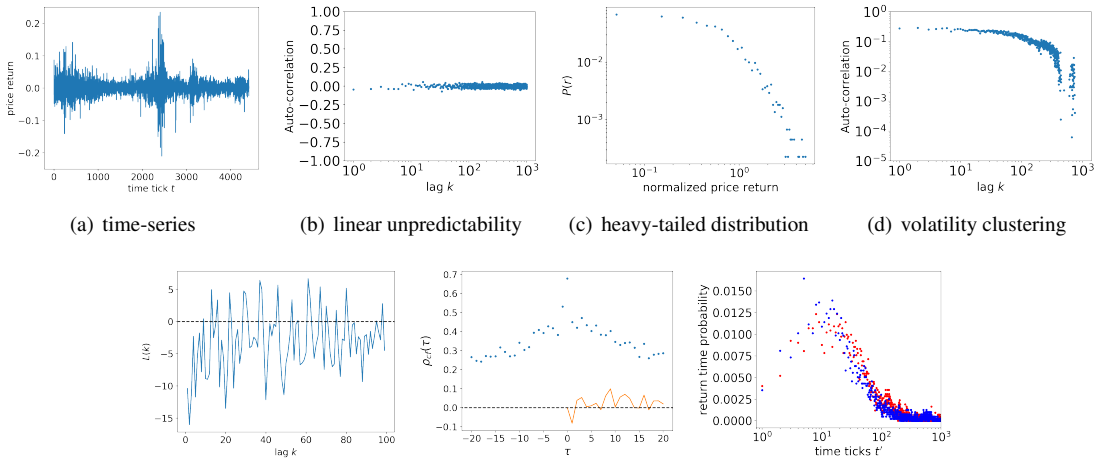
Figure Appendix A1. Time-series and statistical properties of ALK



(a) time-series (b) linear unpredictability (c) heavy-tailed distribution (d) volatility clustering

(e) leverage effect (f) coarse-fine volatility correlation (g) gain/loss asymmetry

Figure Appendix A2. Time-series and statistical properties of CA



(a) time-series (b) linear unpredictability (c) heavy-tailed distribution (d) volatility clustering

(e) leverage effect (f) coarse-fine volatility correlation (g) gain/loss asymmetry

Figure Appendix A3. Time-series and statistical properties of GS

References

[1] R. Engle, Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation, Econometrica 50 (1982) 987–1007.

[2] T. Bollerslev, Generalized autoregressive conditional heteroskedasticity, Journal of Econometrics 31 (1986) 307–327.

[3] D. Challet, Y.-C. Zhang, Emergence of cooperation and organization in an evolutionary game, Physica A 246 (1997) 407–418.

[4] T. Lux, M. Marchesi, Scaling and criticality in a stochastic multi-agent model of a financial market, Nature 397 (1999) 498–500.

[5] H. Malmsten, T. Teräsvirta, Stylized facts of financial time series and three popular models of volatility, European Journal of Pure and Applied Mathematics 3 (2010) 443–477.

[6] A. Radford, L. Metz, S. Chintala, Unsupervised representation learning with deep convolutional generative adversarial networks, in: Proceedings of the International Conference on Learning Representations, 2015.

[7] C. Donahue, J. McAuley, M. Puckette, Synthesizing audio with generative adversarial networks, in: Proceedings of the International Conference on Learning Representations, 2018.

[8] L. Yu, W. Zhang, J. Wang, Y. Yu, Seqgan: sequence generative adversarial nets with policy gradient, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2017.

[9] Y. Zhang, Z. Gan, C. L, Generating text via adversarial training, in: Neural Information Processing Systems workshop, 2017.

[10] Z. Liu, S. Rodrigues, W. Cai, Simulating the ising model with a deep convolutional generative adversarial network, arXiv preprint arXiv:1710.04987 (2017).

[11] A. B. Farimani, J. Gomes, V. S. Pande, Deep learning the physics of transport phenomena, arXiv preprint arXiv:1709.02432 (2017).

[12] O. Shchur, A. Bojchevski, D. Zügner, S. Günnemann, Netgan: Generating graphs via random walks, in: Proceedings of the International Conference on Machine Learning, 2018.

[13] O. Mogren, C-rnn-gan: Continuous recurrent neural networks with adversarial training, arXiv preprint arXiv:1611.09904 (2016).

[14] L. C. Yang, S. Y. Chou, Y. H. Yang, Midinet: A convolutional generative adversarial network for symbolic-domain music generation, in: International Society for Music Information Retrieval Conference, 2017.

[15] C. Esteban, S. L. Hyland, G. Rätsch, Real-valued (medical) time series generation with recurrent conditional gans, arXiv preprint arXiv:1706.02633 (2017).

[16] N. Killoran, L. J. Lee, A. Delong, D. Duvenaud, B. J. Frey, Generating and designing dna with deep generative models, arXiv preprint arXiv:1712.06148 (2017).

[17] A. Gupta, J. Zou, Feedback gan (fbgan) for dna: a novel feedback-loop architecture for optimizing protein functions, arXiv preprint arXiv:1804.01694 (2018).

[18] X. Zhou, Z. Pan, G. Hu, S. Tang, C. Zhao, Stock market prediction on high-frequency data using generative adversarial nets, Mathematical Problems in Engineering 2018 (2018) 4907423.

[19] A. Koshiyama, N. Firoozye, P. Treleaven, Generative adversarial networks for financial trading strategies fine-tuning and combination, arXiv preprint arXiv:1901.01751 (2019).

[20] U. Fiore, A. D. Santis, F. Perla, P. Zanetti, F. Palmieri, Using generative adversarial networks for improving classification effectiveness in credit card fraud detection, Information Sciences 479 (2019) 448–455.

[21] Y.-J. Zheng, X.-H. Zhou, W.-G. Sheng, Y. Xue, S.-Y. Chen, Generative adversarial network based telecom fraud detection at the receiving bank, Neural Networks 102 (2018) 78–86.

[22] N. Hadad, L. Wolf, M. Shahar, A two-step disentanglement method, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018.

[23] A. Kumar, A. Biswas, S. Sanyal, ecommercegan: A generative adversarial network for e-commerce, in: International Conference on Learning Representations workshop, 2018.
[24] T. Doan, N. Veira, B. Keng, Generating realistic sequences of customer-level transactions for retail datasets, in: Proceedings of the IEEE International Conference on Data Mining Workshops, 2018.
[25] I. Kaastra, M. Boyd, Designing a neural network for forecasting financial and economic time series, Neurocomputing 10 (1996) 215–236.
[26] E. Saad, D. Prokhorov, D. Wunsch, Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks, IEEE Transactions on Neural Networks 9 (1998) 1456–1470.
[27] R. Cont, Empirical properties of asset returns: stylized facts and statistical issues, Quantitative Finance 1 (2001) 223–236.
[28] A. Chakraborti, I. Toke, M. Patriarca, F. Abergel, Econophysics review: I. empirical facts, Quantitative Finance 11 (2011) 991–1012.
[29] R. Cont, Volatility Clustering in Financial Markets: Empirical Facts and Agent-Based Models, Springer Berlin Heidelberg, 2007, pp. 289–309.
[30] Y. Liu, P. Gopikrishnan, M. Meyer, C. Peng, H. Stanley, Statistical properties of the volatility of price fluctuations, Physical Review E 60 (1999) 1390–1400.
[31] J. Alstott, E. Bullmore, D. Plenz, powerlaw: a python package for analysis of heavy-tailed distributions, PLoS One 9 (2014) e85777.
[32] A. Clauset, C. Shalizi, M. review, Power-law distributions in empirical data, SIAM review 51 (2009) 661–703.
[33] J.-P. Bouchaud, A. Matacz, M. Potters, Leverage effect in financial markets: The retarded volatility model, Physical Review Letters 87 (2001) 228701.
[34] T. Qiu, B. Zheng, F. Ren, S. Trimper, Return-volatility correlation in financial dynamics., Physical Review E 73 (2006) 065103.
[35] X.-F. Jiang, B. Zheng, F. Ren, T. Qiu, Localized motion in random matrix decomposition of complex financial systems, Physica A 471 (2017) 154–161.
[36] U. A. Müller, M. M. Dacorogna, R. D. Davé, R. B. Olsen, O. V. Pictet, J. E. von Weizsäcker, Volatilities of different time resolutions - analyzing the dynamics of market components, Journal of Empirical Finance 4 (1997) 213–239.
[37] T. H. Rydberg, Realistic statistical modelling of financial data, International Statistical Review 68 (2000) 233–258.
[38] V. V. Gavrishchaka, S. B. Ganguli, Volatility forecasting from multiscale and high-dimensional market data, Neurocomputing 55 (2003) 285–305.
[39] M. H. Jensen, A. Johansen, I. Simonsen, Inverse statistics in economics: The gain-loss asymmetry, Physica A 324 (2006) 338–343.
[40] J. V. Siven, J. Lins, Gain/loss asymmetry in time series of individual stock prices and its relationship to the leverage effect, arXiv preprint arXiv:0911.4679 (2009).
[41] D. Challet, M. Marsili, Criticality and market efficiency in a simple realistic model of the stock market, Physical Review E 68 (2003) 036132.
[42] J.-J. Chen, B. Zheng, L. Tan, Agent-based model with asymmetric trading and herding for complex financial systems, PLoS One 8 (2013) e79531.
[43] A. van den Oord, N. Kalchbrenner, K. Kavukcuoglu, Pixel recurrent neural networks, in: Proceedings of the International Conference on Machine Learning, 2016.
[44] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, K. Kavukcuoglu, Wavenet: A generative model for raw audio, arXiv preprint arXiv:1609.03499 (2016).
[45] D. Kingma, M. Welling, Auto-encoding variational bayes, in: Proceedings of the International Conference on Learning Representations, 2014.
[46] S. R. Bowman, L. Vilnis, O. Vinyals, A. Dai, R. Jozefowicz, S. Bengio, Generating sentences from a continuous space, in: Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, 2016.
[47] J. Chung, K. Kastner, L. Dinh, K. Goel, A. Courville, Y. Bengio, A recurrent latent variable model for sequential data, in: Advances in Neural Information Processing Systems, 2015.
[48] R. Luo, W. Zhang, X. Xu, J. Wang, A neural stochastic volatility model, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2018.
[49] A. Borovykh, S. Bohte, C. W. Oosterlee, Conditional time series forecasting with convolutional neural networks, arXiv preprint arXiv:1703.04691 (2017).
[50] B. Malkiel, E. Fama, Efficient capital markets: A review of theory and empirical work, The Journal of Finance 25 (1970) 383–417.
[51] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Advances in Neural Information Processing Systems, 2014.
[52] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (2015) 436–444.
[53] D. Kingma, J. Ba, Adam: A method for stochastic optimization, in: Proceedings of the International Conference on Learning Representations, 2015.
[54] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: Proceedings of the 32nd International Conference on Machine Learning, 2015.
[55] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, Improved techniques for training gans, in: Advances in Neural Information Processing Systems, 2016.
[56] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016.
[57] S. Zhang, L. Wen, X. Bian, Z. Lei, S. Z. Li, Single-shot refinement neural network for object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018.
[58] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Łukasz Kaiser, I. Polosukhin, Attention is all you need, in: Advances in Neural Information Processing Systems, 2017.