RECURRENT NEURAL NETWORKS FOR VOLATILITY ESTIMATION

A COMPARATIVE STUDY OF MACHINE LEARNING AND TRADITIONAL METHODS FOR VOLATILITY ESTIMATION

ALICIA OHLSSON

MAXIMILIAN KARLSTRÖM

Bachelor Thesis

Stockholm School of Economics

2020

Recurrent Neural Networks for volatility estimation – A comparative study

Financial decisions are largely based on a tradeoff between risk and return. While the definition of risk is not equal to volatility, it is often used as a proxy for it. Hence, volatility forecasting is of great importance and an essential part of asset pricing, portfolio optimization and risk management. The purpose of this paper is to investigate if a Recurrent Neural Network could provide more precise estimations of seasonal volatility and if so, how it compares to other commonly used models. We prove that they do provide good estimations as well as outperform the other models in doing so.

Keywords:

Volatility forecasting, Recurrent Neural Networks, GARCH, ARCH, Machine Learning

Authors:

Alicia Ohlsson (24159)

Maximilian Karlström (23799)

Tutor:

Marcus Opp, Associate Professor, Department of Finance

Examiner:

Adrien d´Avernas, Assistant Professor, Department of Finance

Bachelor Thesis

Bachelor Program in Business and Economics

Stockholm School of Economics

1. Introduction

Volatility is related to the dynamics of the time-dependent variance in series of returns on an asset. The estimation of volatility is of great interest to the financial market, and it is the most important variable when pricing derivative securities. It is used as a risk measure for other types of assets and it is an important parameter to take into consideration when hedging for risk. As opposed to asset prices, volatility cannot be directly observed but must be estimated. Volatility modeling has been a popular research topic for the last 40 years and will probably continue to be. (Poon, Granger 2003) (Bauwens, Hafner et al. 2012).

According to financial theory, the price of an asset is the present value of its expected future cash flows. However, the price might change when the investors' expectations about future incomes change or when new information becomes available. This results in sudden changes in prices and returns, which therefore affects volatility. (Bauwens, Hafner et al. 2012) An assumption to make is that some of the change in volatility is due to seasonal behavior that repeats from year to year, such as scheduled releases of information, while all other parts are due to other circumstances. This raises the question of how to estimate the seasonal component of the volatility in the best way possible.

While there are currently many ways to estimate volatility, we see a gap in the research when it comes to using machine learning, and more precisely neural networks to do so. To reduce this gap and estimate time-series volatility with machine learning, we started with a Recurrent Neural Network (RNN), an existing machine learning architecture for time-series prediction. We adapted the model to not only predict the next day returns but also the uncertainty in the prediction in terms of the standard deviation. With these two parameters, we assumed a normal distribution of each log-return and optimized the model's parameters by maximizing the log-likelihood of the data by using a numerical optimizer.

In this paper the purpose was to examine whether a model using an RNN structure could provide more precise predictions than two other commonly used models for predicting seasonal

volatility. Hence, the research question was stipulated as follows: How do recurrent neural networks compare to classical models, for seasonal volatility estimation?

The results from our study show that recurrent neural network tends to perform volatility estimation well. Compared to the two other models (the Auto Regressive Conditional Heteroscedasticity model, ARCH and Generalized Auto Regressive Conditional Heteroscedasticity model), GARCH, our model performs better on all error and performance metrics we measured. All results are statistically significant at a confidence level of 95 %.

However, in this paper, we have used synthesized data. This implies that we might have missed some important phenomena existing in the real market due to, for example, the day of the week, holidays, and dividend payments. This could affect the external validity of our model, and in future research, it could be interesting to look at real time-series data. Moreover, the type of RNN model used in this paper is less complex compared to other neural networks models, and one could in further research develop the machine learning architecture to gain further knowledge and do better predictions.

2. Background

One of the most commonly used assumptions in financial economics is "The law of one price", which states that for any given security there can only be one price. This theory implies that a financial asset should be equal to the present value of future cash flows, regardless of how these cash flows are generated. The changes in the price of an asset are due to changes in the expectations of the future cash flows, which could be due to both macro- and micro-events. The bigger price fluctuations the bigger variance of the price, and with the assumption that these updated expectations often come in clusters and are not evenly spread throughout the year, the variance is not constant over time and hence, the volatility for a security is not constant. (Bauwens, Hafner et al. 2012)

Volatility is defined as the standard deviation of the log-returns of an underlying asset. As the volatility cannot be directly observed it must be estimated. Furthermore, as all returns are

assumed to be an outcome of a stochastic variable with a time varying standard deviation, only looking at the realized return at one point in time will not reveal much information about the underlying volatility, so other methods have to be employed. When calculating volatility, two different volatility measures are frequently used: realized volatility and implied volatility. The realized volatility is the estimation of the actual volatility seen in the market. Therefore, assumptions must be made regarding the structure of the volatility and several points in time must be used. Implied volatility is the market's expectation of the volatility for an underlying asset and is often calculated with the inverted Black-Scholes model. By using assuming that the price, which is given, is a function of volatility as described in the Black Scholes model one can solve for the unknown volatility. (Shaikh, Padhi 2013) These two volatilities do not necessarily coincide with each other nor the true volatility.

The simplest type of option consists of a contract giving the owner a right to sell or buy a certain quantity of an asset. The contract stipulates a predetermined price at a predetermined time in the future, relative to the beginning of the contract, called the exercise date. A call option gives the right to buy and a put option the right to sell. If the predetermined price is greater than the spot price at the exercise date, a put option is valuable, and the call option is worthless and vice versa if the spot price is greater. Options are used to mitigate risk. With a call option on oil, you have a guarantee that you will be able to heat your house for, at most, a predetermined price during winter. A put option on wheat allows you to sell what you have produced for, at least, some fixed price. If the underlying asset has a stable, non-volatile, return one can assume where the spot price will be with more certainty, lowering the risk for the one selling the option. This, in turn, lowers the price of the option and explains why volatility is such an important factor for deciding the price of an option. Options are a commonly traded security, allowing producers and hedge fund manager to hedge for risk. To price these options, one must take into consideration the volatility of the underlying asset price.

In 1973, Fischer Black and Myron Scholes introduced a new method to pricing options with the Black-Scholes model. The model requires five different parameters in order to calculate the option price. (Black, Scholes 1973)

- The spot price of the underlying asset,
- The strike-price
- The risk-free rate
- The time to maturity
- The implied volatility of the underlying

The only non-observable parameter is the implied volatility of the underlying, which has led to that the model is often used in reverse to observe the volatility in the market. However, having a reliable way to forecast the volatility and price the option independently, is of great importance. (Bossu 2014)

To derive their model, Black and Scholes made several assumptions regarding the underlying stock data, which is used in this paper. One assumption regarding the stocks in the Black Scholes model is that the stocks follow a random walk in continuous time with a variance rate proportional to the square of the stock price. This implies that the distribution of future stock prices at the end of an interval will be log-normal. (Black, Scholes 1973)

Not long ago, theoretical models assumed constant volatility, as in the Black-Scholes model. (Black, Scholes 1973) In applied econometrics, the ordinary least squares model is a very common tool to use, since it works well to determine how a variable change in response to a change in some other variable(s). However, the forecasting part of econometrics is becoming more and more important, such as forecasting and analyzing the size of the errors of the model. Two of the most well-known non-linear volatility models are the Auto Regressive Conditional Heteroscedasticity model (ARCH) and the Generalized Auto Regressive Conditional Heteroscedasticity model (GARCH). (Engle, Robert 2001)

The ordinary least squares model assumes that the expected value of the squared error terms is constant at any given time, an assumption that often is called homoscedasticity, and it is this assumption that is the focus of the ARCH and the GARCH models. When this assumption is not

fulfilled, due to the variance of the error terms is instead larger at some point and lower at other, the error terms are heteroscedastic. With heteroscedasticity, an ordinary least squares model will still be unbiased, but the standard errors and confidence intervals estimated will be biased. What is distinct with the ARCH and GARCH models is that they do not see this as a problem to be corrected, but rather as a variable to be modeled. (Engle 2001)

At the foundation of the ARCH and the GARCH model are two assumptions. Firstly, the volatility of a certain time-series depends on where in time we look. Secondly, the volatility today will likely be close to the volatility yesterday, and the day before. This is called volatility clustering, a phenomenon implying that there tend to be periods of low volatility as well as periods of high volatility, and as previously explained, the volatility depends on the updated expectations. However, there seems to be a trend that news that update expectations come in a cluster, resulting in periods of volatility trends. For example, in times of crisis, news and specifically bad news, tend to not come alone. The presence of volatility clustering leads to heteroscedasticity (time varying variance), and it also results in positive autocorrelation coefficients of squared returns. This means that an error of a given sign tends to be followed by an error of the same sign, causing periods of high volatility and periods of low volatility. (Bauwens, Hafner et al. 2012)

The Autoregressive Conditional Heteroscedastic Model (ARCH) was first published by Robert F. Engle in 1982. The ARCH model forecasts the volatility using yesterday's time-series value and a constant parameter. ARCH assumes that the error term of the variance of a time-series is a function of previous days' error terms, often as the squares of the previous error terms. The error terms in an ARCH model is assumed to follow an autoregressive model (AR). (Engle, Robert F. 1982) (Engle, Robert F. 2001)

To explain the theory behind the model, we let the dependent variable be labeled r_t, which represents the return on an asset and that $r_t = \mu + \epsilon_t$. Further, we assume that $r_t = s_t z_t$, where z_t is a white noise, and that $z_t \sim \mathcal{N}(0,1)$. With this notation, the ARCH model assumes that the volatility $\sigma(t)$ will satisfy the equation

$$\sigma_t^2 = \alpha_0 + \alpha_1 r_{t-1}^2 + \cdots + \alpha_q r_{t-q}^2 = \alpha_0 + \sum_{i=1}^{q} \alpha_i r_{t-i}^2$$

where

$$\alpha_0 > 0 \text{ and } a_i \geq 0, i > 0$$

The GARCH model relies on the same logic as the ARCH model, however, the GARCH model forecasts today's volatility using both the value of the time-series yesterday and the volatility yesterday, while the ARCH model only uses yesterday's time-series value. The GARCH model assumes that the error terms follow an autoregressive moving average model (ARMA) ((Bauwens, Hafner et al. 2012) (Engle, Robert F. 2001)

As in the ARCH model, we let the dependent variable be labeled r_t, which represent the return on an asset. Further, we assume that $r_t = \mu + \epsilon_t$ and that $\epsilon_t = s_t z_t$, where z_t is a white noise, and that $z_t \sim N\ (0,1)$. Finally, we assume that σ_t^2 satisfy the equation

$$\sigma_{\mathrm{t}}^2 = \omega + \alpha_1 r_{\mathrm{t0}}^2 + \cdots + \alpha_{\mathrm{q}} r_{\mathrm{t-q}}^2 + \beta_1 \sigma_{\mathrm{t-1}}^2 + \cdots + \beta_{\mathrm{p}} \sigma_{\mathrm{t-p}}^2 = \omega + \sum_{\mathrm{i=1}}^{\mathrm{q}} \alpha_{\mathrm{i}}\, r_{\mathrm{t-i}}^2 + \sum_{\mathrm{i=1}}^{\mathrm{p}} \beta_{\mathrm{i}}\, \sigma_{\mathrm{t-i}}^2$$

where

$$\omega > 0$$

With the formulation of the GARCH and ARCH model, we have to estimate the models' parameters. The parameter estimation for optimal variance forecasting can be formulated as an optimization problem, where the goal is to find the forecast that minimizes the expected loss conditional on the available information at the time. (Patton 2011)

Mathematical optimization is the science of finding a value that is optimum for a given function with some known prerequisites and constraints. (Kropko, Jonathan 2016). Essentially, any mathematical model with adjustable parameters could, in theory, be used to estimate volatility.

Now, how well these different models would predict the volatility in practice is a different matter. In the field of machine learning, several models that have shown their adapting capabilities in various settings such as image recognition (Miles 2014), (He, Zhang et al. 2016), text classification (Liu, Qiu et al. 2016) and time-series prediction (Connor, Martin et al. 1994)

Machine learning is the study of computer algorithms that improve through or learn from experience. The foundation of most of machine learning is optimization. (Tom, Mitchell 1997) The way many machine learning models use optimization is through adjustment of the model's parameters to extremize a specific optimization objective. The assumption is that such an extremization results in a set of parameters that allows the model to find patterns in the training data that exist also exists in unseen data. This is called to generalize. To summarize, machine learning models are trained to learn from data, find patterns, and make decisions with minimal human interference. (P. Bennett, Kristin et al. 2006)

In this thesis, a specific type of machine learning model is used, namely a neural network. A neural network can be thought of as a universal approximator. It can be used for regression and classification. Regression is, for a given input return a value belonging to some continuous range of values. An example would be predicting sales of ice-cream on a sunny day. Classification is to return a value from a finite set given some other input. For example, predict whether someone is allowed to drink alcohol given the country they are in and their age. A neural network consists of a sequence of layers. Each layer is a function with parameters that can be adjusted to change the transformation that the layer performs. Each layer transforms its input and pass it to the next layer(s). The layers are chosen based on their input data. In this thesis, all layers are linear transformations of their inputs, the output of the transformation is then transformed with the SoftPlus function, a non-linear function allowing the network to approximate non-linear relationships. (Csanád C., Balázs 2001)

Recurrent neural networks (RNN) are a certain type of neural network designed to recognize patterns in sequences of data. These networks have an inbuilt summary, or memory, of what it has previously seen, making it suitable for estimation of time-series. As depicted in Figure 1, the input x is used together with the hidden state h to create an output o and a new hidden state. The

hidden state works as the memory or summary of the sequence of x's the network have seen so far. This gives the network an ability to summarize an arbitrary number of previous inputs as compared to many other models that use a fixed number of previous values as parameters, for example linear models that uses a sliding window for volatility prediction, such as ARCH. (Cleeremans, Servan-Schreiber et al. 1989)

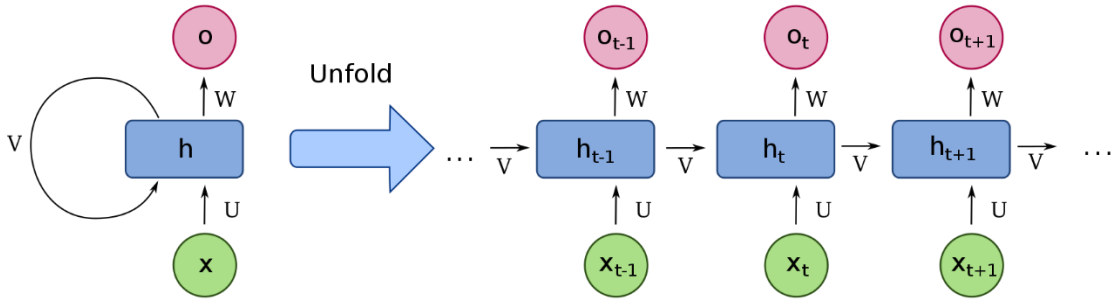

Figure 1. This figure shows the folded and unfolded graph of a recurrent neural network. We see that the input U and the previous hidden state V together form the current hidden state that is used to generate the output W. (Wikipedia 2020)

3. Literature review

In the paper "Neural network volatility forecasts", the authors show that a one-layer neural network with enough neurons activated with a non-linear function can approximate any non-linear function. (Aragonés, Blanco et al. 2007) We know that the formula calculating the standard deviation for a set of points is a non-linear function of their values. Neural networks should, therefore, provide a solid foundation for transforming prices into volatility in a slightly more advanced manner than taking the standard deviation of all prices. Aragonés, Blanco et al. further proves in their paper that neural networks are good at predicting volatility. (Aragonés, Blanco et al. 2007)

In addition, RNNs works well for problems where the data set has sequential nature, and therefore, time-series estimation with RNNs has been a successful research area even though it is not the primary usage area. In the paper "Recurrent Neural Networks and Robust Time-series Prediction" by Connor and Martin, they successfully used a recurrent neural network to predict

time-series. Further, the RNNs are especially good at finding seasonal patterns in time-series data, and in the paper “Seasonality in Variance is common in Macro Time-series" from 2000, seasonal patterns were identified and statistically significant in the variance of macro time-series, which leads to a hypothesis that RNNs would work well in predicting these. (Jaditz 2000)

Moreover, in the article “Volatility Prediction with Mixture Density Networks” by Schittenkopf and Dorffner, a comparison of the performance of standard volatility models and the performance of a Mixture Density Networks was made. A Mixture Density Network is a network used for problems where regular regression models are inapplicable because the variable of interest cannot be used as the target in the regression, such as volatility estimation. The concept is to generate the parameters of a chosen probability distribution for each sample. With these parameters, each sample is given a probability. By maximizing the joint probability of the data, we can hope to get reasonable parameters of the generating distribution for each datapoint. In this paper, we have used a normal distribution to produce the likelihood. This means that the model produced a mean and a standard deviation for the log-return at each point in time. The likelihood is the largest when the model outputs the correct mean and the correct standard deviation at each point in time. In the study by Schittenkopf and Dorffner, the GARCH model was the model that performed best. This was because it seemed to be important to not only look at the time-series value from previous days but also the conditional variance. Hence, it was found that there is a need for long-term memory in the models estimating for volatility, and they recommended further research in the area. The writers especially recommended looking at a recurrent structure of the machine learning model. (Schittenkopf, Dorffner et al. 1998)

4. Data

In this study, we have used synthesized data. Although real-world stock data is not hard to come by, the underlying volatility is unknown. As we wanted to compare our predicted volatility with some ground truth, we needed data for comparison. By artificially generating stock data, a precise value for the volatility generating the logarithmic returns is known at all points in time. For real-world stock data, as mentioned, the implied volatility and realized volatility are imperfect proxies for true underlying volatility, and a comparison with these proxies could lead

to some models overperforming due to bias. Therefore, we chose to generate data from time-varying volatility.

To generate the data, we assumed that $log(r_t) \sim N(0, \sigma(t))$ where $\sigma(t)$ is a function of our choice. We assume that the data is without trend by setting the mean of log-returns to zero. This way we can isolate the objective of the model and only estimate the volatility without having to estimate returns at the same time. By sampling a logarithmic change for all trading days and exponentiate the logarithmic changes we got a series of returns. Coupling these with an initial price set to 1 we got the price for all points in time. We chose 1 as the initial price does not affect the returns.

Although every logarithmic return is normally distributed at every point in time, all logarithmic returns, in general, are not. To illustrate this, we conducted a Shapiro-Wilks test of normality. This test tests the null hypothesis of whether a list of values come from a normally distributed population. (Razali, Wah 2011) We did the test with the hypothesis that they are not normally distributed. In Figure 2 we can observe the results of the test. As no p-value is larger than the chosen significance level of 5 %, we can reject the null hypothesis that they are normally distributed at the chosen level of significance.

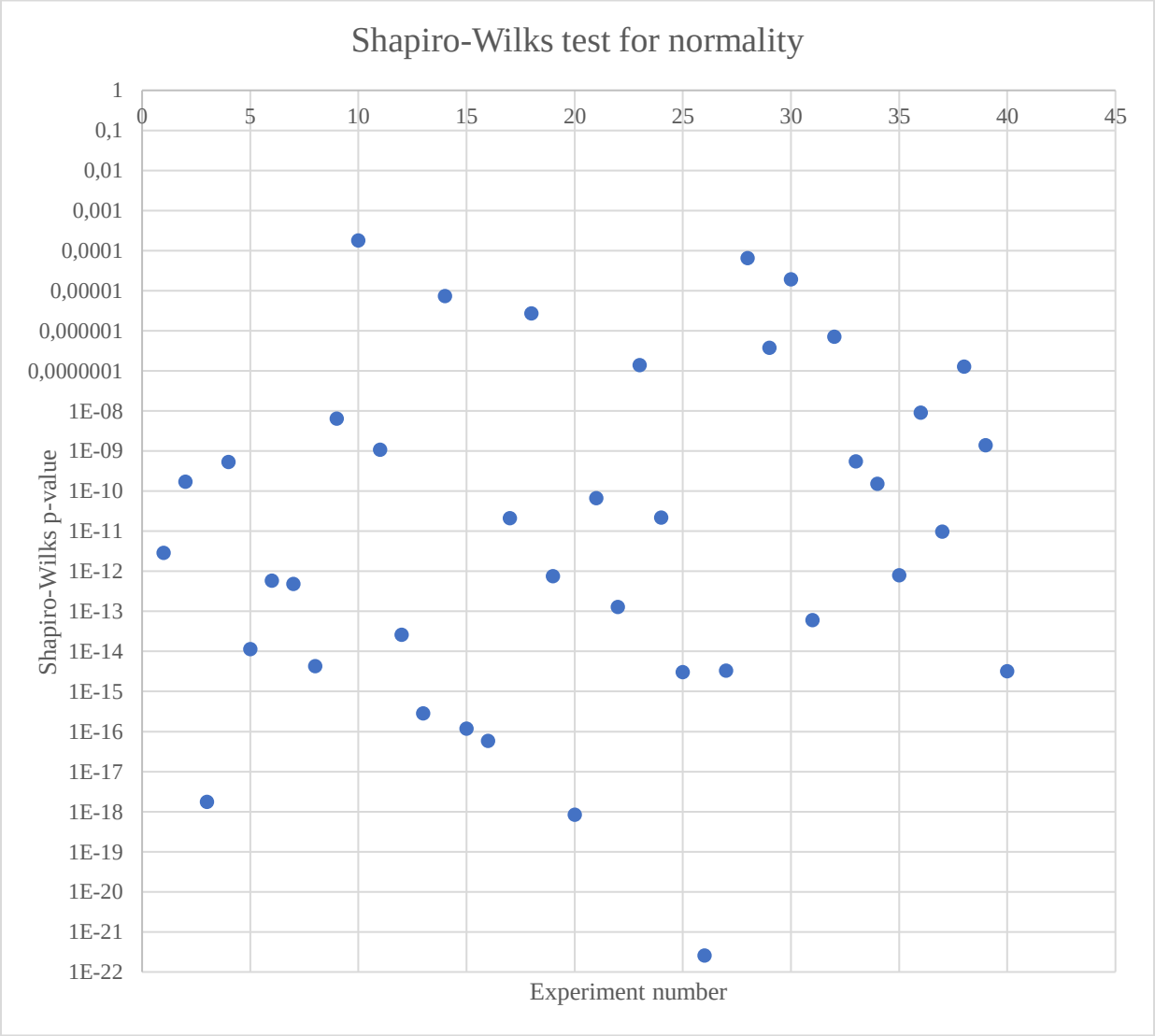

Figure 2. This graph shows the p-value for the Shapiro-Wilks test for the log-returns of each experiment. This test tests if the distribution is normal. As we sample the log-returns from a normal distribution at each point in time but with different standard deviations we expect that this distribution is not normal. The test confirms that at a confidence level of 95 %, the distributuon is .

With the Shapiro-Wilks p-values we can look at the distribution of log-return that look the most normally distributed in Figure 3 and we can see its returns in Figure 4.

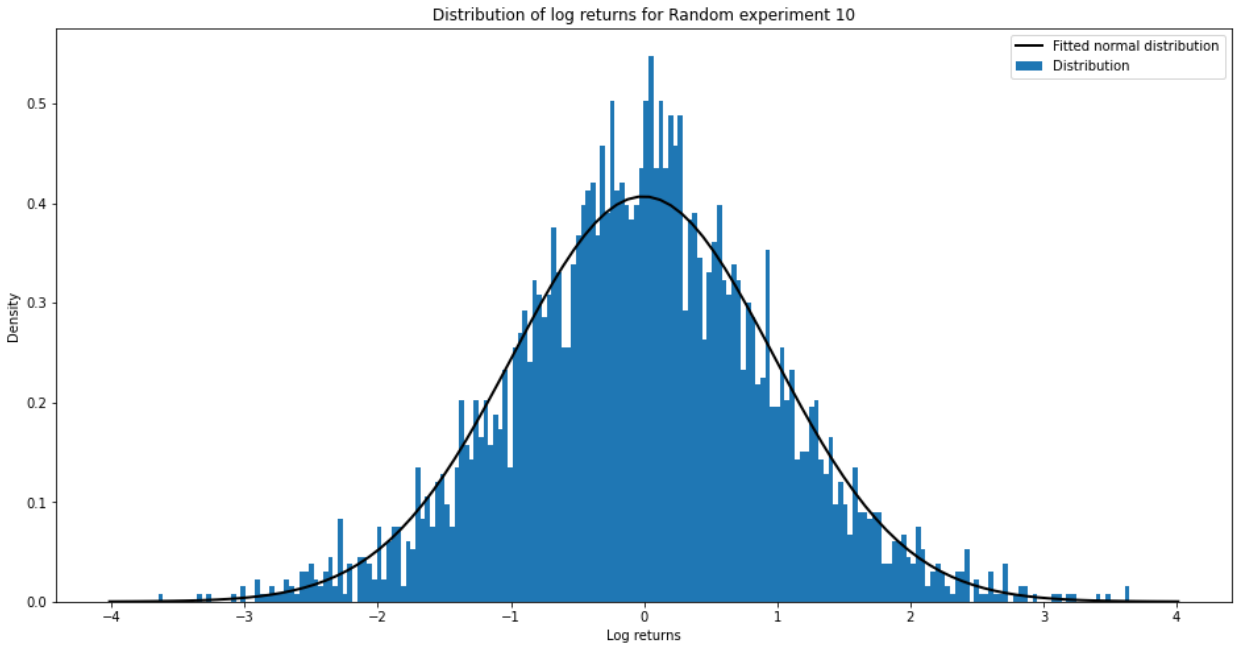

Figure 3. This is a histogram showing the distribution of log-returns for the tenth random experiment. This experiment is the one where the distribution of log-returns looks the most normal based on the p-values of the Shapiro-Wilks test.

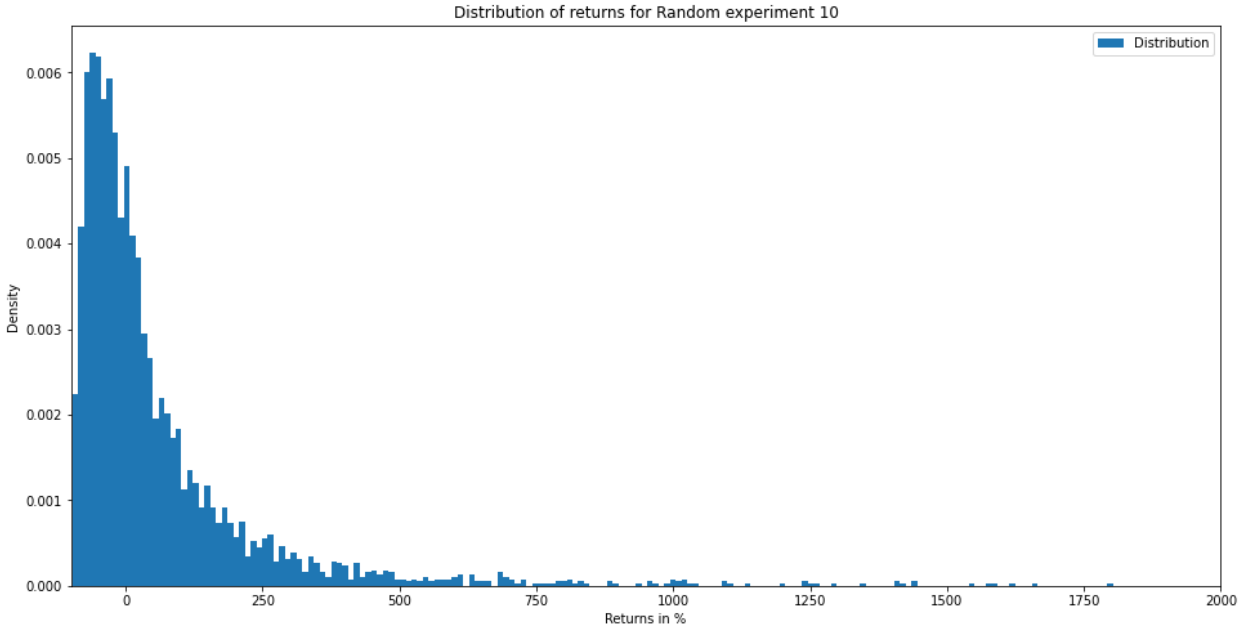

Figure 4. This histogram shows how the returns are distributed for one of our experiments. The lowest limit is -100 % which is not observed but some returns come close.

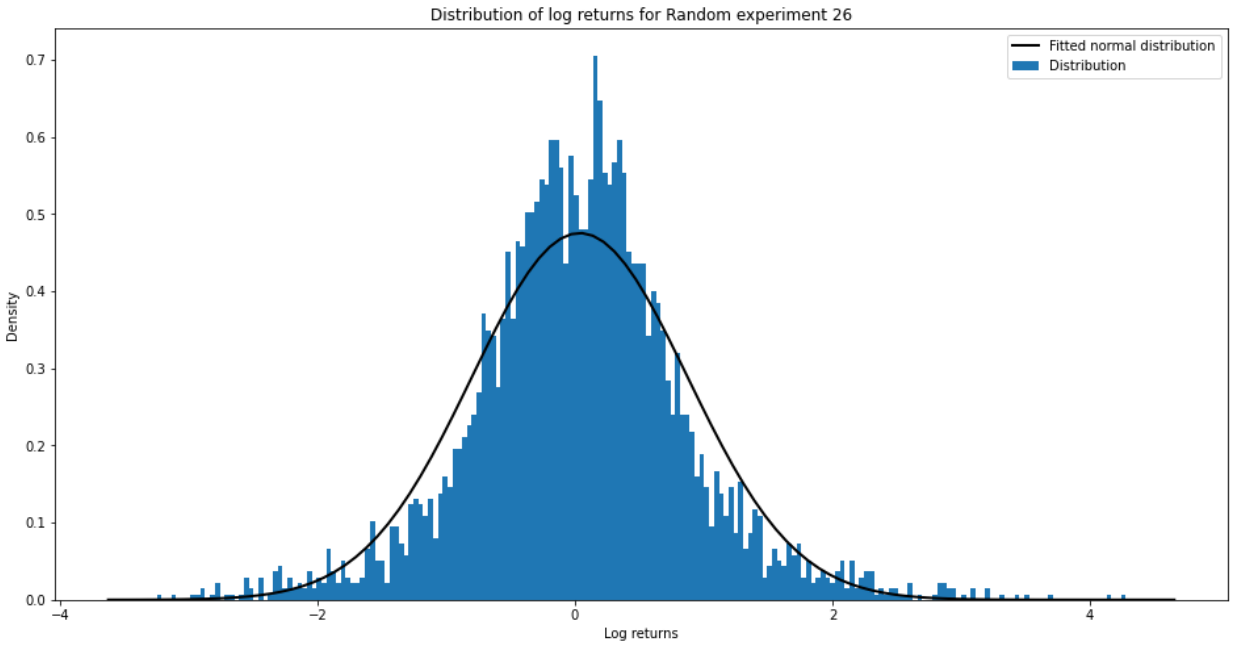

Figure 5. This is a histogram showing the distribution of log-returns for the twenty-sixth random experiment. This experiment is the one where the distribution of log-returns looks the least normal based on the p-values of the Shapiro-Wilks test.

We wanted the data to display seasonal behavior in volatility. To create a seasonal trend in volatility, we chose sigma(t) to be the cumulative sum of uniformly distributed variables between – 1 and 1 for each day in a year. This allows the volatility to increase and decrease. We normalized the volatility so that the volatility is the same the first and last day of the year and between 0.01 and 0.045 for all days. This results in a set of cyclical volatilities that repeat every 365 days. An example is shown in Figure 6.

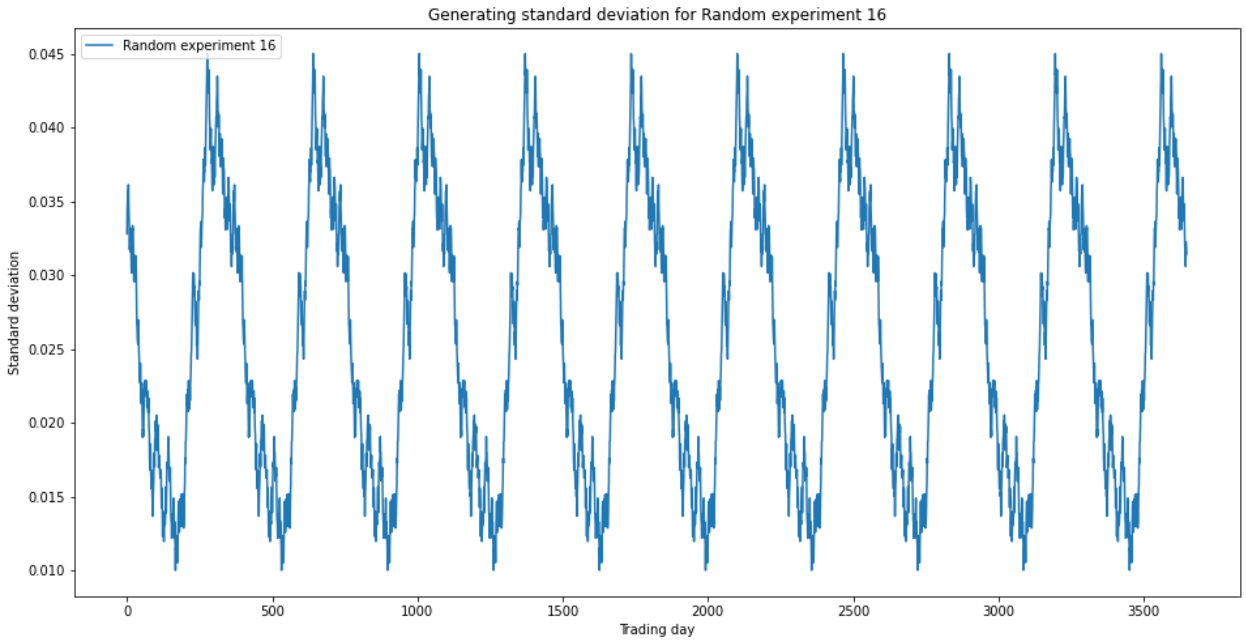

Figure 6. This graph shows the standard deviation used to sample the log-returns for all trading days in the data set. The graph displays the intended repeating seasonality in the synthesized data.

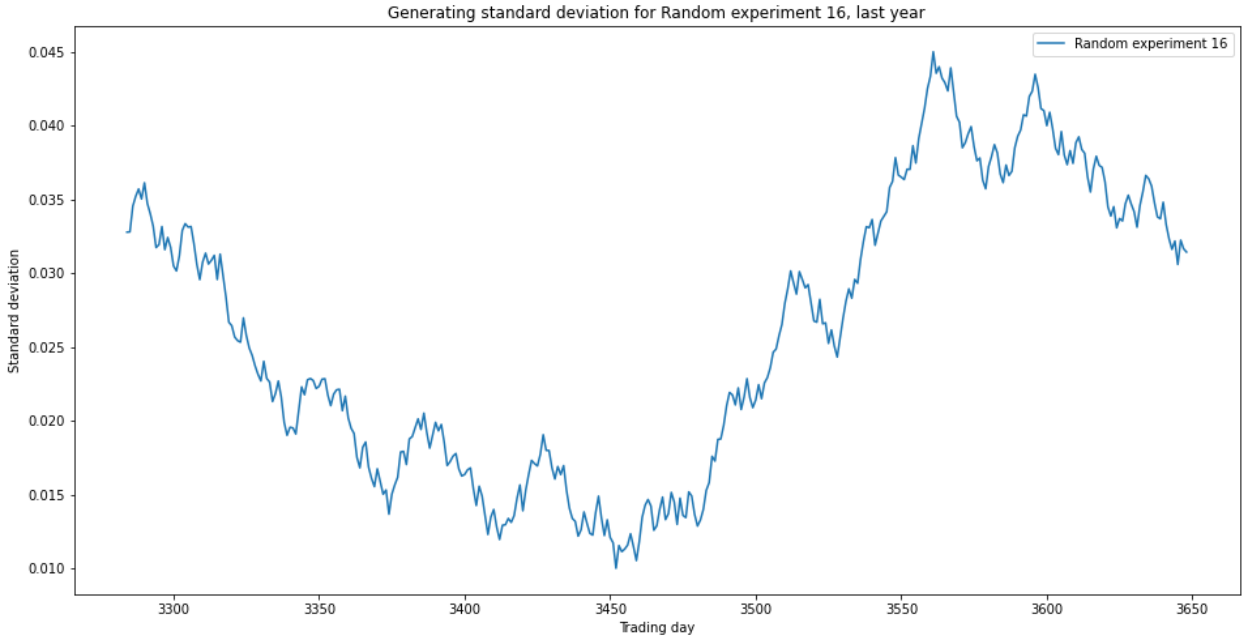

Figure 7. This graph shows the standard deviation for the log-returns of the last trading year for the previous figure.

The maximum of 0.045 and minimum of 0.01 for the value of the volatility was chosen based on some real-world stocks shown below. Their volatility was calculated as a 31-day rolling window standard deviation. Although none of our stocks cycled between these two extremes, we assume that there are some far more and some far less volatile stocks so it seemed like a reasonable tradeoff.

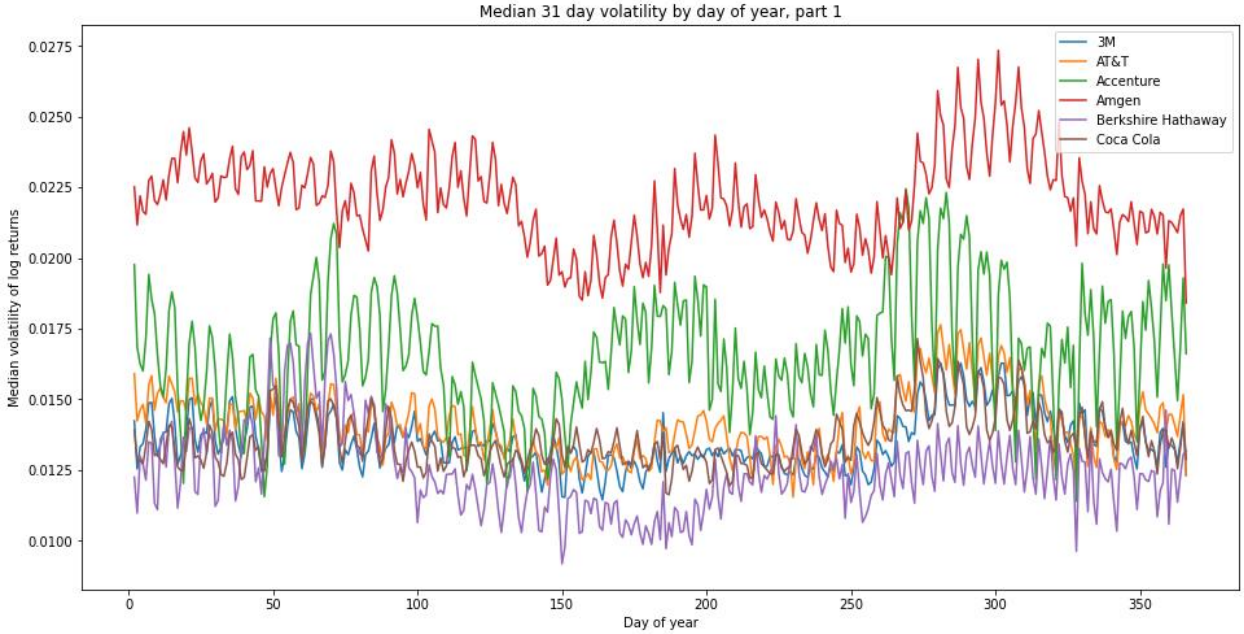

Figure 8. Graph showing the median volatility for some stocks by the day of the year

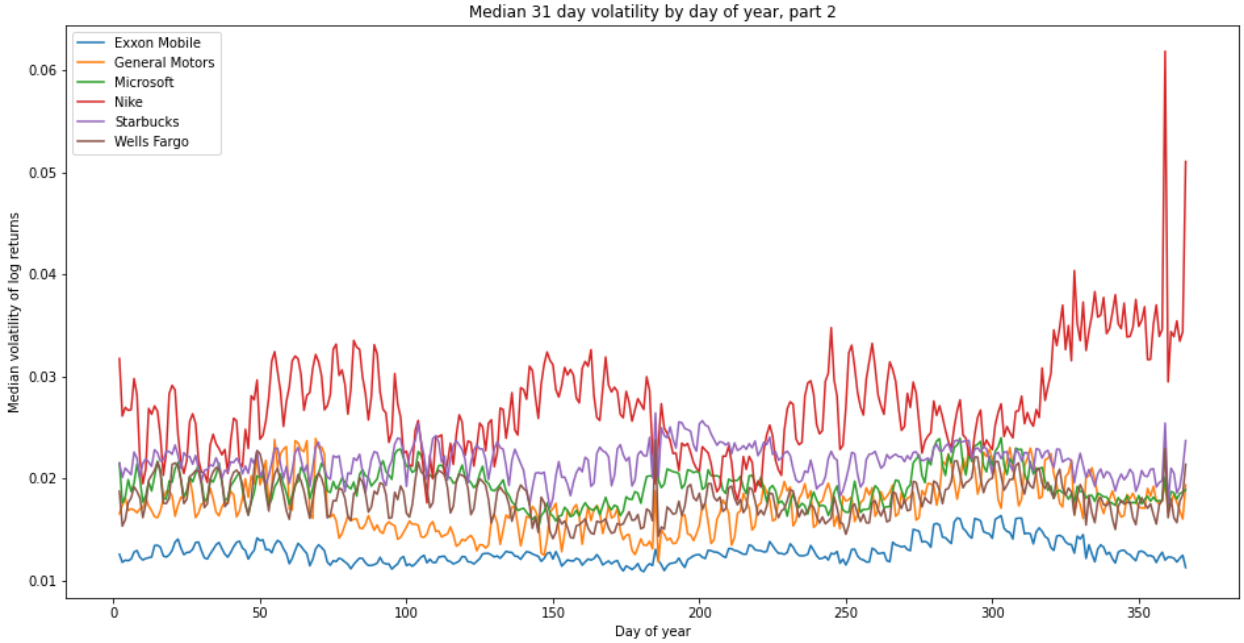

Figure 9. Graph showing the median volatility for some stocks by the day of the year

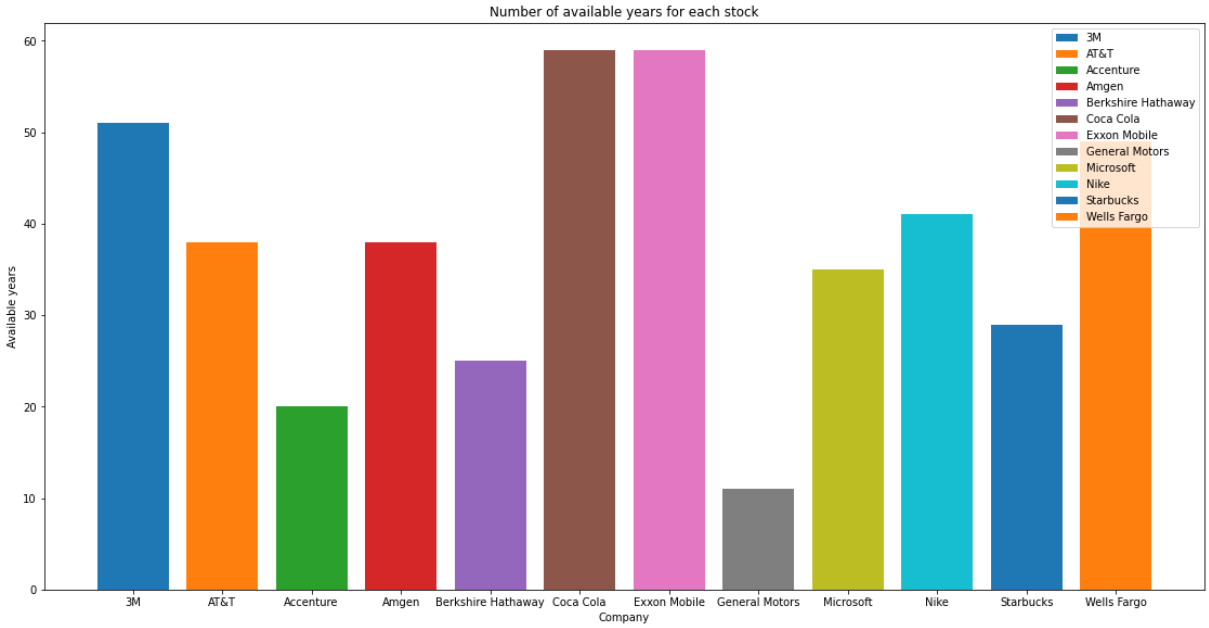

Figure 10. Bar chart showing the number of years per stock used to calculate the mean volatility by day in that can be seen in Figure 8 and Figure 9.

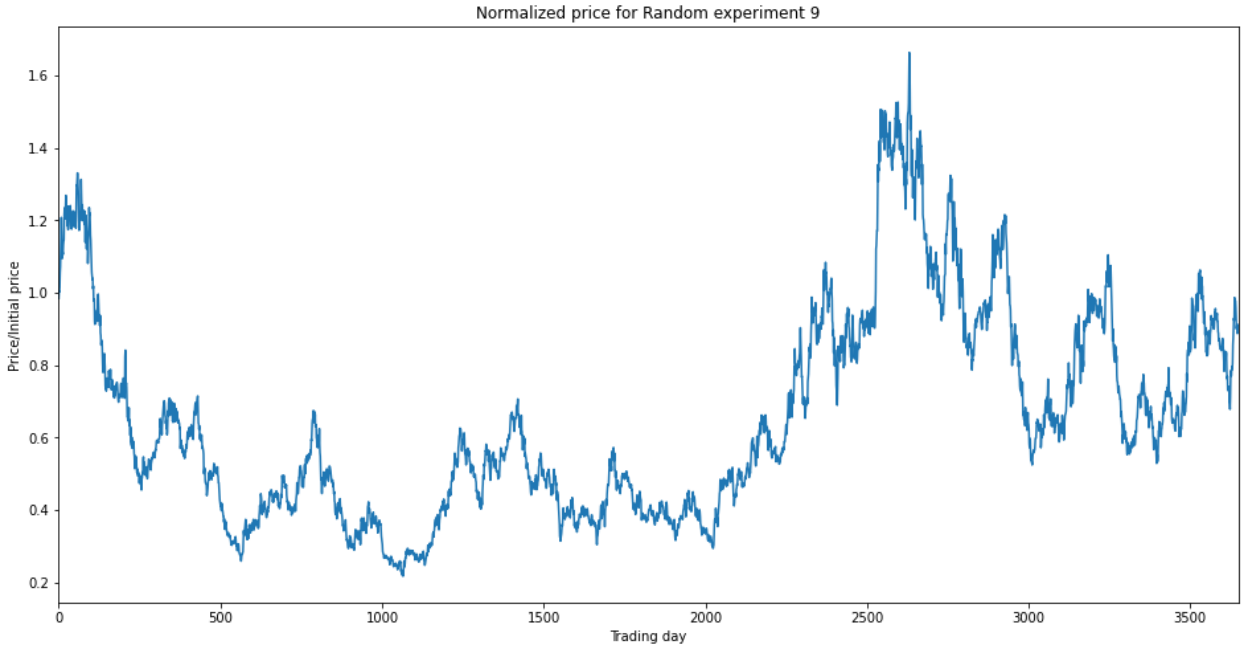

Figure 11. The graph shows a synthesized, normalized sample stock price for all trading days. In this graph we can observe that the generated stock prices look like actual stock prices.

5. Method

We built a single layer per output recurrent neural network with a hidden dimension of 20 and trained it on training data. With a trained model available we estimated its performance on unseen data by letting it predict future volatility. To quantitatively assess its performance, the same procedure was made with two well-used models for volatility estimation, an ARCH and a GARCH model, as well and compared the results of the three models.

The recurrent neural network used in the study has a hidden state dimension of 20. The input used is the logarithm of the change factor. To use both the hidden state and the input they are concatenated to create a combined state. The network then uses two different linear transforms activated with Softplus to create a new hidden state and output standard deviation.

Softplus

$$\text{softplus}(x) = \log(1 + \exp(x))$$

Hidden state at time d:

$$h_d = \left(h_{d,1}, h_{d,2}, \dots, h_{d,20}\right)^\top$$

Which means that the standard deviation at time i+1 is predicted the following manner

$$input_d = \left(h_{d,1}, h_{d,2}, \dots, h_{d,20}, log(r_d)\right)^\top$$
$$\sigma_{d+1} = \text{softplus}(A \cdot input_d + B)$$
$$h_{d+1} = \text{softplus}(C \cdot input_d + D)$$

Where A is a 1 by 21 matrix, B a scalar, C is a 20 by 21 matrix and D is a 20 by 1 matrix, all randomly initialized.

To avoid the problem of overfitting, we used a train, validation and test split of the data. The training dataset consisted of the initial 70 % trading day prices, the validation data contains the

next 20 %, and the test data contains the last 10 %. The different datasets are used in the following way: the model's parameters are adjusted as to increase the optimization objective (the log-likelihood) evaluated on the training data set until the loss log-likelihood no longer increase on the validation set. To get an unbiased estimation of the model's performance, the model is then used to predict volatility for the test set. The optimization objective was the following, where the expression inside the logarithm is the likelihood of the log returns in the dataset it is evaluated on:

$$\max_{\text{parameters}} \log\left(\prod_{d\in P}\frac{1}{\sigma_d\sqrt{2\pi}}\exp\left(-\frac{1}{2}\left(\frac{\log(r_d)}{\sigma_d}\right)^2\right)\right)$$

P is the set of trading days on which to evaluate the loss. We maximized the log likelihood with a numerical optimizer called ADAM that iteratively update the parameters of the model. ADAM has some parameters that can be fine-tuned but we used the default values in PyTorch, a library for graph calculations in which we constructed the RNN. We chose ADAM because it is a well performing numerical optimizer.(Kingma, Ba 2014)

For numerical reasons, we used the log-likelihood to provide more stability. This is because the joint likelihood of all datapoints consists of a product of their respective probability. As all probabilities are smaller than one, this number quickly becomes small with the number of data points. With the log-likelihood, this product becomes a sum of the logarithms for the individual probabilities which do not become small in the same way. For consistency in objectives it is worth noting that the logarithm is a monotonic function, so if the log-likelihood is extremized, it corresponds to the maximization of the regular likelihood. (Bishop 1994)

We trained the ARCH and GARCH models in a similar manner, but instead of dividing the data into three parts as before, these models were trained on the first 90 % of the datapoints and then they did a prediction on the last 10 %, like the RNN. This is because they are less prone to overfitting and that this way of updating the parameters is not supported by the programming package that we used. For the optimization of the ARCH and GARCH models, the package's inbuilt optimizer was used with its default values.

To compare our model with the classic volatility estimation models, we used some different error metrics. An error is the difference between the predicted value and the actual value in a regression. An error metric aims to aggregate all errors a model makes for all data points. In the formulas below, P is the set of all trading days, and d is an index in the set P.

RMSE is the root of the average squared error. This is an error measurement that gives large errors greater importance. While being more sensitive to outliers than other measures it gives a better picture when larger errors are relatively less desired.

$$RMSE = \sqrt{\frac{1}{|P|}\sum_{d \in P}(y_d - f_d)^2}$$

MSE is the average squared error. It is the RMSE but squared and their properties are almost identical, except that MSE cannot be compared to MAE as it represents a squared error.

$$MSE = RMSE^2$$

MAE is the average absolute error. This gives equal weight to all error measures and is less prone to outliers. This error measurement provides a more natural interpretation of the error compared to RMSE. (Willmott, Matsuura 2005)

$$MAE = \frac{1}{|P|}\sum_{d \in P}|y_d - f_d|$$

MAPE is the average percentage error. That is, the error is divided by the actual value. This makes the error measurement more interpretable.

$$MAPE = \frac{1}{|P|}\sum_{d \in P}\frac{|y_d - f_d|}{|y_d|}$$

MaxAE is the largest absolute error and can be used to see how a model performs at its absolute worst.

$$MaxAE = \max_{d \in P}|y_d - f_d|$$

R^2 tells us how much of the variance in the dependent variable that can be predicted from the independent variable relative to all variance in the dependent variable. (Miles 2014) R^2 can be thought to measure the goodness of fit and as opposed to the error metrics, higher R^2 is better.

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} = 1 - \frac{\sum_{d \in P}(y_d - f_d)^2}{\sum_{d \in P}(y_d - \bar{y})^2}$$

To compare the performance of the model with the performance of the ARCH and GARCH model we did a pairwise test of the difference in error metrics as well as the coefficient of determination, R^2. We tested one-sidedly if the mean difference is non-zero. The statistical significance for the difference in performance for the models was also tested for.

6. Results

The ARCH model, the GARCH model and the RNN were trained on 40 different generated datasets to be able to compare their performance. In the graphs, we can see a sampled experiment and, in the tables, we can see aggregations for all experiments.

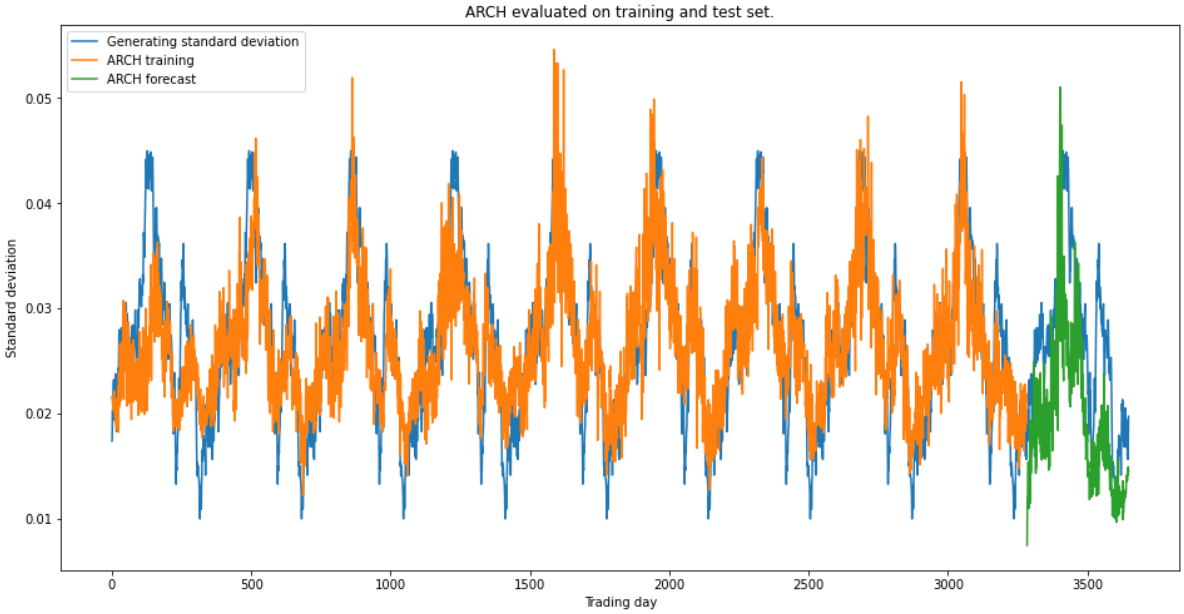

Figure 12. This is the training and test set for the ARCH model for random experiment 8. This can provide some illustration to the tables of metrics.

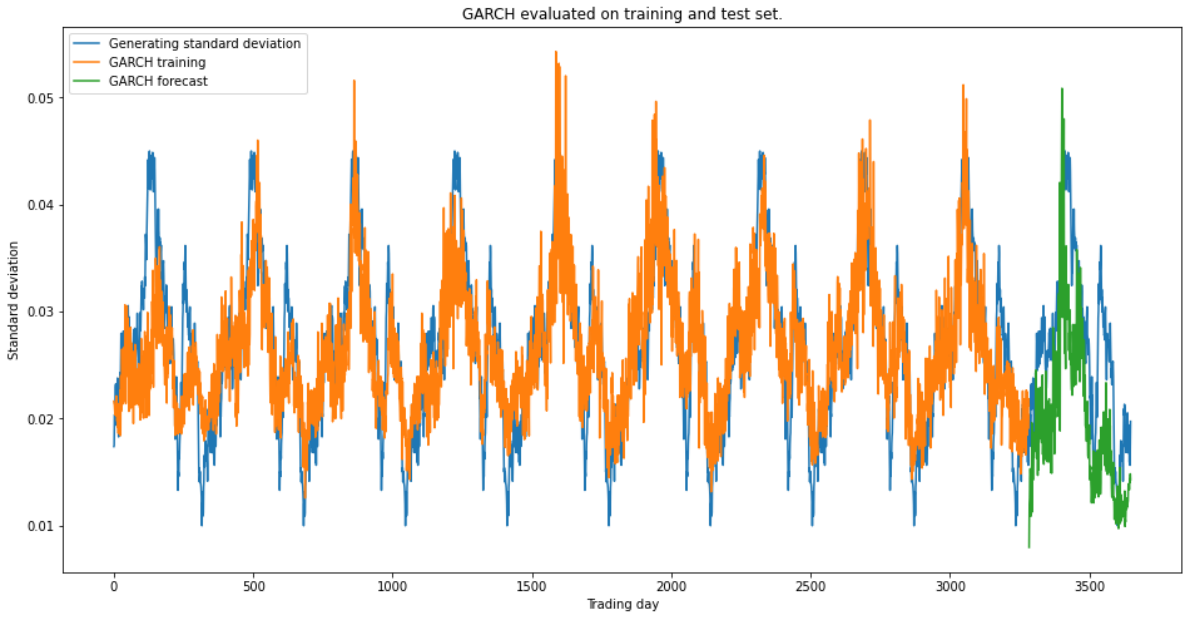

Figure 13. This is the training and test set for the GARCH model for random experiment 8. This can provide some illustration to the tables of metrics.

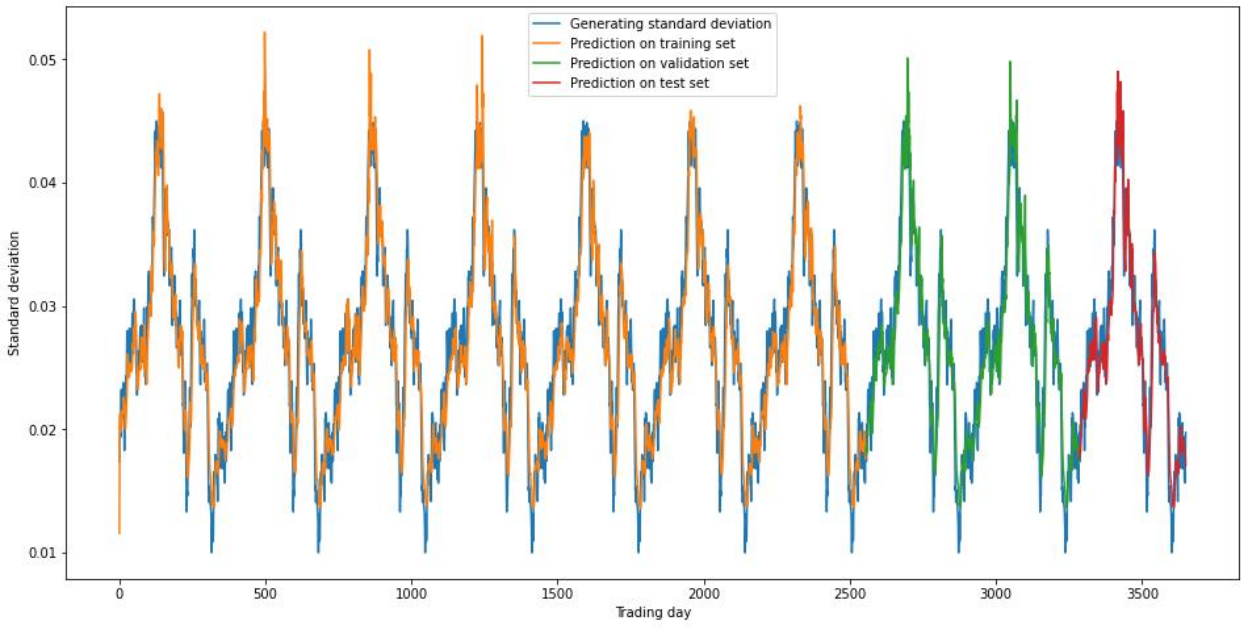

Figure 14. This is the training and test set for the RNN model for random experiment 8. This can provide some illustration to the tables of metrics.

From each training session, the metrics were recorded as previously described, and the results could be found below.

Average value	Arch	Garch	RNN
MAE	0.0062	0.0060	0.0023
MAPD	0.2272	0.2235	0.0929
MaxAE	0.0265	0.0252	0.0102
MSE	0.0001	0.0001	0.0000
R^2	0.5889	0.6306	0.8707
RMSE	0.0076	0.0073	0.0029

Figure 15. This table shows the average value for each metric recorded in all experiments for every model. The RNN model model consistently performs better than the GARCH mode which performs better than the ARCH model.

Standard deviation	Arch	Garch	RNN
MAE	0.00103	0.00100	0.00037
MAPD	0.03939	0.03998	0.01432
MaxAE	0.00963	0.00868	0.00321
MSE	0.00002	0.00002	0.00000
R^2	0.14877	0.13642	0.05645
RMSE	0.00127	0.00120	0.00045

Figure 16. This table shows the standard deviation for each metric. We note that the GARCH model has a lower standard deviation in its performance than the ARCH model but a larger one than RNN for all metrics.

To see if these error metrics differ significantly for the models, a significance test was performed to test the hypothesis that an error metric of RNN is higher than the same error metric for the other models and lower for the performance metric R^2. This was done through a pairwise difference test at a significance level of 95 %.

For all error metrics we want to test if:

$$H_0: \mu_{RNN} \geq \mu_{ARCH} \quad H_1: \mu_{RNN} < \mu_{ARCH}$$
$$H_0: \mu_{RNN} \geq \mu_{GARCH} \quad H_1: \mu_{RNN} < \mu_{GARCH}$$

And for R^2 we want to test the opposite, namely:

$$H_0: \mu_{RNN} \leq \mu_{ARCH} \quad H_1: \mu_{RNN} > \mu_{ARCH}$$
$$H_0: \mu_{RNN} \leq \mu_{GARCH} \quad H_1: \mu_{RNN} > \mu_{GARCH}$$

The pairwise t-statistics for the ARCH and GARCH model can be found in the table below together with the critical value for the t-statistics.

Metric	t-statistics ARCH - RNN	t-statistics GARCH - RNN	Critical t-value
MAE	8.091	7.754	1.685
MAPD	19.359	18.723	1.685
MaxAE	11.626	11.905	1.685
MSE	15.406	15.577	1.685
R^2	16.961	16.030	1.685
RMSE	23.004	22.872	1.685

Figure 17. This table shows the t-statistics for the hypothesis test that the RNN-model would perform worse than ARCH and GARCH at each metric together with the critical t-value. All t-statistics are larger than the critical value, so the hypothesis is rejected at a confidence level of 95 %.

As all observed t-statistics exceed the critical value we reject the null hypothesis that RNN would perform worse than ARCH. We also reject the null hypothesis that RNN would perform worse than GARCH for the same reason.

Standard deviation	ARCH Lower limit	ARCH Upper limit	GARCH Lower limit	GARCH Upper limit	RNN Lower limit	RNN Upper limit
MAE	0.00085	0.00133	0.00082	0.00129	0.00030	0.00048
MAPD	0.03227	0.05058	0.03275	0.05133	0.01173	0.01839
MaxAE	0.00789	0.01237	0.00711	0.01114	0.00263	0.00412
MSE	0.00002	0.00003	0.00002	0.00002	0.00000	0.00000
R^2	0.12187	0.19102	0.11175	0.17517	0.04624	0.07248
RMSE	0.00104	0.00163	0.00099	0.00155	0.00037	0.00058

Figure 18. This table shows the confidence interval for each model and metric at a 95 % confidence level.

Looking at Figure 18 we can see that the upper limit on the RNN confidence interval for each error metric is lower than the lower limit for both the ARCH and GARCH model at the confidence level of 95 %. To see if the difference in standard deviations is statistically significant, we test the following hypotheses.

$$H_0: \sigma_{ARCH} = \sigma_{RNN} \quad H_1: \sigma_{ARCH} \neq \sigma_{RNN}$$

$$H_0: \sigma_{GARCH} = \sigma_{RNN} \quad H_1: \sigma_{GARCH} \neq \sigma_{RNN}$$

The result of the hypothesis test can be found in Figure 19.

Metric	F-statistic ARCH/RNN	F-statistic GARCH/RNN	F-critical	p-Value
MAE	7.7663	7.3089	0.5867	3.4035E-09
MAPD	7.5656	7.7905	0.5867	5.1746E-09
MaxAE	9.0159	7.3117	0.5867	2.9757E-10
MSE	59.327	49.443	0.5867	1.5776E-25
R^2	6.9454	5.8405	0.5867	1.9936E-08
RMSE	7.8394	7.0448	0.5867	2.9278E-09

Figure 19. Hypothesis test of difference in standard deviation in metrics for the models. As all F-statistics exceed the critical value we reject the hypothesis that the standard deviations are the same at a confidence level of 95 %.

7. Conclusions

This study aims to explore the possibility of volatility forecasting with the help of a Recurrent Neural Network. Our results indicate that RNN outperforms ARCH and GARCH. The mean absolute error is about a third of that of ARCH and GARCH meaning that the model performs better on average. The RMSE error which is related to MAE tells a similar story. It also shows that the RNN does not perform better on the average error by only shifting its distribution of errors to create more small errors at the expense of some larger errors. By looking at the largest absolute error, we see that it is about half for RNN compared to ARCH and GARCH. This means that at its absolute worst, RNN is still a better model than both the ARCH and GARCH model.

The mean average percentage deviation shows that the RNN performs better not only in absolute terms but also relative to the value in each data point. Finally, as the RNN has a higher R^2 than ARCH and GARCH, we could conclude that the RNN model is better at explaining the variance in the time-series volatility.

Our tests of the standard deviations in the error and performance metrics also show that while RNN not only performs better than ARCH and GARCH in terms of the metrics employed, it also performs significantly different in terms of the standard deviation in all metrics. Given that its standard deviation is lower for all metrics, we assume that it performs better in terms of the standard deviation, but we cannot test this statistically.

The purpose of this paper was to examine if RNNs would provide knowledge in the research of volatility estimation. With our results, we have proven that this is the case, and this implies that RNNs could be a valuable tool for this purpose in the future, both in research, but also for businesses.

8. Discussion

As previously brought up, our analysis is based on synthesized stock price data, which therefore brings the question of the external validity of the results. As we have sampled a large number of different time-varying but yearly repeating stock volatilities our results should not be only due to selection bias in the volatility functions. However, we assumed that the change in volatility was uniformly distributed at each point in time. Figure 7 shows what the generating standard deviation looks like for an experiment, this can be compared to Figure 8 in which some calculated real-world standard deviations are shown. Although they look slightly less jagged, they were calculated with a 31-day rolling window which smoothens some of the irregularities. By decreasing the window size, the graphs look more similar which implies that this assumption made to create the time-varying standard deviations is reasonable. Arguably, this could prove to be detrimental to external validity if the real-world distribution of log-returns and volatilities might differ from the distribution in this study. Since we created the generating volatility in a certain manner, we cannot guarantee that our conclusions hold for other types of generating

volatilities. With that said, as neural networks are universal approximators, it should in theory be possible to model any kind of generating volatility, although for some cases ARCH or GARCH might yield better results.

An assumption made in the synthetization process was that the data was without a trend. While real stock world data usually have trends, there are methods than can remove the trends in time series data. (Watson 1986) This would allow the RNN model to perform well on such data.

The stocks that we used were chosen to reflect several American industries. By choosing only American stocks this study might be subject to selection bias. While it does not necessarily have to be the case, the conclusions from this study might not be applicable on, for example, Asian companies. However, as the stocks were only used to find span in which the volatility lies and the model should in theory be able to predict volatilities in any scale this the span itself is not detrimental external validity as one could simply scale the log-returns on other stocks for the volatility to match the American volatility span.

The stocks were also chosen from companies that have been around for at least 10 years. Although this gives us more proof of repeating seasonal volatility it also raises the question of survivorship bias. Fortunately, as we are interested in estimating the seasonal volatility and not testing if it exists, this does not affect the external validity of the experiment.

Therefore, we conclude that some assumptions were made in the synthetization of data that limit the applicability of this method on real-world data. However, if methods for removing trends in data are used, none of the assumptions should diminish the external validity of the results.

It is worth noting, that the model's performance on the other component of the volatility, the non-repeating part, which cannot be explained by the date data alone, is unlikely to be properly estimated by any model, including this, as long as the model cannot take other data into account. Fortunately, (recurrent) neural networks are useful in that way as they can take several types of input into account. With more factors known to affect the volatility in the model it should, in theory, be possible to estimate that component to some degree as well. By incorporating more

variables than the log return in the input, the model should also be able to handle effects due to the day of the week, holidays and dividend payments. As this was not performed in our experiment, these conclusions are not fully applicable for stocks were these effects occur unless the model is given more variables as inputs.

References

ARAGONÉS, J.R., BLANCO, C. and ESTÉVEZ, P.G., 2007. Neural network volatility forecasts. *Intelligent Systems in Accounting, Finance & Management: International Journal,* **15**(3- 4), pp. 107-121.

BAUWENS, L., HAFNER, C. and LAURENT, S., 2012. *Handbook of volatility models and their applications.* Hoboken, N.J: John Wiley & Sons, Inc. **3**(1) pp. 1-33, 50-61

BISHOP, C.M., 1994. Mixture density networks. Aston University. pp. 4

BLACK, F. and SCHOLES, M., 1973. The pricing of options and corporate liabilities. *Journal of political economy,* **81**(3), pp. 637-654.

BOSSU, S., 2014. *Advanced equity derivatives: volatility and correlation.* Hoboken, New Jersey: John Wiley & Sons.

CLEEREMANS, A., SERVAN-SCHREIBER, D. and MCCLELLAND, J.L., 1989. Finite state automata and simple recurrent networks. *Neural Computation* **1**(1) pp. 372-381

CONNOR, J.T., MARTIN, R.D. and ATLAS, L.E., 1994. Recurrent neural networks and robust time series prediction. *IEEE Transactions on Neural Networks,* **5**(2), pp. 240-254.

BALÁZS, C.C., 2001. Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungar,* **48**(24) pp. 22.

ENGLE, R., 2001. GARCH 101: The Use of ARCH/GARCH Models in Applied Econometrics. *The Journal of Economic Perspectives,* **15**(4), pp. 157-168.

Engle RF. Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. *Econometrica*. **1982;50**(4), pp. 987-1007.

HE, K., ZHANG, X., REN, S. and SUN, J., 2016. Deep residual learning for image recognition, *Proceedings of the IEEE conference on computer vision and pattern recognition* 2016, pp. 770-778.

KINGMA, D.P. and BA, J., 2014. Adam: A method for stochastic optimization. *Conference paper at ICLR 2015, pp. 1-8*

KROPKO and JONATHAN, 2016. *Mathematics for Social Scientists.* Reading [u.a]: Taylor & Francis.

LIU, P., QIU, X. and HUANG, X., 2016. Recurrent neural network for text classification with multi-task learning. *Shanghai Key Laboratory of Intelligent Information Processing, Fudan University*, pp. 2873-2879

MILES, J., 2014. R squared; adjusted R squared. *Wiley StatsRef: Statistics Reference Online.*

P. BENNETT, KRISTIN, PARRADO-HERNANDEZ and EMILIO, 2006. The Interplay of Optimization and Machine Learning Research. *Journal of Machine Learning Research, p*p 1-8.

PATTON, A.J., 2011. Volatility forecast comparison using imperfect volatility proxies. *Journal of Econometrics,* **160**(1), pp. 246-256.

POON, S. and GRANGER, C.W.J., 2003. Forecasting Volatility in Financial Markets: A Review. *Journal of Economic Literature,* **41**(2), pp. 478-479, 508.

RAZALI, N.M. and WAH, Y.B., 2011. Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests. *Journal of statistical modeling and analytics*, **2**(1), pp. 21-33.

SCHITTENKOPF, C., DORFFNER, G. and DOCKNER, E.J., 1998. *Volatility prediction with mixture density networks.* SFB Adaptive Information Systems and Modelling in Economics and Management Science, WU Vienna University of Economics and Business, pp 1-15.

SHAIKH, I. and PADHI, P., 2013. A Simultaneous Equation Approach on the Relationship between Implied, Realised and Historical Volatility. *Asia Pacific Journal of Management Research and Innovation*, **9**(2), pp. 139-155.

TOM and MITCHELL, 1997. *Machine Learning*, McGraw Hill, pp. 15-30, 59-70

WATSON, M.W., 1986. Univariate detrending methods with stochastic trends. *Journal of Monetary Economics* **18**(1), pp 49-75

WILLMOTT, C.J. and MATSUURA, K., 2005. Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. *Climate research,* **30**(1), pp. 79-82.

Wikipedia (2020). *Recurrent Neural Networks*. https://en.wikipedia.org/wiki/Recurrent_neural_network (accessed 2020-04-15)

Appendices

The following table shows all results from all experiments.

	MAE	MAPD	MaxAE	MSE	R2	RMSE
ARCH						
Random experiment 1	0.004797	0.168841	0.017958	0.000037	0.603296	0.006073
Random experiment 10	0.006099	0.209510	0.032092	0.000059	0.351146	0.007660
Random experiment 11	0.006269	0.251495	0.021158	0.000055	0.498935	0.007412
Random experiment 12	0.005790	0.198231	0.037481	0.000058	0.696003	0.007633
Random experiment 13	0.006050	0.228928	0.027024	0.000054	0.630323	0.007350
Random experiment 14	0.005914	0.193823	0.019407	0.000053	0.376869	0.007285
Random experiment 15	0.007052	0.271594	0.037397	0.000068	0.767637	0.008218
Random experiment 16	0.005826	0.259690	0.023957	0.000044	0.795160	0.006641
Random experiment 17	0.006763	0.231955	0.032238	0.000069	0.431564	0.008330
Random experiment 18	0.007862	0.270888	0.022294	0.000081	0.570284	0.009023
Random experiment 19	0.005935	0.223174	0.022094	0.000050	0.672470	0.007073
Random experiment 2	0.007174	0.263078	0.040676	0.000079	0.677145	0.008896
Random experiment 20	0.005393	0.225680	0.016961	0.000043	0.796113	0.006538
Random experiment 21	0.004270	0.151795	0.017662	0.000029	0.571733	0.005357
Random experiment 22	0.005629	0.239889	0.016177	0.000045	0.763026	0.006696
Random experiment 23	0.005208	0.175726	0.023792	0.000045	0.499391	0.006683
Random experiment 24	0.009572	0.335641	0.024580	0.000117	0.400672	0.010832
Random experiment 25	0.005397	0.231659	0.019493	0.000041	0.741731	0.006409
Random experiment 26	0.006569	0.249351	0.050169	0.000089	0.759197	0.009424
Random experiment 27	0.004866	0.203768	0.032287	0.000039	0.780877	0.006231
Random experiment 28	0.006027	0.205587	0.022384	0.000053	0.526094	0.007297
Random experiment 29	0.005332	0.185045	0.024959	0.000045	0.579728	0.006683
Random experiment 3	0.008581	0.334722	0.021451	0.000093	0.653198	0.009636
Random experiment 30	0.004970	0.180167	0.016393	0.000037	0.332852	0.006074
Random experiment 31	0.007879	0.253968	0.059704	0.000130	0.625732	0.011392
Random experiment 32	0.006855	0.245520	0.021611	0.000070	0.413813	0.008372
Random experiment 33	0.005924	0.213152	0.023654	0.000054	0.391888	0.007331
Random experiment 34	0.005660	0.199252	0.015600	0.000043	0.692317	0.006576
Random experiment 35	0.006212	0.226734	0.025728	0.000057	0.602229	0.007568
Random experiment 36	0.006260	0.237327	0.031299	0.000058	0.329412	0.007626
Random experiment 37	0.006567	0.251429	0.018708	0.000061	0.633598	0.007781
Random experiment 38	0.005760	0.214252	0.025449	0.000050	0.517968	0.007054
Random experiment 39	0.005497	0.187496	0.014374	0.000042	0.632974	0.006507
Random experiment 4	0.006361	0.219951	0.041864	0.000074	0.473117	0.008627
Random experiment 40	0.006729	0.249052	0.032167	0.000077	0.706583	0.008757
Random experiment 5	0.006296	0.250003	0.028137	0.000056	0.720483	0.007458
Random experiment 6	0.005538	0.187573	0.028036	0.000052	0.760467	0.007186

Random experiment 7	0.006316	0.242686	0.022452	0.000065	0.703166	0.008082
Random experiment 8	0.006510	0.247840	0.030430	0.000061	0.603156	0.007822
Random experiment 9	0.004933	0.173239	0.019274	0.000039	0.271683	0.006251
GARCH						
Random experiment 1	0.004614	0.168012	0.015555	0.000032	0.693274	0.005698
Random experiment 10	0.005384	0.175972	0.033918	0.000047	0.564969	0.006892
Random experiment 11	0.006239	0.250568	0.021092	0.000054	0.506122	0.007372
Random experiment 12	0.005786	0.198129	0.037560	0.000058	0.696041	0.007604
Random experiment 13	0.006038	0.228319	0.026760	0.000054	0.632376	0.007332
Random experiment 14	0.005892	0.193187	0.019382	0.000053	0.379746	0.007259
Random experiment 15	0.007027	0.270553	0.037628	0.000067	0.766883	0.008195
Random experiment 16	0.005759	0.257280	0.024261	0.000043	0.799122	0.006577
Random experiment 17	0.005427	0.202493	0.021113	0.000042	0.680529	0.006505
Random experiment 18	0.007213	0.253029	0.019897	0.000068	0.661278	0.008245
Random experiment 19	0.005827	0.224335	0.018187	0.000046	0.756479	0.006794
Random experiment 2	0.007195	0.268330	0.040751	0.000077	0.692803	0.008779
Random experiment 20	0.005323	0.222573	0.016910	0.000042	0.797737	0.006467
Random experiment 21	0.004184	0.147733	0.017465	0.000028	0.594233	0.005245
Random experiment 22	0.005618	0.239146	0.016482	0.000045	0.762371	0.006682
Random experiment 23	0.005220	0.176156	0.023866	0.000045	0.498297	0.006694
Random experiment 24	0.009485	0.332829	0.024735	0.000115	0.409724	0.010735
Random experiment 25	0.005109	0.221659	0.019009	0.000036	0.769910	0.006039
Random experiment 26	0.006344	0.247965	0.039451	0.000075	0.818024	0.008683
Random experiment 27	0.005539	0.235797	0.028946	0.000046	0.794180	0.006790
Random experiment 28	0.005989	0.204148	0.022320	0.000053	0.526496	0.007258
Random experiment 29	0.005305	0.184082	0.024323	0.000044	0.582735	0.006649
Random experiment 3	0.008217	0.325357	0.019584	0.000083	0.707368	0.009136
Random experiment 30	0.004896	0.177040	0.015918	0.000035	0.357650	0.005950
Random experiment 31	0.007679	0.249010	0.053126	0.000116	0.714173	0.010791
Random experiment 32	0.006599	0.242248	0.020809	0.000062	0.515369	0.007889
Random experiment 33	0.005943	0.213744	0.023680	0.000054	0.392884	0.007349
Random experiment 34	0.005632	0.198190	0.015541	0.000043	0.692717	0.006543
Random experiment 35	0.006160	0.224242	0.025366	0.000056	0.602580	0.007512
Random experiment 36	0.006252	0.237014	0.031289	0.000058	0.329175	0.007618
Random experiment 37	0.005754	0.232785	0.017985	0.000047	0.750155	0.006837
Random experiment 38	0.005751	0.213848	0.025448	0.000049	0.519542	0.007035
Random experiment 39	0.005420	0.185134	0.014241	0.000041	0.635918	0.006425
Random experiment 4	0.006375	0.225330	0.038941	0.000069	0.536168	0.008319
Random experiment 40	0.006729	0.248821	0.032684	0.000076	0.703878	0.008735
Random experiment 5	0.006266	0.249259	0.028527	0.000055	0.724410	0.007432
Random experiment 6	0.005484	0.186100	0.027815	0.000051	0.760998	0.007115
Random experiment 7	0.006265	0.245293	0.022249	0.000063	0.728473	0.007931

Random experiment 8	0.006065	0.237474	0.028519	0.000053	0.693698	0.007286
Random experiment 9	0.004246	0.148377	0.015502	0.000029	0.474178	0.005346
RNN						
Random experiment 1	0.002087	0.088861	0.006967	0.000007	0.905363	0.002613
Random experiment 10	0.002791	0.096900	0.019589	0.000014	0.745476	0.003691
Random experiment 11	0.002237	0.096883	0.008756	0.000008	0.826489	0.002805
Random experiment 12	0.001740	0.066697	0.010017	0.000005	0.939415	0.002258
Random experiment 13	0.002554	0.106968	0.010782	0.000010	0.857034	0.003201
Random experiment 14	0.002491	0.093670	0.008202	0.000010	0.803824	0.003118
Random experiment 15	0.002146	0.081531	0.015230	0.000008	0.924186	0.002769
Random experiment 16	0.001630	0.068737	0.008546	0.000004	0.957113	0.002079
Random experiment 17	0.002076	0.077714	0.006826	0.000007	0.899938	0.002587
Random experiment 18	0.002576	0.099562	0.012862	0.000010	0.847785	0.003178
Random experiment 19	0.002213	0.088378	0.009760	0.000008	0.907112	0.002826
Random experiment 2	0.002142	0.083189	0.007321	0.000007	0.908901	0.002672
Random experiment 20	0.002052	0.081031	0.014923	0.000007	0.926137	0.002719
Random experiment 21	0.002165	0.084727	0.006832	0.000007	0.870392	0.002639
Random experiment 22	0.001887	0.076108	0.009734	0.000006	0.929395	0.002402
Random experiment 23	0.003008	0.121312	0.010236	0.000014	0.771197	0.003788
Random experiment 24	0.003022	0.113173	0.010248	0.000014	0.812662	0.003709
Random experiment 25	0.002156	0.095408	0.010968	0.000008	0.909157	0.002843
Random experiment 26	0.002510	0.102971	0.017749	0.000012	0.904120	0.003471
Random experiment 27	0.002554	0.109449	0.011728	0.000012	0.885037	0.003402
Random experiment 28	0.002558	0.097552	0.009905	0.000010	0.824541	0.003084
Random experiment 29	0.002579	0.101258	0.008158	0.000010	0.825061	0.003184
Random experiment 3	0.001688	0.070583	0.013698	0.000005	0.931191	0.002208
Random experiment 30	0.002548	0.100556	0.008824	0.000010	0.786272	0.003117
Random experiment 31	0.002671	0.096747	0.015719	0.000012	0.891842	0.003459
Random experiment 32	0.002485	0.102378	0.008025	0.000009	0.825209	0.003051
Random experiment 33	0.002192	0.089827	0.008739	0.000007	0.856621	0.002708
Random experiment 34	0.001787	0.069757	0.005987	0.000005	0.914162	0.002186
Random experiment 35	0.002429	0.102519	0.011115	0.000009	0.893155	0.002993
Random experiment 36	0.002389	0.104141	0.008872	0.000009	0.768520	0.003017
Random experiment 37	0.002430	0.107538	0.007227	0.000009	0.885327	0.002930
Random experiment 38	0.002477	0.100174	0.009524	0.000010	0.783273	0.003119
Random experiment 39	0.003372	0.132241	0.010544	0.000016	0.897696	0.004015
Random experiment 4	0.002393	0.087152	0.006865	0.000008	0.858411	0.002906
Random experiment 40	0.002105	0.083927	0.007331	0.000007	0.927323	0.002620
Random experiment 5	0.001925	0.081864	0.010165	0.000006	0.923961	0.002356
Random experiment 6	0.002185	0.082281	0.015002	0.000008	0.911995	0.002783
Random experiment 7	0.002293	0.093261	0.007067	0.000008	0.917287	0.002777
Random experiment 8	0.001974	0.079778	0.007651	0.000006	0.899230	0.002489

Random experiment 9	0.002620	0.097623	0.008314	0.000010	0.776055	0.003184