Capturing dynamics with noisy quantum computers

Dechant, D.S.

Citation

Dechant, D. S. (2026, February 17). *Capturing dynamics with noisy quantum computers*. Retrieved from https://hdl.handle.net/1887/4290771

CHAPTER 5

Quantum Generative Modeling for Financial Time Series with Temporal Correlations

5.1 Introduction

In recent years, the use of machine learning - in particular neural-network based approaches - has expanded across many domains [48]. A particular approach are the generative adversarial networks (GANs), in which successively a generator and a discriminator are trained [210, 211]. The generator learns the underlying distribution and samples from it, while the discriminator learns to distinguish real data from generated samples. Typically, GANs are applied in image generation [212–214], but also to other data [215].

The ability of machine learning models to generalize well relies on the availability of large datasets [48]. Data augmentation methods are techniques which increase the training data set in order to alleviate these limitations [216]. Those methods typically involve slightly modifying the training data, but synthetic data generation by GANs is used as an approach for data augmentation [216, 217].

This is particularly relevant for finance, a computationally-heavy, yet difficult-to-model field [77]. Unlike domains where there is an abundance of high-quality data to train on, finance faces a fundamental challenge: the inherent non-repetitive nature of financial events. For example, the time-series of a specific

---

The contents of this chapter have been published in Ref. [91].

asset's price can only be observed once. A machine learning model that aims to learn properties based on the time-series of a specific asset therefore is heavily limited, as their ability to generalize well relies on large datasets. By learning the underlying distribution of financial time series as well as its desired temporal properties, one can generate new data that enables the creation of richer training sets [218, 219]. In particular, temporal correlations such as volatility clustering (periods of large variation are followed by periods of large variation, as do periods of low variation) are important for single time series.

Parallel to the development of machine learning, research in quantum computing and its potential application has increased. Motivated by the progress in quantum hardware, quantum algorithms research has been focusing on variational quantum algorithms in the last decade [20]. These hybrid algorithms consist of succinct calculations on a parameterized quantum circuit (PQC) and a classical optimizer [21]. The PQC contains tunable and fixed gates, and the classical optimizer calculates updated parameters based on measurements conducted on the final quantum state of the PQC at each step, until a certain precision or goal is reached.

Previous work proposed replacing the classical generator of a GAN with a parameterized quantum circuit [220, 221]. Quantum circuits have been proven to enable sampling from distributions which are intractable for classical circuits, and so the set of distributions they access is in general different than what classical models access. Consequently, it is expected they may be more effective with some classes of distributions that classical models struggle with [55, 56, 79, 222]. Subsequent studies expanded on this idea and examined if this property can be harnessed in the context of learning financial distributions. In particular, in [223], the idea to use QGANs for synthetic data generation of financial time series has been proposed and tested, using a quantum circuit Born machine, which performed better than a classical restricted Boltzmann machine on learning the distribution of correlated asset pairs with respect to the Wasserstein distance. However, generating financial time series which do not only follow the same distribution as real-world data, but also show their temporal correlations is challenging [78].

In this chapter, we develop a quantum GAN with a PQC as an expectation value sampler-based generator and a classical neural network as a discriminator for synthetic data generation and examine its ability in generating time series replicating the S&P 500 index, including its distribution and temporal correlations. The discrete time series spans from 20 to 40 points in time, where the expectation value of single-qubit Pauli-X and Pauli-Z operators is interpreted as the log return of the value of the index at each time step. The choice of these observables will be motivated in Sec. 5.3.2. We simulate the quantum circuits both with full-state simulations and with matrix product state (MPS) simulations (also known as the tensor train) [224, 225]. While full-state simulations are limited to short time intervals due to their exponential scaling, MPS simulations enable us to model longer time series by exploiting their ability to efficiently replicate linear structures such as financial time series. In the MPS simulations, we vary the bond dimension (also referred to as the tensor train rank) to balance computational costs and simulation accuracy.

We show that our model generates time series whose distribution closely matches that of the real time series. Furthermore, the generated samples show temporal correlations, which are qualitatively similar to those observed in real-world data. Our work shows the potential use of QGANs in learning time series with specific temporal correlations.

This chapter is organized as follows. In Sec. 5.2, we introduce the concepts of financial time series and its properties, as well as the concept of QGANs. Further, we cover related work. In Sec. 5.3, we present the simulations we used. We detail the data pre-processing as well as the quantum generator and the matrix product state simulation. We show the results of our simulations in Sec. 5.4, discuss them in Sec. 5.5, and conclude in Sec. 5.6.

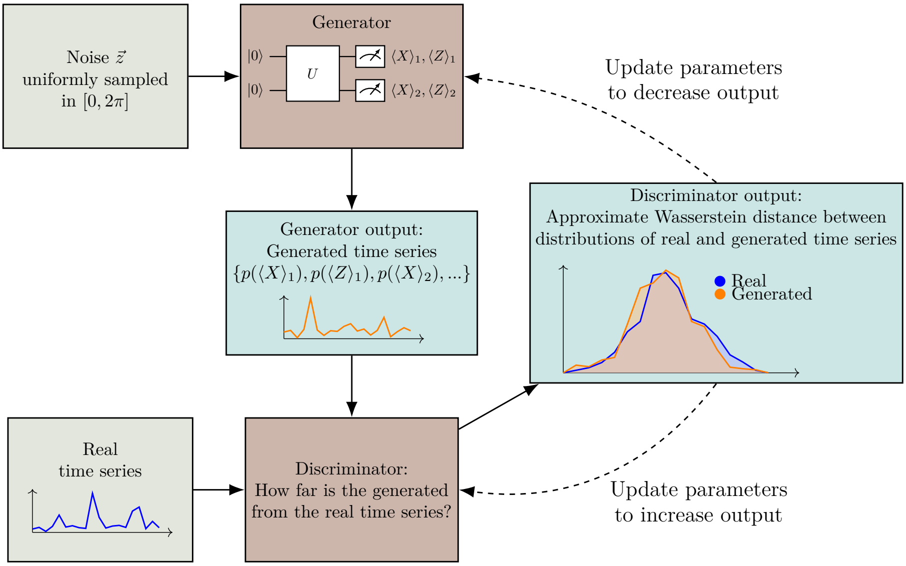


Figure 5.1: Structure of a generative adversarial network (GAN) used for time series generation. The discriminator takes both the generated and real time series as input and outputs its estimate for the Wasserstein distance $W_1(\mathcal{P}_r, \mathcal{P}_g)$ (see Eqs. (5.7) and (5.8)). The generator is trained in order to bring the distribution of the generated time series closer to the one of the real time series, the discriminator is trained to approximate the Wasserstein distance between them. Both the generator and discriminator are trained using different loss functions derived from the discriminator's output.

5.2 Background

In this section, we will introduce both financial time series and its temporal correlations as well as generative adversarial networks, and their adaptations based on the Wasserstein distance and with a quantum generator. Furthermore, we will present related work.

5.2.1 Financial time series

A time series is a set of data points ordered over a given time frame, typically at equally spaced time intervals. A financial time series is the set of financial variables such as prices, returns and volatility of assets, indices or other financial instruments.

An example of a financial time series is the S&P 500 index, which includes 500 of most valuable companies that are listed on US stock exchanges [75]. The price of many instruments, such as the S&P 500 index, deviates around a mean value that grows in time. Instead, examine the time series of the log return r_t, which is not dependent on this general market growth:

$$r_t = \log\left(\frac{S_t}{S_{t-1}}\right), \tag{5.1}$$

where S_t is the price of the index at time t. Simple models such as the Black-Scholes model [81] assume a normal distribution of these log returns. However, the distributions observed in the market have more complicated properties. The returns of assets do not typically follow a normal distribution, their distributions are more narrowly and spiked around the mean and have heavier tails (extreme events are more likely), which is in contrast to the normal distribution of the Black-Scholes model. Therefore, a main concern of research in finance is about creating models that can mimic time series with higher accuracy, and machine learning approaches have been increasingly explored for this case [77]. Many observed log returns share common properties, also called stylized facts, that originate mostly from behavior of parties that interact with the market [76]. These properties can be used in order to assess the quality of models of financial time series. In practice, it is considerably more difficult to generate time series that observe all of the stylized facts, than to only match the target distribution. However, for finance practitioners it is often vital to use models which show the stylized facts that are relevant for their use case [78] .

In this chapter, we focus on four stylized facts: non-Gaussianity, the absence of linear autocorrelation, volatility clustering and the leverage effect. The first of them is describing the behavior of the time-aggregated distribution. As written above, it is not shown in the Black-Scholes model, but is generally seen in real-world data. The latter three stylized facts are all temporal correlations between different values of the same time series. They can be analyzed with the help of the correlation function $corr(X, Y)$, which for the random variables X and Y is defined as:

$$corr(X,Y) = \frac{cov(X,Y)}{\sigma_X \sigma_Y} = \frac{\langle (X-\mu_X)\,(Y-\mu_Y)\rangle}{\sigma_X \sigma_Y}, \tag{5.2}$$

where $cov(X, Y)$ is the covariance, σ_X and σ_Y the standard deviations of the random variables and μ_X and μ_Y denote their respective means. For a sequence of independent random variables, the autocorrelation function vanishes at all nonzero time lags. For instance, the increments of Brownian motion are independent and therefore exhibit zero autocorrelation. The stylized facts of the absence of linear autocorrelation, volatility clustering and the leverage effect each have an intuitive reason for their emergence and can be observed by analyzing the autocorrelaiton of the absolute and identical values of the time series.

Firstly, the current and past values of financial time series are typically not linearly autocorrelated. At time t, this means that for all time differences $\tau > 0$, the expectation value $\mathbb{E}\left[corr(r_t, r_{t+\tau})\right]$, taken over different realizations of the time series, is close to zero. Intuitively this comes from the fact that any trend in the return is exploited by traders, which in turn weaken the effect. This exploitation of traders is a corollary of the so-called efficient market hypothesis.

Secondly, the absolute returns typically do exhibit correlation that slowly decays in time. This effect is also called volatility clustering, and can be examined by calculating the quantity $corr(|r_t|, |r_{t+\tau}|)$. It quantifies the observation that large changes in the price are followed by large changes, and equivalently small changes are followed by small changes.

Thirdly, the leverage effect describes the rise in volatility when the price of an asset sinks. It can be observed by measuring the quantity $corr(|r^2_{t+\tau}|, r_t)$.

The reader can find more details of these properties in [76].

Synthetic data generation of financial time series concerns the generation of artificial time series that observe these stylized facts. These properties and their importance for practitioners differ depending on the time series and the application, and they are difficult to compare in general. Furthermore, different models generate time series with greatly varying quality in reproducing the stylized facts. Therefore, the resulting synthetic time series are typically assessed qualitatively if they are able to capture those properties [78].

However, in this chapter, we provide several quantitative metrics. In order to quantify how closely the generated time series reproduce the stylized facts of the S&P 500 index, we define the following metrics:

$$EMD(\theta) = \frac{1}{\tau_{max}+1} \sum_{\tau=0}^{\tau_{max}} \left| r_{t+\tau}^{(SP500)} - r_{t+\tau}^{(\theta)} \right| \tag{5.3}$$

$$E_{id}^{ACF}(\theta) = \left(\frac{1}{\tau_{max}} \sum_{\tau=1}^{\tau_{max}} corr\left(r_t^{(\theta)}, r_{t+\tau}^{(\theta)}\right)^2 \right)^{1/2} \tag{5.4}$$

$$\begin{aligned} E_{abs}^{ACF}(\theta) =& \left(\frac{1}{\tau_{max}} \sum_{\tau=1}^{\tau_{max}} \left[corr\left(|r_t^{(SP500)}|, |r_{t+\tau}^{(SP500)}|\right) \right.\right. \\ & \left.\left. - corr\left(|r_t^{(\theta)}|, |r_{t+\tau}^{(\theta)}|\right) \right]^2 \right)^{1/2} \end{aligned} \tag{5.5}$$

$$\begin{aligned} E_{Lev}(\theta) =& \left(\frac{1}{\tau_{max}} \sum_{\tau=1}^{\tau_{max}} \left[corr\left(|r_t^{(SP500)}|^2, r_{t+\tau}^{(SP500)}\right) \right.\right. \\ & \left.\left. - corr\left(|r_t^{(\theta)}|^2, r_{t+\tau}^{(\theta)}\right) \right]^2 \right)^{1/2} . \end{aligned} \tag{5.6}$$

They are quantifying non-Gaussianity, absence of linear autocorrelation, volatility clustering and leverage effect, respectively. The first is based on the earth-movers distance, which is the discretized form of the Wasserstein distance, and the latter quantities are derived from the correlation functions describing these properties, as explained above. Here, the log returns of the generated time series are written as $r_t^{(\theta)}$, where θ stand for the parameters and hyperparameters of the simulated QGAN, and the log returns of the S&P 500 index are written as $r_t^{(SP500)}$. The lower these metrics are, the closer the stylized facts of the generated time series $r_t^{(\theta)}$ are resembling those of the time series $r_t^{(SP500)}$.

5.2.2 Wasserstein QGAN

Generative adversarial networks (GANs) [210, 211] are unsupervised machine learning-based methods that are powerful in generating images and have also been successfully applied to the generation of financial time series [226]. They consist of two neural network that compete in a game-like setup. The generator takes random noise as input and aims to create artificial data that is indistinguishable from real data. Both real data from the training set and artificial data from the generator is then fed to a discriminator which is being trained to detect the generated data, outputting a probability of the input data being real or fake. They are trained in an alternating fashion until the generator is able to create data indistinguishable from real data. GANs face challenges in training instability (one neural network overpowers the other) and mode collapse (The GAN focuses on creating data with limited variety) [227, 228].

Those challenges can be mitigated by replacing the discriminator with a critic that learns the Wasserstein distance between the real and generated data distributions, in the so-called Wasserstein GAN [228]. The Wasserstein distance between the real and generated probability measures $\mathcal{P}_r$ and $\mathcal{P}_g$, respectively, is defined as:

$$W_1(\mathcal{P}_r, \mathcal{P}_g) := \inf_{\pi \in \Gamma(\mathcal{P}_r, \mathcal{P}_g)} \mathbb{E}_{(x,y)\sim\pi} \left(\|x - y\|\right). \tag{5.7}$$

Here, $\Gamma(\mathcal{P}_r, \mathcal{P}_g)$ denotes the set of all couplings of $\mathcal{P}_r$ and $\mathcal{P}_g$, i.e., all joint probability measures whose marginals are $\mathcal{P}_r$ and $\mathcal{P}_g$. And by $(x, y) \sim \pi$, we denote that the random pair (x, y) is distributed according to the coupling π. Calculating this infimum is not feasible in practice, but the Kantorovich-Rubinstein duality delivers a quantity that can be used in a machine learning context [228, 229]:

$$W_1(\mathcal{P}_r, \mathcal{P}_g) = \sup_{\|f\|_L \le 1} \left(\mathbb{E}_{x\sim\mathcal{P}_r} \left(f(x)\right) - \mathbb{E}_{\tilde{x}\sim\mathcal{P}_g} \left(f(\tilde{x})\right)\right), \tag{5.8}$$

where $\sup_{\|f\|_L \le 1}$ is the supremum over all 1-Lipschitz functions. The role of the critic D is to maximise the loss function

$$\begin{aligned} L_D(D, \mathcal{P}_g) =& \mathbb{E}_{x\sim\mathcal{P}_r} \left(D(x)\right) - \mathbb{E}_{\tilde{x}\sim\mathcal{P}_g} \left(D(\tilde{x})\right) && (5.9) \\ & + \lambda \mathbb{E}_{\hat{x}\sim\mathcal{P}_{\hat{x}}} \left(\left(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1\right)^2 \right), && (5.10) \end{aligned}$$

where $D(x)$ is trained to approximate $f(x)$ in Eq. (5.8). The latter term is a gradient penalty regularization [230] that enforces the 1-Lipschitz condition by a scaling parameter λ and where $\hat{x} = \epsilon x + (1 - \epsilon)\tilde{x}$ with the random parameter $\epsilon \sim U[0, 1]$. This loss function will train the critic to approximate the Wasserstein distance between the probability distributions of real and generated data. In contrast, the role of the generator is to maximize

$$L_G(D, \mathcal{P}_g) = \mathbb{E}_{\tilde{x}\sim\mathcal{P}_g} \left(D(\tilde{x})\right). \tag{5.11}$$

See Fig. 5.1 for a sketch of a Wasserstein GAN.

Quantum generative adversarial networks (QGANs) are GANs in which the classical generator and/or the classical discriminator are replaced by a quantum circuit [220, 221].

They are motivated by the fact that quantum circuits can learn distributions efficiently that are hard to model by classical means [55, 56, 79, 222]. The proofs of advantage showcase that learning can not be achieved when the distribution is hard (generated by a quantum process). Market data is manifestly not so. However, in other models it was shown that one can have learning separations even if the generation of the data is classically tractable [231]. It remains an open question if such separations can also hold for estimation value sampler as we use here. However, even without separations it may be the case that quantum models simply have more convenient inductive biases than classical models and understanding those is valuable, and the interests of this work go

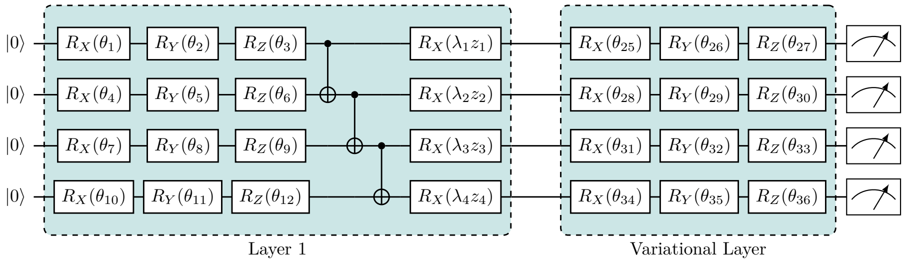


Figure 5.2: Example of a parameterized quantum circuit with 4 qubits and 1 layer used as the generator in the QGAN. Each layer consists of single-qubit Pauli rotations, CNOT gates and data uploading gates, which upload particular realizations of the random noise $\vec{z}$. Each element of $\vec{z}$ is uniformly sampled in $[0, 2\pi]$. For parameterized quantum circuits with more layers, each additional layer follows layer 1 and has the same layout. Before the measurement, there is a variational layer consisting of single-qubit Pauli rotations. The measurements are conducted single-qubit Pauli-X and Pauli-Z bases. The single-qubit Pauli rotations are tunable with parameters θ_i and the data-uploading gates are tunable with parameters λ_i. This Ansatz is a hardware-efficient Ansatz, which is commonly chosen in the field [21, 238].

into this direction.

A parameterized quantum circuit (PQC) consists of tunable and fixed gates, and measurements at the end. The parameters of the tunable gates are updated based on the loss function, which consists of the measurement results and the output of the discriminator.

The PQC can be used in different ways, for example as a quantum circuit Born machine [223, 232, 233] or as an expectation value sampler [234–237]. The latter approach is the one which we use for our QGANs. In the former case, a quantum circuit is used to learn an underlying distribution and every single sample forms a bit string corresponding to the learned distribution. The probability of each sample depends on the amplitudes of the final quantum state. However, the precision of the generated values is limited by the discrete nature of this approach. In contrast, the expectation value sampler identifies expectation values of quantum circuit measurement outcomes with samples of a distribution. The underlying randomness comes from classical noise uploaded to the quantum circuit.

A sketch of the PQC architecture used in this chapter is given in Fig. 5.2.

5.2.3 Related work

In recent years, classical algorithms for generating synthetic financial data have been proposed and explored, in GAN settings [78, 226, 239–241], and with other approaches [78, 242]. A common challenge is the generation of time series that exhibit all stylized facts sufficiently well [78].

QGANs were introduced in [220] and [221], substituting generator and discriminator with quantum circuits. Other approaches for QGANs, such as combining a classical discriminator with a quantum generator, and their applications have been explored as well [243, 244]. In [245] the generation of certain probability distributions and in [246], the generation of correlated stocks has been examined.

Our work was motivated by [223], which compared the performance of generating synthetic financial data of correlated asset pairs by the two models of restricted Boltzmann machines and quantum circuit Born machines, observing an advantage of the quantum circuit Born machine for comparable model sizes. Here we go beyond learning time-aggregated distributions of financial time series as in [223], by additionally examining the temporal correlations of generated time series. In contrast to the models used in [223], our method is based on the expectation value sampler, which was introduced in [234], proven to be universal in [236] and further generalized in [237]. An expectation value sampler outputs Pauli string expectation values in the range $[-1, 1]$, producing continuous variables. This is fundamentally different from quantum circuit Born machines, which generate discrete bit strings according to the Born rule, and Boltzmann machines, which also produce discrete outputs [223]. In Appendix 5.B, we adapted our approach for learning the distribution of correlated pairs of foreign exchanges and compare our results with the results of [223].

QGANs have also been used for other applications, such as image generation [247–250] and other discrete distributions [251], in generative chemistry [252], fraud detection [253], option pricing [254], and high-energy physics [255, 256]. Furthermore, other quantum machine learning strategies have been used in learning financial time series [257].

The Wasserstein QGAN, proposed in [258, 259] by substituting both generator and discriminator with a quantum circuit, shows improvement in training stability and efficiency compared to QGANs based on other metrics. Our simulations are using Wasserstein QGANs in which the generator is a PQC, whereas the discriminator is a classical neural network. The full-state simulation of such a Wasserstein-QGAN with gradient penalty as an application to generate financial time series has been explored in [260] and compared to classical GANs.

5.3 Implementation

For this chapter, we aim to generate time series whose distribution approximates the empirical distribution of real financial data. We train a Wasserstein QGAN for generating time series based on training data originating in the time series of the daily closing prices of the S&P 500 index [75], collected from 03.01.1950 until 29.09.2021. We use a hybrid approach, with a classical neural network as a discriminator and a parameterized quantum circuit (PQC) as a generator.

We use a convolutional neural network as a discriminator, motivated by [239], where it was used as the discriminator of a GAN that generates financial time series. All its activation functions are rectified linear units, except the last single neuron in the critic, which uses a linear activation function. The architecture of the discriminator is detailed in Appendix 5.A.

We simulated the QGAN based on a full simulation of the PQC by using the Tensorflow software library [261], and the QGAN based on the MPS approximation of the PQC with the JAX [262] and Quimb [263] software libraries. The gradients are calculated via automatic differentiation. As an optimizer for the training of the QGAN, we chose the Adam optimizer with a learning rate of 10^{-3}. All simulations were conducted on the Alice and XMARIS computing clusters at Leiden University.

In the following, we outline the data pre-and post-processing, the setup of the quantum generator, the full-state simulation and the MPS simulation applied in this chapter.

5.3.1 Data pre-and post-processing

The outputs of expectation value samplers are the expectation values of Pauli strings which lie in the range $[-1, 1]$. As this is a key difference to the raw time-series data in the form of log returns, which have unbounded support (see Eq. (5.1)), we perform data pre-and post-processing. For that, we follow the same approach as taken in Refs. [240, 264].

We first describe the approach for the pre-processing, which transforms the raw time series data of the S&P 500 index into training data used in our numerical simulations. This process consists of six steps: (i) data normalization, (ii) the inverse Lambert-W transform, (iii) data normalization, (iv) data clipping, (v) data rescaling and (vi) a rolling window.

(i) We normalize the time series data to have a mean of 0 and a variance of 1:

$$r_{t,(i)} := \frac{r_t^{(SP500)} - \mu_r}{\sigma_r}, \tag{5.12}$$

where $r_t^{(SP500)}$ is the original log return, calculated from the daily closing prices S_t of the S&P 500 index by $r_t^{(SP500)} = \log\left(\frac{S_t}{S_{t-1}}\right)$. By μ_r and σ_r, we denote the estimates of the mean and standard deviation of the log returns over the whole period of collected time series data (03.01.1950-29.09.2021).

(ii) As learning a heavy-tailed distribution can be challenging due to a limited number of samples in the tails, we implement the inverse Lambert-W transform on the normalized log returns. This transformation will bring the heavy-tailed distributed data closer to a Gaussian distribution. Given Lambert's W function, which is the inverse of $z = u\exp(u)$ with $z : \mathbb{R} \to \mathbb{R}$, we can define the following transform on the normalized heavy-tailed data set $V = \{r_{t,(i)}\}_t$:

$$W_\delta(r_{t,(i)}) := \operatorname{sgn}(r_{t,(i)}) \left(\frac{W(r_{t,(i)}^2 \delta)}{\delta} \right)^{1/2} \tag{5.13}$$

with $\delta \geq 0$ a tunable parameter, $\operatorname{sgn}(r_{t,(i)})$ the sign of $r_{t,(i)}$ and W the Lambert's W function. The inverse of this function is given by $r_{t,(i)} = W_\delta(r_{t,(i)}) \exp\left(\frac{\delta}{2} W_\delta(r_{t,(i)})^2\right)$ [265]. Throughout this chapter, we pick $\delta = 0.5$.

(iii) We normalize the transformed time series again such that it obtains a mean of 0 and a variance of 1:

$$r_{t,(iii)} := \frac{W_\delta(r_{t,(i)}) - \mu'_r}{\sigma'_r}, \tag{5.14}$$

where by μ'_r and σ'_r, we denote the estimates of the mean and standard deviation of the transformed time series $\{W_\delta(r_{t,(i)}\}_t$.

(iv) As the inverse Lambert-W transform can be ill-behaved at specific data points, we discard outliers with large deviations outside the 0.05% tails.

(v) Afterwards, we linearly map the data to the interval $[-1, 1]$. Let min and max denote the minimum and maximum values of the set $\{r_{t,(iv)}\}_t$, respectively. The transformation is given by

$$r_{t,(v)} = 2\,\frac{r_{t,(iv)} - \min}{\max - \min} - 1\,, \tag{5.15}$$

where $\{r_{t,(iv)}\}_t$ is the time series obtained after step (iv).

(vi) After these transformations of the log returns of the S&P 500 index, we divide the time series into smaller batches. We achieve this by applying a rolling window of window length m and stride s to the time series, which divides it into multiple subsequences. This creates subsequences with length m, which overlap and consequently correlate if the stride is shorter than the length of the window, $s < m$. For each of these subsequences, we then compute its probability distribution, which constitutes a sample of the training data set. Although the correlation between training samples is not ideal, the more extensive set of training samples can be beneficial for model performance. Throughout all simulations shown in this chapter, we used a stride of $s = 5$ and a window length of $m = 20$ and $m = 40$.

One sample of the resulting training data set is thus a subsequence of length 20 or 40 of the transformed time series of daily log returns of the S&P 500 index.

Each generated sample consists of the expectation values of $2n$ Pauli operators, $\{r_{t,PQC}\}_{t=1}^{2n} = \{\langle X\rangle_1, \langle Z\rangle_1, \langle X\rangle_2, \langle Z\rangle_2, ...\}$ from a PQC with n qubits (see Sec. 5.3.2), which are then post-processed by taking the following steps: (i)* data rescaling, (ii)* data renormalization, (iii)* forward Lambert transform, (iv)* data renormalization.

(i)* The data is rescaled by reversing step (v) in the pre-processing: The inverse mapping is given by

$$r_{t,(i)^*} = \frac{r_{t,PQC} + 1}{2} (\max - \min) + \min \,, \tag{5.16}$$

where $\{r_{t,PQC}\}_{t=1}^{2n}$ are the measured expectation values.

(ii)* The resulting set is normalized by reverting step (iii):

$$r_{t,(ii)^*} = \sigma_r' \, r_{t,(i)^*} + \mu_r' \,, \tag{5.17}$$

where μ_r' and σ_r' are calculated in the pre-processing step (iii).

(iii)* The inverse Lambert-W transformation is reversed by applying

$$r_{t,(iii)^*} = r_{t,(ii)^*} \exp\left(\frac{\delta \, r_{t,(ii)^*}^2}{2} \right) \,, \tag{5.18}$$

where δ is the parameter that we fix to 1/2 throughout the chapter.

(iv)* Finally, we also reverse the first normalization:

$$r_{t,gen} = r_{t,(iv)^*} = \sigma_r \, r_{t,(iii)^*} + \mu_r \,, \tag{5.19}$$

where μ_r and σ_r are the mean and standard deviation of the original time series.

At the end of the post-processing, each sample is a time series of length $2n$, which we write as:

$$\begin{aligned} r_{\text{gen}} &= \{r_1, r_2, r_3, r_4, ..., r_{2n}\} \\ &= \{p(\langle X \rangle_1), p(\langle Z \rangle_1), ..., p(\langle X \rangle_{2n}), p(\langle Z \rangle_{2n})\} \,, \end{aligned} \tag{5.20}$$

where by $p(\cdot)$ we denote the post-processing.

5.3.2 Quantum generator and full-state simulation

As a generator of the QGAN, we chose a PQC with an architecture that is sketched in Fig. 5.2, based on the hardware efficient Ansatz [21, 238]. The qubits are initialized in the $|0\rangle$ state. Each layer consists of single-qubit Pauli rotations, CNOT gates connecting nearest neighbors and noise encoding gates. The latter encode each a uniformly distributed noise sample with single-qubit rotations with trainable parameters. Such circuit architectures suffer from barren plateaus when scaled up in the number of qubits and layers [266]. Therefore, we do not consider them to be scalable in their current form. Instead, our investigation should be understood as establishing lower bounds on what can be achieved with quantum circuits: if these perform well at small scales, it motivates efforts to refine them for improved trainability at larger scales. Conversely, if they fail to perform even at small scales, this indicates that the application may be less promising than one might have hoped.

In Sec. 5.4, we present results from training circuits with 10 and 20 qubits and between 1 and 18 layers. This choice of the number of qubits and layers makes the QGANs classically simulatable. After the n-th layer, we apply single-qubit Pauli rotations and we measure each qubit in two bases: the Pauli-Z basis and the Pauli-X basis. We chose these measurements in order to enable the simulation of longer time series using fewer qubits. The exact consequences in terms of expressivity and potential greater sampling costs was discussed in [236, 267].

For the training of the QGAN and the analysis of the generated data at the end of the training, which we present in Sec. 5.4, we create samples of generated time series, each with a different random noise input. First, we collect the expectation values $\{\langle X\rangle_1, \langle Z\rangle_1, \langle X\rangle_2, \langle Z\rangle_2, ...\}$ at the end of the PQC, where $\langle X\rangle_i, \langle Z\rangle_i \in [-1, 1]$ are the expectation values of the measurement in the Pauli-X and Pauli-Z basis on the i-th qubit, respectively. Subsequently, we post-process the data as described in Sec. 5.3.1. After the post-processing, the obtained set $\{p(\langle X\rangle_1), p(\langle Z\rangle_1), p(\langle X\rangle_2), p(\langle Z\rangle_2), ...\}$, where $p(\cdot)$ stands as the post-processing map, constitutes one sample of a generated time series r_{gen} (see also Eq. (5.20)). A circuit of n qubits thus generates a time series of length $2n$. By generating multiple such time series samples from different random noise inputs, the QGAN approximates the empirical distribution of time series of the same window length in the training data.

The first part of our simulations is based on the full-state simulation of PQC. Let $|\psi_{\text{full}}\rangle$ be the quantum state that describes the state at the end of the parameterized quantum circuit (PQC) used in our quantum generative adversarial network (QGAN). In the computational basis, it takes the form

$$|\psi_{\text{full}}\rangle \approx \sum_{i_1, i_2, ..., i_n} c_{i_1, i_2, ..., i_n} |i_1, i_2, ..., i_n\rangle \ , \tag{5.21}$$

where $i_j \in \{0, 1\}$. The number of coefficients $c_{i_1, i_2, ..., i_n}$ of this state, and thus the memory requirement, scales exponentially with the number of qubits n. Moreover, the time cost of the classical full-state simulation scales linearly with the number of layers. This makes the full-state simulation quickly infeasible.

5.3.3 Matrix product state simulation

For being able to simulate PQC with a higher number of layers and qubits, we use matrix product states (MPS) as efficient approximation methods under some circumstances [224, 225, 268], which in the context of machine learning is also called the tensor-train decomposition [269]. They provide a compact representation of quantum states with limited entanglement and have been extensively used in physics [268]. This makes them well suited for simulating quantum states that are prepared by the PQC used in our QGAN.

An MPS represents an n-qubit quantum state $|\psi_{\text{full}}\rangle$ as a product of local tensors [268]:

$$|\psi_{\text{full}}\rangle = \sum_{i_1,\ldots,i_n} A^{[1]}_{i_1} A^{[2]}_{i_2} \cdots A^{[n]}_{i_n} |i_1 i_2 \cdots i_n\rangle \,, \tag{5.22}$$

where each $A^{[k]}_{i_k}$ is a $\chi_{k-1} \times \chi_k$-dimensional tensor. We call χ_k the bond dimensions of the MPS that controls the amount of entanglement the MPS can represent. The contraction of these tensors yields the amplitude corresponding to each computational basis state. For simplicity, we choose $A^{[1]}_{i_1}$ as $1 \times \chi$-dimensional, $A^{[n]}_{i_n}$ $\chi \times 1$-dimensional, and each remaining tensor $A^{[k]}_{i_k}$ with the dimensions $\chi \times \chi$. Such an MPS is described by $(2n-1)\chi^2 + 2\chi$ coefficients, which for constant χ scales linearly in the number of qubits, making it more efficient than the full-state simulation with 2^n coefficients. Fig. 5.3 provides a sketch of an MPS representation.

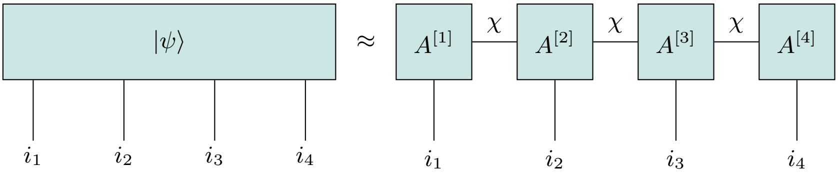


Figure 5.3: A matrix product state (MPS) consists of a chain of local tensors $A^{[j]}$ connected by virtual bonds of dimension χ, each with a physical leg representing a qubit index. A virtual bond represents a sum over a particular index of the tensors. The left term represents the state ψ, which lives in a 4-qubit space, and the right term shows its MPS approximation.

We simulate the quantum circuit using MPS in the following way: We start with the trivial tensor corresponding to the initial state of the circuit $|0\rangle^n$. Then, we apply each layer sequentially to the MPS, on which single-qubit Pauli rotations are trivially applied. After each CNOT gate, we recalculate the tensors by singular value decompositions (SVD) [270]. We partition the system into left and right parts and apply SVD, truncate the number of singular values to the bond dimension χ_k, and the left unitary multiplied with the truncated singular value matrix forms the tensor $A^{[k]}_{i_k}$. After applying every layer in this way, we get an MPS approximation of the output state $|\psi_{\text{full}}\rangle$ of the PQC.

To assess the quality of the MPS approximation for systems that can efficiently be simulated with the full-state, we compute the fidelity between the full quantum state $|\psi_{\text{full}}\rangle$, obtained from an exact state vector simulation, and the MPS-approximated state $|\psi_{\text{MPS}}\rangle$:

$$F = |\,\langle\psi_{\text{full}}|\,|\psi_{\text{MPS}}\rangle\,|^2. \tag{5.23}$$

We evaluate this fidelity for various values of the maximum bond dimension χ. As shown in Fig. 5.4, the fidelity increases with χ, indicating improved approximation accuracy. For a higher number of layers, the PQC increases the entanglement across the qubits, which decreases the fidelity for fixed bond dimension. For sufficiently large χ ($\chi = 32$ for 10 qubits), the MPS becomes numerically indistinguishable from the full-state [271].

In general, for a higher number of layers and qubits, it is not possible anymore to calculate this fidelity as it is not feasible to simulate the full-state. Instead, one can calculate the fidelity between MPS simulations of different bond dimensions, and choose the lowest bond dimension at which this fidelity does not increase anymore.

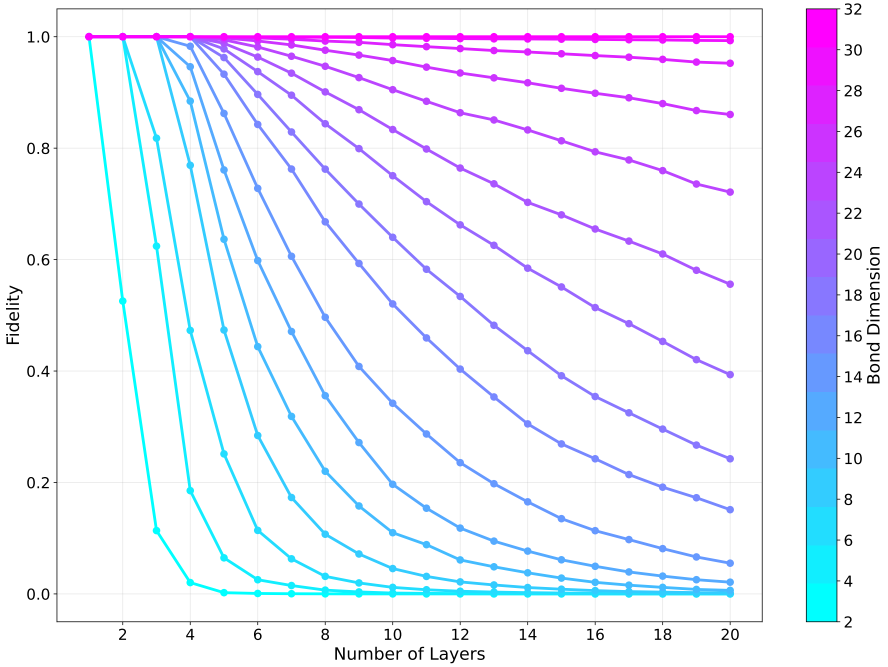


Figure 5.4: Fidelity between the exact quantum state prepared by a PQC consisting of 10 qubits, in the architecture as sketched in Fig. 5.2 and the MPS approximation as a function of the depth of the PQC, for different bond dimensions χ.

This tunability of the bond dimension provides a practical trade-off between simulation efficiency and accuracy. In the context of training the QGAN, where the PQC must be evaluated many times, moderate bond dimensions already yield sufficiently high fidelity while significantly reducing computational cost.

5.4 Results

In this section, we describe the results of our simulations. We simulated the parameterized quantum circuit (PQC) with the architecture shown in Fig. 5.2 in two ways, as described in Sec. 5.3. First, we performed full-state simulations for systems with up to 10 qubits and up to 6 layers. Second, we used matrix product state (MPS) simulations for systems with 10 and 20 qubits and between 1 and 18 layers. We performed 5 runs for each of these simulations, and describe their results in separate subsections. The results are discussed in Sec. 5.5.

5.4.1 Full-state simulation

For the full-state simulation, we chose a PQC consisting of 10 qubits and 8 layers. Therefore, for the generation of the training set from the historical S&P 500 time series, we set the window size to 20. In 5 runs, we trained a QGAN for 7900 epochs, and plotted the metrics of the generated time series of the best out of these 5 runs in Fig. 5.5. For this figure (and also for Figs. 5.8, 5.9, 5.11 and 5.12), we generated 1500 samples (each with different random noise inputs) of the time series, and calculate the correlations as the average of all samples and of all pairs of lags in each time series. The confidence intervals are calculated as in [272, p.51].

In **(a)**, we plot the probability density functions and in **(b)** the quantile-quantile plot of both the S&P 500 index and the generated time series. In **(c)**-**(h)**, we plot the metrics absolute autocorrelation, linear autocorrelation and the leverage effect, as an indication of the stylized facts as described in Sec. 5.2.1. The Subfigures **(c)**, **(e)** and **(g)** show the metrics of the S&P 500 index and the Subfigures **(d)**, **(f)** and **(h)** the metrics of the generated time series, respectively. Confidence intervals are calculated as in [272].

The generated time series closely resembles the distribution of the S&P 500 index, as shown in Subfigures **(a)** and **(b)**. Similar to the S&P 500 index, the generated time series shows a weaker, but decaying absolute autocorrelation (Subfigures **(c)** and **(d)**) and does not show linear autocorrelation (Subfigures **(e)** and **(f)**). The leverage effect, which is negative and increasing in the S&P 500 index (Subfigure **(g)**, is also reproduced in a weaker way in the generated time series (Subfigure **(h)**). As can be seen in Fig. 5.6, both the loss function and the temporal metrics decrease with the number of epochs, indicating stable training of the QGAN.

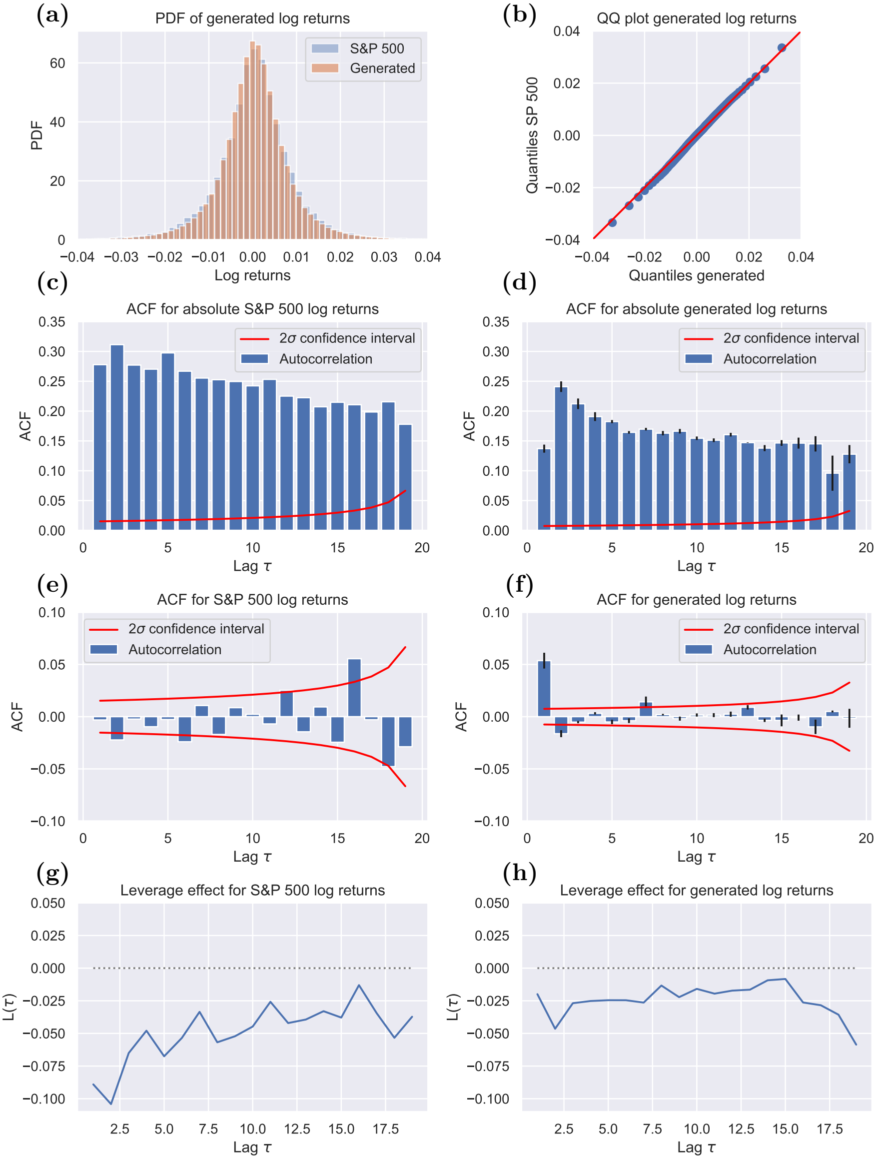


Figure 5.5: Metrics of the stylized facts for a synthetic time series of window size 20 generated by a QGAN, compared to the metrics of the S&P 500 index. The generator of the QGAN is a PQC based on the architecture shown in Fig. 5.2 consisting of 10 qubits and 8 layers, simulated with the full-state approach.

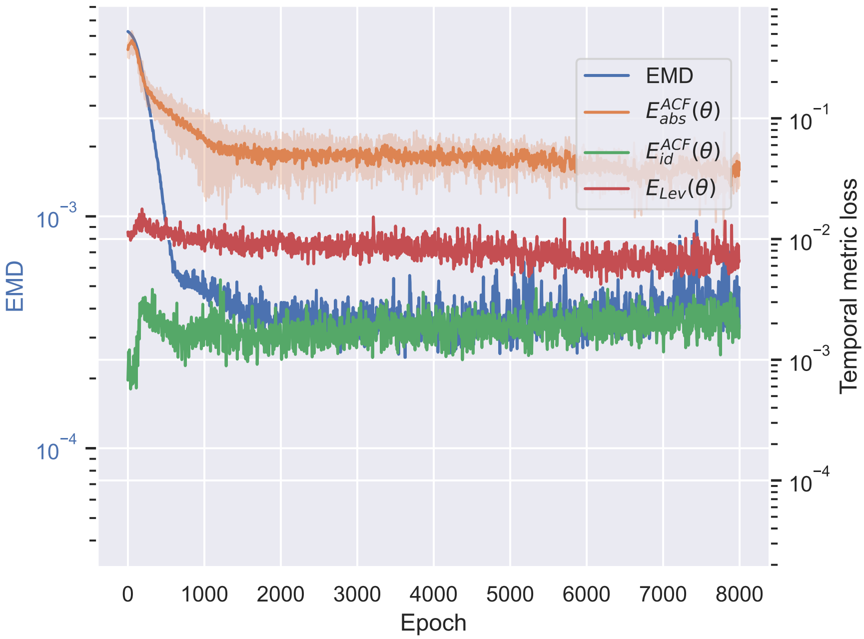


Figure 5.6: Wasserstein loss as defined in Eq.(5.9) (here called the EMD) and metrics corresponding to the temporal correlations as described in Sec. 5.2.1 in the training of the QGAN in the full-state simulation with 10 qubits and 8 layers (see Fig. 5.5 for the final metrics), depending on the number of epochs. We show the mean and standard deviation of the 5 training runs.

We show the metrics of the best out of 5 runs that correspond to the time series as shown in Fig. 5.5 in column **(a)** of Table 5.1.

Metrics	**(a)**	**(b)**	**(c)**	**(d)**	**(e)**
EMD	$3.3 \cdot 10^{-4}$	$6.1 \cdot 10^{-4}$	$2.7 \cdot 10^{-4}$	$2.9 \cdot 10^{-4}$	$4.2 \cdot 10^{-3}$
$E_{id}^{ACF}(\theta)$	$1.9 \cdot 10^{-3}$	$2.4 \cdot 10^{-3}$	$1.5 \cdot 10^{-3}$	$4.5 \cdot 10^{-4}$	$1.1 \cdot 10^{-3}$
$E_{abs}^{ACF}(\theta)$	$3.7 \cdot 10^{-2}$	$5.5 \cdot 10^{-2}$	0.15	0.31	0.99
$E_{Lev}(\theta)$	$6.6 \cdot 10^{-3}$	$5.8 \cdot 10^{-3}$	$4.7 \cdot 10^{-3}$	$2.2 \cdot 10^{-2}$	$4.4 \cdot 10^{-2}$

Table 5.1: Comparison of different metrics defined in Eqs. (5.3)-(5.6) for the best out of the 5 runs of **(a)** the full-state simulation with 10 qubits and 8 layers as in Figs. 5.5 and 5.6, **(b)** the full-state simulation with 10 qubits and 8 layers with a different circuit architecture as described in Appendix 5.C, **(c)** the full-state simulation with 10 qubits and 8 layers based on a circuit with CZ gates instead of CNOT gates as described in Appendix 5.D, **(d)** the MPS simulation with 10 qubits, 18 layers and a bond dimension 32 as in Figs. 5.8 and 5.7, **(e)** the MPS simulation with 20 qubits, 6 layers and a bond dimension 70 as in Fig. 5.9.

The Wasserstein QGAN does not explicitly account for temporal effects, so any such structure in the generated time series must result from other aspects of the model. To investigate the influence of the PQC architecture on these temporal effects, we trained a QGAN with a different PQC and present the results in Appendix 5.C. We indeed see that the absolute autocorrelation of the generated time series increases at larger time lags, and the leverage effect is less pronounced in comparison to the time series generated in Fig. 5.5. In contrast, no substantial difference in the quantile-quantile plots and the absolute autocorrelation can be observed. See Table 5.1 for a comparison of the metrics, which are higher than for the simulation shown in Fig. 5.5 apart from the leverage effect.

We also trained a QGAN with the full-state simulation based on a circuit that uses control-Z (CZ) gates instead of CNOT gates, and show the results in Appendix 5.D. The absolute autocorrelation decreases faster, and the leverage effect is more closely pronounced compared with the results shown in Fig. 5.5.

Additionally, to compare with the results of a GAN based on a quantum circuit Born machine [223], we trained the QGAN (based on the circuit with CZ gates instead of CNOT gates) on generating currency pairs; the results are shown in Appendix 5.B.

We analyze these results in Sec. 5.5.

As explained in Sec. 5.3, full-state simulation of PQCs quickly becomes infeasible as the number of layers and qubits increases. In the following, we describe MPS-based simulations, which make it feasible to simulate PQCs with larger numbers of layers and qubits.

5.4.2 MPS simulation

For the MPS simulation, we first chose a PQC of 10 qubits, with a varying number of layers (1, 5, 10 and 18) and different bond dimensions (1, 8, 16, 24 and 32) of the MPS. The MPS simulation of PQCs with 10 qubits generates time series with the same window size as the full-state simulation, making the results directly comparable. For higher numbers of layers and bond dimensions below $\chi = 32$, the MPS simulation is also faster than the full-state simulation.

For higher bond dimensions and number of layers, the training time in the MPS simulation increases, and the stylized facts of the generated time series vary considerably for each choice of the number of layers and bond dimension. See in Appendix 5.E for a comparison of the Wasserstein distance and metrics for the temporal effects for simulations of different numbers of layers and bond dimensions. In Fig. 5.8, we show the metrics of a generated time series from a well-performing QGAN that is trained for 7032 epochs, whose PQC consists of 18 layers and is simulated as an MPS with bond dimension 32. The metrics of this generated time series are shown in column **(c)** of Table 5.1. We chose to show the results for this particular model, as they match the stylized facts of the time series of the S&P 500 index qualitatively well, and as it proves that it is possible to train a QGAN for which the PQCs in the MPS simulation has more layers than what would be feasible with the full-state simulation.

The quantile-quantile plot shows that the generated time series matches the distribution of the S&P 500 index closely. In contrast to the time series generated with the full-state simulation shown in Fig. 5.5, the absolute autocorrelation (Subfigure **(d)**) that indicates volatility clustering is lower, but also decreasing for all time lags. Also the leverage effect is weaker than in the time series generated by the full-state simulation. The quantitative metrics decrease more slowly during training compared to the full-state simulation, as can be seen in Fig. 5.7.

Across the QGANs trained with different numbers of layers and bond dimensions in the MPS simulation, we generally observe that the generated time series reproduces the distribution, absence of linear autocorrelation, and volatility clustering, while the leverage effect is less pronounced.

In order to show that MPS can also be used for simulating QGANs that can generate time series with a larger window, we trained a QGAN with the MPS simulation of a PQC that consists of 20 qubits. Such a simulation would be infeasible with full-state simulation. We show the results of this simulation in Fig. 5.9 and in column **(d)** of Table 5.1. Since increasing the number of qubits and the bond dimension raises the time required to train each epoch, the QGAN is trained for only 650 epochs.

In the following section, we will analyze and compare the results of the different simulations shown here.

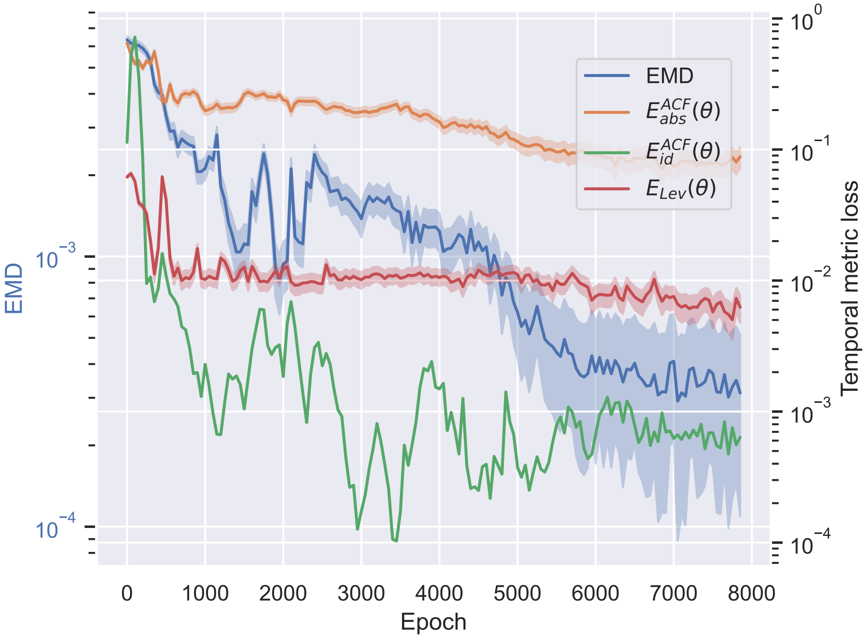


Figure 5.7: Wasserstein loss as defined in Eq. (5.9) (here called the EMD) and metrics corresponding to the temporal correlations as described in Sec. 5.2.1 in the training of the QGAN in the MPS simulation with 10 qubits, 18 layers and a bond dimension of 32 (see Fig. 5.8 for the final metrics), depending on the number of epochs. We show the mean and standard deviation of the 5 training runs.

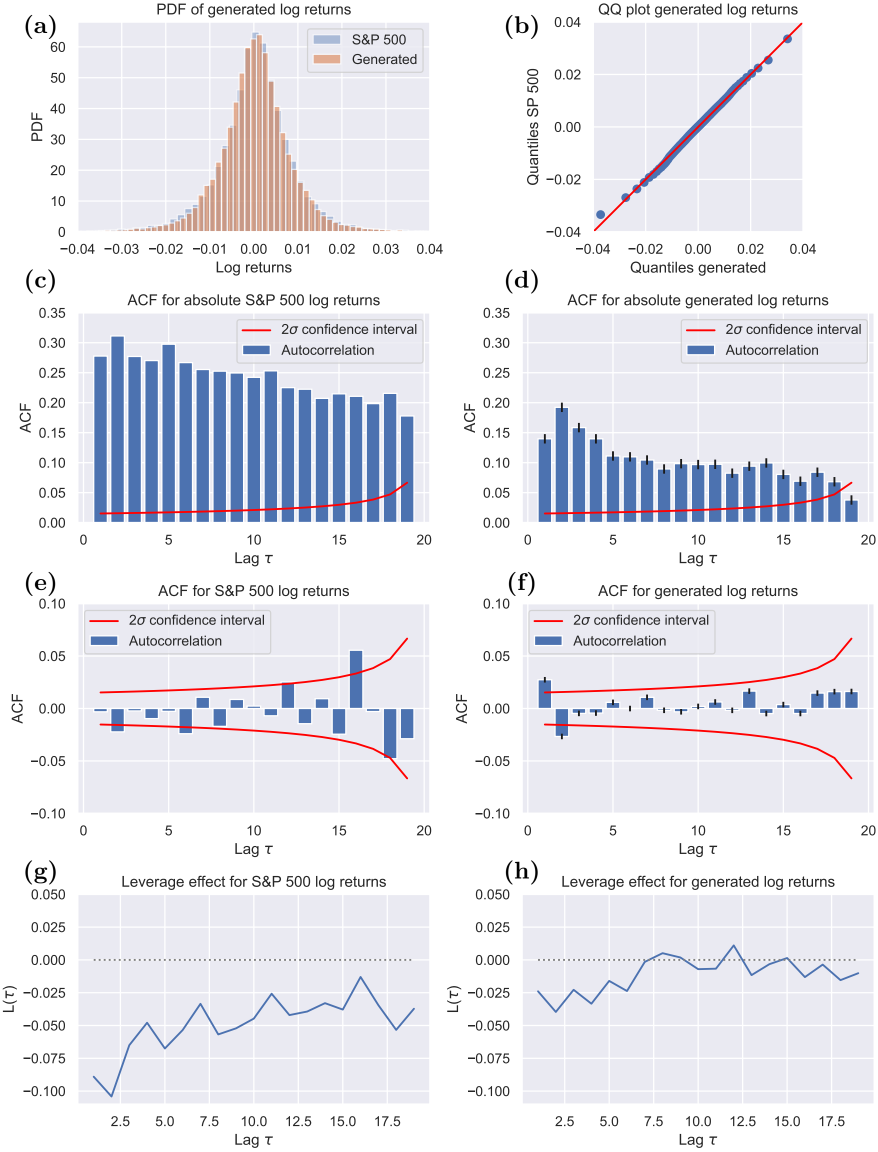


Figure 5.8: Metrics of the stylized facts for a synthetic time series of window size 20 generated by a QGAN, compared to the metrics of the S&P 500 index. The generator of the QGAN is a PQC consisting of 10 qubits and 18 layers, simulated with the MPS approach with bond dimension 32.

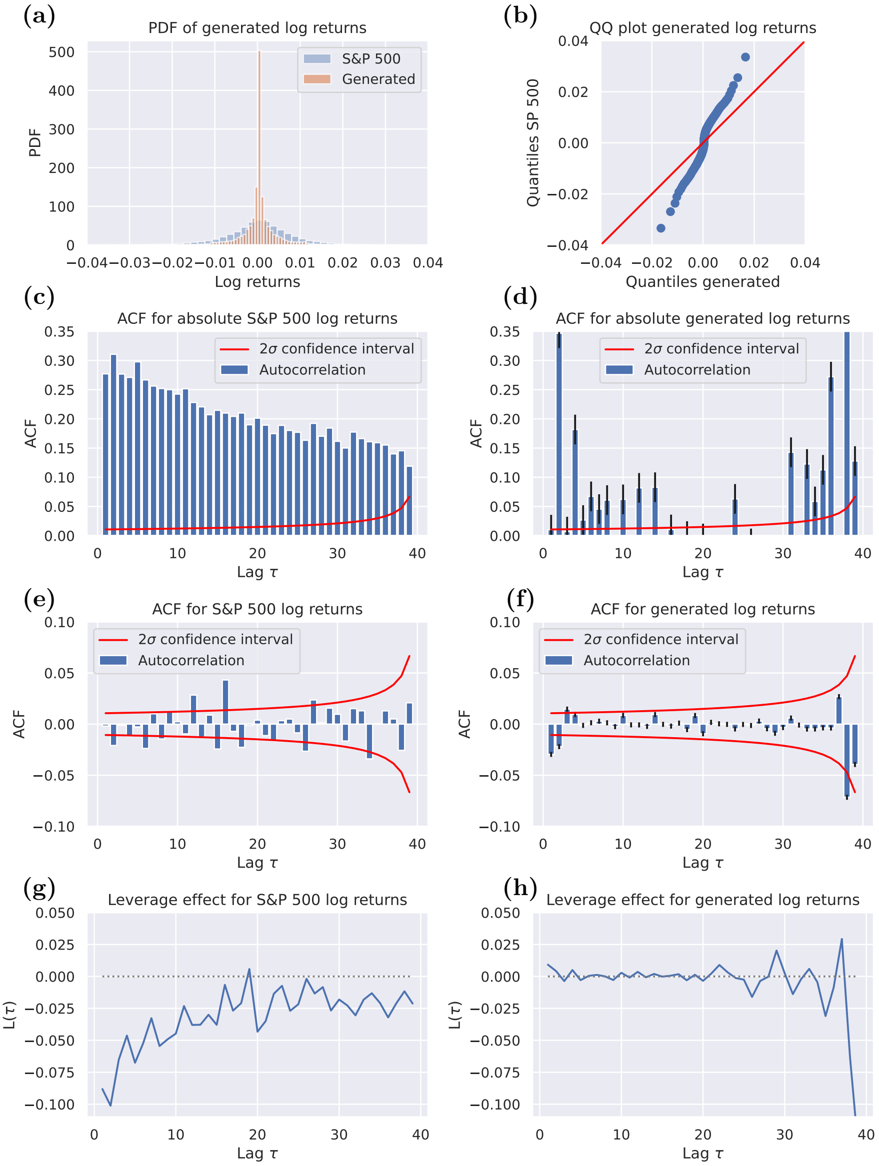


Figure 5.9: Metrics of the stylized facts for a synthetic time series of window size 40 generated by a QGAN, compared to the metrics of the S&P 500 index. The generator of the QGAN is a PQC consisting of 20 qubits and 7 layers, simulated with then MPS approach with bond dimension 70.

5.5 Analysis of the results

In all simulations, the probability distributions of the generated time series closely resembles the distribution of the S&P 500 index. The temporal correlations show significant differences between the simulations. While the absence of linear autocorrelations is visible in all simulated time series, their absolute autocorrelation (indicating volatility clustering) and the leverage are of different quality.

The full-state simulation (see Fig. 5.5) shows both effects, even though the absolute autocorrelation and the leverage effect are weaker than in the S&P 500 index.

The MPS simulation with 10 qubits, 18 layers, and bond dimension 32 (see Fig. 5.8) shows even weaker absolute autocorrelation and leverage effect.

Adding a CNOT gate between the first and the last qubits in each layer and performing the full-state simulation, leads to an increase in the qubit correlation (see Appendix 5.C). This might be a reason for the observation that the absolute autocorrelation of the generated time series increases at larger time lags. However, the leverage effect is pronounced weaker, and the other stylized facts do not differ substantially compared to the simulation shown in Fig. 5.5. Furthermore, in particular the Wasserstein loss is higher, as shown in Table 5.1, showing that the model generates time series that are further away from the real probability distribution. This proves that the architecture of the circuit indeed plays an important role on the quality of the generated time series.

The QGAN simulated with the full-state simulation which is based on a circuit that uses control-Z (CZ) gates instead of CNOT gates in Appendix 5.D, shows a faster decreasing absolute autocorrelation but more clearly pronounced leverage effect.

We benchmarked the QGAN with a full-state simulation against a quantum circuit Born machine in modeling the time-aggregated distribution of foreign exchange pairs yielding a better approximation of those distributions (see Appendix 5.B).

Using the MPS simulation, we also trained a QGAN with 20 qubits, 5 layers, and a bond dimension of 70 (see Fig. 5.9). This demonstrates that MPS can handle QGANs of greater complexity than those feasible with full-state simulation. However, the training of QGANs with PQCs of a higher number of layers and qubits and MPS of higher bond dimensions increases the number of epochs needed in the training. Additionally, each training epoch takes a longer time for these more complex models. For an equal computational cost, the generated time series therefore does not resemble the distributions and temporal effects of the target time series as closely as in the simulations with 10 qubits. But, by using a PQC with 20 qubits, it is possible to simulate time series with a larger window size of 40.

We remark that the loss landscapes differ significantly between full-state and

MPS simulations due to their different approximation and simulation structure. The difference in the quality of the generated time series can be partially attributed to the different features of the loss landscape.

Compared to the classical GAN experiments in [260], which use multi-layer dense neural networks as generators and either a multi-layer dense or convolutional neural network as discriminator (with the same specifications as described in Appendix 5.A), both of our quantum simulation methods yield qualitatively improved results, particularly with respect to the Wasserstein distance and volatility clustering, as observed in the plots of the stylized facts. Note that the window size used in the classical experiments differs from ours, which may influence the comparison.

5.6 Conclusions

We constructed a Wasserstein quantum generative adversarial network (QGAN) with a classical convolutional network as a discriminator and an expectation value sampler based on a parameterized quantum circuit (PQC) as a generator, in order to assess whether these quantum architectures have suitable inductive biases for generating synthetic financial time series known to be problematic for classical models. This approach leverages the PQC architecture to intrinsically capture temporal correlations in the time series, while the QGANs are trained solely on matching the aggregated distribution of the time series, by using a discriminator which learns the Wasserstein distance between the distributions of the generated time series and of the training data. We simulated a PQC with 10 qubits and 8 layers with a full-state simulation and a PQC with 10 and 20 qubits and with up to 18 layers as an approximation by a matrix product state (MPS) simulations with bond dimensions of up to 70. The latter approach allowed us to simulate PQCs with a higher number of layers and qubits, which makes it possible to train the generation of longer time series.

We compare the generated time series qualitatively with the S&P 500 index by their distributions and their temporal correlations, also called the stylized facts. These stylized facts are typically assessed qualitatively rather than quantitatively [78].

In this chapter, we showed that our trained QGANs generate time series that match the desired distributions and exhibit some of the temporal correlations seen in financial time series, such as in the S&P 500 index. Simulating the PQC with full-state simulations and MPS simulations yield different results, with circuit depth and the MPS bond dimension further influencing the performance. The three simulations performed with the full-state simulation show different behavior in particular of the absolute autocorrelation of the generated time series, indicating different qualities in capturing volatility clustering. The QGAN using the PQC given in Fig. 5.2 shows the closest match of this property (see

Fig. 5.5), whereas PQC architectures where an additional CNOT gate is added at the end of each layer leads to an increase in the absolute autocorrelation for higher time lags (see Fig. 5.11). Using CZ gates instead of CNOT gates in the PQC causes a quicker decrease of this effect (see Fig. 5.12). The MPS approach leads to weaker absolute autocorrelation and leverage effect compared to the full-state simulation (compare Fig. 5.5 and 5.8), but is able to simulate QGANs with a longer time window (see Fig. 5.9). Both simulation methods motivate the study of quantum hardware in their ability to generate financial time series with stylized facts. Our work has already motivated studies in which the effect of such generated data on the training of neural networks has been explored [273, 274].

The application of these QGANs as subroutines for applications such as option pricing [275] and risk analysis [276] can be explored as well. Furthermore, a possible extension of our method is to train the model to replicate correlated stocks of the S&P 500 index, motivated by research in community detection [277]. This could possibly be achieved by either learning the underlying distributions (in a similar way as done in Appendix 5.B), or by learning the individual time series similar to the ones in Sec. 5.4. As the number of qubits restricts the number of time steps and the number of stocks that can be generated, one could examine if quantum generators consisting of circuits on qudits can be successful, as that enables more independent measurements on each qudit. Specifically for qudits, not only superconducting qubits form a suitable experimental platform, but also trapped ions, neutral atoms and integrated photonics are excellent candidates for manipulating higher-dimensional quantum information [278, 279].

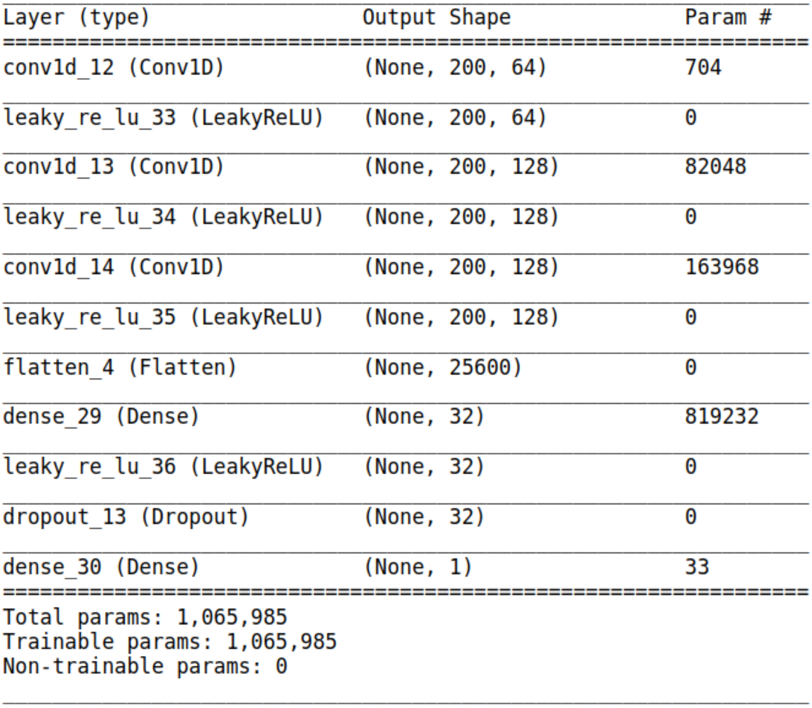
An improvement of the training of the QGANs could be achieved in several ways. Firstly, the effects of shot noise [237] in the training of the quantum generator could be explored. Secondly, different design choices, like choosing a different classical or quantum discriminator in the QGAN, diffusion model [242], or quantum long-short time memory models [280] might lead to different results. Thirdly, as the QGAN is trained with Wasserstein loss functions (see Eqs. (5.9) and (5.11)) that are taking the distribution of the time series into account, but not the temporal effects, an adaption of the training to consider them as well might lead to a better recovery of those temporal effects. In particular, it might be possible to not only gain a better match in the absolute autocorrelation and leverage effect, but also in the exact reproduction of the autocorrelation. Lastly, one could try different definitions of the quantum Wasserstein distance [281] that give theoretical improvements over the qualitative accuracy.

5.7 Code availability

The code supporting this chapter is available at the following repository: `https://github.com/LucasAugustusvd/Quantum-Finance`

5.A Architecture of the discriminator

We trained the classical discriminator in our QGAN simulations with a convolutional neural network. Table 5.2 summarizes its properties and hyperparameters. This choice is motivated by [239], where it successfully was applied as a discriminator of a GAN that generates financial time series.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_12 (Conv1D)           (None, 200, 64)           704
_________________________________________________________________
leaky_re_lu_33 (LeakyReLU)   (None, 200, 64)           0
_________________________________________________________________
conv1d_13 (Conv1D)           (None, 200, 128)          82048
_________________________________________________________________
leaky_re_lu_34 (LeakyReLU)   (None, 200, 128)          0
_________________________________________________________________
conv1d_14 (Conv1D)           (None, 200, 128)          163968
_________________________________________________________________
leaky_re_lu_35 (LeakyReLU)   (None, 200, 128)          0
_________________________________________________________________
flatten_4 (Flatten)          (None, 25600)             0
_________________________________________________________________
dense_29 (Dense)             (None, 32)                819232
_________________________________________________________________
leaky_re_lu_36 (LeakyReLU)   (None, 32)                0
_________________________________________________________________
dropout_13 (Dropout)         (None, 32)                0
_________________________________________________________________
dense_30 (Dense)             (None, 1)                 33
=================================================================
Total params: 1,065,985
Trainable params: 1,065,985
Non-trainable params: 0
_________________________________________________________________
```

Table 5.2: Hyperparameters and properties of the convolutional neural network used as the discriminator in the QGANs.

5.B Comparison with quantum circuit Born machine

In [223], a QGAN is constructed where the quantum generator was used as a quantum circuit Born machine. It was trained to generate distributions of foreign exchange pairs, producing samples that better matched the true distributions than those from a classical restricted Boltzmann machine with a comparable model size.

We also trained our QGAN, where the quantum circuit consisting of 4 qubits and 4 layers, is simulated with the full-state approach with CZ gates (instead of CNOT gates compared to Fig. 5.2), in reproducing the same pairs of foreign exchanges as in [223]. We trained the single-qubit Pauli-X and Pauli-Z observables on the distributions of the EUR/USD and the GBP/USD foreign exchange log returns, respectively. Fig. 5.10 shows the quantile-quantile plot comparing samples from our trained model with the target distribution. Our trained QGAN samples match the target distribution more closely than the results for the quantum circuit Born machine and the classical restricted Boltzmann machine shown in Fig. 10 of [223], while using fewer qubits than used for the quantum circuit Born machine. This difference to the results from the quantum circuit Born machine comes from to the discrete nature of that model, which has naturally a higher imprecision of generated samples compared to the expectation value sampler used in our model.

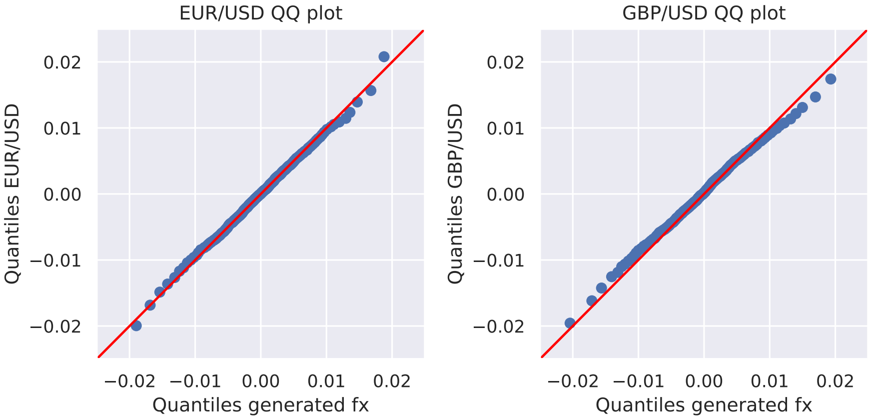


Figure 5.10: Quantile-quantile plot comparing samples from the trained QGAN model with a PQC of 4 qubits and 4 layers to the target distribution of EUR/USD and GBP/USD log returns.

5.C Full-state simulation: alternative circuit architecture

In addition to the PQC shown in Fig. 5.2, we trained a QGAN using a modified PQC architecture simulated with the full-state approach. In order to increase long-range qubit correlations, we added a CNOT gate between the first and 10th qubit in each layer of the PQC (results in Fig. 5.11). Subfigure **(d)** shows that this architectural change increases the absolute autocorrelation at larger time lags. The metrics of the generated time series are shown in column **(b)** of Table 5.1.

5.D Full-state simulation: CZ gates instead of CNOT gates

In Fig. 5.12 , we show the results of a full-state simulation using a circuit architecture in which the CNOT gates were substituted with control-Z (CZ) gates. Compare with the architecture sketched in Fig. 5.2 and the corresponding simulations shown in Figs. 5.5 and 5.6. Subfigure **(d)** shows that this architectural change leads to a faster decrease in the absolute autocorrelation. The metrics of the generated time series are shown in column **(c)** of Table 5.1.

5.E MPS simulations for different numbers of layers and bond dimensions

In Fig. 5.13, we show the quantitative metrics of training a QGAN where the PQC consisting of 10 qubits are simulated with the MPS approach for $1, 5, 10$ and 18 layers and bond dimensions of $1, 8, 16, 24$ and 32. See Sec. 5.4.2. Note that a bond dimension of 32 is giving an exact MPS approximation of the 10-qubit state.

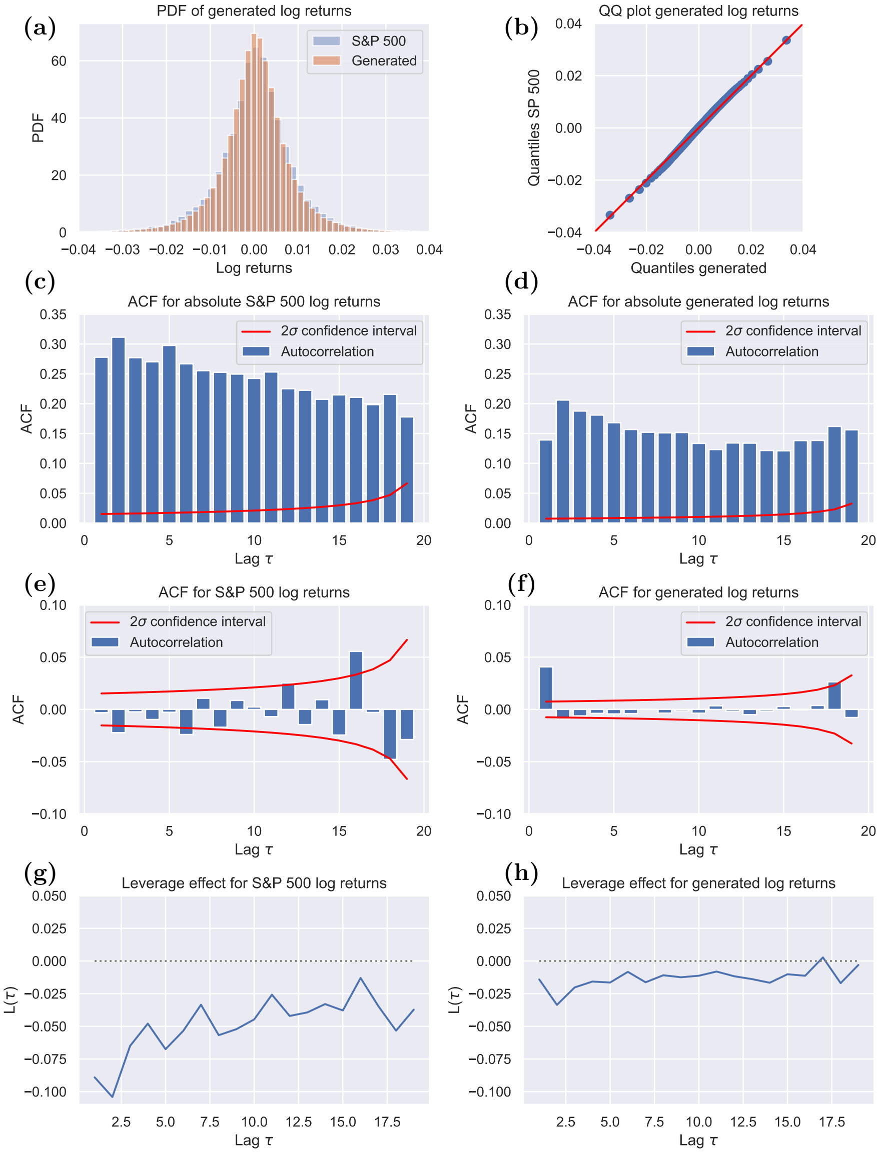


Figure 5.11: Metrics of the stylized facts for a synthetic time series of window size 20 generated by a QGAN, compared to the metrics of the S&P 500 index. The generator of the QGAN is a PQC consisting of 10 qubits and 8 layers, simulated with the full-state approach. Contrary to the PQC used in Fig. 5.5, we added an additional CNOT gate between the first and the 10th qubit in each layer.

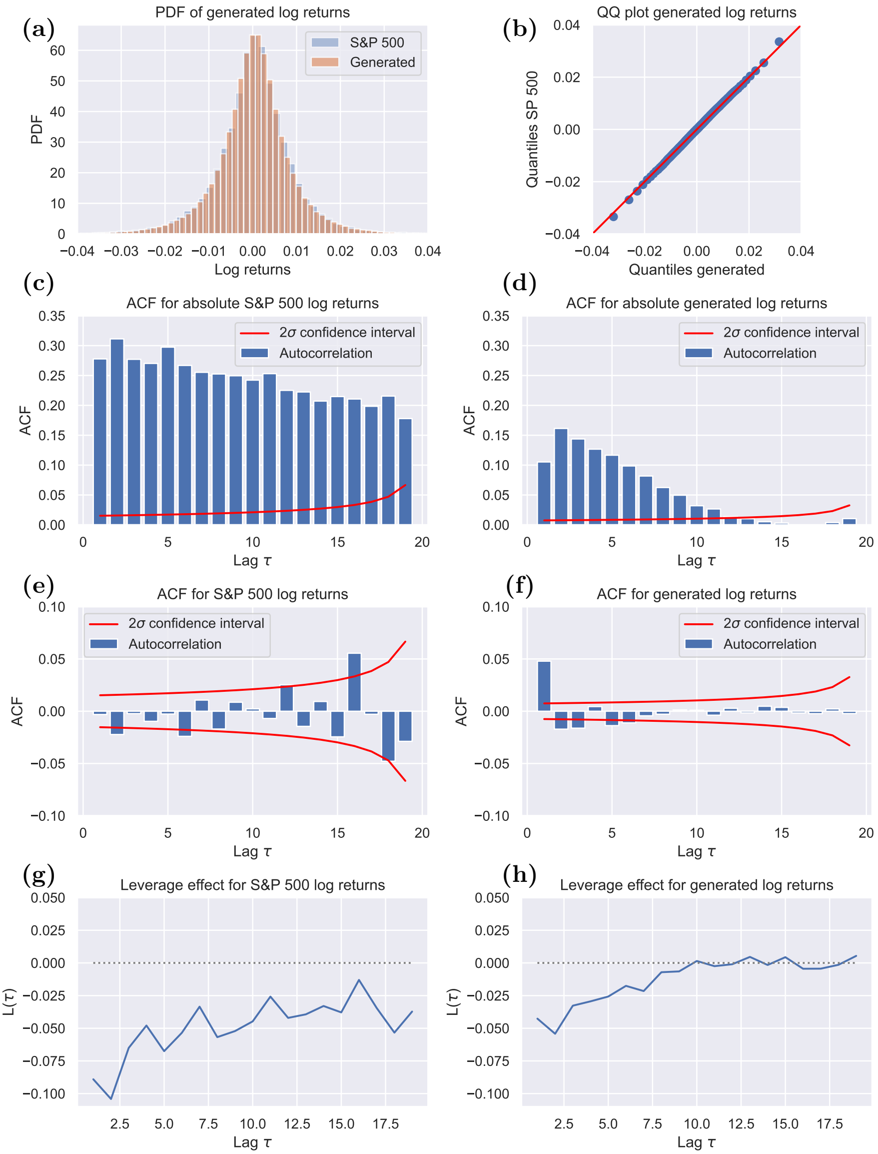


Figure 5.12: Metrics of the stylized facts for a synthetic time series of window size 20 generated by a QGAN, compared to the metrics of the S&P 500 index. The generator of the QGAN is a PQC based on the architecture shown in Fig. 5.2 consisting of 10 qubits and 8 layers and CZ gates instead of CNOT gates, simulated with the full-state approach.

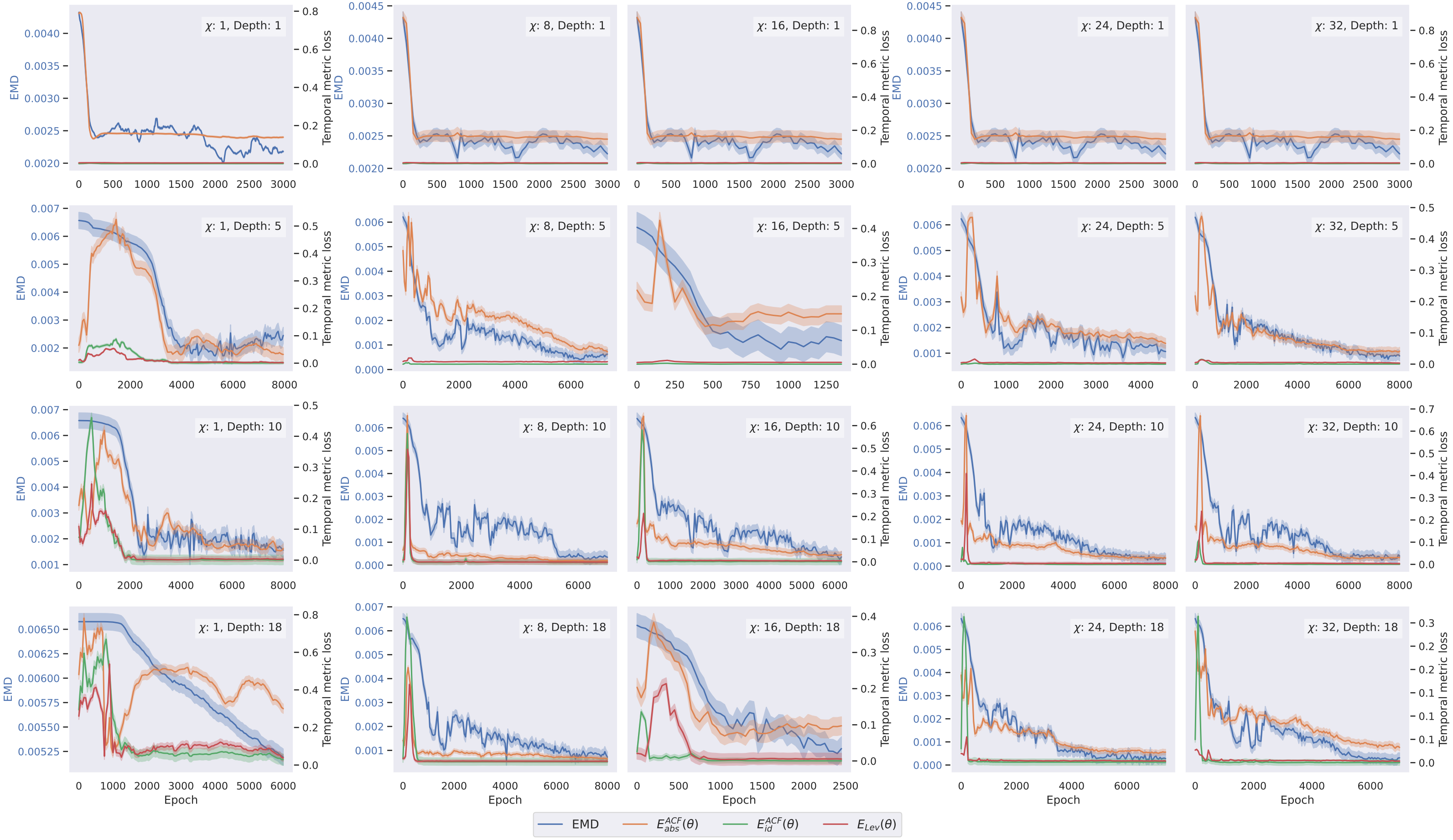


Figure 5.13: Wasserstein loss as defined in Eq. (5.9) (here called the EMD) and metrics corresponding to the temporal correlations as described in Sec. 5.2.1 in the training of the QGAN in the MPS simulation with 10 qubits, $1, 5, 10$ and 18 layers and bond dimensions χ of $1, 8, 16, 24$ and 32, depending on the number of epochs.