Università degli Studi di Padova

final report for *Neural Networks and Deep Learning*
AY 2022/23 - master degree in Physics of Data
Padua, 26 Jan 2022

Financial Time-series Modeling Using Wasserstein Generative Adversarial Networks (WGANs)

Amjadi Bahador[*], Martemucci Walter[†]
Dipartimento di Fisica e Astronomia "Galileo Galilei"
University of Padua
Email: [*]bahador.amjadi@gmail.com, [†]walter.martemucci@studenti.unipd.it,

Abstract

Financial time-series modeling has been a demanding problem since one should consider the convoluted and stochastic entity of financial processes. Most of the applications of deep neural networks focus on price or volatility prediction but our goal is to utilize *Generative Adversarial Networks (GANs)* to build a model which generates realistic financial time series. As a criterion of how much these generated time series resemble the real data, one can check whether the model is able to reproduce a set of well-known statistical characteristics of real financial data, or so-called *Stylized Facts* IV-A. Stylized facts are robust empirical regularities one can observe when studying financial data. They are key indicators of economic behavior and can provide valuable insights into the underlying mechanisms that drive the economy. By implementing a *Wasserstein Generative Adversarial Network*(WGAN) and using different models for the generator and discriminator, we are able to, up to some degree, reproduce the stylized facts. To further work on this route, one can consider expanding the database in terms of using more significant features. Deeper networks can also be implemented hoping to achieve a higher degree of similarity between generated and real financial data.

***Index Terms*—Generative Adversarial Network, Deep Learning, Stylized Facts**

I Introduction

Studying financial markets using quantitative approaches requires sufficient care and attention. Especially when someone wants to design a model to predict future values. In terms of physical sciences, this issue is due to the so-called *complex* entity of such processes. Processes in which stochasticity plays a central role and a proper model is a model that can imitate the behaviors originating from this stochasticity.

In this report, we discuss the implementation of a *Wasserstein Generative Adversarial Network (WGAN)* to model the S&P500 index. The main idea is to implement models which are able to produce so-called stylized facts. stylized facts are well-known statistical characteristics of financial time series which are fully discussed in section IV-A.

The WGAN proposed in this paper will be tested under three different models for the generator and the discriminator; as can seen in figure 4.

A summary of the main contributions carried out in this report is brought as the following:

- *Using WGAN for Generation of financial time Series: We use three different models for the generator and discriminator of the WGAN model and train different configurations using stock market data in order to produce financial time series*
- *Reproducing the Stylized Facts: As mentioned before, stylized facts are the statistical features of financial time series and as the main goal of this report we try to generate financial time series that resembles the real data thus preserving the stylized facts*

The report is organized as follows. In Section III we demonstrate a high-view diagram of the main processes carried out in the report which will be discussed in detail in section IV. The theoretical background of our work, alongside the description of the models we use, is brought in IV. We present our results with the corresponding analysis in section V and finally, we conclude what is done in the whole paper in part VI.

II Related Works

A large literature of financial time series models exists ranging from a variety of discrete-time GARCH-inspired models *Bollerslev, 1986*[1] to models in continuous-time such as Black and *Scholes, 1973* [2] and *Heston, 1993* [3].

However, the development and innovation of new models are difficult: it took many years to extend the Black-Scholes model, which assumes that asset prices can be described through geometric Brownian motions, to the more sophisticated Heston model, which accounted for stochastic volatility and the leverage effect.

Concerning stock market modeling and forecasting, there is a comprehensive literature of which some of the more related works will be mentioned as the following.

First, there is the research by *Koshiyama et al, 2019* [4] in which they try to calibrate trading strategies using conditional generative adversarial networks(cGANs).

Speaking of using CNNs for financial time-series prediction we have the work by *Jou-Fan Chen et al, 2016* [5] in which they probe the possibility of utilizing a CNN in the area of trading, more precisely, to enhance the *Algorithmic Trading* framework.

Additionally, the research of *Xingyu Zhou et al, 2018* [6] in which a generic framework of Long Short-Term Memory, alongside a CNN is proposed for training to forecast the high-frequency stock market.

Another utilization of GAN to model financial time-series is [7] by *Shuntaro Takashi et al, 2019* They present a deep neural network-based approach and generative adversarial network (GANs) for financial times series modeling.

We should also mention the work, *Antonio Rosolia et al, 2021* [8] in which they implement a GAN and apply it to various financial time-series for different stock classes, hoping their model reflects all the statistical features of financial time-series like heavy-tailed price, return distributions, volatility clustering, and Coarse-fine volatility correlation.

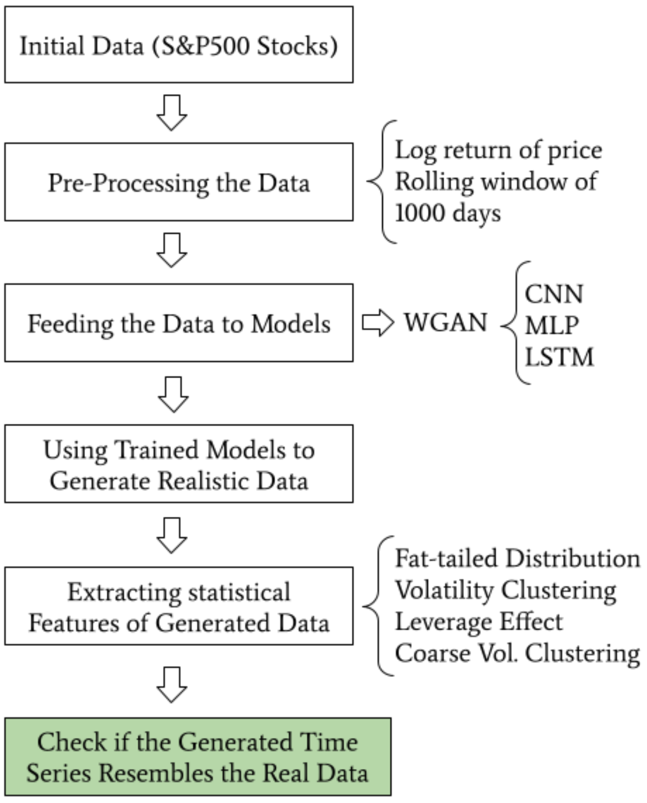
There is also the study [9] by *Magnus Wiese1 et al, 2019.* They introduce *Quant GANs.* They use temporal convolutional networks(TCNs) as their GAN's discriminator. By doing so they are able to capture long-range dependencies such as the presence of *volatility clusters.*

The main framework of the current report is based on [10] in which *Fernando de Meer Pardo* tries to Fernando de Meer Pardo build a framework in which he explores the generating capabilities of GANs when applying them to financial time series. In other words, there it is investigated whether or not GANs are able to generate realistic financial time series. In the current research, we try to use a similar framework but use different architectures and some variations to obtain plausible results and perform proper statistical analysis under the topic of stylized facts.

III Processing Pipeline

As we can see, figure 1 is a demonstration of the main parts and processes carried out in our paper. We start by pre-processing the S&P500 stock data as explained in part IV-B. Then train three different models for each generator and discriminator. These three models are further described in section IV. The time series generated using these models are then qualitatively examined to see whether they resemble the real data and thus, represent stylized facts or not.

Figure 1: A high-view Diagram of the Processes Carried in the Report


Initial Data (S&P500 Stocks)
Pre-Processing the Data
Log return of price
Rolling window of 1000 days
Feeding the Data to Models
WGAN
CNN
MLP
LSTM
Using Trained Models to Generate Realistic Data
Extracting statistical Features of Generated Data
Fat-tailed Distribution
Volatility Clustering
Leverage Effect
Coarse Vol. Clustering
Check if the Generated Time Series Resembles the Real Data

What about splitting data into train/test/validation sets? For the current project, we do not consider any test/validation sets. The reason behind this decision is as follows: It makes sense to use a test set if somebody wants to test the discriminator's ability to distinguish between true and fake samples. But in this project, we are only interested in the generator's ability to produce proper time series. Thus instead of using tests/validation sets or loss curves, we evaluate the quality of the generator by observing the generated samples.

IV Learning Framework

In this section, we will first build a theoretical background that is required to better capture the goal of this project. Then we would present the deep learning models we implement to satisfy our goal. In the following section, we introduce the stylized facts which is a central notion that we use in this project.

A. Stylized facts

What are Stylized Facts?

The performance of an asset during a certain period of time is given by its relative return, either $R_t = \frac{S_t - S_{t-1}}{S_{t-1}}$ or its log return $R_t = \log(S_t) \ \ \log(S_{t-1})$

The characteristic properties of asset returns are well-studied and commonly known as *Stylized Facts.* The main goal of this report is to generate asset returns that reproduce the stylized facts. Studying stylized facts helps economists to develop theories that better explain the behavior of the economy and make more accurate predictions about its future. A list of the most important stylized facts is as follows:

1) Fat-tailed distribution:

Asset returns admit heavier tails than the normal distribution. In other words, it is observed that the probability distribution of price returns $P(r)$ has a power-law decay in the tails $P(r) \propto r^{-\alpha}$

In which the exponent typically takes the values:
$3 < \alpha < 5$

2) Volatility Clustering:

Asset returns admit phases of high activity and low activity in terms of price changes, and the effect is called *Volatility Clustering.* Volatility clustering refers to the fact that large/small price fluctuations tend to cluster together temporally. Quantitatively, volatility clustering is characterized by the power-law decay of the auto-correlation function of the absolute price returns.

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

3) The Leverage effect

The volatility of asset returns is negatively correlated with the return process, an effect named *Leverage Effect.* In other words, The leverage effects refer to the tendency that past price return has a negative correlation with future volatility/

4) Coarse-fine Volatility Correaltion

Coarse-Fine Volatility Correlation refers to the relationship between the volatility of a portfolio's underlying assets and the volatility of the portfolio as a whole. The idea behind this concept is that the overall volatility of the portfolio is not simply the sum of the individual volatilities of its elements. Instead, it is a function of both

Table I: WGAN Generator Architecture

Layer	Layer Type	Parameters	Output Shape
1	Input	Noise Input	(None,50)
2	Dense	Units=100, LeakyReLu(0.2)	(None, 100)
3	Add Dim	—	(None,100,1)
4	Conv1D	Filters=32, kernel size = 3, padding='same' SN, LeakyReLu(0.2)	(None, 100, 32)
5	Unsapmling 1D	—	(None, 200, 32)
6	Conv1D	Filters=32, kernel size = 3, padding='same' SN, LeakyReLu(0.2)	(None, 200, 32)
7	Unsapmling 1D	—	(None, 400, 32)
8	Conv1D	Filters=32, kernel size = 3, padding='same' SN, LeakyReLu(0.2)	(None, 400, 32)
9	Unsapmling 1D	—	(None, 800, 32)
10	Conv1D	Filters=1, kernel size = 3, padding='same' SN, LeakyReLu(0.2)	(None, 800, 1)
11	Squeeze Dim	—	(None, 800)
12	Dense	Units = 100, LeakyReLu(0.2)	(None, 100)

Table II: WGAN Discriminator Architecture

Layer	Layer Type	Parameters	Output Shape
1	Input	—	(None,100)
2	Add Dim	—	(None,100,1)
3	Conv1D	Filters=32, kernel size = 3, padding='same' SN, LeakyReLu(0.2)	(None, 100, 32)
4	MaxPooling	Pool Size = 2	(None,50,32)
5	Conv1D	Filters=32, kernel size = 3, padding='same' SN, LeakyReLu(0.2)	(None, 50, 32)
6	Maxpooling	Pool Size = 2	(None, 25, 32)
7	Conv1D	Filters=32, kernel size = 3, padding='same' SN, LeakyReLu(0.2)	(None, 25, 32)
8	MaxPooling	—	(None, 25, 32)
9	Flatten	—	(None, 800)
10	Dense	Units = 50, LeakyReLu(0.2)	(None, 50)
11	Dense	Units = 15, LeakyReLu(0.2)	(None, 15)
12	Dense	Units = 1, LeakyReLu(0.2)	(None, 1)

the individual volatilities and the correlations between them. This matter is more discussed in the results V.

B. Data and Pre-processing

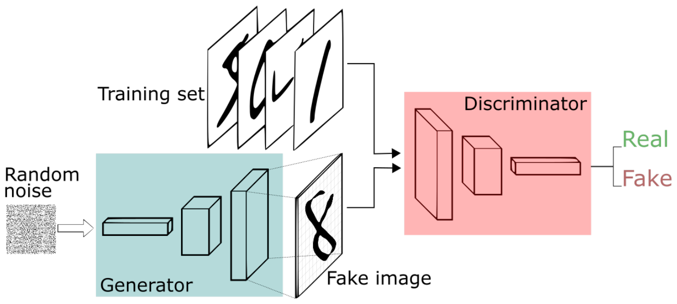
Prior to passing a realization of a financial time series to the discriminator, the time series has to be pre-processed. We briefly explain each of the steps taken. We collect the historical data of the S&P500 and the constituents firms, along with their respective index symbols, with an *R* script. Then we randomly choose an index symbol and consider the corresponding company *csv* and calculate the log returns according to IV-A First, we took a rolling window over 100 trading days but we found out that 100 days is to short to study the statistical properties of financial data so we expanded our rolling window to 1000 days. The amount of collected data is based on the batch size. For each series, the NaN values are dropped and fulfilled with a minimum length of 4096 samples; subject to the necessity to inspect the stylized facts of the time series.

We are interested in one of the columns of data which is the log return. Append data collected and reshape them as follow (batch size, 4096, 1).

The processed data can now be fed to our model; A WGAN which will be discussed in the next section.

C. Wasserstein Generative Adversarial Network (WGAN)

Figure 2: The Generative Adversarial Network (GAN) Schematic Figure



Generative modeling is an unsupervised machine learning method in which new outputs are generated through learning the patterns in data.

Generative Adversarial Networks or GANs, as depicted in 2 are a smart way of training a generative model by re-framing the problem as a supervised learning task with two inner models:

- *The generator model: For generating the new samples*;
- *The discriminator model: For classifying the generated samples as real or fake*;

D. What is the difference between GAN and WGAN?

The key difference between a Wasserstein GAN and a GAN is in the lost function they use. As one can see in 3, GAN uses cross-entropy for the lost function. On the other hand, WGAN uses Earth-Mover (Wasserstein) distance, which measures the difference between the two distributions in a more meaningful way. The smoothness of the loss function in WGAN leads to a more stable behavior during the optimization task.

Figure 3: The WGAN and GAN loss functions

$$\text{GAN: } \nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[logD(\mathbf{x}^{(i)}) + log(1 - D(G(\mathbf{z}^{(i)}))) \right]$$

$$\text{WGAN: } \nabla_{w} \frac{1}{m} \sum_{i=1}^{m} \left[f(\mathbf{x}^{(i)}) - f(G(\mathbf{z}^{(i)})) \right]$$

Another challenge when working with GAN is a phenomenon called *Mode Collapse* which is further discussed in section V-B. This problem happens when the generator is unable to cover the diversity of the target distribution and produces a limited range of samples from the target distribution. On the other hand, it is experimentally observed that WGANs are less probable to be affected by this phenomenon.

For all the reasons discussed earlier, we choose to work with WGAN as an alternative to GAN. First to have an improved stabilization and second to obtain an overall better performance.

E. Network Structure

Our WGAN model, as demonstrated layer by layer in I and II, obeys a generic GAN structure, containing a generator neural network and a discriminator one. The normally distributed 100-dimensional noise is fed to the generator and a 4096 steps time series is produced as the output.

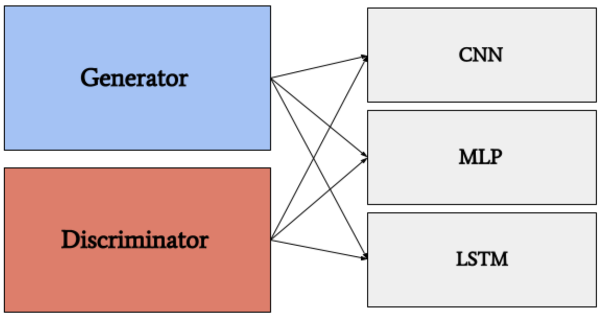
The lack of available data makes the training of machine learning algorithms difficult in finance. Hence, there is a great interest in the development of time series generators as a data augmentation method. However, Financial Time Series exhibit different properties that are often difficult to reproduce with simple models. A good generator needs to be able both to get the properties of the general distribution of the data (Moments for instance) and also to get its temporal properties (Autocorrelation structures).

As shown in figure 4, an overall of three different models are considered for the generator and the discriminator are as the following:

A brief description of each of these three models is brought as the following:

- *MLP: Multi-layer preceptron*
- *CNN: Convolutional Neural Networks*
- *LSTM: Long-Short Term Memory*

Figure 4: Three different models are considered for the generator and the discrimination



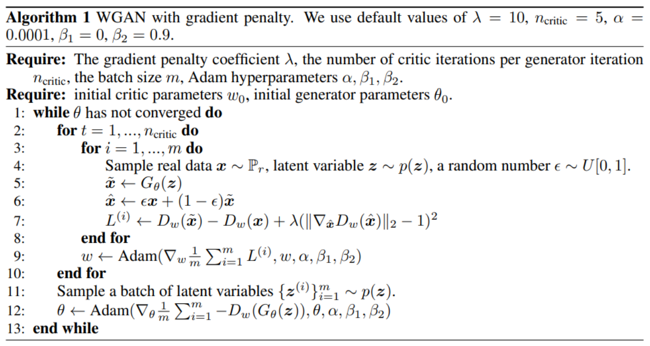
F. *The Algorithm*

The algorithm of our model as demonstrated in figure 5 follows the algorithm of a generic GAN in which the generator is trained to produce samples close to reality and at the same time, the discriminator becomes more and more potent in distinguishing between fake and real samples. The model uses *RMSProp* as the stochastic gradient descent algorithm for weight optimization. The parameters are considered as the following:

- *Learning Rate:* 10^{-5}
- *Batch Size: 128*
- λ *(Gradient Penalty Weight): 10*
- *N Critic (The number of iterations per generator iteration): 5*
- *No Beta, since we use RMSProp*
- *Optimizer: RMSProp*

Finally, As mentioned earlier, 4096 time-steps time-series are collected as the output.

Figure 5: WGAN algorithm, similar to [11] algorithm, except that we use RMSProp instead of ADAM as the optimizer



Algorithm 1 WGAN with gradient penalty. We use default values of $\lambda = 10$, $n_{\text{critic}} = 5$, $\alpha = 0.0001$, $\beta_1 = 0$, $\beta_2 = 0.9$.

```
Require: The gradient penalty coefficient $\lambda$, the number of critic iterations per generator iteration
    $n_{\text{critic}}$, the batch size $m$, Adam hyperparameters $\alpha, \beta_1, \beta_2$.
Require: initial critic parameters $w_0$, initial generator parameters $\theta_0$.
 1: while $\theta$ has not converged do
 2:     for $t = 1, ..., n_{\text{critic}}$ do
 3:         for $i = 1, ..., m$ do
 4:             Sample real data $\boldsymbol{x} \sim \mathbb{P}_r$, latent variable $\boldsymbol{z} \sim p(\boldsymbol{z})$, a random number $\epsilon \sim U[0,1]$.
 5:             $\tilde{\boldsymbol{x}} \leftarrow G_\theta(\boldsymbol{z})$
 6:             $\hat{\boldsymbol{x}} \leftarrow \epsilon \boldsymbol{x} + (1-\epsilon)\tilde{\boldsymbol{x}}$
 7:             $L^{(i)} \leftarrow D_w(\tilde{\boldsymbol{x}}) - D_w(\boldsymbol{x}) + \lambda(\|\nabla_{\hat{\boldsymbol{x}}} D_w(\hat{\boldsymbol{x}})\|_2 - 1)^2$
 8:         end for
 9:         $w \leftarrow \text{Adam}(\nabla_w \frac{1}{m}\sum_{i=1}^{m} L^{(i)}, w, \alpha, \beta_1, \beta_2)$
10:     end for
11:     Sample a batch of latent variables $\{\boldsymbol{z}^{(i)}\}_{i=1}^{m} \sim p(\boldsymbol{z})$.
12:     $\theta \leftarrow \text{Adam}(\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} -D_w(G_\theta(\boldsymbol{z})), \theta, \alpha, \beta_1, \beta_2)$
13: end while
```

V Results

In this section, we present all the results we obtain using our WGAN model with different generators. For more clearance, we divide these sections into different topics, then discuss the results under each topic by referring to its corresponding plots.

A. *Robustness of training*

We need to make a few points about the training of our WGAN models.

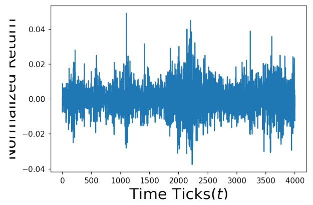
As one can see in figure 7 we only show discriminator losses because, in WGANs, generator loss is generally considered not a very meaningful criterion. On the other hand, discriminator loss is an approximation of the negative Wasserstein distance between a generated distribution and real data distribution. Thus making it an interpretable and useful tool for the evaluation of the model.[12]

Additionally, based on 7, We can deduce that the loss of our discriminators, after struggling in the initial phase of the training, gradually tends to decrease toward zero and finally fluctuates over the zero value. Even when considering the loss function related to the LSTM discriminator, which is regarded as one of the most unstable loss functions in the training phase based on the literature [13], one can see robust training gains.

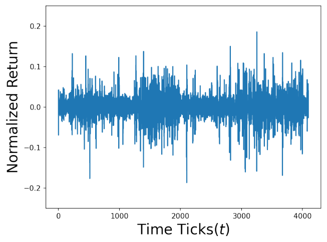
The stable behavior of the loss value shows that the Wasserstein metric serves its purpose and the WGAN

Table III: Producibility of Stylized Facts of Using Different Generators

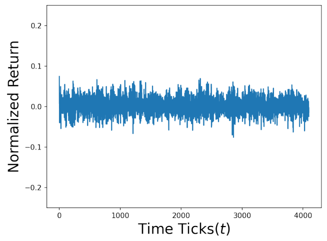
Model	Heavy-tailed distribution	Volatility Clustering	Leverage Effect	Coarse-fine volatility correlation
CNN	×	✓	✓	×
MLP	✓	×	✓	×
LSTM	✓	×	✓	×
Garch [7]	✓	×	×	×
EGArch [7]	✓	×	✓	NA



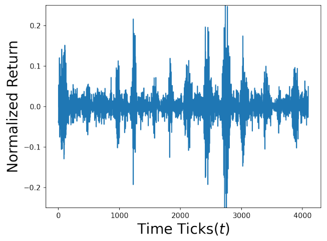
(a) S&P500 real data



(b) CNN



(c) MLP



(d) LSTM

Figure 6: Time series of price returns, generated using different generators for our WGAN model, alongside the S&P500 real data

model seems to be a suitable framework for working on financial historical data.

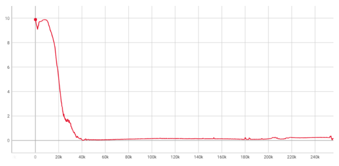
B. Mode collapse

We have to mention one major failure we incurred during the training process; the Mode Collapse, a common problem in GAN that can occur when the generator gets trapped in a definite setting where it always produces the same output. Therefore so it fails to capture the complexity of real data [14].

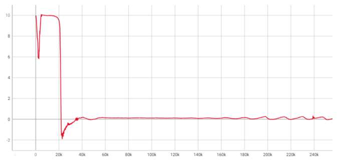
To check for the mode collapse effect, whenever we use generators to create new samples during training we used different latent tensors, therefore we are able to visualize time series and check whether they are significantly different when sampling different sections of the noise sequence. If the generator keeps producing almost identical time series, we get sure that we have encountered mode collapse [15]. Furthermore, We diminished the learning rate to values lower than 0.00002 and decided to adopt a value of 10^{-5} [16]. There is also another approach we investigate which is adding noise to the real-time series before feeding them into the discriminator. But we found this strategy not as effective as the previous method [17].

C. Analysis of Synthetic Generated Time Series

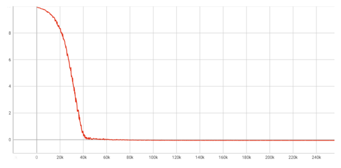
The statistical properties of the generated time series from our models are analyzed and compared with that of a real financial time series, S&P500 stylized facts.



(a) CNN

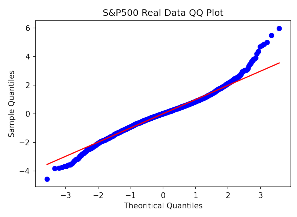


(b) MLP

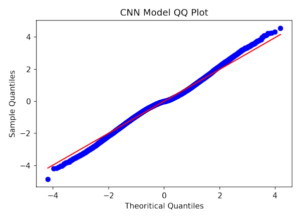


(c) LSTM

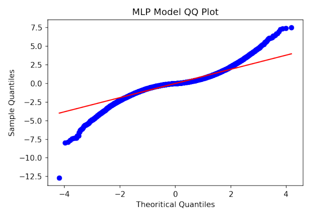
Figure 7: The WGAN discriminator loss value for different generator models



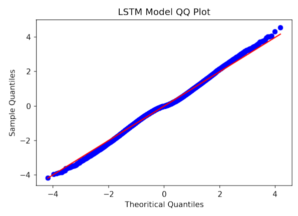
(a) S&P500



(b) CNN



(c) MLP



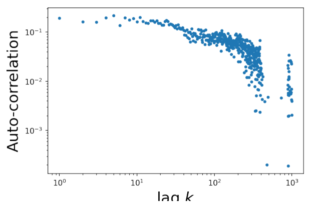
(d) LSTM

Figure 8: The quantile-quantile plot or QQ plot of generated time series w.r.t normal distribution for different generator models alongside the S&P500 real data.

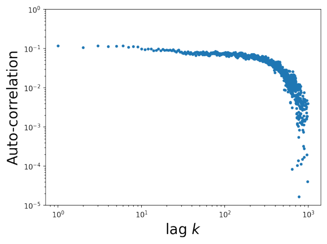
As one can see in figure 6, there are quite interesting visual differences in log-returns movements for the generated data, compared to a randomly extracted time series of the S&P500 index. In the S&P500 case, returns reveal a much smaller scales (returns tend to flatten over many companies); which is acceptable since we randomly fed single constituents (single companies log-returns) time series to the models.

Generated Time series has zero mean and sufficient price return fluctuations but we need further analysis of the statistical properties to assess the possibility to generate synthetic representative financial datasets.

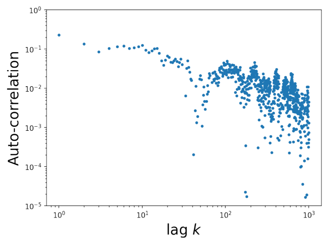
First, we compare the kurtosis and skewness distributions of the historical samples in the S&P500 collection with the generated time series samples, figures 13 and



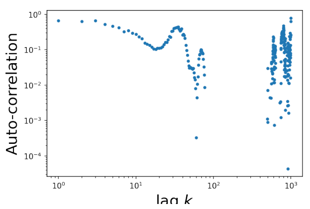
(a) S&P500 real data



(b) CNN

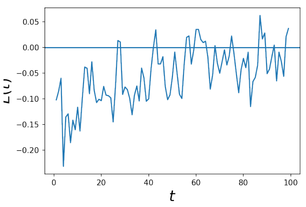


(c) MLP

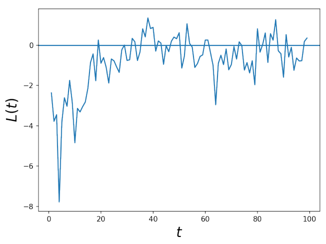


(d) LSTM

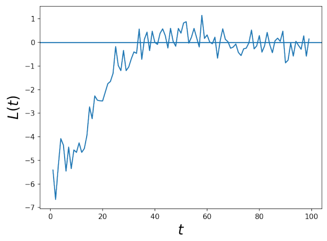
Figure 9: The absolute autocorrelation function (ACF) of the generated time series using different generator models, alongside the ACF of S&P500 real data.



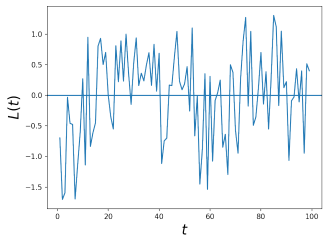
(a) S&P500 real data



(b) CNN



(c) MLP



(d) LSTM

Figure 10: The leverage effect of the generated time series using different generator models, alongside the leverage effect obtained out of S&P500 real data.

14 show box-plots for synthetic data sampled every 200 epochs.

Kurtosis is similar to the variance, but it measures how a distribution spreads with a focus on the tails. Whether the data are heavy-tailed or light-tailed relative to a normal distribution. On the other hand, Skewness measures the lack of symmetry around the mean in a distribution.

The kurtosis of CNN and LSTM models in figure 13 presents higher values compared to the MLP architecture while the skewness values are not so far from being neutral (symmetric distribution), much different from the historical sample values which present a broader distribution of the outliers.

Additionally, as mentioned in the stylized facts section IV-A, assets log returns frequently have fat-tails. Considering 8, each of the quantiles of the historical and synthetic log returns of a whole epoch is plotted against the quantiles of the normal distribution. One can observe that historical data and synthetic data generated with LSTM architectures have consistently heavier tails compared to data generated with the MLP model, while CNN shows a mild fat-tail effect.

The result suggests that historical and LSTM-generated returns are more volatile, incurring a more extreme outcome than MLP and CNN data.

Other stylized facts, as previously discussed in IV-A are Volatility clustering 9, leverage effect 10, and coarse-fine volatility correlation 11. The slow decay of the autocorrelation is observed for the CNN architecture but MLP and LSTM models show an anomalous inversion of the trend for higher k-lags values.9

11 shows that plots for generated data have higher values corresponding to the peak of the distribution with respect to the S&P500 samples.

The leverage effect, in contrast to the statistical properties above, is a market-dependent property [18], Therefore we expect different-looking plots 10 that depend on which samples the neural networks have been trained on. Leverage effects are present as the lead–lag correlation $L(k)$ takes negative values for small k and roughly follows the exponential decay.

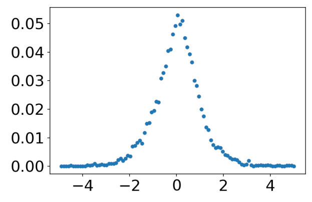
15 shows the statistical properties of a single firm. As one can also observe in this case, While the general trend regarding the stylized fact is confirmed, the data points are noisier and the functional forms of the properties are less clear than those averaged in the stylized facts corresponding to S&P500 data.

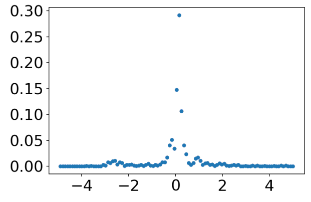
Finally, to capture all the results corresponding to different models at one glance, the reproducibility of the major stylized facts of the models is summarized in III

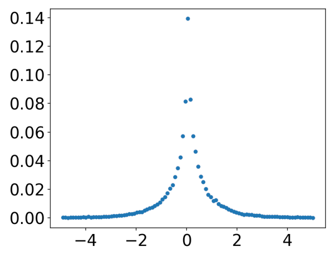
VI Conclusion

In this paper, we showed that multi-layer perception and convolutional neural networks, and LSTM architectures can be theoretically used in an adversarial modeling framework to approximate financial time series that retains the major statistical properties such as stylized facts.

However in practice, WGAN employed for financial purposes in this report is far from optimal in generating reliable, robust data that would precisely represent every


(a) S&P500 real data


(b) CNN


(c) MLP

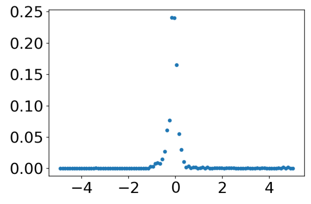
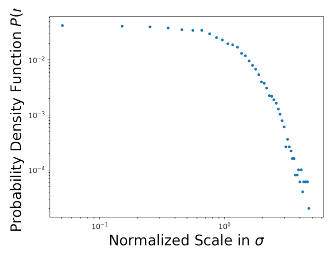
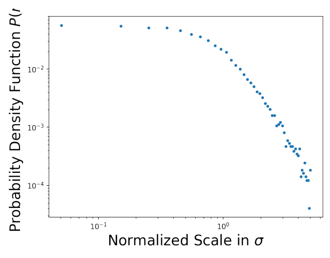

(d) LSTM

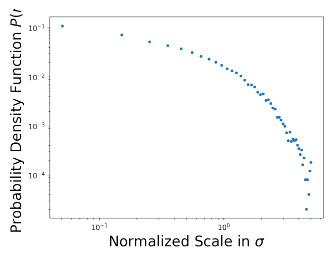
Figure 11: Distribution of the generated time series using different generator models, alongside the distribution of S&P500 real data.



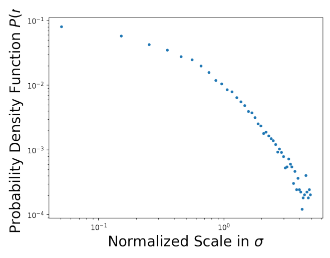
(a) S&P500 real data



(b) CNN

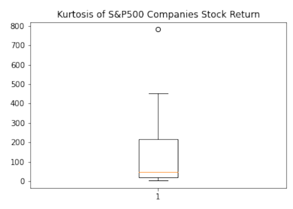


(c) MLP

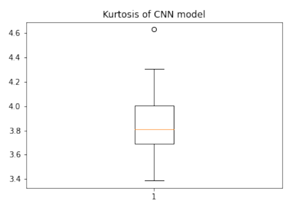


(d) LSTM

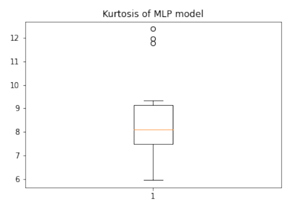
Figure 12: The heavy tail effect in generated time series distribution using different generator models, alongside the heavy-tail effect obtained out of S&P500 real data.



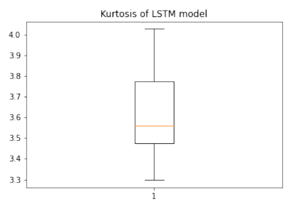
(a) S&P500



(b) CNN

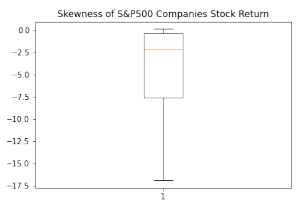


(c) MLP

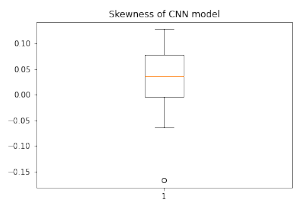


(d) LSTM

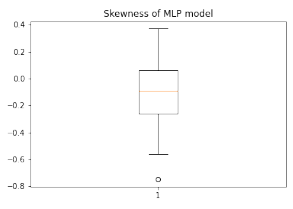
Figure 13: *Kurtosis* boxplots of multiple generated time series obtained using different models for the generator. In probability theory, Kurtosis is a measure of the "tailedness" of the probability distribution of a real-valued random variable. Notice that for each generator we need to produce multiple distinct time series, then measure the Kurtosis of each of these different time series, and finally, we plot the boxplot for all of the Kurtosis values corresponding to a specific generator model



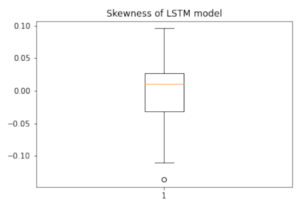
(a) S&P500



(b) CNN

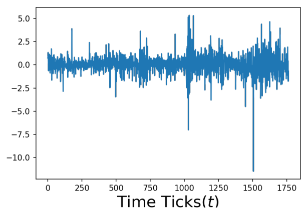


(c) MLP

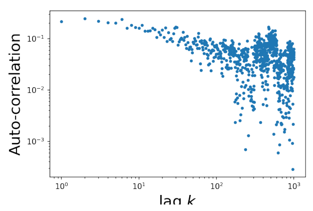


(d) LSTM

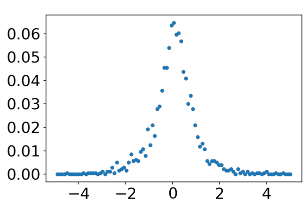
Figure 14: The *Skewness* boxplots of multiple generated time series obtained using different models for the generator. In probability theory, Skewness is a measure of the asymmetry of the probability distribution of a real-valued random variable around its mean. Notice that for each generator we need to produce multiple distinct time series, then measure the Skewness of each of these different time series, and finally, we plot the boxplot for all of the Kurtosis values corresponding to a specific generator model



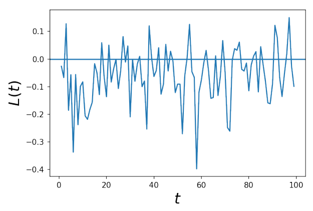
(a) Price return



(b) Auto correlation (ACF)



(c) Linear Distribution



(d) Leverage Effect

Figure 15: Stylized facts corresponding to a random company (*PayPal, 2016-2023*) price return.

single aspect of the underlying assets. We expect this topic as an important addition to the trading board, along with other established methodologies, to ensure an accurate analysis of financial matters.

The next development stage of our research is the implementation and comparison with other forecasting methods in mathematical finance, many other network architectures should be tested with the goal of selecting the most effective ones in financial time-series modeling.

VII Further Information

What have we learned from this project and which difficulties have we encountered?
In general, we learned to apply different deep learning architectures to practice financial time series generation with a focus on the reproducibility of stylized facts. We also learned to implement models using *Pytorch* as a prominent machine learning framework.

One of the challenges we encountered during this task was finding a proper model which remains stable and avoids model collapse. In order to do so we used WGAN instead of GAN which as mentioned earlier, is less prone to mode collapse due to their utilization of the Wasserstein distance and weight clipping constraint.

The contributions to this project are summarized as follows:

- *Walter Martemucci: WGAN architecture, MLP, and CNN*
- *Bahador Amjadi: LSTM, preprocessing, and visualization*

References

[1] T. Bollerslev, "Generalized autoregressive conditional heteroskedasticity," 1986. [Online]. Available: https://econpapers.repec.org/paper/eeirpaper/eeri__5frp__5f1986__5f01.htm

[2] M. Scholes, "The Pricing of Options and Corporate Liabilities." [Online]. Available: https://www.jstor.org/stable/1831029#metadata__info__tab__contents

[3] S. L.Heston, "A Closed-Form Solution for Options with Stochastic Volatility with Applications to Bond and Currency Options," 1993. [Online]. Available: https://www.jstor.org/stable/2962057

[4] P. T. Adriano Koshiyama, Nick Firoozye, "Generative Adversarial Networks for Financial Trading Strategies Fine-Tuning and Combination," 2019. [Online]. Available: https://arxiv.org/abs/1901.01751

[5] C.-P. H. Jou-Fan Chen, Wei-Lun Chen, "Financial Time-series Data Analysis using Deep Convolutional Neural Networks," 2016. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7979885/

[6] G. H. S. T. C. Z. Xingyu Zhou, Zhisong Pan, "Stock Market Prediction on High-Frequency Data using Generative Adversarial Nets," 2018. [Online]. Available: https://downloads.hindawi.com/journals/mpe/2018/4907423.pdf?

[7] K. T.-I. Shuntaro Takashi, Yu Chen, "Modeling financial Time-series with Generative Adversarial Networks," 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378437119307277

[8] J. O. Antonio Rosolia, "Analayzing Deep Generated Financial Time Series For Various Asset Classes," 2021. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract__id=3898792

[9] R. K. Magnus Wiese1, "Deep Generation of Financial Time Series," 2019. [Online]. Available: https://arxiv.org/pdf/1907.06673.pdf

[10] F. de Meer Pardo, "Enriching Financial Datasets with Generative Adversarial Networks," 2019. [Online]. Available: https://markrbest.github.io/assets/2021-03-22/Enriching__Financial__Datasets__with__Generative__Adversarial__Networks.pdf

[11] F. A. M. A. Ishaan Gulrajani, "Improved Training of Wasserstein GANs," 2017. [Online]. Available: https://arxiv.org/abs/1704.00028

[12] S. C. Martin Arjovsky and L. Bottou, "Wasserstein GAN ," 2017. [Online]. Available: https://arxiv.org/pdf/1701.07875.pdf

[13] X. W. Zijian Niu, Ke Yu, "LSTM-Based VAE-GAN for Time-Series Anomaly Detection ," 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/13/3738

[14] Stephan Müller, "Generative adversarial networks: How data can be generated with neural networks," https://www.statworx.com/en/content-hub/blog/generative-adversarial-networks-how-data-can-be-generated-with-neural-n

[15] T. T. Hoang Thanh-Tung, "On Catastrophic Forgetting and Mode Collapse in Generative Adversarial Networks ," 2018. [Online]. Available: https://arxiv.org/abs/1807.04015

[16] R. G. Karttikeya Mangalam, "Overcoming Mode Collapse with Adaptive Multi-Adversarial Training," 2021. [Online]. Available: https://arxiv.org/abs/2112.14406

[17] P. F. Simon Jenni, "On Stabilizing Generative Adversarial Training with Noise," 2019. [Online]. Available: https://arxiv.org/abs/1906.04612

[18] F. R. T. Qiu, B. Zheng and S. Trimper, "Return-volatility correlation in financial dynamics," 2006. [Online]. Available: https://journals.aps.org/pre/abstract/10.1103/PhysRevE.73.065103