Annals of Operations Research (2021) 297:293–308
https://doi.org/10.1007/s10479-019-03144-y

S.I.: NETWORKS AND RISK MANAGEMENT

Neural networks in financial trading

Georgios Sermpinis[1] · Andreas Karathanasopoulos[2] · Rafael Rosillo[3] · David de la Fuente[4]

Published online: 24 January 2019

Abstract
In this study, we generate 50 Multi-layer Perceptons, 50 Radial Basis Functions, 50 Higher Order Neural Networks and 50 Recurrent Neural Network and we explore their utility in forecasting and trading the DJIA, NASDAQ 100 and the NIKKEI 225 stock indices. The statistical significance of the forecasts is examined through the False Discovery Ratio of Bajgrowicz and Scaillet (J Financ Econ 106(3):473–491, 2012). Two financial everages, based on the levels of financial stress and the financial volatility respectively, are also applied. In terms of the results, we note that RNN have the higher percentage of significant models and present the stronger profitability compared to their Neural Network counterparts. The financial leverages doubles the trading performance of our models.

Keywords Neural networks · Forecasting · Trading · Multiple hypothesis testing

1 Introduction

Neural networks (NNs) are a class of non-linear models inspired by the work and functioning of biological neurons. They have applications in all aspects of Science and are considered as the formulation that will have the greatest impact to our lives in the future (Maren et al.

✉ Georgios Sermpinis
georgios.sermpinis@glasgow.ac.uk

Andreas Karathanasopoulos
akarathanasopoulos@ud.ac.ae

Rafael Rosillo
rosillo@uniovi.es

David de la Fuente
david@uniovi.es

[1] University of Glasgow Business School, University of Glasgow, Adam Smith Building, Glasgow G12 8QQ, UK

[2] Dubai Business School, University of Dubai, Dubai 14143, UAE

[3] Business and Management Department, University of Oviedo, Campus de Viesques, s/n. 33204, Gijón, Spain

[4] Business and Management Department, University of Oviedo, Campus de Viesques, s/n. 33204, Oviedo, Spain

2014). In finance and trading, there is a paradox in NNs and their applications. From the one hand, there is an industry (asset management) with hundreds of thousands of employees and billions of dollars in revenues that relies heavily on their applications. Chui et al. (2016) and the Boston Consulting Group (2016) estimate that by 2025 the field of wealth management will be dominated by the interaction of artificial intelligence and big data. On the other hand, the academic research of NNs and their applications in financial forecasting and trading is extensive, but it is usually presented in no mainstream outlets. The applications of NNs in trading rely on the assumption that financial asset price time series are systematically linked and that previous prices contain substantial relevant information that could be used to predict future price trends. This assumption contradicts the Efficient Market Hypothesis (EMH), the foundation of the modern finance theory, which states that asset prices reflect all available information and it is not possible to beat the market with time series models (such as NNs). Theorists in finance argue that the studies that present profitability in trading with NNs are results of extensive data mining or luck.

The purpose of this study is to contribute to this debate and explore the utility of NNs through a multiple hypothesis testing (MHT) framework. NNs are widely applicable in trading desks but their performance relies on the choice of hyperparameters and inputs. In the absence of any formal theory behind their selection, practitioners rely on their experience to make that selection. In this research, a large number of NNs is generated and applied to the task of forecasting and trading the 1-day ahead logarithmic return of DJIA, the NASDAQ 100 and the NIKKEI 225 stock indices. The generated NNs differ in terms of hyperparameters and inputs. The four main NN architectures, the Multi-layer Percepton (MLP), the Radial Basis Function (RBF), the Higher Order Neural Network (HONN) and the Recurrent Neural Network (RNN) are explored. To be more specific, 50 NNs from each of the four families are generated and the percentage of statistically significant NNs is explored through the False Discovery Ratio (FDR) of Bajgrowicz and Scaillet (2012). Then based on the significant NNs, series of portfolios are generated and their profitability is explored on the three stock indices on two different datasets, 2007–2008 and 2016–2017. The first period covers the recent global financial crisis, while the second one is free from strong financial stress factors. Two financial leverages, based on the levels of financial stress and the financial volatility respectively, are also introduced.

The literature of NNs in Finance focuses on forecasting and trading applications (Zhang et al. 1998). Fulcher et al. (2006) apply HONN in forecasting the AUD/USD exchange rate and Panda and Narasimhan (2007) use a MLP to produce statistical accurate forecasts of the INR/USD exchange rate. Andreou et al. (2008) use NNs to forecast European options with disappointing results, while Yang et al. (2008) employ a MLP and other regression techniques to examine the potential martingale behaviour of Euro exchange rates in the context of out-of-sample forecasts. Dunis et al. (2010, 2011) highlight the advantages of HONNs in a series of financial forecasting exercises. Sermpinis et al. (2013) introduces a semi adaptive RBF algorithm, while Yang et al. (2010) study the predictability of eighteen stock indexes with NNs and linear models. Bekiros and Georgoutsos (2008) forecast the NASDAQ index with RNNs. In a similar application, Matías and Reboredo (2012) forecast successfully with NNs and other nonlinear models intraday stock market returns. Adeodato et al. (2011) show that an approach based on the use of median for combining MLP forecasts won the NN3 Forecasting. Guresen et al. (2011) forecasts the daily closing prices of the NASDAQ index with MLPs and dynamic NNs. Bagheri et al. (2014) applies a similar set of algorithms to the task of trading four foreign exchange rates while Krauss et al. (2017) presents an application of deep learning NNs in trading the S&P500. It is worth noting that none of the aforementioned studies considers the data snooping bias and examines the statistical significance of the

results. Surveys on the applications of NNs in trading have been presented by Zhang et al. (1998), Bahrammirzaee (2010) and Cavalcante et al. (2016).

It terms of the results of this study, we prove that there are NN architectures that provide statistically significant forecasts for the series under study. The generated portfolios are profitable in terms of Sharpe Ratio and annualized return after transaction costs in both in-sample and out-of-sample. RNNs seems the architecture that presents the higher percentage of significant NNs and the stronger profitability. We note that the profitability of all models is higher in the first dataset that covers the recent global financial crisis. The application of the two financial leverage doubles the profitability of our models. Our study highlights the importance of NNs in financial trading. Their non-linear nature allows them to capture the pattern of financial trading series. Our results explain the popularity of NNs in the finance industry and confirm to some extent the Adaptive Market Hypothesis (ADM) of Lo (2004). The ADM states that complex models should have an advantage but their performance should vary through time.

The rest of the paper is organised as follows. In Sect. 2, there is a description of the four NN architectures under study. In the same section, we discuss the role of the hyperparameters and the FDR, the MHT framework under study. Section 3 presents the empirical results of all models considered and investigates their performance with the introduction of the two financial leverages. Section 4 provides some concluding remarks.

2 Methodology

NNs are linear and polynomial approximation methods that relate a set of inputs variables $\{x_t^i\}$ where i = 1…n is the number of inputs, to an output $\{\tilde{y}_t\}$. They are consisted by layers that are connected with weights. These layers can transform the data and introduce non-linear features. In the relevant literature, hundreds of NNs architectures have been presented, each with its unique characteristics. In this study, we will focus on the most popular NNs architectures applied in financial forecasting and trading, namely the MLP, RBF, HONN and RNN.

2.1 MLP

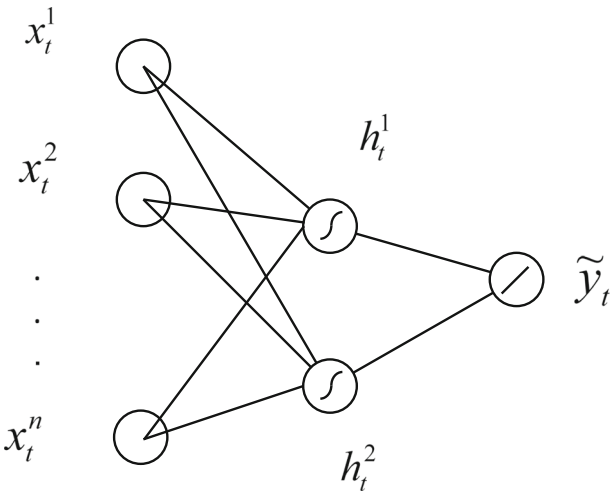
MLPs are the most popular and simple form of a NN. It is consisted by three different types of layers. The input, the hidden and the output layers. The input layer includes the inputs nodes and each node corresponds to a different explanatory variable. Input nodes incorporate simple linear functions such as the sum. The output layer is usually formed with one node which represents the output of the NN. The hidden layers, separate the input from the output layer and define the amount of complexity the model is capable of fitting. The number of hidden layers and the number of nodes in each layer are based on the complexity of the model under study and a trial and error approach. In financial application, the hidden nodes usually contain the sigmoid function. In the cases where more than one hidden layer exists, MLPs are also called Deep Learning NNs.

The network process information based on a feed-forward approach. The input nodes contain the explanatory variables. Each input node is connected with all the hidden nodes of the first hidden layer. These connections are weights which are trained by the NN. The information in the hidden nodes is passed through a nonlinear activation function and transferred to all nodes of the next hidden layer. This process is repeated until we reach the output

layer, which represents the result of the network. MLPs are trained by adjusting the weights (the connections between the nodes) through a learning algorithm, the backpropagation of errors[1] (Shapiro (2000)). This 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 (early stopping). This is achieved by dividing the in-sample dataset to two subsets, the training and the test sets. MLPs are trained in the training set and their performance is checked regularly in the test subset. The training stops when the error between the estimated and the actual values in the test subset, starts to increase. The training subset usually includes the first 4/5 of the in-sample and the test subset the last 1/5. A MLP with a single hidden layer is presented in Fig. 1:

Figure 1: MLP with a single hidden layer and two hidden nodes.



where x_t^i (i = 1…n) are the model inputs at time t, h_t^i (i = 1 or 2) are the hidden nodes outputs at time t, $\tilde{y}_t$ is the MLP model output at time t

$$\text{is the transfer sigmoid function: } S(x) = \frac{1}{1+e^{-x}} \tag{1}$$

$$\text{is a linear function: } F(x) = \sum_i x_i \tag{2}$$

The error function to be minimised is : $$E(u_{jk}, w_j) = \frac{1}{T}\sum_{t=1}^{T}\left(y_t - \tilde{y}_t(u_{jk}, w_j)\right)^2 \tag{3}$$

where y_t being the target value, T is the number of iterations, u_{jk} is the network weight between input j and hidden node k and w_k is the network weight between the hidden node k and the single output node.

[1] Backpropagation is the most common learning algorithm in MLPs and the most commonly used type in financial time series forecasting (Kaastra and Boyd 1996).

As discussed before, MLPs are trained by updating the weights (u_{jk} and w_k) in a series of iterations with the objective of minimizing Eq. (3). The main advantage of MLPs is their simple architecture and their flexibility in terms of complexity. The universal approximation theorem states that every continuous function that maps intervals of real numbers to some output interval of real numbers can be approximated closely by a MLP with just one hidden layer (Hornik et al. 1989). On the other hand, MLPs training can be slow when the number of hidden nodes and layers is large and often stuck to a local optima.

2.2 RBF

RBF networks were introduced by Broomhead and Lowe (1988) as an approach to improve accuracy of MLPs while decreasing training time and complexity. Their architecture is almost identical with the single hidden layer MLPs (see, Fig. 1). The difference is that the weights between the input and the hidden layer are fixed to 1 and the function of the hidden nodes is:

$$\phi(x) = e^{-\frac{\|x-C\|^2}{2\sigma^2}} \tag{4}$$

where C is a vector indicating the center of the Gaussian Function and σ is a value indicating its width. These variables are determined by the training process. When presented with the x input vector, a hidden neuron computes the Euclidean distance of the test case from the neuron's center point and then applies the RBF kernel function to this distance using the spread values. The resulting value is passed to the summation (output) layer. In the literature, there is a wealth of methods to select the center and the width of Eq. (4). The most common approaches are the K-mean clustering, orthogonal least squares or to select them randomly from a subset (Yingwei et al. 1998). The first two approaches are computationally intensive procedures that do not always reach to the optimal solution. In this study we are interested in generating a large number of RBFs and evaluating them through a MHT framework. Thus, we randomly chose the centres we define as widths the average distance between the chosen centres. This ensures that the hat the individual RBFs are neither too wide, nor too narrow, for the given training data. The computation of the optimal weights between the neurons in the hidden layer and the summation layer is done using ridge regression (Orr 1996).

RBFs are easily designed, have good generalization and show strong tolerance to input noise (Yu et al. 2011). They are also universal function approximators (Park and Sandberg 1991).

2.3 HONN

HONN or Tensor Networks were introduced from Giles and Maxwell (1987). They are feedforward networks characterized from their simplicity and short computational time. The major advantage of HONNs compared to their NNs counterparts is that they are able to provide some rationale for the simulations they produce and thus can be regarded as "open box" rather than "black box" (Zhang 2009).

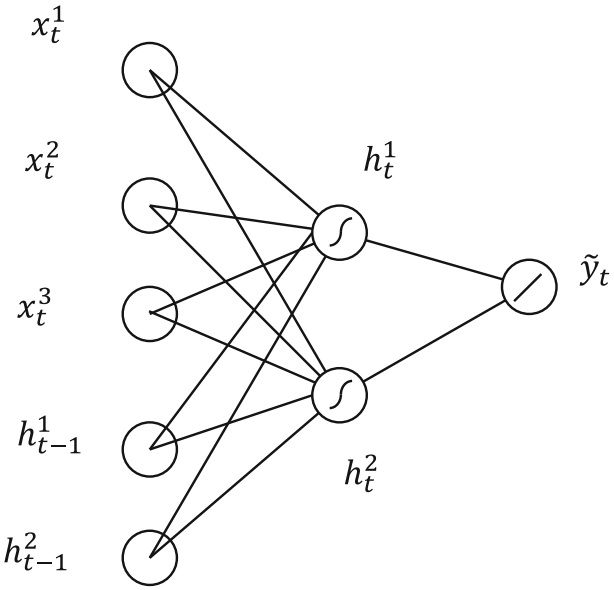
A HONN is consisted by only two layers (inputs and output). The input nodes are consisted by the explanatory variables and polynomials and trigonometric functions of them. In the output node a sigmoid function is applied. HONNs use joint activation functions, a technique that reduces the need to establish the relationships between inputs when training. It also reduces the number of free weights, which means that HONNS are faster to train than similar MLPs. They are able to simulate higher frequency and higher order non-linear data, and

consequently provide superior estimations of the underlying patterns compared to the more classical NN architectures. On the other hand, because the number of inputs can be very large for higher order architectures, orders of 4 and over are rarely used. HONNs are widely applicable in the fields of pattern recognition and associative recall. However, unlike the MLPs and RBFs, they are not universal function approximators.

2.4 RNN

RNN is a class of artificial NN where at least one connection is formed between the nodes form a directed cycle. This allows them to exhibit dynamic temporal behavior. In the previous architectures the information is always fed forward (from the inputs to the hidden layer and then to the output layer). In RNN, the input nodes contain, apart from the explanatory variables, what the network has achieved in the previous period. In other words, the decision that a RNN is reaching at time t is affected not only from the new data but also from what the network has achieved on time t − 1. Figure 2 presents a RNN with three inputs and two hidden nodes.

Figure 2: RNN with a three inputs and two hidden nodes.



where x_t^i (i = 1…n) are the model inputs at time t, h_t^i (i = 1 or 2) are the hidden nodes outputs at time t, $\tilde{y}_t$ is the MLP model output at time t

$$\text{is the transfer sigmoid function: } S(x) = \frac{1}{1+e^{-x}} \tag{1}$$

$$\text{is a linear function: } F(x) = \sum_i x_i \tag{2}$$

The error function to be minimised is: $E(u_{jk}, w_j) = \frac{1}{T}\sum_{t=1}^{T}\left(y_t - \tilde{y}_t(u_{jk}, w_j)\right)^2$ (3)

The weights now determine how much importance to accord to both the present input and the past hidden state. The error will return via backpropagation and be used to adjust the RNN weights until the error cannot go any lower. This additional memory naturally leads to a superior performance in pattern recognition compared to feedfoward networks (Elman 1990). On the other hand, RNNs require substantially more connections and more memory in simulation, thus resulting in a substantial increase in computational time (Tenti 1996).

2.5 Hyperparameters

The performance of NNs is heavily dependent on the choice of their inputs and their hyperparameters (Donaldson and Kamstra (1996) and Lisboa and Vellido (2000)). The choice of the inputs is based on the nature of the series under study and the practitioners' knowledge/beliefs. In the literature, researchers often apply Principal Component Analysis (PCA), metaheuristics or other econometrics [see amongst others, Kaastra and Boyd (1996) and Sermpinis et al. (2013)]. Hyperparameters are the variables that the practitioner needs to set before optimizing its series with NNs. Setting these variables can been seen as model selection. Similarly with the inputs, they are set either based on objective criteria or based on a separate statistical or heuristics procedure (such as grid search, cross-validation or genetic algorithms). For example, the hyperparameters of MLP are the learning rate (how fast the NN is converging), the number of the hidden layers and the number of the hidden nodes at each layer, the momentum (how many steps ahead the network is looking) and the size of training blocks (how often the network will check if it has reached the global optima in the test sub-set). For RBF, the practitioner also needs to set the center and the width of the Gaussian Function. Different sets of hyperparamters naturally lead to different forecasts and empirical evidence. In this study, we examine 50 MLPs, 50 RBFs, 50 HONNs and 50 RNNs (else l is set to 200). The NNs differ between them in terms of inputs and hyperparameters. The range of the hyperparameters is set by the guidelines of Hornik et al. (1989), Tenti (1996) and Yu et al. (2011), while the inputs are combinations of autoregressive lags of the target series.

2.6 Data-snooping

Data snooping (also p-hacking, data dredging, looking elsewhere…) is a statistical bias that appears when exhaustively searching for combinations of variables. In such a case, the probability that a result arose by chance grows with the number of combinations tested (White 2000). In other words, if you try hard enough you will find good results, but this does not mean that your results can be generalized. As discussed before, the performance of NNs depends on the choice of their inputs and their hyperparameters. Practitioners experiment with their dataset for their optimal setting. This inadvertently leads to models that suffer from data snooping. These models might have a good in-sample and out-of-sample performance but when applied on a different dataset, their accuracy will be reduced dramatically. In order to test for the data snooping bias, the researcher needs to apply MHT framework. Under these framework, it is possible to check simultaneously if the performance of each model (from a set of competing models) is statistically different from zero or not. In order to test for data snooping, one needs to control either the Family Wise Error Rate (FWER) (the probability of making one or more false discoveries, or type I errors) or the FDR (the probability of having a rate of false discoveries). The FDR provides a less stringent control of the Type I errors

and is more suitable for large and multiple datasets (Romano and Wolf 2007). Thus, this is the approach that we follow.

In FDR, we try to control the expected false discovery proportion amongst all discoveries. In a more formal setting we control FDR at a level γ if $FDR \equiv E(false\ discovery\ proportion) \leq \gamma$. The level γ is defined by the practitioner and represents the percentage of false discoveries that he/she is willing to accept. In this study, we adapt the FDR procedure of Storey (2002) and Bajgrowicz and Scaillet (2012). Under these frameworks, we measure the performance in terms of annualized return of each NN and we estimate the return matrix G of all models. Then, using the stationary bootstrap method of Politis and Romano (1994) we resample the returns G and create $b = 1, \ldots, 1000$ bootstrap realizations G_{jb} for each NN return series. Based on the original and bootstrapped returns, a p value is estimated for each NN. Then starting from the smallest p-value, we estimate:

$$\widehat{\pi_0}(\lambda) = \frac{\#\{p_j > \lambda; j = 1, \ldots, l\}}{l(1-\lambda)} \tag{5}$$

where l is the number of NNs and λ is a used defined parameter. The procedure is stopped when the desired FDR level is reached. Following Bajgrowicz and Scaillet (2012), we set λ as 0.6 while we stopped the procedure when $\widehat{\pi_0}(\lambda)$ reaches 10%. In other words from our identified NNs as genuine, the 10% of them are false discoveries.

For more details on the FDR algorithm, we refer the reader to Storey (2002) and Bajgrowicz and Scaillet (2012). In this study, we examine 50 MLPs, 50 RBFs, 50 HONNs and 50 RNNs (else l is set to 200). The NNs differ between them in terms of inputs and hyperparameters. The range of the hyperparameters is set by the guidelines of Hornik et al. (1989), Tenti (1996) and Yu et al. (2011), while the inputs are combinations of autoregressive lags of the target series.

2.7 Portfolio construction

All NNs are applied to the task of forecasting the logarithmic returns of the DJIA, the FTSE 100 and the NIKKEI 225 indices. Following the FDR approach we identify the genuine profitable NNs and we construct portfolios. Based on each model's forecast a long, short or a neutral trading signal is generated. We invest an equal proportion of wealth to the signals generated by each individual rule. After pooling the signals, we invest a proportion of our portfolio to the risk free rate while the remaining to a long or short position.

For example, imagine that the FDR identifies 30 NNs as genuine. From these 20 generate a buy signal, 5 a short signal and 5 a neutral signal. After pooling, we obtain 15 buy signals and 10 neutral signals. Hence, we invest 60% of our wealth in the index while the remaining 40% is invested on the risk free rate.[2]

3 Empirical section

The purpose of this study is to examine the profitability of a large pool of NNs through a multiple hypothesis testing framework. As discussed before a series of NNs are applied to the task of forecasting the 1 day ahead logarithmic return of the DJIA, the NASDAQ 100 and

[2] As risk free rate we consider the effective federal funds rate. This is the interest rate at which depository institutions trade federal funds (balances held at Federal Reserve Banks) with each other overnight.

Table 1 Summary statistics

Period	Index	DJIA	NASDAQ 100	NIKKEI 225
1/1/2007–31/12/2008	Mean	− 0.0007	− 0.0013	− 0.0007
	Standard deviation	0.0181	0.0223	0.0206
	Skewness	0.1063	− 0.3841	− 0.0323
	Kurtosis	10.2447	10.1993	9.0919
	Jarque–Bera (*p* value)	0	0	0
	ADF (*p* value)	0.0001	0.0001	0.0001
1/1/2016–31/12/2017	Mean	0.0006	0.0004	0.0006
	Standard deviation	0.0085	0.0131	0.0085
	Skewness	− 0.6914	− 0.2991	− 0.6914
	Kurtosis	6.6337	10.7754	6.6337
	Jarque–Bera (*p* value)	0	0	0
	ADF (*p* value)	0.0001	0.0001	0.0001

The values in the Table represent the summary statistics of the series under study

the NIKKEI 225 stock indices. Then, the genuine profitable models are identified through the FDR approach and portfolios are constructed based on their forecasts.

We examine two separate datasets, namely the periods of 2007–2008 and 2016–2017. The first dataset is dominated by the global financial crisis, while the more recent one is free from financial stress factors. Thus, we will also examine if the performance of our models is affected by the side-effects of the recent financial crisis.

3.1 Dataset

The summary statistics of the logarithmic returns of the three series are presented in Table 1.

We note that all series exhibit small skewness and high kurtosis. The Jarque–Bera statistic confirms that all return series are non-normal at the 99% confidence level, while the Augmented Dickey-Fuller (ADF) reports that the null hypothesis of a unit root is rejected at the 99% statistical level in all cases.

We set as in-sample the first 6 months of each dataset and as out-of-sample the following month. The estimation is rolled forward by 1 month till the end of the dataset. In other words for the 2007–2008 dataset, the first in-sample is 1st January 2007 to 31st June 2007 and the first out-of-sample is 1st July 2007 to 31st July 2007. Then, after we execute our exercise, the sample is rolled forward 1 month (1st February 2007 to 31st July 2007 in-sample and 1st August 2007 to 31st August 2007 acts as out-of-sample). Thus, 18 consecutive exercises are conducted for each of the two datasets. This procedure protects our findings for microstructure events and structural breaks within our series and add robustness to our findings.

3.2 In-sample performance

In this Section, the in-sample performance of our NNs is presented. Table 2 reports the average percentage of NNs that have been identified as significant in the in-samples for the three indices under study.

Table 2 Significant NNs

Period	Index	DJIA (%)	NASDAQ 100 (%)	NIKKEI 225 (%)
1/1/2007–31/12/2008	MLP	12	10	12
	RBF	8	14	6
	HONN	16	14	**16**
	RNN	**18**	**20**	**16**
1/1/2016–31/12/2017	MLP	4	8	10
	RBF	8	6	10
	HONN	8	12	4
	RNN	**16**	**14**	**16**

The table presents the average percentage of NNs identified as significant in the in-samples for the three indices. The values in bold represent the higher percentage at each case

Table 3 In-sample trading performance

Period	Index	DJIA	NASDAQ 100	NIKKEI 225
1/1/2007–31/12/2008	MLP	7.29% (0.52)	6.09% (0.55)	7.72% (0.71)
	RBF	10.01% (1.09)	9.39% (1.00)	5.10% (0.52)
	HONN	8.77% (0.86)	9.01% (0.93)	**10.05% (1.18)**
	RNN	**10.16% (1.19)**	**9.50% (1.02)**	9.49% (0.09)
1/1/2016–31/12/2017	MLP	8.12% (0.68)	5.22% (0.43)	7.00% (0.76)
	RBF	7.93% (0.75)	6.20% (0.52)	6.01% (0.59)
	HONN	6.40% (0.57)	7.41% (0.60)	8.30% (0.78)
	RNN	**8.28% (0.79)**	**8.29% (0.74)**	**8.42% (0.80)**

The values in the Table are the average annualized returns after transaction costs of the significant portfolios. The values in the parenthesis are the related Sharpe Ratios. The values in bold represent the higher values at each case

We note that the percentage of RNN is higher than the ones of their counterparts. The RNN specification seems to produce more significant NNs on average in the in-samples of both datasets. It is also interesting to note that the percentage of significant models is higher in the first dataset, which is characterised from the global crisis. The next Table presents the average in-sample trading performance of the FDR portfolios constructed by the identified as significant NNs (Table 3). Transaction costs are estimated to 20 basis points per trade (see, https://www.interactivebrokers.com).[3]

We note that the RNN portfolios are the more profitable in terms of annualized return and Sharpe ratio. HONN present a strong trading performance that in several cases is better that the ones of RBF and MLP. These results are surprising considering HONN simplistic architecture. It is also interesting to note that the trading performance of all models worsens on the second dataset. It seems that the financial stress conditions that dominate the first sample assisted our NNs to achieve even better trading performance.

[3] Stock indices can be traded with three separate ways. A trader can buy or short all stocks that constitute the index, however this might lead to substantial transaction costs and liquidity issues. The second way is through futures and the third is through an Exchange Traded Fund (ETF) on the index. The transaction costs considered in this study are an approximation of the costs that large institutional investors will be facing if they decided to trade the stock indices under study with futures or ETFs.

Table 4 Out-of-sample trading performance

Period	Index	DJIA	NASDAQ 100	NIKKEI 225
1/1/2007–31/12/2008	MLP	3.18% (0.26)	3.11% (0.23)	4.58% (0.34)
	RBF	4.89% (0.39)	4.26% (0.37)	4.10% (0.30)
	HONN	4.07% (0.35)	**5.35% (0.53)**	5.91% (0.61)
	RNN	**6.06% (0.58)**	5.28% (0.49)	**5.95% (0.62)**
1/1/2016–31/12/2017	MLP	3.00% (0.30)	2.84% (0.19)	3.47% (0.33)
	RBF	4.34% (0.41)	3.91% (0.36)	4.46% (0.47)
	HONN	3.12% (0.33)	4.01% (0.39)	3.52% (0.35)
	RNN	**5.72% (0.54)**	**4.28% (0.42)**	**5.24% (0.51)**

The values in the Table are the annualized returns after transaction costs of the significant portfolios. The values in the parenthesis are the related Sharpe Ratios. The values in bold represent the higher values at each case

3.3 Out-of-sample performance

The in-sample performance of all portfolios is positive but what has the outmost important in any forecasting/trading exercise is the out-of-sample. As discussed before, the in-sample is set to 6 months and the out-of-sample to 1 month. This exercise is repeated 18 times for each of the two datasets by rolling forward 1 month the sample. Table 4 presents the average out-of-sample trading performance.

Table 4 reveals a similar trend for our portfolios. RNN clearly outperform its benchmarks in both datasets. HONN present the second best performance in 5 out of 6 cases and the best in one case. The trading performance of all models is better in the first dataset that covers the recent financial crisis. It seems that the additional memory nodes of RNN lead not only to an increased percentage of significant models but also to a better trading performance.

3.4 Financial leverages

In this Section, two financial leverages are applied. The first one is inspired by the increased trading performance of our models in the first dataset. It is based on the level of financial stress of the stock indices under study. The second one is based on the trading notion that when volatility is high is better to stay out of the market. A similar leverage has been introduced by Dunis et al. (2010, 2011) and Sermpinis et al. (2014) and is applied in practice by Société Générale and Horus Partner Wealth Management (ISIN: FR0011176544). In periods of high volatility, the markets are becoming erratic and the risk unbearable. This fact has been observed in a series of empirical studies and has led to a series of financial instruments (see amongst others, Dunis and Miao (2006), Bertolini (2010) and Sermpinis et al. (2014)).

In both leverages, we multiple or divide the percentage of wealth with which we long or short the index. At all cases, this percentage ranges from 100% of our wealth to 0%. In the latter case, 100% of our wealth is invested in the risk-free rate.

3.4.1 Financial stress leverage

The systemic financial stress represents disruptions in the normal functioning of financial markets. It rises when financial or economics factors deviate from the normal trends. Naturally

Table 5 Out-of-sample trading performance with financial stress leverage

Period	Index	DJIA	NASDAQ 100	NIKKEI 225
1/1/2007–31/12/2008	MLP	5.49% (0.40)	5.19% (0.45)	7.40% (0.68)
	RBF	7.12% (0.64)	6.02% (0.52)	5.71% (0.45)
	HONN	6.96% (0.62)	7.58% (0.71)	6.18% (0.72)
	RNN	**9.52% (0.95)**	**8.18% (0.84)**	**8.81% (0.89)**
1/1/2016–31/12/2017	MLP	5.37% (0.59)	4.54% (0.32)	6.51% (0.50)
	RBF	6.62% (0.64)	5.28% (0.40)	5.92% (0.51)
	HONN	4.41% (0.39)	5.63% (0.44)	6.35% (0.47)
	RNN	**6.78% (0.69)**	**7.09% (0.71)**	**7.40% (0.77)**

The values in the Table are the annualized returns after transaction costs of the significant portfolios. The values in the parenthesis are the related Sharpe Ratios. The values in bold represent the higher values at each case

financial stress levels are high when there is a financial crisis. We note from Tables 2 and 3 that the trading performance of our portfolios is higher in the first period that covers the recent global finance crisis. This motivates us to generate an adaptive financial stress leverage.

As a first step, we select a metric for the level of financial stress. This is the Office of Financial Research (OFR) stress index. The OFR index is constructed from 33 financial market variables, such as yield spreads, valuation measures, and interest rates. It is considered to provide a daily snapshot of the level of financial stress. It covers the USA and other advanced economies.[4] Secondly, we match our portfolios with the OFR stress levels for USA (DJIA and NASDAQ 100) and other advanced economies (NIKKEI 225). Then, based on the signal of our portfolio and the level of financial stress in the previous month, we apply a leverage. The level of this leverage is based on:

$$\text{Average level of financial stress (5 days)} - \text{Average level of financial stress (1 month)}$$

If the average level of financial stress in the last 5 days is higher than relevant level of the last month, we consider that the level of stress is increasing. Then, on the following day we double the percentage of our wealth invested to the trading signal generated from our portfolios. For example, let us consider that our models for tomorrow indicate that we should invest 40% on the index and 60% on the risk free rate. If the level of stress on the previous 5 days is higher than the one of the previous year, we will invest 80% on the index and 20% on the risk free rate. On the opposite scenario, we invest to the index as usual. In other words, when financial stress levels are increasing we invest more based on our NNs and less on the risk free rate. Table 5 presents the relevant performance of our models after the application of the financial stress leverage.

We note that the performance of all portfolios is increased. These results confirm the findings of Sect. 3.3 and Table 4. NNs seems to work better when financial stress conditions dominate the stock markets. The ranking of our models is not changed and RNN continue to provide more profitable forecasts. There are cases where the performance of our portfolios

[4] The finance literature is rich with financial stress indices. In our application, we need an index that is updated on a daily basis and covers USA and Japan. The OFR has these two characteristics. However, it should be noted that in the other advanced economies are included not only Japan but also the Eurozone countries. Although the fit is not perfect for this case, it is the closest to the best of our knowledge. Other stress indices have lower frequency or focus on a single economy.

Table 6 Sub-periods and leverages

	Extremely low vol.	Medium low vol.	Lower low vol.	Lower high vol.	Medium high vol.	Extremely high vol.
Leverage	2.5	2	1.5	1	0.5	0

is doubled. This happens more notably in the first dataset. These results further highlight the effectiveness of NNs in trading and more especially when financial markets are in turbulence.

3.4.2 Financial volatility leverage

In order to further improve the trading performance of our models we apply a leverage based on RiskMetrics[5] 1 day ahead volatility forecasts. The intuition of the strategy is to avoid trading when volatility is very high and exploiting days when the volatility is relatively low. The leverage scales the relevant position sizes inversely to the recent risk behaviour.

As a first step, we forecast with RiskMetrics the 1 day ahead realised volatility of our series. Then, the average the average (μ) difference between the actual volatility in day t and the forecasted for day t + 1 and its 'volatility' (measured in terms of standard deviation σ) are calculated over the last 6 months is estimated. Based on these estimations, we define six sub-periods, ranging from periods with extremely low volatility to periods experiencing extremely high volatility. The periods where the difference is between the μ plus one σ are classified as 'Lower High Vol. Periods'. Similarly, 'Medium High Vol.' (between $\mu + \sigma$ and $\mu + 2\sigma$) and 'Extremely High Vol.' (above $\mu + 2\sigma$) periods can be defined. Periods with low volatility are defined following the same approach, but with a minus sign. The leverages and the sub-periods are presented on Table 6.

Based on these sub-periods, we invest more or less each day on the signal that our NNs portfolios generate. For example, on the first day of the out-of-sample we check our RiskMetrics volatility forecast at which sub-period falls. As mentioned before, these sub-periods are based on the last 6 months. Let us assume that the volatility forecast falls on the extreme low volatility period and that on the same day we have to invest based on our portfolios 20% on the index (long position) and 80% on the risk-free rate. Based on our leverage (2.5), we will invest 50% on the index and 50% on the risk-free rate. On the other hand, if on the same day the volatility forecast is falling instead to the extreme high volatility period, 100% of our wealth would have been invested on the risk free rate (leverage 0).

The parameters of our strategy (μ and σ) are updated every month by rolling forward the estimation period. For example, for the first month of the out-of-sample, μ and σ are computed based on the 6 months of the in-sample. For the following month, the two parameters are computed based on the last 6 months of our dataset (the last 5 months of the in-sample and the first month of the out-of-sample). Table 7 presents the trading performance of our NN portfolios with the time varying volatility leverage.

The time varying volatility leverage is doubling the annualized returns of our NNs portfolios in most cases. The RNN continue to outperform its counterparts in terms of annualized returns and Sharpe Ratios. Comparing the results of Tables 6 and 7, we note that the volatility leverage is more successful than the one based on the financial stress. Unlike the previous sets of results, NNs seem to perform equally well in both datasets.

[5] A brief description of the RiskMetrics volatility model is provided in "Appendix".

Table 7 Out-of-sample trading performance with financial stress leverage

Period	Index	DJIA	NASDAQ 100	NIKKEI 225
1/1/2007–31/12/2008	MLP	6.02% (0.53)	6.71% (0.59)	7.35% (0.74)
	RBF	6.61% (0.60)	6.63% (0.57)	6.97% (0.70)
	HONN	5.92% (0.49)	6.66% (0.57)	6.70% (0.67)
	RNN	**7.40% (0.77)**	**8.29% (0.89)**	**9.33% (1.01)**
1/1/2016–31/12/2017	MLP	6.48% (0.56)	5.81% (0.48)	7.59% (0.77)
	RBF	6.12% (0.55)	6.47% (0.64)	6.15% (0.64)
	HONN	5.60% (0.47)	6.52% (0.66)	6.54% (0.68)
	RNN	**7.53% (0.81)**	**8.60% (0.93)**	**8.00% (0.88)**

The values in the Table are the annualized returns after transaction costs of the significant portfolios. The values in the parenthesis are the related Sharpe Ratios. The values in bold represent the higher values at each case

4 Concluding remarks

This study explores the profitability of 200 NNs through a MHT framework. 50 MLPs, 50 RBFs, 50 HONNs and 50 RNNs are generated and applied to the task of forecasting the 1 day ahead logarithmic return of three stock indices. The NNs differ in terms of hyperparameters and inputs. The genuine NNs are identified through a FDR approach and portfolios are constructed. Their profitability is examined in terms of annualized return after transaction costs and Sharpe Ratio. Two financial leverages are introduced based on the level of financial stress and the financial volatility respectively. Their application aims to test whether they can improve the portfolios profitability.

In terms of the results, we note that the FDR approach identifies a large percentage of NNs as genuine. The profitability of NNs is positive in both in-sample and out-of-sample. From the four NN families under study, we note that RNN has the higher percentage as genuine forecasts and the better trading performance. The performance of all portfolios is better in the sample that covers the recent global financial crisis. The application of the two financial leverages is successful and doubles the profitability of NNs in the majority of cases.

These results explain the popularity of NNs in trading desks and in wealth management. They are capable of producing high profitability that is further enhanced in periods of uncertainty and financial stress. Profitability enhancement can be achieved with the help of simple financial leverages that try to exploit financial stress and volatility. From the four NN families under study, the most promising architecture is the RNN. Their long memory and the fact that they incorporate past errors in their architecture give an advantage compared to their counterparts. Finally, our results should go forward convincing academics and professional on the utility of NNs in financial trading.

Appendix: RiskMetrics volatility

The RiskMetrics volatility model is one of the simpler methods to forecast the financial volatility. Nevertheless, it is popular among quantitative finance professionals. Derived from the from the GARCH(1,1) model but with fixed coefficients, the daily RiskMetrics volatility forecast for day t is estimated as:

$$RiskMetrics\ volatility_t^2 = 0.94\sigma_{t-1}^2 + 0.06R_{t-1}^2 \quad (6)$$

where σ_{t-1}^2 is the volatility of the index at day t − 1 and R_{t-1}^2 is the squared logarithmic return of the index on the same day.

References

Adeodato, P., Arnaud, A., Vasconcelos, G., Cunha, R., & Monteiro, D. (2011). MLP ensembles improve long term prediction accuracy over single networks. *International Journal of Forecasting, 27*(3), 661–671.

Andreou, P. C., Charalambous, C., & Martzoukos, S. H. (2008). Pricing and trading European options by combining artificial neural networks and parametric models with implied parameters. *European Journal of Operational Research, 185*(3), 1415–1433.

Bagheri, A., Peyhani, H. M., & Akbari, M. (2014). Financial forecasting using ANFIS networks with quantum-behaved particle swarm optimization. *Expert Systems with Applications, 41*(14), 6235–6250.

Bahrammirzaee, A. (2010). A comparative survey of artificial intelligence applications in finance: Artificial neural networks, expert system and hybrid intelligent systems. *Neural Computing and Applications, 19*(8), 1165–1195.

Bajgrowicz, P., & Scaillet, O. (2012). Technical trading revisited: False discoveries, persistence tests, and transaction costs. *Journal of Financial Economics, 106*(3), 473–491.

Bekiros, S. D., & Georgoutsos, D. A. (2008). Direction-of-change forecasting using a volatility-based recurrent neural network. *Journal of Forecasting, 27*(5), 407–417.

Bertolini, L. (2010). *Trading foreign exchange carry portfolios*. PhD thesis, Cass Business School, City University London.

Boston Consulting Group. (2016). *The factory of the future: The ghost in the machine, Boston Consulting Group*. Online report.

Broomhead, D. S., & Lowe, D. (1988). Radial basis functions, multi-variable functional interpolation and adaptive networks (No. RSRE-MEMO-4148). Royal Signals and Radar Establishment Malvern.

Cavalcante, R. C., Brasileiro, R. C., Souza, V. L., Nobrega, J. P., & Oliveira, A. L. (2016). Computational intelligence and financial markets: A survey and future directions. *Expert Systems with Applications, 55,* 194–211.

Chui, M., Manyika, J., & Miremadi, M. (2016). Where machines could replace humans—and where they can't (yet). *Mckinsey Quarterly, 7*.

Donaldson, R. G., & Kamstra, M. (1996). Forecast combining with neural networks. *Journal of Forecasting, 15*(1), 49–61.

Dunis, C., Laws, J., & Sermpinis, G. (2010). Modelling and trading the EUR/USD exchange rate at the ECB fixing. *The European Journal of Finance, 16*(6), 541–560.

Dunis, C., & Miao, J. (2006). Volatility filters for asset management: An application to managed futures. *Journal of Asset Management, 7*(3–4), 179–189.

Dunis, C. L., Laws, J., & Sermpinis, G. (2011). Higher order and recurrent neural architectures for trading the EUR/USD exchange rate. *Quantitative Finance, 11*(4), 615–629.

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

Fulcher, J., Zhang, M., & Xu, S. (2006). The application of higher-order neural networks to financial time series. In J. Kamruzzaman, R. Begg, & R. Sarker (Eds.), *Artificial neural networks in finance and manufacturing*. Hershey, PA: Idea Group, London.

Giles, C. L., & Maxwell, T. (1987). Learning, invariance, and generalization in high-order neural networks. *Applied Optics, 26*(23), 4972–4978.

Guresen, E., Kayakutlu, G., & Daim, T. U. (2011). Using artificial neural network models in stock market index prediction. *Expert Systems with Applications, 38*(8), 10389–10397.

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

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

Krauss, C., Do, X. A., & Huck, N. (2017). Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500. *European Journal of Operational Research, 259*(2), 689–702.

Lisboa, P. J. G., & Vellido, A. (2000). Business applications of neural networks, vii–xxii. In P. J. G. Lisboa, B. Edisbury, & A. Vellido (Eds.), *Business applications of neural networks: The state-of-the-art of real-world applications*. Singapore: World Scientific.

Lo, A. W. (2004). The adaptive markets hypothesis. *Journal of Portfolio Management, 30*(5), 15–29.

Maren, A. J., Harston, C. T., & Pap, R. M. (2014). *Handbook of neural computing applications*. Cambridge: Academic Press.
Matías, J. M., & Reboredo, J. C. (2012). Forecasting performance of nonlinear models for intraday stock returns. *Journal of Forecasting, 31*(2), 172–188.
Orr, M. J. L. (1996). *Introduction to radial basis function networks*. Centre for Cognitive Science. Scothland: Edinburgh University. http://www.anc.ed.ac.uk/&mjo/rbf.html.
Panda, C., & Narasimhan, V. (2007). Forecasting exchange rate better with artificial neural network. *Journal of Policy Modeling, 29*(2), 227–236.
Park, J., & Sandberg, I. W. (1991). Universal approximation using radial-basis-function networks. *Neural Computation, 3*(2), 246–257.
Politis, D. N., & Romano, J. P. (1994). The stationary bootstrap. *Journal of the American Statistical Association, 89*(428), 1303–1313.
Romano, J. P., & Wolf, M. (2007). Control of generalized error rates in multiple testing. *The Annals of Statistics, 35*(4), 1378–1408.
Sermpinis, G., Stasinakis, C., & Dunis, C. (2014). Stochastic and genetic neural network combinations in trading and hybrid time-varying leverage effects. *Journal of International Financial Markets, Institutions and Money, 30,* 21–54.
Sermpinis, G., Theofilatos, K., Karathanasopoulos, A., Georgopoulos, E. F., & Dunis, C. (2013). Forecasting foreign exchange rates with adaptive neural networks using radial-basis functions and particle swarm optimization. *European Journal of Operational Research, 225*(3), 528–540.
Shapiro, A. F. (2000). A Hitchhiker's guide to the techniques of adaptive nonlinear models. *Insurance, Mathematics and Economics, 26*(2), 119–132.
Storey, J. D. (2002). A direct approach to false discovery rates. *Journal of the Royal Statistical Society: Series B (Statistical Methodology), 64*(3), 479–498.
Tenti, P. (1996). Forecasting foreign exchange rates using recurrent neural networks. *Applied Artificial Intelligence, 10*(6), 567–581.
White, H. (2000). A reality check for data snooping. *Econometrica, 68*(5), 1097–1126.
Yang, J., Cabrera, J., & Wang, T. (2010). Nonlinearity, data-snooping, and stock index ETF return predictability. *European Journal of Operational Research, 200*(2), 498–507.
Yang, J., Su, X., & Kolari, J. W. (2008). Do Euro exchange rates follow a martingale? Some out-of-sample evidence. *Journal of Banking & Finance, 32*(5), 729–740.
Yingwei, L., Sundararajan, N., & Saratchandran, P. (1998). Performance evaluation of a sequential minimal radial basis function (RBF) neural network learning algorithm. *IEEE Transactions on Neural Networks, 9*(2), 308–318.
Yu, H., Xie, T., Paszczynski, S., & Wilamowski, B. M. (2011). Advantages of radial basis function networks for dynamic system design. *IEEE Transactions on Industrial Electronics, 58*(12), 5438–5450.
Zhang, G., Patuwo, B. E., & Hu, M. Y. (1998). Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting, 14*(1), 35–62.
Zhang, M. (2009). *Artificial higher order neural networks for economics and business*. Hershey: IGI Global.