A predictive analysis of the Goldman Sachs closing price

Abdollah RIDA

March 25, 2019

Abstract

In this report we try to conduct a classical statistical analysis of financial time-series to 1) study the Goldman Sachs closing price using ARIMA/GARCH processes, and 2) leverage these methods to better the prediction quality of several machine and deep learning models, mainly recurrent neural networks (RNN). We also discuss the possibility of using ARIMA/GARCH and Fourier analysis as features to generate realistic financial time-series through a Generative Adversarial Network [4] [6] architecture inspired by [3]. The code and the data are available in [1].

1 Introduction

Financial mathematics is a young field of applications of mathematics which experienced a huge growth during the last thirty years. It is by now considered as one of the most challenging fields of applied mathematics by the diversity of the questions which are raised, and the high technical skills that it requires.

Modeling financial markets is mainly done nowadays through means of continuous time stochastic processes, but the statistical approach to studying these time series and forecasting them is still useful.

Our purpose here is to study stock price movements from a statistical standpoint. We will heavily focus on time series analysis techniques such as ARIMA and GARCH. We will also discuss how to use these results to better the performance of Recurrent Neural Networks (RNN) and discuss how we can leverage the statistical tools already existing to improve other Neural Network models. We will conclude by a quick discussion on how to use ARIMA/GARCH and Fourier analysis to generate realistic time-series using a Generative Adversarial Network (GAN) based on stacked Long-Short Term Memory (LSTM) cells. A model highly inspired by [3].

2 The data

2.1 Basic description

The data used to fit the model are the daily closing prices of the Goldman Sachs stock from March 13, 2006 to March 13, 2019, which corresponds to a total of 3272 observations. We save one third of the data to perform validation later. The data is compiled from Yahoo! Finance (the data is available in [1]) and covers a daily database denominated in US dollar. We calculate the log-returns by taking the natural logarithm of the ratio of two consecutive prices, as a good approximation of daily percentage changes in prices.

Figure 1: daily Goldman Sachs stock closing price

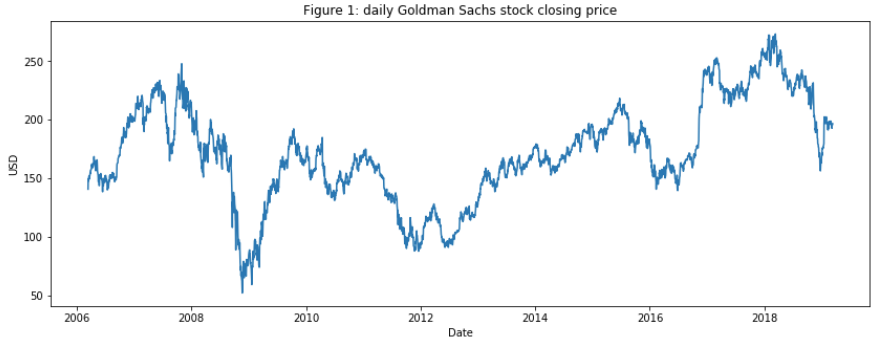
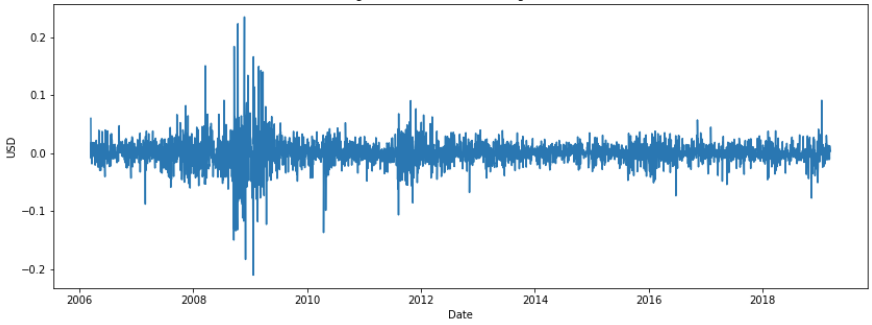


Figure 2: Goldman Sachs log-returns



A simple statistical description of the data set shows that the highest value achieved is $273.38 while the lowest one is $52 which corresponds to the 2008 subprime crisis. As for the log-returns, the maximum is 0.23 while the minimum is -0.21, which shows a really high volatility of the financial asset. The mean return is 0.0001 with a moderately high standard deviation of 0.023. The returns are also positively skewed (0.28) and the kurtosis (15.96) suggests evidence of a distribution with a fat tail.

2.2 Stationarity check

Before modeling time index data, we need to check the stationarity, as a lot of statistical and econometric methods are based on stationarity. We will use the Augmented Dicky-Fuller (ADF) test.

The ADF tests the null hypothesis that a unit root is present in the auto-regressive model. The alternative hypothesis is stationarity. Since we're considering fitting ARIMA and GARCH models (which are auto-regressive), this test is a good idea.

Test Statistic	-13.505784
p-value	0
Critical value (10%)	-2.567
Critical value (5%)	-2.862
Critical value (1%)	-3.432

Based on the results of the Augmented Dickey–Fuller (ADF) tests as shown in Table 2, we fail to accept the null hypothesis of a unit root for the returns, and, hence, stationarity is guaranteed for the log-return series of the Goldman Sachs stock price.

3 Methodology

3.1 Justification of the ARIMA/GARCH model

One common observation we get out of the economic and financial data is volatility clustering. Suppose we have noticed that recent daily returns have been unusually volatile. We might expect that tomorrow's return is also more variable than usual. We can also observe that the squared returns of an asset are usually positively auto-correlated, i.e. if an asset price made a big move yesterday, it is more likely to make a big move today. With economic and financial data, time varying volatility is more common than constant volatility, and accurate modeling of time varying volatility is of great importance.

In our case, we have already known from the excess kurtosis that an obvious fat tails displayed in our series, a typical evidence of heteroskedastic effects as clustering of volatility. We can also observe from the squared log-return that the squared returns appear to fluctuate around a constant level, but exhibit volatility clustering. Large changes in the squared returns tend to cluster together, and small changes tend to cluster together, which also indicates that the series exhibits conditional heteroscedasticity.

It is even more clear if we plot the sample autocorrelation function (ACF) and partial autocorrelation (PACF). The sample ACF and PACF show significant autocorrelation in the squared log-return series.

As illustrated in the ACF and PACF plot of squared returns, there is clearly an autocorrelation. The model we are going to look at will attempt to capture the autocorrelation of squared returns, clustering volatility, as well as the

Figure 3: Goldman Sachs squared log-returns

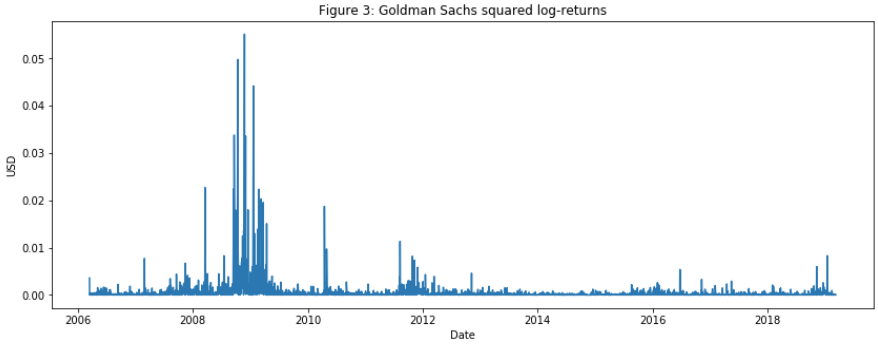


Figure 4: Goldman Sachs squared log-returns autocorrelation

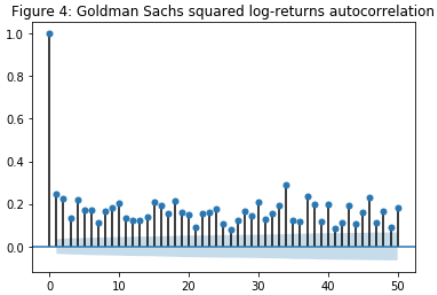
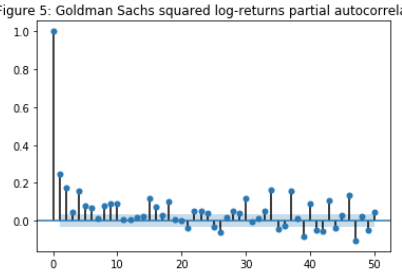


Figure 5: Goldman Sachs squared log-returns partial autocorrelation



heteroscedasticity. The significance of the lags in both the ACF and PACF indicate we need both AR and MA components for our model. As we know that ARMA models are used to model the conditional mean of the process given past information, which however, assumes the conditional variance given the past is constant. We give the definition of the ARIMA model bellow, which is just a generalization of the ARMA model. ARMA/ARIMA model alone fails to capture the volatility clustering behavior. Thus, we will use GARCH process that has become widely used in econometrics and finance, to correct the heavy tails and model the randomly varying volatility in Goldman Sachs's return.

3.2 ARIMA model

Recall that:

Definition 1. Given a time series of data X_t where t is an integer index and the X_t are real numbers, an $ARMA(p', q)$ model is given by:

$$\left(1 - \sum_{i=1}^{p'} \alpha_i L^i\right) X_t = \left(1 + \sum_{i=1}^{q} \theta_i L^i\right) \varepsilon_t$$

where L is the lag operator, the α_i are the parameters of the autoregressive part of the model, the θ_i are the parameters of the moving average part and the

ε_t are error terms. The error terms ε_t are generally assumed to be independent, identically distributed variables sampled from a normal distribution with zero mean.

An $ARIMA(p, d, q)$ process expresses this polynomial factorisation property with $p = p' - d$, and is given by:

$$\left(1 - \sum_{i=1}^{p} \phi_i L^i\right) (1 - L)^d X_t = \left(1 + \sum_{i=1}^{q} \theta_i L^i\right) \varepsilon_t$$

and thus can be thought as a particular case of an $ARMA(p + d, q)$ process having the autoregressive polynomial with d unit roots. (For this reason, no ARIMA model with d > 0 is wide sense stationary.)

To choose the best order (p, q, d) , we try out different combinations using a parameter grid-search and select the one with the lowest AIC and BIC in Python. We find that the best parameters are $(5, 1, 0)$ for the price series and $(2, 1, 0)$ for the log-returns.

3.2.1 ARIMA on closing price

After fitting the ARIMA process on our closing price series we get the following results:

ARIMA Model Results

Dep. Variable:	D.Close	No. Observations:	3271
Model:	ARIMA(5, 1, 0)	Log Likelihood	-8590.292
Method:	css-mle	S.D. of innovations	3.344
Date:	Tue, 02 Apr 2019	AIC	17194.585
Time:	16:38:26	BIC	17237.235
Sample:	1	HQIC	17209.859

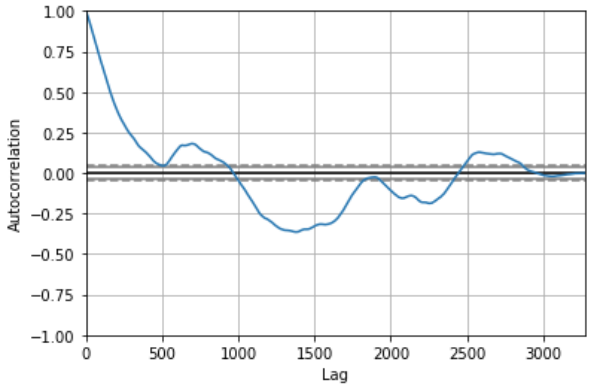
	coef	std err	z	P>|z|	[0.025	0.975]
const	0.0168	0.053	0.317	0.751	-0.087	0.121
ar.L1.D.Close	-0.0418	0.017	-2.393	0.017	-0.076	-0.008
ar.L2.D.Close	-0.0090	0.017	-0.513	0.608	-0.043	0.025
ar.L3.D.Close	0.0109	0.017	0.623	0.533	-0.023	0.045
ar.L4.D.Close	-0.0251	0.017	-1.432	0.152	-0.059	0.009
ar.L5.D.Close	-0.0390	0.017	-2.231	0.026	-0.073	-0.005

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	1.4582	-1.1417j	1.8519	-0.1057
AR.2	1.4582	+1.1417j	1.8519	0.1057
AR.3	-0.7393	-1.7456j	1.8958	-0.3138
AR.4	-0.7393	+1.7456j	1.8958	0.3138
AR.5	-2.0802	-0.0000j	2.0802	-0.5000

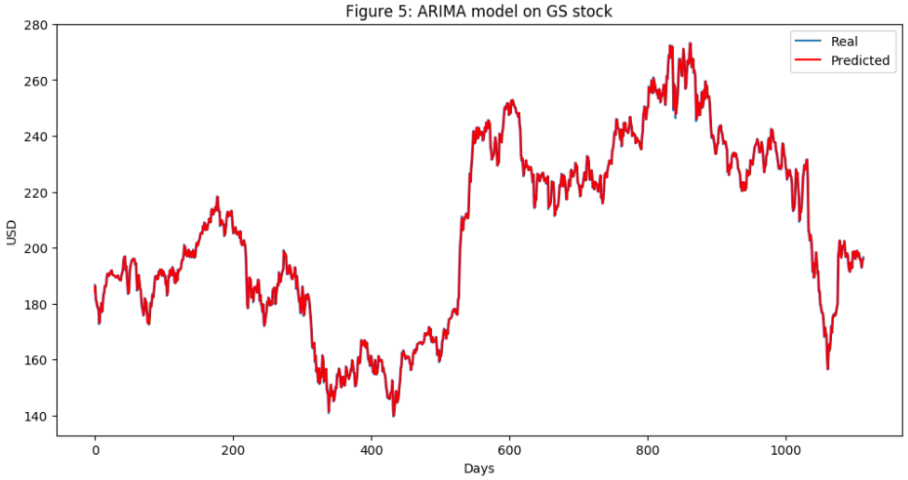
The autocorrelation plot is the following:

As we can see from the following figure ARIMA gives a very good approximation of the real stock price. We will use the predicted price through ARIMA



as an input feature into the LSTM because, as we mentioned before, we want to capture as many features and patterns about Goldman Sachs as possible. We go test MSE (mean squared error) of 9.184, which by itself is not a bad result (considering we do have a lot of test data), but still we will only use it as a feature in the LSTM.

Figure 5: ARIMA model on GS stock



3.2.2 ARIMA on the log-returns

Next, we need to check the residuals after fitting ARIMA(2,1), which should display heteroscedasticity as discussed previously. We can see this in the following figures.

The prediction is not as good as what we might have seen with the closing price series.

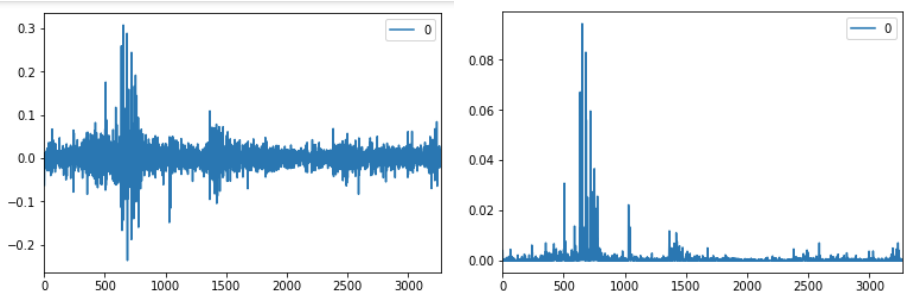


Figure 6: Squared residuals autocorrelation

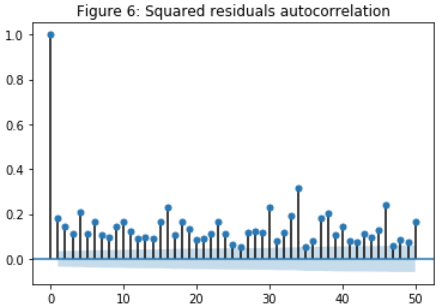


Figure 7: Squared residuals partial autocorrelation

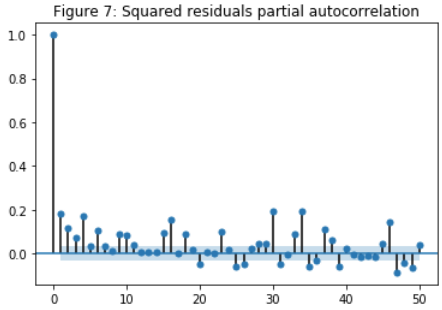
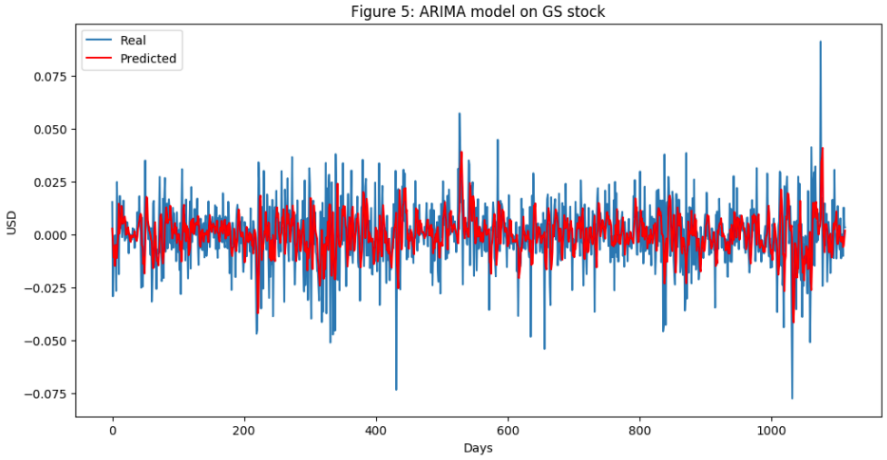


Figure 5: ARIMA model on GS stock



3.3 GARCH process

As we already detected the autocorrelation effects in our residual/innovation series, we now need to apply a GARCH(p,q) model in order to estimate the conditional variance going forward, using:

$$\varepsilon_t = \eta_t \, \sigma_t$$

$$\sigma_t^2 = \omega + \sum_{i=1}^{p} \alpha_i \, \varepsilon_{t-i}^2 + \sum_{j=1}^{q} \beta_j \, \sigma_{t-j}^2$$

Where

$$\sigma_t^2 = Var(\varepsilon_t | \varepsilon_{t-1}, \varepsilon_{t-2}, ...)$$

denotes the conditional variance and η_t a series of iid random variables having a probability density function with mean 0 and variance 1.

The model tells us that tomorrow's variance is a function of today's squared innovations, today's variance, and the weighted average long-term variance. The estimation of (ω, α, β) can be conducted utilizing the maximum likelihood method, which is an iterative process that looks for the maximum value of:

$$-\sum_{i=3}^{N} ln(\sigma_i^2) + \frac{\varepsilon_i^2}{\sigma_i^2}$$

A model built using the *arch* package in python yields the following results:

AR - GARCH Model Results

Dep. Variable:	y	R-squared:	-0.055
Mean Model:	AR	Adj. R-squared:	-0.055
Vol Model:	GARCH	Log-Likelihood:	-9249.30
Distribution:	Normal	AIC:	18508.6
Method:	Maximum Likelihood	BIC:	18537.0
		No. Observations:	2159
Date:	Sat, Apr 06 2019	Df Residuals:	2154
Time:	11:42:23	Df Model:	5

Mean Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
Const	162.4367	0.624	260.401	0.000	[1.612e+02,1.637e+02]

Volatility Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
omega	3.9549	0.765	5.167	2.377e-07	[2.455, 5.455]
alpha[1]	0.8729	5.431e-02	16.074	3.904e-58	[0.766, 0.979]
beta[1]	0.0588	6.202e-02	0.949	0.343	[-6.272e-02, 0.180]
beta[2]	0.0682	6.230e-02	1.095	0.273	[-5.386e-02, 0.190]

Covariance estimator: robust

GARCH modeling builds on advances in the understanding and modeling of volatility. It takes into account excess kurtosis (i.e. fat tail behavior) and volatility clustering, two important characteristics of financial time series, which are also observable in any financial time series. It's theoretically able to provide accurate forecasts of variances and covariances of returns through modeling

time-varying conditional variances. As a consequence, GARCH models have become quite popular in diverse fields as risk management, portfolio management and asset allocation, option pricing, foreign exchange, and the term structure of interest rates.

4 Conclusion: Realistic financial data generation

4.1 Motivation

Access to data is one of the bottlenecks in the development of machine learning solutions to domain specific problems. The availability of standard datasets (with associated tasks) has helped advance the capabilities of learning systems in multiple tasks. The generation of statistically and qualitatively realistic time series would greatly help many fields where the data is lacking, unreachable because of legal barriers or greatly unbalanced.

In finance, stock prices are non reproducible random events: we observe a stock price from 10 years ago to now, and from now to 10 years, but we cannot experiment. There is no control. Therefore it is hard to escape the stochastic calculus framework. By generating time series one can have more examples of financial crisis, more examples of defaulting individuals, and better the overall prediciton of stock price movements.

Our goal in this section is to discuss the generation of realistic financial time series by leveraging the work done on previous sections and a GAN [4] [6] using Long-Short Term Memory (LSTM) cells [5].

4.2 Proposed methodology

We need to understand what affects whether GS's stock price will move up or down. It is what people as a whole think. Hence, we need to incorporate as much information (depicting the stock from different aspects and angles) as possible.

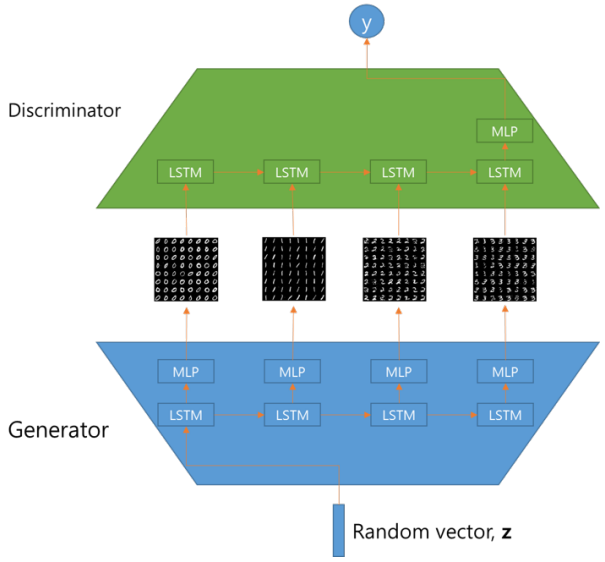
- **Correlated Assets:** these are other assets (any type, not necessarily stocks, such as commodities, FX, indices, or even fixed income securities). A big company, such as Goldman Sachs, obviously doesn't 'live' in an isolated world - it depends on, and interacts with, many external factors, including its competitors, clients, the global economy, the geo-political situation, fiscal and monetary policies, access to capital, etc.

- **Fourier Transforms:** Along with the daily closing price, we will create Fourier transforms in order to generalize several long- and short-term trends. Using these transforms we will eliminate a lot of noise (random walks) and create approximations of the real stock movement. Having trend approximations can help the LSTM-GAN perform better.

We therefore add other assets (Morgan Stanley, JP Morgan, etc...) as features in our dataframe, and use PCA to build an eigen-portfolio that we will use to extract features. We also use Fourier analysis to extract approximations of the time series (more on this in the following subsections) and the previously discussed ARMA/GARCH modelling to create more features.

4.3 The general architecture

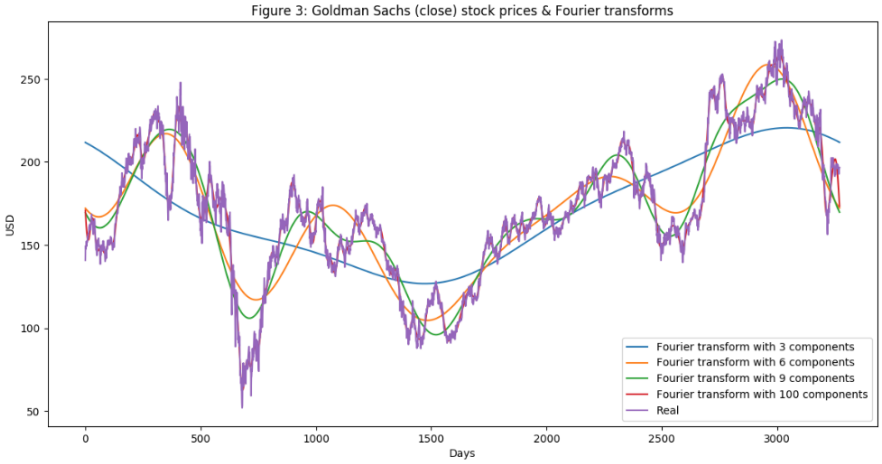
We will use the following architecture, inspired by [2]:



Each modules, generator and discriminator are designed with LSTMs and one Fully Connected Network. Generator is designed as one-to-many, and gets one random vector as input, and generates sequential images. Discriminator is designed as many-to-one model, which get sequential images, and decides what is real or fake.

4.4 Fourier analysis and RGAN

As you see in the following the more components from the Fourier transform we use the closer the approximation function is to the real stock price (the 100 components transform is almost identical to the original function - the red and the purple lines almost overlap). We use Fourier transforms for the purpose of extracting long- and short-term trends so we will use the transforms with 3, 6, and 9 components. These will give us more features to work with. We will also use the 100 component Fourier transform as an input for our RGAN.



4.5 LSTM GAN and sine waves

As previously done in the $MAP582$ project, the RGAN can generate sine waves with an amazing accuracy. The following results from [3] show that the Recurrent GAN (RGAN) is able to "understand" concepts such as frequancy, phase and period and generate convincing signals. This "knowledge" also extends to smooth signals who are nothing but an extension through Fourier series of the sine waves.

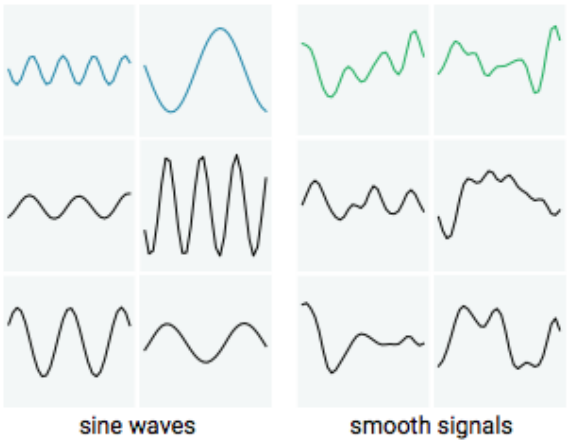
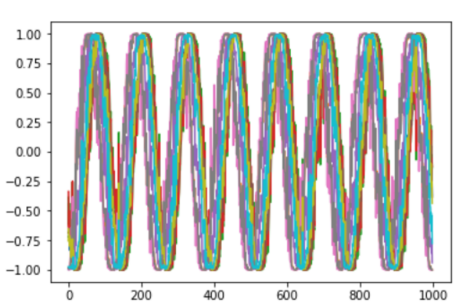
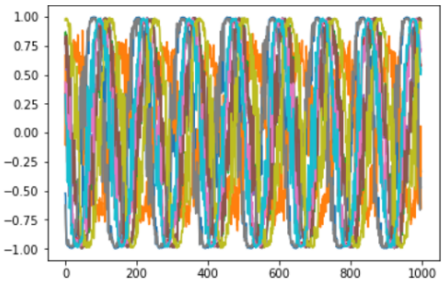


Figure 1: Results of signal generation from the RGAN paper

Our own implementation on *pytorch* was unfortunately not as successful. While our network could easily pick up notions such as phase and period, he had a lot of difficulties in smoothing the signal.



(a): Generated sine waves with different phases



(b): Generated sine waves with different phases, periods and amplitudes

Using less LSTM stacks did greatly improve the smoothness of the generated fake signal but introduced a bias.

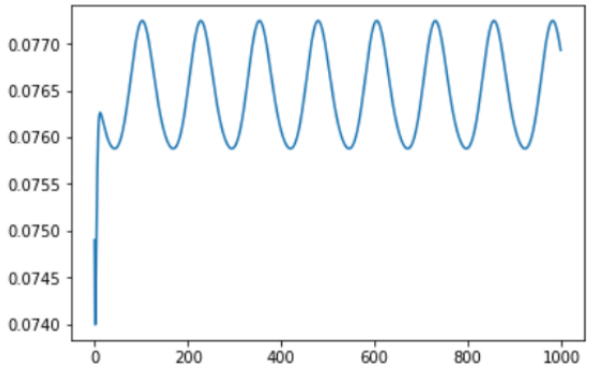


Figure 2: Generated "sine" wave with less stacks of LSTM

The idea is to use a tensor (which is equal to a 100 components Fourier transform of the series) as an input in our GAN. [3] have shown that the RGAN can easily deal with multidimensional time series. We can therefore hope that the RGAN can generate realistic-looking financial time series approximations through their 100 components Fourier representation. Another approach is to use the ARMA/ARIMA prediction as an input. Since it is smoother than the actual time-series, one can hope that the GAN will have less difficulties dealing with it.

References

[1] https://github.com/abdollahrida/deeplearning/tree/master/rgan-pytorch/map565.

[2] https://github.com/ckmarkoh/gan-tensorflow.

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

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

[5] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

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