A Neural Networks Filtering Mechanism For Foreign Exchange Trading Signals

Alex Kayal

Abstract—
Neural Networks have been successfully used in several financial applications. In the stock market and foreign exchange domains, Neural Networks have been used with considerable success to predict the future prices of stocks and currency pairs, their rate of return, risk analysis, and several other features that might be of benefit. In this paper, we present a methodology to filter the high-frequency signals of a rule-based foreign exchange trading strategy, through a neural network-based, intelligent selection mechanism. We then compare the results vs. a random selection mechanism and again vs. the overall signal pool, in terms of profit and correctness. We can clearly show that the neural network filtering approach yields a better performance than its random-based counterpart.

I. INTRODUCTION

A. Background

Neural Networks (NNs) have been used in several financial and non-financial fields, such as image processing, pattern recognition, biomedical applications[1], parallel computing and optimization[2], in addition to a large number of financial applications, of which Financial Forecasting is most relevant to this work. NNs are characterized by a non-linear, brain-like structure, and the ability to learn complex relationships amongst several related and non-related elements. These characteristics provide NNs with the ability to be a candidate for tasks such as the prediction of the movements of financial instruments, the analysis of chaotic time-series, and the optimization of trading strategies.

To further expand on this idea, NNs can be used to optimize the parameters in rule-based trading strategies, as well as a carefully select entry/exit signals. And while the majority of related works focus on the prediction of time-series elements themselves (for example, to estimate the expected closing price of a certain stock for the following day), this approach can be taken even further by using the learning and generalization abilities of Neural Networks to fine-tune a rule-based strategy in order to achieve a better performance. The approach is not entirely far from the train-and-predict methodology used in most feed-forward neural networks, as our network model is trained on a set of historical inputs consisting of prices and price-derived technical indicators, and a target data that consists of the results (a win or a loss) of the trading signals on which the training is to be performed. After training, the trained network is simulated on a set of new inputs and the results are compared to the actual, live-trading target results. The results are then used to determine whether a better performance was obtained in comparison to a random signal selection strategy.

The consistency of the selected trading signals is one of the good side-effects of this procedure. By that, we mean that the signals’ win-to-loss ratio will not be changing much as time goes on. Unfortunately, most rigid, rule-based trading strategies are discarded after a certain period of time, when they stop generating as much profit as they were. This happens due to the lack of flexibility in the parameters of the strategy itself - the rules that make up a traditional trading systems are usually few, simple, and unchangeable - and with the markets’ changing nature and momentum shifts, a strategy with such rules will start to show lower profits over time, then eventually turn into a loser - causing traders to look for a different strategy with a better recent track record, and the cycle goes on. A neural network-based optimization strategy will not only fine-tune the system for current use, but it will also make sure the strategy remains up to date for a long time, due to the constant updating it goes through, thus prolonging its lifespan. An intelligent selection of signals will also lower the average length of loss streaks, thus making a trading strategy more sustainable as well.

In the following sections, we start with a brief explanation of the underlying mechanism of both neural networks, technical analysis, and rule-based trading strategies, followed by a summary of relevant research on neural networks in financial applications. Then we present our model and a simulation of the network and strategy we have used for this work. Afterwards, we show the results of the training and simulation in a comparison to similar results generated by a random selection filter, using various benchmarks. We also compare the results of our neural model to the overall signal selection.

B. Methodology

The method in which this work was done solely focuses on testing. Feed Forward Neural Networks, along with non-sophisticated indicators (such as moving averages and oscillators) have been used in related works as we will see in Chapter 4. But since the idea present in this work - in this specific shape - has not been attempted before, then no real guidance to what type of neural networks and technical data can work best together in order to solve the problem at hand. MATLAB proved to be the perfect choice for the problem,

Alex Kayal is a masters student at the Royal Institute of Technology (KTH), Stockholm, Sweden. This master’s thesis corresponds to 20 weeks of full-time work.
with an extensive library of neural network models that can be flexibly edited to suit a specific problem, a wide range of technical indicators implemented as functions, a scripting language that is both easy and powerful, and excellent data visualization abilities.

The methodology followed here is closest to [10] and [13], being considered the most related. Yet still, there was no real guarantee that any of the tools used in those works were going to fit, and therefore the actual work was then to take an exploratory approach, to feed several combinations of data and indicators into different types of neural networks, then trying to settle down on the best few choices. Later, we fine-tune our parameters to try and get the lowest possible error out of the networks in both training and simulation phases. No general aim or purpose was formulated until this “exploration” has gone deep enough to find a meaningful, untackled problem in the stock/foreign exchange field that such networks are prepared to solve.

Testing started with simple networks and a small number of indicators, and went upward in terms of size and complexity, going through several combinations of network types and technical indicators. Though not very different results were obtained, it was still easy to identify a type/size of network that is most useful to this kind of work, and the most useful indicators - though with more testing, better choices could well be reached. The primary results proved that the idea was indeed solid, and to confirm the validity of the research, a comparison had to be done with a random model. This random model was created to filter the signals randomly rather than use a method to carefully decide good and bad signals. The aim was to obtain better than random results when testing the model, and that was generally what happened as we can see the full results in Chapter 7.

Choosing the data provided another slight problem as well, because too little data will make it difficult for the network to learn well, and too much might cause overfitting. Data size was experimented with and several data set sizes were tested until the we obtained the samples in Chapter 6. Data from after september 2008 was not chosen, the reason for that is the unusual trendiness for the data in times of financial crisis. Technical analysis performs much better in trendy situations, and using such data might lead us to a false result of extreme success. Major forex pairs were chosen, because of the availability and the great quality of data, as opposed to other less traded pairs.

II. NEURAL NETWORKS

A. Overview

A Neural Network is “A collection of parallel processors connected together in the form of a directed graph, organized such that the network structure lends itself to the problem being considered”[3]. Neural networks vary in structure, parameters, and purpose, from simple one layered perceptrons through radial basis networks and from feed-forward time-delay networks to self-organizing maps, all solving different tasks ranging from character recognition to clustering of elements of similar types.

The basic unit in a every neural network is a Neuron. A unit modeled after human braincells, and is essentially an elementary information processor[4]. A certain number of neurons form a layer, and a neural network consists of a number of those layers, of which the first layer is called the input layer, followed by one or a number of layers called the hidden layers, then a final layer called the output layer, out of which we obtain the output. A neuron is connected to others through weight-labeled links, and the data-value in the neuron is typically altered by a transfer function such as the linear, log-sigmoid, tan-sigmoid (the one used in this study) functions that - along with the weight of the connection - combine to pass an output in the range of $[-1, 1]$ to the next neuron. Those values are altered through training in every training cycle. This structure is analogous to the physical brain in such that our neurons need enough stimuli to respond and pass it along. The learning process takes place when the output that is obtained from the last layer is compared to the expected data during the training period, and the measured error (the difference between the output and real data) is used to adjust the weights connecting the neurons.

With every cycle (also called epoch) the network measures the error through an error or performance function, a function that compares output data to target data and adjusts the weights connecting the neurons accordingly. This process is repeated until the error is minimized to a satisfactory degree, or another halting condition is triggered. Most suitable for time-series prediction are feed-forward neural networks with multiple hidden layers, which may also include a time delay input vector for one-step-ahead prediction[2]. In our case, we have chosen a feed forward neural network model, using both the Levenberg-Marquardt learning algorithm (LM) and scaled conjugated gradient (SCG) as training functions, finally settling on LM for memory and efficiency purposes.

B. Feed Forward Neural Networks

A popular type of neural networks, Multi-Layered Feed Forward Neural Networks (FFNNs) consist of several layers of neurons, where a neuron in a certain layer $x$ is connected to every neuron that belongs to the layer $x+1$, and so on. Cycles cannot form, and FFNNs can be described as directed acyclic graphs (DAGs). This is the model that was used in this research, after trying out several others. The network model used in this research is explained in detail in Chapter 5. FFNNs are typically first thought of when problems of approximation, fitting, forecasting and filtering come across. They are considered the standard type of NNs for a problem such as the one in this work, but Self-Organizing Maps (SOMs) are starting to draw recent attention despite being ruled out in earlier studies.

III. FOREIGN EXCHANGE

A. Market

The average daily turnover of the The foreign exchange (Forex) market reached USD3.2 trillion in April 2007, an increase of 69% over the 2004 average[4]. With such turnover still on the rise day by day - it stands out as the single largest
market in the world. Previously available only to a limited number of wealthy individuals and large banking corporations, nowadays online brokers such as FXCM and OANDA provide almost everyone with a chance to open a micro account for as little as a thousand US Dollars, placed as a deposit against as much as 50:1, 100:1, and up to a high 500:1 margin with some brokers such as North Finance[5]. The risk of such high-margin is not to be overlooked, since sharp drops in the market can result in a margin call, as mentioned in section 1.2. Unlike stocks and options, a trader buys a contract (also called a lot) which is equal to USD100,000 of a currency pair (readers must keep in mind that with a 100:1 margin, one would only require USD1000 to purchase one lot). A pair is made up of two currencies, of which one is usually (but not necessarily) the US Dollar. In common terminology, A trader can either go long or short on a currency pair, going long means they are betting on the value of the pair to rise, while going short means they are betting on the value to drop. For example, a trader going long on the Euro vs. US Dollar at a price of 1.3540 means that they win as long as the value of the pair rises above 1.3540, and lose if that value drops down.

Popular pairs that are traded are the Euro vs. US Dollar (EURUSD), the US Dollar vs. Japanese Yen, the Great British Pound vs. US Dollar (GBPUSD) and the US Dollar vs. the Swiss Franc (USDCHF). Those pairs are called the four majors and account for much of the volume traded. Slightly less popular are the Australian Dollar vs. US Dollar (AUDUSD), New Zealand Dollar vs. US dollar (NZDUSD), and the US Dollar vs. Canadian Dollar (USDCAD). Traders can also long or short cross pairs, which are pairs that do not contain the US Dollar, such as the Euro vs. Japanese Yen (EURJPY). Majors are traded in large volumes daily, rendering their charts smoother and less prone to noise. Other, less popular pairs are traded with less volume and consequently, each has an individual, distinct style. In a neural network, smooth data is dealt with more efficiently than noisy data, as feed-forward neural networks primarily belong to the tools of fitting and approximation.

Traders can enter the market without a predetermined plan of action, and exit when enough earnings/losses have accumulated; and can also set several types of limit orders, such as Target Profit (TP) and Stop Loss (SL) orders, the first exists the market when a the price reaches a point in which the number of earned pips is equal to a certain number - to “bring home” the winnings - while the latter exists the market when a the price reaches a point in which the number of lost pips is equal to a certain number, to prevent an even bigger loss. Pending orders, are orders to enter the market when the price reaches a pre-determined point. A pip is a basis point (for example, 0.01 in pairs that include JPY, and 0.0001 on other major pairs).

The forex market is an example of a non-linear, complex, virtual reality where players have various levels of experience. Factors involved in the rise and fall of a currency pair throughout the day are virtually endless with the countless automated robots, individual personalities, and central banks’ policies involved. Upon facing a problem that contains a large number of unrelated inputs, neural networks stand out amongst other structures due to their great ability to model complex systems[1].

**B. Technical Analysis**

Technical analysis is a branch of finance that uses past, price-related, mostly mathematically derived values (called technical indicators) to assist in the decision making process when trading in the stock or forex market, though it can be applied in almost any type of market. Two of the founding principles behind technical analysis are (a) all information are contained in price movement, and (b) markets tend to move in trends.

Ever since technical analysis started becoming popular in the 70’s, several attempts have been made to challenge the ability of technical analysis in predicting the market, most notably Burton Malkiel’s “A random walk down wall street”, a study that attempts to prove that technical analysis is no better than random systems, citing that historical market data cannot be used to predict the future [6]. Renowned stock market trader Warren Buffet criticized technical analysis harshly as well. However, a recent study was conducted on 75 of the most popular technical indicators and methodologies used by technical traders, testing their predicting abilities of future values. The results were surprisingly good, providing irrefutable evidence that technical analysis is a valid form of prediction. The complete study can be found in [7]. It should also be noted that technical analysis draws some of its power through its continuous wide use around the world, as the more traders use an interpretation of a certain indicator (or a popular combination of indicators) to place their orders, the more likely that interpretation will manifest in the market due to the sheer volume of orders placed based on that interpretation alone.

Technical indicators vary in calculation methodologies and purpose. Amongst the most widely used indicators are Moving Averages, that calculate the average of a fixed number of previous prices, and can be used to predict the beginning and the end of the current trend. Many other popular indicators are based on primitive moving averages, such as Moving Average Convergence Divergence (MACD), and Bollinger bands.

Support, resistance, and trendlines are limitation levels drawn on the chart, either horizontally or diagonally, connecting past peaks or bottoms on that chart to form an expectation of the upcoming highs and lows. Candlestick figures are also a popular way to chart the Open, High, Low, and Close prices of a certain interval of time into one coherent unit, thus creating several shapes and patterns whose impact on the future movement can be well studied. Other types of indicators include oscillators, a class of often limited-output indicators, measuring the overbought/oversold conditions of a certain financial instrument. Popular indicators in this category include Stochastics, Relative Strength Index (RSI) and Commodity Channel Index (CCI). Classic statistical measurements can also be fitted to be used as technical indicatros, such as Standard Deviation (StdDev) and regression channels. There are hundreds of technical indicators in use in today’s different markets and more are being created on a daily basis, some shared in online communities for a chance to be tested and reviewed by peer traders.
Left to say that chart objects such as support, resistance, trendlines and fibonacci retracements are usually drawn as lines on the chart and no mathematical formulae are defined to help accommodate their analysis into a programmable expert advisor, however it would be easy to do so if the scales of time/price chart were clearly defined, linearly or logarithmically, then trigonometry principles were used to define the points and angles of a trendline for example.

Technical analysis contains plenty of other more complex indicators, and combinations of several indicators are used to create trading systems. A more detailed look into the derivation and possible uses of technical indicators can be found in [8].

C. Expert Advisors

Expert Advisors (EAs), also called trading strategies or trading systems— all coin a term that identifies methodologies which uses a set of rules to enter and exit the market, based primarily on the readings of technical indicators. EAs are formed by creating a rule for entry and a rule for exit. An entry rule would trigger a signal for the trader to enter the market, based on a set of conditions that need to be met by a certain combination of technical indicators. The exit rule is usually based on the reverse of the entry conditions, or on a target profit (TP) or stop loss (SL) level being hit, although it can depend on a totally different set of conditions altogether. Expert advisors are used by many traders, as a stand-alone strategy or in conjunction with fundamental (economic data and political events) analysis to create a profit in the forex strategy or in conjunction with fundamental analysis to create a profit in the forex strategy. EAs can depend on a totally different set of conditions altogether.

Expert Advisors (EAs), also called trading strategies or trading systems—all coin a term that identifies methodologies which uses a set of rules to enter and exit the market, based primarily on the readings of technical indicators. EAs are formed by creating a rule for entry and a rule for exit. An entry rule would trigger a signal for the trader to enter the market, based on a set of conditions that need to be met by a certain combination of technical indicators. The exit rule is usually based on the reverse of the entry conditions, or on a target profit (TP) or stop loss (SL) level being hit, although it can depend on a totally different set of conditions altogether.

Expert Advisors can be grouped based on the type of signal generator used (for example, a dual moving average crossover or a price channel) and the filtering mechanism used (typically oscillators).

An example of a popular, Dual moving average cross with a filter oscillator is shown here:

```plaintext
if {  
  SMA(10) crosses SMA(20) from below  
  and  
  RSI(14)<50  
  and  
  MACD(12,26,9) >0  
  if NumberOfPositions ==0  
  buy one lot  
  at the close  
  of the bar  
  if NumberOfPositions ==1  
  close an already shortsold lot.  
}

if {  
  SMA(10) crosses SMA(20) from above  
  and  
  RSI(14)>50  
  and  
  MACD(12,26,9) <0  
  if NumberOfPositions ==0  
  buy one lot  
  at the close  
  of the bar  
}
```

A summary of the most popular classes of EA's can be found in [9].

This approach has several drawbacks as discussed in the introduction, as most of the rules used in EAs are rigid and non-flexible, and their unchangeable nature mean that they do not adjust as the market goes along, unless the system was manually edited for optimal performance. The result is that many systems are created, based on an extensive testing of historical data, such as studying of the validity of indicators in the past and backtesting the system on a years’ worth of price movements—yet due to the lack of flexibility, the system fails to sustain its initial profit average after a seemingly successful start, due to the inevitable change of the market momentum that is to come. This problem is a good candidate for neural network, given the type and complexity of the parameters involved, and the networks’ self-learning nature.

D. Expert Advisors in a Neural Network

The technical indicators that make up the rules of an expert advisor can act as a part of the neural network inputs. With the weights and connections amongst the hidden layers adjusting as the network runs through the time-series data, learning the relevancy of indicators to the target output. A large number of research works used technical analysis as the neural networks’ basis for prediction, and we present some of the most relevant results in the next Chapter. One will have to note here that the effect of combining Neural Networks and Technical Indicators in one system is not necessarily synergistic, but technical indicators provide rich material though which a neural network can learn and subsequently form decisions. In the model presented in this work, technical indicators construct a type of “environment” that represents happenings around the signal at the time of execution, as we will later see.

IV. RELATED WORK

The studies in [10] and the updated version of the same study found in [1] were conducted to predict the value of the AUD against six other currencies, USD, CHF, GBP, NZD, SGD (Singapore Dollar), and JPY. The study was originally conducted to show the superiority of ANN’s over traditional time series forecasting methods such as ARIMA, and has shown that ANN’s have a great potential in predicting the Foreign Exchange rates. The data set used was the historical exchange rates of AUD against the six aforementioned currencies from January 1991 until July 2002, namely 565 weeks closing prices, taken from the Reserve Bank of Australia. The first 500 weekly closing prices were used as the training set, and the remaining 65 weeks were used as the testing set. The inputs used were based on Technical Analysis, namely MA5, MA10, MA20, MA60, MA120 (The moving averages of one week, two weeks, one month, one quarter, half a year) in
addition to $X_r$, which represents last week closing price as the final input.

The network’s model had six input neurons (one for each of the indicators mentioned above) in the first layer, one hidden layer, and one neuron in the output layer that outputs $X_{t+1}$. Three different learning algorithms were used separately: Standard BackPropagation (SBP), Scaled Conjugated Gradient (SCG) and BackPropagation with Baysian Regularization (BR). The performance metrics in the updated work were based on Normalized Mean Squared Error (NMSE), Mean Absolute Error (MAE), Directional Symmetry (DS), Current Uptrend (CU) and Current Downtrend (CD). The formulae for those performance metrics can be found in [11]. 30 different networks were trained each with different values for initial weights, learning parameters, number of neurons in the hidden layer(s). The results showed that BR and SCG had at least one half of the NMSE of SBP, and were on average 10% better in predicting trend direction. What is interesting to mention is that the directional change prediction accuracy, one of the most sought after forecasts, was above 80%, which is 10% better than a similar previous study that can be found in [11]. In [11] The value of the USD against the Deutsch Mark (DEM), GBP, CHF, AUD, and JPY was studied similarly, and 510 weeks of data were used for training. Indicators used were also MA5, MA10, MA20, MA60, and MA120, and several types of networks were used with several hidden layers and initial weights configurations. However, the classic BackPropagation was used.

A comparative study conducted in [12], in several domains not only limited to forecasting upcoming prices, but to also classify stock performance as either positive or negative. Both Technical and Fundamental analysis were included in the analysis, with inputs including qualitative analysis of companies’ CEO speeches, financial ratios and return on equity (ROE), absolute variation, direction of variation and short-term historical prices for stocks, monthly growth rate of money supply for the S&P 500 index, in addition to several other factors. ANN Algorithms used included BackPropagation, Boltzmann’s Machine, ADALINE/MADALINE, and one included a hybrid approach of a backpropagation + expert system combination. The inputs varied from only 3 in number in the S&P 500 study to 88 in a study that used a Boltzmann machine. The output was not always the forecast of the price, but sometimes included possible stock returns, market entry recommendations, or just a simple long/short recommendation analysis. The total number of cases studied in [12] was seven. Results varied in “correctness” and all of the ANN models outperformed traditional statistical methods. One study that used BackPropagation performed as well as 90-100% correct, 3 studies (One using BP, one using Boltzmann’s Machine, one using ADALINE/MADALINE) performed in the 70-80% range, and a study using Perceptrons performed in the 60-70% range. The other two studies had a qualitative output that was not measured in [12] but it was mentioned that the results outperformed those of statistical methods. It can also be noted that ANN models that performed the best had a relatively simple structure and mostly not more than 10 inputs. The Boltzmann’s Machine with 88 inputs and Perceptron with 40 inputs were ranked amongst least successful in the study. One can clearly note the effects of slow learning and over-fitting that occur when large networks with noisy data and not enough/more than enough training, as results will tend to lag behind seemingly less sophisticated networks with less relevant data.

Error Correction Neural Networks (ECNN) Along with BackPropagation were used in a 2004 masters thesis at the Royal Institute of Technology [13] to forecast the prices of the Swedish Stock Index SXGE along with individual stocks Volvo B and Ericsson B. Performance metrics were based on Hit Rate and Realized Potential, comparing both ECNN and another “naive” method. The data inputs in this study was the timeseries itself without any derived technical indicators, and a curve-fitting, error minimizing approach was used. Inputs were the Close, High, Low of the previous day, along with the transaction volume. Along with external index and commodity prices such as the Dow Jones, DAX, S&P 500, Nikkei 225, USDSEK rate, Gold rate, Swedish interest rates, etc..., depending on a significant correlation between several of those indices/rates and the stocks that were tested. There were both daily data and weekly data and each was tested separately, with 5000 and 10000 epoch’s of training performed respectively, after which the networks with the most optimal weights were chosen to run on the testing data. Results of the testing outperformed the naive method, with the SXGE index prediction showing better results than individual stocks. And in a comparison to a similar study performed on the German DAX index in 1997, the network model in [13] had a 56.8% hit rate for the SXGE in comparison to 55% to the previous DAX study.

A study to predict the exchange rates of the Istanbul Stock Exchange (ISE) index was conducted in [14] and the inputs used were the previous ISE close, previous USD/TL (Turkish Lira) close, previous simple interest rate weighted average close, and 5 other dummy variables representing the 5 business days of the week. 417 days worth of data were gathered from the Central Bank of Republic of Turkey and 90% of them were used for training, the remaining 10% for testing. 6 different models of ANNs (varying according to the number of hidden layers, layer structure, error minimization) were used. Mean Relative Percentage Error was used as a performance metric, and the result was that ANNs used in this study performed better than a Moving Average traditional model that did not use Neural Networks.

One of the most interesting studies was conducted in [15] in which NASDAQ and DOW time series data was tested - combined with one unique technical indicator in every test - through the Levenberg-Marquardt algorithm, a gradient based BackPropagation neural network. The Indicators used were Moving Averages, Exponential Moving Averages, MA difference, EMA difference, Rate of Return for Stocks, RSI and MACD. 310 daily Close, High, Low rates were gathered of each of the two indices, and used to create 130 weekly data units of which 110 were in the training set. The network architectures tested had one internal (hidden) layer of neurons varying from 5 to 8 units, Results were best when time series was used with MACD and MA difference (separately),
and worst when used with RSI. Time series + MACD in predicting the DOW had merely a 0.70% difference than the target price. Even with the worst result, Time series + RSI on NASDAQ had a 2.53% difference in percentage. This study shows what potential technical indicators have when used along with neural networks.

In [16] a neural network models were constructed to generate buying and selling signals in the Tokyo Stock Exchange Prices Index (TOPIX). 260 weeks worth of data were used for training, and 119 weeks were used for testing. 11 technical indicators acted as inputs, including Moving Averages, Deviations from moving averages, psychological lines and RSI. Three types of experiments were conducted, One is Equalized Learning (EL) in which data is pre-optimized though several steps before its being fed to the network. Another one is a traditional neural network, and the third model was a statistical methodology based on discriminant analysis. In results, The Neural Network with Equalized Learning performed the best, with a correctness ratio of 45% in selling signals, 66% in nochange signals, and 76% in buying signals making that an average of 63% overall. The neural network model with normal learning performed correctly in 61% overall, (which includes a mere 16% selling signal correctness ratio), while the traditional statistical method had a correctness percentage of 50%. The problem with the normal learning neural network is that it was under-learning in the buy and sell category, and over-learning in the no-change category, which is the non-action taking category. A following comparison between the yearly profits generated by neural networks based models and other RSI, psychological, and buy-and-hold strategies showed a better performance for the ANN based methods.

Other works included constructing a neural networks model to predict the Riga Stock Exchange (RSE) index[17], and a study of using Recurrent Neural Networks which include an extra layer alongside the hidden layer that feeds the hidden layer directly from the output or input layers[18]. Those studies had a moderate level of success.

The concept portrayed in this work, however is not directly related to any of the previous work. It draws some ideas from the tests in which technical indicators and time-series related data were fed as network inputs, however the usage and purpose of the network is an entirely different idea. Early on, technical analysts used indicators such as RSI and MACD to help filter out good and bad decisions. In our approach, we use such indicators as feeding material for a neural network that is going to perform the aforementioned filtering operation.

V. MODEL

A. Trading Rules

We have used a simple set of two trading rules that will generate entry and exit signals with a very high frequency. The system is based on a dual moving average crossover strategy, similar to one that can be found in [9]. The price-band used for filtering in [9] is not used, instead, filtering is the primary job of the Neural Network. The system uses two Exponential Moving Averages (EMAs) of the past x and y hourly closing prices as the entry/exit level, and it is constantly in the market in one direction or the other. The trading algorithm goes as follows:

\[
\text{if } \begin{cases} 
(\text{EMA}(x) \text{ crosses } \text{EMA}(y) \text{ from above}) \\
\text{close_current_order}(); \\
\text{go_long}();
\end{cases}
\]

\[
\text{if } \begin{cases} 
(\text{EMA}(x) \text{ crosses } \text{EMA}(y) \text{ from below}) \\
\text{close_current_order}(); \\
\text{go_short}();
\end{cases}
\]

The values of x and y have an inverse relationship with the number of generated signals over a period of time; the smaller the value we assign to x and y the more signals we obtain, and since a clear shift in direction is less guaranteed with small values moving-average crossings, more of those signals will be generated during whipsaws - market periods when the price moves sideways - and the quality of the positions to be opened will decrease significantly.

In this paper, we have chosen the hourly OHLC (open, high, low, close) prices of three major pairs, GBPUSD, USDJPY, and USDCHF, and three permutations of x and y, (5,10),(7,15), and (10,20), to create various trading strategies which result in all types of results: successful, break-even, and even losing outcomes, in an attempt to experiment with our neural model in various environments. An example of the signals generated by this system can be seen in figure 1, where every time the red line EMA(5), crosses the blue line, EMA(10), a signal is taken at the close of the hour. Notice the high frequency and the usual lack of quality of such signals.

![Figure 1. The signals generated under the system used in this project, using EMA5 (red) and EMA10 (blue), on random data of GBPUSD. Every red/blue crossing point generates a signal at the end of the bar, either long (if red crossed blue going upwards) or short (if red crossed blue going downwards).](image)
of a real plan behind such a naive system leads to results that vary widely in degree, and that provides a wide range of cases for the filtering mechanism to test itself on (i.e. the filter will be tested on winning, failing, and in-between cases, rather good performing cases alone). Plus, the simplicity of the system makes it easier to derive the results using MATLAB code and later integrate in a neural network. The choice of permutations in this paper was made to model systems with high, mid, low frequency of signal generating, to simulate the usual day traders, swing traders, and mid-range traders (Long term traders sometimes open only a few positions per year, and therefore not enough data would be available for simulation considering that good hourly prices data is not generally available in a good format before 2003).

It should be noted here that in systems of such high frequency of trades, the brokerage fees (around 2-3 pips per transaction on majors, 4-5 pips on non majors), and slippage along with other unexpected conditions (that occur especially during volatile hours) will eliminate a large portion of the profit. This deducted portion is not accounted for in this simulation, due to the various changes that occur hourly and during news times, and the fact that in this approach, NNs are treating the price/time series as a mere set of numbers rather than economic data. And given the high possibility of loss streaks taking place in random systems, it is very likely that a single, large-enough sequence of losing trades might lead to a massive or even total loss of equity, given the absence of proper money management techniques. A filtering mechanism is therefore needed to ensure that traders that use such a system would only act upon “fit enough” signals.

B. Network

- The structure of the neural network that we have chosen is a multi-layer feed-forward neural network (FFNN).
- The input layer consisted of 28 technical indicators, that included 13 simple moving averages (SMAs) of different periods: SMA3, SMA5, SMA8, SMA13, SMA21, SMA34, SMA50, SMA75, SMA100, SMA150, SMA200, SMA250, and SMA300; and 8 relative strength indices (RSIs) of different periods: RSI3, RSI5, RSI8, RSI13, RSI21, RSI21, RSI34, RSI55, and RSI89; and 7 standard deviations (StdDevs) of different periods: StdDev8, StdDev13, StdDev21, StdDev34, StdDev55, StdDev89, StdDev144. These values are taken only on bars where signals are generated (i.e. we have a moving average crossover). In addition to the previous set of indicators, a variable type that is valued either at -1 denoting a downward cross, or at 1 denoting an upwards cross is added to the inputs. We therefore have an input vector of 29*S where S is the number of signals in the training set.
- The 2 hidden layers contain 40 and 20 nodes successively.
- The output layer consists of one neuron. Our target vector consists of an S sized column, containing the values of 0 and 1, denoting that the corresponding row in the input vector - which contains the technical indicators’ values at the time of the signal - has resulted in a negative or a positive profit, respectively. We therefore have a network structure of 29-40-20-1.
- The training algorithm used is the Levenberg-Marquardt (LM) algorithm.
- Weights are initiated to a set of random numbers, data division is set to random, and learning rate is set to default. Maximum validations is set to 10 failures.
- All other parameters are set to the default value provided by the constructor (in MATLAB’s Neural Network Toolbox).
- Performance function used is Mean Squared Error (MSE).

The choices of both the technical indicators, the network structure and parameters were taken based on the best results obtained. However, it is still important to say that such parameters are not as critical as they might appear. The inclusion/uninclusion of SMA300 or StdDev144 for example are not going to affect the results widely, and changing the number of nodes/layers and learning algorithm -unless such changes were drastic- will not cause the network to fail either. The structure that was chosen was processor-friendly enough to conduct a large number of tests, try as many parameters and combinations of inputs as time allows, and still had the depth and power to learn well and generate satisfactory results. A diagram of the network can be seen in figure 2.

![Network Diagram](image)

Figure 2. The 29-40-20-1 neural network model chosen for this work.

During testing, several combinations of technical indicators were tried out, and these combinations included SMA’s, EMA’s, MACD, RSI, StdDev, ADX, and Parabolic SAR. The network models varied between Radial Basis Networks and Multi-Layer Feed Forward, and the performance function largely relied on during testing was Mean Absolute Error (MAE) before finally settling on Mean Squared Error, which performed much better. Both large and small network sizes were tested - as large as a 500-200-100-50-1 with the processor-economic Scaled Conjugated Gradient algorithm, to
networks as small as 8-3-1 structure with the Levenberg-Marquardt algorithm. The choices presented above were finally selected due to both their good error performance and CPU-friendly nature. It should be noted that larger networks did not necessarily perform better, on the contrary, most of them took much larger time to converge to an acceptable error rate, and results showed that such networks did not learn the problem properly - as the testing error rate was much closer to a random one than the result obtained in the training phase.

VI. Data Preparation, Training and Simulation

Data was obtained from Dukascopy, a Swiss foreign exchange data source and broker[19]. The data consisted of the hourly OHLC prices of the GBPUSD, USDJPY, USDCHF, from May 28th, 2007 09:00 GMT to July 18th, 2008 00:00 GMT. The data had to be cleared from non-trading hours (between Friday 21:00 GMT and Sunday 20:00 GMT), in which the OHLC prices are all equal with a zero trading volume, causing unneeded horizontal lines in the chart. Afterwards, the data is loaded into MATLAB as a Financial Time-Series (FTS) object, courtesy of the MATLAB financial toolbox. Indicators are then derived from the time series, and input/target vectors are constructed by selecting data rows that correspond to the hours when signals were generated.

An hourly OHLC row is comprised of the following elements: 1) the date and time, which define the hour to which the prices in that row are related. 2) the volume (not used in this work), which -since cannot be measured accurately in forex-refers to the number of data ticks during that time period, in this case an hour, rather than the actual traded volume of a stock, but still gives a good approximation. 3) the price of the pair at the beginning of the hour referred to in the first row. 4) the price of the pair at the end of the hour referred to in the first row. 5) the lowest point the price of the pair reached during the hour. 6) the highest point the price of the pair reached during the hour. Every other data, such as technical indicators, the expert system’s buy and sell signals (EMA crossing points), TP and SL levels, was extracted from these 6 columns (except volume which was not a factor in any of the indicators used). Then we proceeded to construct the data vectors which would form the elements to be used as the NN’s inputs.

A typical data vector (one row in the array) would therefore consist of 1 cell containing the type of the cross (upward, denoted by 1 or downward, denoted by -1), 13 cells containing the values of the selected EMA’s, followed by 8 cells containing the values of the selected RSI’s, then 7 cells containing the values of the selected StdDev’s. The final cell contains either 0 or 1, to denote whether the result of the buy/sell operation at this signal has ended in a profit or a loss. The derivation of this last cell uses a look-ahead mechanism, however, when the data is fed to the network the last column is separated and only the type, EMA’s, RSI’s and StdDev’s are considered as input \( p \) and the result column is made the target \( t \) of learning and later, for comparison after simulation. All the data in \( p \) and \( t \) were, as mentioned above, extracted solely from the raw obtained data.

Following the research methodologies presented in Chapter 4, we had to divide the input/target vectors \((p,t)\) into two parts, the larger one provides learning material, and the smaller one is left so that the ready-network can simulate itself on it after learning through the first part. The input/target vectors are divided into two parts, within a ratio of 70-80% for training \((p_1,t_1)\), and 20-30% for testing \((p_2,t_2)\). A new multi-layer feed-forward neural network is constructed as follows:

\[
\text{net} = \text{newff}(p_1,t_1,[40 20]);
\]

after setting up the network parameters, the network is trained on the training set:

\[
\text{net} = \text{train}(\text{net},p_1,t_1);
\]

the resulting network net is then simulated on the test inputs:

\[
\text{result} = \text{sim}(\text{net},p_2);
\]

After the simulation, the predicted value \( \text{result} \) is a number that ranges normally between 0 and 1, the values which were learned during training. But many times, the learning does not have to be perfect, and the values could be outside these boundaries. Afterwards, signals with value \( \geq th \) - a threshold selected to filter out a certain portion of the signals - were deemed fit, and several threshold values were chosen in every simulation to verify that the network selection constantly outperforms a random selection mechanism that chooses an equal number of the signals left after every value of \( th \). A typical value of \( th \) is 0.5, if the simulation results were distributed nicely between 0 and 1, and therefore any value above 0.5 would lean towards a fit signal rather than unfit or bogus. However, in some cases, the resulted values were distributed in the lower regions, and not many exceeded 0.7. One more observation was that the signals tended to form densely around a certain point, and a fine 0.05 increase/decrease in \( th \) resulted in the inclusion/disclosure of a substantial amount of signals.

After the threshold operation is performed to set the bar on the signal selection, every selected signal is simulated in the timeseries data and the corresponding profit/loss value is added to the total sum (profit), and the ratio of profitable signal to the sum of selected signals is calculated as well. The same procedure is repeated with a 1000 sets generated randomly, with profit and hit rate (the ratio of correct signals to all signals selected) recorded. Results are presented in the following Chapter.

VII. Results

In the case of GBPUSD selected time-series, the EMA(5)/EMA(10) system produced 633 signals, 500 of which were chosen for training, leaving 133 signals for testing. The EMA(7)/EMA(15) system produced 421 signals, 330 of which were chosen for training, leaving 91 signals for testing. The EMA(10)/EMA(20) system produced 293 signals, 230 of which were chosen for training, leaving 63 signals for testing.

In the case of USDJPY selected time-series, the EMA(5)/EMA(10) system produced 604 signals, 500 of which
were chosen for training, leaving 104 signals for testing. The EMA(7)/EMA(15) system produced 412 signals, 320 of which were chosen for training, leaving 92 signals for testing. The EMA(10)/EMA(20) system produced 288 signals, 220 of which were chosen for training, leaving 68 signals for testing.

In the case of USDCHF selected time-series, the EMA(5)/EMA(10) system produced 595 signals, 480 of which were chosen for training, leaving 115 signals for testing. The EMA(7)/EMA(15) system produced 445 signals, 350 of which were chosen for training, leaving 95 signals for testing. The EMA(10)/EMA(20) system produced 317 signals, 250 of which were chosen for training, leaving 67 signals for testing.

The results of the training and simulation are presented in the tables at the end of this section. In these tables, \( th \) represents the threshold selected with every simulation, \( selected \) refers to the number of signals selected when the corresponding \( th \) was applied, \( all \) and \( "all\%" \) respectively represent the profit (sum of pips) of the entire signal pool and the ratio of positive outcome signals to overall signals. To exemplify, we take the data in Table 1. It represents the first test, which was conducted on our previously mentioned GBPUSD data. The remaining samples after training that were for simulation were 133. The sum of pips generated if all 133 signals were taken would be -702 (a loss of 702 pips), and the winning percentage of those signals would be 24.06%. In the first row of the table, we applied a threshold (filter) value of \( th=0.45 \) that left the “fittest” 107 of those 133 signals. the sum of pips generated by those 107 signals was significantly better (-379), and the winning signals percentage raised a little up to 24.30%. The average of 1000 randomly selected sets of 107 signals out of those 133, yielded a slightly improved sum of pips (-562), but the winning percentage of signals remained almost the same (24.05%). We continued to raise the value of \( th \) to 0.5 and 0.55, getting the fittest 96 signals and 70 signals respectively, gaining even better results for our neural model. The same goes for almost all the tables. However, raising the value of \( th \) much higher will sometimes leave us with a very low number of signals and that would not be almost too random to judge on a network’s ability as a filter. It is best to stick to values where the filtering process is not severe.

Overall, in any similar type of test that is applied on the selected data, the NN results are bound to achieve the highest results with different values of \( th \). In Figure 3, which represents a random test done on the USDCHF using an EMA5/EMA10 expert advisor, the constant blue line represents the \( all \) in the above tables, the green line represents the values of \( rand \) as \( th \) changes, and the red line represents the \( NN \) values as \( th \) changes. The first graph represents \( all, rand, NN \) for sum of pips, and the second one represents their values for the \( hitrate\% \). It can be clearly seen that NN’s provide better results in the areas where \( th \) is doing a good amount of filtering.

<table>
<thead>
<tr>
<th>Table I</th>
<th>GBPUSD, EMA(5)/EMA(10), 133 TOTAL SIGNALS.</th>
</tr>
</thead>
<tbody>
<tr>
<td>( th )</td>
<td>( selected )</td>
</tr>
<tr>
<td>0.45</td>
<td>107</td>
</tr>
<tr>
<td>0.50</td>
<td>96</td>
</tr>
<tr>
<td>0.55</td>
<td>70</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table II</th>
<th>GBPUSD, EMA(7)/EMA(15), 91 TOTAL SIGNALS.</th>
</tr>
</thead>
<tbody>
<tr>
<td>( th )</td>
<td>( selected )</td>
</tr>
<tr>
<td>0.40</td>
<td>33</td>
</tr>
<tr>
<td>0.45</td>
<td>25</td>
</tr>
<tr>
<td>0.50</td>
<td>17</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table III</th>
<th>GBPUSD, EMA(10)/EMA(20), 63 TOTAL SIGNALS.</th>
</tr>
</thead>
<tbody>
<tr>
<td>( th )</td>
<td>( selected )</td>
</tr>
<tr>
<td>0.40</td>
<td>28</td>
</tr>
<tr>
<td>0.50</td>
<td>22</td>
</tr>
<tr>
<td>0.60</td>
<td>15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table IV</th>
<th>USDJPY, EMA(5)/EMA(10), 104 TOTAL SIGNALS.</th>
</tr>
</thead>
<tbody>
<tr>
<td>( th )</td>
<td>( selected )</td>
</tr>
<tr>
<td>0.20</td>
<td>81</td>
</tr>
<tr>
<td>0.25</td>
<td>58</td>
</tr>
<tr>
<td>0.30</td>
<td>32</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table V</th>
<th>USDJPY, EMA(7)/EMA(15), 92 TOTAL SIGNALS.</th>
</tr>
</thead>
<tbody>
<tr>
<td>( th )</td>
<td>( selected )</td>
</tr>
<tr>
<td>0.30</td>
<td>59</td>
</tr>
<tr>
<td>0.35</td>
<td>57</td>
</tr>
<tr>
<td>0.40</td>
<td>52</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table VI</th>
<th>USDJPY, EMA(10)/EMA(20), 68 TOTAL SIGNALS.</th>
</tr>
</thead>
<tbody>
<tr>
<td>( th )</td>
<td>( selected )</td>
</tr>
<tr>
<td>0.20</td>
<td>58</td>
</tr>
<tr>
<td>0.25</td>
<td>50</td>
</tr>
<tr>
<td>0.30</td>
<td>35</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table VII</th>
<th>USDCHF, EMA(5)/EMA(10), 115 TOTAL SIGNALS.</th>
</tr>
</thead>
<tbody>
<tr>
<td>( th )</td>
<td>( selected )</td>
</tr>
<tr>
<td>0.45</td>
<td>49</td>
</tr>
<tr>
<td>0.50</td>
<td>22</td>
</tr>
<tr>
<td>0.55</td>
<td>12</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table VIII</th>
<th>USDCHF, EMA(7)/EMA(15), 95 TOTAL SIGNALS.</th>
</tr>
</thead>
<tbody>
<tr>
<td>( th )</td>
<td>( selected )</td>
</tr>
<tr>
<td>0.15</td>
<td>27</td>
</tr>
<tr>
<td>0.20</td>
<td>23</td>
</tr>
<tr>
<td>0.25</td>
<td>16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table IX</th>
<th>USDCHF, EMA(10)/EMA(20), 67 TOTAL SIGNALS.</th>
</tr>
</thead>
<tbody>
<tr>
<td>( th )</td>
<td>( selected )</td>
</tr>
<tr>
<td>0.45</td>
<td>20</td>
</tr>
<tr>
<td>0.50</td>
<td>17</td>
</tr>
<tr>
<td>0.55</td>
<td>12</td>
</tr>
</tbody>
</table>
with SMA5/SMA10 crossing system.

the finding in [7] that technical analysis is much more reliable or targeted output, our results form a practical validation of no fundamental analysis included as a part of the input vector filter, and did that based on technical analysis alone. And with better in separating profitable and bogus signals than a random overall, we could see that that our neural network was much signals than needed, thereby reducing its performance. Still, to the network filtering less losing signals or more winning complete shift in profitability between the two sets would lead in outcome between the training and the testing sets; because a

signals, that our neural network model was able to identify itself was behaving very poorly - as in the case of USDCHF in the total sum.

It can be seen that in the vast majority of cases, the neural network-filtered set of signals was of a much higher quality than the average of randomly filtered sets, and collected a higher number of pips than both the overall signal pool and its randomly filtered set . With a hit rate usually much better than the filtered method and the overall set, in some cases almost 20% better. Despite varying results being obtained when \( th \) was changed, in the average cases (when \( th \) filtered an acceptable number of signals), the neural filtered method well outperformed random selection. Such over-performance can be seen in a losing system much clearer than a winning one, as in the case of USDJPY with EMA(10)/EMA(20), which produced a high hit rate for a trend following strategy (such as 29.41%), attempts for filtering such a set of signals would be weakest, since a large number of the signals is winning anyway, and the winnings are concentrated in a few signals so that missing one or two will lead to a massive difference in the total sum.

The best filtering was achieved when the original system itself was behaving very poorly - as in the case of USDCHF with EMA(7)/EMA(15) - leading to several uniformly losing signals, that our neural network model was able to identify and eliminate. One more factor to consider is the difference in outcome between the training and the testing sets; because a complete shift in profitability between the two sets would lead to the network filtering less losing signals or more winning signals than needed, thereby reducing its performance. Still, overall, we could see that that our neural network was much better in separating profitable and bogus signals than a random filter, and did that based on technical analysis alone. And with no fundamental analysis included as a part of the input vector or targeted output, our results form a practical validation of the finding in [7] that technical analysis is much more reliable in stock and foreign exchange trading than a random picking system, contrary to what is stated in [6], and our initial claim that a neural network is one of the most suitable tools to model the complex relationship between different types of technical indicators.

B. Suggested Improvements

Plenty of improvements can be applied to the model. First and foremost, the expert advisor/network/filtering systems were all chosen for simplicity and ease of integration, so that this research stays within the scope of a masters thesis.

1) Expert Advisor: The simple EMA cross system selected for this research is considered by many an outdated, unprofitable system that is a part of the first legacy of stock trading techniques. Expert Advisors nowadays accommodate several other advanced indicators (although most of them are derived originally from moving average techniques). ADX, Bill Williams’s chaos trading and Ichimoku Kinko Hyu systems (even though invented years ago) are now gaining a lot of popularity. Integrating such systems with a neural network isn’t as intuitive as a simple moving average crossing system, but such efforts can be very rewarding.

A Neural Network may also be used to assist in decision taking with “vague” Expert Advisors. Some Expert Advisors contain rules that are not fixed, but rather subjective, giving the trader more of a hint to buy or sell for quite a varying period of time, rather than a definitive signal at a specific moment. A neural network can take a good role in deciding when and how to act upon such a suggestion.

2) Neural Network: There can be a few improvements in the error measurement mechanism. After all, in systems where TP/SL can vary to a large degree between signals, a simple win/loss terminology is not enough to denote the outcome. And while in our case the winnings and losses are not too deviated from the mean, still sometimes, 3 losses can clear out 10 or more winnings, and vice-versa. And therefore, a good measurement in error should be able to denote the degree of how well the signal performed rather than just a discrete win/loss. Both the target vector and the usual error functions used in this research did not accommodate that. The network treated all positive signals similarly, and all negative signals similarly, with no distinction made between highly positive and just above zero for example. Improvements can be made to ensure a network could learn the varying degrees of profitability as opposed to the binary fashion in which this work was carried.

3) Filtering: The threshold filtering that had been used to define the line that separates fit and unfit signals could be improved in many ways. This filter is a simple horizontal line, and one can rather use a linear or either a quadratic equation to decide which signals stay and which do not, according to the time and index number of the signal. The quality of the signal separation will in itself improve if the error functions were to be implemented according to the previous section. Besides, there should be a mechanism to ensure the outputs remain as bounded between 0 and 1, as much as possible.
ACKNOWLEDGEMENTS

I would like to thank my supervisor at KTH Mr. Wah-Sui Almberg, and my examiner Prof. Magnus Boman for their constant input and support, Tommy Jonsson and Thomas Gustavsson from North Front Trading AB for providing their assistance and suggestions, and sharing their experience with me. and I would also like to thank Dukascopy.com for their excellent data quality and online CSV export system.

REFERENCES