LUND UNIVERSITY

SCHOOL OF ECONOMICS AND MANAGEMENT

Predicting Exchange Rate Value-at-Risk and Expected Shortfall: A Neural Network Approach

Master Thesis in Finance, June 2019

School of Economics and Management, Lund University

Anna Bijelic & Tilila Ouijjane

Supervisors: Anders Vilhelmsson & Sara Moricz

Abstract

On the basis of the recommendation of the Basel Committee on Banking Supervision to transition from Value-at-Risk (VaR) to Expected Shortfall (ES) in determining market risk capital, this paper attempts to investigate whether a Recurrent Neural Network provides more accurate VaR and ES predictions of the EUR/USD exchange rate compared to the conventional GARCH(1,1) model. A number of previous studies has confirmed the forecasting ability of a plain vanilla Feedforward Neural Network over traditional statistical models. However, standard neural networks have limitations. Most notably, they rely on the assumption of independency among data observations, which presents a problem when data points are related in time. To circumvent this restriction, this study employs a Gated Recurrent Unit type of neural network to produce one-step-ahead volatility forecasts of the EUR/USD exchange rate, which are then used to compute VaR and ES predictions. The VaR and ES forecasts for both models are obtained through a Volatility Weighted Historical Simulation, and evaluated with backtesting procedures. The empirical results indicate that the GARCH(1,1) model outperforms the Gated Recurrent Unit neural network for $VaR_{95\%}$, while the Gated Recurrent Unit neural network appears more adequate in forecasting ES at a 95% confidence level.

Keywords: Value-at-Risk, Expected Shortfall, Recurrent Neural Networks, GRU, GARCH(1,1), Exchange Rate Volatility, Intra-day Data

Acknowledgements

We would like to express our gratitude to our supervisors, Anders Vilhelmsson and Sara Moricz. Your assistance, comments and ideas have provided valuable insights in the fulfilment of this study.

Table of Contents

1. Introduction

The 2007-2008 global financial crisis has shed light on the deficiencies of traditional risk management, resulting in a desire to improve risk management tools and practices. The Fundamental review on the trading book (2013) is a popular reform initiative taken by the Basel Committee on Banking Supervision to mark its ambition to revise and strengthen the regulatory standards for banking institutions. Through this consultative document, the Basel Committee supports a change to the risk measure used for determining market risk capital, that is, a transition from Value-at-Risk (VaR) to Expected Shortfall (ES).

In the last two decades, VaR and – to a lesser extent – ES have been used as downside risk measures, where both measures contributed greatly to the amelioration of traditional risk assessment. A wide range of studies has focused on estimating and predicting VaR and ES for different financial instruments, among them foreign exchange rates. Since the abandonment of the Bretton Woods system, and thereby the advent of the floating exchange rate system, the behaviour of exchange rates has become increasingly volatile and complex, making its non-linearities more difficult to capture. Consequently, the interest in understanding the movements on the Foreign Exchange (FOREX) market has increased, as a deeper understanding of foreign exchange rate volatility is crucial for the decision-making of both monetary policy makers and practitioners. This explains, in conjunction with the rekindled interest for further research in forecasting risk, the importance of forecasting both VaR and ES on exchange rates. For this purpose, two main approaches are identified in the existing literature: conventional volatility forecasting models and Artificial Neural Networks (ANNs).

The non-linear GARCH model proposed by Bollerslev (1986) and Taylor (1986) is often used to forecast exchange rate volatility. Nonetheless, a considerable disadvantage of the GARCH(1,1) model is that it is conceptualized with an explicit assumption of the underlying functional relationship of the data series at hand, *i.e.* it is not model-free. Another non-linear model that has gained tremendous popularity in the recent years, and that has been acknowledged as an attractive and powerful alternative to traditional volatility models is the Artificial Neural Network (ANN). In

contrast to conventional volatility forecasting models, an ANN has a flexible non-linear function mapping capability, and does not require a prior assumption of the functional relationship of the dataset. Specifically, an ANN is a machine-learning based model inspired by the human brain, which is able to classify patterns and make predictions based on past experience. The two main models within the concept of ANNs are the Feedforward Neural Network (FNN) and the Recurrent Neural Network (RNN), where the main difference between the two lies in the way information circulates within the neural network. Unlike the FNN, the RNN allows for loops within the network, meaning that the prediction an RNN node makes at time step t affects the prediction it will make one moment later, at time step $t+1$.

The characteristics of RNNs suggest that it should be preferred over FNNs when analyzing time-series data. As a matter of fact, several authors, such as Connor et al. (1993) and Dunis and Huang (2002), affirm that RNNs accommodate time series data better than FNNs. However, in the preponderance of the academic literature, the use of a FNN is surprisingly the prevailing choice, despite the suggested superiority of RNNs. Furthermore, no justification is provided to explain the choice of using a FNN when studying foreign exchange rates. One possible reason could be the computational difficulties one faces when implementing a RNN, as it requires a larger number of connections and more memory in comparison to a standard FNN. Additionally, a plethora of existing research papers focuses on forecasting exchange rate returns, rather than predicting their second moment. As argued by Dunis and Huang (2002), previous research on implementing a neural network approach "has been so far seldom devoted to FX volatility forecasting". Surprisingly, and to the best of our knowledge, no previous paper has examined the forecasting ability of a RNN on exchange rate volatility for predicting *both* VaR and ES. In an attempt to fill this theoretical void, the objective of this study is to compare VaR and ES predictions of the EUR/USD exchange rate, obtained by:

1) predicting exchange rate volatility with a Recurrent Neural Network

2) predicting exchange rate volatility with the traditional GARCH(1,1) model

The remainder of this paper is organized as follows. The next section presents the preliminary theory pertaining to the objective of the paper. The third section provides a review of the most prominent literature on VaR and ES, the two risk measures of interest in this study, as well as on different volatility forecasting models, such as the GARCH(1,1) model. Moreover, an evaluation of the prevalent literature on ANNs is provided in this section. The fourth section presents a detailed explanation of the method employed to model the RNN architecture, and to estimate and predict VaR and ES. The fifth section presents the empirical results along with their possible implications. Final conclusions and suggestions for further research are discussed in the last section.

2. Theoretical Background

2.1 Value-at-Risk and Expected Shortfall

2.1.1 Definition

Value-at-Risk (VaR) is a measure used to assess downside risk, which depends on two parameters: the time horizon T and the confidence level α. VaR is usually calculated from the probability loss distribution, where positive values are interpreted as losses and negative values as profits. VaR is defined as the smallest loss l in a portfolio, such that the probability of a future portfolio loss L, which is larger than l, is less than or equal to $1 - \alpha$:

$$VaR_{\alpha,t}(L_t) = min\{l: \Pr(L_t > l) \leq 1 - \alpha\} \qquad (1)$$

Following this definition, VaR can statistically be interpreted as the α-quantile of the probability loss distribution, with α being the chosen confidence level:

$$VaR_{\alpha,t}(L_t) = q_\alpha(L_t) \qquad (2)$$

Expected Shortfall (ES) is derived such that it takes the average of all VaR values over all confidence levels above α. As such, ES calculates the expected loss at time T conditional on the future portfolio loss L being greater than the α-quantile of the loss distribution:

$$ES_{\alpha,t}(L_t) = E\big(L: L > VaR_{\alpha,t}\big) \qquad (3)$$

In contrast to VaR, ES quantifies the expected loss beyond the VaR level by considering the largest losses in a portfolio, *i.e.* losses larger than $VaR_{\alpha,t}$. Following the mathematical expression of ES presented above, a VaR estimate or forecast must be calculated in order to compute an ES estimate or forecast. For this purpose, parametric and non-parametric approaches are of use.

2.1.2 Parametric and Non-parametric Approaches

Parametric approaches assume that portfolio losses follow a pre-determined distribution, such as a Normal distribution or a Student t-distribution, whereas non-parametric approaches do not depend on any distributional assumption. Instead, non-parametric approaches focus directly on the underlying loss distribution of a portfolio, by relying on a sample of observed losses. As a result, these approaches are applicable to any instrument, while parametric approaches, such as the Normal distribution, might not be. For example, under normality, the probability loss distribution is assumed to have no excess kurtosis, however, it has been empirically observed that financial data indeed exhibits excess kurtosis. Thus, assuming normally distributed losses when using financial data may lead to incorrect VaR and ES estimates.

On the other hand, non-parametric approaches are too dependent on the sample of observed losses. They may provide underestimated VaR and ES estimates or forecasts if the chosen sample reflects a relatively stable period or otherwise, overestimated VaR and ES estimates if the chosen sample reflects a stressed period. Additionally, an important drawback of non-parametric approaches is the assumption of identically and independently distributed returns, which has empirically been proven untrue, due to the existence of serial correlations and volatility clustering. In practice, financial institutions use non-parametric approaches to calculate VaR and ES for market risk, where the Basic Historical Simulation (BHS) and the Volatility Weighted Historical Simulation (VWHS) are among the most popular ones.

Through the BHS, calculations of VaR and ES estimates are based on the empirical loss distribution, consisting of collected past data. Each loss observation is given the same probability, meaning that older loss observations are as relevant as newer loss observations. The BHS approach expects the number of losses larger than $VaR_{\alpha,t}$ to be equal to $(1-\alpha)N$, with N representing the number of sample loss observations. Thus, $VaR_{\alpha,t}$ is estimated to be the $(1-\alpha)N+1$ largest loss in the sample and $ES_{\alpha,t}$ is computed as an average of the $(1-\alpha)N$ largest losses.

The VWHS approach, introduced by Hull and White (1998), applies a BHS to a sample of rescaled losses, and is considered more relevant for time-series data than the BHS. By using rescaled losses, the VWHS takes current market conditions into account, which is highly relevant when analyzing time dependent series, such as financial data. Indeed, financial data manifests signs of volatility clustering, which is a phenomenon describing the fact that if volatility in the current holding period is lower than the average, it will likely be lower than the average in the next holding period as well. The reverse is also true with an observed volatility that is higher than the average. Through the VWHS approach, VaR and ES reflect these specific volatility patterns, as a result of the rescaling of the sample losses with a volatility forecast:

$$l_T^R = \frac{\sigma_{T+1}}{\sigma_T} \cdot l_T \qquad (4)$$

where l_T^R is the rescaled loss at time T, σ_{T+1} the volatility forecast for the next holding period and σ_T the volatility associated to the sample loss l_T observed at time T.

Among the various existing volatility models, GARCH (1,1) and the Exponentially Weighted Moving Average (EWMA) approaches are often employed to obtain volatility forecasts. A BSH is subsequently applied to the rescaled sample of losses, in order to calculate VaR and ES.

2.1.3 Backtesting Value-at-Risk and Expected Shortfall

Following the estimation of VaR and ES, backtesting is of great importance as it allows one to assess the efficiency of a risk measure. Kupiec (1995) introduces the main ideas behind the backtesting procedure for VaR, where the so-called Kupiec frequency test, an exact binomial test, compares the actual frequency of VaR violations, denoted k, with the predicted frequency of VaR violations. For a given day, a VaR violation; also called *exception* (Hull (2006), Acerbi and Szekely (2014)), occurs when the observed loss is larger than the VaR estimate. Accordingly, the number of predicted VaR violations should then be equal to $(1-\alpha)N$, which is the expected number of losses larger than the VaR estimate. Since the number of VaR

violations has a binomial distribution, the probability of observing $X \leq k$ violations under the assumption of a correct underlying model is defined as follows:

$$\Pr(X \leq k) = \sum_{i=0}^{k} \binom{N}{i} p^i (1-p)^{N-i} \qquad (5)$$

where p is the probability of a VaR violation on any given day. If the calculated probability is less than $(1-\alpha)$, *i.e.* the significance level of interest, the underlying VaR model is statistically rejected, that is to say that the actual frequency of VaR violations deviates significantly from the predicted one. In practice, backtesting VaR proves itself to be conceptually easier than backtesting ES, which is arguably less straightforward. As mentioned previously, the difficulty to backtest ES is the main reason to why financial institutions have been reluctant to support the Basel Committee's initiative to choose ES as the standard risk measure over VaR.

Acerbi and Szeleky (2014) propose three different backtests for ES, and according to the authors, these tests "introduce no conceptual limitations nor computational difficulties of any sort". The second test called "testing ES directly" is computationally attractive, as it requires the estimation of only two parameters: the one-day ahead ES and the magnitude of a loss if a VaR violation occurs $(L_t I_t)$. Moreover, the critical level associated with the second test is remarkably stable across various distributional assumptions for a 5% confidence level, and is estimated to be -0.70. The aforementioned mathematical expression of ES can be rewritten as:

$$ES_{\alpha,t}(L_t) = E\big(L: L > VaR_{\alpha,t}\big) = \frac{E(L_t I_t)}{1-\alpha} \qquad (6)$$

Given a confidence level α, the indicator function I_t is defined such that it takes on the value 1 in the case of a VaR violation and the value 0 if no VaR violation occurs:

$$I_t = \begin{cases} 1 & if \quad L_t > VaR_{\alpha,t}(L_t) \\ 0 & if \quad L_t < VaR_{\alpha,t}(L_t) \end{cases} \qquad (7)$$

Under the null hypothesis, the underlying ES-model is correct, implying that it provides an efficient ES estimate. The test statistic Z proposed by Acerbi and Szeleky (2014) is defined such that:

$$Z = -\frac{1}{T(1-\alpha)}\sum_{t=1}^{T}\frac{L_t I_t}{ES_{\alpha,t}(L_t)} + 1 \qquad (8)$$

If the null hypothesis is rejected, the underlying model underestimates ES. As mentioned previously, the authors suggest comparing the value of the test statistic with a critical value of -0,70 when a 95% confidence level is chosen.

2.2 GARCH(1,1)

As previously indicated, it has empirically been observed that financial data is subject to the phenomenon of volatility clustering due to the time-varying characteristic of the variance of financial returns. An appropriate stochastic process to model volatility clustering would therefore be one with a non-constant conditional variance. A GARCH (1,1) model fulfils this condition, as it allows the conditional variance to be dependent on its past values. Thus, a GARCH model does not only provide a measure of the conditional mean, but of the conditional variance as well. Assuming that the return of a financial asset, r_t, depends on a stochastic error term, η_t, and on the expected return, μ, then the mean equation, the variance equation and the expression of the residuals used in the GARCH(1,1) are defined respectively as:

$$r_t = \mu + \eta_t \qquad (9)$$

$$\sigma_t^2 = \omega + \alpha\eta_{t-1}^2 + \beta\sigma_{t-1}^2 \qquad (10)$$

$$\varepsilon_t = \frac{\eta_t}{\sigma_t^2} \sim t_{\alpha,v^1} \qquad (11)$$

As time-varying variance is assumed, *i.e.* non-linearity, Maximum Likelihood (ML) is used to estimate the relevant parameters. ML is an optimization method that maximizes the likelihood of a set of parameters θ, defined in this case as $\theta = \omega, \alpha, \beta, \mu$. Some restrictions on the parameters should be set, in order to allow for mean-reverting volatility and positive conditional variance: $\alpha + \beta < 1$ and $\omega \geq 0, \alpha \geq 0, \ \beta \geq 0$. The log likelihood function for a Student t-distribution is such that:

$$\ln L(\mu, \omega, \alpha, \beta) = Tln\left[\frac{\Gamma\left(\frac{v+1}{2}\right)}{\sqrt{\pi(v-2)\Gamma\left(\frac{v}{2}\right)}}\right] - \frac{1}{2}\sum_{t=1}^{T}\ln(\sigma_t^2) - \frac{(v+1)}{2}\sum_{t=1}^{T} ln\left[1 + \frac{\varepsilon_t^2}{\sigma_t^2(v-2)}\right] \qquad (12)$$

[1] v, degrees of freedom.

After the optimization of the parameters, the estimated values are the ones that maximize the likelihood that θ produced the data that was actually observed. These values are then used to produce volatility estimates.

2.3 Artificial Neural Networks

An Artificial Neural Network, ANN, is a deep learning system that loosely mimics the human brain's ability to classify patterns and make predictions based on past experience. In general, an ANN builds on a collection of connected nodes called neurons, which schematically resemble the neurons in the biological brain. Just like the synapses in the brain, each connection in an ANN can transfer a signal from one neuron to another (Gately, 1996).

However, unlike the brain which relies on inputs from the five senses, the inputs in an ANN are real numbers from labelled datasets. Object recognition is a common example of a task for an ANN, where the network is presented with a number of images of a certain type, and by analyzing the recurring patterns in the presented objects, the network learns to categorize new images.

2.3.1 Neural Network Architectures

The manner in which the input neurons produce a certain output is intimately linked to the structure of the neural network. In terms of foreign exchange rate forecasting, two fundamentally different classes of network architectures are identified, both of which will be outlined below.

2.3.1.1 Multilayer Feedforward Neural Network

In a multilayer feedforward neural network, the neurons are organized into one input layer, one or more hidden layers, and one output layer, where each neuron in a particular layer is connected with all neurons in a subsequent layer. The information flow in the network is of feedforward type, meaning that the output from one layer of neurons feeds forward into the next layer of neurons. In other words, the connections can never skip a layer, or form any loops backwards. The architectural layout of a multilayer feedforward neural network for the case of a single hidden layer is presented in *Figure 1*.

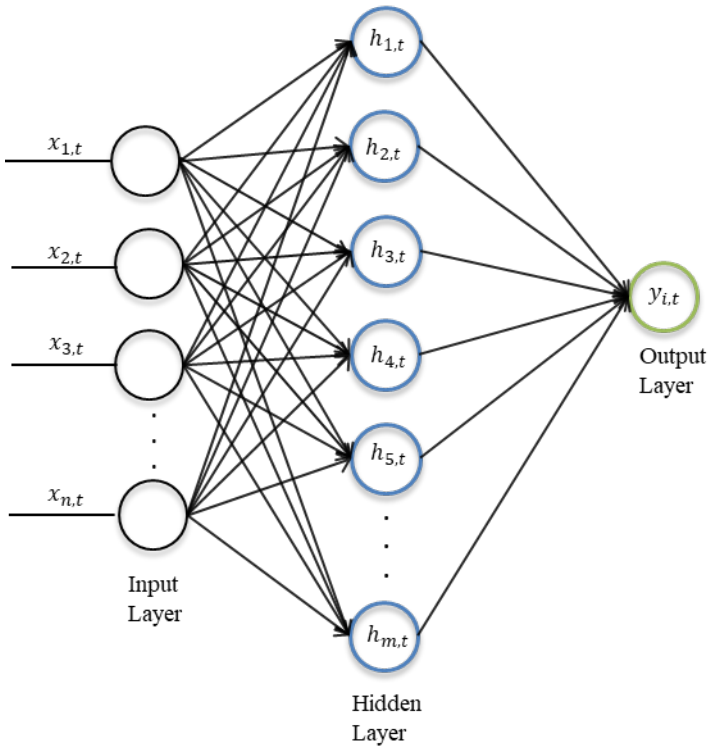


Figure 1: Overview of a fully connected multilayer feedforward neural network with one hidden layer.

As shown in *Figure 1*, input values are forward propagated into the hidden layer through connections, each being characterised by a certain weight coefficient, $w_{i,j}$. These weight coefficients reflect the degree of importance of a given connection between an input node, $x_{i,t}$, and a hidden node, $h_{j,t}$. By defining the input signals as a vector $[x_{1,t}; x_{2,t}; \ldots; x_{n,t}]$ and the values of the hidden nodes as a vector $[h_{1,t}; h_{2,t}; \ldots; h_{m,t}]$, the transformation of the input nodes to one hidden node can mathematically be described by:

$$h_{j,t} = \sum_{i=1}^{n} w_{i,j} \cdot x_{i,t} \; for \; j = 1,2, \ldots, m \qquad (13)$$

An apparent undesired property of the formula is given by its linear representation, which, if applied, would suggest that the output prediction would be a linear function. Consequently, in order to deal with the non-linear characteristics inherent to most real world data, a non-linear activation function, $\emptyset(\cdot)$, is applied to the weighted sum of inputs into a hidden node. This activation function, which in the majority of applications takes the form of a *sigmoid* function or a *ReLu* function, makes the neural

network capable of approximating virtually any function. However, before applying the activation function, a bias vector $[b_1; b_2; \ldots; b_m]$ is added, which essentially indicates whether a neuron tends to be active or inactive in the prediction process. The transformation from the input layer to the hidden layer in a feedforward neural network can then be reformulated to:

$$h_{j,t} = \emptyset\left(b_{j,0} + \sum_{i=1}^{n} w_{i,j} \cdot x_{i,t}\right) for\ j = 1,2,\ldots,m \qquad (14)$$

In order to determine the accuracy of the feedforward neural network, a loss function, $L(\cdot)$, is introduced. This function is ultimately a measurement of how wrong the neural network is in terms of its ability to estimate the relationship between the inputs and a particular output. Typically, the loss function is expressed as the difference between the predicted output and the expected output, where the robustness of the network increases with the decrease of the loss function. The most commonly used loss functions are the Mean Absolute Error (MAE), the Mean Squared Error (MSE) and the Mean of the Fourth Power Error (MFPE), defined as:

$$L_{MAE} = \frac{1}{N}\sum_{k=1}^{N} |\hat{y}_k - y_k| \qquad (15)$$

$$L_{MSE} = \frac{1}{N}\sum_{k=1}^{N} (\hat{y}_k - y_k)^2 \qquad (16)$$

$$L_{MFPE} = \frac{1}{N}\sum_{k=1}^{N} (\hat{y}_k - y_k)^4 \qquad (17)$$

where $\hat{y}_k$ is the predicted output and y_k is the true output for a training set k = 1, 2, …, N. The primary objective of the feedforward neural network training process is therefore to find the weights and biases that minimize the loss function. For this purpose, Haykin (2009) and Bishop (2006) endorse the use of the gradient descent through a backpropagation optimization algorithm by emphasizing its practical simplicity, but also its computational efficiency. The general idea behind the gradient descent is to optimize the weights in the network by computing the partial derivative

of the loss function with respect to the weights and biases. By doing so, each weight's contribution to the loss function is determined. The partial derivatives are given by:

$$\left[\frac{\partial L}{\partial w_{1,1}}, \dots, \frac{\partial L}{\partial w_{n,m}}, \frac{\partial L}{\partial b_1}, \dots \frac{\partial L}{\partial b_n}\right]$$

The sign of each partial derivative conclusively determines in which direction the weights should be nudged in order to incrementally decrease the produced total error of the network. If the sign is positive, it means that it negatively impacts the loss; hence the weight should be decreased. Similarly, if the sign is negative, the weight should be increased. The modification of weights is performed iteratively until the discrepancy between the predicted output and the expected output reaches its minimum point, at which the training process stops.

The magnitude of the updated weights is determined by a learning rate hyperparameter. As explained by Haykin (2009), a lower learning rate implies that the changes to the synaptic weights will be smaller from one iteration to the next, resulting in a smoother trajectory. However, this improvement is attained at the cost of excessive computational time. On the other hand, a learning rate set too high may lead to instability of the network as the resulting large weight steps may overshoot the minimum point. As a result, this leaves the designer of the network to try different heuristics based on trial and error.

Although the model-free assumption underlying the feedforward neural network theoretically suggests that it should fare better than the conventional GARCH(1,1) in predicting exchange rate volatility, the feedforward neural network is subject to a major concern in that it precludes modelling time-dependencies in the data. This deficiency of not being able to take correlations between inputs into account is however resolved in the *recurrent neural network*, which is able to selectively pass information across sequences of elements by creating cycles in the network.

2.3.1.2 Recurrent Neural Network

Contrary to multilayer feedforward neural networks, Recurrent Neural Networks (RNN) are able to handle sequential data due to the capability of each neuron to maintain information about previous inputs. This means that the prediction a recurrent neural network node made at time step *t-1* affects the prediction it will make one moment later, at time step *t*. By taking as inputs not only the current signal, but also what has been perceived previously in time, RNN nodes can be thought of as having "memory". Adding memory to neural networks has an important purpose: because there may be information in the data sequence itself, recurrent nets are able to use it to perform prediction tasks that feedforward networks cannot.

In order to preserve information from one node to another while reading in inputs, RNNs contain feedback loops from the so-called hidden states. This feedback loop mechanism occurs at each time step in the data series, which results in each hidden state containing traces not only of the previous hidden state, but also of all the preceding ones, for as long as the memory of the network persists (Skymind, 2019). To better describe the distinction between feedforward neural networks and recurrent neural networks, a representation of an unrolled RNN is presented in *Figure 2*.

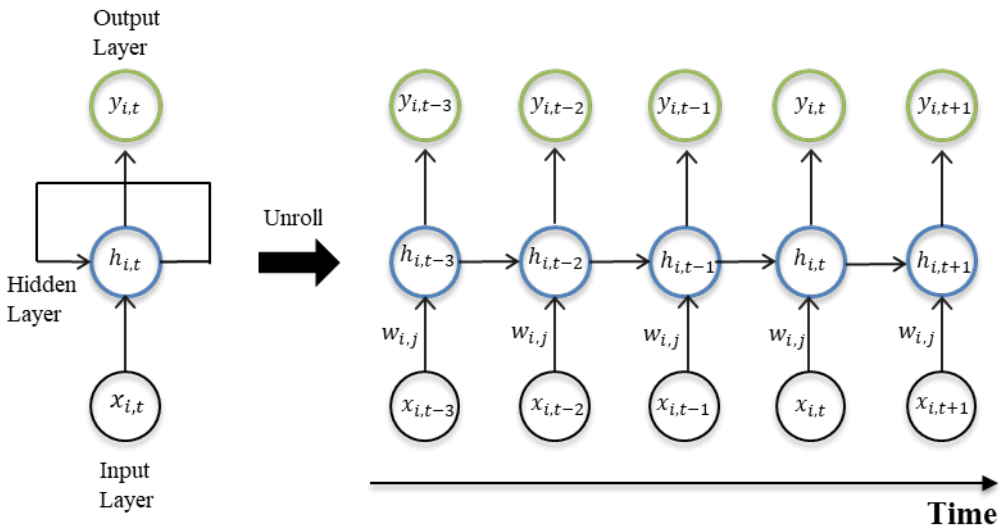


Figure 2: Representation of an unrolled plain vanilla recurrent neural network.

The unrolled RNN illustrates how the network allows the hidden neurons to see their own previous output, so that their subsequent behavior can be shaped by previous responses (Tenti, 1996). Furthermore, by introducing time-lagged model components, it becomes evident that the utilization of a RNN is particularly desired when there are time dependencies in the data series at hand.

Using the previous notation and assuming that the hidden states are the ones looped back, the output from a hidden node in the RNN model depends on the input values at time *t,* but also on its own lagged values at order *p* as shown below:

$$h_{j,t} = \emptyset\left(b_{j,0} + \sum_{i=1}^{n} w_{i,j} \cdot x_{i,t}\right) + \sum_{j=1}^{m} \gamma_j \cdot h_{j,t-p} \ \ for\ j = 1,2,\dots,m \qquad (18)$$

where $h_{j,t-p}$ represents the lagged hidden state values at order p, and γ_j a coefficient. The training process of a RNN is similar to the one of a multilayer feedforward network. The standard backpropagation algorithm is used to update the weight connections, however, because the gradient at each output depends on the calculations at all previous time steps, the value of the gradient has a tendency to vanish when approaching the earliest time steps. Effectively, this prevents the weights of the earlier inputs from being adjusted, which is a serious issue as it deteriorates the training process and thereby degrades the performance of the network. As a consequence, the plain vanilla RNN might forget what it has seen in longer sequences, thus having a short-term memory.

2.3.1.3 Gated Recurrent Unit

The Gated Recurrent Unit (GRU) network is a sophisticated type of recurrent network, explicitly designed to combat the long-term dependency problem inherent to the plain vanilla RNN. To solve the vanishing gradient problem, the GRU network contains internal mechanisms called gates, which regulate how previous information should be incorporated into the current output. In particular, the architecture is composed of an *update gate* and a *reset gate* connected to the hidden states. The update gate helps the model to determine how much of past information needs to be passed along to the future, while the reset gate is used to decide how much of past information should be forgotten. As such, these gates act on signals they receive, where they block or pass on information based on its strength and importance. If the memory content is considered important, the reset gate will be closed and carry the content across several consecutive time steps, which is equivalent to capturing a long-term dependency (Chung, 2015). In contrast, a neuron may decide to reset the memory content by opening the forget gate.

In analogy with the input nodes and hidden states, the update gates and reset gates have their own set of weights, which are adjusted via the recurrent network learning process. That is, the gates learn when to allow content to enter, or be deleted through an iterative process *via* the gradient descent algorithm. Owing to the attractive characteristics of the GRU, this particular network will be used to forecast volatility of the EUR/USD exchange rate. To illustrate how data flows and is controlled by the gates, a short sequence of the GRU recurrent network is presented in *Figure 3,* and a single GRU cell is displayed in *Figure 4*.

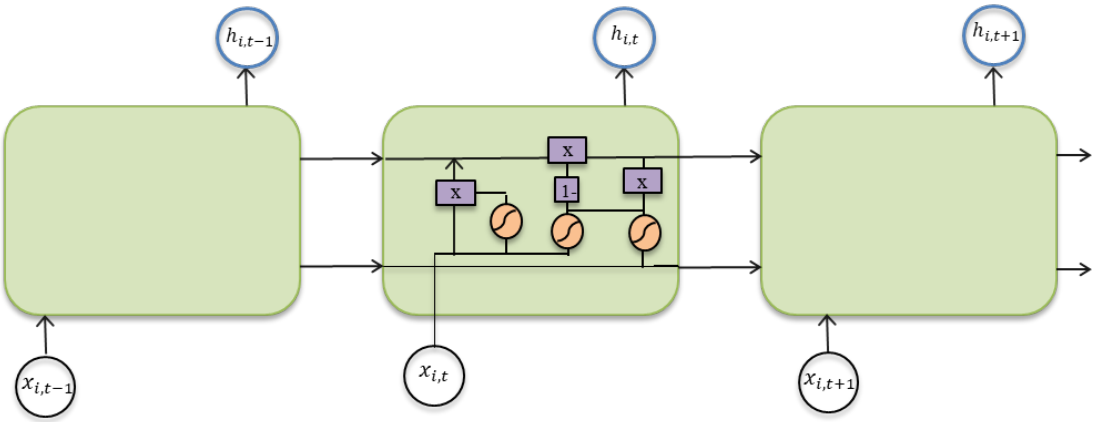


Figure 3. Structure of the GRU network chain. The operations within the GRU cell eliminate the vanishing gradient problem as the model keeps relevant information and passes it down to the next time steps of the network. In the figure, the GRU cell combines new input, $x_{i,t}$, with the previous memory contained in $h_{i,t-1}$ to produce the new hidden state, $h_{i,t}$

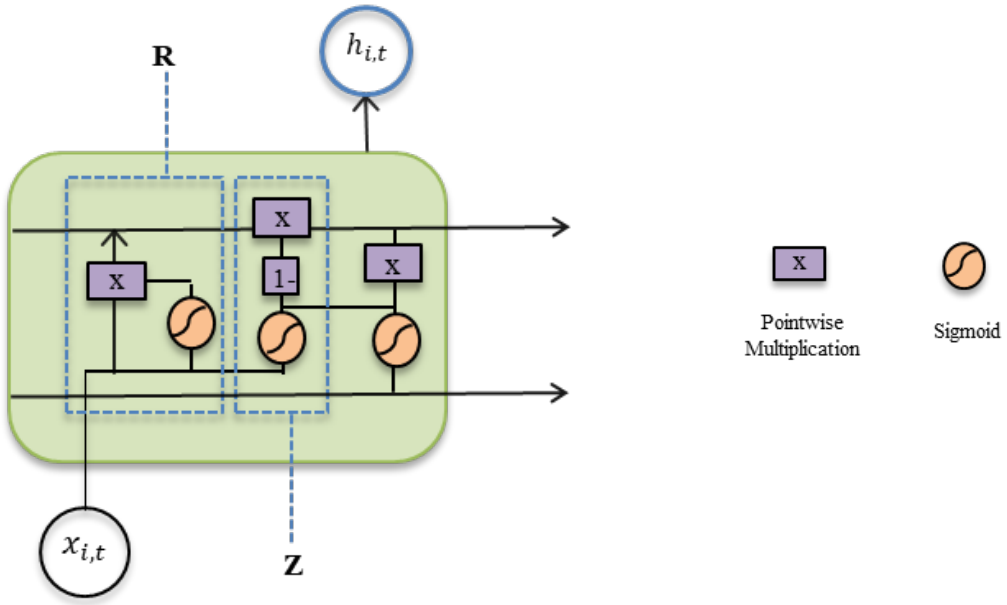


Figure 4. A visualization of the information flow in a single GRU cell with an activation function set to sigmoid. Each line in the figure carries an entire vector from the output of one node to the input of another. R and Z denote the reset and update gate, respectively. For a deeper understanding of the operations within the GRU cell, the reader is referred to Nguyen (2018).

3. Literature Review

The purpose of this section is to present a comprehensive overview of the most renowned literature on value-at-risk and expected shortfall, statistical forecasting models and artificial neural networks. The chapter begins by exploring the cardinal studies covering the concepts of value-at-risk and expected shortfall. This is followed by an evaluation of the prevalent literature on conventional exchange rate risk forecasting models. Thereafter, the body of literature covering the use of artificial neural networks to provide volatility predictions is introduced, while outlining weaknesses inherent to previous studies completes the section.

3.1 Value-at-Risk and Expected Shortfall

3.1.1 Advantages and Shortcomings of Value-at-Risk

In financial risk management, Value-at-Risk (VaR) is acknowledged as a standard and comprehensive measure to estimate the total risk in a portfolio. According to Yamai and Yoshiba (2002), the attractiveness of VaR lies in "its conceptual simplicity, computational facility, and ready applicability". Indeed, VaR is easily interpretable as it quantifies portfolio risk in a single number (Jorion, 1997). Additionally, VaR is applicable to any financial instrument and uses the same unit of measurement, "money loss". However, various studies challenge the effectiveness of VaR as a risk measure and argue that VaR exhibits several conceptual drawbacks.

Artzner et al. (1997, 1999) demonstrate the conceptual problems inherent to VaR by using two different situations; a short position on digital options and a concentrated credit portfolio. Based on their results, Artzner et al. (1997, 1999) conclude that VaR is not subadditive when normality is not assumed, and takes no account of tail-events if they occur, nor provide information on their severity. As a remedy to these conceptual drawbacks, Aztner et al. (1997,1999) introduce the concept of a *coherent* risk measure, which satisfies several desirable properties, such as subadditivity. However, Aztner et al. (1997, 1999) and Rootzén and Klüppelberg (1999) point out

that the pertinency of the subadditivity problem to risk managers depends on their preferences and can be irrelevant to some.

Whereas the relevancy of the subadditivity problem is contingent on risk managers' discretion, the so-called fat tails problem is recognized as the most problematic shortcoming of VaR. Indeed, the characteristics inherent to the tails of the loss distribution are of importance in financial risk management. Danielsson et al. (1998) examine the ability of VaR to model tails' behavior and explain that risk managers are concerned about the occurrence and size of large losses beyond VaR, especially when there is suspicion of a fat-tailed distribution. More precisely, modelling the behavior of the right tail of the loss distribution is crucial in financial risk management, considering that events occurring in the right tail are threats to institutional solvency, as explained by the Basel Committee on Banking Supervision (2013).

3.1.2 Transition from Value-at-Risk to Expected Shortfall

In response to these critics, other risk measures were developed and introduced, such as the worst conditional expectation (WCE), presented by Artzner et al. (1999). The WCE was introduced as a coherent risk measure, which can capture the behavior of the loss distribution. However, despite being coherent, WCE lacks practicality due to computational difficulties. Acerbi et al. (2001) succeed to construct a risk measure that is coherent and straight-forward in its computation, namely Expected Shortfall (ES). They define ES as "the expected value of the loss of the portfolio in the 5% worst cases in 7 days", when considering a time horizon of 7 days and a probability of 5%. The mathematical definition provided by Acerbi et al. (2001) shows the subadditivity of ES in all cases, without any specific distributional assumption. Thus, Acerbi et al. (2001) claim that, if correctly estimated, ES is a satisfactory replacement of VaR for financial risk management uses.

Emmer et al. (2013) find that ES seems to be a superior risk measure, when compared to VaR and Expectiles. Yet, they put emphasis on the difficulty of backtesting ES, whereas backtesting VaR is rather uncomplicated. In contrast, Acerbi and Szekely (2014) claim that ES indeed can be backtested and introduce "three model-free, nonparametric backtest methodologies" for ES. They show that these backtesting

methods are superior to the standard Basel backtest for VaR, whereby the authors simultaneously answer to the criticisms made to the Basel Committee for choosing ES over VaR, despite its supposedly impossible backtesting. Indeed, the Basel Committee on Banking Supervision (2013) stated the necessity to transition from VaR to ES after identifying various deficiencies when using VaR to assess regulatory capital requirements. The Committee's decision to adopt ES as a standard risk measure over VaR reflects the various discussions regarding the coherence and relevancy of VaR as a risk measure.

3.2 Conventional Statistical Forecasting Models

3.2.1 Characteristics of Exchange Rates

Although the concepts of VaR and ES appear simple and rather straightforward, producing robust VaR estimates is in fact a challenging statistical task, as it requires accurate and reliable measures of volatility. Dating back to Mandelbrot (1963) and Fama (1965), it is now a well documented fact that time series of exchange rates tend to be characterized by leptokurtosis and conditional heteroscedasticity, the latter producing positive serial correlation in squared returns. These features of exchange rate series imply that the null hypothesis of independence can be strongly rejected, demonstrating the existence of non-linearities in exchange rates. Zhang et al. (1998) emphasize this fact by stating that forecasting time series has long been the domain of linear statistics. However, real world systems are often non-linear, hence it is unreasonable to assume a priori that a particular realization of a given time series is generated by a linear process.

3.2.2 GARCH(1,1): An Exchange Rate Volatility Forecasting Model

In the wake of the findings by Mandelbrot (1963) and Fama (1965), a broad array of studies was carried out in order to shed light on the phenomenon of volatility and capture the dynamics of time series data. The GARCH(1,1) model, originally proposed by Bollerslev (1986) and Taylor (1986), gained tremendous popularity and was considered ground-breaking due to its ability to capture the salient features of volatility in financial data. Nevertheless, an important characteristic inherent to the conventional GARCH(1,1) model concerns the symmetry in the conditional variance. As stated by Nelson (1991), the GARCH(1,1) model exhibits a fundamental shortcoming as it neglects the well-established property that returns are negatively correlated with changes in volatility, *i.e.* volatility tends to rise in response to bad news and fall in response to good news. In other words, GARCH models assume that only the magnitude and not the sign of unanticipated excess returns determine the conditional variance.

As a consequence of the impediments inherent to the traditional GARCH(1,1) model, different extensions have been proposed in the literature that allow for the presence of asymmetries and other departures from the standard GARCH specification. These include the E-GARCH model developed by Nelson (1991), the GJR-GARCH model of Glosten et al. (1993) and the T-GARCH by Zakoian (1991).

An extensive review of 93 papers on the forecasting ability between the asymmetric models and the traditional GARCH(1,1) is provided by Poon and Granger (2003). Across the evaluated sample, the authors find that models incorporating volatility asymmetries perform better than GARCH. However, the authors simultaneously stress upon the fact that an explicit consideration should be given to the use of varying data sets, different sampling frequencies and the variety of evaluation techniques. In contrast to the findings by Poon and Granger (2003), Hansen and Lunde (2005) find that the GARCH model produces better volatility predictions for exchange rates, but when the focus is switched towards stocks, models incorporating leverage effects are found to be more suitable. Closely related results have been reported by Liu and Hung (2010).

3.2.3 Drawbacks of Traditional Volatility Models

A considerable disadvantage with the standard volatility models is that they are conceptualized with an explicit assumption of the underlying functional relationship of the data series at hand, *i.e.* they are not model-free. As stated by Zhang et al. (1998), the formulation of a nonlinear model to a particular data set is in fact a major challenge due to the existence of a vast number of possible nonlinear patterns, implying that a certain pre-specified nonlinear model may not be general enough to capture all the important features of volatility. Consequently, conventional volatility models may be seriously mis-specified and may hence provide poor volatility forecasts. Moreover, as pointed out by Chen et al. (2009), with the growing globalization of capital markets, the empirical distributions of modern asset returns have become more and more complicated, making their market risk more difficult to capture. This thorny issue concerning the specification of a correct functional relationship has imposed the necessity to bring forward new alternative models that are less sensitive to model mis-specification

3.3 Artificial Neural Networks

3.3.1 Artificial Neural Network: A Model-free Approach

In the recent literature, a growing attention has been given to the prediction capability of artificial neural networks (ANNs). Cybenko (1989) and Hornik et al. (1989) state that one of the main advantages of ANNs is that they have a flexible nonlinear function mapping capability, meaning that any model can be arbitrarily well approximated by a sufficiently large ANN. This model-free property of ANNs is an important advantage since exchange rates do not display a specific nonlinear pattern. Haykin (2009) extend the notion of flexibility by emphasizing the adaptive capability of ANNs, which he argues provides a more robust performance when the system is required to operate in a non-stationary environment.

3.3.2 Exchange Rate Return Forecasting Using Artificial Neural Networks

Considerable research effort has gone into feedforward neural networks for predicting future values in the foreign exchange rate market. Zhang et al. (1998) employ a multilayer feedforward neural network in forecasting the GBP/USD exchange rate and find that the neural network outperforms linear models, particularly when the forecast horizon is short. Hann and Steuer (1996) find similar results when comparing the performance of feedforward neural network models with those of random walk and linear models in forecasting the USD/DEM exchange rate using both monthly and weekly data. The out-of-sample results show that, for weekly data, neural networks outperform the other comparable models. However, when monthly data is used, neural networks do not show much improvement over linear models, suggesting that the forecasting ability is contingent upon the sampling frequency. Nevertheless, when turning to the more recent strand of the literature domain, Kumar and Pradhan (2010) document empirical results that favour the implementation of neural networks irrespective of the sampling frequency. In their study, a feedforward neural network is applied to the Indian Rupee (INR) against the USD, GPB, EUR and JPY, using both daily and monthly data. The empirical results confirm the accurate prediction power of the neural network for both frequencies. Similar results are reported by Panda and Narasimhan (2007). In their study, they show that a feedforward neural network

outperforms linear autoregressive and random walk models when creating one-step-ahead return predictions of the weekly INR/USD exchange rate.

Another strand of the literature has emphasized on recurrent neural networks in predicting exchange rates. Kuan and Liu (1995) report a superior prediction accuracy of recurrent ANNs over feedforward ANNs in a study on five different currencies against the USD, suggesting that despite the requirement of a large number of connections and a large memory in the simulation, the richer dynamic structure of RNNs is able to yield significantly better results. Several other researchers have confirmed the superiority of RNNs over feedforward networks when performing exchange rate predictions (Connor et al. (1993), Tenti (1996)).

3.3.3 Exchange Rate Volatility Forecasting Using Artificial Neural Networks

By analyzing the literature domain on artificial neural networks, it can be recognized that there is an apparent lack of studies investigating exchange rate volatility through ANNs. In an attempt to fill this theoretical void, Dunis and Huang (2002) perform a comparative study between neural network regressions (NNR), recurrent neural networks (RNN) and the simpler GARCH(1,1) model, with an application to the GBP/USD and USD/JPY exchange rate volatilities. Using daily data from 1993-2000, the empirical results demonstrate that the RNN model appears as the single best modelling approach. In a more recent study, Lahmiri (2016) proposes an ANN model based on a set of technical indicators as inputs. In his study, the forecast accuracy of GARCH family models under different distributional assumptions is compared to the ANN in the context of USD/CAD and USD/EUR exchange rate volatilities. He finds that the proposed approach based on ANN with technical analysis indicators outperforms the conventional GARCH family models in terms of mean absolute error and mean of squared errors. Nevertheless, it is worthwhile to mention that although foreign exchange volatility forecasting through ANNs have gained some attention in the academic field, it still remains a fairly undeveloped area.

3.4 Estimating and Predicting Value-at-Risk and Expected Shortfall

Several attempts have been made to predict VaR on the foreign exchange rate market with conventional volatility models. Bredin and Hyde (2002) examine a standard variance covariance approach, an EWMA approach and an Orthogonal-GARCH approach in generating VaR forecasts of the Irish Punt against six different currencies. According to their results, the Orthogonal-GARCH model provides the most accurate VaR measures, yet they argue that the EWMA approach is the most appropriate as it produces more conservative estimates. In a more comprehensive study, Degiannakis and Potamia (2016) estimate both VaR and ES across a variety of financial markets; stock indices, commodities and foreign exchange rates, with the use of two distinctive volatility models. Their research, which is based on the AR(1)-GARCH(1,1) and the AR(1)-HAR-RV-skT models, utilizes a 95%, 97.5% and 99% confidence interval, with empirical results pointing towards the AR(1)-GARCH(1,1) in terms of prediction accuracy. The authors further recommend risk modelling at a confidence level of 97.5%, which is consistent with the proposed replacement of 99% VaR by 97.5 % ES by the Basel Committee.

VaR estimation on the exchange rate market in the context of ANNs is dealt with in Locarek-Junge and Prinzler (1999), who illustrate how VaR estimates can be obtained by using a USD-portfolio. The empirical outcomes demonstrate an evident superiority of the neural network to other VaR models. Similar results are put forth by He et al. (2018), who propose an innovative EMD-DBN type of ANN to estimate VaR on the USD against the AUD, CAD, CHF and the EUR. The authors find positive performance improvement in the risk estimates, and argue that the utilization of an EMD-DBN network can identify more optimal ensemble weights and is less sensitive to noise disruption compared to a FNN.

However, rather surprisingly, no previous paper has been found to estimate ES through ANNs on the foreign exchange rate market. Selected applications, such as Musah et al. (2018) and Sun et al. (2008), focus on the stock exchange market. Hence, the foreign exchange rate market remains unexplored, which justifies further investigation.

4. Methodology

The subsequent chapter presents the methodology of the implemented study for estimating one-day-ahead VaR and ES of the EUR/USD exchange rate. The first section describes the data employed in the study, and explains how the raw data is manipulated and the series of interest constructed. Following this, section 4.2 describes the approach taken to obtain one-day-ahead volatility predictions through the GRU neural network, while section 4.3 describes the implementation of the GARCH(1,1) benchmark model. Section 4.4 explains how the VaR and ES estimates are calculated. Finally, in order to assess the ex-post predictive performance of the two models, a backtesting framework is outlined in section 4.5.

4.1 Data Description and Data Preprocessing

Due to the rather low number of prevalent neural network applications evaluating a sample period longer than a decade, the EUR/USD exchange rate data used in this study encompasses a time frame beginning in *01.01.1999* and ending in *30.09.2018*, thus comprising a total number of 7212 days. As a result of the longer sample period, this study will show further uniqueness by investigating whether the forecasting capability of the neural network persists during various periods of global financial turmoils.

The one-step-ahead volatility forecasts used to obtain VaR and ES predictions are based on the EUR/USD exchange rate, which is extracted from a historical exchange rate database provided by Olsen Financial Technologies. The motivation for analyzing this particular exchange rate is given by the fact that it constitutes the world's most actively traded currency pair, making it extremely deep and liquid. In terms of sampling frequency, Hansen and Lunde (2004) state that there is an inherent trade-off between bias and variance when choosing frequency. Particularly, ultra-high frequencies such as tick data are subject to microstructure bid/ask frictions, whereas lower frequencies, although avoiding this problem, lose more and more information with longer sampling intervals. Following the proposition by Hansen and Lunde (2004), a moderate sampling frequency at 5-min intervals is chosen for this study.

In terms of data pre-processing, the EUR/USD data series analysed in this study is manipulated according to the following two steps.

Step one – Return Calculation. The following continuous compounding transformation is used in order to transform the data to return series, thereby keeping additivity in time:

$$r_{i,t} = \ln\left(\frac{FX_{i,t}}{FX_{i,t-1}}\right) \quad (19)$$

where $r_{i,t}$ is the return series on a 5-min frequency and $FX_{i,t}$ is the mid-quote EUR/USD exchange rate series. The daily return is subsequently obtained by summing the 5-min returns.

Step two – Calculation of Realized Variance. Due to the fact that volatility is latent, squared 5-min returns are used as a proxy for the variance. This conforms to the suggestion outlined by Andersen and Bollerslev (1998), who emphasize that although squared daily returns have been chosen as an indicator of daily variance in several previous studies, this measure is a noisy estimator, and should instead be replaced by the sum of intra-day squared returns. Their suggested alternative proxy, realized variance, is calculated as:

$$RV^{(N)}_{[t-1;t]} = \left[\sum_{i=1}^{N} {r_{i,t}}^2\right] \quad i = 1, \dots, 7112 \qquad (20)$$

where $RV^{(N)}_{[t-1;t]}$ is the realized variance during the interval $[t-1; t]$.

Figure 5 in Appendix A depicts the return series of the EUR/USD exchange rate over the total review period, while the distributional characteristics of the realized variance series are shown in *Figure 6* in Appendix A, in which prevalence of prominent volatility clustering can be observed. Notably, the effect of the global financial crisis in 2007-2008 appears to have had strong influence on the exchange rate volatility.

[INSERT FIGURE 5 AND 6 ABOUT HERE]

In terms of software implementations, all analysis concerning the GRU neural network is performed by using the Keras package in Python 3.6, with a Tensorflow backend. The GARCH(1,1) model is implemented in Eviews.

4.2 The GRU Neural Network Model

4.2.1 Portioning the Dataset

In general, it can be observed that the academic field agrees on the separation of the sample into a training set and a testing set. Sometimes an additional intermediate set, a validation set, is employed in order to avoid overfitting, or to determine the stopping point in the training process (Huang et al., 2004). However, it can be observed that the existing literature shows no particular consistency in the choice of training and testing set sizes. Yao et al. (1996) suggests allocating 70% of the collected data to the training set, 20% to the validation set and 10% to the testing set, while Lahmiri (2016), Dunis and Huang (2002) and Hann and Steurer (1996) employ a training set containing 80% of the total sample and a testing set containing the remaining 20%. In further terms of the selection of training sets, Zhang and Hu (1998) perform a comparative study using two different training sample sizes in an attempt to test if there is a significant difference between large and small training samples in forecasting exchange rates. They find that the large sample, consisting of 887 observations, outperforms the smaller sample of 261 data points.

The aggregated data in this study is divided into three sub-sets; a training set, a validation set and a testing set. Following the aforementioned suggestion by Yao et al. (1996), the training set is allocated 70% of the collected data, which is utilized for model selection purposes, whereas the validation set is allocated 20%, and the testing set 10%. The order of the data is preserved due to the presence of time-dependencies between observations, and is chronologically fed into the model to further avoid any bias when conducting the training. The dates for the data split are reported in *Table 1*.

Table 1. Dataset splits

	Start	**End**
Training	01.01.1999	27.10.2012
Validation	28.10.2012	08.10.2016
Testing	09.10.2016	30.09.2018

Since both the volatility inputs and the actual outputs of the training set are known beforehand when implementing the GRU neural network, the training process is said to be supervised. As mentioned previously, the GRU neural network is fed with volatility inputs and then trained, in order to generate volatility outputs. The resulting outputs are compared with the actual outputs to determine if the predictive power of the GRU neural network is satisfactory. While the GRU neural network is trained, the same dataset is processed several times in order to find the most appropriate set of connection weights, that is, a set of weights within the training set that allows for the minimum loss between the predicted outputs and the actual outputs. As such, the training process is interpreted as the equivalent of the minimization of the loss function.

To evaluate the GRU neural network fit on the training set, the validation set is used as a prevention tool against network overfitting. Within the validation set, the hyper parameters are fine-tuned to ensure that a decrease in the loss function over the training set yields a decrease in the loss function over the validation set as well, or else the network is over-fitted and further training is unnecessary. The validation process is unbiased as the volatility inputs contained in the validation set are only fed to the GRU neural network after the training process, thus avoiding any data memorization.

Once the GRU neural network goes through the complete training and validation process, it is exposed to the test set in order to evaluate the applicability of the network to unknown data. Hence, the testing set corresponds to the evaluation of the predictive performance of the GRU neural network onwards in time. The volatility outputs obtained through the test process will be used to compute VaR and ES predictions.

In conformity with the existing literature, all datasets used in the GRU neural network are normalized for the purpose of improving the performance of the network. Normalization essentially removes the dependence of measurement unit by transforming the data into a standardized range of values. In most cases, using unscaled data worsens the learning process, leading to an ineffective network

(Brownlee, 2016). RNNs are particularly exposed to the consequences of using unscaled data, as they are highly sensitive to exploding gradients when the magnitude of weight changes is significant between different time steps. A common technique for normalizing the data is the min-max scaler:

$$x' = \frac{x - min_x}{max_x - min_x} \qquad (22)$$

where x' is the normalized data point, min_x the minimum value of the data point x and max_x the maximum value of x. The min-max scaler normalizes the data into the range $[0,1]$.

4.2.2 Fixed Hyper Parameters of the GRU Neural Network

Among all parameters within the GRU neural network, the loss function, the optimizer, the performance measure, the batch size and the look back are kept unchanged. These fixed hyper parameters are presented below:

- The Mean Squared Error (MSE), defined in equation (16), is chosen as the loss function and measures by definition the average squared difference between the predicted outputs and the actual outputs.

- The optimizer mirrors the use of the gradient descent method, as it controls the magnitude of changes to the weights within the network, with regard to the loss gradient. The optimization algorithm Adam is chosen as the optimizer in the GRU neural network, because in contrast to the stochastic gradient descent, which maintains one single learning rate for all weight updates, Adam separately adapts a learning rate for each network weight, and has the attractive feature of being able to change learning rate during training (Brownlee, 2017).

- The Mean Absolute Error (MAE), defined in equation (15), is used in the GRU neural network as a performance measure to assess the model fit while training and validating the network. In other words, the MAE is the metric

used to assess the accuracy of the GRU neural network. It is possible to use the same metric for the loss function and the performance measure, yet these two parameters are specified differently in a large segment of the literature review using RNNs.

- The batch size defines the number of inputs that will be propagated in the GRU neural network during the training process, and in this case the batch size is set to 32, which is a conventional setting. As such, volatility inputs are fed in the network through numerous batches, each containing 32 inputs. After the propagation of a batch, the network is trained before receiving another batch of 32 inputs. This operation is repeated until the end of the training and validation processes, *i.e.* when all inputs are propagated. As the dataset used in this study is rather large, using a batch size is relevant, since it allows the GRU neural network to use less memory.

- The look back function specifies the number of time steps, *i.e.* the lagged inputs the RNN should use to forecast the desired outputs. As mentioned previously, RNNs have difficulties learning long-term time dependencies, hence the importance of introducing the GRU, which remedies the short-term memory problem of the RNNs. To enforce the ability of the RNN to find repeating temporal patterns, the look back function or *sliding window technique* (Frank et al., 2001) is used in combination with the GRU neural network. Applying a sliding window technique seems even more relevant, as Ben Taieb et al. (2011) discuss the advantages of using such a technique when forecasting a single output for each time step, which is the case in this study. However, the sliding window should not be too small or the time dependencies will not be well captured, nor too large to avoid feeding excessively the same inputs, as this could lead to overfitting problems (Frank et al., 2001). For all trials, the look back is set to 100 lagged inputs, which corresponds to a 3-month period in the data sample.

4.2.3 Fine-Tuning the Hyper Parameters

Following the description of the validation process, fine-tuning the hyper parameters appears to be critical in order to improve the accuracy of the GRU neural network. For this purpose, several trials are run on the training and validation sets, while modifying the settings of a few chosen hyper parameters, as presented below:

- An increasing number of neurons in the hidden layer can lead to an increase in accuracy of the GRU neural network. For this reason, the number of hidden neurons is set to 50 neurons and will then be increased to 100.

- The dropout function is a regularization method used to prevent overfitting by allowing the GRU neural network to drop a random set of neurons while training the network. Ignoring several neurons for each iteration during the training process is necessary, because if the network is fully connected, neurons will become interdependent, leading to overfitting of the training data. The dropout function is first set to 0.25, implying that 25% of the existing neurons within the network will be ignored during the training process. The dropout is later set to 0.3, in order to test if the accuracy of the GRU neural network increases when the dropout function is increased.

- A simple change of the activation function may as well increase the accuracy of the GRU neural network. As such, both the sigmoid and the ReLU activation functions will be used.

- The number of epochs can also influence the accuracy of a neural network. It refers to the number of times all the training and validation datasets are propagated through the GRU neural network. The standard procedure is to increase the number of epochs until the chosen metric – in this case the MAE – decreases for the validation set, while it continues to increase for the training set, *i.e.* when the training set shows signs of overfitting.

4.2.4 Choosing the GRU Architecture

Following the fine-tuning of the hyper parameters, the GRU neural network that performs best should be identified. The most accurate GRU neural network is the one, which best will minimize the loss function of the validation set. As the MAE is the selected performance measure, the most accurate GRU neural network is also the one that provides the lowest MAE value.

4.3 GARCH(1,1) Volatility Predictions

In order to estimate and forecast one-day-ahead volatilities from the GARCH(1,1) model, the daily return series is used as an input, which is specified in the conditional mean equation according to:

$$r_t = \mu + \eta_t \qquad (21)$$

where μ is a constant and η_t a random error term at time *t*. In the empirical estimation of the model, the Student t-distribution is utilized for the random errors in order to allow for a leptokurtic behavior.

To ensure congruence with the neural network implementation, the data set is divided into an in-sample period from *01.01.1999* to *08.10.2016*, while the remaining time period, *09.10.2016* to *30.09.2018*, is reserved for the out-of-sample period. The obtained forecasts in the out-of-sample period are static, meaning that the first forecast is made on the *09.10.2016*, and then additional forecasts are made on each consecutive day for the entire out-of-sample period.

4.4 Calculation of VaR and ES

As previously introduced, there is a trade-off between using non-parametric and parametric approaches predict VaR. Whereas the main drawback of parametric approaches is their reliance on a distributional assumption, non-parametric approaches may be too slow-paced to take actual market conditions into account. However, due to the inherent characteristics of RNNs, it would seem more reasonable to use a non-parametric approach when predicting VaR. Given that a RNN is a model-free, non-parametric approach, which considers the underlying loss distribution of a sample in contrast to a parametric approach, should be more appropriate. Indeed, as the attractiveness of a RNN lies in its model-free characteristic, this should be reflected as closely as possible in the method used to forecast VaR. For consistency purposes and to mimic financial institutions' practices as mentioned in 2.1.2, a non-parametric approach will also be used to compute VaR, based on volatility forecasts obtained from a GARCH(1,1).

In order to minimize the aforementioned drawback of non-parametric approaches, a Volatility Weighted Historical Simulation (VWHS) appears to be a better alternative to a Basic Historical Simulation (BHS). Hull and White (1998) conclude that using volatility forecasts for rescaling, in combination with a BHS, gives better results than the BHS itself to calculate VaR predictions. Prior to proceeding with a BHS on a dataset containing the losses of nine different currencies, the authors implement a so-called "volatility updating" based on volatility estimates obtained from the EWMA. The "volatility updating", in combination with a BHS, results in significant 1-percentile estimates of daily losses for all currencies.

According to the VWHS, the losses are rescaled as shown by equation (4) in 2.1.2. Volatility forecasts obtained through the implementation of either a GARCH(1,1) or the RNN are used to rescale the daily losses of the EUR/USD exchange rate.

Due to the implementation of the look back function within the GRU neural network, no estimates are provided for the first 100 days, nor the last day of the training and validation sets. Following the same argumentation, no forecasts are available for the first 100 days nor the last day of the test set. As all sets work independently of each other, the sliding window does not overlap between sets. Consequently, no inputs are available to estimate or forecast the first 100 days nor the last day in each set. Accounting for the missing days implies a redefinition of the in-sample and out-of-sample periods when using a GRU neural network, as shown in *Table 2*.

Table 2. Redefinition of in-sample and out-of-sample periods

In-sample	**Out-of-sample**
Training set: 11/04/1999 – 26/10/2012 Validation set: 05/02/2013 – 07/10/2016	Test set: 17/01/2017-29/09/2018

For consistency purposes when computing VaR and ES forecasts, the redefinition of the aforementioned periods is also applied to the volatility estimates and forecasts obtained with the GARCH(1,1) model.

According to the newly defined out-of-sample period, 621 rescaled loss series are computed for each implementation. A BHS is then applied to the loss series to obtain VaR forecasts for a 95%, 97,5% and 99% confidence level, using a sliding window of 6288 days. The number of days included in the sliding window corresponds to the number of days in the in-sample period, or in both the training and validation sets when referring to the GRU neural network. Based on the VaR forecasts, ES forecasts for a 95%, 97.5% and 99% confidence level are computed as shown by equation (3) in 2.1.1.

4.5 Backtesting Value-at-Risk and Expected Shortfall

As mentioned in 2.1.3, the Kupiec test is used to backtest the VaR forecasts obtained with the GARCH(1,1) and the GRU neural network. In addition to the one-sided test presented in 2.1.3, the Kupiec test can also be implemented with a confidence interval, *i.e.* a two-sided test, which is the backtesting method that will be used in this study. To apply the two-sided Kupiec test, the number of VaR forecasts, the confidence level under which they were computed, the number of exceptions, as well as the desired backtesting confidence level are necessary (Dowd, 2006). Under the null hypothesis, the backtested model is not rejected, implying that the number of exceptions lies between the lower and upper bounds of the confidence interval. It is customary to choose a 95% confidence level for the backtesting and to apply this level to different VaR modelling, independently of the chosen VaR confidence level. Indeed, the power of the Kupiec test when using higher backtesting confidence levels weakens, implying that the two-sided Kupiec test is likely not to reject an incorrect model. As such, a 95% confidence interval will be applied to backtest VaR forecasts at a 95%, 97,5% and 99% confidence level.

The second test presented by Acerbi and Szekely (2014) is implemented to backtest ES, and the test statistic is calculated as presented by equation (8) in 2.1.3. Following the same argumentation as for the choice of the backtesting confidence level for VaR, the critical value -0.70 at a 95% confidence level is used to backtest ES forecasts at a 95%, 97,5% and 99% confidence level.

5. Empirical Results

The subsequent chapter presents and discusses the main empirical findings of the stated methodology. Section 5.1 is devoted to the trial results of GRU neural network. Section 5.2 presents the in-sample and out-of-sample volatility predictions for both the GARCH(1,1) model and the GRU neural network. The final section presents the obtained VaR and ES predictions from the two models.

5.1 Trial Results of the GRU Neural Network

As mentioned in 4.2.3, different trials are used for the training and validation processes, in order to find the best performing GRU neural network architecture by fine-tuning the hyper parameters. The first trial, as defined in *Table 3*, is run with a considerably large number of epochs and will thus provide an interesting overview of the training and validation loss functions' behaviour.

Table 3. Setup Trial n° 1

Trial n°	Activation function	Hidden neurons	Epochs	Dropout
1	Sigmoid	50	1000	0.25

[INSERT FIGURE 7 ABOUT HERE]

Under the circumstances of the first trial, both training and validation loss functions are decreasing for a low number of epochs, as shown by *Figure 7* in Appendix B. However, signs of overfitting appear before the 200th epoch, as the validation loss function starts to increase, while the training loss function continues to decrease. At approximately the 500th epoch, overfitting is clearly identified since the validation loss function is without any doubt greater than the training loss function. As such, Trial n°2 and Trial n°3 are run with 500 epochs to disregard the utmost of the overfitting, and to confirm the intuition that the lowest loss on the validation set exists before the 200th epoch. Yet, as seen in *Table 4* and *Table 5*, the second and third trials are fine-tuned for different activation functions. In fact, the second and third trial use a sigmoid function and a reLU function respectively.

Table 4. Setup Trial n° 2

Trial n°	Activation function	Hidden neurons	Epochs	Dropout
2	Sigmoid	50	500	0.25

Table 5. Setup Trial n° 3

Trial n°	Activation function	Hidden neurons	Epochs	Dropout
3	reLU	50	500	0.25

[INSERT FIGURE 8 AND 9 ABOUT HERE]

Figure 8 and *Figure 9* in Appendix B corroborate the hypothesis that the solution to the minimization problem of the validation loss function resides between the 1st and the 200th epoch. Outside of this interval, the validation loss function increases, a behavior which can be graphically identified in both *Figure 8* and *Figure 9*, despite the different activation functions. In terms of performance measure, the lowest MAE value on the validation set is 0.0148 when using a sigmoid function, whereas it is 0.0127 with a reLU function. Accordingly, the reLU activation function performs better than the sigmoid function so far. Moreover, the sigmoid function identifies the lowest MAE value at the 98th, 194th and 200th epochs, while the reLU function does at the 132th epochs, implying that it is necessary to decrease the number of epochs in order to obtain the lowest MAE value. Thus, further trials defined in *Table 6* and *Table 7* are run with 200 epochs.

Table 6. Setup Trial n° 4

Trial n°	Activation function	Hidden neurons	Epochs	Dropout
4	sigmoid	50	200	0.25

Table 7. Setup Trial n° 5

Trial n°	Activation function	Hidden neurons	Epochs	Dropout
5	reLU	50	200	0.25

As presented in *Table 6* and *Table 7*, both Trial n°4 and Trial n°5 use the same settings for all hyper parameters, except the activation function. In order to investigate if the sigmoid function is truly outperformed by the reLU function, both activation functions are used once more. Depending on the outcome of these two trials, the reLU function will be chosen over the sigmoid function if it is still identified as the best performing activation function.

[INSERT FIGURE 10 AND 11 ABOUT HERE]

Figure 10 and *Figure 11* in Appendix B present plots of the loss functions corresponding to Trial n°4 and Trial n°5. With 200 epochs, the lowest MAE provided by the reLU function on the validation set is equal to 0.0125 at the 101^{th} and 121^{th} epoch. On the other hand, the lowest MAE obtained with the sigmoid activation function is 0.0140 at the 191^{th}, 193^{th} and 200^{th} epoch. Once again, the reLU function outperforms the sigmoid function, suggesting that only the reLU function should be used for the upcoming trials. Additionally, the number of epochs can still be decreased since the lowest MAE first appears at the 101^{th} epoch, as shown by the reLU function.

Following the conclusions drawn from previous trials, Trial n°6 and Trial n°7 use the reLU function and a decreased number of epochs, that is 150 epochs. However, the number of neurons in the hidden layer differs in each trial, in order to verify if the accuracy of the model increases, when the number of hidden neurons increases as well. The settings of Trial n°6 and Trial n°7 are presented below in *Table 8* and *Table 9*.

Table 8. Setup Trial n° 6

Trial n°	Activation function	Hidden neurons	Epochs	Dropout
6	reLU	50	150	0.25

Table 9. Setup Trial n° 7

Trial n°	Activation function	Hidden neurons	Epochs	Dropout
7	reLU	100	150	0.25

[INSERT FIGURE 12 AND 13 ABOUT HERE]

Figure 12 and *Figure 13* in Appendix B provide plots of the loss functions corresponding to Trial n°6 and Trial n°7 respectively. According to both plots, the training and validation loss function behave similarly, suggesting that the overfitting problem occurs at higher epochs. Under both trials, a decrease or an increase of the training loss function is in most instances reflected in the evolution of the validation loss function. Graphically in *Figure 12* and *Figure 13*, several interesting local minima of the training and validation loss functions are identified between the 100^{th} and 150^{th} epoch. Hence, the lowest MAE provided by Trial n°6 and Trial n°7 should be located between this interval of epochs. As expected, the lowest MAE under Trial n°6 occurs at the 146^{th} epoch and is equal to 0.0134. The lowest MAE found under Trial n°7, which is equal to 0.0120 and occurs at the 105^{th} epoch, meets the graphical expectations as well. Considering the magnitude of each given MAE, the hyper parameter settings of Trial n°7 appear to be better as they allow for the lowest MAE value out of both trials. Consistently with the findings of Trial n°7, the number of epochs should be lowered to 105 for the following trials.

As previously introduced in 4.2.3, different settings of the dropout parameter may increase the performance of the RNN. Thus, Trial n°8 and Trial n°9 are set for a number of epochs of 105 and a dropout of 0.25 and 0.3 respectively, while keeping other parameters fixed in accordance with the settings of trial n° 7.

Table 10. Setup Trial n° 8

Trial n°	Activation function	Hidden neurons	Epochs	Dropout
8	reLU	100	105	0.25

Table 11. Setup Trial n° 9

Trial n°	Activation function	Hidden neurons	Epochs	Dropout
9	reLU	100	105	0.3

[INSERT FIGURE 14 AND 15 ABOUT HERE]

Figure 14 and *Figure 15* in Appendix B present plots of the loss functions corresponding to Trial n°8 and Trial n°9. Under Trial n°8, the lowest MAE is identified at the 100^{th} epoch and is equal to 0.0129, whereas the lowest MAE under Trial n°9 is identified at the 103th epoch and is equal to 0.0127. A change in the dropout parameter does not seem to significantly improve the results, as the lowest MAE values provided by both trials only differ by 0.0002, and the difference between the necessary number of epochs is very small. Choosing the settings of Trial n°9 should be preferred as they result in the lowest MAE value and the cost of using a higher number of epochs is minimal in this case, *i.e.* only 3 epochs separate Trial n°8 from Trial n°9. Consequently, the GRU neural network will be run on the test set using a fine-tuning of hyper parameters similar to that of Trial n°9, as shown in *Table 12*.

Table 12. Final GRU Neural Network

Activation function	Hidden neurons	Epochs	Dropout
reLU	100	105	0.3

5.2 GARCH(1,1) and GRU Neural Network Volatility Predictions

Table 13 in Appendix C displays the in-sample estimation results of the GARCH(1,1) benchmark model with a Student t-distribution, covering a time period from *01.01.1999* to *08.10.2016*. As demonstrated by the table, the estimated parameters alpha and beta are significant at a 1 % level.

[INSERT TABLE 13 ABOUT HERE]

In terms of forecasting accuracy, the one-step-ahead volatility predictions from both models are evaluated based on the MSE and MAE metrics. As such, the predictions from the two models are compared with the realized volatility proxy across the divided data sample. In *Table 14*, the performance metrics for GARCH(1,1) are reported for the entire in-sample as well as for the out-of-sample period, whereas in *Table 15*, the metrics for the training, validation and test sets are displayed in terms of the GRU neural network. Furthermore, all metrics are calculated over the newly defined samples, following the date modifications required by the look back function, as explained in section 4.4. According to *Table 14*, both the MSE and MAE values in the out-of-sample period are lower than the corresponding figures in the in-sample period for the GARCH(1,1) model, suggesting that the produced forecasts fit the actual realized volatility rather well. To visualize the GARCH(1,1) model's accuracy, the volatility estimates and predictions are plotted together with the ex-post realized variance realizations in *Figure 16* (*cf.* Appendix C). The apparent non-continuity in the GARCH(1,1) volatility series is consistent with the new date definition. It can be inferred that the GARCH(1,1) model seems to be underestimating volatility during the entire window frame, particularly when there are pronounced spikes, such as during the dot-com bubble in 2000 and the global financial crisis in 2008.

Table 14. Average MSE and MAE values for volatility forecasting using GARCH(1,1)

Model	In-sample $\overline{\text{MSE}}$	Out-of-sample $\overline{\text{MSE}}$	In-sample $\overline{\text{MAE}}$	Out-of-sample $\overline{\text{MAE}}$
GARCH(1,1)	1.977E^{-09}	4.286E^{-10}	2.786E^{-05}	1.445E^{-05}

Table 15. Average MSE and MAE values for volatility forecasting using GRU

Model	Training $\overline{\text{MSE}}$	Training $\overline{\text{MAE}}$	Validation $\overline{\text{MSE}}$	Validation $\overline{\text{MAE}}$	Test $\overline{\text{MSE}}$	Test $\overline{\text{MAE}}$
GRU-RNN	1.072E^{-09}	1.556E^{-05}	1.950E^{-09}	2.414E^{-05}	2.880E^{-10}	1.056E^{-05}

[INSERT FIGURE 16 ABOUT HERE]

The results in *Table 15* suggest, in similarity with the findings of the GARCH(1,1) model, that the volatility predictions based on the test set are more satisfactory than the ones from the training and validation sets, as both the MSE and MAE metrics attain their lowest values for the test set. This empirical outcome is in disparity with the findings of the majority of previous neural network applications. Nevertheless, it is considered promising, since it may imply that the learning process of the GRU neural network has been very effective.

Figure 17 (*cf.* Appendix C) compares the volatility forecasts obtained from the GRU neural network with the ex-post realized volatility. It can be observed that the GRU, although to a somewhat lower degree than the GARCH(1,1) model, fails to accurately estimate and predict the volatility spikes present in the whole data sample. Notwithstanding, it should be emphasized that both models seem to display an adequate overall forecasting ability when taking the low magnitude of the MSE and MAE metrics into account, and when graphically analyzing *Figure 18* (*cf.* Appendix C), in which volatility forecasts for both models are plotted against the realized volatility.

[INSERT FIGURE 17 AND 18 ABOUT HERE]

5.3 Value-at-Risk and Expected Shortfall Predictions

Following the volatility predictions obtained from the two models, VaR and ES forecasts of the EUR/USD exchange rate are computed, for which a summary is displayed in *Table 16* and *Table 17*. The reported results are based on a rolling window approach with a fixed window length of 6288 days. Hence, the VaR and ES models are re-estimated every day.

Table 16. Summary of VaR results

Model	$\overline{VaR}$			Number of VaR Violations			Confidence interval 95%		
	95%	97.5%	99%	95%	97.5%	99%	95%	97.5%	99%
VWHS with GARCH(1,1)	0.066	0.083	0.100	23	7	2	[21,42]	[8,24]	[2,11]
VWHS with GRU-RNN	0.078	0.100	0.124	14	7	4			

Table 17. Summary of ES results

Model	$\overline{ES}$			ES Test Statistic		
	95%	97.5%	99%	95%	97.5%	99%
VWHS with GARCH(1,1)	0.087	0.099	0.113	0.291	0.519	0.588
VWHS with GRU-RNN	0.109	0.127	0.152	0.547	0.514	0.346

The VaR forecasts are obtained with three different confidence levels; 95%, 97.5% and 99%, and then evaluated with a two-sided Kupiec test. When comparing the observed frequency of VaR violations in the out-of-sample period with a 95% confidence interval, the underlying VaR model based on GARCH(1,1) cannot be statistically rejected. In contrast, the VaR model based on the volatility predictions from the GRU is rejected, since the number of VaR exceedances fall outside the confidence interval. When considering a confidence level of 97.5%, the models produce an equal number of VaR violations, *i.e.* 7, resulting in both models being statistically rejected. For the 99% confidence level, the performance of the GRU

dominates that of the GARCH(1,1). However, albeit the VaR model based on GARCH(1,1) is not rejected on this level, the non-rejection occurs precisely at the lower bound of the interval, meaning that the GARCH(1,1) model is not indisputably non-rejected. To conclude, it can be inferred that the results for the VaR models are rather inconclusive across the different confidence levels. Hence, there is no clear-cut conclusion as to which of the two models provides the most accurate and reliable VaR forecasts of the EUR/USD exchange rate. However, despite the fact that a 99% VaR is required by the Basel II regulation, a 95% VaR is more relevant in this case, because models based on a 99% level are in general subject to a low power when applied on sample periods longer than one year. This means that, with a 99% confidence level, the probability of rejecting an incorrect model is very low.

Furthermore, it can be seen that when decreasing the confidence level, the forecasting performance of the GARCH(1,1) model improves while it decreases for the GRU neural network. This finding, in conjunction with the fact that the power of the test is higher for the 95% level than for the 99%, indicates that the GRU neural network is deemed rather unsatisfactory for forecasting VaR compared to the GARCH(1,1).

When the focus is switched towards ES, the test statistic for the underlying ES model based on the GARCH(1,1) with 95% confidence level is 0.291. Contrasting this test statistic to the critical value of -0.70, as proposed by Acerbi and Szeleky (2014), it is concluded that the null hypothesis is not rejected for a confidence level of 95%, suggesting that no statistical underestimation of the exchange rate risk can be concluded. In fact, neither of the two models is rejected at *any* confidence level for ES, since the test statistic is larger than the critical value for all levels. When further evaluating the test statistic values, the ANN performs better than the GARCH(1,1) at the 95% level, while the performance of the two models is fairly similar at the 97.5% level. Nonetheless, GARCH(1,1) appears to be the better performing model at the 99% confidence level. According to the overall findings, both the GARCH(1,1) and the GRU neural network are suitable models in forecasting ES for all confidence levels.

6. Conclusion

The rationale behind this study was to examine and compare the performances of the GRU neural network and the conventional statistical model GARCH(1,1) when predicting VaR and ES, using EUR/USD exchange rate volatility forecasts.

The results suggest that the forecasting performance of neither the GRU neural network nor the GARCH(1,1) model could be clearly identified as superior to provide accurate predictions of both VaR and ES. Analyzed independently, the GARCH(1,1) model outperforms the GRU neural network when providing $VaR_{95\%}$ and $ES_{99\%}$ forecasts, whereas the GRU neural network shows better forecasting abilities when computing $VaR_{99\%}$ and $ES_{95\%}$ predictions. The intermediate confidence level of 97,5%, does not permit any differentiation between both models with regards to VaR predictions. A comparable conclusion is made for $ES_{97.5\%}$ forecasts, as the GARCH(1,1) model outperforms the GRU neural network to a very small extent. However, it is important to highlight that both approaches are not rejected at the chosen confidence levels. As such, the GRU neural network cannot be deemed very satisfactory for predicting VaR. The most pronounced difference between the two approaches can be found at the 95% level, where the GRU neural network is clearly rejected, while the GARCH(1,1) model is not. Similarly, the GRU neural network is more relevant to forecast ES at a 95% confidence level, as it appears to be the better performing model when it is weighted against the GARCH(1,1) model.

Nevertheless, there are a few areas where further research is welcomed for the improvement of this study. First, it could be fruitful to define the look back as a hyper parameter to fine-tune, based on trial and error. Whilst a greater look back value incorporates more memory, it does not retain the entirety of the sample period when producing out-of-sample forecasts. As such, it seems prudent to test different settings in order to achieve the best possible neural network architecture. Additionally, forecasting realized higher moments of the exchange rate, such as skewness and kurtosis, are considered potential extensions that this body of research could benefit from. Moreover, since the presented empirical findings may partially be contingent upon the choice of the EUR/USD exchange rate, additional currency pairs should be evaluated in order to increase the robustness and the generalizability of the study.

Moreover, there is reason to believe that other determinants, such as the interest rate differential between the EUR/USD, affects the evolution of the exchange rate. Hence, it would be of relevance to use it as a complementary feature to the realized volatility for providing VaR and ES predictions.

Bibliography

Acerbi, C., Nordio, C., Sirtori, C. (2001). Expected shortfall as a tool for financial risk References

Acerbi, C., Nordio, C., Sirtori, C. (2001). Expected shortfall as a tool for financial risk management, working paper.

Acerbi, C., Székely, B. (2014). Back-testing expected shortfall, *Risk Magazine*.

Artzner, P., F. Delbaen, J. M. Eber, and D. Heath (1997). Thinking Coherently, *Risk*, **10** (11), 68–71.

Artzner, P., F. Delbaen, J. M. Eber, and D. Heath (1999). Coherent Measures of Risk, *Mathematical Finance*, **9** (3), 203–228.

Bank for International Settlements (2013). Fundamental Review of the Trading Book: A Revised Market Risk Framework, consultative document, Basel Committee on Banking Supervision.

Ben Taieba, S., Bontempia, G., Atiyac A. and Sorjamaab, A. A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition, *Expert Systems with Applications*, 39.

Bishop, C. (2006). Pattern Recognition and Machine Learning. Springer Science, 240-246.

Bollerslev, T. (1986). Generalized Autoregressive Conditional Heteroskedasticity, *Journal of Econometrics*, **31**, 307–327.

Bredin, D. and Hyde, S. (2002). FOREX Risk: Measurement and Evaluation Using Value-at-Risk, *Journal of Business Finance & Accounting*, vol. 31, issue 9-10, 1389-1417.

Brooks, C. (1996). Testing for non-linearity in daily sterling exchange rates, *Applied Finance Economics*, **6**, 307–317.

Brownlee, J., (2016). Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras, *Deep Learning for time series*.

Brownlee, J., (2017). Gentle Introduction to the Adam Optimization Algorithm for Deep Learning, *Better Deep Learning*.

Connor, J., and L. Atlas. (1993). Recurrent neural networks and time series prediction, *Proceedings of the International Joint Conference on Neural Networks*, **I**, 301-306.

Chen, X., Lai, K and Yen, J. A Statistical Neural Network Approach for Value-at-Risk Analysis. *International Joint Conference on Computational Sciences and optimization,* working paper.

Chung, J., Gulcehre, C., Cho, K., Bengio, Y., (2015). Gated Feedback Recurrent Neural Networks, working paper.

Cybenko, G. (1989). Approximations by superpositions of a sigmoidal function, Mathematics of Control, *Signals and Systems*, **2**, 183-192.

Danielsson J., Hartmann P., and de Vries C. G. (1998). The Cost of Conservatism: Extreme Value Returns, Value-at-Risk, and the Basle 'Multiplicator Factor', *Risk*, **11** (1), 101-103.

Degiannakis, S. and Potamia, A. (2016). Multiple-days-ahead value-at-risk and expected shortfall forecasting for stock indices, commodities and exchange rates: Inter-day versus intra-day data. *International Review of Financial Analysis*, 2017, vol. 49, issue C, 176-190.

Dowd, K. (2006). Retrospective Assessment of Value-at-Risk, *Risk Management: A Modern Perspective*, 183 – 202, Elsevier.

Dunis, C. and Huang, X. (2002). Forecasting and Trading Currency Volatility: An Application of Recurrent Neural Regression and Model Combination, *Journal of Forecasting*, **21**, 317-354.

Emmer, S., Kratz, M. and Tasche, D. (2013). What is the best risk measure in practice? A comparison of standard measures, working paper.

Fama, E., (1965). The Behavior of Stock Market Prices, *Journal of Business*, **38**, 34-105.

Frank, R., Davey, N., and Hunt, S.P, (2001). Time Series Prediction and Neural Networks, *Journal of Intelligent and Robotic Systems*, **31**, 91-103.

Gately, E. (1996). Neural Networks for Financial Forecasting. 1st edition. New York: John Wiley & Sons, Inc.

Glosten, L., Jagannathan, R. and Runkle, D. (1993). On the relation between expected value and the volatility of the nominal excess return on stocks. *Journal of Finance*, **48**, 1779-1801.

Hann, T., and Steurer, E., (1996). Much ado about nothing? Exchange rate forecasting: Neural networks versus linear models using monthly and weekly data, *Neurocomputing*, **10**, 323-339.

Hansen, P. R., and Lunde, A. (2004). An Unbiased Measure of Realized Variance, working paper.

Hansen, P. R., and Lunde, A. (2005). A Forecast Comparison of Volatility Models: Does Anything Beat a GARCH(1,1)?, *Journal of Applied Econometrics*, **20**, 873-889.

Haykin, S. (2009). Neural Networks and Learning Machines. 3d edition. McMaster University, Ontario Canada.

He, K., Ji, L., Tso, G., Zhu, B., Zou, Y., (2018). Forecasting Exchange Rate Value at Risk using Deep Belief Network Ensemble based Approach, *Elsevier B.V.*, vol. 139, 25-32.

Hornik, K., Stinchcombe, M, and White, H. (1989). Multilayer feedforward networks are universal approximators, *Neural Networks*, **2**, 359-366.

Hsieh, D. A. (1989). Testing for nonlinear dependence in daily foreign exchange rates, *Journal of Business*, **62**, 329–368.

Huang, W., Wang, S., and Lai, K., (2004). Forecasting Foreign Exchange Rates With Artificial Neural Networks: A Review, *International Journal of Information Technology and Decision Making*, Vol. 3, No. 1, 145-165.

Hull, J.C., and White, A. (1998). Incorporating volatility updating into the historical simulation method for value-at-risk, *Journal of Risk,* **1**(1), 5-19.

Hull, J.C. (2006). Risk Management and Financial Institutions, 1st edition, Prentice Hall.

Jorion, P. (1997). Value at Risk: The New Benchmark for Controlling Market Risk, Irwin, Chicago.

Kuan, C.M., Liu, T. (1995). Forecasting Exchange Rates Using Feedforward and Recurrent Neural Networks, *Journal of Applied Econometrics*, Vol. 10, No. 4 (Oct. - Dec., 1995), 347-364.

Kumar, R. and Pradhan, R (2010). Forecasting Exchange Rate in India: An Application of Artificial Neural Network Model, *Journal of Mathematics Research*, Vol. 2, No. 4 (Nov, 2010), 111-117.

Kupiec, P.H. (1995). Techniques for Verifying the Accuracy of Risk Measurement Models, *Journal of Derivatives*, **3**(2), 73-84.

Lahmiri, S. (2016). Modeling and predicting historical volatility in exchange rate markets, *Physica A: Statistical Mechanics and its Applications*, Elsevier, Vol.471, 387-395.

Liu, H.C., Hung, J.C. (2010). Forecasting volatility and capturing downside risk of the Taiwanese futures markets under the financial tsunami, *Managerial Finance*, Vol. 36, Issue: 10, 860-875.

Locarek-Junge, H. and Prinzler, R. (1999). Using ANN to Estimate VaR, working paper.

Mandelbrot, B.B., (1963). The Variation of Certain Speculative Prices, *Journal of Business*, **36**, 394-419.

Meese, R. A., & Rogoff, K. (1983). Empirical exchange rate models of the seventies: Do they fit out of sample?, *Journal of international Economics*, **14**, 3–24.

Musah A.,, Du, J., Khan, H and Abdul-Rasheed Akeji, A., (2018). The Asymptotic Decision Scenarios of an Emerging Stock Exchange Market: Extreme Value Theory and Artificial Neural Network, Risks, **6** (132).

Nelson, D.B., (1991). Conditional heteroskedasticity in asset returns: A new approach, *Econometrica*, **59**, 347-370.

Panda, C., Narasimhan, V. (2007). Forecasting Exchange Rate Better with Artificial Neural Network, *Journal of Policy Modeling*, **29** (2), 227–236.

Poon, S. and Granger, C. (2003). Forecasting Volatility in Financial Markets: A Review, *Journal of Economic Literature*, **41**, 478-539.

Rootzén, H., and Klüppelberg C. (1999). A Single Number Can't Hedge against Economic Catastrophes, *Ambio*, **26** (6), 550–555.

Skymind (2019). A Beginner's Guide to LSTM's and Recurrent Neural Networks, *A.I. Wiki*

Sun, W., Rachev, S., Chen, Y and Fabozzi, F., (2008). Measuring Intra-Day Market Risk: A Neural Network Approach.

Taylor, S. (1986). Modelling financial time series, John Wiley & Sons, Chichester.

Tenti, P. (1996). Forecasting foreign exchange rates using recurrent neural networks. *Taylor & Francis*.

Yamai, Y. and Yoshiba, T. (2002). On the Validity of Value-at-Risk: Comparative Analyses with Expected Shortfall, *Monetary and Economic Studies*, **20**, 57-85.

Yao, J., Poh, H. and Jasic, L., (1996). Foreign exchange rates forecasting with neural networks, *Proceedings of the International Conference on Neural Information Processing,* 754-759.

Zakoian, J.M. (1991). Threshold Heteroskedastic Models, *Journal of Economic Dynamics and Control*, **18**, 931-955, North Holland.

Zhang, G., Patuwo, E.P. and Hu, M.Y. (1998). Forecasting with artificial neural networks: the state of the art, *International Journal of Forecasting*, **14**, 35-62.

Zhang, G., and Hu, M., (1998). Neural network forecasting of the British Pound/US Dollar exchange rate, *Journal of Management Science*, **26**, 495-506.

Appendix A

Figure 5. Return series of the EUR/USD exchange rate from 01.01.1999 to 30.09.2018

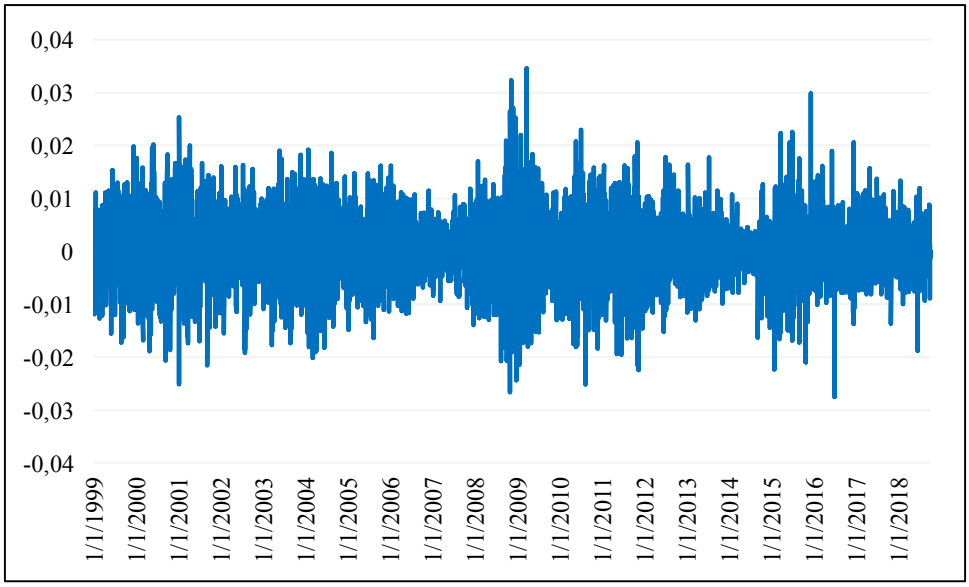
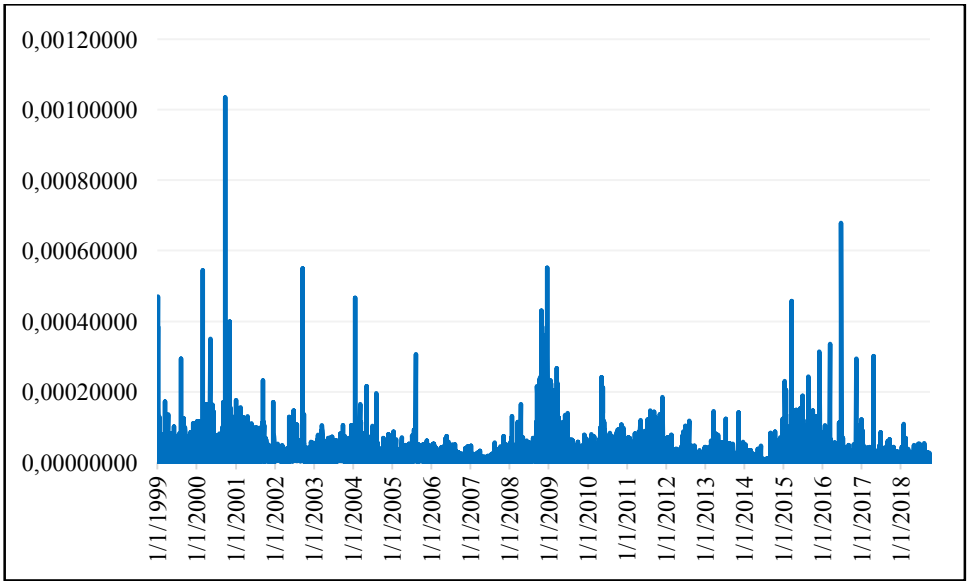


Figure 6. Realized Variance series



Appendix B

For all graphs shown in Appendix B, number of epochs is represented by the horizontal axis (X-axis), while the loss functions are represented by the vertical axis (Y-axis).

Figure 7. Training and Validation loss functions under Trial n°1

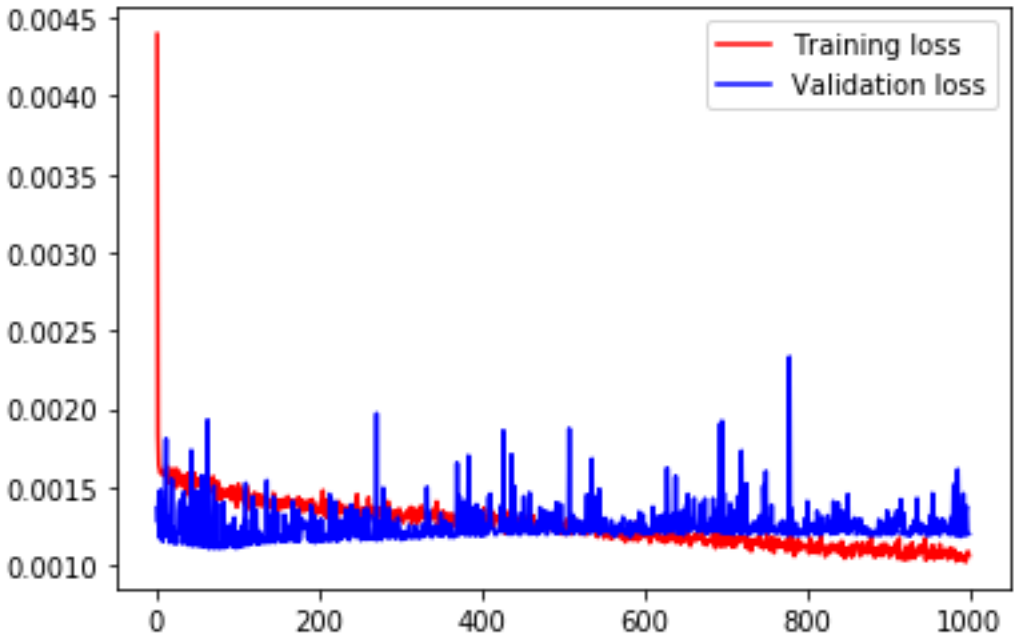


Figure 8. Training and Validation loss functions under Trial n°2

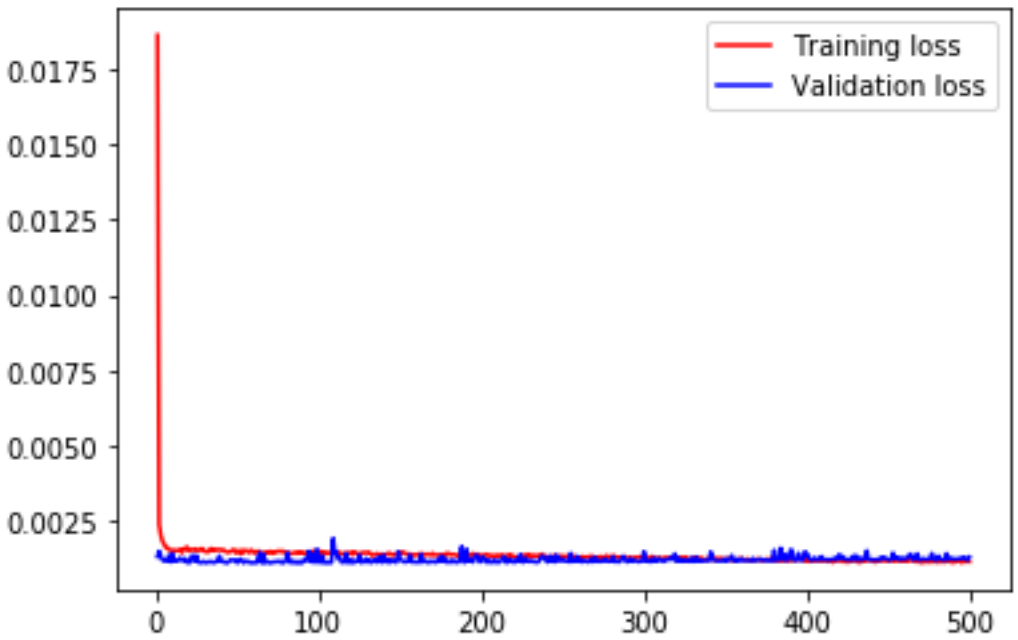


Figure 9. Training and Validation loss functions under Trial n°3

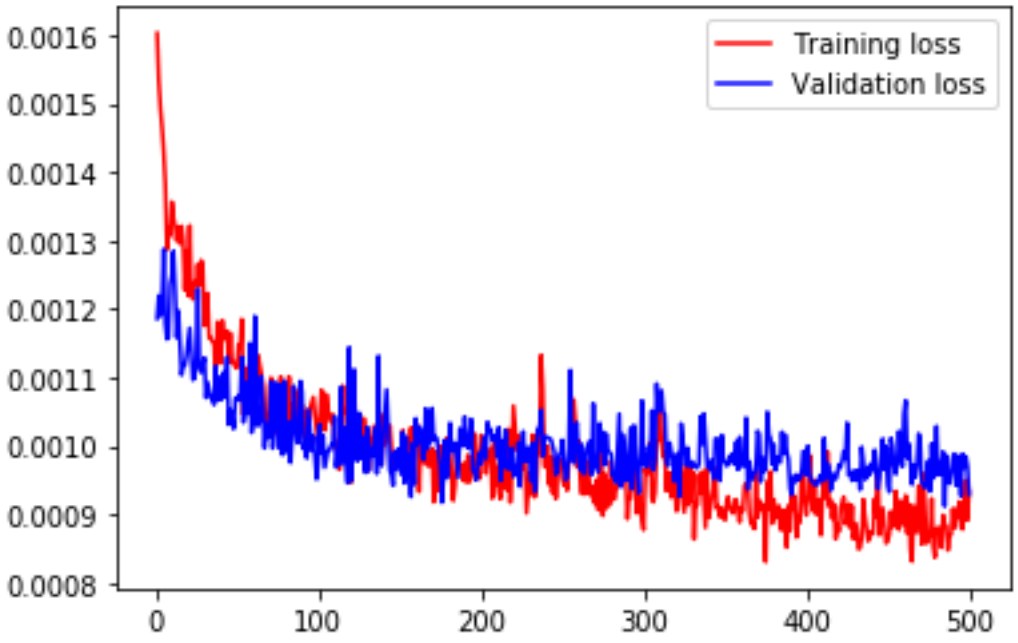


Figure 10. Training and Validation loss functions under Trial n°4

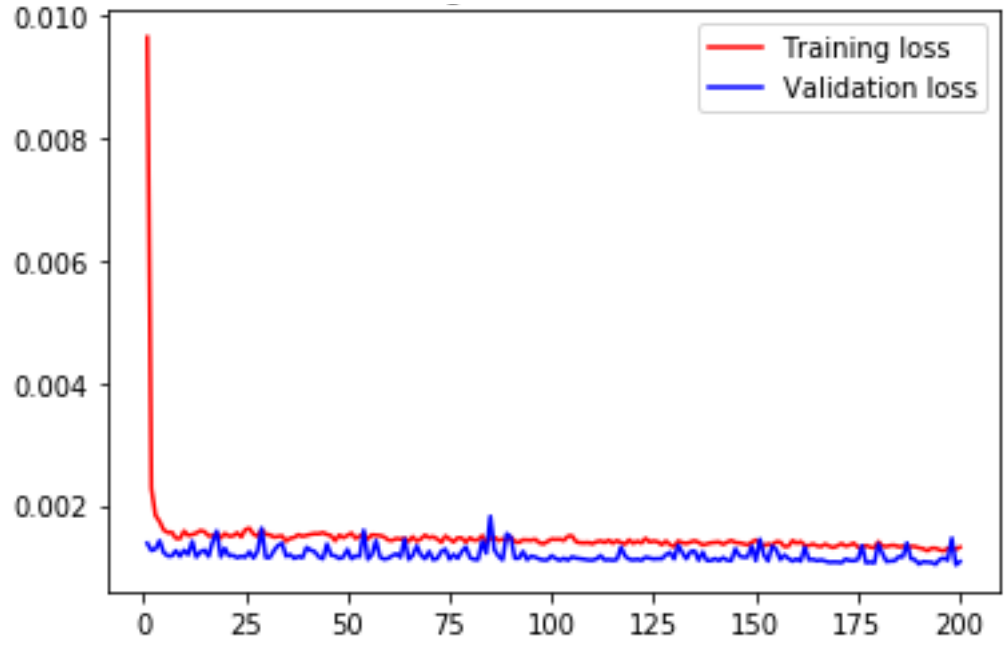


Figure 11. Training and Validation loss functions under Trial n°5

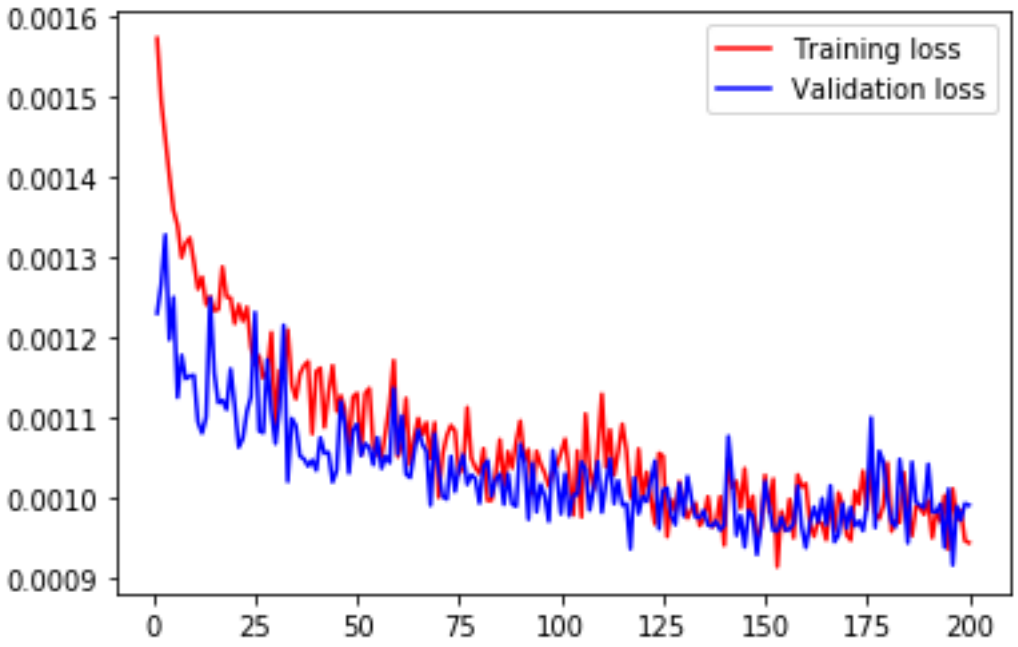


Figure 12. Training and Validation loss functions under Trial n°6

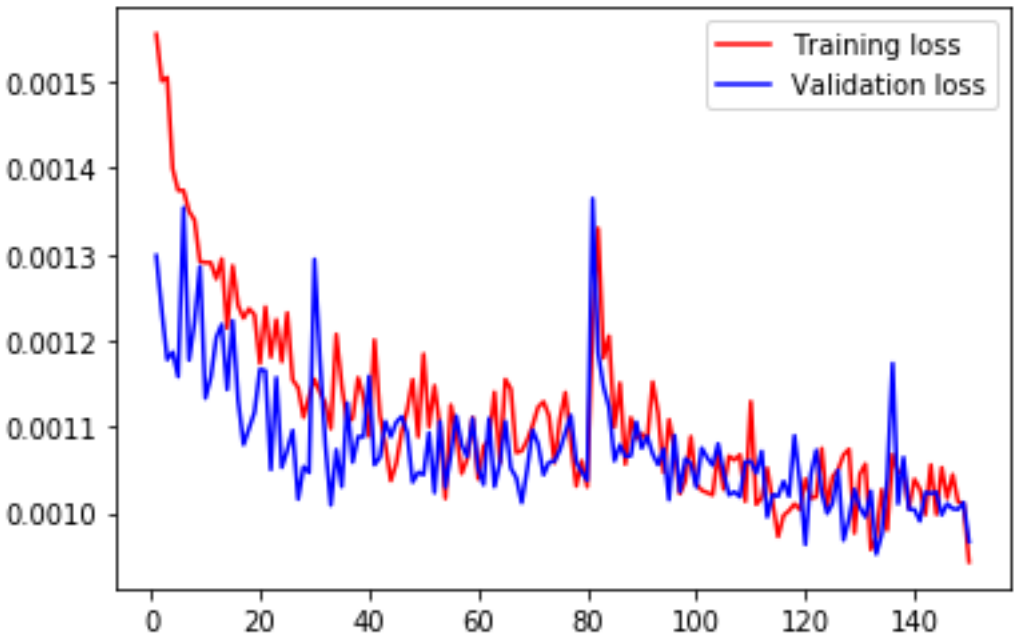


Figure 13. Training and Validation loss functions under Trial n°7

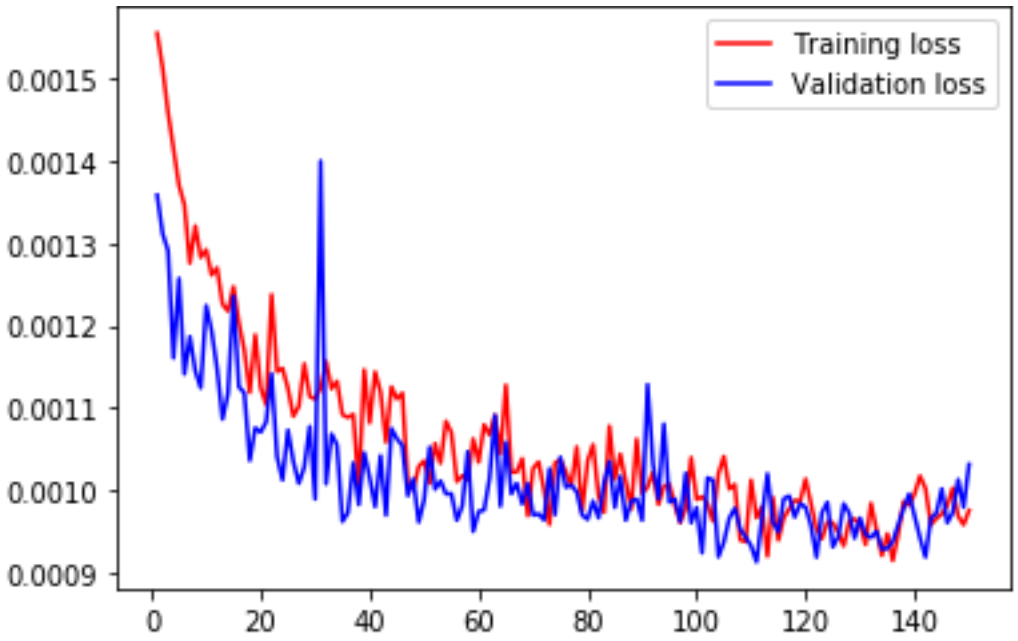


Figure 14. Training and Validation loss functions under Trial n°8

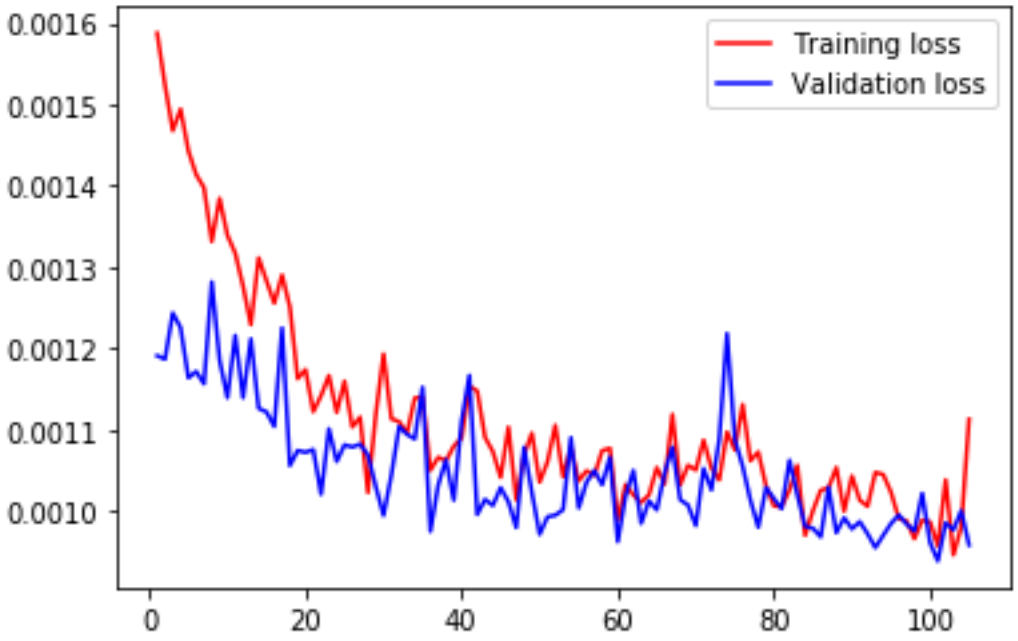
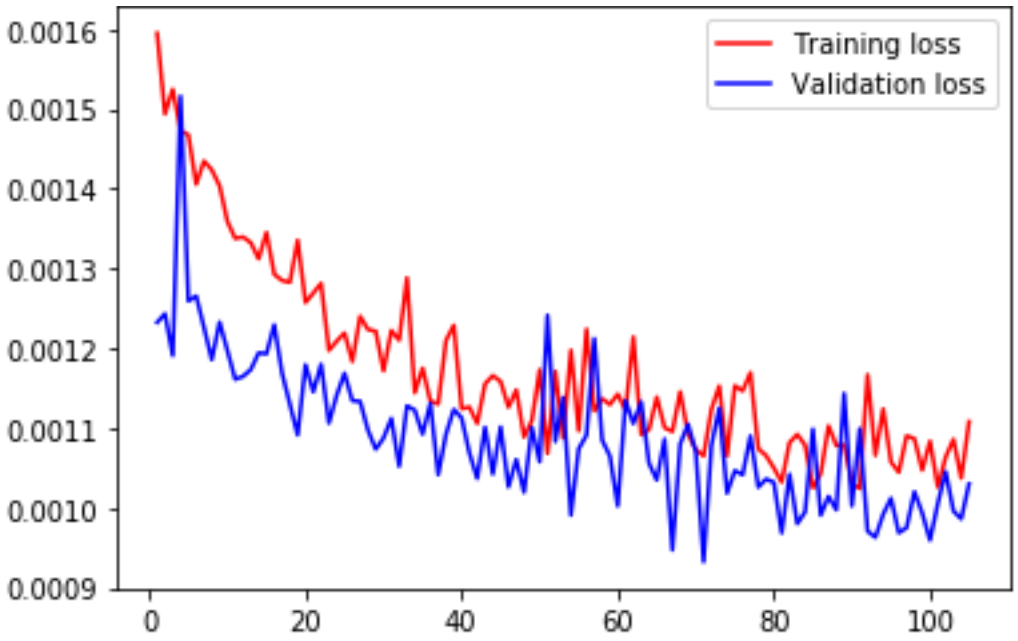


Figure 15. Training and Validation loss functions under Trial n°9



Appendix C

Table 13. In-sample estimation results of the GARCH(1,1)

Dependent Variable: DAILY_RETURNS
Method: ML ARCH - Student's t distribution (BFGS / Marquardt steps)
Date: 05/29/19 Time: 12:58
Sample: 1/01/1999 8/10/2016
Included observations: 6431
Convergence achieved after 111 iterations
Coefficient covariance computed using outer product of gradients
Presample variance: backcast (parameter = 0.7)
GARCH = C(2) + C(3)*RESID(-1)^2 + C(4)*GARCH(-1)

Variable	Coefficient	Std. Error	z-Statistic	Prob.
C	7.97E-06	4.85E-05	0.164407	0.8694
Variance Equation				
C	2.90E-08	2.91E-08	0.994534	0.3200
RESID(-1)^2	0.036822	0.007220	5.099927	0.0000
GARCH(-1)	0.976893	0.002965	329.4837	0.0000
T-DIST. DOF	2.669845	0.149405	17.86979	0.0000

R-squared	-0.000009	Mean dependent var	-8.15E-06
Adjusted R-squared	-0.000009	S.D. dependent var	0.005505
S.E. of regression	0.005505	Akaike info criterion	-7.804299
Sum squared resid	0.194843	Schwarz criterion	-7.799036
Log likelihood	25099.72	Hannan-Quinn criter.	-7.802478
Durbin-Watson stat	2.075851		

Figure 16. Realized Variance and GARCH(1,1) Variance

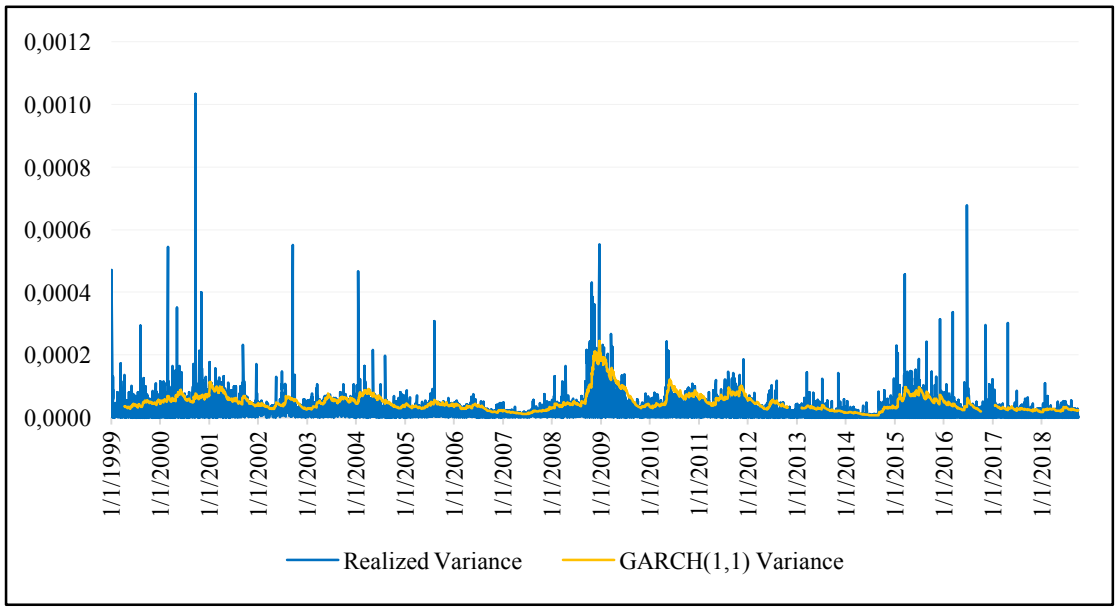


Figure 17. Realized Variance and GRU Neural Network Variance

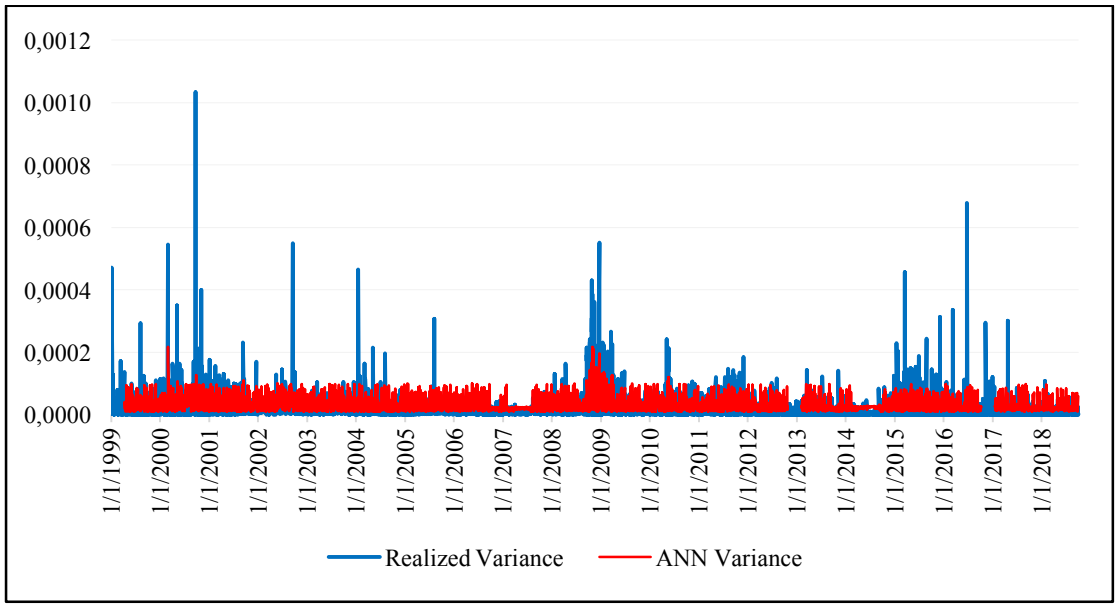


Figure 18. Realized Variance, GARCH(1,1) Variance and GRU Neural Network Variance