A prediction based model for forex markets combining Genetic Algorithms and Neural Networks

Rui Furtado
rui.furtado@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

October 2018

Abstract
Investing in financial markets is always a complex and difficult task. To raise the small chances of beating the market, investors usually rely on several techniques that attempt to determine the underlying trading signal, and hopefully predict future market entry and exit points. This thesis proposes a trading system optimized for the Foreign Exchange Market, widely known as FOREX. To perform such task, we use a Feedforward Neural Network (FNN), that take as input features a set of technical indicators (TI), calculated using FOREX hourly data. To optimize the TI and FNN parameters, we deployed an Evolutionary Strategy (ES) based on a Genetic Algorithm (GA). The GA also deploys an automatic Feature Selection (FS) mechanism that enables the FNN to use only relevant features for the given problem. The proposed system is tested with real hourly data from 5 different markets, each one exhibiting different behavior during the sampled time. The produced investment strategies are compared with classical trading strategies for the sake of comparison. The achieved results show that this approach is capable to outperform the Buy and Hold strategy (B&H) in the GBP/USD market, achieving an average result of 14.19% of Return of Investment (ROI), against 10.69% of ROI for B&H. The system also outperformed the Sell&Hold (S&H) strategy for the USD/CHF, achieving a result of 4.45% of ROI against 4.09% for S&H. Furthermore, it is also discussed the usage of Batch Normalization as preprocessement technique during the development of each market strategy.

Keywords: FOREX, Genetic Algorithm, Neural Networks, Supervised Learning, Machine Learning

1. Introduction
Computational Finance has been growing over the years. The latest progresses in the Machine Learning (ML) area and the increase of complexity in financial securities, backed up with a huge growth in computer processing power, have boosted up the area of quantitative analysis. The potential and diversity shown by such methods lead to the creation of powerful systems capable of providing promising results in forecasting how markets will react in a relative near future. These systems are often used by large firms. But since they do not reveal how they are performing nor what results they are achieving, this domain remains in a highly competitive and private environment. Although it is undeniable that this is a field with an outstanding potential for improvement, the activity of forecasting the market is and will always be considered a controversial activity.

There is a well established community of critics who claim that the market cannot be predicted, and all efforts to accomplish this goal are useless due to the randomness associated with market variations. The Random Walk Theory (RWT)[1] affirms that stock prices are completely random, making it impossible to outperform the market. This randomness is explained by the Efficient Market Hypothesis (EMH), which states that financial markets are efficient, and that prices already reflect all known information concerning a stock [2]. This implicitly states that trends and patterns observed in past data are not correlated with future outcomes, and the occurrence of new information is apparently random as well.

This is in direct opposition to Technical Analysis (TA), which claims that a stock’s future price can be forecast based on historical information, through observing chart patterns created by Technical Indicators (TI). We directly apply this classic trading methodology to our model, to empirically prove that the EMH is not completely right. The hypothesis is that if people using this type of procedure can consistently beat the market, then ML methods backed up by Evolutionary Computation (EC), such as Genetic Algorithms (GA) used in this work, should also be able to reproduce an identical behavior in an automatic way, avoiding the need of manual labour in defining a trading strategy.

However, there is always a certain degree of randomness associated with every market, and that trait could never be dissociated from them. The market is and will
always be a noisy, non-stationary, non-linear and temporally ordered system.

1.1. Motivation
The forecasting theme in financial markets has become a major interest topic to investors, used as a measure to secure and manage their portfolio’s "health". Therefore, the main motivation of this work, is to create a suitable system for the FOREX market, that is capable of providing the best possible return of investment (ROI).

It is intend to study if Deep Learning (DL) algorithms like Feedforward Neural Network (FNN) are a suitable tool to deal with financial time series, exploring their adaptability to different FOREX markets.

We also want to identify if evolutionary techniques such as GA are a good feature selection and optimization tool, minimizing the used features to a number that is capable to extract the maximum performance out of each produced Feedforward Neural Network (FNN) model. An important point in the optimization process, is also to study if for the tested markets, the accuracy achieved by the model is correlated with the achieved ROI.

1.2. Proposed Solution
This work tries to approach the forecasting theme by proposing a sequential system that as stated before, combines two powerful algorithms: FNN and GA. The market forecast is going to be performed by the FNN by itself. The provided data is going to be given by raw periodical FOREX data combined with Technical Analysis (TA) features, which are going to be applied to this data to generate relevant features. Predictions are going to be given in the form of binary returns, which will lead to the creation of trading strategy, that will be evaluated making use of relevant financial and statistical metrics.

The GA is going to be used to converge the FNN performance in a faster way then more exhaustive and potentially expensive parameter tuning methods like Grid Search and Random Search. In addition, Feature Selection and Neuroevolution are going to be performed by the GA, selecting a relevant number of initial features, and controlling the architecture of the FNN, respectively.

1.3. Foreign Exchange Market
The FOREX market is the buying and selling market for currencies. In this market it is not necessary to invest in a specific company or sector. The only choice needed to be made is the currency pair that is going to be used during the trading period. Currency pairs represents the quotation of one currency against a second one. The value given to the pair, represents how much one unit of the base currency, is worth when converted to the quote currency. For example if a EUR/USD pair as the value of 1.5 means that one Euro(EUR) worth 1.5 US dollars(USD). Figure 1 presents the evolution of the EUR/USD index from 2013 to 2018.

![EUR/USD market index](image)

1.4. Technical Analysis
Technical analysis (TA) is a methodology used to forecast the future price direction of the market by analyzing market data gathered from trading activity, commonly used in the FOREX market [3]. Technical analysts focus on charts of price movement and various analytic tools, purely relying on statistical metrics to evaluate the health and strength of a given security. Therefore, TA is conceptually opposed to the EMH, believing that past periods could be potentially helpful to understand the market current behavior. TA assumes 3 basic principles [4]:

1. The market discounts everything;
2. Price moves in trends;
3. History tends to repeat itself;

TA is applied by the usage of Technical Indicators (TI). TI are simply mathematical calculations, traditionally in price or volume, based on the past variations of the market and defined by a formula [5]. In this work TIs are used as feature generation tool, in order to create more information to feed to the developed model. In this work we used the following TI:

<table>
<thead>
<tr>
<th>Type</th>
<th>Technical Indicator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trend Following</td>
<td>SMA, EMA, HMA, Aroon</td>
</tr>
<tr>
<td>Momentum</td>
<td>MOM, ROC, RSI, MACD, BB,</td>
</tr>
<tr>
<td></td>
<td>CCI, CMO, ADX, ATR</td>
</tr>
<tr>
<td>Other</td>
<td>DPO, Kurtosis, Skewness, STD,</td>
</tr>
<tr>
<td></td>
<td>Variance</td>
</tr>
</tbody>
</table>

In table 1 we divided the used indicators by type. Trend Following indicators represent indicators that focus on identifying trends in the market, while Momentum indicators measure the velocity of directional price movements. The other row simple contains indicators that do not fit in any one of those two types.
2. Related Work

This section is intended to address and present some related solutions to the work that was developed. The section will be divided in 2 subsections. The first one is directly related with works where NN take a major part in the problem solution, in financial contexts. The second subsection reports works performed with GAs.

2.1. Related works on Neural Networks

Yao and Tan [6] proposed a neural network model that received as input a set of different technical indicators along with weekly time series data, to capture the underlying “rules” of the movement in currency exchange rates. They structured the problem as a regression one, forecasting the weekly market closing price for each market. Evaluation was performed using the Normalized Mean Squared Error (NMSE), weekly return and directional change, expressed as gradient. Regarding this metrics, the authors acknowledged that the aim of market forecasting is achieving the highest possible value of trading profits, and it does not matter whether the forecasts are accurate or not in terms of NMSE or gradient. They were capable of achieving as their best result a return of 28.49% for the CHF market.

Kara et al.[7], also confirmed the supremacy of NN models in financial applications, in relation to classical ML approaches in the Istanbul Stock Exchange(ISE) National 100 Index. They deployed a comparison between two classification techniques, NN and Support Vector Machines (SVM). They also used TA as input feature generator, selecting ten TI as inputs of the proposed models. Instead of predicting future market values, they decided to forecast direction of daily change in the stock price index, creating labels that identify stock movements. Labels are categorized as 0 (downtrend) or 1 (uptrend). To evaluate the deployed models they used daily stock prices from 1997 to 2007. They conclude that both the ANN and SVM models showed significant performance in predicting the direction of stock price movement. However, performance of the ANN model (75.74% ACC) was found significantly better than that of the SVM model (71.52% ACC).

Jigar Patel et al.[8] predicted CNX Nifty and S&P Bombay Stock Exchange (BSE) Sensex indices from Indian stock markets with a fusion of different machine learning techniques. A two stage fusion approach supporting ten different technical indicators as input features was proposed, and applied to three different models: a SVM combined with an Artificial Neural Network (SVR-ANN), a twofold SVR model (SVR-SVR) and finally a SVR combined with a Random Forest model (SVR-RF). The first stage of each one of these models applied a SVR model to each input feature in order to predict the next day value of that feature. The second stage consisted in a ANN, SVR or RF model, fed with ten future values of the previously predicted statistical parameters. The results proved that a combination of two techniques could achieve impressive results, totally overcoming a single layer methodology. The best overall prediction performance was achieved by the SVR-ANN model, getting as results 139.39 of MAE, and 3.41 of RMSE for the CNX Nifty, and 449.75 of MAE and 3.34 RMSE for the BSE Sensex.

A common approach to financial market analysis, is also the forecasting of returns. Qio et al. [9] applied this methodology to the Japanese Nikkei 225 index. They collected 71 variables that included financial indicators and macroeconomic data, divided in a monthly fashion, with a covering period from November 1993 to July 2013. A feature selection algorithm called fuzzy surfaces was used to reduce data dimensionality, to a minimal combination of explanatory variables. They discovered that from the initially collected 71 features only 18 statistically significant. Finally, data was fed into a three layer ANN with a regular back propagation mechanism, with a GA for parameter optimization. Performance was accessed using the Mean Squared Error (MSE). Results shown a MSE value of 0.0017 for the best model, and an average MSE of 0.1219, obtained from 900 training experiments.

2.2. Related works on Genetic Algorithms

As it was previously stated, a Genetic Algorithm (GA), is an evolutionary computing technique designed to search an optimal or near optimal solution in a search space were the algorithm is confined, always following a methodology that tries to mimic and approximately replicate the principles of genetic and natural selection.

A good example of how parameter optimization could be performed, was presented by Chih-Hung Wu et. al. [10]. This work aimed to develop a genetic-based SVM (GA-SVM) model that aimed to automatically determine the optimal SVM parameters, $C$ and $\sigma$, with the highest predictive accuracy and generalization ability simultaneously. The model was built to predict bankruptcy, and was tested on the prediction of financial crisis in the Taiwan market, achieving results that empirically confirm that the GA-SVM model achieved the best predictive accuracy when compared with the other tested models, namely classic financial statistic predictive methods and a FNN. To optimize the initial parameters of the SVM, the GA-SVM first generated a random population, where real values of $C$ and $\sigma$ are coded into the chromosome structure of each element of every generation. Finally, the model searches for optimal values applying a survival of the fittest strategy. They achieved 76% of accuracy.

A more similar work with the one proposed in this thesis was done by Sezer et al. [11]. They built a deep MLP neural network for buy-sell-hold predictions, with TI parameters optimized by GA. As input data, they used daily stock prices of Dow 30 stocks between 1/1/1997 to 12/31/2006, for training purposes, and stock prices between 1/1/2007 to 1/1/2017 for testing.
The target vector of buy-sell-hold points was created based on values given a RSI trading strategy in combination with a trading strategy based on SMA, to identify uptrend and downtrend market periods. The GA was used to find best RSI values for buy and sell points in downtrend and uptrend in a random initialized population of 50 individuals. Generated chromosomes are divided in two distinct parts: 4 initial genes for identified uptrend periods and 4 genes for downtrend periods. The most profitable chromosome is retrieved, and training data is generated according to it. The achieved results proved that using this strategy, the created system had the capability to beat the classical Buy & Hold trading strategy achieving 22.4% of ROI.

Yusuf and Asif Perwej [12] also proposed a system that combined GA with a FNN, optimized for the BSE market. They used a GA to search a space of ANN topologies and select those that optimally matched their criteria. The deployed network consisted of one input layer, two hidden layers and one last output layer. The searched topologies included the number of neurons of the input and hidden layers, since the last layer was always confined to one neuron. The defined output was the prediction of tomorrow’s excess return. They compared their results with classical Time Series prediction methods namely Autoregressive models, and concluded that ANN models are superior, due to being able to capture not only linear but also non linear patterns in the underlying data.

Gorgulho, Neves and Horta [13] proposed a work that used a GA kernel to optimize technical analysis rules for stock picking, and portfolio composition. Their work aimed to manage a financial portfolio by using technical analysis indicators as trading rules. The used TIs were EMA, HMA, ROC, RSI, MACD, TSI and OBV. The Dow Jones Industrial Average Index (DJI), was used as the selected market, giving to the system user the possibility of choosing data frequency: daily, weekly or monthly. For each trading rule, a score is assigned according to a specific set of hard-coded rules, different from TI to TI. Regarding this mechanism, 4 scores were assigned. A very low score indicates a strong sell/short signal, and a value of 1 is given to it. An equal score is attributed to a very high score, indicating a strong buy signal. To low and high scores a value of 0.5 is attributed. In this work the GA is used to optimize classified trading rules. After the optimization performed by the algorithm, resulting on a classifier equation where a set of technical indicators are correctly balanced, all the assets within the market are classified with weights. In order to validate the developed solution an evaluation against the market itself and several other investment methodologies, such as Buy and Hold and a purely random strategy was performed. The testing period from 2003 to 2009 allowed the performance evaluation under distinct market conditions, culminating with 2008-2009 financial crash. The results were promising since the number of positions with positive return exceeds 80%, for the GA, confirming the high confidence level of the proposed approach.

3. Methodology
In this section, it is intended to provide a general overview over the solution that was developed during the development phase. The overall approach to market prediction, is decomposed into small sets, were each part has a distinct role in the deployed model.

3.1. User input
The user input module is responsible for the interaction between the user and the system. All the system parameters and configurations are defined in a configuration file provided by the user. We can divide the initially given configurations in different sections:

- **Data**: Path to the initially .csv file which must comprise the market data along with Date, Open, Close, High and Low fields.
- **TI features**: Selection of which TI features will be used along with the initial data, and also the range of past periods that are going to be considered for their calculation.
- **GA**: All the parameters that define the GA, the initial number of individuals, number of generations, values for the evolutionary operators and the desired fitness function.
- **Neural net**: All the parameters that define the FNN, type of activation function, number of epochs, batch size, type of optimizer and usage of Batch normalization or not. The percentage of data that is used for train, validation and test is also specified here.
- **Investment**: Parameters that reflect how the system is going to invest in the Market Simulator module, namely how much initial capital should be used, how are the transaction costs, and number of assets to be acquired upon transaction.

3.2. Feature calculation
This is the module that adds and calculates TA features using the initial given data. As we previously specified, this corresponds to the addition of a set of different TIs that use past information to generate new signals.

Throughout the system workflow, feature data will be accessed several times by different GA individuals that consider different types of features that account with a different number of past periods. With that in mind, we chose to calculate each feature, compressing the entire range initially defined by the user, only at the beginning of the program execution, avoiding the need for repeated calculations.

3.3. Optimization
The optimization layer is the core layer of the created system. In it, the previously provided market data,
goes through several different procedures in order to create an individual that has the best possible performance according to the defined fitness function.

### 3.3.1 Population generation

The GA is going to create a population of individuals, each one of them represented by a chromosome, which is no more that an array structure with each element codifying an integer value being known as gene. Genes were randomly initialized, with values within the range of past periods initially defined. The created genes had 3 different tasks:

- **TI creation:** For each selected TI a TI creation gene was assigned. Gene values codified the number of desired past periods.

- **Feature Selection:** To perform Feature Selection, and select the most relevant set of features, presence genes were created. For each selected TI a presence gene was assigned, that indicated if the respective feature would be used in the model creation or not. This was done by using the statistical median of the selected past periods range. If the codified value was higher than the median, the feature would be used in the model creation, otherwise it would be discarded.

- **Neuroevolution:** To evolve the FNN architecture through time, 2 parameters corresponding to the number of neurons present in the input, and hidden layer of the network, were assigned to each created individual.

![Chromosome Structure](image)

### 3.3.2 Model creation

After initializing a population of individual chromosomes, the system creates a FNN model, for each individual, considering the GA selected parameters. First, the feature matrix $X$ is created by fetching data according to the chromosome genes. A label vector $y$ is also created, containing correctly labelled predictions. Predictions are going to be given by binarized market returns. This was calculated by formula 1:

$$ y_t = \begin{cases} 0 & \text{if } \frac{\text{Close}_t - \text{Close}_{t-1}}{\text{Close}_{t-1}} \leq 0 \\ 1 & \text{if } \frac{\text{Close}_t - \text{Close}_{t-1}}{\text{Close}_{t-1}} > 0 \end{cases} $$

(1)

If the calculated return is positive, a label of 1 is created, otherwise, a 0 is given, meaning that the return at time $t$ is negative. Finally, since we are dealing with market prediction, the $y$ vector is shifted 1 row up, in order to make each row $t$ in matrix $X$, to correspond to the return achieved at $t+1$.

After producing matrix $X$ and vector $y$, the system is going to transform the created data. Data is going to be cleaned since the TI feature calculation process generates a high number of not-a-number values (NaN). Data is also going to be splitted in 3 different sets: train - used to train the model ($X_{\text{train}}, y_{\text{train}}$), validation - used to validate the model ($X_{\text{val}}, y_{\text{val}}$), and test - used to test the model ($X_{\text{test}}, y_{\text{test}}$). Finally each value of feature $x$ is going to be normalized by subtracting the mean and dividing by standard deviation of all feature elements (equation 2).

$$ x_t = \frac{x_t - \mu_i}{\sigma_i} $$

(2)

After gathering and transforming all the needed data, data is given into a 3 layer FNN that forecasts market returns, as a binary vector of predictions $\hat{y}$. The validation accuracy will be the optimized metric during the backpropagation process. In each epoch run, the FNN is going to train and test the network, continuously calculating the train and validation accuracy. The run with the highest value of validation accuracy is going to be saved as the optimized FNN for that individual.

### 3.3.3 Fitness computation

To evaluate each individual we propose 2 different fitness functions, 1 directly related with the system performance, and other related with market profitability (equation 3 and 4):

- **Accuracy:** It gives the percentage of correct predictions by comparing the correctly labelled vector $y_{\text{val}}$ or $y_{\text{test}}$, with the prediction vector $\hat{y}$.

$$ ACC = \frac{1}{N} \sum_{i=1}^{N} \left( 1 - \left| \frac{y_i - \hat{y}_i}{y_i} \right| \right) \times 100 $$

(3)
With $N$ being the total number of samples to be predicted, $y_i$ the individual values of the $y$ vector, and $\hat{y}_i$ the individual values of the $\hat{y}$ vector.

- **Return of investment**: Measures the efficiency of the performed investments, giving achieved gains in terms of percentage.

$$ROI = \frac{\text{Returns} - \text{InitialInvestment}}{\text{InitialInvestment}} \times 100 \quad (4)$$

### 3.3.4 GA operators

Until reaching the final number of generations, the GA performs a evolutionary process, taking advantage of the evolutionary operators:

- **Selection**: Selection selects the fittest solutions ranked with higher fit values, to take part in the breeding process of new chromosomes generation. To perform the selection phase, we use the *Tour- nament Selection* method [14].

- **Mutation**: The mutation operator is responsible for mutating one or more genes according to a certain mutation probability. The mutation process is done by sampling an integer uniformly drawn between the range of values initially defined for genes.

- **Crossover**: The crossover operator is used to combine different individuals by mixing each chromosome genes according to a certain crossover probability. The result is 2 new individuals with switched genes between them, picked according a randomly selected crossover point.

### 3.4. Model prediction

After finding the fittest individual, a final prediction is made using the FNN parameters of the optimized version of the system. The prediction is going to be performed, with $X_{test}$ and evaluated with $y_{test}$ (which had been both held until this procedure), instead of $X_{val}$ and $y_{val}$ as it was performed during the optimization process. Figure 5 represents the architecture of the described process.

### 3.5. Market simulation

The market simulation module was designed to test market trading strategies in a simulated environment. It receives the binary returns prediction vector outputted by the optimized model, and tests it against the market. We defined 2 main market positions, long and short. When the simulator orders a long position, the system invests in the market and purchase a predefined number of stocks. Going short means exactly the opposite, with the system selling first and buying when the position is closed.

Since the received signal is composed by 1s and 0s, which indicate proper investment conditions (positive return at time $t+1$) and inappropriate investment conditions (negative return at time $t+1$), a new signal must be generated to accommodate the two types of investment. In order to simulate the transition from long to short investing, a third label, -1, is then create. Therefore, the system investment procedure condenses 3 different states. 1 represents a long position, -1 represents a short position and 0 stands for a non-investing behaviour. The new signal was created by making the difference between $t+1$ and $t$ for each value present in the prediction vector. Figure 4 presents this mechanism.

### 4. Results

To evaluate the performance and robustness of the previously defined model, we tested 5 different FOREX currency pairs market data - EUR/USD, GBP/USD,
GBP/JPY, USD/JPY and USD/CHF. As selected sample rate, we chose to use hourly data over the period of 12/03/2013 to 12/03/2018, which represents a 5 year period of 31167 trading hours. The chose data split was 80% of total data for train, 20% for test and an additional 20% of train data for validation. Regarding the GA, we decided to initialize it with an initial population of 200 individuals and optimized through 20 generations. In terms of TI usage, all the available features were enabled.

4.1. Case study A - Simple prediction
In this first case study, we intend to showcase the system behaviour without performing any type of optimization. The idea is to get the most simple, pure predictions from the created FNN, using as input vector a GA individual created from a single run. Table 2 presents the achieved ROI for validation and test, and the number of long, short and hold positions assumed during the trading period.

<table>
<thead>
<tr>
<th>Market</th>
<th>Val ROI</th>
<th>Test ROI</th>
<th>Long</th>
<th>Short</th>
<th>Hold</th>
</tr>
</thead>
<tbody>
<tr>
<td>EUR/USD</td>
<td>-13.36%</td>
<td>-9.91%</td>
<td>475</td>
<td>411</td>
<td>5219</td>
</tr>
<tr>
<td>GBP/USD</td>
<td>-8.18%</td>
<td>-4.05%</td>
<td>154</td>
<td>153</td>
<td>5807</td>
</tr>
<tr>
<td>GBP/JPY</td>
<td>-0.46%</td>
<td>-10.37%</td>
<td>444</td>
<td>444</td>
<td>5316</td>
</tr>
<tr>
<td>USD/JPY</td>
<td>-7.47%</td>
<td>-9.70%</td>
<td>516</td>
<td>514</td>
<td>5172</td>
</tr>
<tr>
<td>USD/CHF</td>
<td>-9.33%</td>
<td>-9.79%</td>
<td>538</td>
<td>536</td>
<td>5129</td>
</tr>
</tbody>
</table>

Table 2: Financial results

We also studied if Batch Normalization could potentially contribute to the improvement of the above results. Batch Normalization is known for speeding up the training procedure, making the model to converge faster [15]. We obtained:

<table>
<thead>
<tr>
<th>Market</th>
<th>Val ROI</th>
<th>Test ROI</th>
<th>Long</th>
<th>Short</th>
<th>Hold</th>
</tr>
</thead>
<tbody>
<tr>
<td>EUR/USD</td>
<td>-6.69%</td>
<td>-8.29%</td>
<td>276</td>
<td>275</td>
<td>5652</td>
</tr>
<tr>
<td>GBP/USD</td>
<td>-10.76%</td>
<td>-12.35%</td>
<td>278</td>
<td>277</td>
<td>4619</td>
</tr>
<tr>
<td>GBP/JPY</td>
<td>-3.00%</td>
<td>1.58%</td>
<td>52</td>
<td>51</td>
<td>6102</td>
</tr>
<tr>
<td>USD/JPY</td>
<td>2.82%</td>
<td>-2.09%</td>
<td>223</td>
<td>222</td>
<td>5758</td>
</tr>
<tr>
<td>USD/CHF</td>
<td>-2.70%</td>
<td>-0.45%</td>
<td>538</td>
<td>537</td>
<td>5729</td>
</tr>
</tbody>
</table>

Table 3: Financial results with Batch Normalization

By inspecting the achieved results, we can conclude that the appliance of Batch Normalization contributed to improvement of each market except the GBP/USD.

4.2. Case study B.1 - Accuracy as fitness function
In this test case, we tested how accuracy impacts the overall performance when used as GA fitness function. The optimization process is performed on the validation set, and for the fittest individual, the associated FNN model weights are saved in order to previously load the network and evaluate the test set under the same experimental conditions. For this study we enabled Batch Normalization, due to the positive results achieved in the first case study. We also decided to keep track of Maximum Drawdown, a metric that provides to the user in percentage, the biggest drawdown encountered during the trading period. We obtained:

<table>
<thead>
<tr>
<th>Market</th>
<th>Val ROI</th>
<th>Test ROI</th>
<th>MDD</th>
<th>Long</th>
<th>Short</th>
<th>Hold</th>
</tr>
</thead>
<tbody>
<tr>
<td>EUR/USD</td>
<td>-14.92%</td>
<td>-13.94%</td>
<td>1.91%</td>
<td>52</td>
<td>54</td>
<td>6107</td>
</tr>
<tr>
<td>GBP/USD</td>
<td>13.40%</td>
<td>5.39%</td>
<td>6.08%</td>
<td>104</td>
<td>103</td>
<td>5961</td>
</tr>
<tr>
<td>GBP/JPY</td>
<td>12.29%</td>
<td>-2.34%</td>
<td>7.14%</td>
<td>144</td>
<td>144</td>
<td>5918</td>
</tr>
<tr>
<td>USD/JPY</td>
<td>9.79%</td>
<td>-1.21%</td>
<td>6.65%</td>
<td>103</td>
<td>102</td>
<td>6011</td>
</tr>
<tr>
<td>USD/CHF</td>
<td>10.41%</td>
<td>4.55%</td>
<td>4.17%</td>
<td>56</td>
<td>55</td>
<td>6077</td>
</tr>
</tbody>
</table>

Table 4: Financial results with ACC Fitness

By looking into results in table 5 we can clearly see that both val ROI and test ROI improved their results.

4.3. Case study B.2 - ROI as fitness function
Likely to what has been done for the previous case study, we also decided to test the system using as GA fitness function the ROI function. This way, we make the GA evolutionary process to search for the fittest individual in terms of validation ROI and check if its superior performance also holds true for the test set. We obtained:

<table>
<thead>
<tr>
<th>Market</th>
<th>Val ROI</th>
<th>Test ROI</th>
<th>MDD</th>
<th>Long</th>
<th>Short</th>
<th>Hold</th>
</tr>
</thead>
<tbody>
<tr>
<td>EUR/USD</td>
<td>-9.89%</td>
<td>-13.86%</td>
<td>6.72%</td>
<td>347</td>
<td>346</td>
<td>5513</td>
</tr>
<tr>
<td>GBP/USD</td>
<td>-31.90%</td>
<td>-37.54%</td>
<td>40.12%</td>
<td>1090</td>
<td>1090</td>
<td>4033</td>
</tr>
<tr>
<td>GBP/JPY</td>
<td>-12.91%</td>
<td>-5.36%</td>
<td>6.22%</td>
<td>367</td>
<td>367</td>
<td>5470</td>
</tr>
<tr>
<td>USD/JPY</td>
<td>5.06%</td>
<td>-2.40%</td>
<td>5.10%</td>
<td>113</td>
<td>112</td>
<td>5495</td>
</tr>
<tr>
<td>USD/CHF</td>
<td>-2.19%</td>
<td>-1.82%</td>
<td>6.72%</td>
<td>199</td>
<td>198</td>
<td>5609</td>
</tr>
</tbody>
</table>

Table 5: Financial results with ROI Fitness

The achieved results indicate worse returns in all markets, when compared to the non-optimized version of the system. This may suggest that for the developed system, accuracy is not a well suited metric for conducting valuable market investments. Nonetheless, despite not being included here, the validation and test accuracy suffered a substantial improvement for all markets, reaching values from 55.48% to 66.16% for test and 52.54% to 62.78%, which indicate that in fact the system improved its learning practice but performed bad investments.

By looking into results in table 5 we can clearly see that both val ROI and test ROI improved their results.

Figure 6: Best, average and worst system individuals for GBP/USD

Also most of the MDD values decreased which indicates a decrease in the associated investment risk. We
can also see that the number of investments performed by the system decreased in every market. This indicates that instead of performing a big number of investments, the system prefers to perform a small number of precise investments, which eventually conducted to better returns. The best performance was achieved by the GBP/USD and USD/CHF. Figure 6 presents the average, best and worst runs performed for GBP/USD.

4.4. Case Study 3 - Further investigation on profitable markets

This section was created in order to extend and enhance the previously performed system analysis, on a specific set of currency pairs that achieved promising results in the proposed case studies. We decided to conduct our analysis on the GBP/USD and USD/CHF markets, with the architecture of case study 4.3, since both obtained positive test ROI.

4.4.1 Benchmark comparisons

This section focuses on comparing and evaluate the results obtained by the two proposed investment solutions against traditional trading benchmarks. We used 3 benchmarks:

- **Buy & Hold**: Buy&Hold strategy is a passive investment where the trader opens a long position and holds it for a long period of time until an opinion reversal.
- **Sell & Hold**: The Sell&Hold in a passive investment operation, where trader opens a short position, and closes it only after a change in opinion.
- **Random Walk**: This benchmark is resultant of the Random Walk Theory [1], which states that market fluctuations are randomly generated and completely unpredictable. The application of this strategy is done by generating a binary random signal

We present the results form the best, worst and average of all system runs, since it is important to understand if there is too much variance in the achieved results or not. For the USD/CHF market we obtained:

<table>
<thead>
<tr>
<th>Strategies</th>
<th>ROI</th>
<th>Profit. trans.</th>
<th>Days w/pos. ROI</th>
<th>MDD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg</td>
<td>4.55%</td>
<td>67.84%</td>
<td>83.11%</td>
<td>3.18%</td>
</tr>
<tr>
<td>Best</td>
<td>6.53%</td>
<td>78.03%</td>
<td>86.71%</td>
<td>3.11%</td>
</tr>
<tr>
<td>Worst</td>
<td>3.41%</td>
<td>58.27%</td>
<td>81.41%</td>
<td>3.48%</td>
</tr>
<tr>
<td>Buy&amp;Hold</td>
<td>-10.69%</td>
<td>0%</td>
<td>2.51%</td>
<td>14.58%</td>
</tr>
<tr>
<td>Sell&amp;Hold</td>
<td>-12.98%</td>
<td>21.55%</td>
<td>2.25%</td>
<td>29.29%</td>
</tr>
<tr>
<td>Random</td>
<td>-15.78%</td>
<td>14.52%</td>
<td>2.25%</td>
<td>4.05%</td>
</tr>
<tr>
<td>Table 6: USD/CHF strategies comparison</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

By looking into figure 7 and table 6. Both Buy&Hold, and the Random Walk strategy are clearly outperformed by the proposed solution with the two reaching a negative ROI of -4.09% and -12.98% respectively. The Sell&Hold strategy is the only benchmark that is capable to present a performance suited for the behavior displayed by the USD/CHF, reaching 4.09% a value to the average run of 4.55%.

![Figure 7: USD/CHF strategies evolution over time](image)

Following the same methodology, we also present the same comparison but now for GBP/USD currency pair.

<table>
<thead>
<tr>
<th>Strategies</th>
<th>ROI</th>
<th>Profit. trans.</th>
<th>Days w/pos. ROI</th>
<th>MDD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg</td>
<td>5.65%</td>
<td>45.87%</td>
<td>98.74%</td>
<td>5.97%</td>
</tr>
<tr>
<td>Best</td>
<td>11.51%</td>
<td>47.34%</td>
<td>99.95%</td>
<td>4.24%</td>
</tr>
<tr>
<td>Worst</td>
<td>2.98%</td>
<td>42.45%</td>
<td>97.47%</td>
<td>5.76%</td>
</tr>
<tr>
<td>Buy&amp;Hold</td>
<td>10.69%</td>
<td>100%</td>
<td>99.38%</td>
<td>3.94%</td>
</tr>
<tr>
<td>Sell&amp;Hold</td>
<td>-10.69%</td>
<td>0%</td>
<td>1.59%</td>
<td>14.58%</td>
</tr>
<tr>
<td>Random</td>
<td>-15.78%</td>
<td>14.52%</td>
<td>2.25%</td>
<td>4.05%</td>
</tr>
<tr>
<td>Table 7: GBP/USD strategies comparison</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

By looking into table 7, we can clearly see that only the best individual is capable to outperform the Buy&Hold strategy. This behavior is confirmed in figure 8, where each strategy ROI is plotted over time.

![Figure 8: GBP/USD strategies evolution over time](image)

It is possible to conclude that this strategy is not
capable to beat the benchmarks, since the average ROI value is way lower than the ROI achieved by the Buy&Hold strategy, which is also less risky.

4.4.2 GBP/USD without Batch Normalization

Since the last experiment proved that the deployed system was not capable to develop an enough reliable investment strategy for the GBP/USD market, we decided to extend the experimental process in order to improve the achieved results. Since on the simple prediction test case, for GBP/USD market, the system performed worse with the inclusion of Batch Normalization, we decided to remove it and perform a new evaluation. Table 8 presents the achieved results.

<table>
<thead>
<tr>
<th>Strategies</th>
<th>ROI</th>
<th>Profit. trans.</th>
<th>Days w/pos. ROI</th>
<th>MDD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg</td>
<td>14.19%</td>
<td>45.87%</td>
<td>99.79%</td>
<td>2.94%</td>
</tr>
<tr>
<td>Best</td>
<td>17.81%</td>
<td>48.80%</td>
<td>99.90%</td>
<td>1.35%</td>
</tr>
<tr>
<td>Worst</td>
<td>8.06%</td>
<td>46.53%</td>
<td>99.74%</td>
<td>1.88%</td>
</tr>
<tr>
<td>Buy&amp;Hold</td>
<td>10.69%</td>
<td>100%</td>
<td>99.36%</td>
<td>4.94%</td>
</tr>
<tr>
<td>Sell&amp;Hold</td>
<td>-10.69%</td>
<td>0%</td>
<td>1.61%</td>
<td>14.58%</td>
</tr>
<tr>
<td>Random</td>
<td>-19.36%</td>
<td>12.78%</td>
<td>1.53%</td>
<td>40.25%</td>
</tr>
</tbody>
</table>

Table 8: GBP/USD strategies comparison

By analyzing the above results, it is possible to conclude that the system performance was greatly improved by the removal of the batch normalization layer. Figure 9 depicts the evolution of the new configuration against the 3 benchmarks. For the new proposed solution, the achieved average ROI is 14.19%, a value that significantly beats the Buy&Hold produced ROI, indicating less variance in the produced solution. The best and worst individuals also improved its results in every evaluation metric.

To complement this information, in figure 10, we can see the entry and exit points assumed by the strategy, along the market index. The green points indicate long positions, while the orange points indicate the opening of short positions.

As its possible to observe for this test period, we have a long bullish trend. Hence, the system decided only to make a small number of precise market investments, which eventually conducted to improved results. We believe that with batch normalization, we increase the predictive capacity of the algorithm, making it create an excessive number of incursions to the market, when in the end, for this period, one of the best ways to achieve positive returns, and minimize the investment risk, is to follow the index and encounter some small opportunities to maximize the returns.

5. Conclusions

In this thesis it was presented a financial forecasting system that combines Evolutionary Computing with Deep Learning, in order to provide a trading strategy capable to maximize the obtained returns and minimize the associated investment risk.

To create more data to feed to the developed system, a vast number of Technical Indicators, mathematical formulations that account with past market variations, were used as feature generation tools.

It was also studied if Batch Normalization is a good technique to improve the system convergence and achieve better returns. Results showed that this decision is market dependent, with the displayed market trend playing a crucial role in its inclusion or not.

Regarding the optimization performed by the GA, results showed that between the two available fitness functions, ACC and ROI, the one that provides the best results is ROI. The performed experiments made us conclude that with ROI, the system performs a lower number of investments, but chooses the entry and exit market points in a more precise manner, creating high profit time periods. On the other hand, with ACC the system tries to correctly predict a higher number of market returns, which make it account with all the small price variations, thus losing the overall trend present in the index.
Following this methodology we were able to achieve promising results in some of the tested currency pairs. We were able to come up with two different solutions one for the GBP/USD market, and another one for the USD/CHF. For the GBP/USD we were able to surpass the Buy&Hold strategy by achieving an average ROI of 14.19%, and a 17.81% ROI for the best achieved individual, against 10.69% achieved by the Buy&Hold. Regarding the USD/CHF currency pair, we were able to outperform the Sell&Hold strategy that obtained a ROI of 4.09%, by achieving an average ROI of 4.45% and a ROI of 6.35% for the best seen individual. Such results prove that it is possible possible to extract some profit out of market trading by recurring to previous periods of data, indicating that markets are not completely efficient.

References


