ANALYZING DEEP GENERATED FINANCIAL TIME SERIES FOR VARIOUS ASSET CLASSES *

Antonio Rosolia
Zurich University of Applied Sciences
School of Engineering
8401 Winterthur, Switzerland
antonio.rosolia@protonmail.ch

Jörg Osterrieder
Zurich University of Applied Sciences
School of Engineering
8401 Winterthur, Switzerland
oste@zhaw.ch

August 1, 2021

ABSTRACT

Generative Adversarial Networks (GANs) have shown remarkable success as a framework for training models to produce realistic-looking data. In this work, we propose a GAN to produce realistic real-valued time series, with an emphasis on their application to financial data.

Our aim is having a GAN, applied on various financial time series for various asset classes, that can reflect all the characteristics of them, as well as the characteristics we may be unaware of, as GANs learn the underlying structure of our data, rather than just a set of features. If we are able to achieve this, the synthetic datasets we create could be used for a variety of purposes including model training and model selection. In this paper we try to train a GAN with real data, representing one asset of different asset classes such as commodities, forex, futures, index and shares.

***K*eywords** GAN · Financial Data · Neural Network · deep learning

1 Introduction

Access to large amount of data is key in the development of machine learning and especially neural network solutions to specific problems. The availability of datasets has helped to implement various models in multiple tasks. However, progress appears to lag in some fields such as finance. It is tempting to suggest that tasks in finance are harder or that the environment in finance is slower in catching up with the newest technologies. Regardless of this, the dearth of data accessible to researchers hinders scientific progress. Due to the highly sensitive nature of finance data or the big cost involved with it, its access is highly controlled. The motivation for this work is therefore develop a generative adversial network (GAN) to generate realistic synthetic financial data for various financial instruments, such as bonds, stocks, options and Forex.

In the last few years, GAN have been introduced successfully mainly in the area of image generation. GANs are an exciting and rapidly changing field, delivering on the promise of generative models in their ability to generate realistic examples across a range of problem domains, most notably in image-to-image translation tasks such as translating photos of summer to winter or day to night, and in generating realistic photos of objects, scenes, and people that even humans cannot tell are fake.

*Financial support by the Swiss National Science Foundation within the project "Mathematics and Fintech - the next revolution in the digital transformation of the Finance industry" is gratefully acknowledged by the corresponding author. This research has also received funding from the European Union's Horizon 2020 research and innovation program FIN-TECH: A Financial supervision and Technology compliance training programme under the grant agreement No 825215 (Topic: ICT-35-2018, Type of action: CSA). Furthermore, this article is based upon work from COST Action 19130 Fintech and Artificial Intelligence in Finance, supported by COST (European Cooperation in Science and Technology), www.cost.eu (Action Chair: Joerg Osterrieder).
)

Generative modeling is an unsupervised learning task in machine learning that involves automatically discovering and learning the regularities or patterns in input data in such a way that the model can be used to generate or output new examples that plausibly could have been drawn from the original dataset. GANs are a clever way of training a generative model by framing the problem as a supervised learning problem with two sub-models: the generator model that we train to generate new examples, and the discriminator model that tries to classify examples as either real (from the domain) or fake (generated). The two models are trained together in a zero-sum game, adversarial, until the discriminator model is fooled about half the time, meaning the generator model is generating plausible examples (Goodfellow et al., 2014). Other important variant of GAN have been intoroduced tom improve the training process, namely the Wasserstein Generative Adversarial Network, or Wasserstein GAN (Arjovsky et al., 2017). It is an extension to the generative adversarial network that both improves the stability when training the model and provides a loss function that correlates with the quality of generated images. It is an important extension to the GAN model and requires a conceptual shift away from a discriminator that predicts the probability of a generated image being "real" and toward the idea of a critic model that scores the "realness" of a given dataset. This conceptual shift is motivated mathematically using the earth mover distance, or Wasserstein distance, to train the GAN that measures the distance between the data distribution observed in the training dataset and the distribution observed in the generated examples.

The paper is organized as follows: Section 2 provides a literature review of relevant papers and studies, that use GAN. Section 3 gives some background on neural networks and GANs. Section 4 describes the data and the GAN, that we will use in our experiments in the paper. Section 5 gives an empirical data analysis of the data we will use in our experiment. The design and execution of experiments with GAN will be shown in section 5. The results will be described in Section 6. Finally, Section 7 discusses the impact of this research and potential future applications.

2 Literature Review

The main application in which GANs have been proven to work well has been image generation, in which starting with simple MNIST numbers and going all the way to bedroom pictures (Radford et al., 2016) and celebrity photos ((Diamant et al., 2019), (Karras et al., 2018)), many advancements have been developed to generate synthetic samples. Having a way to artificially create complex and rich artificial datasets can be a very useful tool when training other kinds of models, albeit because the original data is scarce, protected by privacy regulations or simply too expensive to recollect. GAN's success with images has been theorized to be due to the strong spatial relationships of the inputs (meaning pixel values in photographs) and an easy measure of performance, simple visual inspection. Applying GANs to time series data has also been explored, because of the lack of access to medical data due to privacy regulations in the case of (Esteban et al., 2017). The main challenge data generation faces is the possible lack of structure in the input data, but in our case, financial time series have been recognized to share some common traits, such as having heavy tails in the distribution of returns, volatility clustering, mean reversion and momentum (see (Davies and Krämer, 2016), (Malmsten and Teräsvirta, 2004)) so we are not completely hopeless. A new generative model for time series has been introduced s based on Euler discretization that do not require any pre-stationarization procedure and relies on the adaptation of Wasserstein GAN (Arjovsky et al., 2017) and DVD GAN (Clark et al., 2019) to time series. Moreover it has been demonstrated, that GANs can in fact generate high-fidelity and locally-coherent audio by modeling log magnitudes and instantaneous frequencies with sufficient frequency resolution in the spectral domain and thus outperforming WaveNet baselines on automated and human evaluation metrics, and efficiently generate audio several orders of magnitude faster than their autoregressive counterparts (Engel et al., 2019).

In newer research GANs have been applied to quantitative finance such as, generation of financial time series (Wiese et al., 2020) to enrich the available data of an asset, implement them in trading strategies (Miot, 2020) and to portfolio risk minimization problem by adding a regression network to GAN (Pun et al., 2020a). It has been explored the use of machine learning models to build a market generator. The underlying idea is to simulate artificial multi-dimensional financial time series, whose statistical properties are the same as those observed in the financial markets. In particular, these synthetic data must preserve the probability distribution of asset returns, the stochastic dependence between the different assets and the autocorrelation across time. The article proposes then a new approach for estimating the probability distribution of backtest statistics. The final objective is to develop a framework for improving the risk management of quantitative investment strategies, in particular in the space of smart beta, factor investing and alternative risk premia(Lezmi et al., 2020). Another interesting idea featuring GANs has ben shown in Pagan, Portfolio Analysis with Generative Adversarial Networks. The goal is modeling the market uncertainty that ultimately is the main factor driving future trends. The generative model learns the joint probability distribution of price trends for a set of financial assets to match the probability distribution of the real market. Once the model is trained, a portfolio is optimized by deciding the best diversification to minimize the risk and maximize the expected returns observed over the execution of several simulations. Applying the model for analyzing possible futures, is as simple as executing a Monte Carlo simulation, a technique very familiar to finance experts. The experimental results on different portfolios representing

different geopolitical areas and industrial segments constructed using real-world public data sets demonstrate promising results.(Mariani et al., 2019a). Something similiar has been tried out with a Wasserstein GAN for asset allocation in emergin Market space. It succeed at identifying interesting future return opportunities and constructing portfolios accordingly. The methodology yields superior portfolios that outperform the local index and the 1 N -portfolio. The approach entails the Wasserstein Generative Adversarial Network to predict and the Non-Dominated Sorting Genetic Algorithm II with Monte Carlo to forecast closing prices for emerging market assets and construct emerging market portfolios, respectively (202, 2020). Additionally it has been investigated if GANs can also be used to approximate one-dimensional Itô stochastic differential equations (SDEs). (van Rhijn et al., 2021). Existing GAN architectures do a poor job of matching the asymptotic behavior of heavy-tailed distributions, a problem that stems from their construction. That is quite a big problem, as heavy-tailed distributions are prevalent in many different domains such as financial risk-assessment, physics, and epidemiology. These problems were addressed with the Paerto GAN (Huster et al., 2021). A Pareto GAN leverages extreme value theory and the functional properties of neural networks to learn a distribution that matches the asymptotic behavior of the marginal distributions of the features. The issues with standard loss functions were identified and an alternative metric space has been proposed that enables stable and efficient learning.

3 Deep Learning and GAN

Here we give some background information on the how Deep Learning and GANs work.

3.1 Neural Network

Neural Networks are a powerful tool for learning complex non-linear relationships between data, which can be useful, especially within the realm of finance (Fischera and Krausb, 2017). A neural network is built by a network of neurons. A neuron is a logical unit that takes in a set of inputs, computes a linear combination of them using the corresponding edge weights, adds the bias and then applies its activation function and returns the output. Generally an activation function is nonlinear as this allows the network to potentially learn more complicated non-linear relationships(Inzaugarat, 20). One commonly used activation function is the rectified linear unit (RELU), which is given by $g(x) = max(0, x)$.

The major challenges of deep learning methods arise from the task of choosing the "best" model architecture. Facing the lack of computing power to test all possible model structures on any given data set, it is crucial to rely on previous research, data set characteristics, and intuition to design a deep learning model. David Wolpert, Mathematician and Santa Fe Institute Professor, describes the machine learning "no free lunch theorem" as follows: "for any two learning algorithms A and B... there are just as many situations (appropriately weighted) in which algorithm A is superior to algorithm B as vice versa." (Wolpert and Macready, 1997). It follows that there is no universal model structure or learning algorithm, meaning different model structures give more accurate results on different data sets and for different purposes. There is also no universal guide on how to design a model, so intuition and experience are imperative in model design. Further complexities arise from underfitting and overfitting problems and from the task of how to efficiently train a neural network.

Following the proposed structure in (Géron, 2017), we describe the most important aspects of our network. Those are Initialization, Activation function, Normalization, Regularization, Optimizer and the learning rate schedule.

Data Scaling Data scaling or normalization is a process of making model data in a standard format so that the training is improved, accurate, and faster. Additionally, it's useful to ensure that our inputs are roughly in the range of -1 to 1 to avoid weird mathematical artifacts associated with floating point number precision. In short, computers lose accuracy when performing math operations on really large or really small numbers. Moreover, if your inputs and target outputs are on a completely different scale than the typical -1 to 1 range, the default parameters for a neural network will likely be ill-suited for your data. It's a common practice to scale your data inputs to have zero mean and unit variance. This is known as the standard scaler approach.(J.Jordan, 2020)

Optimizer As an optimizer, we are using the Adaptive Moment Estimation (ADAM), which is a method that computes adaptive learning rates for each parameter. In addition to storing an exponentially decaying average of past squared gradients, ADAM also keeps an exponentially decaying average of past gradients:

$$m_t = \beta_1 m_t - 1 + (1 - \beta_1) g_t \tag{1}$$

$$v_t = \beta_2 v_t - 1 + (1 - \beta_2) g_t^2 \tag{2}$$

m_t and v_t are estimates of the mean and the uncentered variance of the gradients. g_t denotes the gradient, i.e. the vector of partial derivatives of f_t evaluated at timestep t. β_1 and β_2 are hyper-parameters that control the exponential decay rates.

Since m_t and v_t are initialized as vectors of 0's they are biased towards zero. To counteract these biases, bias-corrected estimates are computed and used to update the parameters θ with the following ADAM update rule:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t \tag{3}$$

Summarizing, the benefits of ADAM consist of an adaptive learning rate and momentum for each parameter, as well as a non-diminishing learning rate. On the downside, it does not have the ability to "look ahead" before taking the next step like other optimizers, which include an approximation of θ_{t+1} in the calculation.

3.2 GAN

Generative adversarial networks take up a game-theoretic approach, unlike a conventional neural network. The network learns to generate from a training distribution through a 2-player game. The generator G(z) takes random noise vector z as input with corresponding network weights. The discriminator D on the other side is a classifier with a containing loss function. The discriminator has to determine if the sample is real data x or generated data G(z). Depending on whether the discriminator has evaluated the input correctly or wrong it gets an update on the parameters of the generator. The generator and the discriminator can perform almost the same tasks. Nonetheless, there is a big difference in the math. The generative model describes how data is generated to learn any kind of data distribution by using supervised learning. The discriminator, however, models the decision boundary between the classes. Then it outputs the probability that x comes from the data rather than from the probability distribution p_g. So both models predict the conditional probability but from different probabilities. Two networks are connected: Generator G and Discriminator D.(Developers) Each function is differentiable with respect to its inputs and with respect to its own parameters. The overall value function V(D, G), is defined as:

$$E_x[\log(D(x))] + E_z[\log(1 - D(G(z)))] \tag{4}$$

D(x) is the discriminator's estimate of the probability that real data instance x is real. E_x is the expected value over all real data instances. $G(z)$ is the output of the generator when given noise z. D($G(z)$) is the discriminator's estimate of the probability that a fake instance is real. E_z is the expected value over all random inputs to the generator. In other words, it is the expected value over all generated fake instances $G(z)$. The discriminator is trained to maximize the probability of assigning the correct label to train the examples from the generator. The generator, however, is trained to minimize $log(1 - D(G(z)))$. The generator G cannot directly affect $log(D(x))$ in the function, so minimizing the loss is equivalent to minimizing $log(1 - D(G(z)))$. At the beginning of training, when G_z is performing not so well, and generating random noise, D(x) can reject samples from G(z) with great confidence. Both optimize their current weights according to their opposing loss function objective. The network consistently improves each other until the generator generates data that is indistinguishable from real data.(Google-Developers, 2020)

3.3 GAN Training

The generator part of a GAN learns to create fake data by incorporating feedback from the discriminator. Therefore the generator and the discriminator have different training processes. In the first step, the discriminator is fixed. This means that in this step, no adjustments to the parameters are made for the discriminator network. The generator is then trained for a certain number of training steps. The generator is trained using back-propagation. Its goal is that the "fake" outputs resulting from the random input are classified as real examples by the current discriminator. It is important to mention that the training progress depends on the current state of the discriminator. Due to the nature of the alternating training process, various problems can occur when training a GAN. One challenge is that the feedback of the discriminator gets worse, the better the generator is performing during the training. If the generator can generate examples that are indistinguishable from real examples, the discriminator has to guess from which class the example come from. The training has to be finished in time, otherwise the quality of the generator and the discriminator may decrease due to the random return values of the discriminator.(Google-Developers, 2020)

Training the generator During generator training, the generator learns to create fake samples by implementing the feedback from discriminator. Based on the discriminator output, the generator loss is calculated to obtain gradients and penalize the generator for failing to confuse the discriminator (although the discriminator does not update its weights during generator training). Using the gradients, the weights of generator are updated. The objective of generator is to generate fake samples that the discriminator classifies as original and predicts a probability of 0.5 as it cannot differentiate between fake or original samples. During generator training, the discriminator training is stopped and its weights remain fixed. The reason for this is to avoid the discriminator model from becoming extremely strong to

be beaten by the generator. If the discriminator training is kept on during generator training, the generator will never converge or be able to learn the distributions because then the discriminator will continue to improve and become an expert in identifying the fake samples.(Google-Developers, 2020)

Training the discriminator During the training of the discriminator, the generator is not trained. The weights and biases of the generator model are then kept constant while it continues to create samples for discriminator to train on. The discriminator is related to two loss functions, the discriminator loss and generator loss. During training of discriminator, the discriminator only uses the discriminator loss and ignores the generator loss. The generator loss is only accounted during the generator model training. When the discriminator is trained, it attempts to classify original data as 1 and 0 otherwise and predicts a probability value. Based on the discriminator loss, the discriminator is penalized for misclassification of the fake instances as original or original instances as fake. In this manner of backpropagation, the discriminator's weights are updated using the discriminator loss.Google-Developers (2020)

4 Experiment Setup

In this section we give a summary of some of the architectural choices and training settings that we will use in the Experiments section. Additionally we specify the input data used for this experiment.

4.1 Network architecture

The model is a simple GAN architecture with a discriminator, which can be seen in Figure 1 and a generator, which can be seen in Figure 2. The discriminator has an input layer followed by a two blocks of dense layer with a leaky ReLu. The strucutre for the Generator is the same, apart from the last layer. The last layer of the Discriminator has just one value, for classifying the data as real or fake. The noise, for the input of the model, comse from a noarmal distribution. For training we use Adam with learning rate 0.0001 and batch size of 32.

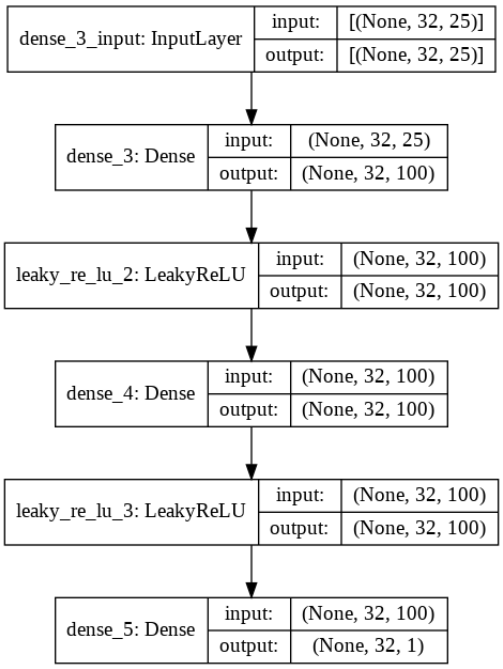

Figure 1: Plot of Discriminator architecture used in the experiments

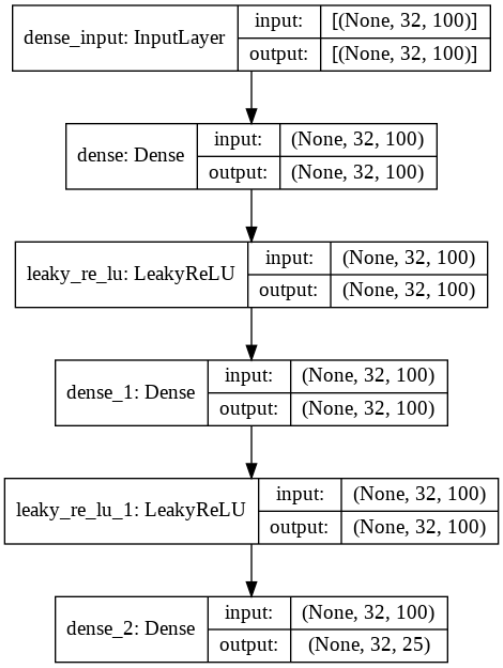

Figure 2: Plot of Generator architecture used in the experiments

4.2 Data

We are going to create synthetic time series with the specified GAN on the following financial Instruments:

Asset Class	Instrument	Symbol (Yahoo Finance)
Commodities	Gold Futures, Continuous Contract	GC=F
Forex	EUR vs USD Foreign Exchange Reference Rate	EURUSD=X
Futures	S&P 500 Volatility Index VIX Futures	VIX
Equity Index	S&P 500 Index, Continuous Contract	GSPC
Shares	Apple Inc. (AAPL) Stock Prices, Dividends and Splits	AAPL

All data is obtained directly from Yahoo Finance. The period for all assets is from and including January 1, 2010 until and including April 1, 2021, with the daily closing available on trading days from Monday to Friday.

5 Data Analysis

Before we dive into the deep waters of training models to create synthetic time series, we want to give a more detailed background on the input data. Therefore an EDA from Apple Inc. is shown here.

First the daily closing price from apple from the period of 01.01.2010 to 01.04.2021, which are captured from Yahoo Finance, is shown in Figure 3.

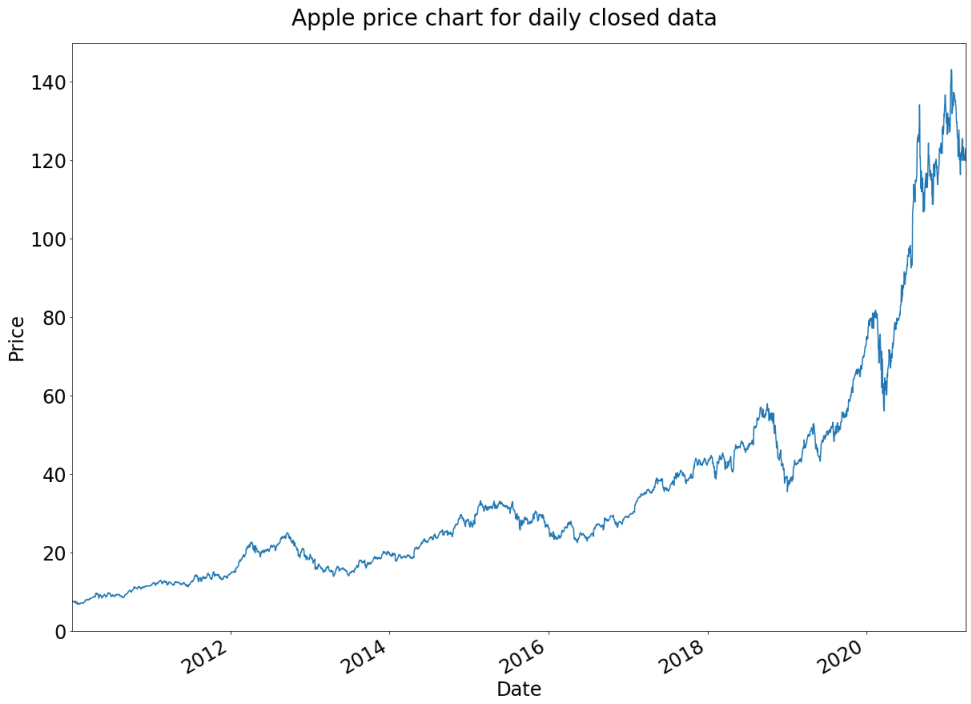

Figure 3: Apple daily closing price from 01.01.2010 to 01.04.2021

Most methods of financial time series analysis necessitates that the data is made stationarity. A stationary time series gets represented by data over time, whose statistical properties remains constant regardless of a change in the time origin. There are various ways of making time series stationary, but those ways are all looking at the difference between values, rather than the absolute values. In the case of market data, the common way to get stationary data is to work with log-returns. They are calculated as the natural logarithm of the asset today divided by the asset yesterday: $\ln(\frac{Vt}{Vt-1})$. The log returns by time is shown in Figure 4.

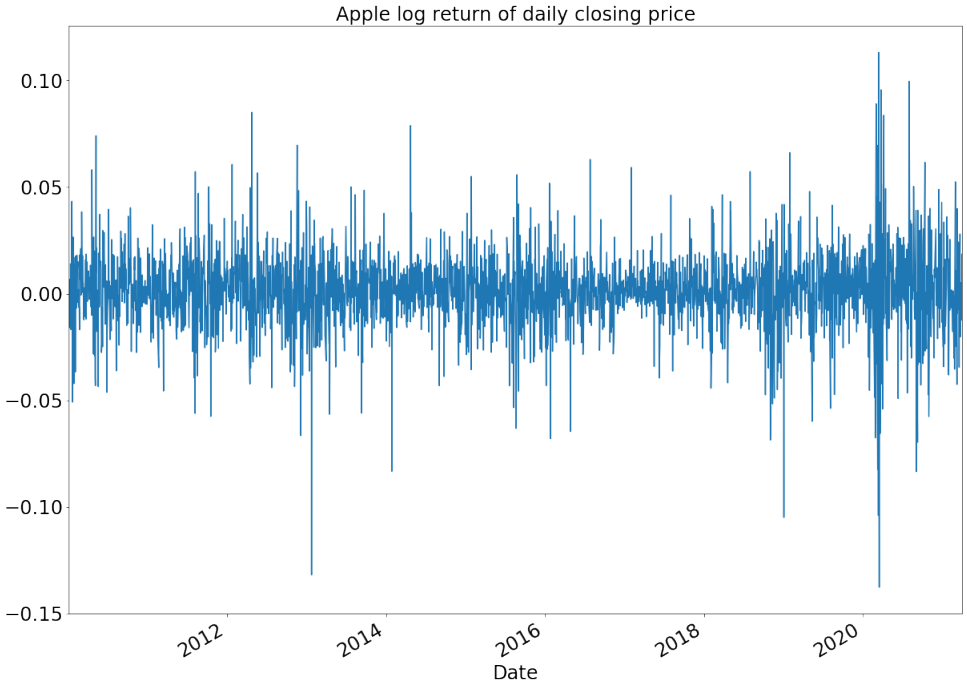

Figure 4: Log returns of Apple daily closing price from 01.01.2010 to 01.04.2021

We observe that the log returns of Apple daily closing price look close to normal in Figure 5, centred about zero in general and almost perfectly symmetrical.

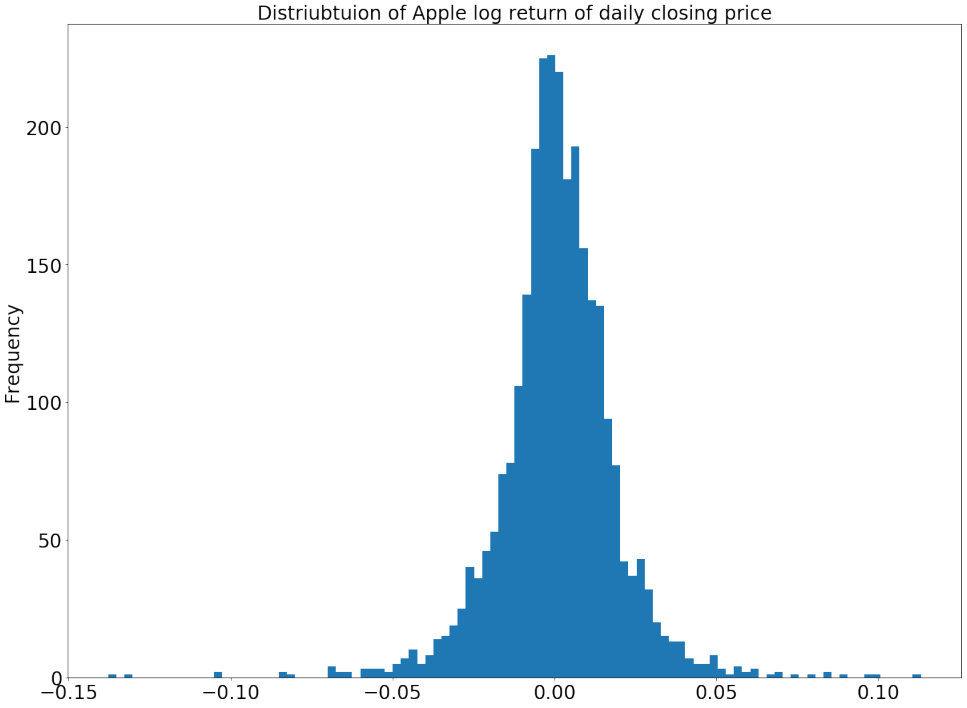

Figure 5: Histogram of log returns of Apple daily closing price from 01.01.2010 to 01.04.2021

6 Results of creating synthetic financial time series

We will use the model discussed as in Section 4.1 which is trained with Apple data from January 1, 2010 until and including April 1, 2021, which was discussed in Section 5. The results from the other asset classes, represented by one asset, are shown in the Appendix. In Figure 12 we show the generator and discriminator loss of as a function of the number of epochs of training for the GAN. It can be seen, that the discriminator loss on the final epochs is around 0.6, thus it almost reached the optimal state of 0.5.

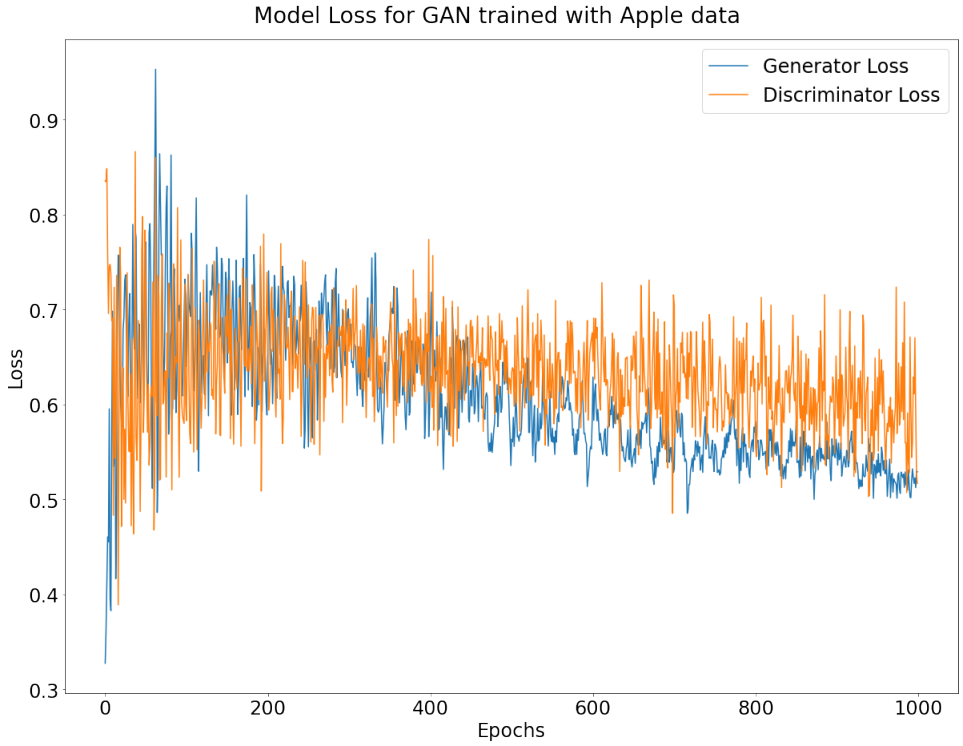

Figure 6: Generator and discriminator loss as a function of training epochs.

As the model has been trained with Apple time series, we are now able to show some synthetic time series generated by the GAN. These time series should look like the Apple price time series and have the same properties. In Figure 7, the apple price time series in red is shown with 50 generated time series of the GAN.

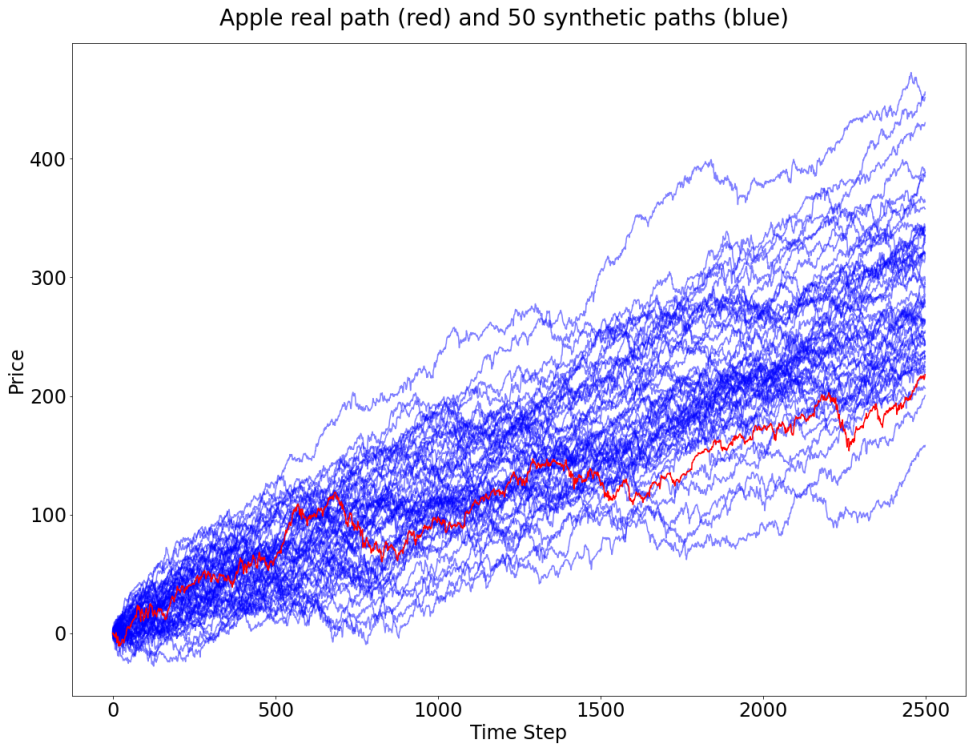

Figure 7: 50 Synthetic time series (blue) and real Apple time series closing price for 2500 time steps.

Moreover we are interested, if the statistical properties of the synthetic returns are the same as the real apple time series. Therefore in Figure 8 the log returns of one synthetic time series is compared to the real apple time series log returns. Also the skewness and the kurtosis of the real and synthetic log returns is shown in that Figure. The distribution of the two returns looks similar which is reflected in the skewness of those two, as they are in the same magnitude. The kurtosis on the other hand looks quite different, as the real distribution of the log returns has a kurtosis of 6.4, while the synthetic has a kurtosis of 1.18.

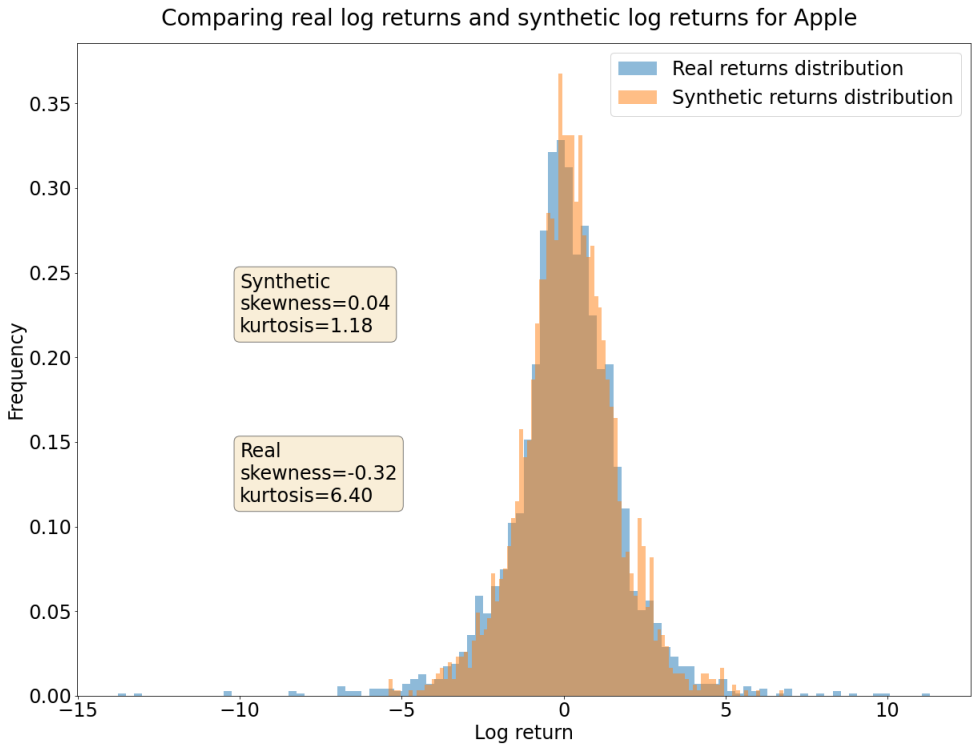

Figure 8: Histogram of log returns of the closing price for Apple and a synthetic time series.

To further check the skewness and kurtosis of the synthetic log returns, 1000 synthetic log return distributions are generated of the trained GAN model. From these 1000 distributions, the skewness and kurtosis are calculated. These 1000 skewness and kurtosis values are plotted in a distribution and compared with the real value of the log return skewness and kurtosis. In Figure 9 the kurtosis of the 1000 synthetical log returns (blue) can be compared to the real kurtosis of the apple log return (red). Moreover in Figure 10 the skewness of the 1000 synthetical log returns (blue) can be compared to the real skewness of the apple log return (red). We can see, that the skewness of the synthetic data is slightly worse than the real one, but the kurtosis is drastically different, thus the model was not able to capture the underlying properties of the real time series.

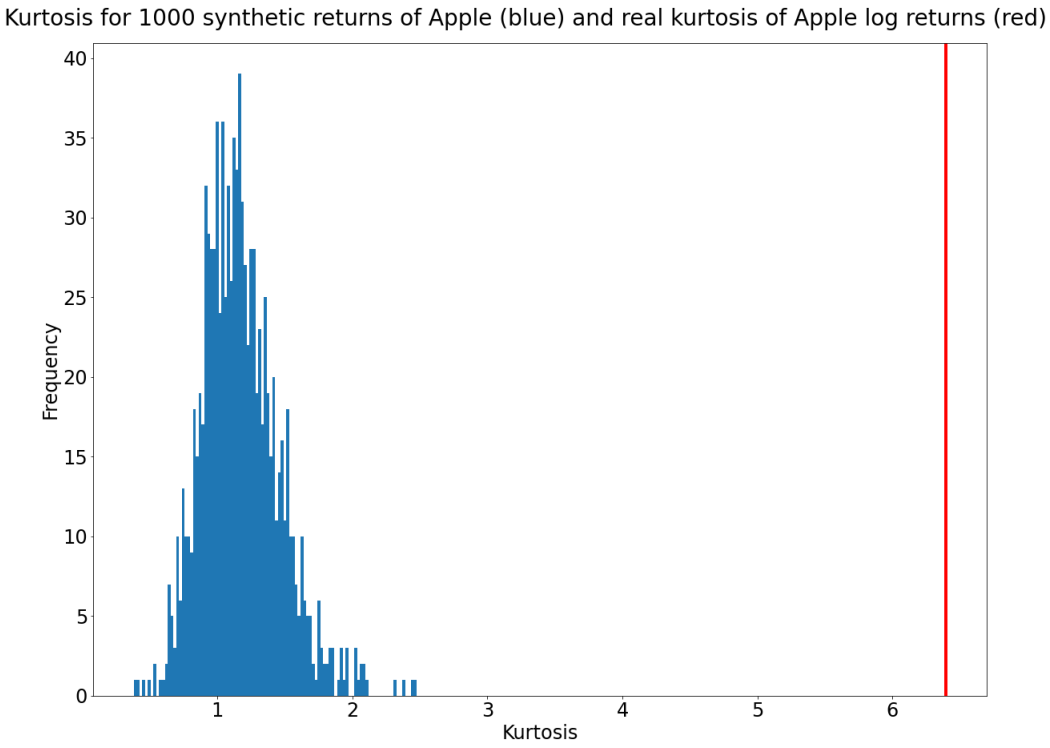

Figure 9: Distribution of kurtosis of 1000 synthetic time series log returns in blue and real apple log return kurtosis in red.

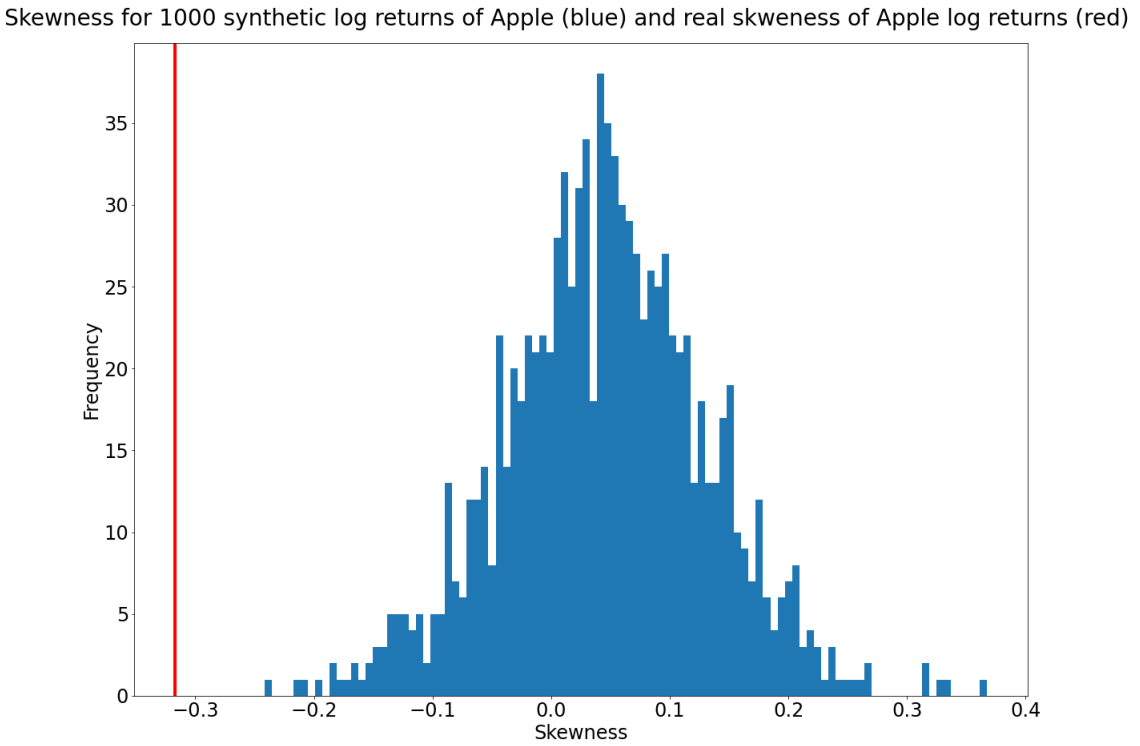

Figure 10: Distribution of skewness of 1000 synthetic time series log returns in blue and real apple log return skewness in red.

To investigate some more in the synthetic time series, 100 synthetic log returns are generated of the trained model. From those 100 log returns, the ACF is plotted. Additionally the ACF of the real log return is plotted. This can be seen in Figure 11, where the 100 synthetic log returns are plotted in orange and the real log returns are plotted in blue. We can see that the ACF of the synthetic data and real data behaves the same.

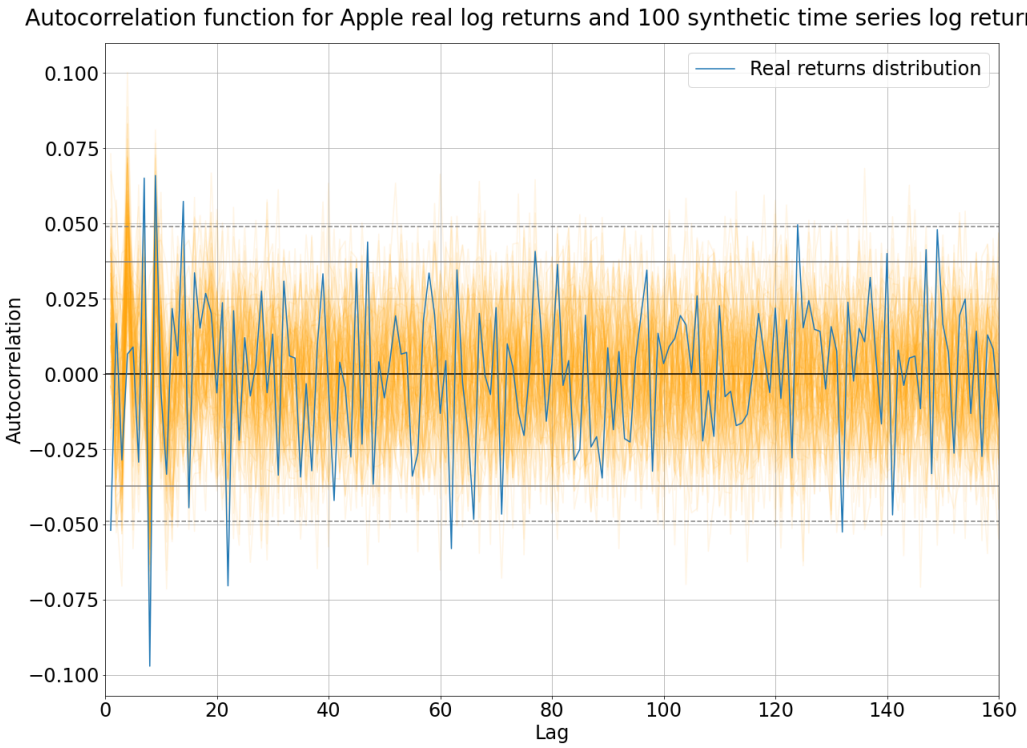

Figure 11: Autocorrelation function for Apple real log returns and 100 synthetic time series log returns.

Furthermore 100 synthetic log returns are generated of the trained model. From those 100 log returns, the ACF of the absolute log return is plotted. Additionally the ACF of the real absolute log return is plotted. This can be seen in Figure 11, where the 100 synthetic absolute log returns are plotted in orange and the real absolute log returns are plotted in blue. We can see that the ACF of the synthetic data behaves quite different for the first 60 lags. Afterwards it looks the same.

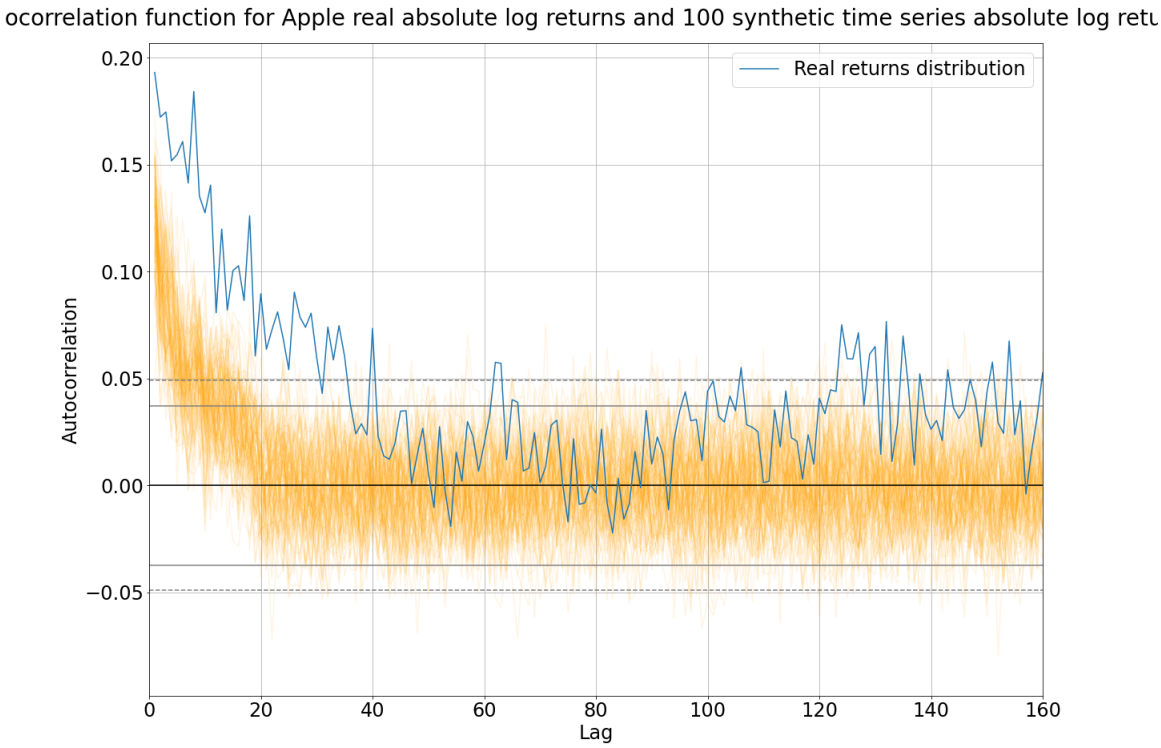

Figure 12: Autocorrelation function for absolute Apple real log returns and 100 synthetic time series absolute log returns

7 Conclusion and summary

In this work we explored the application of Generative Adversarial Networks to time series data, in order to perform data augmentation of financial datasets. We proposed a GAN that was able to generate synthetic time series given an asset. In Section 6 we showed an approach to measuring the performance of the generation procedures for various assets. This approach illustrated how synthetic data can help deep learning models mitigate overfitting and generalize better.

Regarding future works, more complicated neural network layers can be applied in the discriminator and generator. For example, deep convolutional layers have seen huge adoption in computer vision application. GAN with convolutional networks has been developed, and proved to be an appealing alternative to improve the GAN learning process and perform dimension deduction Radford et al. (2016). In addition, recurrent neural networks and LSTM have been successfully applied in GAN for stock market prediction on high frequency data Xingyu Zhou and Zhao (2018).

On the other hand, instead of using the vanilla GAN, more complex architecutres of GANs, such as Wasserstein GANArjovsky et al. (2017), Relativistic Standard GAN Jolicoeur-Martineau (2018) and Conditional GANs Mirza and Osindero (2014) could improve the results to generate synthetic financial price data for various assets.

These results open up many research paths, given the flexibility of our procedures, possibly leading to new applications of deep learning and reinforcement learning techniques in financial settings, as lack of training data and overfitting have been two of the biggest concerns which have set back this kind of developments.

In the GAN community, many papers focus solely on implementing new improved architectures for the networks used as generator and discriminator and demonstrating the usefulness of certain techniques, such as Batch Normalization, Residual Connections, in order to beat state-of-the-art and Inception scores. While some of these breakthroughs are universal to all GANs, they're mostly focused on image generation and time series data presents a completely different structure for which we think a lot could be researched.

Development of GANs for time series generation relies upon having a common baseline metric of success, in the same way as the Fréchet Inception Distance and the Inception Score are used in Image Generation. Having a standard classifier such as the ResNet and taking AUC scores of a fixed hyperparameter classification between real and synthetic data could be an approach

References

Asset allocation in emerging market space using wasserstein generative adversarial networks. 2020.

Understanding Random Forest. https://towardsdatascience.com/understanding-random-forest-58381e0602d2, 2020. Accessed: 2020-10-20.

Francesca Albertini and Eduardo Sontag. For neural networks, function determines form. *Neural Networks*, 6(7): 975–990, 1993.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017.

Wei Bao, Jun Yue, and Yulei Rao. A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PloS one*, 12(7):e0180944, 2017.

L. Breiman. Random Forests. *Machine Learning*, 45(1), 2001.

Menachem Brenner and Dan Galai. New Financial Instruments for Hedge Changes in Volatility. *Financial Analysts Journal*, 45(4):61–65, 1989.

Luyang Chen, Markus Pelger, and Zhu Jason. Deep Learning in Asset Pricing. November 2020. URL https://ssrn.com/abstract=3350138.

Aidan Clark, Jeff Donahue, and Karen Simonyan. Adversarial video generation on complex datasets, 2019.

Laurie Davies and Walter Krämer. Stylized facts and simulating long range financial data, 2016.

Fernando De Meer Pardo. Enriching Financial Datasets with Generative Adversarial Networks. Master's thesis, Delft University of Technology, Netherlands, August 2019. URL http://resolver.tudelft.nl/uuid:51d69925-fb7b-4e82-9ba6-f8295f96705c.

K. Demeterfi, E. Derman, M. Kamal, and J. Zou. More Than You Ever Wanted To Know About Volatility Swaps, 1999.

Google Developers. Loss Functions | Generative Adversarial Networks | Google Developers. [online]. https://developers.google.com/machine-learning/gan/loss. Accessed 21 March 2021.

Luca Di Persio and Oleksandr Honchar. Artificial neural networks approach to the forecast of stock market price movements. *International Journal of Economics and Management Systems*, 1, 2016.

Nir Diamant, Dean Zadok, Chaim Baskin, Eli Schwartz, and Alex M. Bronstein. Beholder-gan: Generation and beautification of facial images with conditioning on their beauty level, 2019.

Cem Dilmegani. The Ultimate Guide to Synthetic Data in 2021, June 2021. URL https://research.aimultiple.com/synthetic-data/.

Dmitry Efimov, Di Xu, Luyang Kong, Alexey Nefedov, and Archana Anandakrishnan. Using generative adversarial networks to synthesize artificial financial datasets. *NIPS 2019*, 2019. URL https://arxiv.org/pdf/2002.02271.pdf.

Jesse Engel, Kumar Krishna Agrawal, S. Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts. Gansynth: Adversarial neural audio synthesis. *ArXiv*, abs/1902.08710, 2019.

Cristóbal Esteban, Stephanie L. Hyland, and Gunnar Rätsch. Real-valued (medical) time series generation with recurrent conditional gans, 2017.

FCA and City of London. Digital Sandbox Pilot, 2021. URL https://www.digitalsandboxpilot.co.uk/.

Financial Conduct Authority. Supporting innovation in financial services: the digital sandbox pilot. Technical Report Pub ref: D495, 12 Endeavour Square London E20 1JN, April 2021. URL https://www.digitalsandboxpilot.co.uk/.

Ugo Fiore, Alfredo De Santis, Francesca Perla, Paolo Zanetti, and Francesco Palmieri. Using generative adversarial networks for improving classification effectiveness in credit card fraud detection. *Information Sciences*, 479:448–455, December 2017. doi: https://doi.org/10.1016/j.ins.2017.12.030.

T. Fischera and C. Krausb. Deep Learning with Long Short-term Memory Networks for Financial Market Predictions. *FAU Discussion Papers in Economics*, 11, 2017.

Aurélien Géron. *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems*. "O'Reilly Media, Inc.", 2017.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

Google-Developers. GAN TRAINING | GENERATIVE ADVERSARIAL NETWORKS | GOOGLE DEVELOPERS. https://developers.google.com/machine-learning/gan/training, 2020. Accessed 21 March 2021.

J. M. Griffin and A. Shams. Manipulation in the VIX? *The Review of Financial Studies*, 31(4):1377–1417, 2017.

JB Heaton, NG Polson, and Jan Hendrik Witte. Deep learning for finance: deep portfolios. *Applied Stochastic Models in Business and Industry*, 33(1):3–12, 2017.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

TJ Horan. Credit Card Fraud: It's Still a Thing (and as Big as Ever), January 2021. URL https://www.fico.com/blogs/credit-card-fraud-its-still-thing-and-big-ever.

Alexandre Hubert. Using Deep Learning for Better Option Pricing, 2020. Whitepaper.

Todd P. Huster, Jeremy E. Cohen, Z. Lin, Kevin S. Chan, C. Kamhoua, Nandi O. Leslie, Cho-Yu Jason Chiang, and V. Sekar. Pareto gan: Extending the representational power of gans to heavy-tailed distributions. *ArXiv*, abs/2101.09113, 2021.

E. Inzaugarat. UNDERSTANDING NEURAL NETWORKS: WHAT, HOW AND WHY? https://towardsdatascience.com/understanding-neural-networks-what-how-and-why-18ec703ebd31, 20. Accessed: 2020-11-19.

S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. https://arxiv.org/abs/1502.03167, 2015.

H. Nguyen J. Hosker, S. Djurdjevic and R. Slater. Improving vix futures forecasts usingmachine learning methods. https://scholar.smu.edu/datasciencereview/vol1/iss4/6. Accessed: 2020-10-20.

J.Jordan. Normalizing your data (specifically, input and batch normalization). Normalizing your data (specifically, input and batch normalization)., 2020. Accessed: 2020-11-19.

Alexia Jolicoeur-Martineau. The relativistic discriminator: a key element missing from standard gan, 2018.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation, 2018.

Katie Martin. Flash Crash — the trading savant who crashed the US stock market. *Financial Times*, May 2020. URL https://www.ft.com/content/5ca93932-8de7-11ea-a8ec-961a33ba80aa.

Yerin Kim, Daemook Kang, Mingoo Jeon, and Chungmok Lee. GAN-MP hybrid heuristic algorithm for non-convex portfolio optimization problem. *The Engineering Economist*, June 2019. doi: 10.1080/0013791X.2019.1620391.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980, 2014. URL http://arxiv.org/abs/1412.6980.

Adriano Koshiyama, Philip Treleaven, and Nick Firoozye. Generative adversarial networks for financial trading strategies fine-tuning and combination. *Quantitative Finance*, September 2020. doi: 10.1080/14697688.2020.1790635. URL https://doi.org/10.1080/14697688.2020.1790635.

L. Kryzanowski, M. Galler, and D. Wright. Using artificial neural networks to pick stocks. *Financial Analysts Journal*, 49(4):21–27, 1993.

Praveen Kumar and Duane J Seppi. Futures Manipulation with "Cash Settlement.". *Journal of Finance*, 47(4):1485–502, 1992.

A. Guizzardi L. V. Ballestra and F. Palladini. Forecasting and trading on the vix futures market: A neural network approach based on open to close returns and coincident indicators. http://www.sciencedirect.com/science/article/pii/S0169207019301372. Accessed: 2020-10-20.

Edmond Lezmi, Jules Roche, T. Roncalli, and Jiali Xu. Improving the robustness of trading strategy backtesting with boltzmann machines and generative adversarial networks. *ArXiv*, abs/2007.04838, 2020.

Junyi Li, Xitong Wang, Yaoyang Lin, Arunesh Sinha, and Micheal P. Wellman. Generating Realistic Stock Market Order Streams. *arXiv:2006.04212 [cs, q-fin]*, June 2020. URL http://arxiv.org/abs/2006.04212. arXiv: 2006.04212.

Elena Popina Lu Wang and Luke Kawa. VIX Rigging Talk Erupts on Wall Street After Another Wild Swing. https://www.bloomberg.com/news/articles/2018-04-18/vix-rigging-talk-erupts-on-wall-street-after-another-wild-swing, 2018. Accessed: 2018-11-27.

Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. Are gans created equal? a large-scale study, 2018.

Kelvin Luu, Rik Koncel-Kedziorski, Kyle Lo, Isabel Cachola, and Noah A. Smith. Citation text generation, 2020.

Hans Malmsten and Timo Teräsvirta. Stylized Facts of Financial Time Series and Three Popular Models of Volatility, August 2004. URL https://ideas.repec.org/p/hhs/hastef/0563.html.

G. Mariani, Y. Zhu, Jian bo Li, F. Scheidegger, R. Istrate, C. Bekas, and A. Malossi. Pagan: Portfolio analysis with generative adversarial networks. *Mutual Funds*, 2019a.

Giovanni Mariani, Yada Zhu, Jianbo Li, Florian Scheidegger, Roxana Istrate, Costas Bekas, and A. Cristiano I. Malossi. PAGAN: Portfolio Analysis with Generative Adversarial Networks. September 2019b. URL https://arxiv.org/pdf/1909.10578.pdf.

Gautier Marti. corrgan.io, 2019. URL http://www.corrgan.io/.

Gautier Marti. CORRGAN: Sampling Realistic Financial Correlation Matrices Using Generative Adversarial Networks. *IEEE Xplore*, May 2020. doi: 10.1109/ICASSP40776.2020.9053276. URL https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9053276.

Alexandre Miot. Adversarial trading, 2020.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets, 2014.

James Montantes. Deep Learning in Finance: Is This The Future of the Financial Industry?, July 2020. URL https://towardsdatascience.com/deep-learning-in-finance-is-this-the-future-of-the-financial-industry-a29b561031e9.

Pratyush Muthukumar and Jie Zhong. A Stochastic Time Series Model for Predicting FinancialTrends using NLP. *arXiv*, February 2021. URL https://arxiv.org/pdf/2102.01290.pdf.

James Hays Zsolt Kira Naveen Kodali, Jacob Abernethy. ON CONVERGENCE AND STABILITY OF GANS. https://arxiv.org/pdf/1705.07215.pdf, 2017. Accessed 21 March 2021.

J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, Q. Le, and A. Ng. On optimization methods for deep learning. *International Conference on Machine Learning (ICML-11)*, pages 265—-272, 2011.

Hao Ni, Lukasz Szpruch, Magnus Wiese, Shujian Liao, and Baoren Xiao. Conditional Sig-Wasserstein GANs for Time Series Generation. *arXiv*, June 2020. URL https://arxiv.org/abs/2006.05421.

D. Olson and C. Mossman. Neural network forecasts of Canadian stock returns using accounting ratios. *International Journal of Forecasting*, 19(3):453–465, 2003.

Chi Seng Pun, Lei Wang, and H. Wong. Financial thought experiment: A gan-based approach to vast robust portfolio selection. In *IJCAI*, 2020a.

Chi Seng Pun, Lei Wang, and Hoi Ying Wong. Financial Thought Experiment: A GAN-based Approach to Vast Robust Portfolio Selection. *IJCAI-20*, 2020b.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2016.

RCMAlternatives. SELLER BEWARE: EVERYBODY'S SHORT VIX THESE DAYS. https://www.rcmalternatives.com/2017/05/seller-beware-everybodys-short-vix-these-days/, 2017. Accessed: 2018-04-29.

J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

Marco Schreyer, Timur Sattarov, Damian Borth, and Bernd Reimer. Adversarial Learning of Deepfakes in Accounting. *arXiv:1910.03810 [cs.LG]*, October 2019. URL https://arxiv.org/pdf/1910.03810.pdf. arXiv:1910.03810v1.

Sigal Shaked. How Do Companies Use Synthetic Data in Machine Learning?, 2021. URL https://www.datomize.com/use-synthetic-data-in-machine-learning/.

Luca Simonetto. Generating spiking time series with Generative Adversarial Networks: an application on banking transactions. Master's thesis, University of Amsterdam, Netherlands, September 2018.

Kaleb E Smith and Anthony O Smith. Conditional GAN for timeseries generation. *arXiv*, June 2020. URL arXiv:2006.16477v1.

Chester Spatt. Security Market Manipulation. *Annual Review of Financial Economics*, 6(1):405–418, 2014.

Ramya Srinivasan, Ajay Chander, and Pouya Pezeshkpour. Generating User-friendly Explanations for Loan Denials using GANs. *arXiv:1906.10244v1 [cs.LG]*, June 2019. URL https://arxiv.org/pdf/1906.10244.pdf.

N. Srivastava, G. Hinton, A. Krizhevsky, and I. Sutskever. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929—-1958, 2014.

Victor Storchan, Tucker Balch, and Svitlana Vyetrenko. MAS-GAN: Adversarial Calibration of Multi-agent Market Simulators. 2021. URL https://openreview.net/pdf?id=1z_Hg9oBCtY. Paper under review.

He Sun, Zhun Deng, Hui Chen, and David C. Parkes. Decision-Aware Conditional GANs for Time Series Data. *arXiv:2009.12682v2 [cs.LG]*, September 2020. URL https://arxiv.org/pdf/2009.12682v2.pdf.

Shuntaro Takahashi, Yu Chen, and Kumiko Tanaka-Ishii. Modelling financial time-series with generative adversarial networks. *Physica A: Statistical Mechanics and its Applications*, 527, April 2019. URL https://reader.elsevier.com/reader/sd/pii/S0378437119307277?token=107F6D3404A17950ACCF842303C9DBC737B7BAA6928AA07382BAD36F2CE37DE0BA5A8FD80B1D7E7074E9AAB2C896FD2F.

M. Telgarsky. Benefits of depth in neural networks. *Conference on Learning Theory (COLT)*, 2016.

Wilfredo Tovar. Deep Learning Based on Generative Adversarial and Convolutional Neural Networks for Financial Time Series Predictions. *arXiv:2008.08041v2 [eess.SP]*, August 2020. URL https://arxiv.org/ftp/arxiv/papers/2008/2008.08041.pdf.

Jorino van Rhijn, C. Oosterlee, L. Grzelak, and Shuaiqiang Liu. Monte carlo simulation of sdes using gans. *ArXiv*, abs/2104.01437, 2021.

Manhar Singh Walia. Synthetic Data Generation Using Wasserstein Conditional Gans With Gradient Penalty (WCGANS-GP). https://arrow.tudublin.ie/cgi/viewcontent.cgi?article=1236&context=scschcomdis, 2020. Accessed 21 March 2021.

Robert E Whaley. Derivatives on market volatility: Hedging tools long overdue. *The Journal of Derivatives*, 1(1): 71–84, 1993.

Halbert White. Artificial Neural Networks: Approximation and Learning Theory. *Blackwell Publishers, Inc., Cambridge, MA, USA*, 1992.

Magnus Wiese, Robert Knobloch, Ralf Korn, and Peter Kretschmer. Quant GANs: Deep Generation of Financial Time Series. *arXiv:1907.06673v2 [q-fin.MF]*, December 2019. doi: 10.1080/14697688.2020.1730426. URL https://arxiv.org/pdf/1907.06673.pdf.

Magnus Wiese, Robert Knobloch, Ralf Korn, and Peter Kretschmer. Quant gans: deep generation of financial time series. *Quantitative Finance*, 20(9):1419–1440, Apr 2020. ISSN 1469-7696. doi: 10.1080/14697688.2020.1730426. URL http://dx.doi.org/10.1080/14697688.2020.1730426.

D. H. Wolpert and W. G. Macready. No Free Lunch Theorems for Optimization. https://ti.arc.nasa.gov/m/profile/dhw/papers/78.pdf, 1997. Accessed: 2018-12-18.

Weijie Wu, Fang Huang, Yidi Kao, Zhou Chen, and Qi Wu. Prediction Method of Multiple Related Time Series Based on Generative Adversarial Networks. January 2021. doi: https://doi.org/10.3390/info12020055. URL https://www.mdpi.com/2078-2489/12/2/55/htm.

Guyu Hu Siqi Tang Xingyu Zhou, Zhisong Pan and Cheng Zhao. Stock market prediction on high frequency data using generative adversarial nets. https://www.hindawi.com/journals/mpe/2018/4907423/, 2018. Accessed 21 March 2021.

Frank Xu. Building your own Self-attention GANs, July 2020. URL https://towardsdatascience.com/building-your-own-self-attention-gans-e8c9b9fe8e51.

Roy Yogev. How Do Financial Services Use Synthetic Data?, 2021. URL https://www.datomize.com/how-do-financial-services-use-synthetic-data/.

Jinsung Yoon, Mihaela van der Schaar, and Daniel Jarrett. Time-series Generative Adversarial Networks. *33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*, 2019.

Kang Zhang, Guoqiang Zhong, Junyu Dong, Shengke Wang, and Yong Wang. Stock Market Prediction Based on Generative Adversarial Network. *Procedia Computer Science*, 147(1877-0509):400–406, 2019. URL https://pdf.sciencedirectassets.com/280203/1-s2.0-S1877050919X00034/1-s2.0-S1877050919302789/main.pdf?X-Amz-Security-Token=IQoJb3JpZ2luX2VjEFMaCXVzLWVhc3QtMSJHMEUCIB11QFDHwGW%2B4A%2BxP2278ZGMpoaZmPrEtd8CWHzSElcdAiEAuWqfbIoUgWCzy2sImuouF4GgmUmi71LHUYN68yrclywqtAMIfBADGgwwNTkwMDM1NDY2FLK%2B0c7VQwMP7r%2BAH5Tl5U86erkz3WcV10Ie6C3xsGeu65kRGfQHb10CxPiL1KuitkwL98FQRLaF80edHhIHrF9M8XR4UnwiC2BswjLPwvU2sO6c0aB0IgHTvOpFrzadqP2k0NRm6lBBzmD40Du71XEnu%2F%2B%2BkafHi6mNHXfAomqsoDU3%2BzaVXDq83aEGlotkmHf8ezFLT81qqLJ%2BgL7edCiFNeHtpC3%2FEKctsCXihcwdu1kb%2BZpR0%2FMQGmPU3WfyZwCqW8LLZn2RITrabCbJ%2BF%2Bmhiso3N2hPrXXLwG1MTNcCLOl%2B1toqFdy9phkzfOIVP39fEYW%2F6F%2BE60a4OEHZ8Hr7KzJuklUPFaZxYmvaIr5sJ6NfOlVDp4%2B25z%

2BsKVB8wSEKnO2Ue8aWYrsholJ5YWuEeC8dd782Y33Fmj0l5z26WpUICf6GQqIGy53KHWUBZRlKELgu4v9DEdn3Qln4Uy39rQMICdp
2FPlpVun1YHoMWdq%2BdaVEzg1Obi7Nz%2Fz7O5ENImt0fdUXuirQTsGNAY8FH5GIyHJ%2ByDh88N7qfRKRdR%
2FJgbIYAgM59OzjKUypVEkLfC7nSYaEYvtZ8RHXjt4VRqny2HKPJjw6Y99b8olQGmBWLD7i2KBqLXkzyYzBb95PDTcNkbBs2T85GPF
2FtlcJ7wSD%2BbDS9xqCmyBUeCPEX0jXXsrs5p2WfWAsrydiptNm%2B1O%2BFkvJKO4XXvC%
2F0ldsmZoD1jvjDf59VUFtylOw6%2FR3qA%3D%3D&X-Amz-Algorithm=AWS4-HMAC-SHA256&
X-Amz-Date=20210310T192600Z&X-Amz-SignedHeaders=host&X-Amz-Expires=300&
X-Amz-Credential=ASIAQ3PHCVTYS6DOBTOJ%2F20210310%2Fus-east-1%2Fs3%2Faws4_request&
X-Amz-Signature=7ef10ea464b16e830245485557bee4eaf6e5e7bf1ecd924ce20fe97d6e38245a&
hash=09e70a24d8662fa4ec317ba3fe45f1d0b123df77ada3c3e6cb00d0d199837092&
host=68042c943591013ac2b2430a89b270f6af2c76d8dfd086a07176afe7c76c2c61&
pii=S1877050919302789&tid=spdf-28620e71-76c2-463f-8454-0e4c85a82003&sid=
2ec8a2bb3fff454aba4a0cb76b7f3ff45ff8gxrqb&type=client.

Zhaohui Zhang, Lijun Yang, and Ligong Chen. A generative adversarial network–based method for generating negative financial samples. *International Journal of Distributed Sensor Networks*, 16(2), February 2020. URL https://doi.org/10.1177/1550147720907053.

Xingyu Zhou, Zhisong Pan, Guyu Hu, Siqi Tang, and Cheng Zhao. Stock Market Prediction on High-Frequency Data Using Generative Adversarial Nets. *Mathematical Problems in Engineering*, 2018, April 2018. doi: https://doi.org/10.1155/2018/4907423.

A Gold

A.0.1 Data Analysis

First the daily closing price of Gold from the period of 01.01.2010 to 01.04.2021, which are captured from Yahoo Finance, is shown in Figure 13.

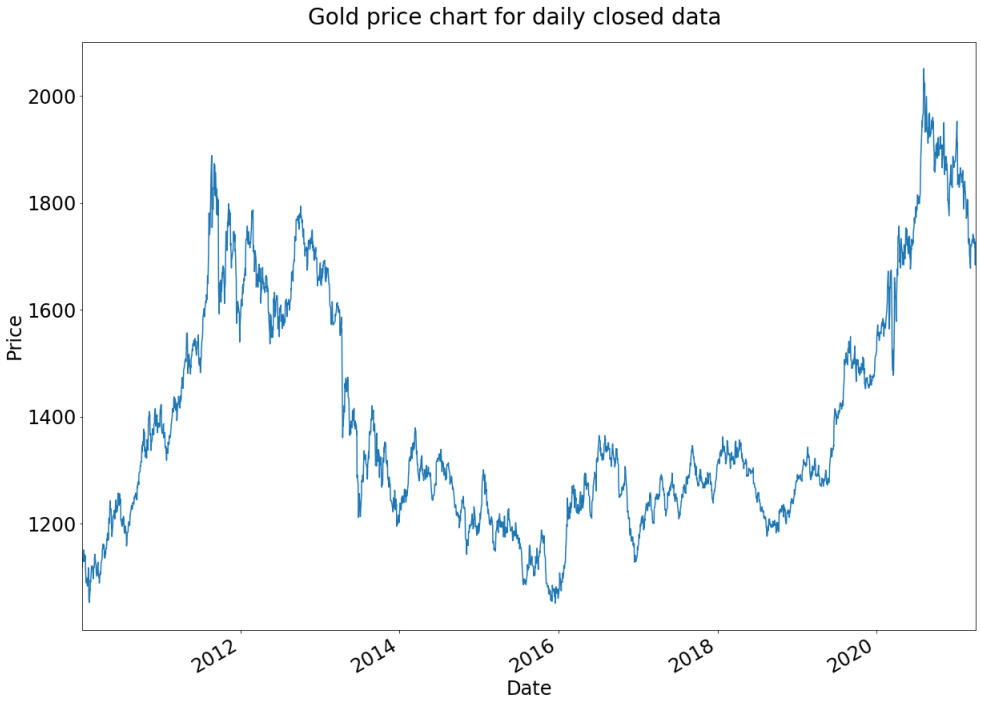

Figure 13: Gold daily closing price from 01.01.2010 to 01.04.2021

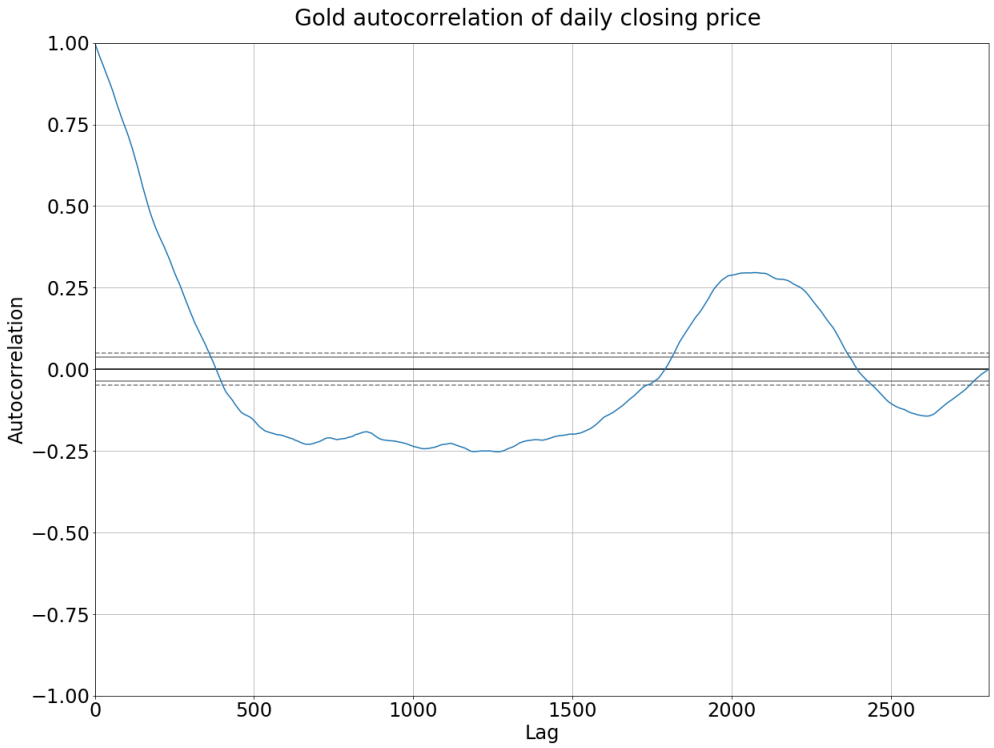

Figure 14: Autocorrelation funciton of Gold daily closing price from 01.01.2010 to 01.04.2021

Financial time series data is usually analyzed by the return instead of the price, to get a more stationary time series data. The log returns simply eliminate non-stationary properties of the data, making it more stable. The log returns by time is shown in Figure 15.

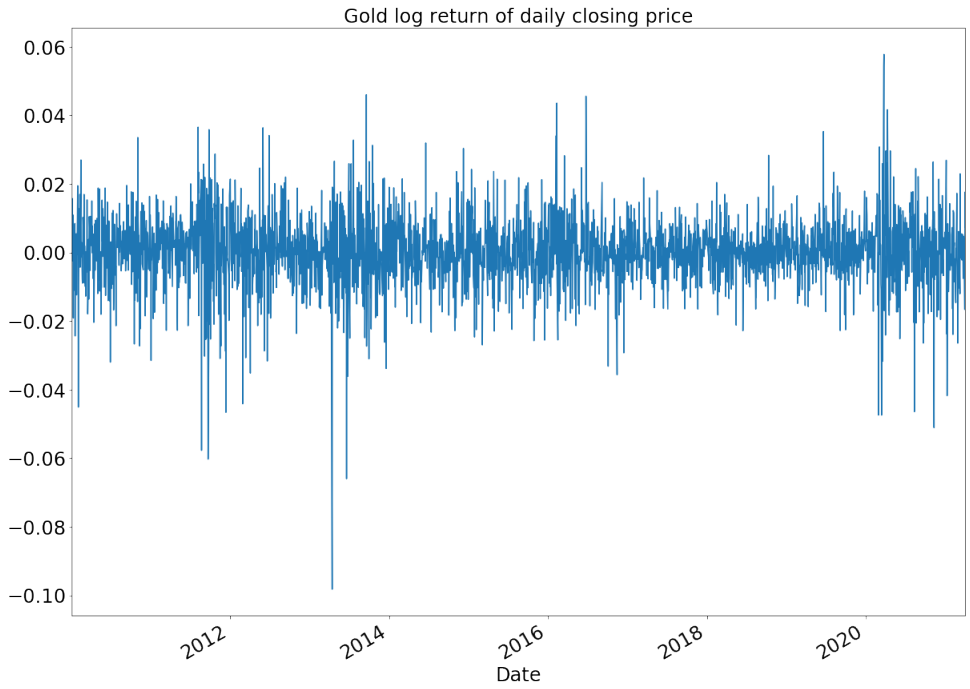

Figure 15: Log returns of Gold daily closing price from 01.01.2010 to 01.04.2021

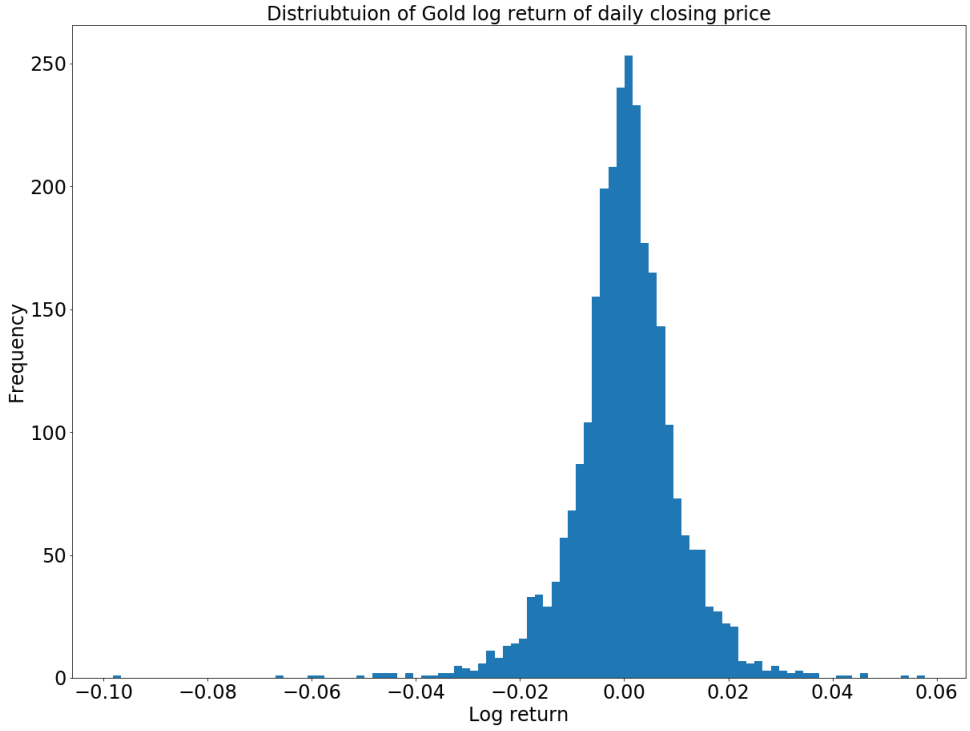

Figure 16: Histogram of log returns of Gold daily closing price from 01.01.2010 to 01.04.2021

A.0.2 Experiment Results

This section examines the resulting synthetic time series of the GAN when trained with Gold data. In Figure **??** we show the generator and discriminator loss of as a function of the number of epochs of training for the GAN.

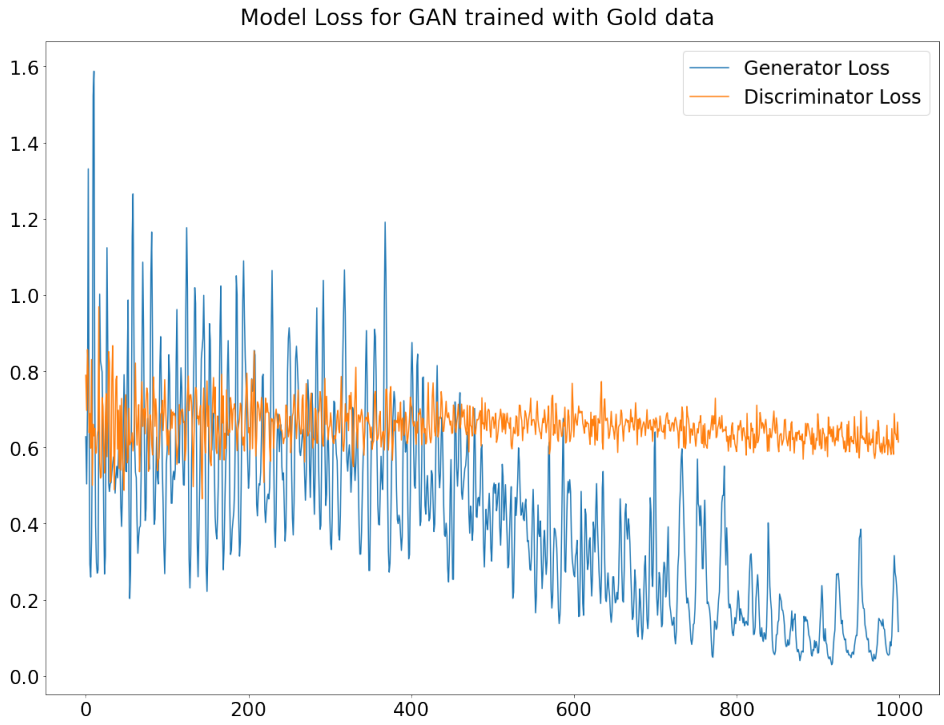

Figure 17: Generator and discriminator loss as a function of training epochs.

As the model has been trained with Gold time

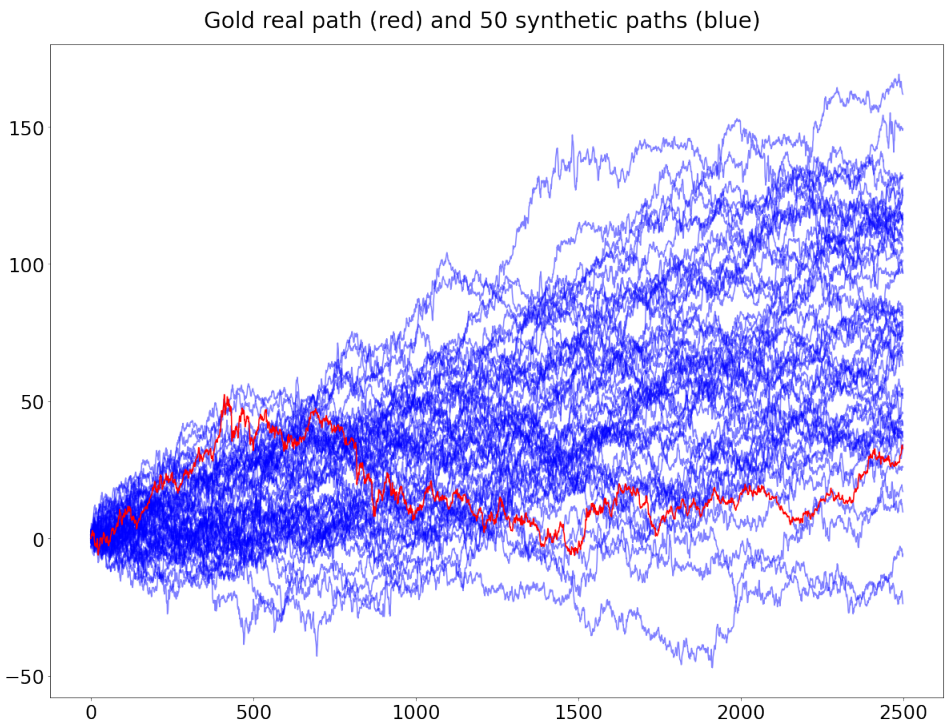

Figure 18: 50 Synthetic time series (blue) and real Gold time series closing price for 2500 time steps.

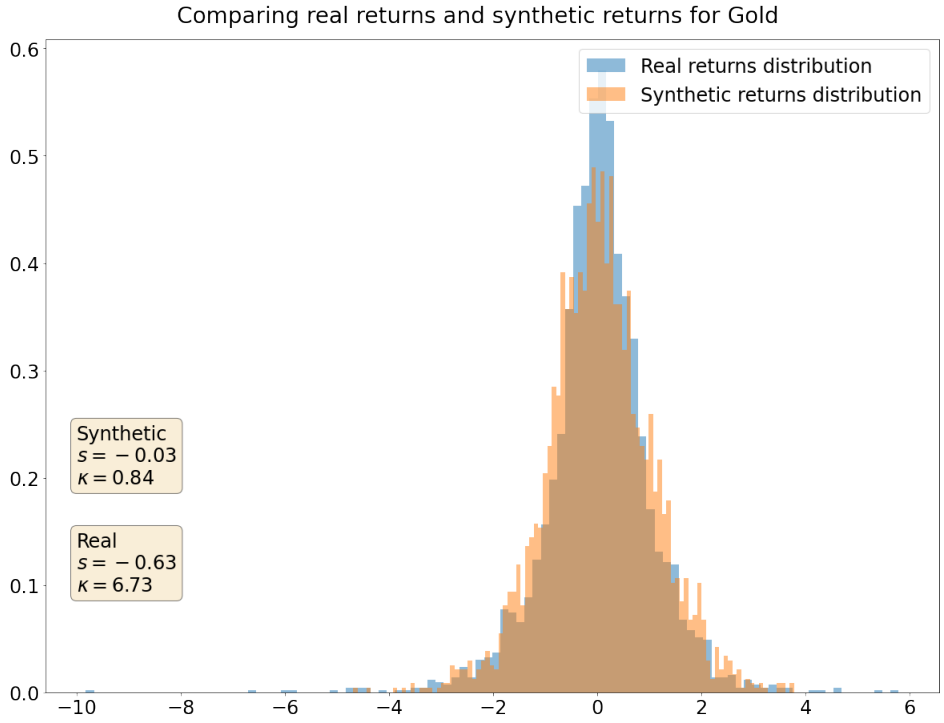

Figure 19: Histogram of log returns of the closing price for Gold and a synthetic time series.

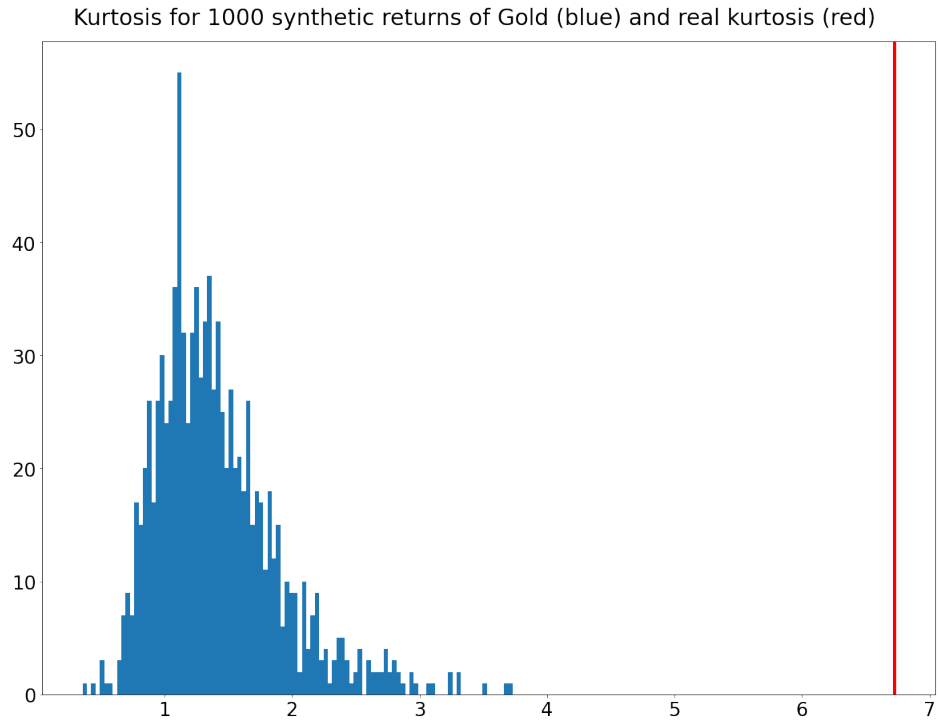

Figure 20: Distribution of kurtosis of 1000 synthetic time series log returns in blue and real Gold log return kurtosis in red.

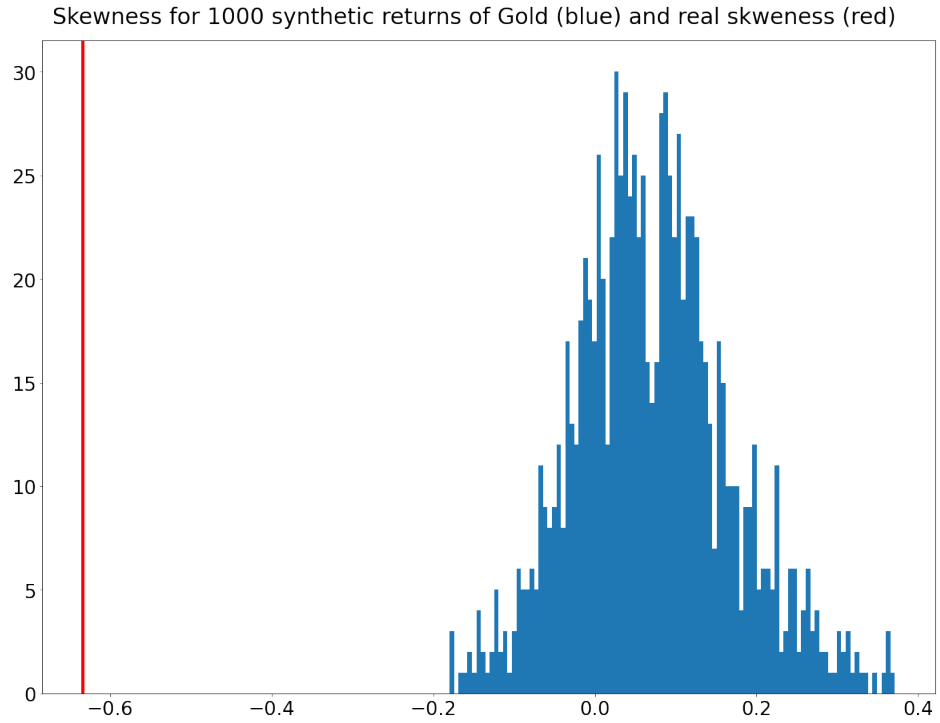

Figure 21: Distribution of skewness of 1000 synthetic time series log returns in blue and real Gold log return skewness in red.

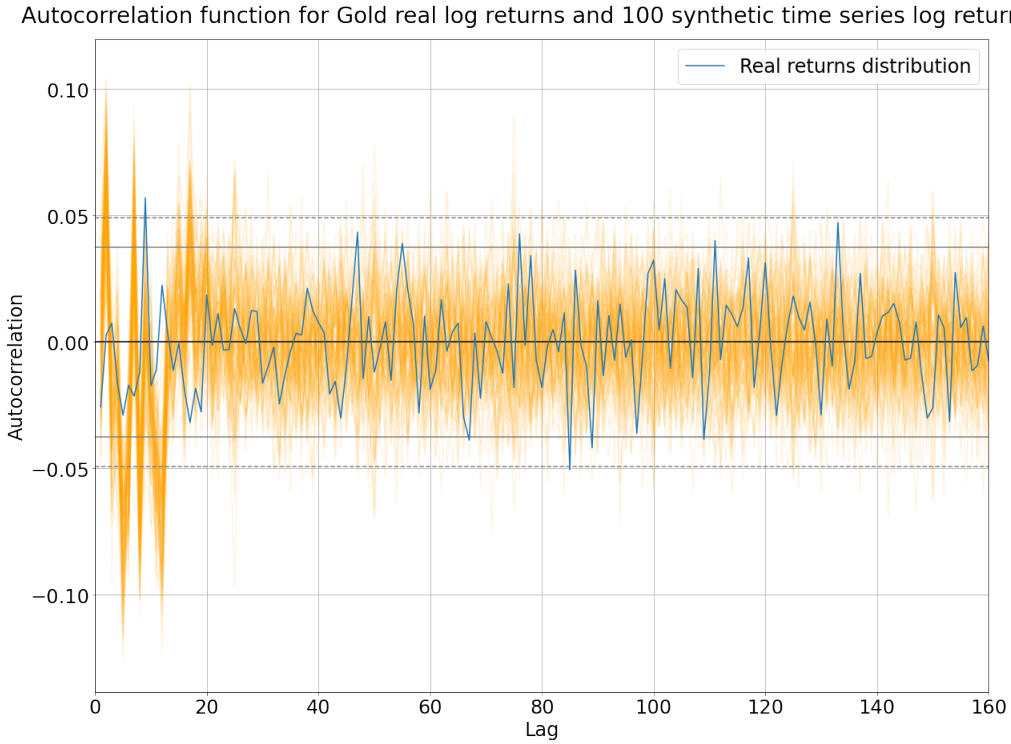

Figure 22: Autocorrelation function for Gold real log returns and 100 synthetic time series log returns.

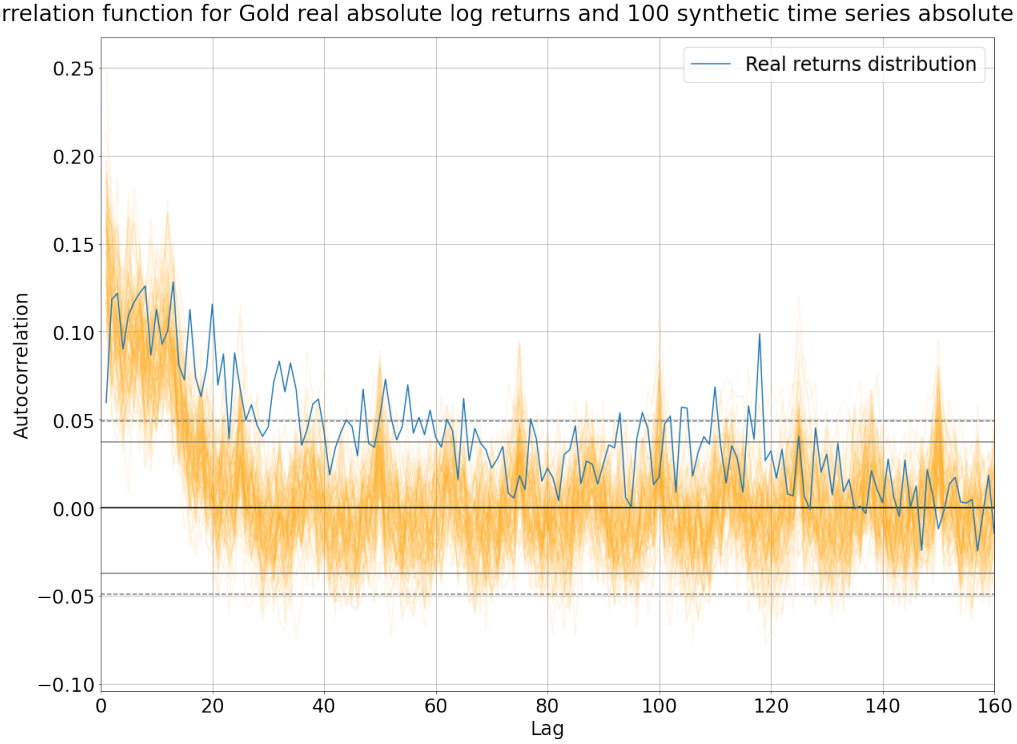

Figure 23: Autocorrelation function for absolute Gold real log returns and 100 synthetic time series absolute log returns

B EURUSD

B.0.1 Data Analysis

First the daily closing price of EURUSD from the period of 01.01.2010 to 01.04.2021, which are captured from Yahoo Finance, is shown in figure 24.

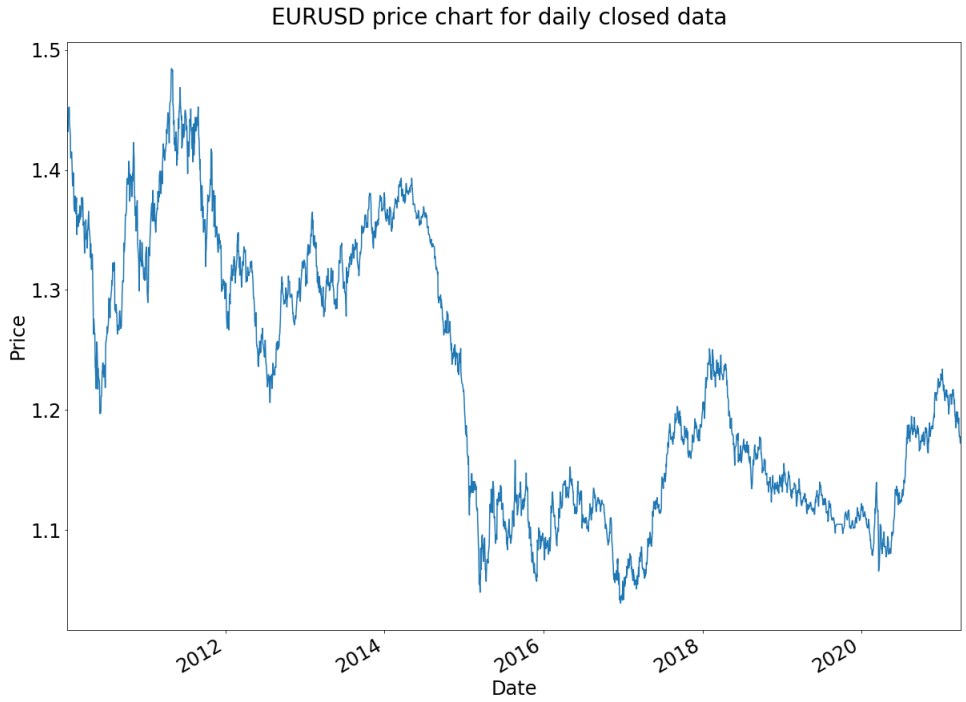

Figure 24: EURUSD daily closing price from 01.01.2010 to 01.04.2021

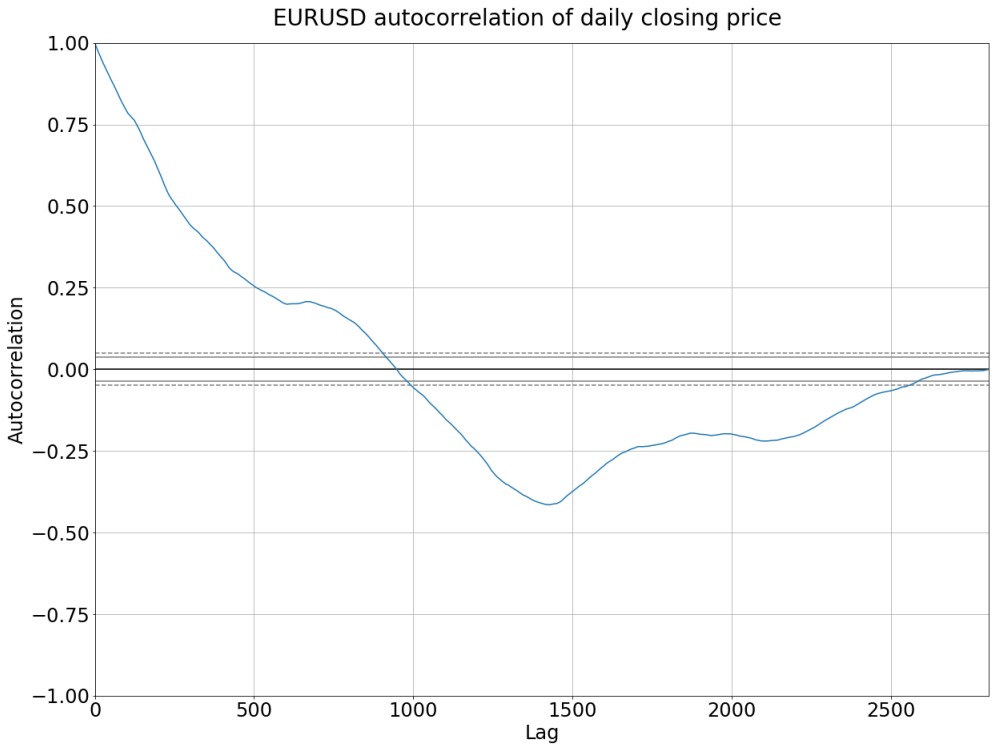

Figure 25: Autocorrelation funciton of EURUSD daily closing price from 01.01.2010 to 01.04.2021

Financial time series data is usually analyzed by the return instead of the price, to get a more stationary time series data. The log returns simply eliminate non-stationary properties of the data, making it more stable. The log returns by time is shown in figure 26.

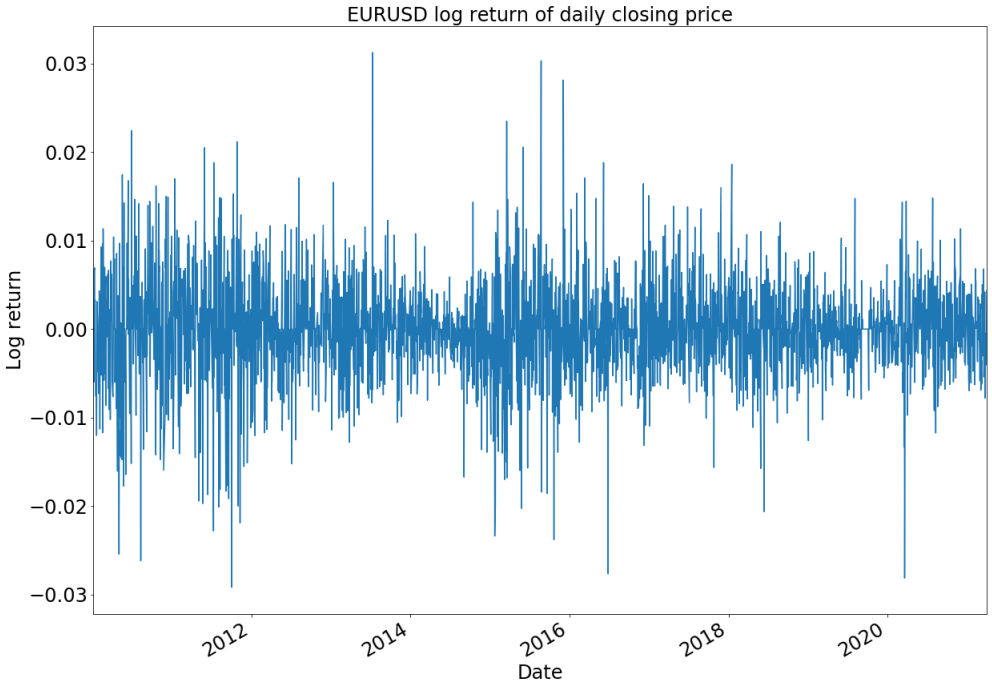

Figure 26: Log returns of EURUSD daily closing price from 01.01.2010 to 01.04.2021

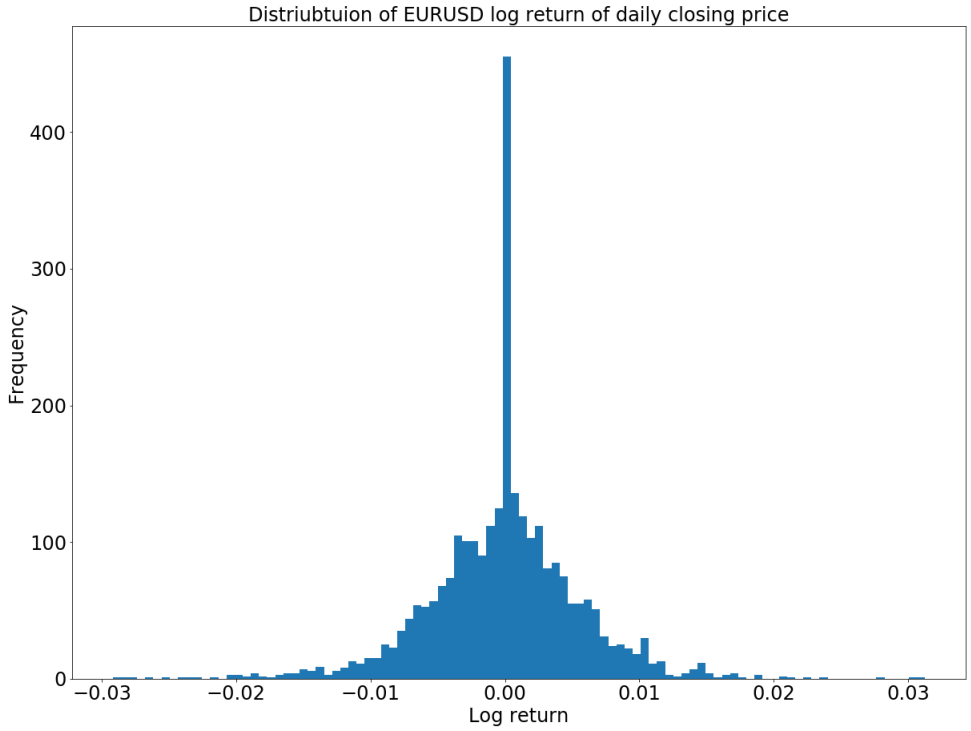

Figure 27: Histogram of log returns of EURUSD daily closing price from 01.01.2010 to 01.04.2021

B.0.2 Experiment Results

This section examines the resulting synthetic time series of the GAN when trained with EURUSD data. In Figure **??** we show the generator and discriminator loss of as a function of the number of epochs of training for the GAN.

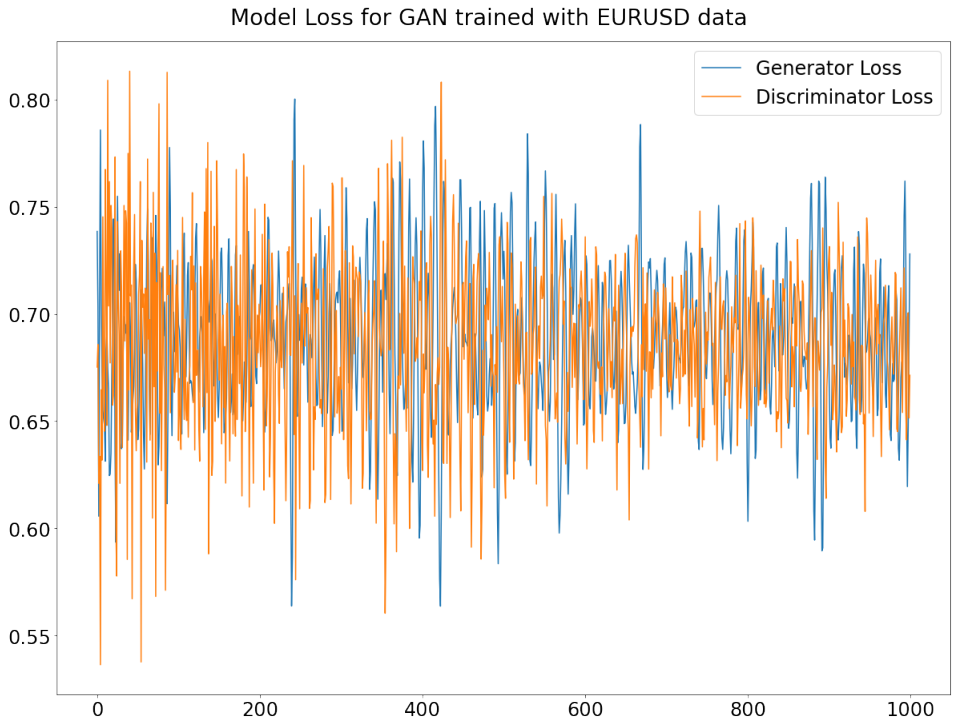

Figure 28: Generator and discriminator loss as a function of training epochs.

As the model has been trained with EURUSD time

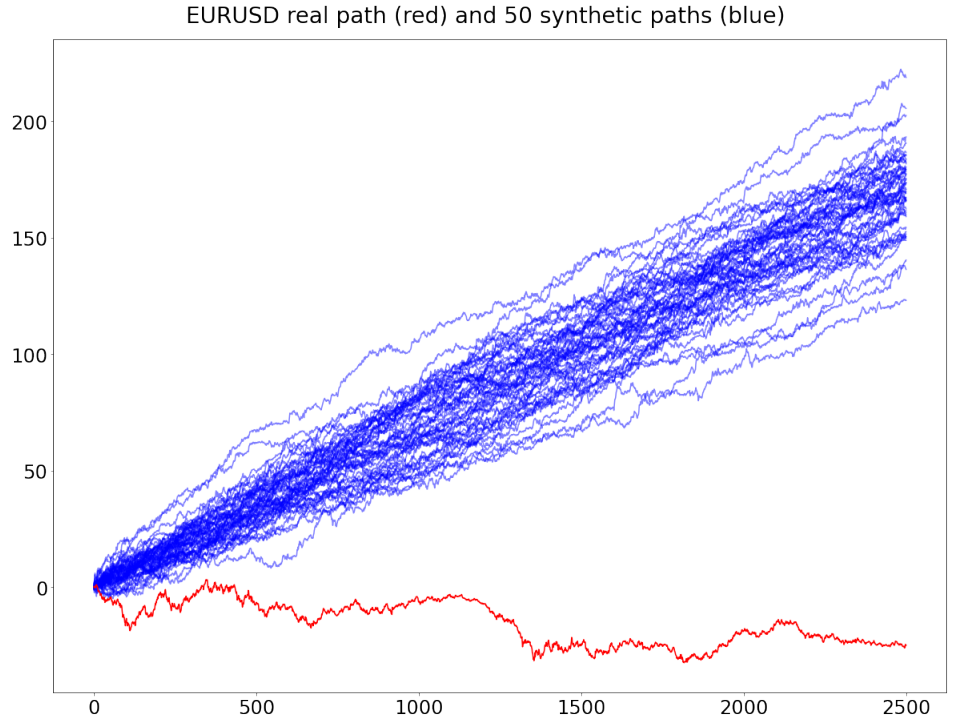

Figure 29: 50 Synthetic time series (blue) and real EURUSD time series closing price for 2500 time steps.

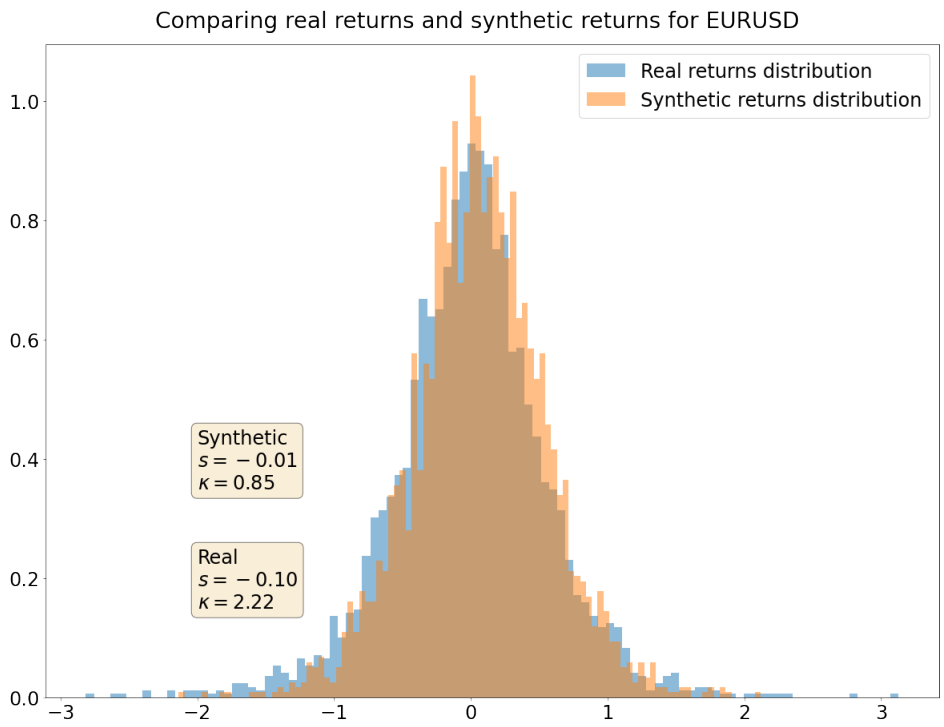

Figure 30: Histogram of log returns of the closing price for EURUSD and a synthetic time series.

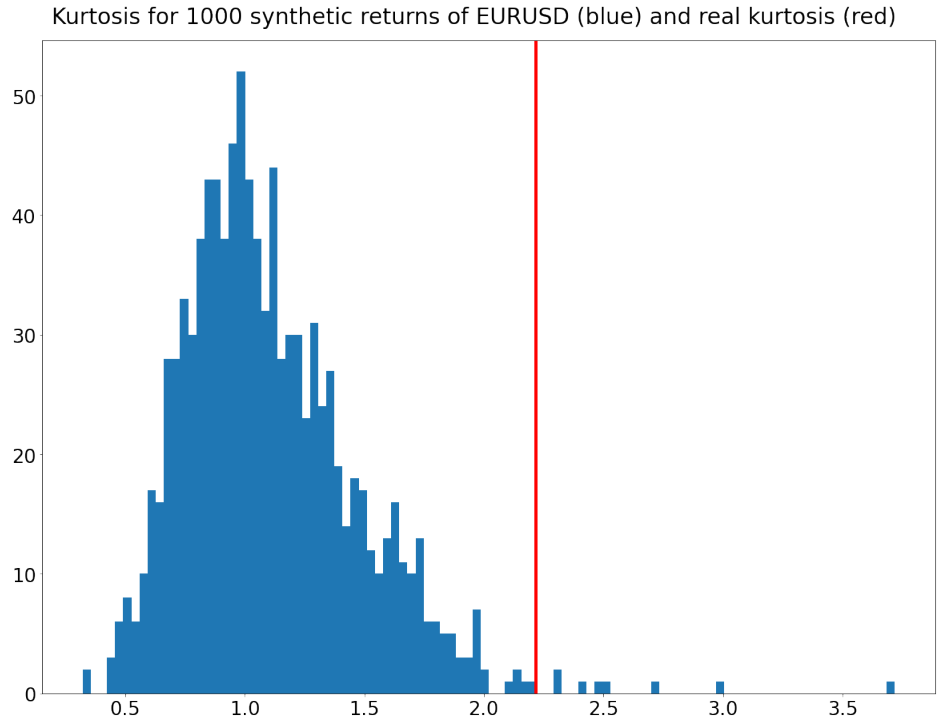

Figure 31: Distribution of kurtosis of 1000 synthetic time series log returns in blue and real EURUSD log return kurtosis in red.

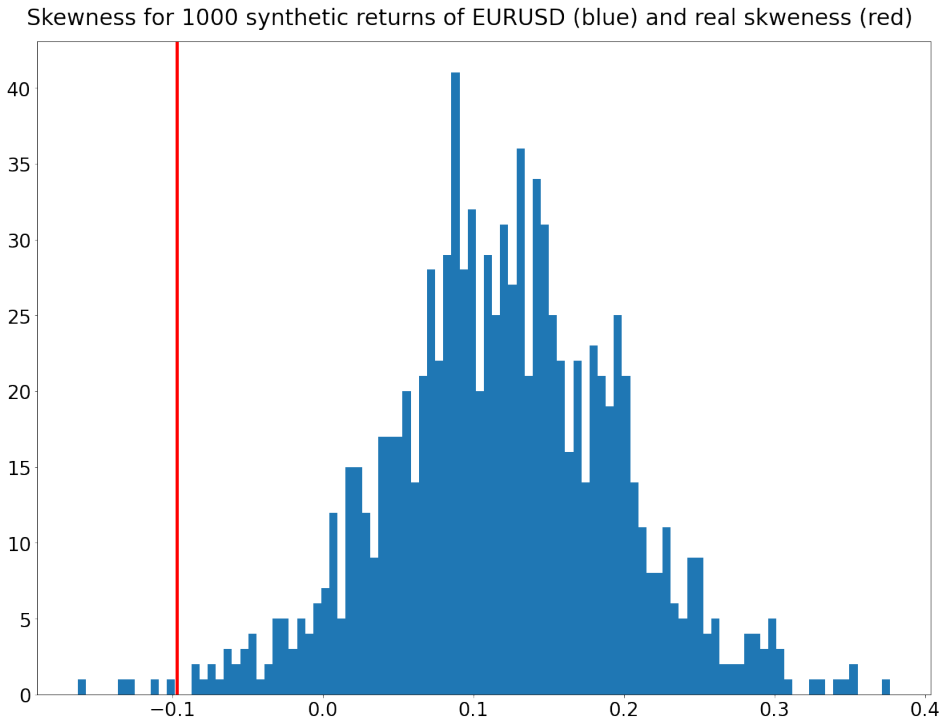

Figure 32: Distribution of skewness of 1000 synthetic time series log returns in blue and real EURUSD log return skewness in red.

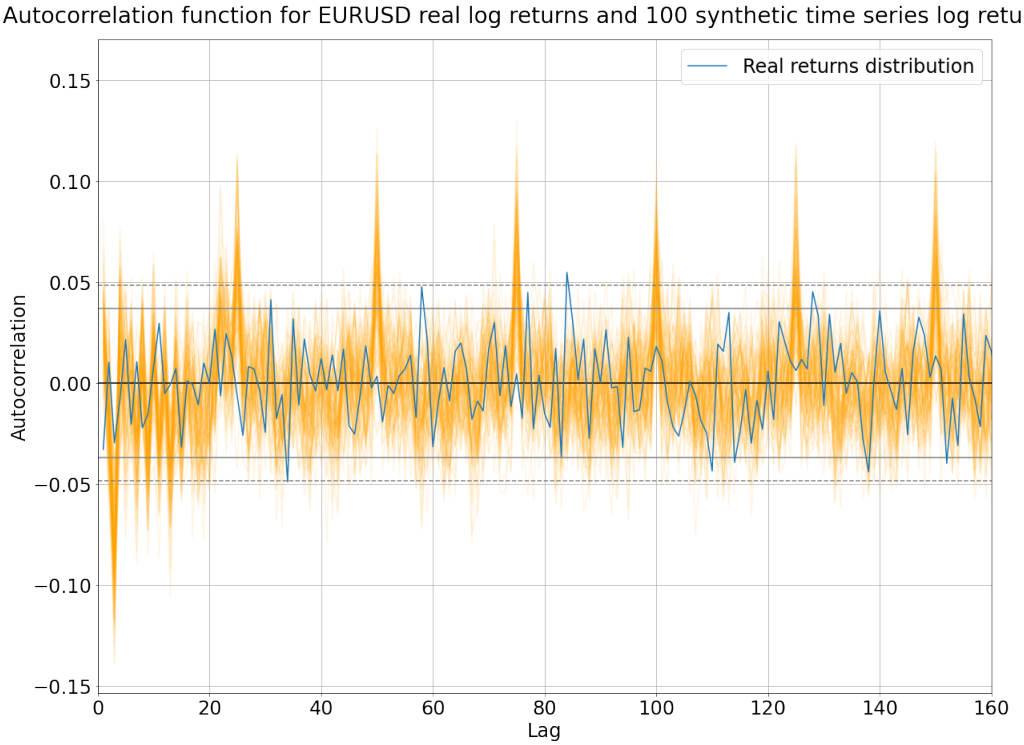

Figure 33: Autocorrelation function for EURUSD real log returns and 100 synthetic time series log returns.

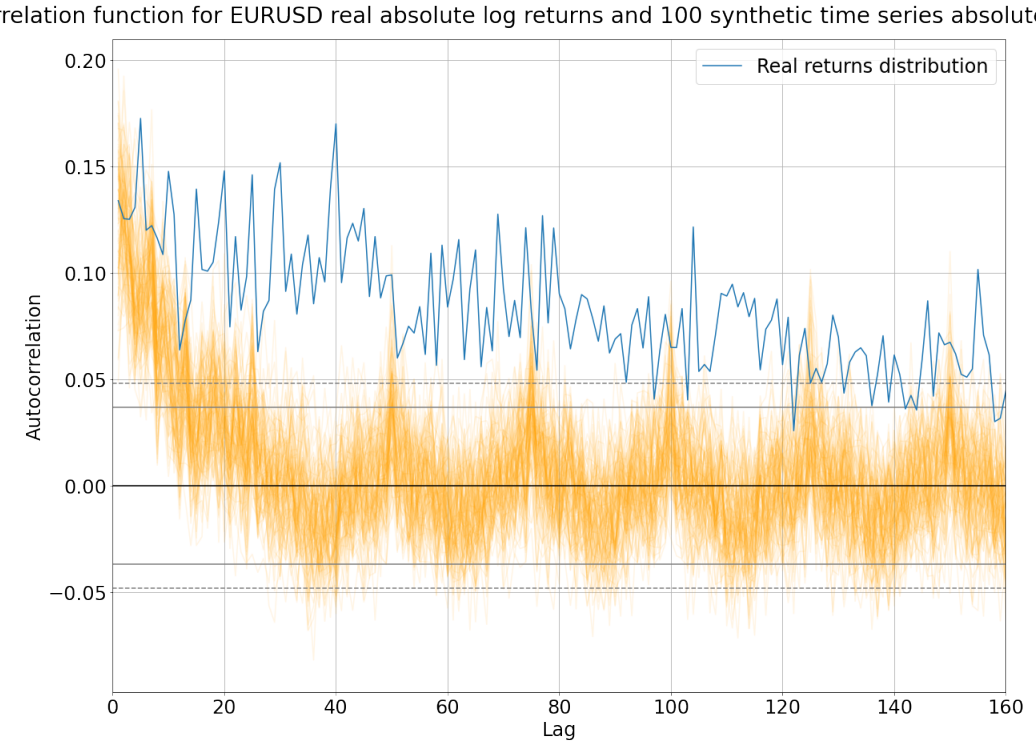

Figure 34: Autocorrelation function for absolute EURUSD real log returns and 100 synthetic time series absolute log returns

C VIX Futures

C.0.1 Data Analysis

First the daily closing price of VIX from the period of 01.01.2010 to 01.04.2021, which are captured from Yahoo Finance, is shown in figure 35.

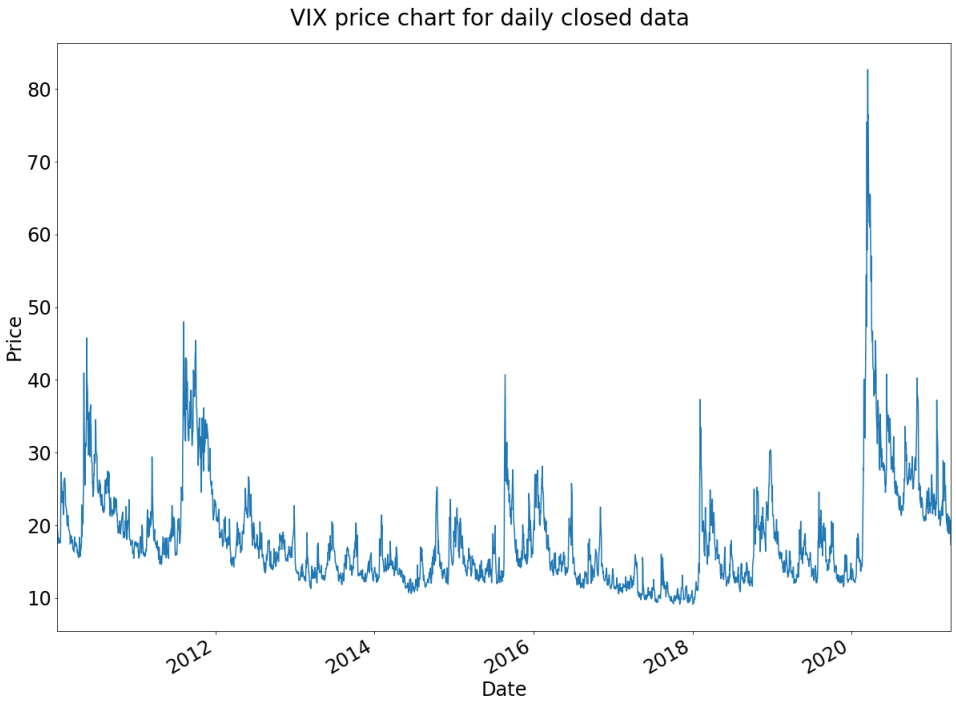

Figure 35: VIX daily closing price from 01.01.2010 to 01.04.2021

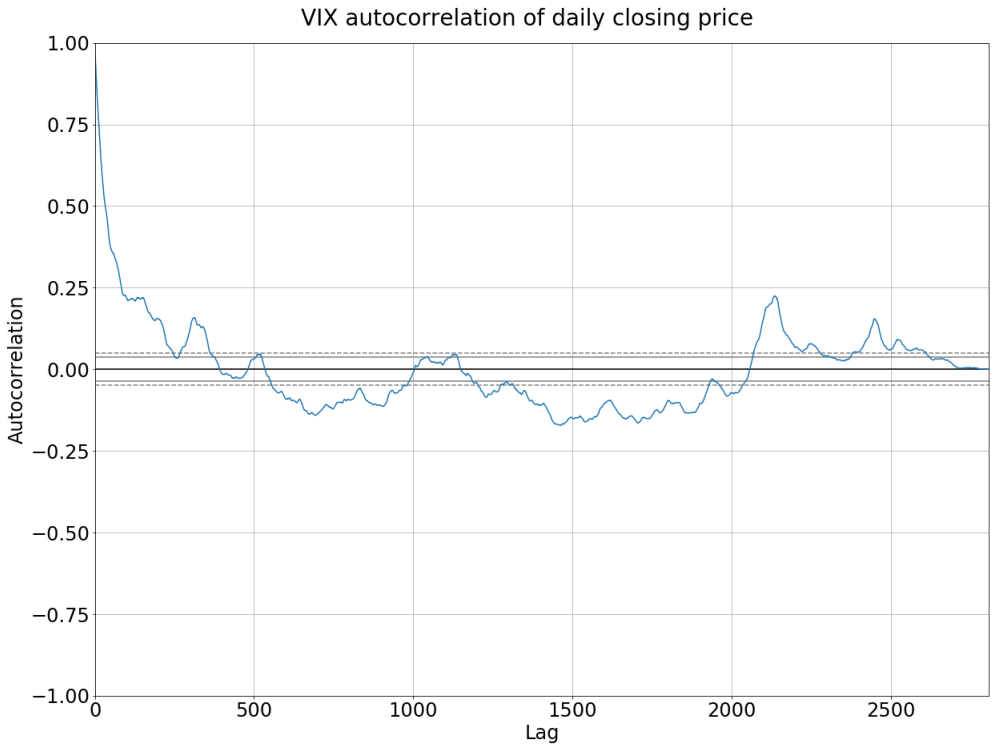

Figure 36: Autocorrelation funciton of VIX daily closing price from 01.01.2010 to 01.04.2021

Financial time series data is usually analyzed by the return instead of the price, to get a more stationary time series data. The log returns simply eliminate non-stationary properties of the data, making it more stable. The log returns by time is shown in figure 37.

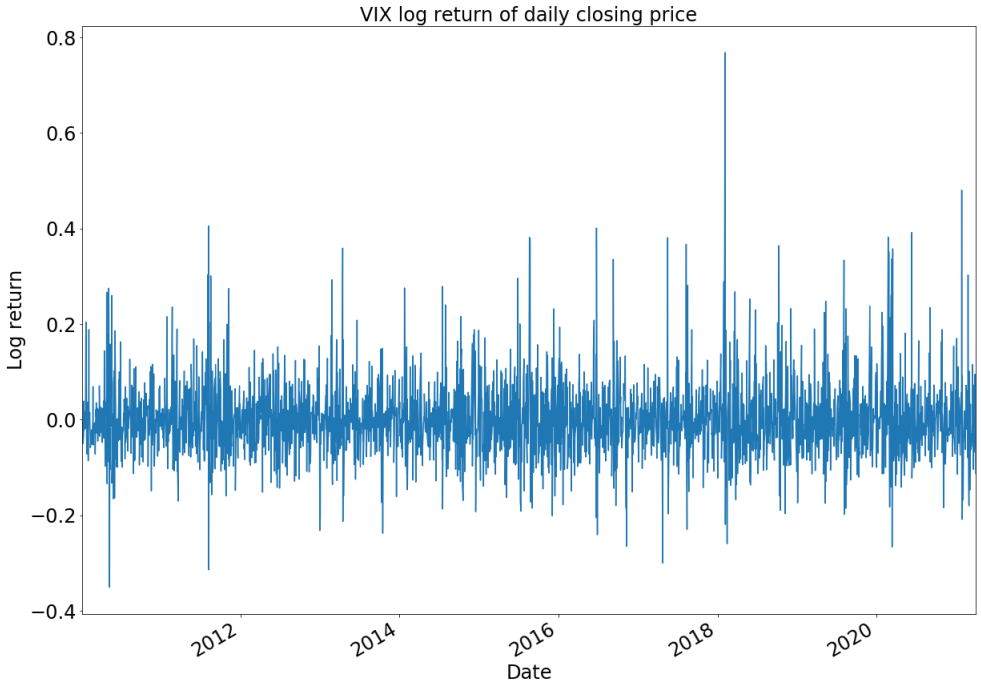

Figure 37: Log returns of VIX daily closing price from 01.01.2010 to 01.04.2021

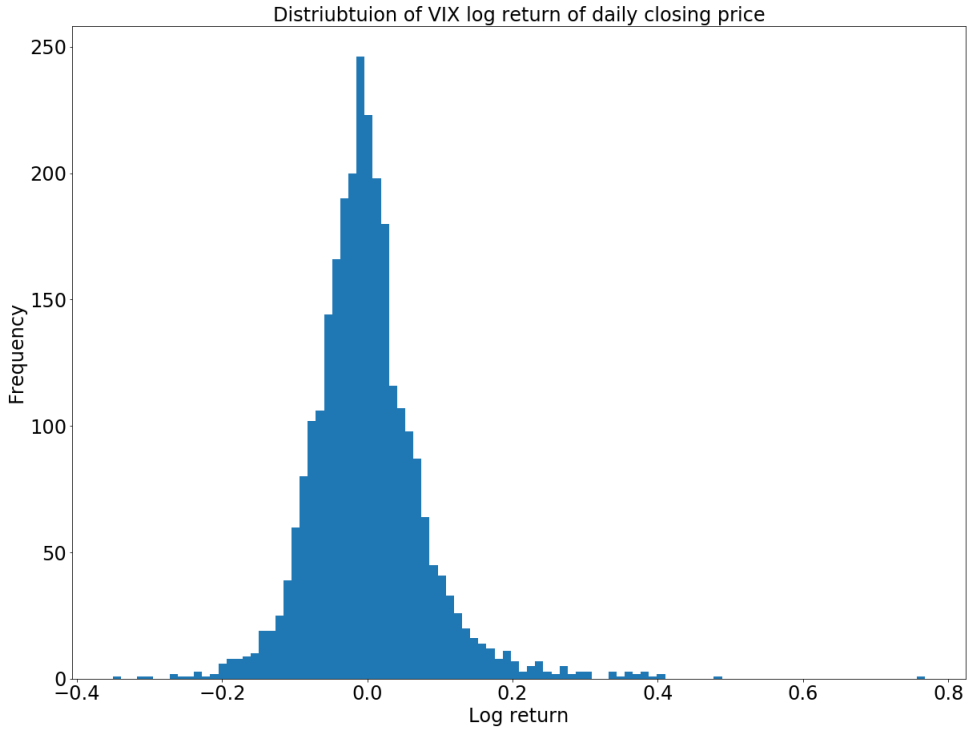

Figure 38: Histogram of log returns of VIX daily closing price from 01.01.2010 to 01.04.2021

C.0.2 Experiment Results

This section examines the resulting synthetic time series of the GAN when trained with VIX data. In Figure **??** we show the generator and discriminator loss of as a function of the number of epochs of training for the GAN.

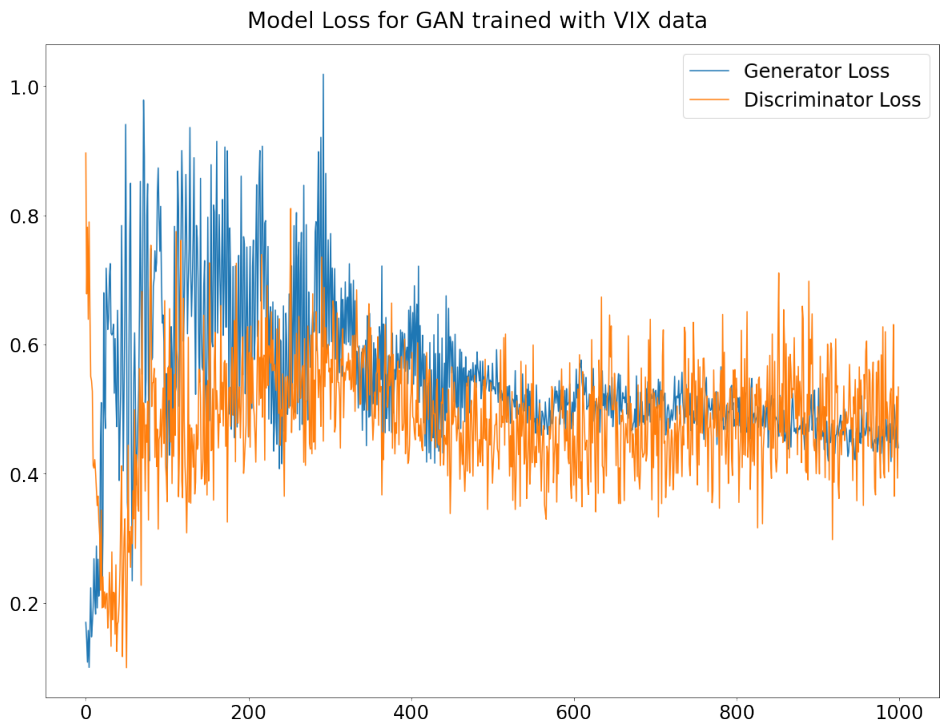

Figure 39: Generator and discriminator loss as a function of training epochs.

As the model has been trained with VIX time

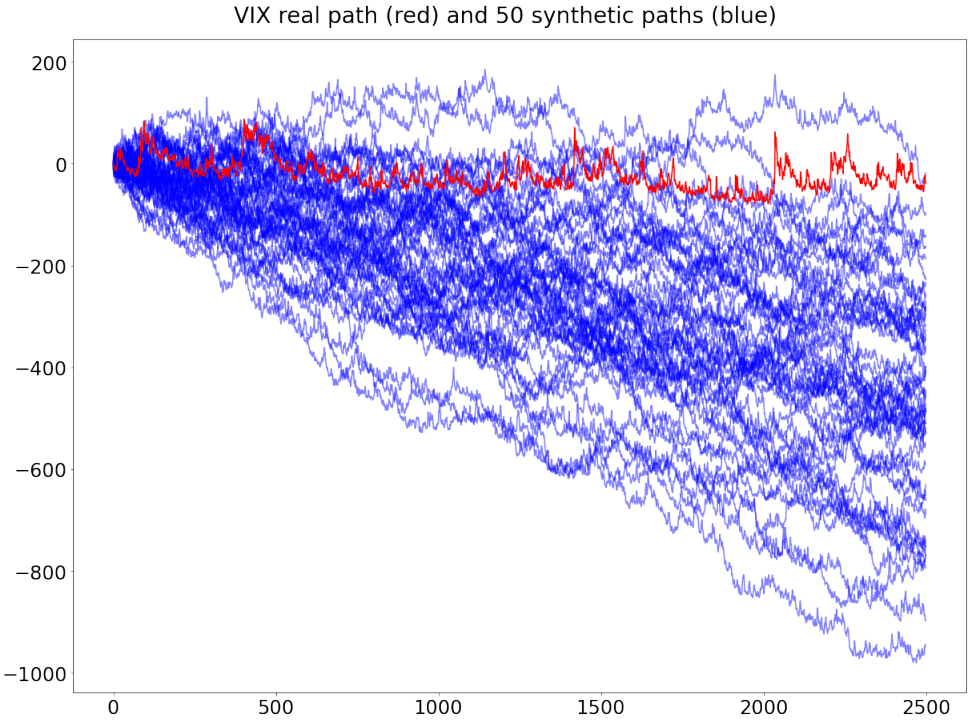

Figure 40: 50 Synthetic time series (blue) and real VIX time series closing price for 2500 time steps.

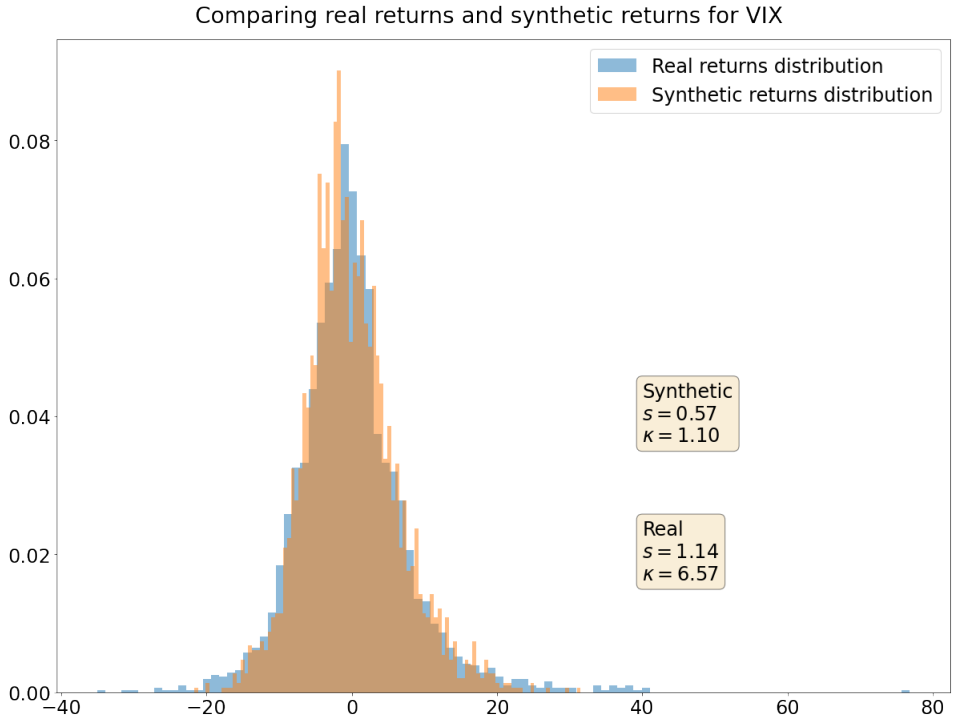

Figure 41: Histogram of log returns of the closing price for VIX and a synthetic time series.

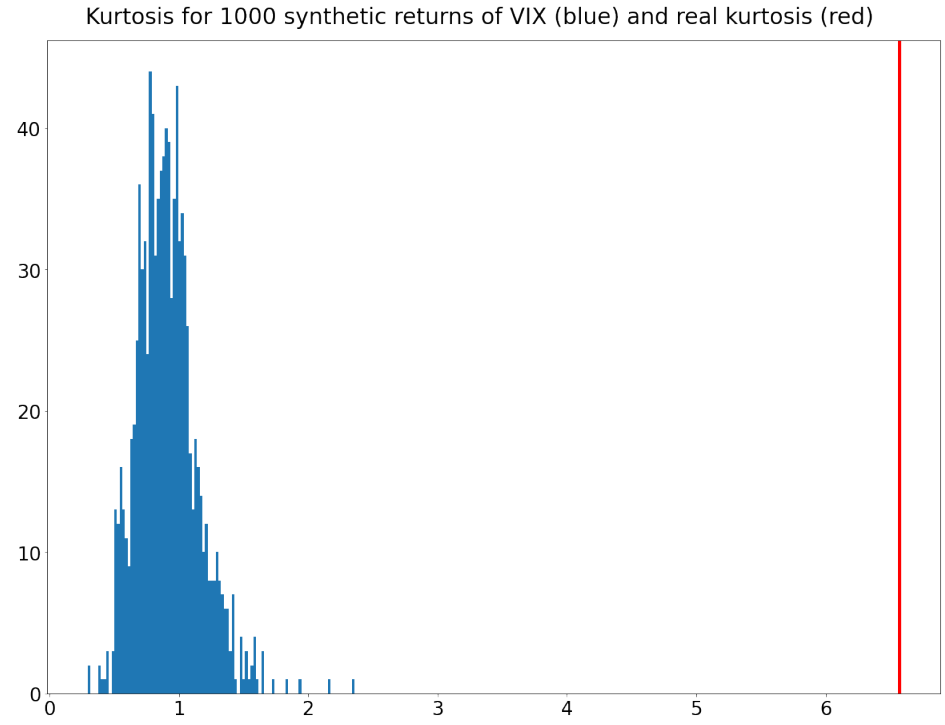

Figure 42: Distribution of kurtosis of 1000 synthetic time series log returns in blue and real EURUSD log return kurtosis in red.

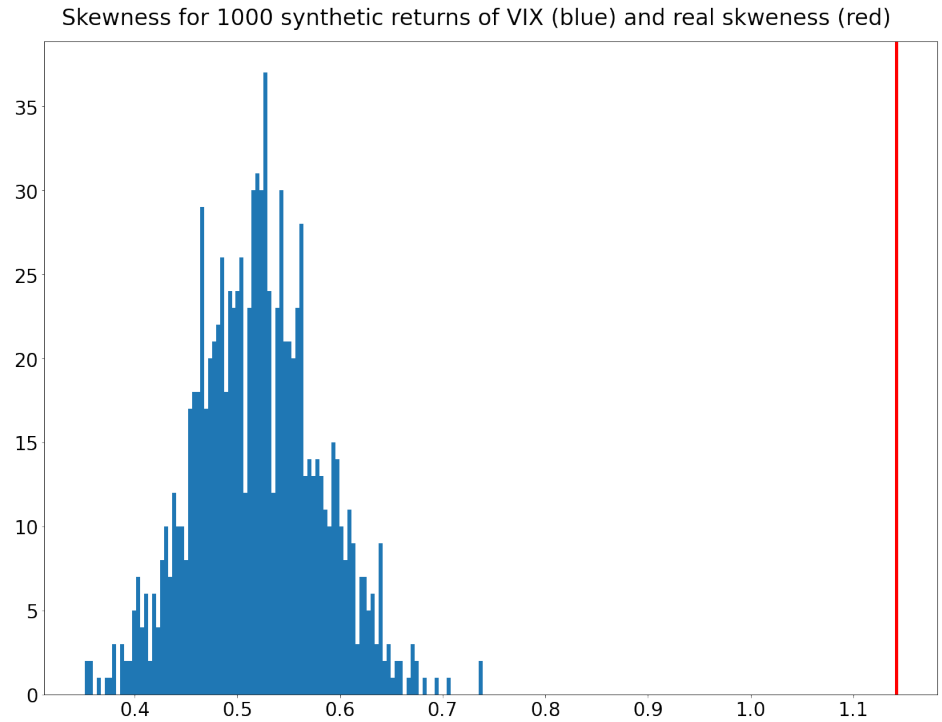

Figure 43: Distribution of skewness of 1000 synthetic time series log returns in blue and real VIX log return skewness in red.

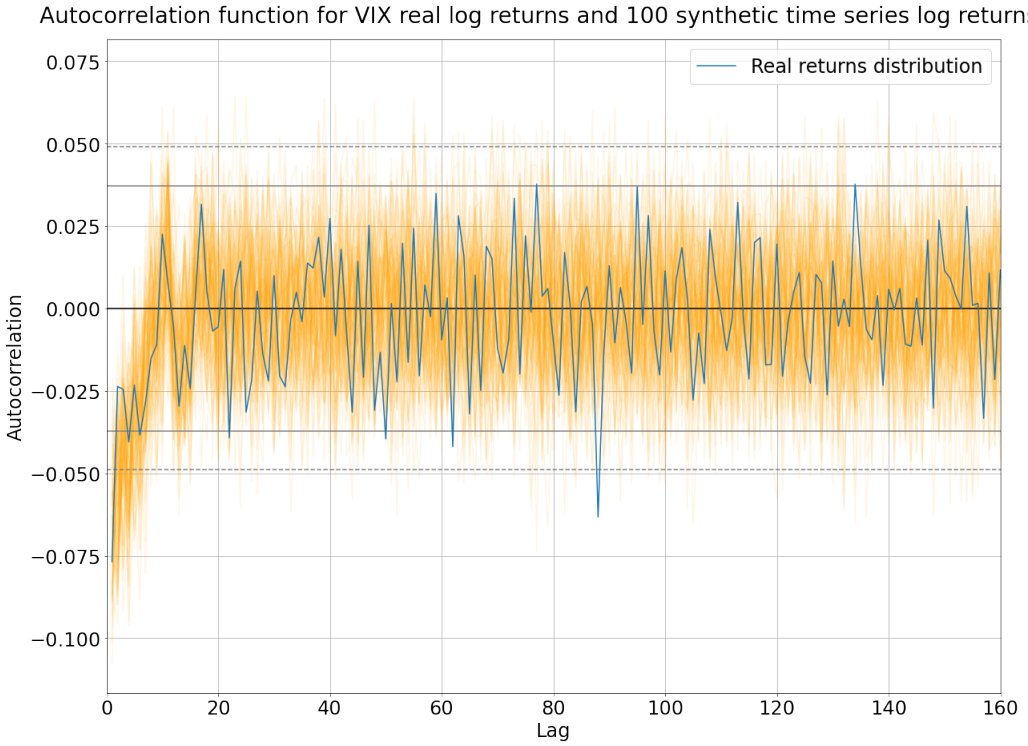

Figure 44: Autocorrelation function for VIX real log returns and 100 synthetic time series log returns.

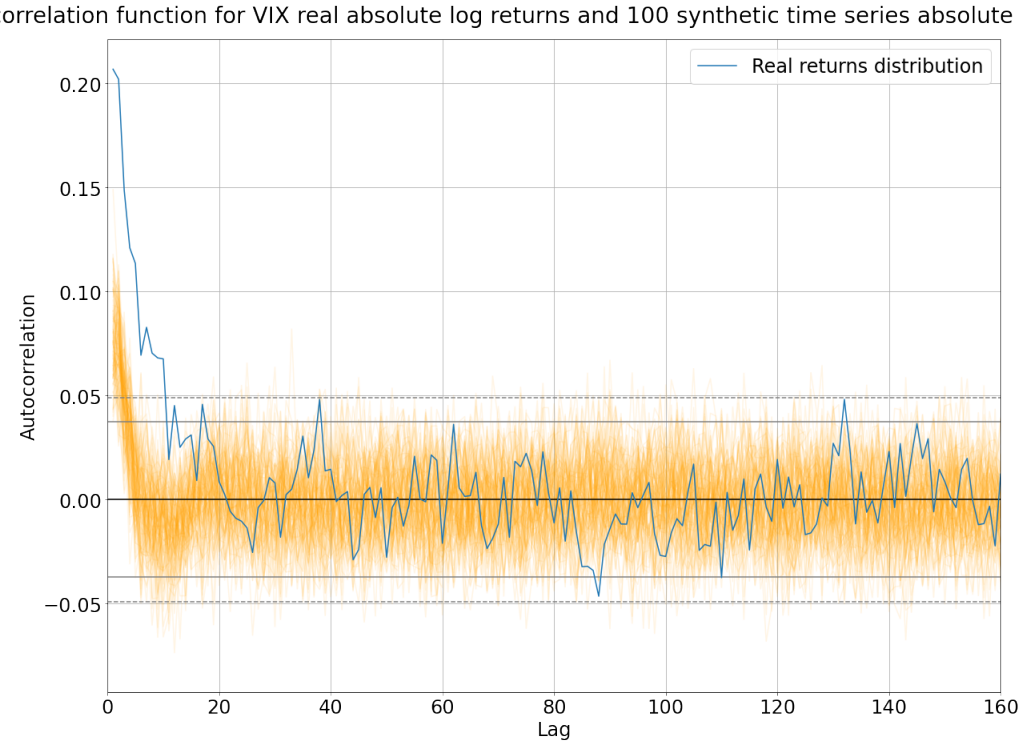

Figure 45: Autocorrelation function for absolute VIX real log returns and 100 synthetic time series absolute log returns

D S&P 500 Index

D.0.1 Data Analysis

First the daily closing price of S&P500 from the period of 01.01.2010 to 01.04.2021, which are captured from Yahoo Finance, is shown in figure 46.

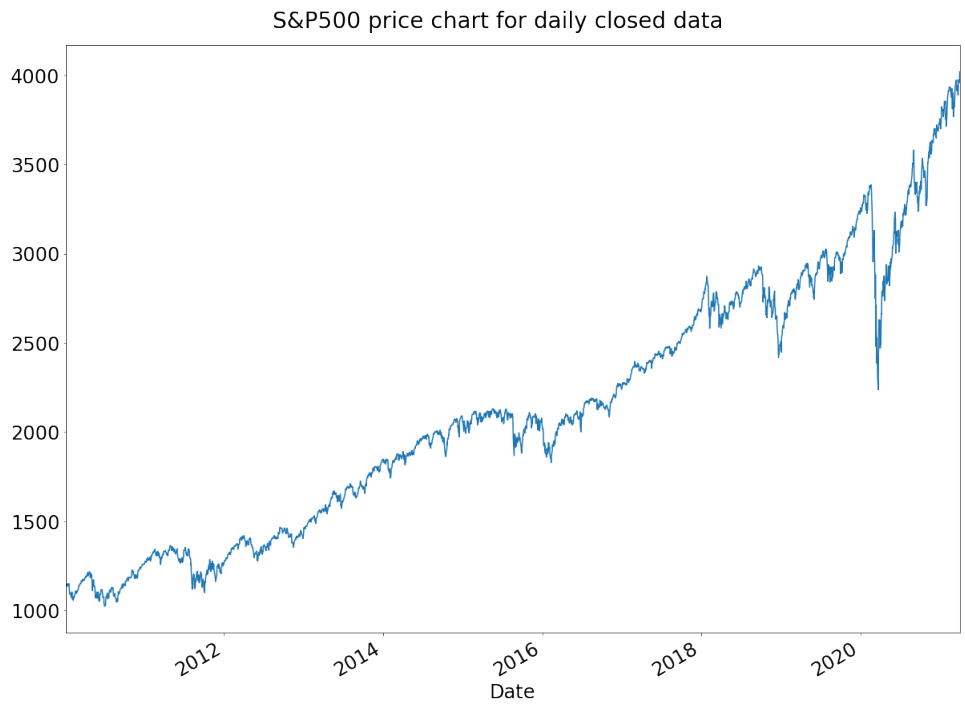

Figure 46: S&P500 daily closing price from 01.01.2010 to 01.04.2021

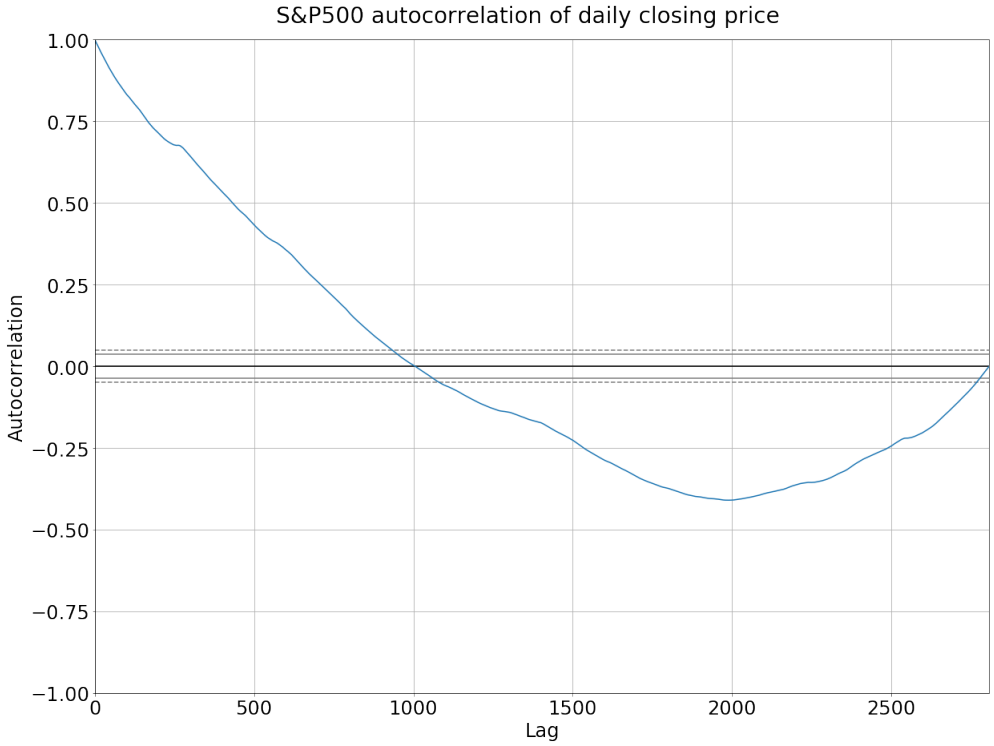

Figure 47: Autocorrelation funciton of S&P500 daily closing price from 01.01.2010 to 01.04.2021

Financial time series data is usually analyzed by the return instead of the price, to get a more stationary time series data. The log returns simply eliminate non-stationary properties of the data, making it more stable. The log returns by time is shown in figure 48.

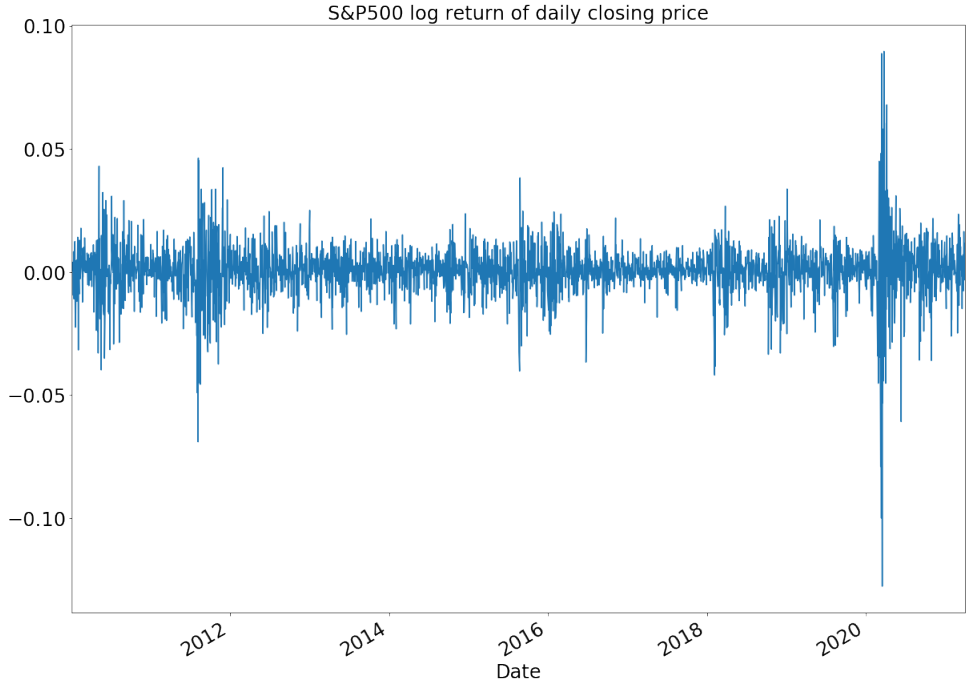

Figure 48: Log returns of S&P500 daily closing price from 01.01.2010 to 01.04.2021

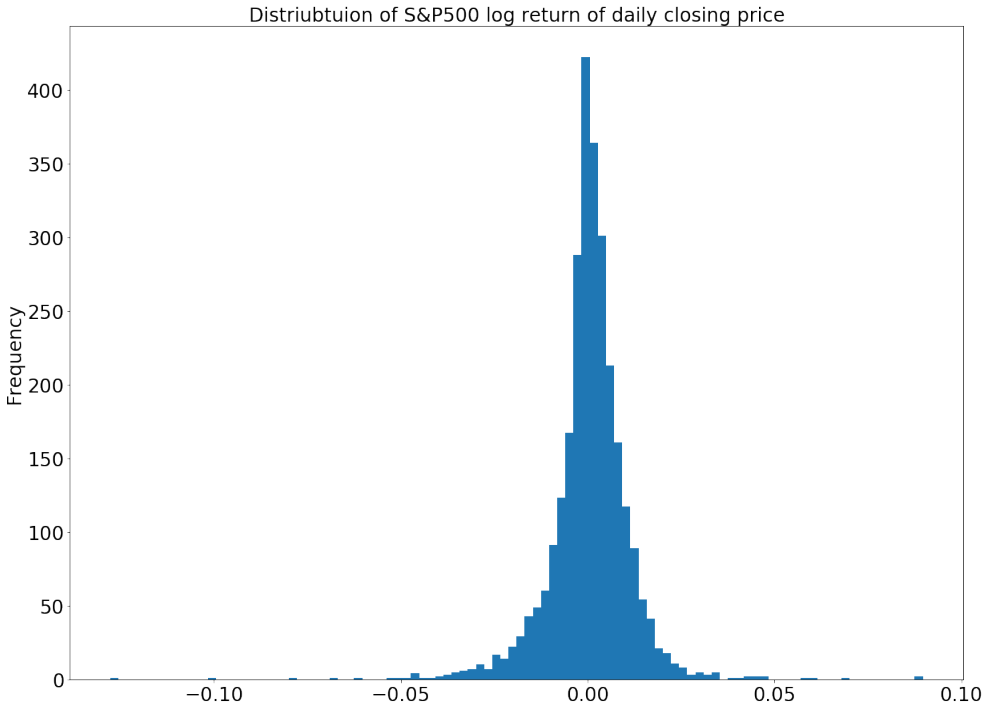

Figure 49: Histogram of log returns of S&P500 daily closing price from 01.01.2010 to 01.04.2021

D.0.2 Experiment Results

This section examines the resulting synthetic time series of the GAN when trained with S&P500 data. In Figure **??** we show the generator and discriminator loss of as a function of the number of epochs of training for the GAN.

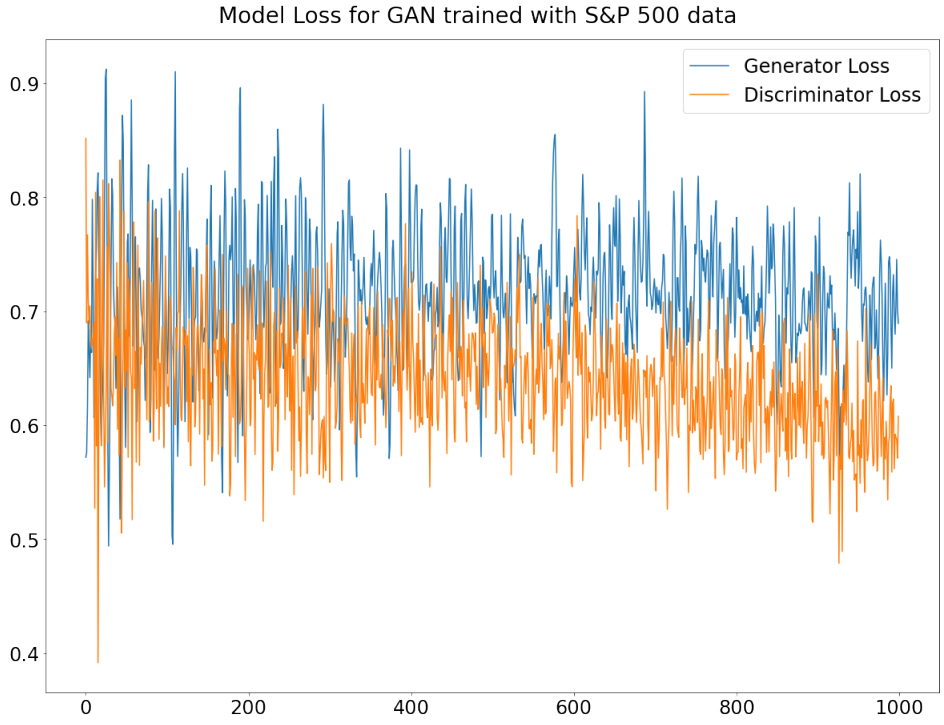

Figure 50: Generator and discriminator loss as a function of training epochs.

As the model has been trained with S&P500 time

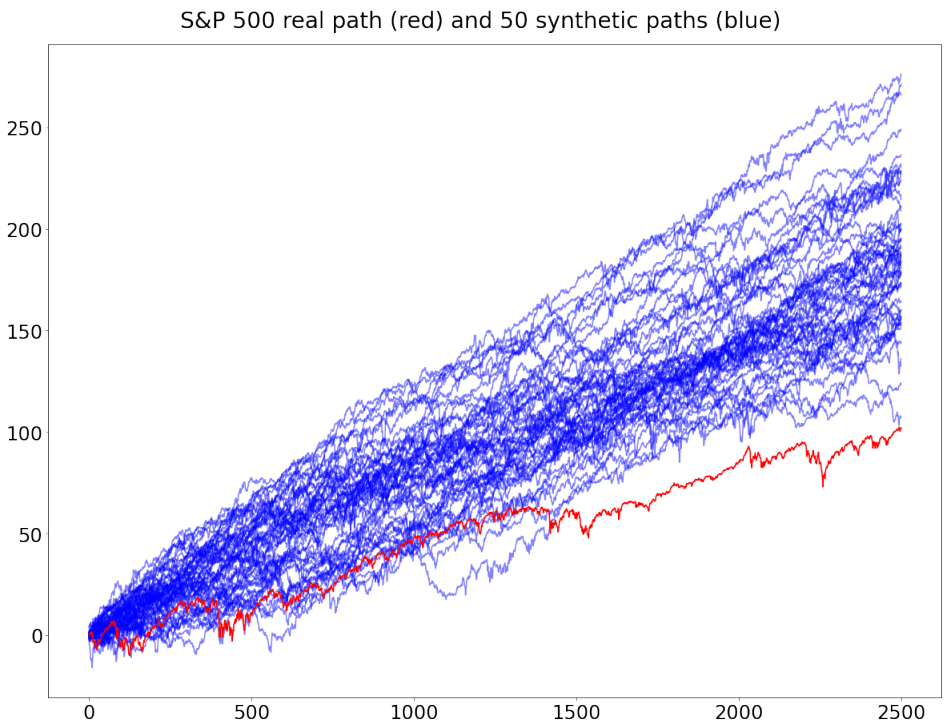

Figure 51: 50 Synthetic time series (blue) and real S&P500 time series closing price for 2500 time steps.

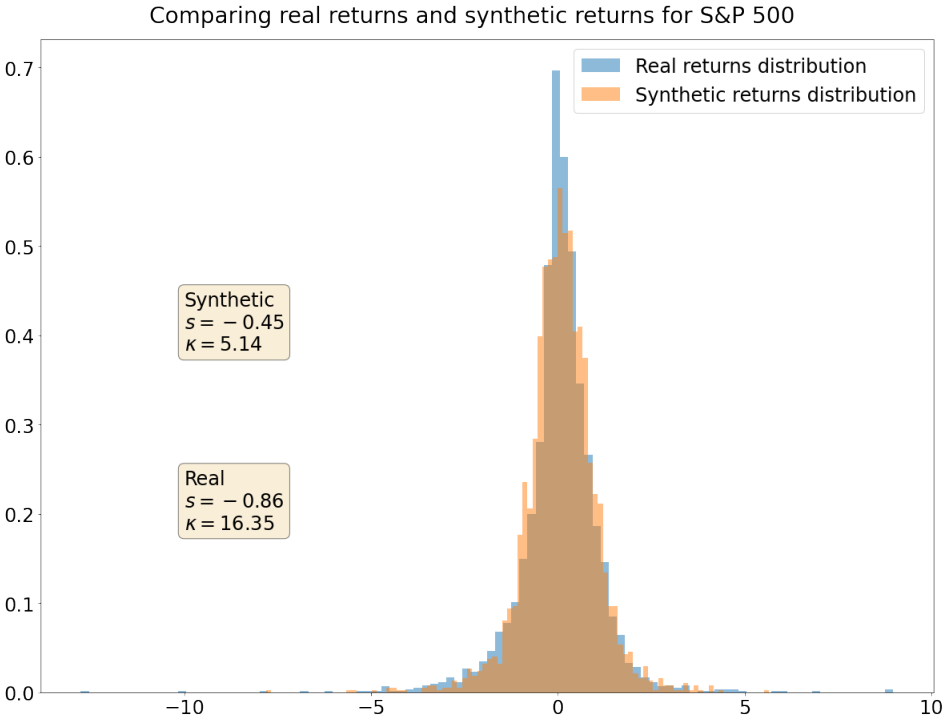

Figure 52: Histogram of log returns of the closing price for S&P500 and a synthetic time series.

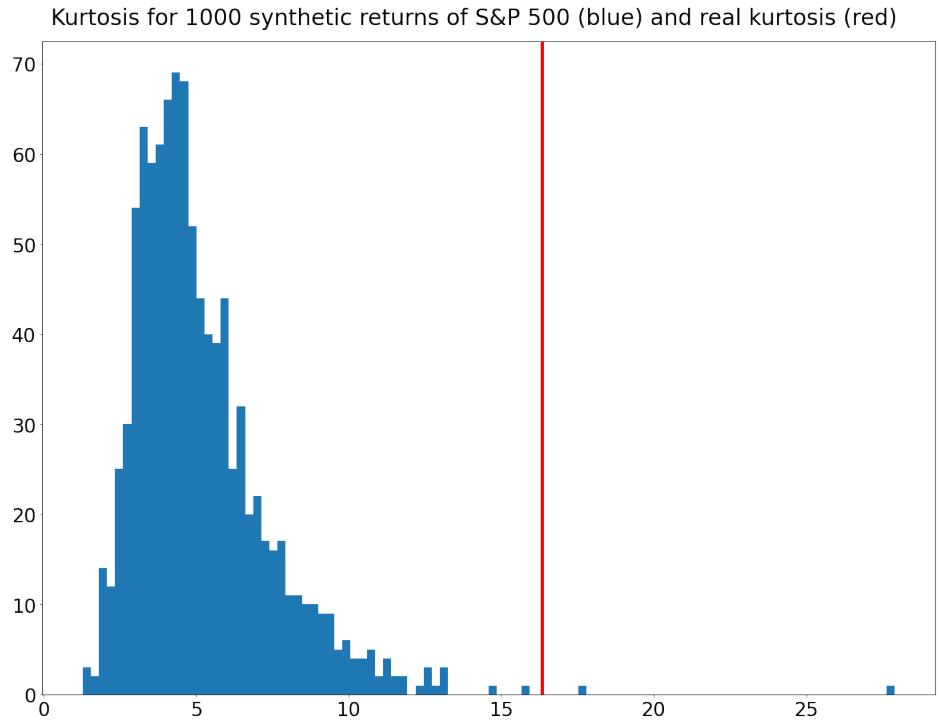

Figure 53: Distribution of kurtosis of 1000 synthetic time series log returns in blue and real S&P500 log return kurtosis in red.

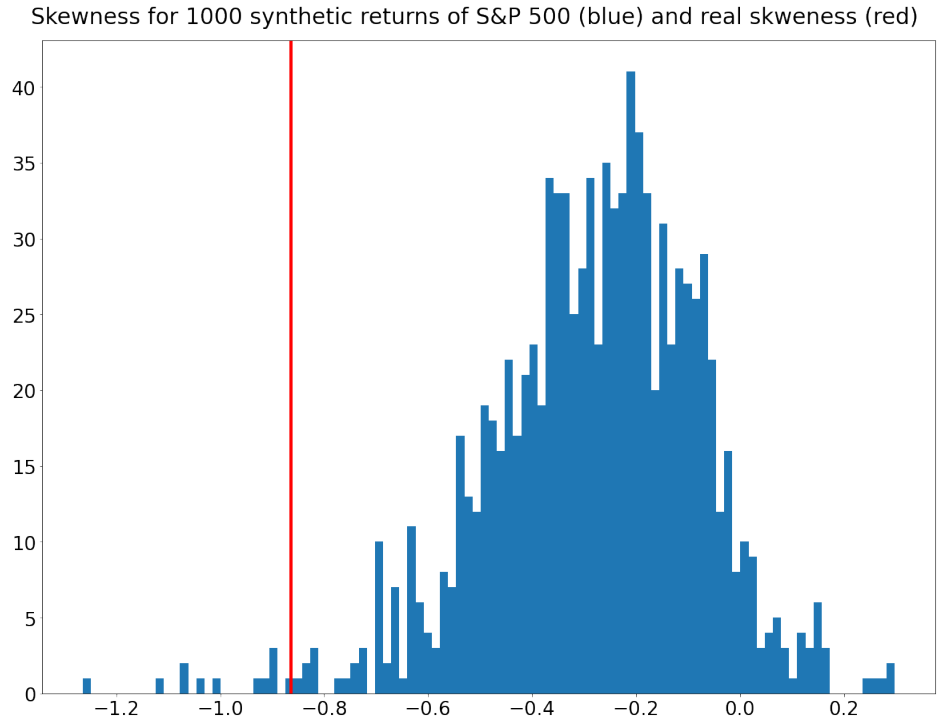

Figure 54: Distribution of skewness of 1000 synthetic time series log returns in blue and real S&P500 log return skewness in red.

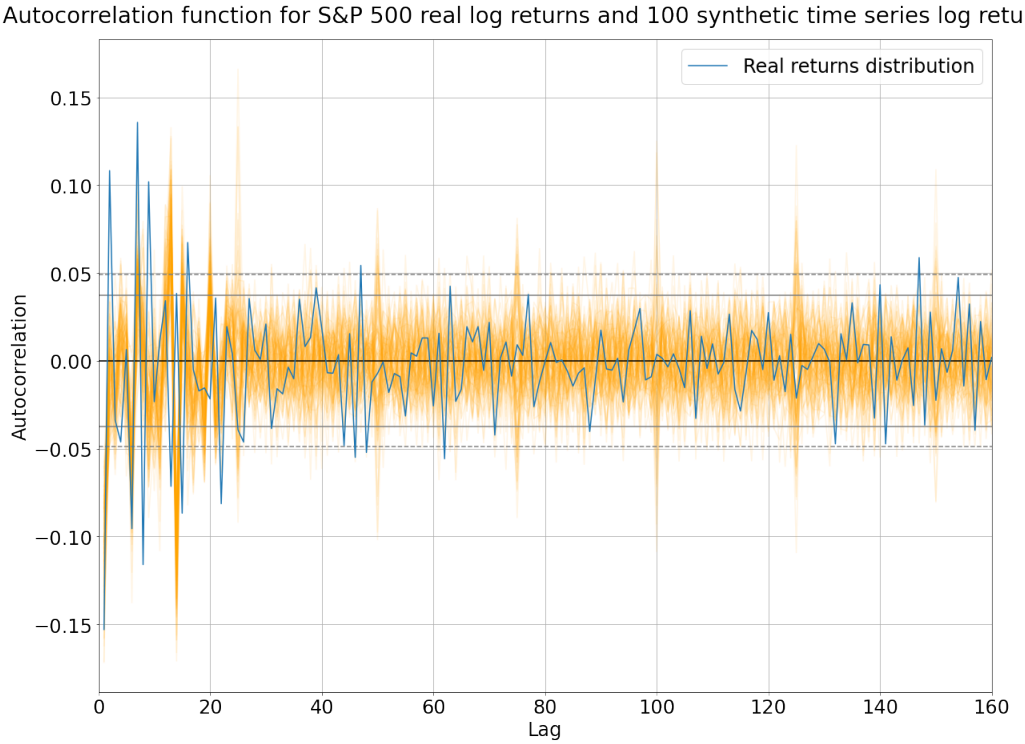

Figure 55: Autocorrelation function for S&P500 real log returns and 100 synthetic time series log returns.

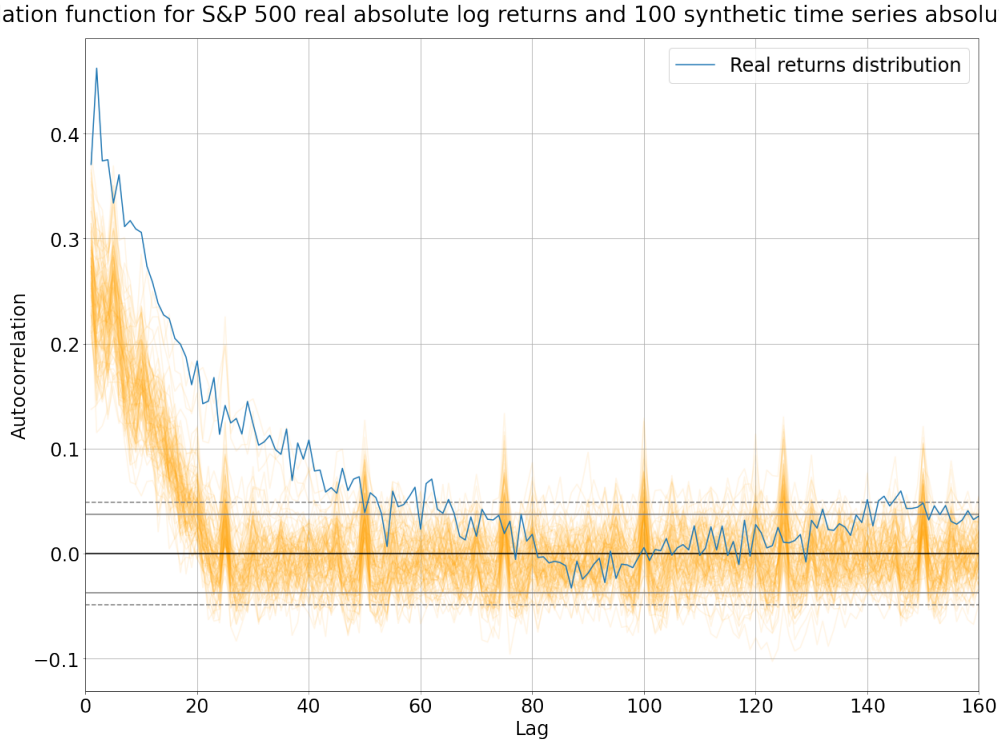

Figure 56: Autocorrelation function for absolute S&P500 real log returns and 100 synthetic time series absolute log returns