# Forecasting the Forex Market


## Abstract

The recent advancements in the field of computing power and efficient neural networks have given time series prediction a new dimension. One such neural network model is called Long Short Term Memory (LSTM) (Jurgen Schmidhuber, 1997). The properties of an LSTM model to retain context gives this model an edge for predicting time series data as indicated by many recent works; (Dadabada Pradeepkumar, 2016), (Xiaolei Ma, 2015), (Felix A. Gers, 2001). This presents the question whether these models can be applied to the Forex market and if there is a possibility for a network to predict when to buy or sell a currency pair in order to maximize the profit. This report suggests an implementation where a range of technical indicators are used as input for an LSTM. Here it is observed that just modeling the Forex market data through an LSTM with some technical analysis is not enough to make successful buy and sell predictions. This paper also observes how training and testing in a Forex data presents drastically different results.



Maastricht University
Department of Data Science and Knowledge Engineering
Maastricht, January 23, 2019

**Group 10**
Brechard Alarcia, R. (i6197660)
Bhat, S. (i6177161)
Debie, P. (i6114650)
Kerkhofs, C. (i6172266)
Manhar, O. (i6131589)
Salam, J. (i6184256)

KEN4130 Research Project 1
**Supervisor:**
Pietro Bonizzi


## I. Introduction

### A. Foreign Exchange Market

The foreign exchange market (FOREX) is used for trading currencies on a global scale. This market decides the foreign exchange rates, which is essentially the value of the currency. The big banks of the world use this decentralized platform to make currency trades and thus form the backbone of the global economy. The market is also used by big companies such as Apple and Shell that operate and trade resources on a global scale. These corporations are moving massive amounts of funds over this market on a daily basis which causes the volume of the market to be extremely high compared to the regular stock market. The average daily volume of the Forex market is approximately 5 trillion USD (bis.org, 2016 [3]) whereas the combined volume of the New York, Tokyo and London stock exchanges comes closer to 50 billion USD (businessinsider.nl,2017 [19]). This means that liquidity is very high on the Forex market and orders are almost instantly filled without having a big impact on the price. Manipulation from big traders is nearly impossible due to these huge volumes. In addition, the market is also influenced by external factors such as the political climate. These elements together make the Forex market hard to predict.

### B. Motivation

Due to the massive size of the Forex market, it encourages the question whether a pattern can be established in the way the market varies over time. Economists across the world have studied the behavior of the Forex market to establish mathematical patterns to predict trends (refer technical indicators). Thanks to the availability of the historic data of the Forex market, it raises the question whether one could use the methods of data science to predict market trends and hopefully make profit out of that. In this research paper, possible approaches to achieve that have been explored. Unlike regular stock exchanges, the Forex market has no daily opening and closing times. Nor does it have a single physical location. Trades are executed via software platforms and brokers that are open 24 hours a day, for 5 days a week (The market closes on Friday 5pm EST and opens again on Sunday 4pm EST).

### C. Problem Statement and Research Questions

The main research question for this project is: *"Is it possible to implement a profitable Forex trading algorithm using neural networks?"*. The following questions will be discussed in order to find an answer to the main research question:

- *What technical indicators have the most influence in determining a good buy/sell signal?*
- *What neural network structure works best on the Forex prediction task?*

- *Can the neural network return profit on different currency pairs?*
- *What other kinds of inputs can be used (e.g. sentiment)?*
- *Is it possible to transfer the knowledge learned in small (e.g. 1 minute) time frames to trade with different time frames?*

In the conclusion, section VI-A, these questions will be answered using arguments from this report.

## II. Background

### A. State of the Art

There have been multiple approaches taken to model the Forex market to predict up-trends and down-trends. This has been modeled in various Machine Learning techniques. (Dadabada Pradeepkumar, 2016) modelled this problem using neural networks and Particle Swarm Optimization (PSO II-D). Whereas (Sitti Wetenriajeng Sidehabi, 2016) attempted to use statistical and machine learning techniques in combination. The methods they used was Support Vector Machines and Genetic Algorithm-Neural Networks. In this research project, multiple approaches have been taken while keeping the results from the recent researches in consideration.

### B. Technical Analyses

Recognizing trends in the market is essential for making profit in the Foreign Exchange. Determining whether a trend change is about to happen and therefore selling or buying currencies is usually a subjective decision based on the traders skill and experience. To predict trends and their reversals, traders rely on patterns. These patterns can confirm that a trend is still in control or that a trend is changing (Kirkpatrick II & Dahlquist, 2010). Technical analysis is a means of forecasting the price changes using historical market data (Ozturk, Toroslu, & Fidan, 2016). It therefore hinges on the idea that history tends to repeat itself. Technical Analyses can be split up into two sections, a qualitative form and a quantitative form. The qualitative form relies on the analysis of charts of past price behavior and loose inductive reasoning. The quantitative form relies on the analysis of time series data to create trading signals. In practice both forms are often used to make a decision (Menkhoff & Taylor, 2007). However in this paper only the quantitative form is applied as it is believed to contain vasts more and trustworthy information than the qualitative form.

*1) Technical Indicators:* Technical indicators are often used as a method to predict trends, they rely on past price and volume patterns to identify price trends believed to persist into the future (Neely, Rapach, Tu, & Zhou, 2014). Each technical indicator has its own set of rules. With the use of these rules, patterns can be identified and used to determine whether a trend is about to shift. They are usually based on the mathematical properties of a sequence, such as moving averages or the difference between the highest and lowest points in the sequence. In this research paper, once can observe that 32 technical indicators have been used together. The idea behind this approach was to make sure that all the information regarding the trend was available for exploitation.

### C. Long Short Term Memory

LSTM (Long Short-Term Memory) (Jurgen Schmidhuber, 1997) models are extremely useful for Time Series Prediction due to the fact that they can persist information for a long time. This allows the model to have a context to look back on. In terms of predicting the Forex market, this can be useful to match the determined patterns on live data. Research has shown encouraging results that have been achieved on predicting Forex market using recurrent neural networks (Maknickiene & Maknickas, 2012). However, the state of the art models available are constantly evolving with breakthroughs and modifications that researchers make to predict similar time series data.

LSTM networks are powerful tools for predicting time series data, specially in the context of identifying patterns from the past data. There have been multiple works done using this network to predict time series data, such as weather forecasting (Mohamed Akram Zaytar, 2016). This implementation aims to direct the features extracted from the time series data to the LSTM network to predict the outcome and a preferable Buy/Sell window.

### D. Particle Swarm Optimization (PSO)

Many researchers prefer back propagation algorithms to learn the weights for their neural networks, however often these gradient descent methods leave them in local minima and therefore return a sub-optimal solution (Gudise & Venayagamoorthy, 2003). Particle swarm optimization (PSO) avoids local minima as it is not based on gradient descent (Abbass, Sarker, & Newton, 2001). It starts off with initializing number of particles randomly in given space and they all explore towards the best solution. The particles which are scattered in the multi-dimensional space will converge towards the position in this space which returns the greatest value. The dimension of this space is based on the number of weights of the ANN. Finding a correct balance between exploration and exploitation will be essential for finding the optimal solution. An unsuitable balance may result in a weak optimization (Gudise & Venayagamoorthy, 2003).

The alternative option to PSO would be the use of Genetic Algorithms, however according to research conducted by Hassan, et al (2005), Genetic algorithms generate the same high quality results as PSO does, yet PSO requires less computational effort to do so (Hassan, Cohanim, De Weck, & Venter, 2005).

In conclusion, PSO was the best optimization algorithm in this context for learning the weights of the ANN.

## III. Design and Specification

### A. Design

This section briefly explains the overall design of the project implementation. First a baseline algorithm is

implemented as a comparison tool and the model is an LSTM network built to predict optimal time to buy or sell.

*1) Benchmark Algorithm:* In order to compare and test the various approaches a baseline algorithm is required to measure the performance of the algorithms. This baseline algorithm is a rule based trading algorithm that can identify trades based on basic patterns that can be hard coded. The baseline algorithm will be the benchmark to which the other implementations can be compared. The benchmark algorithm only uses basic support and resistance zones to determine when to buy and sell. These are called Fibonacci Retracements. Buying and selling only commence once strong support and resistances are found. A zone is strong if it has been tested multiple times without being broken. In the case that the support or resistance is broken, the algorithm calculates the new support and resistant zones. Often the broken support or resistance become the new zones (BabyPips, 2018). Additionally, stop-loss orders is always applied in case of a bad buys or sells.

*2) Model:* The model is used to predict trend changes within the EUR/USD pair values and perform trades accordingly. To identify trend changes, an LSTM is trained with only the available state of the art technical indicators as input. These technical indicators give indications whether trend changes will occur. Only existing indicators are used for this task. Combinations of these technical indicators should allow for more accurate predictions. The reason for technical indicators as input is that many traders use this in real life to determine when to trade. It is therefor thought to be a method that could likely produce some good results. As this research group has no experience when it comes to the use and identification of good combinations of technical indicators, Particle Swarm Optimization was applied. With the use of Particle Swarm Optimization the weights and therefor the influence of the different technical indicators is adapted. This should allow the LSTM to learn which combinations of technical indicators reveal an accurate prediction of the occurring trend. Based on these predictions trades will be made.

### B. Specifications

To implement the solution, the coding language selected was python along with the Tensorflow framework. This decision was made based on the fact that they are the most used coding language and framework for Artificial Intelligence applications, especially when it comes to easy implementation of neural networks. Being the most popular translates to being the one with a regularly updated documentation and an active community for problem discussions.

A very important library used during the project is called ta, a Technical Analysis library that is used to calculate 32 technical indicators (TI) using the price information. These indicators are the input of the network, therefore getting this information was an important step.

To test the performance of the model a connection was made to a live trading platform. In order to retrieve data from a live environment and to be able to open and close positions the online Forex broker FXCM was used. They offer a free demo account with a fake balance of $50.000. Using their fxcmpy python API it was very easy to implement a connection to the live platform. This library was used mainly during testing.

## IV. Implementation

This sections discusses the implementation decisions that were made and aims to explain the model and the learning process that was applied. For some implementation problems multiple solutions were implemented in order to test and find the optimal implementation to achieve the desired result.

### A. The Dataset

Forex data is easily available free of cost through different sources on the internet. They provide the price values of the desired currency pairs on the basis of whatever candle[1] size is provided.

*1) EUR/USD:* The Euro-Dollar pair is one of the most traded currency pairs in the Forex market. The raw data set covers 17 years of historic price data. One row in this dataset consists of a time stamp and a 1 minute candlestick consisting of an opening, close, high and low price indication of the EUR/USD price rate at that time. In figure 1, candlesticks can be seen. In this case, white candlesticks indicate and increase in price showing the opening, close as the body and the high and the high and low as the protruding line.

*2) Technical Indicators:* As described in Section II, technical indicators form the basis of technical analysis. They are typically plots/patterns on a graph that are used to predict an upcoming uptrend or downtrend. The idea for this project is to collect as many of these indicators as possible and use them all as input to an artificial neural network (ANN). A Python library (*Technical Analysis Library in Python*, n.d.) was used to retrieve a usable total of 32 technical indicators from the raw price data. This step was performed during pre-processing of the data so as to limit any delays or duplicate work during training. What's left is a data set of 5.4 million records covering the price and 32 technical indicators for each minute spanning a 17 year period.

Figure 1 shows an example of one such indicator. The indicator in this figure is called Bollinger Bands and gives an indication of market volatility. The Bollinger Bands are plotted two standard deviations above and below the simple moving average of the market rate. When the bands 'squeeze' closer together, this is an indication of decreased volatility which may signal a future increased volatility.

As the project team has no economic background, it was decided to use all indicators produced by the library and let the network find out for itself (during training) which information to use.

[1] Candles contain the high, low, open and closing prices of a currency for a specific period of time

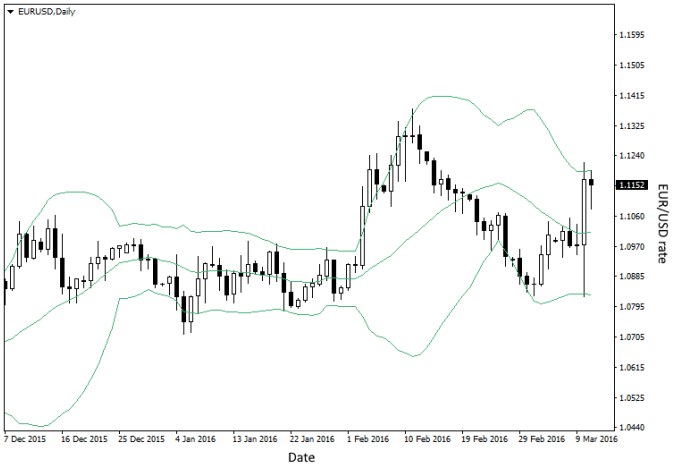



Figure 1: Bollinger Bands are expanded when the market becomes more active (measure of volatility)

### B. Network Structure

The network is divided into two main parts, the feed forward layers and the Long Short Term Memory (LSTM) network. There are two feed forwards layers connected that are charged with pre-processing the data before feeding it to the LSTM which should be able to find patterns in the time series. The input for this network are the technical indicators covering a certain input window. The output of the network is a Nx2 matrix where N is equal to the size of the input window. For each N there are 2 indicators. The first element is considered a bullish indicator (buy) and the second a bearish indicator (sell). This output can then be interpreted in different ways to make a decision on whether to buy, sell or do nothing. The following sections explore the different ways in which the input can be fed to the network and how the output can be interpreted to calculate the profit of the current iteration. Figure 2 shows an overview of the implemented network structure. MACD and RSI are examples of technical indicators used as input.

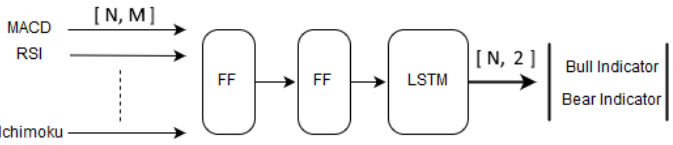



Figure 2: Overview of network structure and input/output

The size of each layer in the network can be adjusted. During testing the best performance was achieved with a smaller network consisting of around 4 nodes in the first feedforward layer, 8 nodes in the second feedforward layer and an LSTM size of 6.

### C. Network Input

As shown in Figure 2, the input for the model is a window with a dimension of NxM where N is the size of the window and M is the number of technical indicators. One such input sequence might for example be the values for each of the 32 technical indicators over a 30 minute consecutive window, resulting in an input matrix of size 30 by 32. During training there is a variety of ways to sample these sequences from the training data and feed them to the model. The following methods were implemented and tested to see which method performs best.

*1) Sequential Input Sequence:* This method tries to make the most of the memory effect of the LSTM. In order to do so all the data is run sequentially for each epoch starting at a random position. A random position is chosen from the training set. All batches of the iteration will then be sampled consecutively from this random starting point in the training set. The idea here is that training batches will also have sequential information which maybe be picked up by the network to make a good prediction.

*2) Random Input Sequence:* After implementing and analyzing the sequential method there was a fear of overfitting the network. Therefore, a new way of inputting the data was designed. This new method, randomizes the windows used as input at every iteration. One single input will still cover a consecutive window, but the iterations will no longer loop over the training data in a sequential way. Instead, a random window is selected for each iteration. This also introduces a different way of interpreting the output that will be explained in the profit calculation section.

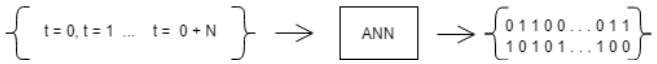



Figure 3: A consecutive window is fed to the ANN to get the output window

Figure 3 shows how a sequential or randomly selected window is fed to the ANN to get the output window.

*3) Overlapping Input Sequences:* In order to more accurately represent the data as it would be used in a simulated or live trading environment, multiple input sequences might be used to feed to the model. The idea here is to take a consecutive window from the training data of size K. From this window, a subset is selected starting at index 0 with a sequence size of N. This input is then fed to the network and the output is stored in another collection. The subset is then shifted to the right by 1 minute by selecting a window starting at index 1 with a size of N. This is repeated until the training window with size K has been looped completely.

The result is a series of overlapping input sequences and their output. The output of each if these executions is then flattened to get an output sequence of size K-N. This output is similar to the way the model would be used in a live environment. The profit function is then called on this output to analyze the performance of the overlapping input sequences.

Figure 4 shows how a single iteration uses multiple consecutive inputs to create a single output window based on multiple inputs. Here the last column of the singular outputs is used, but other methods may be implemented

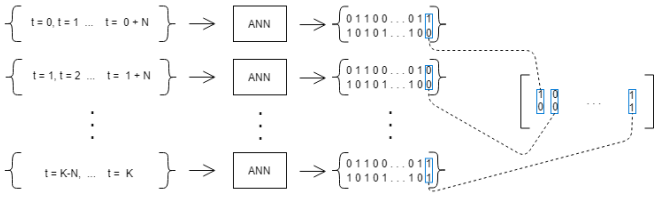



Figure 4: Overlapping inputs are flattened to simulate what would have happened in a live scenario

to flatten the window. The network should learn to ignore the other outputs as a result of training.

### D. Training Batches

During training the selected input method is used to sample training windows from the dataset. In order to reduce noise and allow the profit functions to operate on more data to draw a more meaningful conclusion, one single training step will have multiple batches. For the sequential input method this means that the training step will consist of $B$ sequential input windows while the random input method will have $B$ random windows in a single training step. The overlapping method also uses multiple batches consisting of flattened outputs.

### E. Profit Calculations

Depending on the input method, a profit function is used to analyze the output and assign it a value that indicates the performance of the network as it currently stands. It is important to note that the implementation uses multiple batches in each update. This was done to decrease noise and increase accuracy of the model. This way, the implementation allows for the profit to be calculated for each prediction in the batch and then take the average of this profit to get the final cost for the current set of weights. The section on Particle Swarm Optimization will provide more details as to this implementation.

*1) Sequential based:* Since the data is sequential the trading window has a size of $N \times B$, where $N$ is the size of the window used as input of the network and $B$ the number of batches used. Profit is then calculated considering the first element of the output a bull indicator and the second a bear indicator.

Both short and long positions are allowed in this situation. A position is opened after 5 consecutive timesteps with the same indicator ON and the other OFF. The position is closed after 3 consecutive timesteps with the opposite indicator of the one used to open the position ON, regardless of the value that the opening indicator value.

*2) Random based:* The data found in the batches is not sequential anymore so they have to be treated separately. This means that there are $B$ trading windows. Each of the windows uses the same profit calculation algorithm and the output of the function is then the average profit over all the batches.

Short and long positions are again possible but in a different way. The first element of the output is used for long positions and the second element for short positions. Both kind of positions are opened and closed using the same algorithm. A position is opened when an element goes from a value of 0 to 1 and closed when it goes from 1 to 0. This allows having at the same time short and long positions opened therefore, only half of the money is invested every time a position is opened.

*3) Overlap based:* By using multiple overlapping input windows and flattening the output a series is created that consists of buy and sell signals for each minute. For each buy signal a position is opened at the current market rate. When a sell signal occurs, the position is closed and the gross profit or loss is calculated. This is how the algorithm would be used in the live environment and therefore this approach should produce a more accurate measure of profit for the current model. When a buy signal occurs and a position is opened, all following buy signals are ignored until the position is closed by the first occurring sell signal. Figure 5 illustrates how this process is applied to calculate the gross profit.

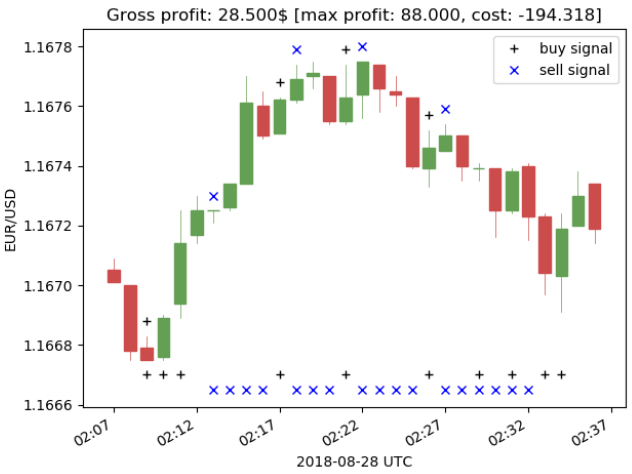



Figure 5: The output sequence of buy and sell signals is used to simulate trading on a 30 minute window

The maximum possible profit is calculated by summing the upwards price movement in the window. E.g. a 12$ profit is very good when the upwards price movement is only 15$. If a 30$ profit is made while the upwards movement was 150$, this is actually a worse result even though the profit is higher. To compensate for this, the profit is taken as a percentage of the best possible outcome. This results in the 12$ profit receiving a lower cost value than the 30$ profit.

To discourage undesired behavior some additional rules are added. When the output contains no buy and no sell signals, or when all signals are always on, the output receives a negative score to encourage some trades to be made. Furthermore, the profit is multiplied by the actual number of positions opened. This last rule is added to force more trades to be made.

*4) Candle based:* In a different approach, a single output from the model is used to get a buy or sell signal from

the input sequence. The random input method is used for this approach. The buy signal can be seen as a bull signal representing an expected increase in the price. The sell signal is seen as a bear signal that indicates an expected decrease in the price. If both signals are off, this is a valid expectation that the price will simply move sideways or only move with a small margin.

The candle based profit function then looks at the next candle following the input sequence. If the candle is green and the buy signal is on, this represents a correct prediction. If the candle is red and the sell signal is on, this also represents a correct prediction. If the price movement in the next candle is smaller than the transaction fee, the buy and sell signal should have been off.

The reasoning for this approach is based on the fact that generally speaking, traders should try to buy low and sell high. If the model can accurately predict that the market is going up, a buy order should be placed. If the model then concludes the market is in a downtrend, all open orders should be closed. This is the most basic trading strategy and should, in theory, return a profit if the model can accurately predict these trends.

### *F. Particle Swarm Optimization*

The particle swarm is initialized as a random uniform distribution over the search space. Here, each particle has a position in the high dimensional space where each value is a weight in the neural network. Each particle also receives a random initial velocity.

When training the ANN, the particles are updated in 2 steps. First, the cost of the current position is calculated by calling one of the profit functions explained in the previous section. Once the profit is calculated, this value is negated to get the cost. This results in a higher cost if the particle does not make profit and a lower cost if the particle is profitable. The cost of each particle is then analyzed to get the personal best position (lowest personal cost) of the particle and also to get the best position of the entire particle swarm (lowest global cost).

The velocity and position of each particle is then updated as such:

$$\begin{aligned} new\_velocity = old\_velocity * \omega + \\ \phi 1 * R1 * (personal\_best - old\_position) + \\ \phi 2 * R2 * (global\_best - old\_position) \end{aligned}$$

$$new\_position = old\_position + new\_velocity$$

where $\omega, \phi 1$ and $\phi 2$ are learning constants. R1 and R2 are random values between 0 and 1. During training, the particles will converge to a global optimal position. The learning constants can be adjusted to promote exploration or social impact.

### *G. Benchmark Algorithm*

The benchmark algorithms starts by identifying the swing high and the swing low for the given window size. The window size is the amount of past data which is looked at to determine the different retracement levels. The swing high and swing low are respectively the highest and lowest value of the EUR/USD pair in that given time window. Given these two values, the algorithm determines whether the market is in an uptrend or a downtrend. The algorithm can identify and uptrend by checking if the position of the swing high happens after the swing low and vice versa for downtrends. The importance of identifying the trend comes when calculating the different retracement levels. The levels are calculated differently for up-trends and downtrends. The levels are based on the following values: 0.236, 0.382, 0.500, 0.618, 0.764. These levels are multiplied by the difference of the swing high and the swing low. Then given an uptrend or a downtrend, these values are then deducted from the swing high given and uptrend or added to the swing low given a downtrend. By doing this, 5 levels are calculated where the prices in theory should bounce of off and indicate buy and sell positions.

To identify a buy or sell position, the algorithm checks the proximity to a retracement level. In the case of buying dollars, if the price stays above and close to a retracement level for a long enough time a buy indication will be given. The idea is that if it the retracement levels holds, the price should increase and therefor create profit. In the case of selling dollars, if the price stays below the retracement level for again a long enough time to indicate the level will hold, a sell indication will be given. Again it is thought that if the level holds the price will bounce off of the level and now decrease. Such a bounce can be seen in figure 6. The price of the dollar can be seen to bounce of the swing low and then consequently bounce off the 0.500 fibonacci retracement level. It can then be said that the level held. Furthermore, if the algorithm sold bought dollars at the swing low and then sold at the 0.500 retracement level, profit will have been made.


Figure 6: Retracement levels; swing high and swing low

If the level does not hold, no buy or sell indication will be given. Instead the algorithm will look to the next retracement level and so try to make more profit. If it is the case that the algorithm gave an indication to buy or sell and the retracement level does not hold, stop losses are placed to reduce losses. Finally, given changes to the swing high and swing low, the retracement levels will be

recalculated.

### H. Simulation

In addition to the training and testing batches, a simulation was implemented in order to simulate how the model would perform in a live environment. This simulation takes a large random batch of testing data and loops over every minute consecutively. Any of the trained models can be loaded by taking their best particle and loading its weights into the network. Each minute the model makes a prediction to buy, sell or do nothing. When the first buy signal occurs, a position is opened at the current rate. The following buy signals are then ignored until a sell signal occurs. When selling, the position is closed at the current rate. The transaction fee and capital are taken into account to calculate the gross profit or loss that was made during this trade. This process repeats until all test data is looped as opposed to the training and testing which was done in much smaller batches. This simulation closely resembles how trading would take place when running the model against a live environment and therefore should give a good indication of the performance.

## V. Experiments and Results

In this section the results from several experiments are reported. A comparison is made between the various approaches and their performance during training, testing and simulation.

### A. Benchmark

The benchmark algorithm was tested on a years worth of data spanning from the begin of October 2017 to the end of September 2018. The results can be seen in figure 7

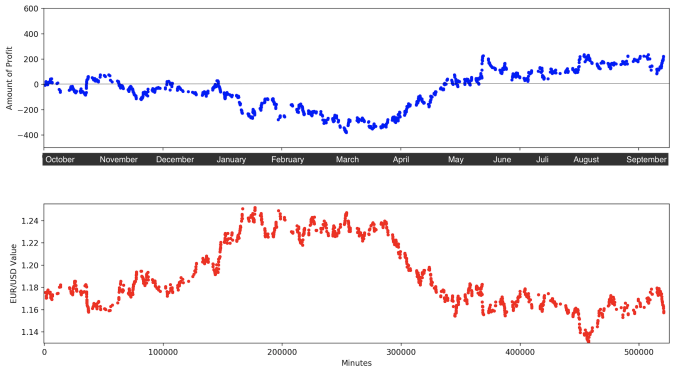



Figure 7: Benchmark results

*1) Analyses:* The benchmark algorithms made a max loss of 388 euros and made a max profit of 234 euros. At the end of the year the algorithm made a profit of 222.50 euros. It can be seen that the benchmark algorithm makes small profits during the first couple of months, however when the the value of the dollar increases the algorithm makes considerable losses. When the value of the dollar decreases in value the algorithm starts to make profit again. This can be seen in figure 7. It is not certain to why the benchmark algorithm makes losses when the value increases and makes profit when the value decreases, however there does seem to be a correlation between the two. Due to the fact that the Fibonacci retracement levels are not usually used alone, improvements could be made to potentially improve performance. Improvements such as combining the buy sell indicators retrieved by the benchmark algorithms with indicators which can be taken from for example the MACD technical indicator. As the MACD indicator also tries to indicate when a potential swing in trend will happen, the combination of the two would increase the probability of successfully trades and therefor more profit.

### B. Sequential and Random

The result of the experiments were very promising, giving big profits over short periods of time. After analyzing the situation it was understood that there was an information leak during the training process and that none of these methods can be used in live trading.

The problem was that there was no prediction in the process, the network was receiving as input the technical indicator and then in the calculation it would use the price of the same window. This makes the problem not a predicting but an analyzing one. That is the reason why the network was performing so well while training and testing with the same function. Nevertheless, When tested with a profit calculation method that would simulate the behaviour of live trading it would open zero positions.

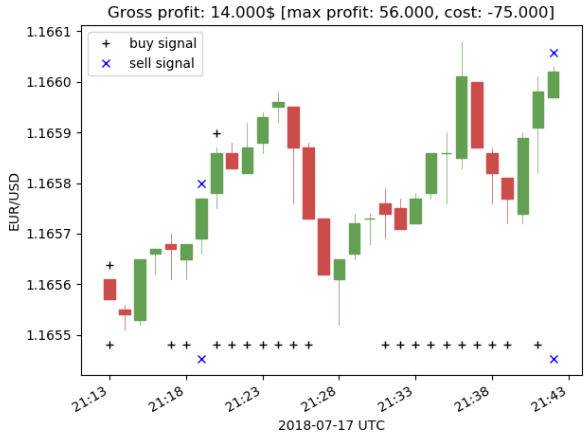
Due to the bad performance of these methods, it was decided to implement the overlapping approach and stop further testing on these methods. This is also main reason why multiple approaches were implemented.

### C. Overlap

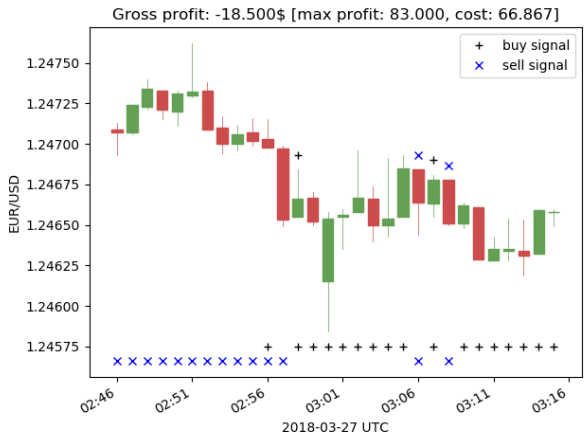
When training using the overlapping approach, the average cost of all particles gradually goes down as is expected. This shows that the implemented particle swarm optimization seems to work as intended. After a while the best particles begin to return a profit as indicated by the profit calculation (E.g. the particle from Figure 5 and Figure 8a). The number of unprofitable particles (E.g. the particle in Figure 8b) gradually decreases.

However, after training for a while the best particles are no longer improving as much. Of course there is a limit to how well a single particle can perform and mostly it seems like this limit was being reached as the particles were returning a profit. Figure 9 shows the best particle and average performance of the swarm during training. As can clearly be seen, the swarm converges on the best particle while the best particle no longer improves after a certain amount of training.

With these promising training results in mind, the trained model was tested on a simulation. While the overlapping profit function was returning good results during training, the simulation showed bad performance. Figure 10 shows the gross profit/loss after simulating real




(a) Profitable particle




(b) Unprofitable particle

Figure 8: Training particles profit calculation

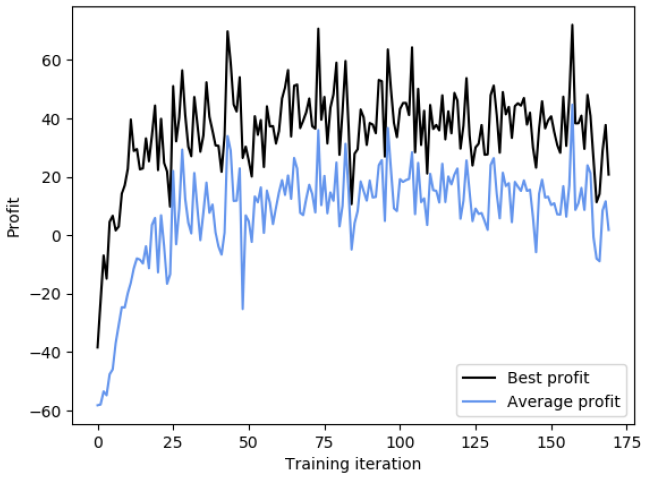



Figure 9: PSO convergence (overlap based approach)

time trading on a 9 month window. The trained model consistently loses money and only seems to be able to make small profitable trades when the market is in an upswing. In this 9 month simulation window the trained model made a total of 14.551 trades, averaging about 3 trades per hour and an average loss of 1.75$ per trade (using a capital of 50.000$ and a transaction fee of 4$ per 100.000$ traded).

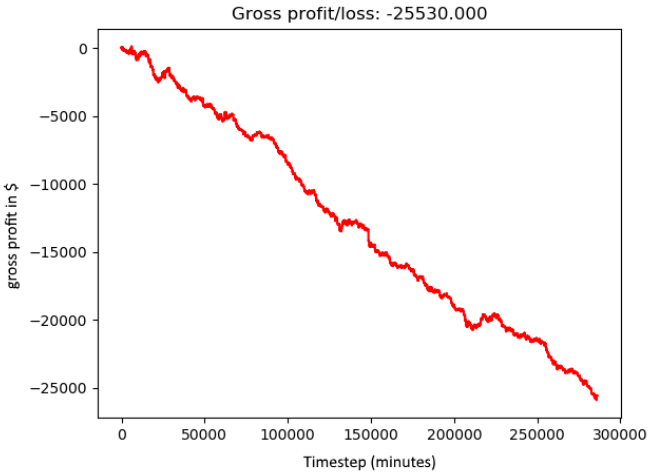



Figure 10: Profit during simulation (overlap based)

Due to the difference in the way the profit is calculated during training and simulation there is no metric to directly compare these two outcomes. Even after testing many different parameter values the outcome is always more or less the same: good results during training but very bad results during simulation. The main reason for this disconnect in results seems to be the approach used to calculate profit and again this comes back to the fact that the cost function used during training does not resemble closely enough the way the output is used during live trading. Instead of expanding upon this model, it was decided to implement one final approach which should at the very least give insight into the disconnect in results by providing a measure of comparison between training, testing and simulation.

### *D. Candle based*

The candle based method showed the most similarities between training, testing and simulation performance. The accuracy of this approach is determined by counting the number of correct and incorrect candle classifications. When a green candle is predicted and the next candle is green, this is a correct classification. When a red candle is predicted and the next candle is green, this is an incorrect classification. The same applies for red/green and green/red predictions. The accuracy can then be easily compared between training, testing and simulation.

First the convergence speed is examined. Figure 11 shows that the implemented PSO solution appears to be working as intended as the average profit of the entire swarm seems to converge on the best particle.

The results of training using the candle based profit function are shown in Figure 12. Note how the best particle already reaches an 55% accuracy in the first iteration. This seems quite high for a randomly initiated particle but not impossible. However, the best particle should increase

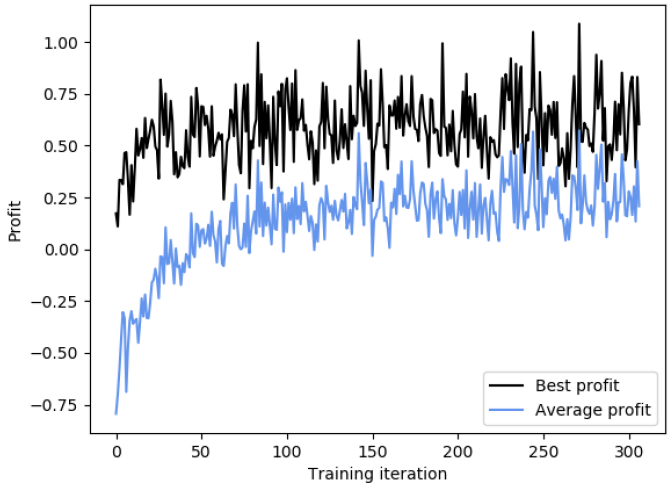



Figure 11: PSO convergence (candle based approach)

its accuracy over time as the particle swarm converges to the global optimum. This does not appear to happen, which may also explain the poor results achieved by the other approaches that were implemented. The best particle reaches an accuracy of roughly 60% on average with a peak accuracy of 73.4% which is explained by a 'lucky' training batch.

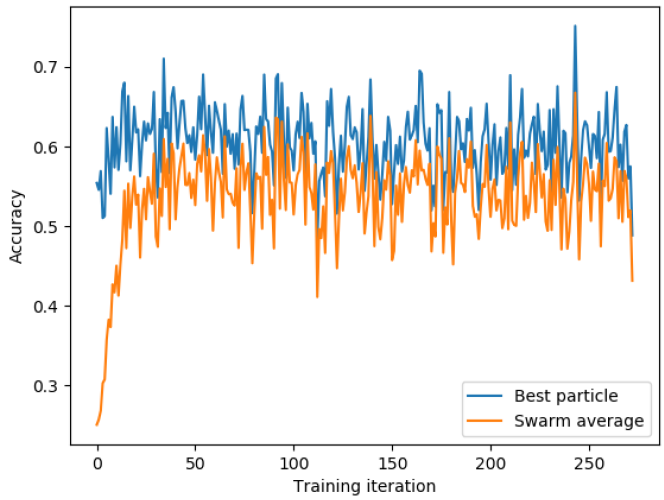



Figure 12: Accuracy during training

Nonetheless, a model with an accuracy of 60% should at the very least make some profitable trades. To confirm these results, the best performing model (average of 60% accuracy on training data) was tested on the test data and achieved a 54.05% accuracy on test data.

The profit of the candle based model when simulating real trading is shown in Figure 13. Here, an accuracy of 46.4% is achieved based on a 9 month simulation of trading with the best trained particle. While there are definitely some profitable trades made by the model, it is still far from profitable over a longer period of trading. The gross loss in this period is less than the gross loss made by the overlapping model but the number of trades is lower at an average of 1.9 trades per hour. This gives an average loss per trade of 2.33$ which is worse than the average loss per trade of the overlap model.

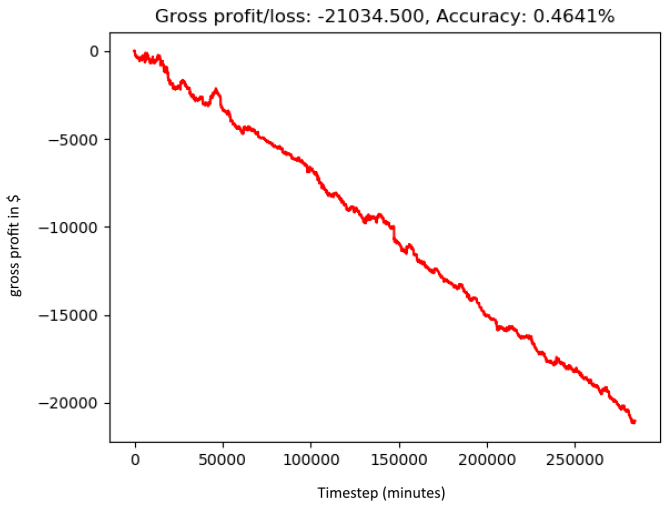



Figure 13: Profit during training (candle based)

## VI. Conclusion

### A. Research Questions Answered

During the planning phase of the project, the following research questions were formulated:

- *What technical indicators have the most influence in determining a good buy/sell signal?*
  It is inconclusive to report on which indicators proved useful, this maybe because the performance didn't seem to fluctuate significantly when certain set of indicators were not used. The model performed the same with different combinations, which didn't give a meaningful conclusion. Overall, the experiments showed that just the technical indicators were not enough to predict Buy/Sell windows.
- *What neural network structure works best on the Forex prediction task?*
  As it is a time series data, recurrent neural networks model such as LSTM was implemented. Research shows their success in time series predictions (Xiaolei Ma, 2015; Felix A. Gers, 2001). However, in this case the LSTM was not able to make profit based on the buy/sell predictions for Forex market.
- *Can the neural network return profit on different currency pairs?*
  The model didn't make profits for intended currency pair (EUR/USD). As expected from the observations, it also gave similar performances for different currency pairs.
- *What other kinds of inputs can be used (e.g. sentiment)?*
  Initially sentiment from the latest news feed about the market was considered as an additional input, however it was not explored in this project.

- *Is it possible to transfer the knowledge learned in small (e.g. 1 minute) time frames to trade with different time frames?*
  The trained model was tested on different window sizes which didn't improve the performance of the network. If it had given optimal results it would give more conclusive results about whether the knowledge transfer is possible at all.

### B. Conclusion from Experiments

From the initial results it became clear that the sequential and random based profit functions are not a good indication of real time trading and thus performed poorly. The overlapping model, which was implemented to address these issues, was able to make profitable trades in the trading simulation. However, the model also made a large amount of unprofitable trades resulting in an average loss per trade of 1.75%.

A final model was implemented in an attempt to simply predict the next candle. If the next candle is predicted to be green, the model should buy. If the next candle is predicted to be red, the model should sell, taking into account the transaction fee of 4$ per 100.000$ traded. This candle based model achieved an accuracy of about 60% on training data, a 54% accuracy on testing data and a 46% accuracy when simulating real-time trading.

When looking at the convergence of the implemented particle swarm optimization method, the results seem to confirm that the implementation is working as intended as is evident from the average cost of the swarm getting closer to the best cost.

In conclusion, the model as a whole does not achieve a high enough accuracy on the task of placing good buy and sell signals. As a result the model performs poorly when trading in real time. As the particle swarm works as intended, the conclusion is that the network structure or the input data is to blame for these poor results by limiting the best possible accuracy to only 60%. Therefore it is concluded that the designed network structure does not represent a good solution to predict the Forex market.

## VII. Future Work

### A. Improvements

The main improvements identified by the results are based on improving the network structure and/or changing the input scheme. The results confirm that the profit function which tries to predict the next candle offers the best trading strategy. To encourage this, a network structure might be implemented which is designed to support this prediction task more adequately. A single LSTM cell might be enough to perform well on this task, meaning the additional layers might not be required for this task.

### B. Researchers and Experts

In order to design a better network to predict optimal buy/sell windows in the Forex market, it should hugely benefit the researchers to employ experts from the economics and business analyst backgrounds. Using their expertise they can help selecting very few technical indicators and how to specifically take advantage of them.

### C. Further Strategies

The sentimental analysis is one of the key strategies to increase the performance of the model.The sentimental data could be used as an add-on to train the neural network.Data could be collected from recognized economic news websites filtered for relevant news articles and relevant trusted twitter feeds. Based on the location the data is found, a weight is given to it. Data collected from news websites therefore have a higher weight. For training a neural network, old articles and twitter messages can easily be found online. The classifiers can be trained to find trigger words and sentences which depict a certain sentiment.

## References


Abbass, H. A., Sarker, R., & Newton, C. (2001). Pde: a pareto-frontier differential evolution approach for multi-objective optimization problems. In *Evolutionary computation, 2001. proceedings of the 2001 congress on* (Vol. 2, pp. 971–978).

BabyPips. (2018). *Babypips: Forex tutorial.* Retrieved from `https://www.babypips.com/learn`

Dadabada Pradeepkumar, V. R. (2016). Forecasting financial time series volatility using particle swarm optimization trained quantile regression neural network.

Felix A. Gers, J. S., Douglas Eck. (2001). Applying lstm to time series predictable through time-window approaches.

Gudise, V. G., & Venayagamoorthy, G. K. (2003). Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks. In *Proceedings of the 2003 ieee swarm intelligence symposium. sis'03 (cat. no. 03ex706)* (pp. 110–117).

Hassan, R., Cohanim, B., De Weck, O., & Venter, G. (2005). A comparison of particle swarm optimization and the genetic algorithm. In *46th aiaa/asme/asce/ahs/asc structures, structural dynamics and materials conference* (p. 1897).

Jurgen Schmidhuber, S. H. (1997). Long short-term memory. *Neural Computation 9(8).*

Kirkpatrick II, C. D., & Dahlquist, J. A. (2010). *Technical analysis: the complete resource for financial market technicians.* FT press.

Maknickiene, N., & Maknickas, A. (2012, 05). Application of neural network for forecasting of exchange rates and forex trading.

Menkhoff, L., & Taylor, M. P. (2007). The obstinate passion of foreign exchange professionals: technical analysis. *Journal of Economic Literature*, *45*(4), 936–972.

Mohamed Akram Zaytar, C. E. (2016). Sequence to sequence weather forecasting with long short-term memory recurrent neural networks.

Neely, C. J., Rapach, D. E., Tu, J., & Zhou, G. (2014). Forecasting the equity risk premium: the role of technical indicators. *Management Science*, *60*(7), 1772–1791.

Ozturk, M., Toroslu, I. H., & Fidan, G. (2016). Heuristic based trading system on forex data using technical indicator rules. *Applied Soft Computing*, *43*, 170–186.

Sitti Wetenriajeng Sidehabi, S. T., Indrabayu. (2016). Statistical and machine learning approach in forex prediction based on empirical data.

*Technical analysis library in python.* (n.d.). Retrieved from `https://github.com/bukosabino/ta`

Xiaolei Ma, Y. H. Y. Y. W., Zhimin Tao. (2015). Long short-term memory neural network for traffic speed prediction using remote microwave sensor data.

# Appendix A
# Gradient Descent Optimization

## A. Method

Gradient descent was used as alternative method for optimizing the LSTM. The same network was used, with the random input sequence method as input, described in section IV.

The main difference between the gradient descent and the PSO optimizer is that the former is setup in a traditional supervised learning scenario. The cost function is defined as the square difference between the predicted future price and the actual future price. While the latter outputs a sequence of events, which can be used to calculate a profit over a time window.

Here we tested the capabilities of the LSTM of predicting the price of the currency 10 minutes in the future, using only the technical indicators of the last 60 minutes. The accuracy is defined as the percentage of the trades which make a profit if a trading order was placed when the model predicts an increase of price above a certain threshold.

## B. Results

Tests of different configurations show that the best performance using a model with 50 and 40 notes in the first and second feed forward layer respectively, and a LSTM state size of 30. After training the network for 200 epochs on the last 2 years of data using the RMSProp optimizer build into TensorFlow, the loss function of the training and testing set converges, but the accuracy defined above still shows that 55% of the trades end up making a loss after transaction costs. Figure 14 shows the training and testing loss.

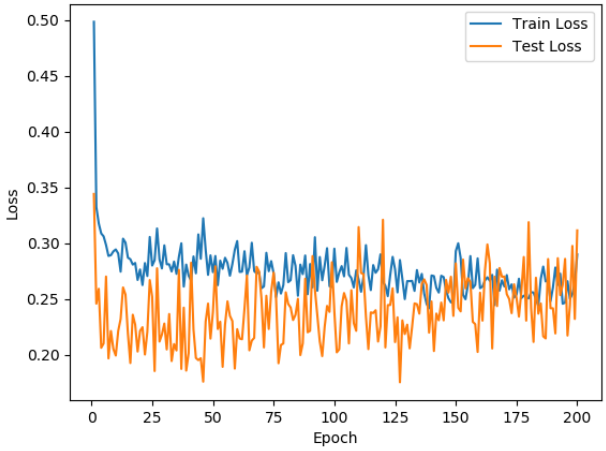



Figure 14: Loss during training using Gradient Recent

## C. Conclusion

First, due to the fact that the testing loss is smaller than the training loss in the first part of the training, we can conclude that the data has a different distribution in the testing set compared to the training set. This can be explained because the testing set is data from a different year.

Furthermore, after increasing the batch size multiple times, there stays a large amount of noise in the loss plot. Which is as expected in any financial market.