Modelling and Trading the EUR/USD Exchange Rate at the ECB Fixing

by

Christian L. Dunis[*]
Jason Laws*
Georgios Sermpinis[*]
([*]Liverpool Business School, CIBEF
Liverpool John Moores University)

June 2008

Abstract

The motivation for this paper is to investigate the use of alternative novel neural network architectures when applied to the task of forecasting and trading the Euro/Dollar (EUR/USD) exchange rate using the European Central Bank (ECB) fixing series with only autoregressive terms as inputs. This is done by benchmarking four different neural network designs representing a Higher Order Neural Network (HONN), a Psi Sigma Network and a Recurrent Network (RNN) with the classic Multilayer Percepton (MLP) and some traditional techniques, either statistical such as a an autoregressive moving average model (ARMA), or technical such as a moving average concergence/divergence model (MACD), plus a naïve strategy. More specifically, the trading performance of all models is investigated in a forecast and trading simulation on the EUR/USD ECB fixing time series over the period 1999-2007 using the last one and half year for out-of-sample testing, a original feature of this paper. We use the EUR/USD daily fixing by the ECB as many financial institutions are ready to trade at this level and it is therefore possible to leave orders with a bank for business to be transacted on that basis.

As it turns out, the MLP does remarkably well and outperforms all other models in a simple trading simulation exercise. However, when more sophisticated trading strategies using confirmation filters and leverage are applied, the HONN network produces better results and outperforms all other neural network and traditional statistical models in terms of annualised return.

Keywords

Confirmation filters, Higher Order Neural Networks, Psi Sigma Networks, Recurrent Networks, Leverage, Multi-Layer Perceptron Networks, Quantitative Trading Strategies.

Christian Dunis *is Professor of Banking and Finance at Liverpool Business School and Director of the Centre for International Banking, Economics and Finance (CIBEF) at Liverpool John Moores University (E-mail: cdunis@tiscali.co.uk).*

Jason Laws *is Reader of Finance at Liverpool Business School and a member of CIBEF (E-mail: J.Laws@ljmu.ac.uk).*

Georgios Sermpinis *is an Associate Researcher with CIBEF (E-mail: G.Sermpinis@2005.ljmu.ac.uk) and currently working on his PhD thesis at Liverpool Business School.*

CIBEF *– Centre for International Banking, Economics and Finance, JMU, John Foster Building, 98 Mount Pleasant, Liverpool L3 5UZ.*

1. INTRODUCTION

Neural networks are an emergent technology with an increasing number of real-world applications including Finance (Lisboa *et al.* (2000)). However their numerous limitations, are often creating scepticism about their use among practitioners. Moreover, the selection of each network's inputs is based more on trial and error and the practitioner's market knowledge rather than on some formal statistical procedure.

The motivation for this paper is to investigate the use of several new neural networks techniques that try to overcome these limitations using as inputs autoregressive terms. This is done by benchmarking four different neural network architectures representing a Multilayer Percepton (MLP), a Higher Order Neural Network (HONN), a Psi Sgima network and a Recurrent Neural Network (RNN). Their trading performance on the Euro/Dollar (EUR/USD) time series is investigated and is compared with some traditional statistical or technical methods such as an autoregressive moving average (ARMA) model and a moving average convergence/divergence (MACD) model, and a naïve strategy.

Moreover, our conclusions can supplement those of Dunis and Williams (2002, 2003), Lindemann *et al.* (2004) and Dunis *et al.* (2008) who conduct a forecasting competition over the Euro/Dollar (EUR/USD) London closing time series for a period of ten years (October 1994 to July 2001) using about the same networks but with multivariate series as inputs.

The main reason behind our decision to use the EUR/USD daily fixing by the ECB is that is possible to leave orders with a bank and trade on that basis. It is therefore a tradable quantity while for example the London closing time series used in the articles mentioned above are not, as there maybe slippage when we come to transact. Moreover, to the best of our knowledge the use of ECB fixing series is an original feature of this paper.

As it turns out, the MLP demonstrates a remarkable performance and outperforms all other models in a simple trading simulation exercise. On the other hand, when more sophisticated trading strategies using confirmation filters and leverage are applied, HONNs outperform all models in terms of annualised return. Our conclusion corroborates those of Lindemann *et al.* (2004) and Dunis *et al.* (2008) where HONNs also demonstrate a forecasting superiority on the EUR/USD series over more traditional techniques such as a MACD and a naïve strategy. However, the RNN which in the previous articles performed remarkably well, show a poor performance in this research: this may be due to their inability to provide good enough results when only autoregressive terms are used as inputs.

The rest of the paper is organised as follows. In section 2, we present the literature relevant to the Recurrent Networks, the Higher Order Neural Networks and their variant Psi Sigma. Section 3 describes the dataset used for this research and its characteristics. An overview of the different neural network models and statistical techniques is given in section 4. Section 5 gives the empirical results of all the models considered and investigates the possibility of improving their performance with the application of more sophisticated trading strategies. Section 6 provides some concluding remarks.

2. LITERATURE REVIEW

The motivation for this paper is to apply some of the most promising new neural networks architectures which have been developed recently with the purpose to overcome the numerous limitations of the more classic neural architectures and to assess whether they can achieve a higher performance in a trading simulation using only autoregressive series as inputs.

RNNs have an activation feedback which embodies short-term memory allowing them to learn extremely complex temporal patterns. Their superiority against feedfoward networks when performing nonlinear time series prediction is well documented in Connor and Atlas (1993) and Adam *et al.* (1994). In financial applications, Kamijo and Tanigawa (1990) applied them successfully to the recognition of stock patterns of the Tokyo stock exchange while Tenti (1996) achieved remarkable good results using RNNs to forecast the exchange rate of the Deutsche Mark. Tino *et al.* (2001) use them to trade successfully the volatility of the DAX and the FTSE 100 using straddles while Dunis and Huang (2002), using continuous implied volatility data from the currency options market, obtain remarkable good results for their GBP/USD and USD/JPY exchange rate volatility trading simulation.

HONNs were first introduced by introduced by Giles and Maxwell (1987) as a fast learning network with increased learning capabilities. Although their function approximation superiority over the more traditional architectures is well documented in the literature (see among others Redding *et al*. (1993), Kosmatopoulos *et al*. (1995) and Psaltis *et al*. (1988), their use in finance so far has been limited. This has changed when scientists started to investigate not only the benefits of Neural Networks (NNs) against the more traditional statistical techniques but also the differences between the different NNs model architectures. Practical applications have now verified the theoretical advantages of HONNs by demonstrating their superior forecasting ability and put them in the front line of research in financial forecasting. For example Dunis *et al.* (2006b) use them to forecast successfully the gasoline crack spread while Fultcher *et al* (2006) apply HONNs to forecast the AUD/USD exchange rate, achieving a 90% accuracy. However, Dunis *et al.* (2006a) show that, in the case of the futures spreads and for the period under review, the MLPs performed better compared with HONNs and recurrent neural networks.

Psi Sigma networks were first introduced as an architecture capable of capturing higher order correlations within the data while avoiding some of the HONNs limitations such as the combinatorial increase in weight numbers. Shin and Ghosh (1991) and Ghosh and Shin (1992) demonstrate these benefits and present empirical evidence on their forecasting ability. For financial applications, Ghazali *et al.* (2006) compare them with HONNs on the IBM common stock closing price and the US 10-year government bond series and prove their forecasting superiority while, in a similar paper, Hussain *et al.* (2006) present satisfactory results of the Psi Sigma forecasting power on the EUR/USD, the EUR/GBP and the EUR/JPY exchange rates using univariate series as inputs in their networks. Moreover, Dunis *et al.* (2008), who also study the EUR/USD series for a period of 10 years, demonstrate that when multivariate series are used as inputs the Psi Sigma, HONNs, RNN and MLP networks have a similar forecasting power.

3. THE EUR/USD EXCHANGE RATE AND RELATED FINANCIAL DATA

The European Central Bank (ECB) publishes a daily fixing for selected EUR exchange rates: these reference mid-rates are based on a daily concertation procedure between central banks within and outside the European System of Central Banks, which normally takes place at 2.15 p.m. ECB time. The reference exchange rates are published both by electronic market information providers and on the ECB's website shortly after the concertation procedure has been completed. Although only a reference rate, many financial institutions are ready to trade at the EUR fixing and it is therefore possible to leave orders with a bank for business to be transacted at this level.

The ECB daily fixing of the EUR/USD is therefore a tradable level which makes using it a more realistic alternative to, say, London closing prices and this is the series that we investigate in this paper[1].

We examined the ECB daily fixing of the EUR/USD since its first trading day on 4 January 1999 until 31 December 2007. The data period is partioned as follows.

Name of period		Beginning	End
Total dataset	2304	4 January 1999	31 December 2007
Training dataset	1921	4 January 1999	30 June 2006
Out-of-sample dataset [Validation set]	383	3 July 2006	31 December 2007

Table 1: The EUR/USD dataset

The graph below shows the total dataset for the EUR/USD and its upward trend since early 2006.

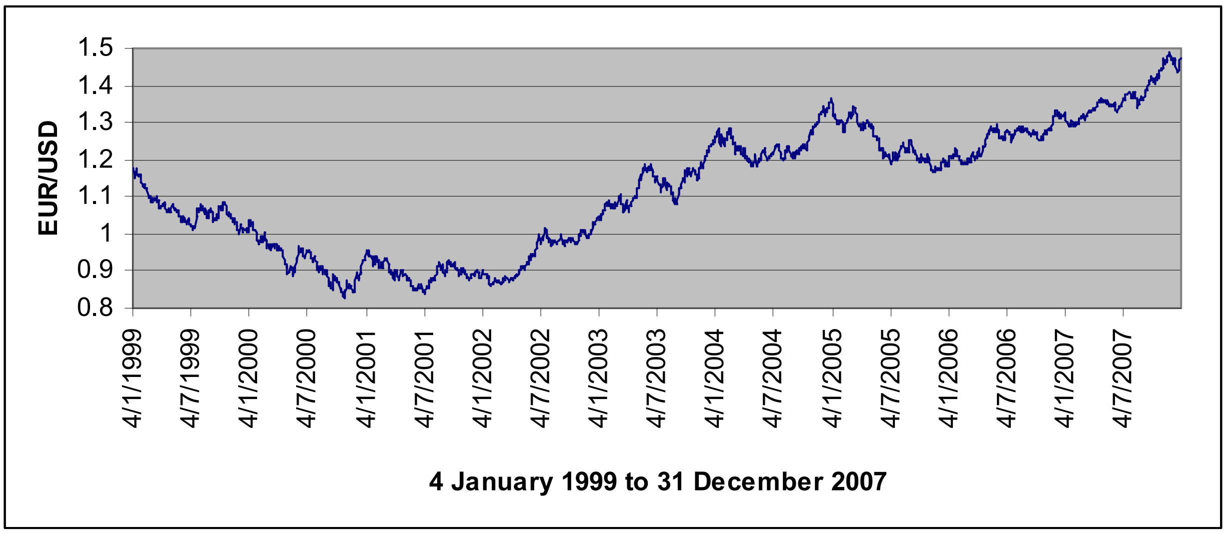


Fig. 1: EUR/USD Frankfurt daily fixing prices (total dataset)

The observed EUR/USD time series is non-normal (Jarque-Bera[2] statistics confirmed this at the 99% confidence interval) containing slight skewness and high kurtosis. It is

[1] EUR/USD is quoted as the number of USD per Euro: for example, a value of 1.2657 is USD1.2657 per 1 Euro. We examine the EUR/USD since its first trading day on 4 January 1999, and until 31 December 2007.

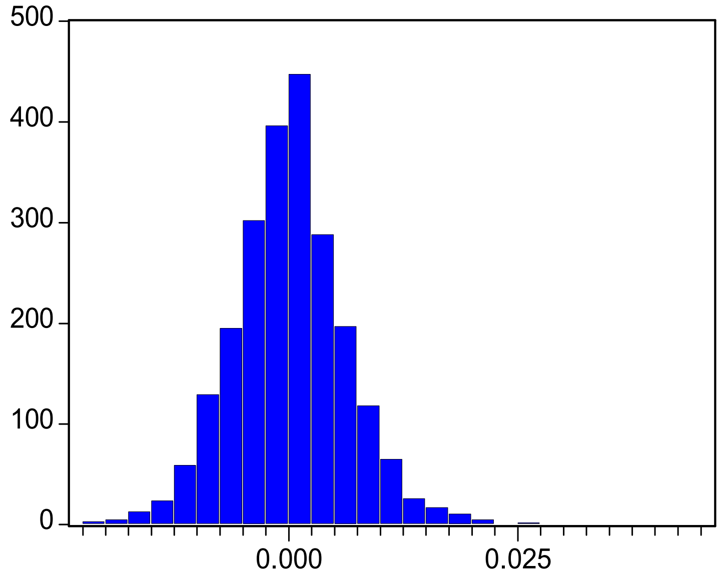
[2] For full discussion of the statistics refer to Bera and Jarque (1981).

also nonstationary and hence we decided to transform the EUR/USD series into stationary daily series of rates of return[3] using the formula:

$$R_t = \left(\frac{P_t}{P_{t-1}} \right) - 1 \qquad [1]$$

Where R_t is the rate of return and P_t is the price level at time t.

The summary statistics of the EUR/USD returns series reveal a slight skewness and high kurtosis. The Jarque-Bera statistic confirms again that the EUR/USD series is non-normal at the 99% confidence interval.



Series: RETURNS
Sample 1 2303
Observations 2303

Mean	0.000115
Median	7.60e-05
Maximum	0.042938
Minimum	-0.022269
Std. Dev.	0.006177
Skewness	0.286813
Kurtosis	4.622358
Jarque-Bera	284.1414
Probability	0.000000

Fig. 2: EUR/USD returns summary statistics (total dataset)

As inputs to our networks and based on the autocorrelation function and some ARMA experiments we selected a set of autoregressive and moving average terms of the EUR/USD exchange rate returns and the 1-day Riskmetrics volatility series.

Number	**Variable**	**Lag**
1	EUR/USD exchange rate return	1
2	EUR/USD exchange rate return	2
3	EUR/USD exchange rate return	3
4	EUR/USD exchange rate return	7
5	EUR/USD exchange rate return	11
6	EUR/USD exchange rate return	12
7	EUR/USD exchange rate return	14
8	EUR/USD exchange rate return	15
9	EUR/USD exchange rate return	16
10	Moving Average of the EUR/USD exchange rate return	15
11	Moving Average of the EUR/USD exchange rate return	20
12	1-day Riskmetrics Volatility	1

Table 2: Explanatory variables

[3] Confirmation of its stationary property is obtained at the 1% significance level by both the Augmented Dickey Fuller (ADF) and Phillips-Perron (PP) test statistics.

In order to train our neural networks we further divided our dataset as follows:

Name of period	Trading days	Beginning	End
Total data set	2304	4 January 1999	31 December 2007
Training data set	1537	4 January 1999	31 December 2004
Test data set	384	3 January 2005	30 June 2006
Out-of-sample data set [Validation set]	383	3 July 2006	31 December 2007

Table 3: The neural networks datasets

4. FORECASTING MODELS

4.1 Benchmark Models

In this paper, we benchmark our neural network models with 3 traditional strategies, namely an autoregressive moving average model (ARMA), a moving average convergence/divergence technical model (MACD) and a naïve strategy.

4.1.1 Naïve strategy

The naïve strategy simply takes the most recent period change as the best prediction of the future change, i.e. a simple random walk. The model is defined by:

$$\hat{Y}_{t+1} = Y_t \qquad [2]$$

where Y_t is the actual rate of return at period t

$\hat{Y}_{t+1}$ is the forecast rate of return for the next period

The performance of the strategy is evaluated in terms of trading performance via a simulated trading strategy.

4.1.2 Moving Average

The moving average model is defined as:

$$M_t = \frac{(Y_t + Y_{t-1} + Y_{t-2} + \ldots + Y_{t-n+1})}{n} \qquad [3]$$

where M_t is the moving average at time t

n is the number of terms in the moving average

Y_t is the actual rate of return at period t

The MACD strategy used is quite simple. Two moving average series are created with different moving average lengths. The decision rule for taking positions in the market is straightforward. Positions are taken if the moving averages intersect. If the short-term moving average intersects the long-term moving average from below a 'long' position is taken. Conversely, if the long-term moving average is intersected from above a 'short' position is taken[4].

The forecaster must use judgement when determining the number of periods *n* on which to base the moving averages. The combination that performed best over the in

[4] A 'long' EUR/USD position means buying Euros at the current price, while a 'short' position means selling Euros at the current price.

sample sub-period was retained for out-of-sample evaluation. The model selected was a combination of the EUR/USD and its 24-day moving average, namely *n* = 1 and 24 respectively or a (1,24) combination. The performance of this strategy is evaluated solely in terms of trading performance.

4.1.3 ARMA Model

Autoregressive moving average models (ARMA) assume that the value of a time series depends on its previous values (the autoregressive component) and on previous residual values (the moving average component)[5].

The ARMA model takes the form:

$$Y_t = \phi_0 + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \ldots + \phi_p Y_{t-p} + \varepsilon_t - w_1 \varepsilon_{t-1} - w_2 \varepsilon_{t-2} - \ldots - w_q \varepsilon_{t-q} \quad [4]$$

where	Y_t	is the dependent variable at time t
	Y_{t-1}, Y_{t-2}, and Y_{t-p}	are the lagged dependent variable
	ϕ_0, ϕ_1, ϕ_2, and ϕ_p	are regression coefficients
	ε_t	is the residual term
	ε_{t-1}, ε_{t-2}, and ε_{t-p}	are previous values of the residual
	w_1, w_2, and w_q	are weights.

Using as a guide the correlogram in the training and the test sub periods we have chosen a restricted ARMA (12,12) model. All of its coefficients are significant at the 95% confidence interval. The null hypothesis that all coefficients (except the constant) are not significantly different from zero is rejected at the 95% confidence interval (see Appendix A1).

The selected ARMA model takes the form:

$$\begin{aligned} Y_t &= 6.35 \cdot 10^{-5} - 0.688 Y_{t-1} - 0.369 Y_{t-2} - 0.219 Y_{t-7} - 0.400 Y_{t-11} - 0.540 Y_{t-12} \\ &+ 0.693 \varepsilon_{t-1} + 0.350 \varepsilon_{t-2} + 0.249 \varepsilon_{t-7} + 0.398 \varepsilon_{t-11} + 0.584 \varepsilon_{t-12} \end{aligned} \quad [5]$$

The model selected was retained for out-of-sample estimation. The performance of the strategy is evaluated in terms of trading performance.

4.1.4 Empirical Results for the Benchmark Models

A summary of the empirical results of the 3 benchmark models on the validation subset is presented in the table below. The empirical results of the models in the training and test sub-periods are presented in Appendix A.3 while Appendix A.2 documents the performance measures.

		NAIVE	MACD	ARMA
Sharpe Ratio	*(excluding costs)*	0.03	0.70	-0.40
Annualised Volatility	*(excluding costs)*	6.34%	6.38%	6.38%
Annualised Return	*(excluding costs)*	0.16%	4.44%	-2.53%
Maximum Drawdown	*(excluding costs)*	-7.32%	-4.73%	-10.12%
Positions Taken	*(annualised)*	132	20	189

Table 4: Trading performance of the statistical model

[5] For a full discussion on the procedure, refer to Box *et al.*, (1994) or Pindyck and Rubinfeld (1998).

4.2 Neural Networks

Neural networks exist in several forms in the literature. The most popular architecture is the Multi-Layer Perceptron (MLP).

A standard neural network has at least three layers. The first layer is called the input layer (the number of its nodes corresponds to the number of explanatory variables). The last layer is called the output layer (the number of its nodes corresponds to the number of response variables). An intermediary layer of nodes, the hidden layer, separates the input from the output layer. Its number of nodes defines the amount of complexity the model is capable of fitting. In addition, the input and hidden layer contain an extra node, called the bias node. This node has a fixed value of one and has the same function as the intercept in traditional regression models. Normally, each node of one layer has connections to all the other nodes of the next layer.

The network processes information as follows: the input nodes contain the value of the explanatory variables. Since each node connection represents a weight factor, the information reaches a single hidden layer node as the weighted sum of its inputs. Each node of the hidden layer passes the information through a nonlinear activation function and passes it on to the output layer if the calculated value is above a threshold.

The training of the network (which is the adjustment of its weights in the way that the network maps the input value of the training data to the corresponding output value) starts with randomly chosen weights and proceeds by applying a learning algorithm called backpropagation of errors[6] (Shapiro (2000)). The learning algorithm simply tries to find those weights which minimize an error function (normally the sum of all squared differences between target and actual values). Since networks with sufficient hidden nodes are able to learn the training data (as well as their outliers and their noise) by heart, it is crucial to stop the training procedure at the right time to prevent overfitting (this is called 'early stopping'). This can be achieved by dividing the dataset into 3 subsets respectively called the training and test sets used for simulating the data currently available to fit and tune the model and the validation set used for simulating future values. The network parameters are then estimated by fitting the training data using the above mentioned iterative procedure (backpropagation of errors). The iteration length is optimised by maximising the forecasting accuracy for the test dataset. Our networks, which are specially designed for financial purposes, will stop training when the profit of our forecasts in the test sub-period is maximized. Then the predictive value of the model is evaluated applying it to the validation dataset (out-of-sample dataset).

4.2.1 THE MULTI-LAYER PERCEPTRON MODEL

4.2.1.1 The MLP network architecture

The network architecture of a 'standard' MLP looks as presented in figure 3[7]:

[6] Backpropagation networks are the most common multi-layer networks and are the most commonly used type in financial time series forecasting (Kaastra and Boyd (1996)).

[7] The bias nodes are not shown here for the sake of simplicity.

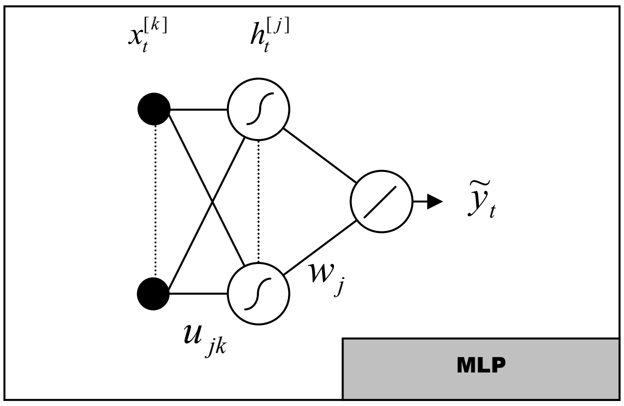


Fig. 3: A single output, fully connected MLP model

where:

$x_t^{[n]}$ $(n = 1,2,\cdots,k+1)$	are the model inputs (including the input bias node) at time t
$h_t^{[m]}$ $(m = 1,2,...,j+1)$	are the hidden nodes outputs (including the hidden bias node)
$\tilde{y}_t$	is the MLP model output
u_{jk} and w_j	are the network weights
∫	is the transfer sigmoid function: $S(x) = \frac{1}{1+e^{-x}}$, [6]
⁄	is a linear function: $F(x) = \sum_i x_i$ [7]

The error function to be minimised is:

$$E(u_{jk}, w_j) = \frac{1}{T}\sum_{t=1}^{T}\left(y_t - \tilde{y}_t(u_{jk}, w_j)\right)^2,$$ with y_t being the target value [8]

4.2.1.2 Empirical results of the MLP model

The trading performance of the MLP on the validation subset is presented in the table below. We chose the network with the highest profit in the test sub-period. Our trading strategy applied is simple: go or stay long when the forecast return is above zero and go or stay short when the forecast return is below zero. Appendix A.3 provides the performance of the MLP in the training and the test sub-periods while Appendix A.4 provides the characteristics of our network.

		NAIVE	MACD	ARMA	MLP
Sharpe Ratio	*(excluding costs)*	0.03	0.70	-0.40	1.88
Annualised Volatility	*(excluding costs)*	6.34%	6.38%	6.38%	6.34%
Annualised Return	*(excluding costs)*	0.16%	4.44%	-2.53%	11.91%
Maximum Drawdown	*(excluding costs)*	-7.32%	-4.73%	-10.12%	-5.05%
Positions Taken	*(annualised[8])*	132	20	189	109

Table 5: Trading performance of the benchmark models

As it can be seen the MLP outperforms our benchmark statistical models.

[8] The number of taken positions can differ from the number of trading days due to the possibility to hold a position for longer than 1 day.

4.2.2 THE RECURRENT NETWORK

Our next model is the recurrent neural network. While a complete explanation of RNN models is beyond the scope of this paper, we present below a brief explanation of the significant differences between RNN and MLP architectures. For an exact specification of the recurrent network, see Elman (1990).

A simple recurrent network has activation feedback, which embodies short-term memory. The advantages of using recurrent networks over feedforward networks, for modelling non-linear time series, has been well documented in the past. However as described in Tenti (1996) "the main disadvantage of RNNs is that they require substantially more connections, and more memory in simulation, than standard backpropagation networks" pp.569, thus resulting in a substantial increase in computational time. However having said this RNNs can yield better results in comparison to simple MLPs due to the additional memory inputs.

4.2.2.1 The RNN architecture

A simple illustration of the architecture of an Elman RNN is presented below.

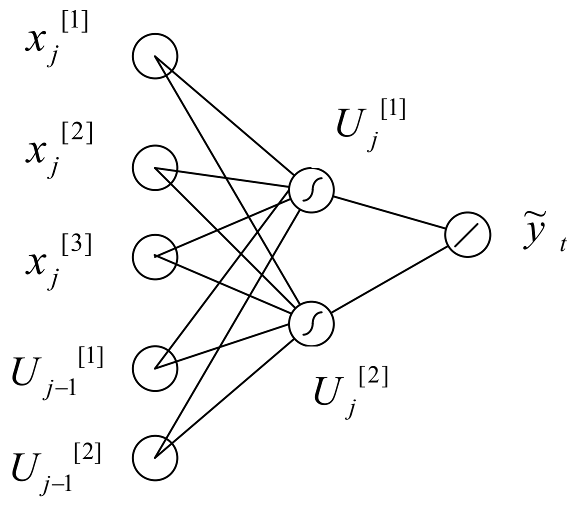


Fig. 4: Elman Recurrent neural network architecture with two nodes on the hidden layer.

where:

$x_t^{[n]}$ $(n = 1,2,\cdots,k+1)$, $u_t^{[1]}, u_t^{[2]}$	are the model inputs (including the input bias node) at time t
$\tilde{y}_t$	is the recurrent model output
$d_t^{[f]}$ $(f = 1,2)$ and $w_t^{[n]}$ $(n = 1,2,\cdots,k+1)$	are the network weights
$U_t^{[f]}$ $(f = 1,2)$	is the output of the hidden nodes at time t
∫	is the transfer sigmoid function: $S(x) = \frac{1}{1+e^{-x}}$, [9]

⊘ is the linear output function: $F(x)=\sum_i x_i$ [10]

The error function to be minimised is:

$$E(d_t, w_t) = \frac{1}{T}\sum_{t=1}^{T}\left(y_t - \tilde{y}_t(d_t, w_t)\right)^2 \quad [11]$$

In short, the RNN architecture can provide more accurate outputs because the inputs are (potentially) taken from all previous values (see inputs $U_{j-1}^{[1]}$ and $U_{j-1}^{[2]}$ in the figure above).

4.2.2.2 Empirical results of the RNN model

The RNNs are trained with gradient descent as were the MLPs. However, the increase in the number of weights, as mentioned before, makes the training process extremely slow taking ten times as long as the MLP.

We follow the same methodology as we did for the MLPs for the selection of our optimal network. The characteristics of the network that we used are on Appendix A.4 while in Appendix A.3 is a summary of the performance of the network in the training and test sub-periods.

The trading strategy is that followed for the MLP. As shown in table 6 below, the RNN has a worse performance compared to the MLP model when measure by the Sharpe ratio and annualised return. The results of other models are included for comparison.

		NAIVE	MACD	ARMA	MLP	RNN
Sharpe Ratio	*(excluding costs)*	0.03	0.70	-0.40	1.88	0.60
Annualised Volatility	*(excluding costs)*	6.34%	6.38%	6.38%	6.34%	6.34%
Annualised Return	*(excluding costs)*	0.16%	4.44%	-2.53%	11.91%	3.82%
Maximum Drawdown	*(excluding costs)*	-7.32%	-4.73%	-10.12%	-5.05%	-4.64%
Positions Taken	*(annualised)*	132	20	189	109	167

Table 6: Trading performance results

4.2.3 THE HIGHER ORDER NEURAL NETWORK

Higher Order Neural Networks (HONNs) were first introduced by Giles and Maxwell (1987) and were called "Tensor Networks". Although the extent of their use in finance has so far been limited, Knowles *et al*. (2005) show that, with shorter computational times and limited input variables, "the best HONN models show a profit increase over the MLP of around 8%" on the EUR/USD time series (p. 7). For Zhang *et al.* (2002), a significant advantage of HONNs is that "HONN models are able to provide some rationale for the simulations they produce and thus can be regarded as "open box" rather then "black box". Moreover, HONNs are able to simulate higher frequency, higher order non-linear data, and consequently provide superior simulations compared to those produced by ANNs (Artificial Neural Networks)" (p. 188).

4.2.3.1 The HONN Architecture

While they have already experienced some success in the field of pattern recognition and associative recall[9], HONNs have not yet been widely used in finance. The architecture of a three input second order HONN is shown below:

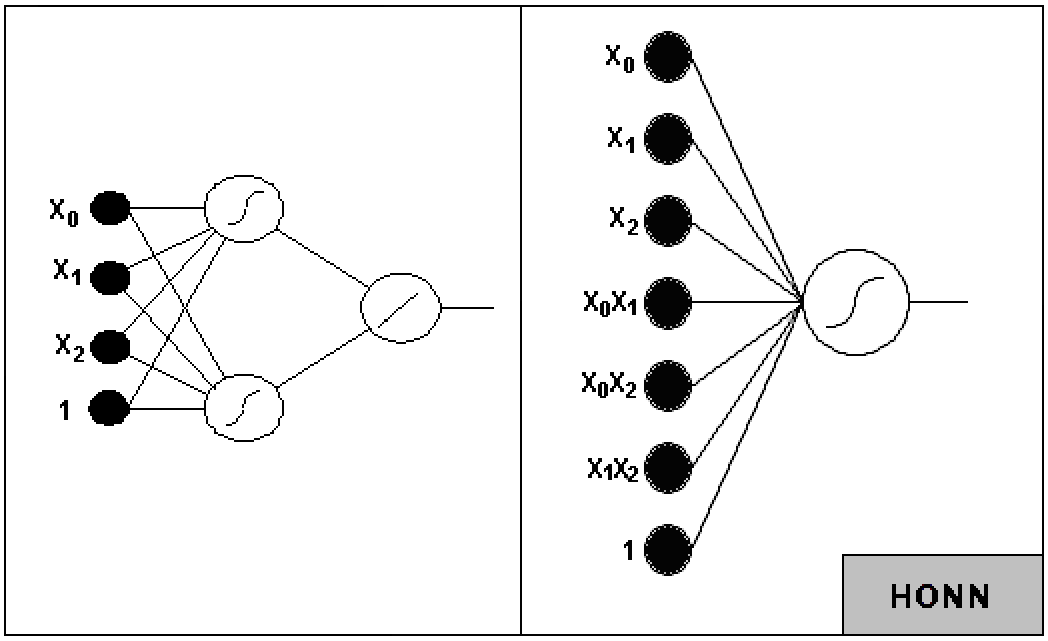


Fig. 5: Left, MLP with three inputs and two hidden nodes; right, second order HONN with three inputs

where:

$x_t^{[n]}$ $(n = 1,2,\cdots,k+1)$	are the model inputs (including the input bias node) at time t
$\tilde{y}_t$	is the HONNs model output
u_{jk}	are the network weights
●	are the model inputs.
∫	is the transfer sigmoid function: $S(x) = \dfrac{1}{1+e^{-x}}$, [12]
⊘	is a linear function: $F(x) = \sum_i x_i$ [13]

The error function to be minimised is:

$$E(u_{jk}, w_j) = \frac{1}{T}\sum_{t=1}^{T}\left(y_t - \tilde{y}_t(u_{jk},)\right)^2 ,$$ with y_t being the target value [14]

HONNs use joint activation functions; this technique reduces the need to establish the relationships between inputs when training. Furthermore this reduces the number of free weights and means that HONNS are faster to train than even MLPs. However because the number of inputs can be very large for higher order architectures, orders of 4 and over are rarely used.

[9] Associative recall is the act of associating two seemingly unrelated entities, such as smell and colour. For more information see Karayiannis *et al.* (1994).

Another advantage of the reduction of free weights means that the problems of overfitting and local optima affecting the results of neural networks can be largely avoided. For a complete description of HONNs see Knowles *et al.* (2005).

4.2.3.2 Empirical results of the HONN model

We follow the same methodology as we did with RNNs and the MLPs for the selection of our optimal HONN. The trading strategy is that followed for the MLP. A summary of our findings is presented in table 7 below while Appendix A.3 provides the performance of the network in the training and the test sub-periods and Appendix A.4 provides its characteristics. The results of other models are included for comparison.

		NAIVE	**MACD**	**ARMA**	**MLP**	**RNN**	**HONN**
Sharpe Ratio	*(excluding costs)*	0.03	0.70	-0.40	1.88	0.60	0.99
Annualised Volatility	*(excluding costs)*	6.34%	6.38%	6.38%	6.34%	6.34%	6.37%
Annualised Return	*(excluding costs)*	0.16%	4.44%	-2.53%	11.91%	3.82%	6.29%
Maximum Drawdown	*(excluding costs)*	-7.32%	-4.73%	-10.12%	-5.05%	-4.64%	-4.37%
Positions Taken	*(annualised)*	132	20	189	109	167	80

Table 7: Trading performance results

We can see that HONNs perform significantly better than the RNNs but still worse than the traditional MLPs.

4.2.4 THE PSI SIGMA NETWORK

Psi Sigma networks can be considered as a class of feedfoward fully connected HONNs. First introduced by Shin and Ghosh (1991), the Psi Sigma network utilizes product cells as the output units to indirectly incorporate the capabilities of higher-order networks while using a fewer number of weights and processing units. Their creation was motivated by the need to create a network combining the fast learning property of single layer networks with the powerful mapping capability of HONNs while avoiding the combinatorial increase in the required number of weights. While the order of the more traditional HONN architectures is expressed by the complexity of the inputs, in the context of Psi Sigma, it is represented by the number of hidden nodes.

4.2.4.1 The Psi Sigma architecture

In a Psi Sigma network the weights from the hidden to the output layer are fixed to 1 and only the weights from the input to the hidden layer are adjusted, something that greatly reduces the training time. Moreover, the activation function of the nodes in the hidden layer is the summing function while the activation function of the output layer is a sigmoid. The figure below shows a Psi Sigma with one output layer.

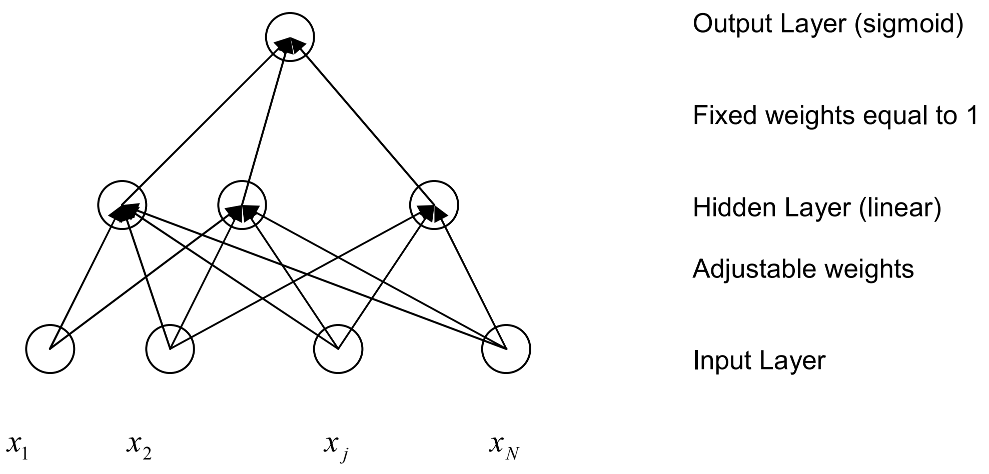


Fig. 6: A Psi Sigma network with one output layer

where:

$x_t \quad (n = 1,2,\cdots,k+1)$ are the model inputs (including the input bias node)

$\widetilde{y}_t$ is the Psi Sigma ouput

w_j is the adjustable weights

$h(x) = \sum_i x_i$ is the hidden layer activation function [15]

$\sigma(x) = \dfrac{1}{1+e^{-xc}}$ is the output unit adaptive sigmoid activation function with c the adjustable term [16]

The error function to be minimised is:

$E(c, w_j) = \dfrac{1}{T}\sum_{t=1}^{T}(y_t - \widetilde{y}_t(w_k, c))^2$ with y_t being the target value [17]

For example let us consider a Psi Sigma network which is fed with a N+1 dimensional input vector $x = (1, x_1,..., x_N)^T$.These inputs are weighted by K weight factors $w_j = (w_{0j}, w_{1j},..., w_{Nj})^T$, $j = 1,2,..K$ and summed by a layer of K summing units, where K is the desired order of the network. So the output of the j-th summing unit, h_j in the hidden layer, is given by: $h_j = w_j^T x = \sum_{k=1}^{N} w_{kj}x_k + w_{oj}$,j=1,2,…, K while the output $\widetilde{y}$ of the network is given by $\widetilde{y} = \sigma(\prod_{j=1}^{K} h_j)$ (in our case we selected as σ the sigmoid function $\sigma(x) = \dfrac{1}{1+e^{-xc}}$ [18]). Note that by using products in the output layer we directly incorporate the capabilities of higher order networks with a smaller number of weights and processing units. For example, a k-th degree HONN with d inputs needs $\sum_{i=0}^{k} \dfrac{(d+i-1)!}{i!(d+1)!}$ weights if all products of up to k components are to be incorporated while a similar Psi Sigma network needs only (d+1)*k weights. Also note that the sigmoid function is neuron adaptive. As the network is trained not only the weights but also c in [18] is adjusted. This strategy seems to provide better fitting properties and increases the approximation capability of a neural network by introducing an extra variable in the estimation, compared to classical architectures with sigmoidal neurons (Vecci *et al.* (1998)).

4.2.4.2 Empirical results of the Psi Sigma model

The price for the flexibility and speed of Psi Sigma networks is that they are not universal approximators. We need to choose a suitable order of approximation (or else the number of hidden units) by considering the estimated function complexity, amount of data and amount of noise present. To overcome this, our code runs simulations for orders two to six and we then select the best network based on statistical criteria on the test and training sample as for the RNN and HONN models. The characteristics of the network that we used are presented on Appendix A.4 while its performance on the training and test sub periods is summarized on the Appendix A.3.The trading strategy is that followed for the MLP. A summary of our findings is presented in table 8 below. Again we include the results of other models for comparison.

		NAIVE	MACD	ARMA	MLP	RNN	HONN	Psi Sigma
Sharpe Ratio	*(excluding costs)*	0.03	0.70	-0.40	1.88	0.60	0.99	1.18
Annualised Volatility	*(excluding costs)*	6.34%	6.38%	6.38%	6.34%	6.34%	6.37%	6.36%
Annualised Return	*(excluding costs)*	0.16%	4.44%	-2.53%	11.91%	3.82%	6.29%	7.53%
Maximum Drawdown	*(excluding costs)*	-7.32%	-4.73%	-10.12%	-5.05%	-4.64%	-4.37%	-4.86%
Positions Taken	*(annualised)*	132	20	189	109	167	80	117

Table 8: Trading performance results

As can been seen the Psi Sigma outperforms all other statistical, technical and neural network models expect the MLPs. Moreover, the major theoretical advantage of Psi Sigma networks, namely their speed, was clearly confirmed as we achieved our results in one half of the time needed to train the MLPs and the HONNs and one tenth of the time needed for the RNNs.

5. TRADING COSTS, FILTERS AND LEVERAGE

Up to now, we have presented the trading results of all our models without considering transaction costs. Since some of our models trade quite often, taking transaction costs into account might change the whole picture.

We therefore introduce transaction costs as well as a filtered trading strategy for each model. The aim is to devise a trading strategy filtering only those trades which have a high probability of being successful. This should help to reduce the negative effect of transaction costs as trades with an expected gain lower than the transaction costs should be omitted.

5.1 TRANSACTION COSTS

The transaction costs for a tradable amount, say USD 5-10 million, are about 1 pip (0.0001 EUR/USD) per trade (one way) between market makers. But since we consider the EUR/USD time series as a series of middle rates, the transaction costs is one spread per round trip.

With an average exchange rate of EUR/USD of 1.341 for the out-of-sample period, a cost of 1 pip is equivalent to an average cost of 0.008% per position.

5.2 CONFIRMATION FILTER STRATEGIES

5.2.1 Confirmation Filters

We now introduce trading strategies devised to filter out those trades with expected returns below the 0.008% transaction cost. Due to the architecture of all our models, the trading strategy will consist by one single parameter.

Up to now, the trading strategies applied to the models use a zero threshold: they suggest to go long when the forecast is above zero and to go short when the forecast is below zero**.** In the following, we examine how the models behave if we introduce a threshold *d* around zero (see figure 7) and what happens if we vary that threshold.The filter rule for all our models is presented in figure 7 below.

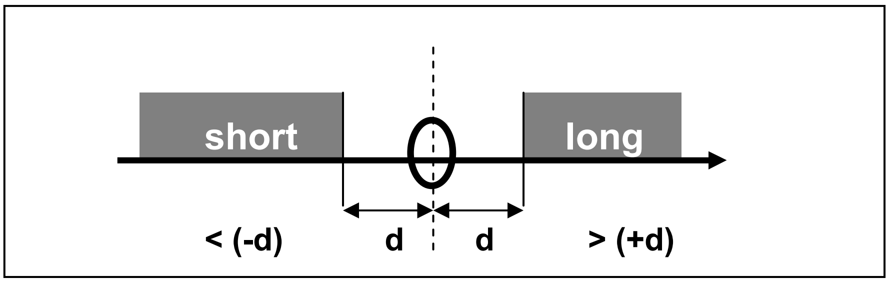


Fig. 7: Filtered trading strategy with one single parameter

5.2.2 Empirical Results

The methodology that we follow for the selection of the optimal thresholds is simple. Taking the test period results, we choose the threshold that gives the higher return and Sharpe ratio. Our chosen parameters are presented in the table below while the detailed results leading to their choice are documented in Appendix A.5.

Model	**Threshold (d)**
Naive	*=0.35*
MA	*=0.00*
ARMA	*=0.10*
MLP	*=0.20*
RNN	*= 0.00*
HONN	*= 0.00*
Psi Sigma	*= 0.30*

Table 9: Chosen parameters for each trading strategy

For the MACD, MLP and the HONN strategies we leave the threshold at zero (d=0.0) since the profit on the test dataset is largest at this value. On the other hand, we selected as threshold the values of 0.35, 0.10, 0.20 and 0.30 for the Naïve, ARMA, RNN and the Psi Sigma models respectively as in these cases the profit in the test sub-period is maximized.

A summary of the out-of-sample trading performance of all models using the selected thresholds and taking into account the transaction costs is on the table below.

	NAIVE	MACD	ARMA	MLP	RNN	HONN	Psi Sigma
Sharpe Ratio (excluding costs)	-0.70	0.70	0.41	1.00	0.60	0.99	0.70
Annualised Volatility (excluding costs)	4.12%	6.38%	3.43%	4.03%	6.34%	6.37%	5.22%
Annualised Return (excluding costs)	-2.90%	4.44%	1.39%	4.02%	3.82%	6.29%	3.63%
Maximum Drawdown (excluding costs)	-7.23%	-4.73%	-3.14%	-2.45%	-4.64%	-4.37%	-5.60%
Positions Taken (annualised)	79	20	80	83	167	80	91
Transaction costs	0.63%	0.16%	0.64%	0.66%	1.33%	0.64%	0.73%
Annualised Return (including costs)	-3.53%	4.28%	0.75%	3.36%	2.48%	5.65%	2.90%

Table 10: Out-of-sample results for the chosen parameters

From the final row of table 10 we can see that, after transaction costs, the HONN network outperforms all the other strategies based on the annualised return. The MACD strategy also performs well and presents the second best performance in terms of annualized return. On the other hand, the MLP networks which performed best without trading filter seem to be unable to fully exploit the introduction of the modified trading strategy. The Psi Sigma which also performed well before the introduction of the trading strategy seems also incapable of exploiting it. However, it is worth mentioning that the time used to derive these results with the Psi Sigma network is half that needed with HONNs and the MLPs and one tenth of that needed with RNNs[10].

5.3 Leverage to exploit high Sharpe ratios

In order to further improve the trading performance of our models we introduce a "level of confidence" to our forecasts, i.e. a leverage based on the test sub-period. For the naïve model, which presents a negative return we do not apply leverage. The leverage factors applied are calculated in such a way that each model has a common volatility of 10%[11] on the test data set.

The transaction costs are calculated by taking 0.008% per position into account, while the cost of leverage (interest payments for the additional capital) is calculated at 4% p.a. (that is 0.016% per trading day[12]). Our final results are presented in table 11 below.

[10] We needed about 3 minutes to train our Psi Sigma network, about 6 minutes to train our MLP and the HONN and about 30 minutes to train our RNN with an Intel Core 2 Duo T7300 Fujitsu Amilo Laptop.

[11] Since most of the models have a volatility of about 10%, we have chosen this level as our basis. The leverage factors retained are given in table 11 below.

[12] The interest costs are calculated by considering a 4% interest rate p.a. divided by 252 trading days. In reality, leverage costs also apply during non-trading days so that we should calculate the interest costs using 360 days per year. But for the sake of simplicity, we use the approximation of 252 trading days to spread the leverage costs of non-trading days equally over the trading days. This approximation prevents us from keeping track of how many non-trading days we hold a position.

	NAIVE	MACD	ARMA	MLP	Recurrent	HONN	Psi Sigma
Sharpe Ratio (excluding costs)	-0.70	0.70	0.41	1.00	0.60	0.99	0.70
Annualised Volatility (excluding costs)	4.12%	7.27%	9.67%	5.5%	7.23%	7.26%	8.30%
Annualised Return (excluding costs)	-2.90%	5.06%	3.93%	5.82%	4.35%	7.17%	5.78%
Maximum Drawdown (excluding costs)	-7.23%	-5.39%	-3.58%	-3.55%	-5.30%	-4.98%	-8.90%
Leverage Factor	-	1.14	2.82	1.45	1.14	1.14	1.59
Positions Taken (annualised)	79	20	80	83	167	80	91
Transaction and leverage costs	0.63%	1.02%	11.79%	3.42%	2.19%	1.50%	4.34%
Annualised Return (including costs)	-3.53%	4.04%	-7.86%	2.40%	2.16%	5.67%	1.44%

Table 11: Trading performance - final results[13]

As can be see from the last row of table 11, HONNs continue to demonstrate a superior trading performance. The MACD strategy also performs well and presents the second higher annualised return. In general, we observe that all models expect the HONNs, show an inability to gain any extra profit from the leverage as the increased transaction costs seems to counter any benefits. Again it is worth mentioning, that the time needed to train the Psi Sigma network was considerably shorter compared with that needed for the MLP, the RNN and the HONN networks.

6. CONCLUDING REMARKS

In this paper, we apply Multi-layer Perceptron, Recurrent, Higher Order and Psi Sigma neural networks to a one-day-ahead forecasting and trading task of the Euro/Dollar (EUR/USD) exchange rate using the European Central Bank (ECB) fixing series with only autoregressive terms as inputs. We use a naïve, a MACD and an ARMA model as benchmarks. We develop these different prediction models over the period January 1999 - June 2006 and validate their out-of-sample trading efficiency over the following period from July 2006 through December 2007.

The MLPs demonstrated the higher trading performance in terms of annualised return and Sharpe ratio before transaction costs and elaborate trading strategies are applied. When refined trading strategies are applied and transaction costs are considered the HONNs manage to outperform all other models achieving the highest annualised return. On the other hand, the RNNs and the Psi Sigma models, which in previous applications over the EUR/USD dollar exchange rate performed remarkably well (Dunis *et al.* (2008)) seem to have a difficulty in providing good forecasts when autoregressive series are only used as inputs.

It is also important to note that the HONN network which presents the higher annualised return after transaction costs and trading strategies are applied, needs

[13] Not taken into account the interest that could be earned during times where the capital is not traded (non-trading days) and could therefore be invested.

about the same training time as the MLPs and far less than the RNNs models, a much desirable feature in a real-life quantitative investment and trading environment: in the circumstances, our results should go some way towards convincing a growing number of quantitative fund managers to experiment beyond the bounds of the more traditional models.

APPENDIX

A.1 ARMA Model

The output of the ARMA model used in this paper is presented below.

```
Dependent Variable: RETURNS
Method: Least Squares
Date: 02/29/08   Time: 12:38
Sample (adjusted): 13 1920
Included observations: 1908 after adjustments
Convergence achieved after 12 iterations
Backcast: 1 12
==========================================================================
       Variable       Coefficient    Std. Error    t-Statistic     Prob.
==========================================================================
          C            6.35E-05      0.000151      0.420579       0.6741
        AR(1)         -0.688152      0.040492     -16.99463       0.0000
        AR(2)         -0.369020      0.067865     -5.437592       0.0000
        AR(7)         -0.218734      0.073635     -2.970535       0.0030
        AR(11)        -0.400372      0.043749     -9.151508       0.0000
        AR(12)        -0.539713      0.052040     -10.37107       0.0000
        MA(1)          0.692697      0.034799      19.90592       0.0000
        MA(2)          0.349884      0.064745      5.404045       0.0000
        MA(7)          0.248779      0.073280      3.394927       0.0007
        MA(11)         0.397565      0.039306      10.11460       0.0000
        MA(12)         0.584116      0.052393      11.14877       0.0000
==========================================================================
R-squared               0.014757    Mean dependent var          7.04E-05
Adjusted R-squared      0.009563    S.D. dependent var          0.006518
S.E. of regression      0.006487    Akaike info criterion      -7.232214
Sum squared resid       0.079834    Schwarz criterion          -7.200196
Log likelihood          6910.533    F-statistic                 2.841359
Durbin-Watson stat      2.006369    Prob(F-statistic)           0.001639
==========================================================================
Inverted AR Roots       .89-.28i      .89+.28i     .61+.70i     .61-.70i
                        .14+.98i      .14-.98i    -.37+.89i    -.37-.89i
                       -.73+.67i     -.73-.67i    -.89+.16i    -.89-.16i
Inverted MA Roots       .90-.28i      .90+.28i     .62+.70i     .62-.70i
                        .14+.98i      .14-.98i    -.37+.89i    -.37-.89i
                       -.73-.68i     -.73+.68i    -.90+.16i    -.90-.16i
==========================================================================
```

A.2 Performance Measures

The performance measures are calculated as follows:

Performance Measure	Description	
Annualised Return	$R^A = 252 * \frac{1}{N} \sum_{t=1}^{N} R_t$ *with R_t being the daily return*	[19]
Cumulative Return	$R^C = \sum_{t=1}^{N} R_t$	[20]
Annualised Volatility	$\sigma^A = \sqrt{252} * \sqrt{\frac{1}{N-1} * \sum_{t=1}^{N} (R_t - \overline{R})^2}$	[21]
Sharpe Ratio	$SR = \frac{R^A}{\sigma^A}$	[22]
Maximum Drawdown	*Maximum negative value of $\sum (R_t)$ over the period* $MD = \underset{i=1,\cdots,t;t=1,\cdots,N}{Min} \left(\sum_{j=i}^{t} R_j \right)$	[23]

Table 12: Trading simulation performance measures

A.3 Empirical Results in the Training and Test Sub-Periods

		NAIVE	MACD	ARMA	MLP	RNN	HONN	Psi Sigma
Sharpe Ratio	*(excluding costs)*	-0.22	2.49	1.20	0.41	0.34	0.51	0.54
Annualised Volatility	*(excluding costs)*	10.28%	10.24%	10.33%	10.36%	10.35%	10.36%	10.35%
Annualised Return	*(excluding costs)*	-2.25%	25.45%	12.40%	4.27%	3.51%	5.31%	5.55%
Maximum Drawdown	*(excluding costs)*	-29.39%	-5.96%	-9.69%	-21.42%	-21.37%	-29.79%	-20.31%
Positions Taken	*(annualised)*	77	10	53	77	97	52	75

Table 13: In-sample trading performance

A.4 Networks Characteristics

We present below the characteristics of the networks with the best trading performance on the test sub-period for the different architectures.

Parameters	**MLP**	**Reccurent**	**HONNs**	**Psi Sigma**
Learning algorithm	*Gradient descent*	*Gradient descent*	*Gradient descent*	*Gradient descent*
Learning rate	*0.001*	*0.001*	*0.001*	*0.5*
Momentum	*0.003*	*0.003*	*0.003*	*0.5*
Iteration steps	*1000*	*1000*	*1000*	*500*
Initialisation of weights	*N(0,1)*	*N(0,1)*	*N(0,1)*	*N(0,1)*
Input nodes	*12*	*12*	*12*	*12*
Hidden nodes (1layer)	*7*	*5*	*0*	*6*
Output node	*1*	*1*	*1*	*1*

Table 14: Network characteristics

A.5 Threshold Selection

The table below shows the results of the filtered trading strategy applied to the test dataset for different values of *d*. We choose the threshold that gives the highest return.

Selection of the optimal threshold based on the test period	Threshold									
	0	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.45
Naive	-1.1% (-0.1)	-3.7% (-0.4)	-2.2% (-0.3)	-1.7% (-0.2)	-2.5% (-0.4)	-1.2% (-0.2)	1.1% (0.2)	**4.5% (0.8)**	2.0% (0.4)	4.0% (0.8)
MACD	**5.3% (0.6)**	-0.7% (-0.1)	-2.5% (-0.5)	3.7% (1.1)	0.3% (0.1)	-0.2% (-0.8)	0.0% (0.0)	0.0% (0.0)	0.0% (0.0)	0.0% (0.0)
ARMA	6.6% (0.8)	3.9% (0.7)	**7.0% (2.0)**	0.5% (0.3)	0.0% (0.0)	0.0% (0.0)	0.0% (0.0)	0.0% (0.0)	0.0% (0.0)	0.0% (0.0)
MLP	6.3% (0.7)	8.7% (1.1)	9.2% (1.2)	1.7% (1.5)	**12.2% (1.8)**	11.1% (1.7)	6.2% (1.0)	5.4% (1.0)	3.9% (0.9)	-0.1% (-0.1)
RNN	**10.1% (1.1)**	8.1% (0.9)	5.6% (0.7)	4.9% (0.6)	4.2% (0.5)	5.4% (0.7)	4.6% (0.6)	7.5% (1.0)	6.6% (0.9)	8.7% (1.2)
HONN	**6.9% (0.8)**	4.2% (0.6)	2.9% (0.5)	3.6% (0.9)	1.3% (0.5)	1.7% (0.8)	0.4% (0.6)	0.2% (0.8)	0.2% (0.8)	0.0% (0.0)
Psi Sigma	6.3% (0.7)	5.0% (0.6)	5.2% (0.7)	5.6% (0.8)	10.4% (1.5)	10.1% (1.5)	**10.5% (1.6)**	9.7% (1.7)	7.7% (1.5)	5.3% (1.2)

Table 15: Results for alternative threshold values

Note: The entries represent the annualized return values while the values in parenthesis represent the Sharpe ratio.

REFERENCES

Adam, O., Zarader, L. and Milgram, M. (1994), 'Identification and Prediction of Non-Linear Models with Recurrent Neural Networks', *Laboratoire de Robotique de Paris*.

Bera, A. and Jarque, C. (1981), 'An efficient large-sample test for normality of observations and regression residuals', *Australian National University Working Paper in Economics and Econometrics* , 40.

Box, G., Jenkins, G. and Gregory, G. (1994), *Time Series Analysis: Forecasting and Control*, Prentice-Hall, New Jersey.

Connor, J. and Atlas, L. (1993), 'Recurrent Neural Networks and Time Series Prediction', *Proceedings of the International Joint Conference on Neural Networks*, 301-306.

Dunis, C. 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.

(DOI: 10.1002/0470013265.ch4)

Dunis, C., Laws, J. and Evans B. (2006a), 'Trading Futures Spreads: An application of Correlation and Threshold Filters', *Applied Financial Economics*, 16, 1-12.

(DOI: 10.1080/09603100500426432)

Dunis, C., Laws, J. and Evans B. (2006b), 'Modelling and Trading the Gasoline Crack Spread: A Non-Linear Story', *Derivatives Use, Trading and Regulation*, 12, 126-145.

(DOI: 10.1057/palgrave.dutr.1840046)

Dunis, C., Laws, J. and Sermpinis, G. (2008) 'Higher Order and Recurrent Neural Architectures for Trading the EUR/USD Exchange Rate', *CIBEF Working Papers*. Available at www.cibef.com

Dunis, C. 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.

(**DOI**: 10.1002/for.935)

Dunis, C. 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 Trading and Investment*, John Wiley, Chichester.

(**DOI**: 10.1002/0470013265.ch1)

Elman, J. L. (1990), 'Finding Structure in Time', *Cognitive Science*, 14, 179-211.

(**DOI** :10.1016/0364-0213(90)90002-E)

Fulcher, J., Zhang, M. and Xu, S. (2006), '*The Application of Higher-Order Neural Networks to Financial Time Series*', Artificial Neural Networks in Finance and Manufacturing, Hershey, PA: Idea Group, London.

Ghazali, R., Hussain, A. and Merabti, M. (2006), 'Higher Order Neural Networks for Financial Time Series Prediction', *The 10th IASTED International Conference on Artificial Intelligence and Soft Computi*ng, Palma de Mallorca, Spain, 119-124.

Ghosh, J. and Shin, Y. (1992) 'Efficient Higher-Order Neural Networks for Classification and Function Approximation', *International Journal of Neural Systems*, 3, 4, 323-350.

(**DOI**: 10.1142/S0129065792000255)

Giles, L. and Maxwell, T. (1987) ‘Learning, Invariance and Generalization in Higher Order Neural Networks’*, Applied Optics,* 26, 4972-4978.

Hussain, A., Ghazali, R and Al-Jumeily D. (2006), ‘Dynamic Ridge Polynomial Neural Network for Financial Time Series Prediction’, IEEE International conference on Innovation in Information Technology, IIT06, Dubai.

Kaastra, I. and Boyd, M. (1996), ‘Designing a Neural Network for Forecasting Financial and Economic Time Series’, *Neurocomputing*, 10, 215-236.

(**DOI:** 10.1016/0925-2312(95)00039-9)

Kamijo, K. and Tanigawa,T. (1990), ‘Stock Price Pattern Recognition: A Recurrent Neural Network Approach’, *In Proceedings of the International Joint Conference on Neural Networks*, 1215-1221.

Karayiannis, N. and Venetsanopoulos, A. (1994), ‘On The Training and Performance of High-Order Neural Networks’, *Mathematical Biosciences*, 129, 143-168.

(**DOI:** 10.1016/0025-5564(94)00057-7)

Knowles, A., Hussein, A., Deredy, W., Lisboa, P. and Dunis, C. L. (2005), ‘Higher-Order Neural Networks with Bayesian Confidence Measure for Prediction of EUR/USD Exchange Rate’, *CIBEF Working Papers*. Available at www.cibef.com.

Kosmatopoulos, E., Polycarpou, M., Christodoulou, M. and Ioannou, P. (1995), ‘High-Order Neural Network Structures for Identification of Dynamical Systems’, *IEEE Transactions on Neural Networks,* 6, 422-431.

Lindemann, A., Dunis, C., and Lisboa P. (2004), ‘Level Estimation, Classification and Probability Distribution Architectures for Trading the EUR/USD Exchange Rate’. *Neural Network Computing & Applications,* 14, 3, 256-271.

(**DOI**: 10.1007/s00521-004-0462-8)

Lisboa, P. J. G. and Vellido, A. (2000), ‘Business Applications of Neural Networks’, vii-xxii, 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,* World Scientific, Singapore.

(**DOI:** 10.1016/S0377-2217(02)00302-8)

Pindyck, R. and Rubinfeld, D. (1998**)**, *Econometric Models and Economic Forecasts*, 4th edition, McGraw-Hill, New York.

Psaltis, D., Park, C. and Hong, J. (1988), ‘Higher Order Associative Memories and their Optical Implementations.’, *Neural Networks,* 1, 149-163.

Redding, N., Kowalczyk, A. and Downs, T. (1993), ‘Constructive Higher-Order Network Algorithm that is Polynomial Time’, *Neural Networks*, 6, 997-1010.

(DOI: 10.1016/S0893-6080(03)00188-6)

Shapiro, A. F. (2000), ‘A Hitchhiker’s Guide to the Techniques of Adaptive Nonlinear Models’, *Insurance, Mathematics and Economics*, 26, 119-132.

(DOI: 10.1016/S0167-6687(99)00058-X)

Shin, Y. and Ghosh, J. (1991) ‘The Pi-Sigma Network: An Efficient Higher-Order Neural Network for Pattern Classification and Function Approximation’, *Proceedings IJCNN*, Seattle, July, 13-18.

Tenti, P. (1996), ‘Forecasting Foreign Exchange Rates Using Recurrent Neural Networks’, *Applied Artificial Intelligence*, 10, 567-581.

(**DOI:** 10.1080/088395196118434)

Tino, P., Schittenkopf, C. and Doffner, G. (2001), ‘Financial Volatility Trading Using Recurrent Networks’, *IEEE Transactions in Neural Networks*, 12, 4, 865-874.

Vecci, L., Piazza, F. and Uncini, A. (1998), ‘Learning and Approximation Capabilities of Adaptive Spline Activation Neural Networks’, *Neural Networks*, 11, 259-270.

(**DOI**: 10.1016/S0893-6080(97)00118-4)

Zhang, M., Xu, S., X. and Fulcher, J. (2002), ‘Neuron-Adaptive Higher Order Neural-Network Models for Automated Financial Data Modelling’, *IEEE Transactions on Neural Networks*,13,1, 188-204.

(**DOI**:10.1109/72.977302)