Original Article

Currency trading in volatile markets: Did neural networks outperform for the EUR/USD during the financial crisis 2007–2009?

Received (in revised form): 13th September 2010

Christian L. Dunis
is an Emeritus Professor of Banking and Finance at Liverpool John Moores University.

Jason Laws
is a Senior Lecturer at University of Liverpool Management School.

Ulrike Schilling
has completed an MSc course at Liverpool John Moores University.

Correspondence: Jason Laws, University of Liverpool Management School, University of Liverpool, UK
E-mail: j.laws@liv.ac.uk

ABSTRACT **The motivation for this article is to check whether neural network models have remained a superior method for forecasting the EUR/USD exchange rate during the financial crisis of 2007–2009. Alternative neural network architectures (Multi-Layer Perceptron (MLP), Recurrent Neural Network and Higher Order Neural Network (HONN)) are benchmarked against a random walk and a traditional ARMA model, and evaluated in terms of statistical accuracy and through a trading simulation on daily data over the period from January 2000 to February 2009, the period from August 2007 to February 2009, providing the out-of-sample testing period. Transaction costs and a confirmation filter devised to reduce false signals and thus also reduce losses and transaction costs were also taken into consideration. It is shown that the HONN structure gives the overall best results on a simple trading simulation; however, for an advanced trading simulation with a confirmation filter, the MLP outperforms all other models on most performance measures. On the whole, the results show that neural networks are still able to produce forecasts that yield a positive return and are superior to those of linear and more traditional models, with respect to both trading performance and statistical accuracy, even under very volatile market conditions.**

Journal of Derivatives & Hedge Funds (2012) **18,** 2–41. doi:10.1057/jdhf.2011.31

Keywords: confirmation filters; higher order neural networks; recurrent networks; Multi-Layer Perceptron networks; quantitative trading strategies

INTRODUCTION

The financial crisis experienced by economies worldwide in 2007–2009 has repeatedly been described as the most severe since the Great Depression by leading economists (Reuters, 2009). Since its beginning, it has led to the collapse of some of the most powerful banks, as well as the bankruptcy or bailout of numerous other companies in financial and non-financial industries. Although the first obvious effects on the real economy became apparent to the broad public only in the third quarter of 2008, the seizure in interbank markets actually started as far back as August 2007 (*The Economist*, 2009). Induced by a variety of interconnected factors, the most important of which may be the subprime mortgage crisis in the US and the usage of complex financial instruments in combination with poor risk analysis, uncertainty in financial markets has subsequently increased tremendously (IMF, 2007).

Extraordinary volatility levels have been shown by risk indicators like the TED spread,[1] a measure for perceived credit risk in interbank lending, or the CBOE Volatility Index, reflecting implied volatility of the S&P500, when both series depicted a significant increase in volatility in the second half of 2007, before peaking in record highs during late 2008.

Turmoil was observed in all kinds of financial markets including bond markets, equity and commodity markets, as well as the foreign exchange (FX) market: whereas EUR/USD volatility had been on average 9.33 per cent annualised over January 2000 through July 2007, our in-sample period, it jumped to 12.96 per cent from August 2007 to February 2009, an almost 40 per cent increase, and reached 19.20 per cent on average from September 2008 to February 2009!

The FX market is known to be one of the largest and most liquid in the world, reaching an average daily trading volume of US$3.2 trillion, of which the EUR/USD currency pair accounts for $840 billion, equivalent to a 27 per cent share (BIS, 2007). Because of its substantial turnover, which in turn is the reason for its importance for market participants like fund managers or corporate treasurers, there have been numerous attempts to improve the forecasting performance of the EUR/USD exchange rate in the past. However, with the lack of an unquestionable theoretical concept modelling the factors determining exchange rate developments, researchers and professionals have frequently been struggling to exceed the out-of-sample forecasting performance of a simple random walk model.

In an unstable environment like this, the task of forecasting and trading in financial markets becomes an even more challenging one. Yet, within the last years, several studies on forecasting and trading the EUR/USD rate and other currency pairs have been conducted, employing simple and traditional, as well as more advanced methods. Among these, Neural Network Regression (NNR) models have been a relatively new methodology with a design inspired by the structure of the human brain.

In recent years, the model class of Neural Networks has gained popularity and importance in many fields of theoretical research and practical application. Lisboa and Vellido (2000) mention several business operations where Neural Networks have been utilised to allocate resources more efficiently, to analyse, organise and visualise highly complex data and therefore help increase planning accuracy of processes, which simpler approaches have failed to model effectively.

With the development towards a knowledge-driven business environment, where information can be gathered and stored more easily, extracting structures and patterns to make use of this information becomes ever so important. Successful business application examples can be found, for instance, in the areas of flight scheduling, fraud detection, demand forecasting, bankruptcy prediction and decision making. But also in financial engineering, the use of Neural Networks has increased because of their extraordinary mapping capabilities. Previous theoretical works and the practical use of Neural Networks in trading and investment show that they have repeatedly delivered outstanding results.

Therefore, the motivation for this article is to investigate whether, in times of economic uncertainty, Neural Networks are able to achieve a significantly better forecasting accuracy and trading performance in exchange rate forecasting, using an example of the EUR/USD rate. Accordingly, the data set underlying the analysis consists of the exchange rate series from 3 January 2000 to 31 July 2007 for the in-sample period, reflecting usual volatility levels, whereas the out-of-sample period comprises data from 1 August 2007 to 27 February 2009, when market conditions change and become more volatile. In addition, the ability of different network types to adjust to the new conditions is assessed by comparing their performances. Therefore, three different network architectures are employed to predict the EUR/USD exchange rate: a classic Multi-Layer Perceptron (MLP), as well as a Recurrent Neural Network (RNN) and a Higher Order Neural Network (HONN).

The empirical results suggest that NNR models are indeed capable of modelling and forecasting the EUR/USD exchange rate series competently, achieving a significant positive trading return. Overall, all three Neural Network architectures used manage to outperform the benchmark models in both a simple trading strategy and even more significantly in a trading simulation using confirmation filters. The HONN model is found to produce the best results with respect to trading performance and forecasting accuracy for the sample on hand.

The study commences with a review of the relevant literature in the section ‘Literature review’, followed by the section ‘Data characteristics and transformations’ describing the data series characteristics and the transformations performed. The section ‘Benchmark models’ explains and discusses the methodology for the benchmark models, followed by the section ‘The neural networks architectures’, which introduces the theory of NNR models. The methodology of the application is illustrated in detail in the section ‘Neural networks application procedure’, the results of which are subsequently presented in the section ‘Empirical results’. Finally, the section ‘Conclusion’ summarises the main results and makes some concluding remarks about their limitations and possible directions for future research.

LITERATURE REVIEW

Since the late 1980s, Neural Network models have been an emerging technique for applications in business and finance. Many studies examined their performance in comparison with other model classes, and also the dependence of their performance on, for example, network architecture, input factors and network parameters in the context of different areas of designated use. In this short review, we focus mainly on FX forecasting in general and Neural Network applications for this task in particular. For a comprehensive summary of Neural Network applications for forecasting, please refer to Zhang *et al* (1998).

Finding a model that is capable of forecasting exchange rates successfully, out of sample remains something that has been desired by forecasters for many years. Meese and Rogoff (1983a, b) applied structural- and time-series models for exchange rate determination to the monthly dollar/pound, dollar/mark, dollar/yen and trade-weighted dollar exchange rates from March 1973 to June 1981, only to conclude that their candidate models significantly failed to outperform the random walk model in out-of-sample forecasting experiments.

Later investigations of statistical properties of exchange rate time series have repeatedly confirmed the existence of non-linearities in the data, explaining the poor forecasting ability using linear models (Hsieh, 1989; Meese and Rose, 1991; Chiarella *et al*, 1994; Brooks, 1996). Consequently, researchers looked for new ways to incorporate these non-linearities.

Among other non-linear approaches, several studies for modelling these properties have been performed using Artificial Neural Networks (ANN); a non-linear, non-parametric methodology that is solely data driven. Zhang *et al* (1998) mention the works of Weigend *et al* (1992), who used ANNs for exchange rate forecasting based on daily data and compared the results with a random walk model, and Refenes (1993) who applied Neural Networks to hourly data of one exchange rate time series and benchmarked the results against an exponential smoothing and an ARIMA model. Both studies report significantly better results of the ANN models.

On the contrary, Tyree and Long (1995) revisit the study of Weigend *et al* (1992) and a piece of work by Refenes *et al* (1993), who also found Neural Networks to be a successful forecasting tool when applied to hourly exchange rates, and criticise the methodologies of many studies to be not stringent enough when it comes to the assessment of Neural Networks' performance. The authors replicate some of the previous work, choosing USD/DEM daily data and using univariate and multivariate models to produce 1-, 5- and 20-day ahead forecasts and comparing the results with a random walk and an autoregressive (AR) model. The random walk was found to outperform Neural Networks at all forecast lead times in terms of forecasting accuracy assessed by the mean squared error.

Krishnaswamy *et al* (2000) discuss the use of Neural Networks in FX markets, citing the conclusions of Mehta (1995) who considers Neural Networks to be the best tool for FX forecasting available. However, they also point at problems and the requirement of a thorough understanding for setting up a network for practical use. A second study pointed out by the authors is Refenes and Zaidi (1995) who use Neural Network models for predicting the USD/DEM exchange rate, achieving superior results to moving average (MA) and mean value-based forecasts.

Leung *et al* (2000) examine General Regression Neural Networks (GRNN) using the monthly exchange rates from March 1990 to July 1995 of the Canadian dollar, Japanese yen and British pound and compare the forecast with another more commonly used network type, the Multi-Layered Feedforward Neural Network (MLFN), and furthermore a model based on multivariate transfer function and a random walk. They find the GRNN superior to all other models regarding forecasting accuracy.

Hann and Steurer (1996), however, can only confirm a superior performance of Neural Networks for weekly data, but conclude that for their investigation of the USD/DEM exchange rate for data from January 1986 to October 1994 linear models and Neural Networks in the framework of error-correction models give almost the same results for monthly data.

Zhang and Hu (1998) look into the impact of parameter settings of Neural Networks, applying them to the daily and weekly GBP/USD exchange rate from the beginning of 1976 to the end of 1993. The authors experiment with model parameters such as the training period and the number of input and hidden nodes. They conclude that the number of input nodes and the number of observations of the training period have a big impact on the forecasting performance, and therefore have to be chosen carefully. Furthermore, they state that Neural Networks outperform linear models for forecasting a short time horizon; however, for longer forecast horizons new observations should be used to revise the model once they become available, in order to reflect changes of the underlying pattern.

Although there are no other exchange rate series examined to give more robustness to the findings, following the findings of Diebold and Nason (1990), who investigate ten major nominal dollar spot rates from 1973 to 1987 and discover that there is little variation in results for different exchange rates when non-parametric techniques are used, it can be assumed that the results of Zhang and Hu (1998) could also apply to the EUR/USD exchange rate.

Huang *et al* (2004) summarise numerous papers that compare the forecasting performance of Neural Networks with other techniques. The review shows that of the eleven studies reported, ten used an MLP neural architecture for estimating and forecasting, benchmarked against most commonly a random walk model, but also frequently against ARMA/ARIMA or several types of GARCH models, whereas the most widely used statistical accuracy measures were root mean squared error (RMSE) and mean absolute error (MAE), in seven and three studies, respectively. The summarised results show that MLPs outperformed other models significantly in five articles, in two the MLP gave similar results as the best benchmark, but outperformed others, or was superior to the benchmark models only for a certain data frequency, whereas the outcome of four papers are mixed results. As possible reasons for these varying conclusions, the authors state the high data-dependency of non-parametric models like Neural Networks. This data-dependency can impact the learning and generalisation abilities of the network and lead to instable results if there are random variations, caused by, for example, data partitioning or subtle shifts in the parameters of the time-series generating process.

RNN models have been previously investigated by Kuan and Liu (1995), who aim at investigating their forecasting performance compared with the MLP. They examine five currency pairs over the time from 1980 to 1985 and observe that for certain, but not all of them, both the MLP and RNN are capable of producing daily forecasts superior to the random walk benchmark in terms of significantly lower out-of-sample mean squared error and the prediction of the correct directional change (CDC), with the RNN results being slightly superior.

Tenti (1996) describes the major drawback of RNN models to be requiring considerably more connections, and hence more time and memory in simulation than standard backpropagation networks. The article is based on opening and closing daily prices for Deutsche mark currency futures over 4 years from 1990 to 1994. He analyses three RNN architectures, employing a learning algorithm with either an output, hidden or input layer feedback link and finds that networks where information is fed back from the hidden layer give best results. However, these were not benchmarked against forecasting performances achieved with other, more traditional techniques.

The successful approximation and forecasting properties of RNNs, even when compared with alternative models, have been documented by Dunis and Huang (2002). They use RNNs for forecasting and trading the daily GBP/USD and USD/JPY FX rate volatilities by identifying mispriced options from December 1993 to May 2000, the performance of which is judged against an MLP structure, as well as a simpler GARCH model and implied volatility. They show that the RNN provides the best returns in a short-term trading context, although the forecasting accuracy can be improved further by using model combinations.

A third structure of Neural Networks that was given attention in research and practice of financial time series forecasting are HONN models. Experimenting with function approximation and financial time series simulations, Zhang *et al* (2002) prove that neuron-adaptive HONN (NAHONN) models can approximate any continuous function to any desired accuracy. Moreover, the results indicate that NAHONNs offer significant advantages over traditional neural networks such as faster learning because of much reduced network size and a smaller simulation error.

Fulcher *et al* (2006) state as one property of HONNs the capability of extracting higher-order polynomial coefficients in the data. Applying this model class, in particular a polynomial HONN and HONN Groups, to the daily Australian–US dollar exchange rate in March 2005 and other data, they find them to give roughly twice the performance on financial time-series prediction as compared with MLP architectures trained with backpropagation and claim another 10 per cent improved performance for HONN Groups.

An improved performance was also detected by Knowles *et al* (2009), who aim at determining whether HONNs in relation to MLP models can give greater return on investment. Using the same data as Dunis and Williams (2002, 2003), they extend the research towards AR and multivariate higher-order networks for the prediction of the EUR/USD exchange rate and implement a Bayesian Confidence measure in the trading strategy. The results show that both the AR and the multivariate HONN models yield significantly higher returns compared with the MLP and the more traditional techniques.

Also based on Dunis and Williams (2002, 2003), a more recent study by Dunis *et al* (2010) investigates the forecasting and trading performance of alternative Neural Network structures applied to the daily EUR/USD exchange rate time series. They find that NNR achieves good forecasting accuracy and outperforms technical and more traditional regression models, that is, a naïve strategy, a Moving Average Convergence Divergence model (MACD), an Autoregressive Moving Average (ARMA) model and a Logit model, significantly with regards to trading performance.

Dunis *et al* (2010) continue the idea of this article and introduce other types of Neural Networks, by sticking to the same data series and time period at the same time. The authors evaluate three more advanced Neural Network architectures, HONNs, Psi Sigma Networks and RNNs. The article provides a comprehensive overview of the structures and characteristics of each network type and compares their performance with the MLP, as well as to the Softmax Cross Entropy and Gaussian Mixture (GM) models evaluated by Lindemann *et al* (2005). The article shows that, for a simple trading strategy, the newly introduced RNN and HONN networks, as well as the most basic network structure MLP, equally produce very good results with respect to annualised return and information ratio. After the application of trading filters and leverage, however, it is shown that the GM model outperforms all other network architectures strikingly.

Finally, Dunis *et al* (2009) use the daily ECB fixing of the EUR/USD from 1999 to August 2008 to provide further evidence of the benefits of alternative neural networks architectures for modelling and trading this exchange rate.

DATA CHARACTERISTICS AND TRANSFORMATIONS

Keeping in mind that a model and its outcomes will always be highly dependent on the quality of data fed into it, taking a closer look at the underlying database, variable characteristics and transformations made is vital.

We use daily historical London closing prices for all variables that have been obtained from Thomson DataStream. Alongside the EUR/USD exchange rate as the independent variable, there are a number of fundamental explanatory variables, which are taken from Dunis and Williams (2003). Listed in Table A1 in the appendix, with their according DataStream mnemonic, they include important stock indices, commodity prices, monetary policy and macroeconomic influences, and other frequently traded currency exchange rates, as well as past values of the EUR/USD exchange rate itself.

The sample period for all variable time series used contains 2390 observations running from 3 January 2000 to 27 February 2009. This set is further divided into an in-sample period, which spans the period from 3 January 2000 to 31 July 2007 comprising 1977 trading days, and an out-of-sample period. On the basis of the in-sample data subset, the model parameters are estimated and then evaluated for the unseen subset retained in the out-of-sample period from 1 August 2007 to 27 February 2009, a total of 413 trading days. For the purpose of this study, the out-of-sample subset was chosen to contain a time period of increased market volatility in the wake of the financial crisis.

The data series were obtained in levels. In order to insure clean, error free and unbiased data for the later analysis, missing data points because of bank holidays were subsequently linearly interpolated for the entire time period. Furthermore, following Dunis and Williams (2003), the bond yield and interest rate series were combined to create interest rate yield curve series for the respective countries, in the manner:

$$YC = 10 \text{ year benchmark bond redemption yield} - 3 \text{ month middle rate} \tag{1}$$

Figure 1 shows the development of the EUR/USD exchange rate, that is, the amount of US dollars payable for 1 euro, for the period from 3 January 2000 to 27 February 2009. It can be seen that over that period of time, the euro overall appreciated against the dollar.

As can be seen from the chart above, the EUR/USD is clearly non-stationary, something

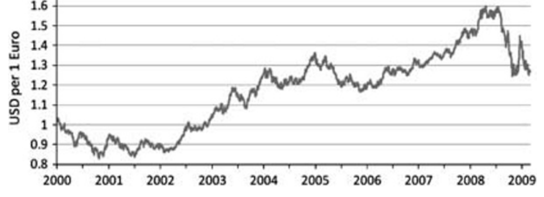

Figure 1: EUR/USD exchange rate (3 January 2000 to 27 February 2009).

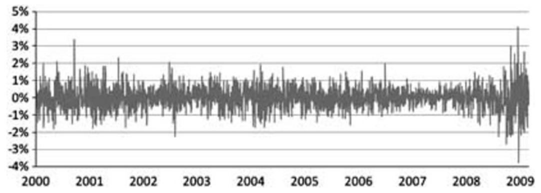

Figure 2: EUR/USD exchange rate returns (3 January 2000 to 27 February 2009).

confirmed by standard statistical tests not reproduced here in order to conserve space. In the same vein, the distribution of observations shows that the series is non-normal. As using a non-stationary data set in regression analysis will lead to spurious and invalid results, it is advisable to convert the series in a way that stationarity is assured. Accordingly, we transform all series into arithmetic rates of returns R using the prices P in time t and $t-1$:

$$R_t = \left(\frac{P_t}{P_{t-1}}\right) - 1. \tag{2}$$

The resulting EUR/USD return series is shown in Figure 2. After a visual inspection it can be assumed that the return series is now stationary, which is indeed confirmed by standard statistical tests again not reproduced here.

Further standard tests confirm the non-normality of the EUR/USD series and the presence of heteroskedasticity at a 99 per cent confidence interval.

BENCHMARK MODELS

Different benchmark models have been applied in the literature, such as a random walk simulation (for example, Zhang and Hu, 1998), technical MACD models and linear regression models such as ARMA models or a logit estimation (for example, Dunis and Williams, 2003). Some of these approaches, however, only produce classification estimates that make it impossible to compare their actual forecasting accuracy in absolute terms. Accordingly, we select a naïve strategy and an ARMA model for this study, as both give a level estimate that allows for a comparison of both statistical forecasting accuracy and trading performance.

Naïve random walk model

As price changes in efficient markets such as exchange rates are assumed to be a random distribution with a mean of 0 (Pindyck and Rubinfeld, 1991, p. 441), the central idea of a naïve random walk is to suppose that the future price will be the same as today's price, and hence that today's actual return value is the best estimate of the next trading day's return. Formally, this can be expressed as:

$$\hat{y}_{t+1} = y_t, \tag{3}$$

where $\hat{y}_{t+1}$ represents the one-day-ahead forecast of the return series and y_t the rate of return in time t. Pursuing this strategy, it is clear that a profit in $t+1$ will be made when the exchange rate moves in the same direction as in the previous period t, and a loss if there is a change of direction – and thus a change of sign of the rate of return – from one period to the next. Hence, this strategy is most profitable when the exchange rate moves following distinct trends with only few fluctuations.

The naïve strategy requires no in-sample regression or optimisation of any other kind. However, for the sake of completeness and comparability, the in-sample performance of the strategy was calculated nevertheless. Furthermore, trading results for the out-of-sample period were determined in order to serve as a benchmark for the more complex models and their performance on unseen data.

ARMA models

Linear regression models have been widely employed for forecasting. Among these, ARMA models are a popular method, as they are comparatively fast and easy to apply. AR MA models are based solely on past observations of the variable to be forecast. Box *et al* (1994) denote the relationship between the independent variable y_t and the explanatory variables, that is, the AR and MA terms, formally as:

$$\begin{aligned} \hat{y}_t = \phi_0 + \phi_1 y_{t-1} + \cdots + \phi_p y_{t-p} \\ + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \cdots - \theta_q \varepsilon_{t-q} \end{aligned} \tag{4}$$

for an ARMA(p, q) model consisting of an AR process of order p and an MA process of order q, where ϕ_0 stands for the intercept and ε_t for the white noise error. The parameters of the model, the coefficients of the AR and MA variables $\phi_0, \phi_1, \ldots, \phi_p, \theta_1, \ldots, \theta_q$ are determined using a least squares regression. Hence, the model does not assume a particular theory or predetermined pattern in the data, but iteratively fits the data around the general model framework to identify a specific model.

In order to find relevant ARMA models, various specifications were tested for their statistical significance. This was done by the setting up unrestricted ARMA(p, q) models for the EUR/USD return series, with $p = q = 1, 2,, \ldots, 10$ based on the in-sample data subset. In accordance with the detection of heteroskedasticity mentioned previously, all ARMA models were adjusted for heteroskedasticity using the White heteroskedasticity-consistent standard error and covariance. As the models established contained several parameter coefficients that failed to be significant at level $\alpha = 5$ per cent or even $\alpha = 10$ per cent, the models had to be further modified by gradually removing insignificant AR and MA terms. This was done by repeatedly eliminating the term with the highest *P*-value in the series and re-estimating the model until all variable coefficients were significantly different from 0 at the 99 per cent confidence interval.

As a result, six ARMA specifications were found to be able to map the EUR/USD return series. These models were then used to produce an in-sample forecast that was implemented in a trading strategy for the evaluation of their trading performance. Overall, the restricted ARMA(6, 6) outperformed all other model specifications in most of the important trading performance measures, particularly annualised return and annualised volatility, where it gave results of 15.22 per cent and 9.27 per cent, respectively. The model was furthermore examined regarding its overall significance. Table A3 shows the model specifications, and shows that all terms included are significant at a 99 per cent confidence interval.

The *F*-test statistic in Table A4 also tests for the joint significance of all coefficients: the null hypothesis that all coefficients jointly are not significantly different from 0 can be rejected at a 98 per cent confidence level, thanks to a *F*-test statistic of 2.32. Ramsey's RESET test for the detection of general misspecification was performed and its results in Table A4 in the appendix show that the null hypothesis that the model is specified correctly cannot be rejected at a 1 per cent significance level. Furthermore, the correlogram of the residuals reflecting the autocorrelation function of the restricted ARMA(6, 6) equation reveals that the residuals are random at a 99 per cent confidence interval (cf. Table A5 in the appendix). Therefore, the restricted ARMA(6, 6) model, formally described by Equation (5), is retained as best of its class for the out-of-sample estimation.

$$\begin{aligned}\hat{y}_t = {} & 0.0003 + 0.3850y_{t-3} + 0.3077y_{t-4} \\ & + 0.6522y_{t-5} - 0.3728y_{t-6} + 0.3868\varepsilon_{t-3} \\ & + 0.2915\varepsilon_{t-4} + 0.6524\varepsilon_{t-5} - 0.3457\varepsilon_{t-6}. \qquad (5)\end{aligned}$$

THE NEURAL NETWORKS ARCHITECTURES

It has been shown that, under certain conditions, neural networks can be universal approximators for any continuous function (Hornik *et al*, 1989). As they are intrinsically non-linear, they have the ability to detect non-linearities in the data, a characteristic that is inherent in financial time series data, and can produce much better forecasting results than linear models. The idea for this class of models was inspired by the structure of the brain and has very powerful information processing properties.

As opposed to many more traditional models, neural networks are non-parametric models and therefore do not have a predefined functional form. This is advantageous particularly for modelling complex financial time series like exchange rates. As there is no indisputable theory explaining the relationship between the movement of FX rates and other potential explanatory factors, the use of neural networks has become popular in recent years in both theory and practice, because they require no *a priori* assumptions about, for example, the distribution of the data, but instead are solely data driven. In the section 'Literature review', we already mentioned how various studies have testified a better relative performance of neural networks when compared with technical analysis or classic linear and non-linear econometric models.

Besides their capability of modelling a function to any desired accuracy, the particular usefulness of neural networks for forecasting purposes is their generalisation ability. After being appropriately trained with an in-sample data set, the model can generalise from the analysed patterns to unseen data.

However, neural networks are also subject to certain drawbacks and pitfalls. If the network is trained too much, the crucial generalisation ability just mentioned fades, as the neural network may learn not only the meaningful patterns in the data, but also the noise inherent in it. This problem is known as overfitting. Other problems are the enormous number of 'parameters' to be adjusted to fix the data and the resulting excessive training time for the model optimisation. Furthermore, because of the lack of a systematic approach for setting up the neural network structure, learning algorithm and parameters, the task of training such a network is still mostly based on trial and error (Vellido *et al*, 1999).

Another common critique made about neural networks is their black-box character. The absence of a functional form, on the one hand a beneficial quality, leads to the weakness that even if a NNR is based on causally related data, the resulting model, that is, the values of the weights and biases, may not give a much insight into the nature or the strength of the functional relationships within it (Tenti, 1996).

Within the class of neural networks, there are several different architectures, which all have specific strengths and weaknesses. The models used in this research are the MLP, the RNN and the HONN. Some properties, however, all architectures have in common. They consist of interconnected processing units, called neurons or nodes. The connections between these nodes are called weights and represent the sign and strength of the relationship of one node with respect to another. These weights are comparable with coefficients in a linear regression, hence they, along with the network architecture, determine and store the 'knowledge' of a trained network (Kaastra and Boyd, 1996).

The nodes are arranged into layers; an input layer, possibly one or more hidden layers, and an output layer, where the number of nodes per layer depends on the modelling task and the characteristics and size of the data fed into the network. The processing nodes in the hidden and the output layer contain the actual processing capacities, the so-called transfer function. This can, depending on the problem, be either linear or non-linear, for example, a sigmoid or hyperbolic tangent function. For each of the hidden and output nodes, there is also a bias parameter that serves an analogous purpose as the intercept in linear regression models. It has always the value of 1.

Once the network architecture and all the parameters are set up, the data from the input nodes are fed through the network. The training process changes the weights, which at the beginning are set to be randomly drawn from a normal distribution, so that the error of the estimated output value and the actual output value is minimised; this mechanism is called backpropagation. Because the data are processed only in one direction, from the input layer through the hidden layer(s) to the output layer, this type of model is known as a feedforward network.

Neural networks are in fact in many ways similar to linear and non-linear parametric least squares regression models. Both neural networks and conventional regression analysis aim at minimising the sum of squared error. Furthermore, the input nodes and the output node(s) are equivalent to the exogenous variables and the endogenous variable, respectively, and the weights can be pictured to resemble the coefficients in a linear regression. Hence, a linear regression model could be expressed as a feedforward neural network with no hidden layers and one output neuron with a linear transfer function.

In order to point out some architecture-specific properties of the different neural network types, the following subsections discuss the MLP, RNN and HONN architectures more in detail.

MLP

Among the different neural network structures, one of the most widely used for practical business and financial applications is the MLP (Vellido *et al*, 1999). At the same time, it is usually described as the most classic network architecture while, however, having still proven to be a very good forecasting method compared with other more complex neural networks.

The MLP architecture is shown in Figure 3, with the explanations of the notations used given below (Dunis *et al*, 2010). The model here consists of three layers, an input layer with three input nodes and the bias node, one hidden layer containing three hidden nodes with a sigmoid transfer function and an output

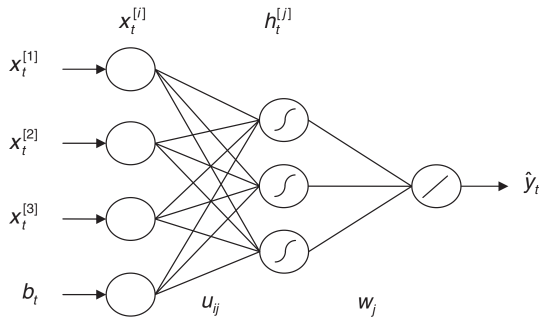

Figure 3: The MLP architecture.

layer with one output node that combines the hidden outputs using a linear transfer function in order to estimate the output, for example, the EUR/USD exchange rate return in time t, denoted with $\hat{y}_t$.

As a kind of MLFN, the MLP can be designed with more than one hidden layer. However, for this study only MLPs with one hidden layer were considered, as Hornik *et al* (1989) proved that a single hidden layer is sufficient for neural network models to map any continuous function, as long as there are an adequate number of hidden nodes in this layer to meet the requirements of the complexity of the data. Where: $x_t^{[i]}$ $(i = 1, 2, \ldots, n)$ are the model inputs (including the input bias node b_t) at time t; $h_t^{[j]}$ $(j = 1, 2, \ldots, k)$ are the hidden nodes outputs (including the hidden bias node); $\hat{y}_t$ is the MLP model output (estimated rate of return) at time t; u_{ij} and w_j are the network weights; $\oslash$ is the transfer sigmoid function: $S(x) = \frac{1}{1+e^{-x}}$, $\oslash$ is a linear function: $F(x) = \sum_i x_i$.

The error function to be minimised during the training is:

$$E\big(u_{ij}, w_j\big) = \frac{1}{T}\sum \left(y_t - \hat{y}_t(u_{ij}, w_j)\right)^2, \qquad (6)$$

where y_t is the known target value, that is, the actual rate of return at time t.

RNN

Another model class that has been shown to be a successful approach for estimating and forecasting financial time series are RNN. The structure and function of an RNN are presented briefly in Figure 4, with the explanation of terms and symbols used given below. For a detailed discussion of the RNN theory and mechanics, see Elman (1990).

RNN models are more complex than MLPs, as they loop back hidden node output information from the previous period *t*-1 to use it as a new input for the estimation of the current period *t*. This gives the model a short-term memory and the ability to recognise very complex patterns in time series, making it intuitively suitable for the EUR/USD exchange rate.

Where: $x_t^{[i]}$ $(i = 1, 2, \ldots, n)$ are the model inputs (including the input bias node b_t) at time t; $h_{t-1}^{[j]}$ $(j = 1, 2, \ldots, k)$ are the hidden nodes outputs of the previous period looped back to the input layer; $h_t^{[j]}$ $(j = 1, 2, \ldots, k)$ are the hidden nodes outputs (including the hidden bias node); $\hat{y}_t$ is the RNN model output (estimated rate of return) at time t; u_{ij} and w_j are the network weights; ∫ is the transfer sigmoid function: $S(x) = \frac{1}{1+e^{-x}}$; ⊘ is a linear function: $F(x) = \sum_i x_i$.

Like for the MLP, the error function to be minimised by adjusting the network weights is:

$$E(u_{ij}, w_j) = \frac{1}{T}\sum (y_t - \hat{y}_t(u_{ij}, w_j))^2, \quad (7)$$

where y_t is the target value, that is, the actual rate of return at time t.

The increased complexity of RNNs is obvious when comparing Figure 4 with Figure 3 as both MLP and RNN have the same number of input and hidden nodes. However, this does not affect the model's forecasting performance, but as a negative side effect computation capacity needs increases, which leads to the training procedure for RNN models being exceedingly time consuming.

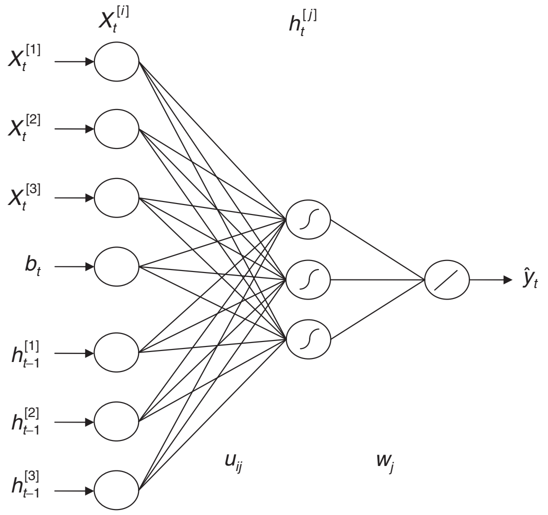

Figure 4: The RNN architecture.

HONN

A less complex approach relative to both RNNs and MLPs are HONN models, which need fewer connections between nodes and therefore fewer weights to be adjusted during their training that results in shorter computational times. This is an important advantage for practical trading and investment applications.

The HONN architecture is presented in Figure 5. It depicts a second order HONN with three inputs and a sigmoid transfer function in the output node, where: $x_t^{[i]}$ $(i = 1, 2, \ldots, n)$ are the ordinary model inputs, the combined inputs and the input bias node b_t at time t; $\hat{y}_t$ is the HONN model output

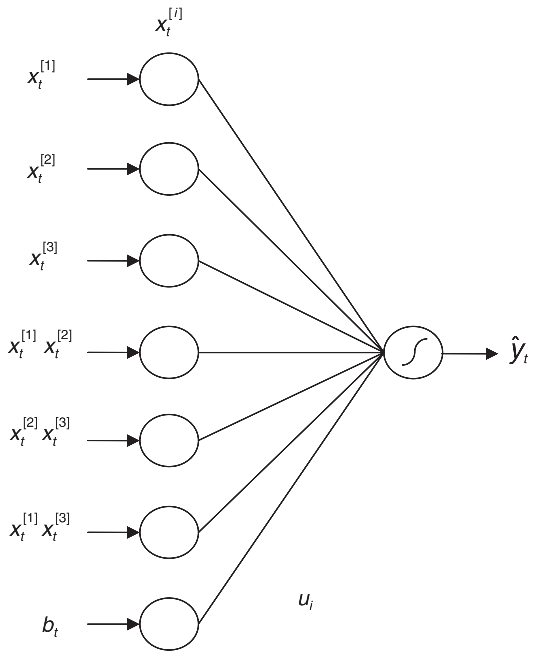

Figure 5: Second order HONN with three inputs.

(estimated rate of return) at time t; u_i are the network weights; ∅ is the transfer sigmoid function: $S(x) = \frac{1}{1+e^{-x}}$. The error function to be minimised during the training is again the mean squared error:

$$E(u_i) = \frac{1}{T}\sum (y_t - \hat{y}_t(u_{ij}))^2 \qquad (8)$$

with y_t being the known target value, that is, the actual rate of return at time t.

It can be seen that for the same number of inputs, the HONN needs significantly fewer weights to be adjusted during the network training, less than half of the number of weights of the MLP and less than a third when compared with the RNN. The smaller number of weights is compensated, however, by the use of joint activation functions, that is, the inputs are combined before being fed into the network. Using this technique, the network does not have to establish the relationships between the inputs; rather, it enables the network to extract higher-order polynomial coefficients in the data (Knowles *et al*, 2009). Another advantage of HONN models is the direct correspondence between network weights and polynomial coefficients. Therefore, this architecture can give valuable information about the functional relationship between inputs and outputs, and hence can be considered an open-box model, as opposed to other neural network types (Fulcher *et al*, 2006).

On the other hand, it has to be mentioned that the number of input nodes for HONNs, because of the combination of inputs, increases over proportionally for additional added explanatory variables, and so the number of inputs, as well as the order of the network, which is usually not chosen to be four or more, has to be limited to avoid the network being outsized (Knowles *et al*, 2009)

NEURAL NETWORKS APPLICATION PROCEDURE

After giving some theoretical illustration of neural networks, we present our application of the methodology to the EUR/USD exchange rate.^{2}

Input selection and preparation

The first important step was choosing the explanatory variables for the models. The financial and economic time series to be used as potential input factors, as well as the transformations made, were already mentioned in the section ‘Data characteristics and transformations’. These included the interpolation of missing data, the calculation of yield curves and the transformation of the series in levels into return series. However, the variables given cannot be used just in their original form, but have to be lagged. For estimation of the EUR/USD rate of return in time *t*, the model can only use explanatory variables from the previous trading day $t-1$. For certain variables, namely the S&P500 Composite Index and the US yield curve, denoted with S&PCOMP and US_YC, respectively, because of the time difference to US American markets, only the second lag can be used for prediction.

Hence, from all explanatory variable time series and the EUR/USD return series itself, lagged series were produced up to lag 20 for the in-sample data set from 3 January 2000 to 31 July 2007. The out-of-sample subset was not included, as the data are retained for validation purposes. Therefore, it remains ‘unseen’ for the model and not used for the model optimisation procedure.

After the creation of the lagged series, their cross-correlations with the EUR/USD return rate were calculated. These were then converted into absolute values, as contrary to their strength the sign of the relationships is of minor importance, as it will be detected by the model. Subsequently, three lagged variable pools were selected and pooled. The first pool, consisting of 19 series, includes the best lag of each variable, the second pool includes the 18 lagged series with an absolute cross correlation with the EUR/UDS return series higher than 0.05 and the third pool comprises the 14 lagged variable series with a cross-correlation higher than 0.055.[3] The second and third selections were chosen regardless of whether all potential variables were represented in the pool. An overview of the variables and related lags selected is given in Table A6.

True, the use of fundamental analysis is a subjective issue, as we might not have included all factors influencing exchange rate movements. Furthermore, the choice of variables based on the cross-correlations, which is a linear concept, is not necessarily a good criterion for choosing input factors for a non-linear model, as the cross-correlation analysis cannot detect non-linearities inherent in the data. Neural networks do not provide significance measures for their ‘coefficients’, that is, weights, like is the case for linear regression with the *t*-test. Another possibility is to solely rely on technical factors like AR terms of the series. However, these are also included in the model, so that it is more likely for the model containing fundamental variables as well to have a broader range of information to extract indications for the model estimate. Therefore, though it might not be the optimal approach, it was considered a most sensible one.

Before being fed into the networks as input factors, some studies argue that the variables should be normalised in order to avoid the functions saturation zones. Yet, there is no consensus in the literature about whether this brings significant benefit for estimation and forecasting. Zhang *et al* (1998) state that the investigation of this issue has led to the conclusion that data normalisation can have positive effects on the classification rate and mean squared error, but that this benefit decreases with a larger sample size and as well usually slows down the training process. Furthermore, the scaling can be undone by the network weights during training. Zhang and Hu (1998) use raw data for their investigation after finding no significant difference between using normalised and original data. Therefore, bearing also in mind that this is much less of an issue here, as all our data are in percentage form, we decided to refrain from normalising the inputs.

Subsample periods

All the selected time series are subsequently divided into subsamples. For the benchmark models, these were the in-sample and out-of-sample period. Neural networks, on the contrary, are trained with three subsample periods, training, test and validation, where training and test combined are equivalent to the benchmark in-sample period, as illustrated in Table 1. The number of observations ratio 'training: test: validation' varies throughout the literature and can depend on the sample size or the particular problem (Zhang *et al*, 1998); in accordance with Dunis and Williams (2002, 2003), for this study it was chosen to be approximately 2/3: 1/6: 1/6, which is equivalent to 1565 observations in the training set, and 412/413 observations in the test and validation set, respectively.

The reason to categorize the in-sample period further is in order to avoid the common problem of overfitting during neural network training. When the network is trained, the forecast error of the training subset decreases. However, once the network starts being trained too much, it does not only learn the patterns and similarities inherent in the data, but also the noise. As a result, it loses its generalisation ability and will not produce good forecasts anymore. To avoid this drawback, the network error is monitored on the test set, it initially decreases, but starts to increase once the network starts overfitting the data: the training procedure is then stopped when the test error is at its minimum.

After the training and the adjustment of the weights to minimise the test error function, the performance of the network is tested on unseen data, that is, data that were not part of the training process. This is to simulate the conditions of the real world and show if the network has indeed managed to map the underlying patterns in the data and generalised to extrapolate them into the future.

Network characteristics

The critical part of the application of neural networks is setting up the optimisation characteristics. These crucially influence the network performance. Kaastra and Boyd (1996) list as some aspects to be considered: the learning rate and momentum, the number of training cycles and block, that is, how often the training cycle is repeated, as well as

Table 1: Sample period subsets

	Benchmark models	*Neural networks*	*Observations*
In-sample	2000:01–2007:07	2000:01–2005:12 (training)	1565 (2/3)
		2006:01–2007:07 (test)	412 (1/6)
Out-of-sample	2007:08–2009:02	2007:08–2009:02 (validation)	413 (1/6)

important typology characteristics of the network. These include the parameters that have the largest impact on the forecasting accuracy, the number of input nodes and hidden nodes (Zhang *et al*, 1998).

One method to determine the optimal number of hidden nodes was introduced by Le Cun *et al* (1990) and is known as Optimal Brain Damage: the idea is to start with an oversized model and to gradually prune redundant weights during the training procedure. Other methods are, for instance, the weight decay or weight elimination, using a penalty term for an increased network size that is added to the error function, a neural network information criterion assessing the trade-off between the accuracy of the approximation and the size of the network or a Principal Component Analysis (Hann and Steurer, 1996).

For the number of input nodes, one possible approach as used by Dunis and Williams (2003) is to start the network with very few inputs and then step by step add more inputs. The added input is retained when the Explained Variance (EV) increases, that is, when the variable adds useful information to the model, but generally a smaller network structure is preferred to a larger one.

For this study, a different approach was used, the idea of which is to make use of the data-driven nature of neural networks. It was mentioned that three different pools with 19, 18 and 14 potential variables were selected. Now, instead of restricting the training process to certain inputs or a certain number of hidden nodes, all possible combinations of input variables from the pool and a given number or range of hidden nodes were calculated. A dedicated combination programme was created to this effect. Formally, this is:

$$z = \sum_{k=x_{\min}^{[i]}}^{x_{\max}^{[i]}} \binom{n}{k} \left[\left(h_{\max}^{[j]}\right) + \left(h_{\min}^{[j]}\right) + 1 \right], \quad (9)$$

where z is the number of combinations, n is the pool size, that is, 19, 18 or 14, k is the number of variables to be chosen from the pool and lies in the range of the set minimum number of inputs $x_{\min}^{[i]}$ and the maximum number of inputs $x_{\max}^{[i]}$; $h_{\min}^{[i]}$ and $h_{\max}^{[i]}$ represent the minimum and maximum number of hidden nodes, respectively.

Starting with the MLP, the pool containing the 19 series that are the respective best lag for the variables, the combination programme is run and z networks are trained. Several programme runs like this with different random initial weights are performed and saved for later evaluation. The initially very large number of trained networks is reduced by filtering the results, that is, properties and outputs of the networks, to find successful models for the forecasting task. The filtering process details are further explained in the next subsection.

The same was done when taking the inputs from the second pool containing the 18 lagged series with a correlation larger than 0.05, in order to evaluate if these gave better results. For both the RNN and HONN, the third pool of 14 variables with a cross-correlation larger than 0.055 was chosen in order to reduce the number of combinations and therefore the computing time. This is known to be higher for RNN models because of the increased complexity of the looped back outputs from the hidden layer; for the HONN, the combination of inputs before being fed into the network also leads to longer training times for a larger choice of variables. For this reason and for the sake of comparability, the 14-variable pool was also used for the MLP.

Further features of the training process, that is, the number of training cycles and blocks, the iterations and momentum rate, can be taken from Table A7 and are applicable to all network architectures used. When looking at other studies that state the number of iterations used (that is, the number of cycles times the number of blocks), the iterations in this work might seem comparatively small, which is because of the large amount of networks trained and the time needed associated with this. As in tests, the error rate did not decrease extensively further when more iterations were used, the trade-off of iterations versus training time was made in favour of the latter.

Weight matrix approach

Acknowledging the lack of a standard statistical test for neural networks, Dunis and Williams (2003, p. 28) state that 'once a tentative model is selected, post-training weights analysis helps to establish the importance of explanatory variables'. This is to be achieved by finding 'a measure of contribution a given weight has to the overall output of the network, in essence allowing detection of insignificant variables'. For this task, they use the Hinton graph, which is a graphical representation of the inputs to hidden nodes weight matrix. The idea is that a variable, for which the weights of its input node to all hidden nodes are small, has only little impact on the model estimation and can be considered insignificant. On the contrary, if an input node exhibits large weights, all of which are of the same sign for all the hidden nodes it is connected to, it appears logical that the variable has a clear impact and influences the output in either a significantly positive or negative way. Furthermore, small bias node weights are preferred.

On the basis of this analysis, the optimisation approach looks for models with a consistent weight matrix, as assessed by the Hinton graph, which would be somewhat meaningful as opposed to a model where the weights of one input change sign and size for different hidden nodes. This idea was implemented into the network selection process.

In the first step, during training only networks survive for which at least 80 per cent of the weights of each input node are of the same sign. Allowing for 20 per cent, deviations increased the likelihood of finding suitable models, particularly for a large number of hidden nodes. The models that are found to meet the criteria are kept and saved for further screening.

In the second step, the weight size is checked to avoid networks with both very small input node weights and very large bias weights. The filter examines the models formerly retained and separates out those for which input weights are smaller than a set threshold value, as well as those with a bias weight that exceeds a set bias threshold. In addition to the weight size, another filter is implemented to measure the CDC of the model forecast for the training and test periods, that is, the rate at which the sign of the actual exchange rate returns was estimated accurately. Although the annualised return not only depends on this rate, but also on the size of the movements predicted correctly, it provides a rough indication of the return that might be achieved when the model estimation is used as the basis of a trading strategy. The CDC is considered in order to not only retain models with a significant weight matrix, but models that also manage to produce profitable forecasts when employed for trading.

However, after several test runs, most of which included more than 100 000 models tested, none of the models managed to outperform the simpler ARMA benchmark with respect to the in-sample annualised return. Table A8 in the appendix shows the weight matrix of the overall best model, which is visualised using the Hinton graph, given in Figure A1. Although the model produces a reasonable in-sample estimation regarding its forecasting accuracy, the out-of-sample results of the trading performance, generating an annualised return of −13.22 per cent, cannot be considered satisfying (cf. Table A9 in the appendix). The result indicates that the consistency of the weight matrix does not affect a model's forecasting or trading performance whatsoever.

Another problem of the approach was the lack of robustness. If a network that met all the criteria was trained again with the same parameters, but different random starting weights, the weight matrix was found to be inconsistent again. Therefore, the approach, though it might be useful for post-training weight adjustments, appears to be ineffective as a basis for the network choice.

EV approach

The second approach that is implemented is based on the EV. This is a measure of the proportion in which the model accounts for the variations in the data set, and can be approximated by the correlation between the actual series and the model forecast estimate.

The filtering process happened again in two steps. First, during the training process only models with an in-sample EV greater than 5 per cent are retained for further filtering. In the next step, another filter is applied to the surviving models, which assesses, as in the previous approach, the CDC of the model forecasts with respect to the actual exchange rate returns.

This approach worked better than the weight matrix approach and was therefore used for all network architectures: MLP, RNN and HONN. Several runs with different input and hidden node ranges, as well as parameter settings were tried. The best model was then chosen as the model that concurrently performed well regarding the forecasting error measure mean absolute percentage error (MAPE), balance of the relative correct prediction of rise or fall periods and CDC. These have shown repeatedly to be a good indicator for models that generalise well. The in-sample return was looked at as a minor guide for choosing a model, as coincidental good returns can be achieved because of the nature of the data, for example, in a rising period a model tends to do better on upward market movements, but will fail to produce a good forecast when the market goes down and vice versa. A further criterion is the number of input and hidden nodes, where more parsimonious models with similar performance were preferred to more complex ones.

Moreover, the correlation of the forecast and actual returns with respect to the different subsamples served as a guiding principle. The scenario of the correlation being very high in the training period but significantly lower in the test period indicates that the model has poor generalisation ability. Thus, the correlation

Table 2: Explained variance of the best MLP model for training, test and validation periods

	Training	*Test*	*Validation*
Explained variance (%)	7.44	6.02	6.79

of the validation period is likely to be low too. Table 2 below presents the correlation for the three subsets of the chosen MLP model, which confirms a balance throughout the entire sample period.

For the three architectures MLP, RNN and HONN, the best in-sample model was chosen, the performance of which is illustrated in Table A10. Furthermore the specifications of the respective best model are given in Table A11–A13 in the appendix. These models are subsequently assessed in terms of their out-of-sample forecasting accuracy and trading performance.

EMPIRICAL RESULTS

In order to assess the actual performance of the models used, their estimates are implemented into a trading simulation. The models are first compared with their own model class according to their in-sample performance and the best model is retained for out-of-sample forecasting. Then model performances are evaluated with respect to unseen data from the out-of-sample period.

The trading simulation is that a long EUR/USD position is taken when a positive return is forecast by the model and a short position otherwise. At the end of the data set, the trading performance is assessed following Dunis and Williams (2003) by calculating common performance measures like the annualised return, the annualised volatility and the resulting information ratio, the maximum drawdown, the average gain/loss ratio and the number of transactions.

Furthermore, the MAE, the MAPE, the RMSE, Theil's U and the CDC are chosen as measures for the statistical forecasting accuracy. The formulae used for calculating these performance measures are given in Table A14 and Table A15 in the appendix.

On the basis of the in-sample results, for the annualised return the ARMA model is to be favoured for trading on unseen data. However, regarding the statistical performance, it is outperformed by all neural network structures (cf. Table A2 and Table A10). The RMSE of the ARMA model is significantly higher than the one of the HONN. It is also outperformed by the MLP with respect to Theil's Inequality Coefficient.

Yet, the performance results can vary for the out-of-sample data, as the models are optimised to fit in-sample.

The results of the out-of-sample forecast are shown below in Table 3. It can be seen that the HONN model dominates top results of both the trading performance and the forecasting accuracy. The MLP model performs as joint best model thrice and is second best for various measures; it outperforms all other models on the maximum drawdown, a much desirable feature in real life trading. The RNN is one of the best models regarding the CDC, as well as the number of trades and the percentage of winning trades.

The ARMA model has an advantage over all other models only regarding the number of transactions. Moreover, it is the model with the

Table 3: Models out-of-sample performances compared

	Random walk	*ARMA*	*MLP*	*RNN*	*HONN*
Trading performance					
Cumulative return (%)	16.65	3.59	28.16	25.45	30.10
Annualised return (%)	10.21	2.20	17.22	15.57	18.41
Annualised volatility (%)	12.96	12.96	12.91	12.92	12.91
Information ratio	0.79	0.17	1.33	1.20	1.43
Maximum drawdown (%)	−9.16	−17.74	−8.14	−15.94	−16.25
Profits *T*-statistics	15.97	3.44	27.07	24.45	28.96
Total trading days	412	412	412	412	412
# Winning periods	216	214	218	218	209
# Losing periods	196	198	194	194	203
Winning periods (%) (=CDC)	52.31	51.94	52.91	52.91	50.73
Maximum gain in winning periods (%)	4.11	4.11	4.11	4.11	4.11
Maximum loss in losing periods (%)	−3.77	−3.77	−2.69	−3.77	−3.01
Average gain in winning periods (%)	0.58	0.56	0.60	0.60	0.63
Average loss in losing periods (%)	−0.56	−0.58	−0.53	−0.54	−0.50
Average gain/loss ratio	1.05	0.95	1.13	1.11	1.26
# Periods market returns rise	212	212	212	212	212
# Winning rise periods	114	175	129	102	103
Winning rise periods (%)	53.77	82.55	60.85	48.11	48.58
# Periods market returns fall	200	200	200	200	200
# Winning fall periods	101	39	89	116	106
Winning fall periods (%)	50.75	19.50	44.50	58.00	53.00
# Transactions (trades)	195	122	203	157	213
Winning trades (%)	47.18	45.08	51.72	53.50	53.52
Losing trades (%)	52.82	54.92	48.28	46.50	46.48
Statistical performance					
Mean absolute error	0.0082	0.0057	0.0057	0.0058	0.0057
Mean absolute percentage error (%)	610.35	120.34	130.09	115.43	99.64
Root mean squared error	0.0110	0.0081	0.0082	0.0083	0.0081
Theil's inequality coefficient	0.6757	0.9289	0.8369	0.8547	0.9585

lowest return and it even underperforms the random walk strategy. The cumulative profit evolution of all strategies over the course of the out-of-sample period is portrayed in Figure 6.

On the whole, all models yield a positive return in the trading simulation. Yet, in contrast to real world trading, these simulation results do not account for transaction costs, which can obviously affect profititability quite significantly.[4]

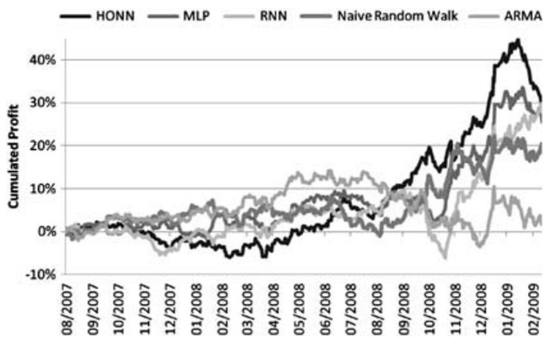

Figure 6: Out-of-sample cumulated profit of all models compared.

Table 4 gives the adjusted returns and information ratios for all strategies. The cost per trade is calculated according to Dunis and Williams (2003), who state that for an amount of \$5–10 million to be traded, the cost per trade usually accounts for 3 pips, that is, 0.0003 EUR/USD. Expressed in returns relative to the average EUR/USD bid rate during the out-of-sample period, the cost of one transaction equals:

$$\frac{0.0003}{1.44033} = 0.02083\%.$$

It can be seen that, although the HONN produces the highest number of transactions, it still offers the highest return and information ratio adjusted for transaction costs of 4.44 per cent.

Table 4: Trading simulation results corrected for transaction costs

	Random walk	*ARMA*	*MLP*	*RNN*	*HONN*
Trading performance					
Cumulative return (%)	16.65	3.59	28.16	25.45	30.10
Annualised return (%)	10.21	2.20	17.22	15.57	18.41
Annualised volatility (%)	12.96	12.96	12.91	12.92	12.91
Information ratio	0.79	0.17	1.33	1.2	1.43
# Transactions (trades)	195	122	203	157	213
Total trading days			412		
Costs per trade (%)			0.02083		
Transaction costs (%)	4.06	2.54	4.23	3.27	4.44
With transaction costs					
Cumulative return (%)	12.59	1.05	23.93	22.18	25.66
Annualised return (%)	7.70	0.64	14.64	13.57	15.70
Annualised volatility (%)	12.96	12.96	12.91	12.92	12.91
Information ratio	0.59	0.05	1.13	1.05	1.22

Table 5: Trading simulation results with confirmation filter

	Random walk	*ARMA*	*MLP*	*RNN*	*HONN*
Trading performance					
Cumulative return (%)	15.38	0.16	31.49	30.81	30.60
Annualised return (%)	9.41	0.10	19.26	18.84	18.71
Annualised volatility (%)	12.94	12.96	12.90	12.90	12.90
Information ratio	0.73	0.01	1.49	1.46	1.45
Maximum drawdown (%)	−9.67	−22.26	−9.47	−13.76	−16.25
Profits *T*-statistics	14.75	0.15	30.30	29.64	29.44
Total trading days	412	412	412	412	412
# Winning periods	218	211	227	221	209
# Losing periods	194	201	185	191	203
Winning periods (%) (=CDC)	52.91	51.21	55.10	53.64	50.73
Maximum gain in winning periods (%)	4.11	4.11	4.11	4.11	4.11
Maximum loss in losing periods (%)	−3.77	−3.77	−2.69	−3.77	−3.01
Average gain in winning periods (%)	0.57	0.56	0.59	0.60	0.63
Average loss in losing periods (%)	−0.56	−0.58	−0.55	−0.53	−0.50
Average gain/loss ratio	1.01	0.95	1.07	1.13	1.26
# Periods market returns rise	212	212	212	212	212
# Winning rise periods	117	190	136	102	103
Winning rise periods (%)	55.19	89.62	64.15	48.11	48.58
# Periods market returns fall	200	200	200	200	200
# Winning fall periods	101	21	91	119	106
Winning fall periods (%)	50.50	10.50	45.50	59.50	53.00
# Transactions (trades)	184	57	128	148	204
Winning trades (%)	46.20	52.63	52.34	52.03	53.92
Losing trades (%)	53.80	47.37	47.66	47.97	46.08

Another approach to deal with transaction costs is using confirmation filters. Lindemann *et al* (2005) and Dunis *et al* (2010) have used this filter, the idea of which is that, looking at the forecast, only those trades are performed for which

$$|\hat{y}_t| > d \tag{10}$$

Table 6: Trading results with confirmation filter corrected for transaction costs

	Random walk	*ARMA*	*MLP*	*RNN*	*HONN*
Trading performance					
Cumulative return (%)	15.38	0.16	31.49	30.81	30.60
Annualised return (%)	9.41	0.10	19.26	18.84	18.71
Annualised volatility (%)	12.94	12.96	12.90	12.90	12.90
Information ratio	0.73	0.01	1.49	1.46	1.45
# Transactions (trades)	184	57	128	148	204
Total trading days			412		
Costs per trade (%)			0.02083		
Transaction costs (%)	3.83	1.19	2.67	3.08	4.25
With transaction costs					
Cumulative return (%)	11.55	−1.03	28.82	27.73	26.35
Annualised return (%)	7.06	−0.63	17.63	16.96	16.12
Annualised volatility (%)	12.94	12.96	12.90	12.90	12.90
Information ratio	0.55	−0.05	1.37	1.31	1.25

where, in this study, the threshold d is set at the amount of transaction costs per trade or 0.02083 per cent. Following this strategy, trades for which the forecast return is smaller than the threshold are not initiated, with potentially two beneficial consequences: forecasts of a very small change that would, however, trigger a temporary signal reversal are discarded, hopefully cutting the number of false signals and thus losses. Second, the reduction in the number of trades directly cuts transaction costs.

The results of the trading simulation with the continuation filter are shown below in Table 5 and with the subtraction of the transaction costs in Table 6.

It can be seen that the use of the confirmation filter has changed trading performances. Annualised returns of both the random walk and the ARMA forecast have decreased, whereas the performace of all neural network models has improved. The MLP outperforms all other models on annualised return, annualised volatility, information ratio and maximum drawdown, as well as CDC. The HONN model is outperformed by both the MLP and RNN this time, although marginally in terms of information ratio.

When transaction costs are taken into account, all models but the ARMA yield a positive outcome.

CONCLUSION

The motivation for this article was to check whether neural network models have remained a superior method for forecasting the EUR/USD exchange rate during the financial crisis of 2007–2009 and the increased market volatility levels associated with it. Alternative neural network architectures (MLP, RNN and

HONN) are benchmarked against a random walk and a traditional ARMA model and evaluated in terms of statistical accuracy and through a trading simulation on daily data over the period from January 2000 to February 2009, the period from August 2007 to February 2009 providing the out-of-sample testing period. Transaction costs and a continuation filter devised to reduce false signals and thus also reduce losses and transaction costs were also taken into consideration.

As a starting point, an automated combination of inputs and hidden nodes from a set range was conducted, producing a very large number of networks, which were filtered in two steps: the EV beween the actual EUR/USD rate and the forecast was used as a measure to find promising networks, which were then further filtered out using different statistical performance measures in order to find a network that was likely to produce satisfying out-of-sample results.

Despite the time and effort put in developing these different networks, and the good results achieved for the EUR/USD out-of-sample, there is no proof that the models presented are the 'best' possible solution for the period under review. A possible field where extending this research is likely to give beneficial results is model combination, as admittedly many researchers in finance have come to the conclusion that individual forecasting models are misspecified in some dimensions and that the identity of the 'best' model changes over time. In this situation, it is likely that a combination of forecasts will perform better over time than forecasts generated by any individual model that is kept constant.

Over the volatile period under review, all models employed were capable of producing positive returns, but the results of the trading simulation show that neural networks clearly outperform the benchmark naïve and ARMA strategies, both with the implementation of a confirmation filter and without. More particularly, the HONN architecture gives the overall best results in a simple trading simulation; however, with a more refined trading simulation with a confirmation filter, the MLP outperforms all other models, showing a good trading performance in a much increased volatility environment.

To conclude, our findings support the view that neural network architectures seem to have a better ability to model exchange rate returns when compared with linear models, indicating at least that there were non-linearities inherent in the EUR/USD exchange rate for the period under review. Furthermore, our results show that neural networks are still able to produce superior forecasts, with respect to both trading performance and statistical accuracy, even under very volatile market conditions like those experienced during the financial crisis of 2007–2009.

NOTES

1. The TED spread is the difference between the 3-month Eurodollar futures contract as represented by the London Interbank Offered Rate and the interest rate for 3-month US Treasury bills.
2. MATLAB programmes developed at CIBEF were used for the network training of the different networks.
3. The values 0.05 and 0.055 were chosen arbitrarily to include a reasonable number of variables.
4. For the sake of simplicity, the impact of interest earned or payed on open EUR/USD

positions is also not taken into account in this study. This impact is negilgible because of the very low EUR and USD interest rates during the review period.

REFERENCES

BIS. (2007) Foreign exchange and derivatives market activity in 2007. *Triennial Central Bank Survey*, December, Basle.

Box, G.E.P., Jenkins, G.M. and Reinsel, G.C. (1994) *Time Series Analysis: Forecasting and Control*, 3rd edn. Englewood Cliffs, NJ: Prentice Hall.

Brooks, C. (1996) Testing for non-linearity in daily sterling exchange rates. *Applied Financial Economics* 5(4): 307–317.

Chiarella, C., Peat, M. and Stevenson, M. (1994) Detecting and modelling nonlinearity in flexible exchange rate time series. *Asia Pacific Journal of Management* 11(2): 159–186.

Diebold, F.X. and Nason, J.A. (1990) Nonparametric exchange rate prediction? *Journal of International Economics* 28(3/4): 315–332.

Dunis, C.L. and Huang, X. (2002) Forecasting and trading currency volatility: An application of recurrent neural regression and model combination. *Journal of Forecasting* 21(5): 317–354.

Dunis, C.L. and Williams, M. (2002) Modelling and trading the EUR/USD exchange rate: Do neural network models perform better? *Derivatives Use, Trading and Regulation* 8(3): 211–239.

Dunis, C.L. and Williams, M. (2003) Applications of advanced regression analysis for trading and investment. In: C. Dunis, J. Laws and P. Naïm (eds.) *Applied Quantitative Methods for Tading and Investment*. Chichester, UK: John Wiley, pp. 1–40.

Dunis, C.L., Laws, J. and Sermpinis, G. (2009) The robustness of neural networks for modelling and trading the EUR/USD exchange rate at the ECB fixing. *Journal of Derivatives and Hedge Funds* 15(3): 186–205.

Dunis, C.L., Laws, J. and Sermpinis, G. (2010) Higher order and recurrent neural architectures for trading the EUR/USD exchange rate. *Quantitative Finance* 4(11): 615–629.

Elman, J.L. (1990) Finding structure in time. *Cognitive Science* 14(2): 179–211.

Fulcher, J., Zhang, M. and Xu, S. (2006) Application of higher-order neural networks to financial time-series prediction. In: J. Kamruzzaman, R.K. Begg and R.A. Sarker (eds.) *Artificial Neural Networks in Finance and Manufacturing*. Hershey, PA: Idea Group Publishing, pp. 80–108.

Hann, T.H. and Steurer, E. (1996) Much ado about nothing? Exchange rate forecasting: Neural networks vs. linear models using monthly and weekly data. *Neurocomputing* 10(4): 323–339.

Hornik, K., Stinchcombe, M. and White, H. (1989) Multilayer feedforward networks are universal function approximators. *Neural Networks* 2(5): 359–366.

Hsieh, D.A. (1989) Testing for nonlinear dependence in daily foreign exchange rates. *Journal of Business* 62(3): 329–368.

Huang, W., Lai, K.K., Nakamori, Y. and Wang, S. (2004) Forecasting foreign exchange rates with artificial neural networks: A review. *International Journal of Information Technology & Decision Making* 3(1): 145–165.

IMF. (2007) Financial Market Turbulence – Causes, Consequences and Policies. Washington: Global Financial Stability Report, October.

Kaastra, I. and Boyd, M. (1996) Designing a neural network for forecasting financial and economic times series. *Neurocomputing* 10(3): 215–236.

Knowles, A., Hussain, A., El Deredy, W., Lisboa, P.G.J. and Dunis, C.L. (2009) Higher-order neural networks with Bayesian confidence measure for prediction of EUR/USD exchange rate. In: M. Zhang (ed.) *Artificial Higher Order Neural Networks for Economics and Business*. Hershey, PA: Information Science Reference, pp. 48–59.

Krishnaswamy, C.R., Gilbert, E.W. and Pashley, M.M. (2000) Neural network applications in finance: A practical introduction. *Financial Practice and Education* 10(1): 75–84.

Kuan, C.-M. and Liu, T. (1995) Forecasting exchange rates using feedforward and recurrent neural networks. *Journal of Applied Econometrics* 10(3): 347–364.

Le Cun, Y., Denker, J.S. and Solla, S.A. (1990) Optimal brain damage. In: D.S. Touretzky (ed.) *Advances in Neural Information Processing Systems 2*. NIPS Conference, Denver, Colorado, USA, 27–30 November 1989, San Mateo, CA: Morgan Kaufmann, pp. 598–605.

Leung, M.T., Chen, A.-S. and Daouk, H. (2000) Forecasting exchange rates using general regression networks. *Computers & Operations Research* 27(11/12): 1093–1110.

Lindemann, A., Dunis, C.L. and Lisboa, P. (2005) Level estimation, classification and probability distribution architectures for trading the EUR/USD exchange rate. *Neural Computing and Applications* 14(3): 256–271.

Lisboa, P.J.G. and Vellido, A. (2000) Business applications of neural networks: Preface. In: P.J.G. Lisboa, B. Edisbury and A. Vellido (eds.) *Business Applications of Neural Networks – The State-of-the-Art of Real-World Applications, Progress in Neural Processing*, Vol. 13. Singapore: World Scientific, pp. vii–xix.

Meese, R.A. and Rogoff, K. (1983a) Empirical exchange rate models of the seventies: Do they fit out-of-sample? *Journal of International Economics* 14(1/2): 3–24.

Meese, R.A. and Rogoff, K. (1983b) The out-of-sample failure of empirical exchange rate models: Sampling error or misspecification? In: J. Frankel (ed.) *Exchange Rates and International Economics*. Chicago, IL: University of Chicago Press, pp. 67–105.

Meese, R.A. and Rose, A.K. (1991) An empirical assessment of non-linearities in models of exchange rate determination. *Review of Economic Studies* 58(3): 603–619.

Mehta, M. (1995) Foreign exchange markets. In: A.-P.N. Refenes (ed.) *Neural Networks in the Capital Markets*. Chichester, UK: John Wiley & Sons, pp. 177–198.

Pindyck, R.S. and Rubinfeld, D.L. (1991) *Econometric Models and Economic Forecast*, 3rd edn. New York: McGraw-Hill.

Refenes, A.-P.N. (1993) Constructive learning and its application to currency exchange rate forecasting. In: R.R. Trippi and E. Turban (eds.) *Neural Networks in Finance and Investing: Using Artificial Intelligence to Improve Real-World Performance*. Chicago, IL: Probus Publishing Company.

Refenes, A.-P.N., Azema-Barac, M., Chen, L. and Karoussos, S.A. (1993) Currency exchange rate prediction and neural network design strategies. *Neural Computing and Applications* 1(1): 46–58.

Refenes, A.-P.N. and Zaidi, A. (1995) Managing exchange rate prediction strategies with neural networks. In: A.-P. Refenes (ed.) *Neural Networks in the Capital Markets*. Chichester, UK: John Wiley & Sons, pp. 213–219.

Reuters. (2009) Three top economists agree 2009 worst financial crisis since great depression – Risks increase if right steps are not taken. February, http://www.reuters.com/article/pressRelease/idUS193520+27-Feb-2009+BW20090227, accessed 3 August 2009.

Tenti, P. (1996) Forecasting foreign exchange rates using recurrent neural networks. *Applied Artificial Intelligence* 10(6): 567–581.

The Economist. (2009) The other-worldly philosophers. Vol. 392, No. 8640, 18 July, pp. 70–72.

Tyree, E.W. and Long, J.A. (1995) Forecasting Currency Exchange Rates: Neural Networks and the Random Walk Model. New York: City University, Proceedings of the Third International Conference on Artificial Intelligence Applications.

Vellido, A., Lisboa, P.J.G. and Vaughan, J. (1999) Neural networks in business: A survey of applications (1992–1998). *Expert Systems with Applications* 17(1): 51–70.

Weigend, A.S., Huberman, B.A. and Rumelhart, D.E. (1992) Predicting sunspots and exchange rates with connectionist networks. In: M. Casdagli and S. Eubank (eds.) *Nonlinear Modeling and Forecasting*. Redwood City, CA: Addison-Wesley, pp. 395–432.

Zhang, G. and Hu, M.Y. (1998) Neural network forecasting of the British Pound/US dollar exchange rate. *Omega – The International Journal of Management Science* 26(4): 495–506.

Zhang, G., Patuwo, B.E. and Hu, M.Y. (1998) Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting* 14(1): 35–62.

Zhang, M., Xu, S. and Fulcher, J. (2002) Neuron-adaptive higher order neural-network models for automated financial data modeling. *IEEE Transactions on Neural Networks* 13(1): 188–204.

APPENDIX

Table A1: Explanatory variables

Number	*Variable*	*Mnemonics*
1	FTSE 100 – PRICE INDEX	FTSE100
2	DAX 30 PERFORMANCE – PRICE INDEX	DAXINDX
3	S&P 500 COMPOSITE – PRICE INDEX	S&PCOMP
4	NIKKEI 225 STOCK AVERAGE – PRICEINDEX	JAPDOWA
5	FRANCE CAC 40 – PRICE INDEX	FRCAC40
6	MILAN MIB 30 – PRICE INDEX	ITMIB30
7	DJ EURO STOXX 50 – PRICEINDEX	DJES50I
8	US EURO-$ 3 MTH (FT/ICAP/TR) – MIDDLE RATE	ECUS$3M
9	JAPAN EURO-YEN 3 MTH (FT/ICAP/TR) – MIDDLE RATE	ECJAP3M
10	GERMANYEU-MARK 3M (FT/ICAP/TR) – MIDDLE RATE	ECWGM3M
11	FRANCE EU-FRANC 3M (FT/ICAP/TR) – MIDDLE RATE	ECFFR3M
12	UK EURO-3M (FT/ICAP/TR) – MIDDLE RATE	ECUK£3M
13	ITALY EURO-LIRE 3M (FT/ICAP/TR) – MIDDLE RATE	ECITL3M
14	JAPAN BENCHMARK BOND –RYLD.10 YR (DS) – RED. YIELD	JPBRYLD
15	GERMANY BENCHMARK BOND 10 YR (DS) – RED. YIELD	BDBRYLD
16	FRANCE BENCHMARK BOND 10 YR (DS) – RED. YIELD	FRBRYLD
17	UK BENCHMARK BOND 10 YR (DS) – RED. YIELD	UKMBRYD
18	US TREAS.BENCHMARK BOND 10 YR (DS) – RED. YIELD	USBD10Y
19	ITALY BENCHMARK BOND 10 YR (DS) – RED. YIELD	ITBRYLD
20	JAPANESE YEN TO US $ (WMR) – EXCHANGE RATE	JAPAYE$
21	US $ TO UK £ (WMR) – EXCHANGE RATE	USDOLLR
22	US $ TO EURO (WMR&DS) – EXCHANGE RATE	USEURSP
23	Crude Oil-Brent Cur. Month FOB U$/BBL	OILBREN
24	Gold Bullion LBM U$/Troy Ounce	GOLDBLN
25	CRB Continuous Commodity Index – PRICE INDEX	NYFECRB

Source: Dunis and Williams, 2003.

Table A2: ARMA models[a] performance compared

	ARMA (2, 2)	*ARMA (5, 5)*	*ARMA (6, 6)*	*ARMA (7, 7)*	*ARMA (9, 9)*	*ARMA (10, 10)*
Trading performance						
Cumulative profit (%)	105.73	84.11	119.04	115.85	107.52	98.96
Annualised profit (%)	13.49	10.75	15.22	14.82	13.77	12.68
Annualised volatility (%)	9.28	9.30	9.27	9.28	9.29	9.28
Information ratio	1.45	1.16	1.64	1.60	1.48	1.37
Maximum drawdown (%)	−9.26	−9.21	−11.52	−10.50	−11.49	−10.57
Profits *T*-statistics	64.61	51.34	72.86	70.89	65.75	60.57
Total trading days	1975	1972	1971	1970	1968	1967
# Winning periods	1055	1026	1068	1068	1047	1054
# Losing periods	920	946	903	902	921	913
Winning periods (%) (=CDC)	53.42	52.03	54.19	54.21	53.20	53.58
Maximum gain in winning periods (%)	3.38	2.24	2.33	3.38	3.38	3.38
Maximum loss in losing periods (%)	−1.88	−3.38	−3.38	−2.10	−2.33	−2.24
Average gain in winning periods (%)	0.47	0.47	0.47	0.47	0.47	0.46
Average loss in losing periods (%)	−0.42	−0.42	−0.42	−0.42	−0.42	−0.43
Average gain/loss ratio	1.11	1.12	1.11	1.10	1.12	1.09

# Periods daily returns rise	996	994	994	993	993	993
# Winning rise periods	538	647	758	677	634	661
Winning rise periods (%)	54.02	65.09	76.26	68.18	63.85	66.57
# Periods daily returns fall	979	978	977	977	975	974
# Winning fall periods	517	379	310	391	413	393
Winning fall periods (%)	52.81	38.75	31.73	40.02	42.36	40.35
# Transactions (trades)	759	1158	600	683	979	472
Winning trades (%)	50.99	49.74	52.67	52.86	51.07	53.39
Losing trades (%)	49.01	50.26	47.33	47.14	48.93	46.61
Statistical performance						
Mean absolute error	0.0045	0.0045	0.0044	0.0044	0.0044	0.0044
Mean absolute percentage error (%)	116.88	114.66	112.12	118.38	116.45	119.86
Root mean squared error	0.0058	0.0058	0.0058	0.0058	0.0058	0.0058
Theil's inequality coefficient	0.8795	0.8920	0.9045	0.8660	0.8721	0.8535

[a]The ARMA models ARMA(5, 5), ARMA(6, 6), ARMA(7, 7) ARMA(9, 9) and ARMA(10, 10) are restricted, that is, not all terms up to (p, q) are included in the actual model. The exact specification of the models can be inspected in detail in the Eviews file 'Arma_Models.wf1'.

Table A3: Restricted ARMA(6, 6) model

Dependent variable: USEURSP

Sample (adjusted): 1/11/2000 7/31/2007

White Heteroskedasticity-consistent standard errors and covariance

	Coefficient	*Standard Error*	*t-statistic*	*Probability*
C	0.000289	7.98E-05	3.618189	0.0003
AR(3)	0.384951	0.097475	3.949213	0.0001
AR(4)	0.307724	0.085192	3.612123	0.0003
AR(5)	0.652163	0.084023	7.761678	0.0000
AR(6)	−0.372798	0.101367	−3.677714	0.0002
MA(3)	−0.386778	0.102149	−3.786406	0.0002
MA(4)	−0.291541	0.088721	−3.286026	0.0010
MA(5)	−0.652437	0.087727	−7.437166	0.0000
MA(6)	0.345683	0.107152	3.226101	0.0013

R^2	0.009394	Mean dependent variable	0.000163
Adjusted R^2	0.005355	SD dependent variable	0.005871
SE of regression	0.005855	Akaike information criterion	−7.438445
Sum squared residual	0.067262	Schwarz criterion	−7.412937
Log likelihood	7339.587	Hannan-Quinn criterion	−7.429072
F-statistic	2.325759	Durbin-Watson statistics	1.989575
Probability (*F*-statistic)	0.017481		

Table A4: Ramsey RESET test for restricted ARMA(6, 6) model

Ramsey RESET test:

F-statistic	0.836649	Prob *F*(2 1960)	0.4333

Table A5: ARMA(6, 6) correlogram of residual

Sample: 1/11/2000 7/31/2007

Autocorrelation	*Partial correlation*		*AC*	*PAC*	*Q-statistics*	*Probability*
| |	| |	8	0.011	0.011	1.0769	
| |	| |	9	−0.018	−0.018	1.6938	0.193
| |	| |	10	−0.011	−0.011	1.9540	0.376
| |	| |	11	−0.013	−0.013	2.2914	0.514
| |	| |	12	0.013	0.012	2.6200	0.623
| |	| |	13	0.005	0.004	2.6652	0.751
| |	| |	14	0.011	0.011	2.9213	0.819
| |	| |	15	−0.012	−0.012	3.2155	0.864
| |	| |	16	0.025	0.025	4.4190	0.817
| |	| |	17	0.004	0.004	4.4459	0.880
| |	| |	18	0.011	0.010	4.6752	0.912
| |	| |	19	0.031	0.032	6.6494	0.827
| |	| |	20	0.005	0.004	6.6975	0.877

Table A6: Variable selections

Pool 1	*Pool 2*	*Pool 3*
Best lags (19)	*Highest cross-correlation (18)*	*Highest cross-correlation (14)*
USEURSP(−12)	FSTE100(−4)	FSTE100(−4)
FSTE100(−4)	SPCOMP(−17)	SPCOMP(−17)
DAXINDX(−10)	SPCOMP(−12)	SPCOMP(−12)
SPCOMP(−17)	SPCOMP(−11)	SPCOMP(−11)
JAPDOWA(−1)	SPCOMP(−2)	SPCOMP(−2)
FRCAC40(−17)	SPCOMP(−10)	JAPDOWA(−1)
ITMIB(−17)	JAPDOWA(−1)	FRCAC40(−17)
DJES50I(−17)	FRCAC40(−17)	US_YC(−15)
US_YC(−15)	DJES50I(−17)	GER_YC(−1)
JAP_YC(−8)	US_YC(−15)	FR_YC(−7)
GER_YC(−1)	GER_YC(−1)	JAPAYE$(−16)
FR_YC(−7)	GER_YC(−3)	JAPAYE$(−5)
UK_YC(−6)	GER_YC(−12)	JAPAYE$(−7)
IT_YC(−10)	FR_YC(−7)	NYFECRB(−9)
JAPAYE$(−16)	JAPAYE$(−16)	
USDOLLR(−10)	JAPAYE$(−5)	
OILBREN(−5)	JAPAYE$(−7)	
GOLDBLN(−14)	NYFECRB(−9)	
NYFECRB(−9)		

Table A7: Network training parameters

Parameter	*MLP*	*RNN*	*HONN*
Number of cycles/epochs	5	5	5
Number of training blocks	10	15	10
Learning rate	0.001	0.001	0.001
Momentum rate	0.001	0.001	0.001

Table A8: Weight matrix

	SPCOMP (−12)	*SPCOMP (−10)*	*FRCAC40 (−17)*	*FR_YC (−7)*	*JAPAYE$ (−16)*	*JAPAYE$ (−5)*	*JAPAYE$ (−7)*	*Bias*
$h_T^{[1]}$	−0.7085	−0.2968	−0.1763	−0.2895	−0.2532	−0.2246	0.2328	−0.0076
$h_T^{[2]}$	−0.0872	−0.3562	−0.5407	−0.2520	−0.5351	−0.5678	0.1006	−0.0020
$h_T^{[3]}$	−0.1115	−0.2557	−0.0670	−0.4819	−0.1592	−0.1330	0.1941	0.0208

Table A9: MLP weight matrix optimisation in-sample and out-of-sample performance

	MLP weight matrix optimisation (in-sample)	*MLP weight matrix optimisation (out-of-sample)*
Trading performance		
Cumulative return (%)	49.33	−21.61
Annualised return (%)	6.35	−13.22
Annualised volatility (%)	9.31	12.93
Information ratio	0.68	−1.02
Maximum drawdown (%)	−15.23	−31.45
Profits *T*-statistics	30.19	−20.74
Total trading days	1957	412
# Winning periods	1003	201
# Losing periods	954	211
Winning periods (%) (=CDC)	51.25	48.79
Statistical performance		
Mean absolute error	0.0045	0.0057
Mean absolute percentage error (%)	101.44	105.79
Root mean squared error	0.0059	0.0082
Theil's inequality coefficient	0.9696	0.9479

Table A10: MLP, RNN and HONN in-sample performance

	MLP	*RNN*	*HONN*
Trading performance			
Cumulative return (%)	77.33	49.80	99.93
Annualised return (%)	9.96	6.41	12.87
Annualised Volatility (%)	9.30	9.31	9.28
Information ratio	1.07	0.69	1.39
Maximum drawdown (%)	−21.79	−14.15	−17.71
Profits *T*-statistics	47.38	30.47	61.32
Total trading days	1957	1957	1957
# Winning periods	1038	1034	1053
# Losing periods	919	923	904
Winning periods (%) (=CDC)	53.04	52.84	53.81
Maximum gain in winning periods (%)	2.33	2.33	3.38
Maximum loss in losing periods (%)	−3.38	−3.38	−2.07
Average gain in winning periods (%)	0.46	0.45	0.46
Average loss in losing periods (%)	−0.43	−0.45	−0.43
Average gain/loss ratio	1.06	1.00	1.08
# Periods market returns rise	991	991	991
# Winning rise periods	679	530	533
Winning rise periods (%)	68.52	53.48	53.78
# Periods market returns fall	966	966	966
# Winning fall periods	359	504	520
Winning fall periods (%)	37.16	52.17	53.83
# Transactions (trades)	815	760	994
Winning trades (%)	46.13	51.05	47.08
Losing trades (%)	53.87	48.95	52.92
Statistical performance			
Mean absolute error	0.0045	0.0045	0.0045
Mean absolute percentage error (%)	106.83	101.08	99.76
Root mean squared error	0.0059	0.0059	0.0059
Theil's inequality coefficient	0.8637	0.9191	0.9827
Training correlation	0.0744	0.0881	0.1459
Test correlation	0.0602	0.0426	0.0459

Table A11: MLP model specifications

————Configuration 2 of 11————

Configuration=

FSTE100(−4)
SPCOMP(−12)
SPCOMP(−11)
SPCOMP(−2)
JAPDOWA(−1)
FRCAC40(−17)
GER_YC(−1)
JAPAYE$(−16)
JAPAYE$(−5)

Hidden nodes: 6 Net weights:

Weights=

0.1108	0.1049	0.4755	−0.0202	0.3702	−0.1624
−0.2613	−0.2528	0.0896	0.0326	−0.2048	−0.0607
−0.8166	0.2895	0.0525	0.1718	−0.5283	0.3005
0.0837	−0.2248	0.4371	0.0498	0.1116	−0.3766
0.0345	0.0017	0.5119	0.0849	0.3047	−0.6327
−0.4068	−0.7158	0.1906	−0.4699	0.4827	0.3879
−0.1293	−0.1151	−0.0844	−0.0642	−0.1308	0.8208
0.2453	−0.0419	0.5115	0.0167	0.3865	0.2347
0.0438	−0.2680	−0.3157	−0.1867	0.4189	−0.0118

Net bias: bias =

0.0070	−0.0044	−0.0216	−0.0112	0.0272	0.0016

CDC: 53.04%

Average correlation in test and training period: 0.067283

Table A12: RNN model specifications

```
————Configuration 2 of 13————

Configuration=

FSTE100(−4)
SPCOMP(−17)
SPCOMP(−12)
SPCOMP(−11)
SPCOMP(−2)
FRCAC40(−17)
US_YC(−15)
FR_YC(−7)
JAPAYE$(−16)
JAPAYE$(−5)
JAPAYE$(−7)

Input nodes: 11
Net weights:

Weights=

  Columns 1 through 9
    0.1493   −0.0644    0.3203   0.0257    0.0239    0.0666    0.4116    0.3549    0.0207
   −0.1825    0.3946   −0.1520   0.2552    0.3250    0.0326    0.2085   −0.0538   −0.3645
    0.0871   −0.3069   −0.0746   0.0876   −0.4119   −0.3677   −0.2609   −0.2713   −0.0236
   −0.0703    0.1514    0.0973   0.4380   −0.0718    0.0165   −0.3047   −0.2505    0.0198
   −0.2479    0.3953   −0.3414   0.1122    0.4585   −0.0262    0.0693   −0.2021    0.2217

  Columns 10 through 11
    0.4029   −0.3510
    0.2503   −0.3914
   −0.3465    0.3398
    0.1989    0.2485
   −0.1741    0.1513

Net bias:

Bias=

    1.5419
    0.7702
   −0.0245
   −0.7859
   −1.5484

Average correlation in test and training period: 0.065348
CDC: 52.84%
```

Table A13: HONN model specifications

```
————Configuration 26 of 52————

Configuration=

SPCOMP(−11)
JAPDOWA(−1)
GER_YC(−1)
JAPAYE$(−16)
JAPAYE$(−7)

Input nodes: 15
Net weights:

Weights=

  Columns 1 through 10
    0.2120       0.0666   −0.0185   −0.4401   −0.2442   0.0643   −0.3460   0.0421   0.1586
    0.1063
  Columns 11 through 15
    0.4908     −0.1702    0.1294    0.2497    0.2387
Net bias:
Bias=
    0

Average correlation in test and training period: 0.095918
CDC: 53.81%
```

Table A14: Statistical forecasting accuracy measures

Performance measure	*Description*	
Mean absolute error (MAE)	$MAE = \frac{1}{T}\sum_{t=1}^{T}\lvert\hat{y}_t - y_t\rvert$	(11)
Mean absolute percentage error (MAPE)	$MAPE = \frac{100}{T}\sum_{t=1}^{T}\left\lvert\frac{\hat{y}_t - y_t}{y_t}\right\rvert$	(12)
Root mean squared error (RMSE)	$RMSE = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(\hat{y}_t - y_t)^2}$	(13)
Theil's inequality coefficient (Theil's U)	$U = \frac{\sqrt{\frac{1}{T}\sum_{t=1}^{T}(\hat{y}_t - y_t)^2}}{\sqrt{\frac{1}{T}\sum_{t=1}^{T}(\hat{y}_t)^2}+\sqrt{\frac{1}{T}\sum_{t=1}^{T}(y_t)^2}}$	(14)
Correct directional change (CDC)	$CDC = \frac{100}{N}\sum_{t=1}^{N} D_t$ where D_t=1 if $\hat{y}_t \times y_t > 0$, else D_t=0	(15)

Source: (Dunis and Williams, 2003).

Table A15: Trading performance measures

Performance measure	*Description*	
Cumulative return	$R^C = \sum_{t=1}^{N} R_t$	(16)
Annualised return	$R^A = 252 \times \frac{1}{N} \sum_{t=1}^{N} R_t$	(17)
Annualised volatility	$\sigma^A = \sqrt{252} \times \sqrt{\frac{1}{N-1} \sum_{t=1}^{T} \left(R_t - \bar{R}\right)^2}$	(18)
Information ratio	$SR = \frac{R^A}{\sigma^A}$	(19)
Maximum drawdown	Maximum negative value of $\sum R_T$ $MD = \min_{t=1,\ldots,N} \left(R_t^C - \max_{t=1,\ldots,t} \left(R_i^C \right) \right)$	(20)
Profits *t*-statistics	$t - \text{statistics} = \sqrt{N} \times \frac{R^A}{\sigma^A}$	(21)
Total trading days	$TTD =$ Number of all $R_t s$	(22)
# Winning periods	$WP = \sum_{t=1}^{N} F_t$ where F_t=1 if $R_t > 0$, else F_t=0	(23)
# Losing periods	$LP = \sum_{t=1}^{N} G_t$ where G_t=1 if $R_t \leqslant 0$ else G_t=0	(24)
Percentage of winning periods (=CDC)	$PWP = \frac{WP}{TTD}$	(25)
Maximum gain in winning periods	$\text{MaxGain} = \max_{t=1,\ldots,T} R_t$	(26)
Maximum loss in losing periods	$\text{MinGain} = \min_{t=1,\ldots,T} R_t$	(27)
Average gain in winning periods	$AG = \frac{\text{Sum of all } R_t > 0}{WP}$	(28)
Average loss in losing periods	$AL = \frac{\text{Sum of all } R_t < 0}{LP}$	(29)
Average gain/loss ratio	$AGLR = \frac{AG}{AL}$	(30)
# Periods market returns rise	$MRR = \sum_{t=1}^{N} H_t$ where H_t=1 if $y_t > 0$, else H_t=0	(31)
# Winning rise periods	$WRP = \sum_{t=1}^{N} K_t$ where K_t=1 if $R_t > 0$ and $y_t > 0$, else K_t=0	(32)
Percentage of winning rise periods	$PWRP = 100 \times \frac{WRP}{MRR}$	(33)
# Periods market returns fall	$MRF = \sum_{t=1}^{N} L_t$ where L_t=1 if $Y_t < 0$, else L_t=0	(34)

Table A15 *continued*

Performance measure	*Description*	
# Winning fall periods	$WFP = \sum_{t=1}^{N} P_t$ where P_t=1 if $R_t > 0$ and $y_t < 0$, else P_t=0	(35)
Percentage of winning fall periods	$PWFP = 100\times \frac{WRP}{MRF}$	(36)
# transactions (trades)	$NT = \sum_{t=1}^{N} Q_t$ where Q_t=1 if trading signal$_t \neq$ trading signal$_{t-1}$, else Q_t=0	(37)
Percentage of winning trades	$WT = 100\times \frac{\sum_{t=1}^{N} S_t}{NT}$ where S_t=1 if transaction profit$_t > 0$, else S_t=0	(38)
Percentage of losing trades	$LT = 100\times \frac{\sum_{t=1}^{N} U_t}{NT}$ where L_t=1 if transaction profit$_t \leqslant 0$, else L_t=0	(39)

Source: (Dunis and Williams, 2003).

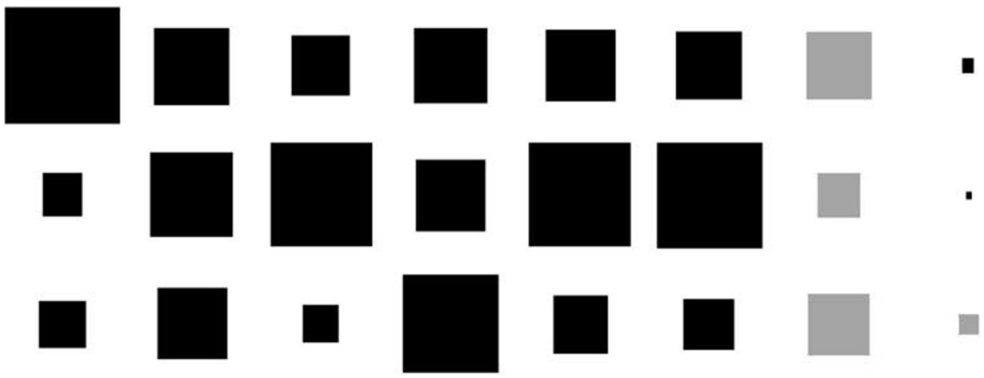

Figure A1: Hinton graph.