FACULTY OF ECONOMICS AND BUSINESS

Applicability of Deep Learning to Construct a Forex Trading Robot

Group report

MASTER OF BUSINESS ENGINEERING

Pieter-Jan Leplae & Robin Meysen
r0482895 & r0588889

Major Data science en business analytics

Promoter: Prof. Dr. Seppe vanden Broucke
Co-promoter: Prof. Dr. Roel Leus
Tutor: Björn Rafn Gunnarsson

Academic year 2018-2019

FACULTY OF ECONOMICS AND BUSINESS

Applicability of Deep Learning to Construct a Forex Trading Robot

Group report

Deep Learning has created a lot of fuzz in various domains of automation, ranging from autonomous driving to even the financial sector. In the latter, curiosity exists whether Deep Learning could be used to fully automate trading. Therefore, this thesis will investigate the applicability of Deep Learning to predict intraday foreign exchange (Forex) rates. This was done by constructing two models based on state-of-the-art techniques for financial predictions: the convolutional neural network (CNN) and the long short-term memory (LSTM) model and comparing their performance with both naïve approaches, such as persistence models, and traditional approaches, such as the Auto Regressive Integrated Moving Average (ARIMA). Throughout this paper, the results of the CNN and LSTM models did not outperform traditional nor naïve models, although they showed some potential for higher forecasting horizons and lower granularities. With neither of the models having learned useful insights into the Forex market, creating a trading robot based on these models might not be useful, as it would most likely not succeed in making consistent profits.

MASTER OF BUSINESS ENGINEERING

Pieter-Jan Leplae & Robin Meysen
r0482895 & r0588889

Major Data Science and Business Analytics

Promoter: Prof. Dr. Seppe vanden Broucke
Co-promoter: Prof. Dr. Roel Leus
Tutor: Björn Rafn

Academic year 2018-2019

Acknowledgements

We would like to express great appreciation to our promotor, Prof. Dr. Seppe vanden Broucke, and our daily supervisor, Björn Rafn, for their great contributions and feedback during the entire process of making this thesis.

Furthermore, we would also like to thank Double Duty NV and therefore also the CEO, Ron Leplae, for providing us with data for our dissertation and some useful insights into some of the market characteristics.

Finally, we would also like to thank our family and friends for the support and encouragement that has been given during the entirety of this process.

Table of Contents

List of figures

List of tables

1 Introduction

Time series forecasting is the practice of applying models on past observations to predict values in the future and is currently one of the most actively used data science methods in business. Popular applications can be found in supply chain management, inventory planning or sales predictions, but also for different applications such as in weather forecasting or for earthquake prediction. Yet another field that has received great attention in the past, is the forecasting of financial time series.

Within the trade industry, predictive models have been used to help make investment decisions with the aim of increasing risk adjusted profits. However, there has been a lot of doubt about making profits off predictive models. According to the efficient market hypothesis (EMH), financial markets reflect all available information perfectly, which makes forecasting in essence useless. In practice, markets are not fully efficient though, making forecasting techniques that are able to capture specific patterns beneficial. Moreover, investment banks such as Goldman Sachs have proven to be profitable on a consistent basis. A figure which shows the daily net revenues for all Goldman Sachs' trading days of 2017 can be found in Appendix 1. A similar distribution is noticeable when inspecting other years. Things like this explain why many judge the EMH to be incorrect.

Although some market-specific characteristics of e.g. the stock market might not support the EMH theory, the spot currency market is still considered to be one of the most efficient markets. With its average daily trading volume of about 5.3 trillion USD (Burns, 2019), which greatly exceeds the volume of the New York Stock Exchange, this market has one of the highest liquidities. Whereas other markets trade stocks of a certain company of the ownership of a certain commodity, this market is focussed on the exchange of one currency into another, e.g. euro to American dollar. Although this market serves as a way of ensuring profits for international companies by buying pull- or call-options for a certain currency pair, 90% of forex trading is of speculative nature (Burns, 2019).

For a long time, researchers and traders have had contradicting views on the characteristics of this market. On the one hand, traders believed that past prices and technical indicators could be used to set up mechanical trading systems. This would then enable them to achieve simple profits with a relative low risk. In academia, on the other hand, many researchers have supported the idea that past movements could not be used to predict future prices. This idea was backed by evidence supporting the random walk hypothesis, meaning that differences in exchange rates have the same distribution as a random walk and are therefore independent of each other. Over the last few decades, however, with the emergence of nonlinear dynamics, strong evidence suggests that exchange rates returns are not independent of each other. Even though there might be little linear dependence, many researches have shown the existence of nonlinear correlations in the currency market (Tenti, 1996).

As mentioned by Borovykh, Bohte and Oosterlee (2018), it is now clear that temporal relationships in foreign exchange rates exist, although they remain difficult to analyse and predict due to the relatively high degree of noise and non-linear trends in the series (Cont, 2001). In the last years, however, machine learning techniques, and more specifically the use of deep learning, have proven to be capable of identifying this kind of nonlinear patterns in financial data (Fischer and Krauss, 2017).

In various fields, Deep Learning has known successful applications. From tools such as Google Translate and the reference-based system of Netflix, and not to mention object detection algorithms and the use of deep learning in digital marketing and robotics. With the huge increase of computational power and storage of data over the last few decades, the number of applications of artificial intelligence has only be increasing. Even in finance, investments in AI by hedge fund companies such as Renaissance, Citadel and BlackRock have surged with around 90% to 940 billion USD in the last decade (Kumar, 2017). Does this mark a bright future for AI in finance?

In this paper, we will focus on applying different deep learning architectures to predict intraday foreign currency exchange movements. More specifically, we will investigate the performance on the prediction of Forex data of two state-of-the-art deep learning algorithms (i.e. CNN and LSTM) and benchmark their performance against a naïve model and a traditional forecasting method (i.e. ARIMA).

By doing this, mainly three contributions will be made to the current literature on deep learning for forex predicting. First of all, novel Deep Learning algorithms will be constructed, and their performance will be compared to more conventional methods for forex trading. Even though there already exists relevant literature, the amount of useful works is rather scarce. Besides the fact that useful literature on forex prediction is limited, we are also motivated by the idea that forecasting foreign exchange rates is seen as an important work. As mentioned by Deng, Yoshima, Mitsubuchi, & Sakurai (2015), it is not only important for investors looking to increase their profits, but also for policy-makers and entrepreneurs. Secondly, to the best of our knowledge, this is the first work to show the application of deep learning models to forecast intraday movements of foreign exchange rates, which might be interesting to investigate as correlations between data points are stronger on an intraday basis. And thirdly, as far as we know, this research is the first to show the effects on model performance of adding different input features such as moving averages and lows and highs of prices, compared to univariate forecasting.

The remainder of this paper is organized as follows. In section 2, the most relevant existing literature will be discussed. Section 3 will then cover the models used throughout this paper and a few theoretical aspects that were implemented along. Section 4 covers our experimental setup and the implementation, including data collection, data preparation, followed by model selection, training, evaluation, finetuning, and prediction. In section 5, the obtained results will be presented, followed by a discussion of these results in section 6. Lastly, section 7 will state our conclusions, limitations and recommendations for further research.

2 Literature review

In 1996, Tenti compared the performance of three recurrent architectures applied on the Deutsche mark and showed that even though the training time is long, recurrent neural networks (RRNs) are very useful for financial forecasting. One problem with RRNs, however, is that long-term dependencies tend to be neglected. Giles, Lawrence & Tsoi (2001) addressed this problem by using a self-organizing map with RNNs, a method which is nowadays known as long short-term model (LSTM). Their model was also applied on forex data to predict the direction of change for the next business day and they managed to obtain an accuracy close to 60% when dropping examples with low confidence.

Dunis and Williams (2002) evaluated neural network regression (NNR) models and benchmarked them against traditional techniques (moving average, logit and ARMA) in forecasting and trading EUR/USD. Even though their NNR model outperformed the traditional models, none of the models achieved a 60% accuracy in predicting a future price movement.

Ashok and Mitra (2002) proposed a hybrid model based on an ANN which uses a Genetic Algorithm (GA) for optimizing the parameters. Their model outperformed a set of conditional heteroskedastic models (such as ARCH, GARCH, and GARCH-M) applied on the major internationally traded daily foreign currencies. In the paper of Pacelli and Bevilacqua (2010) a similar approach was used to forecast EUR/USD exchange rates. Here, however, the model was benchmarked against other (non-optimized) ANNs. Both papers used a model with 2 hidden layers and measured the performance with different metrics such as mean average percentage error (MAPE), mean squared error (MSE) and R-square (RSQ).

For a large-scale comparison between different machine learning models, we refer to Ahmed, Atiya, Gayar and El-Shishiny (2010). In this paper, the authors evaluated eight different techniques applied on the monthly M3 time series competition data (a set of almost 1500 different time series, of which 145 related to finance) and concluded that the MLP was in most cases the best performing one. A great overview of more traditional models can be found in the paper by De Gooijer and Hyndman (2006).

Kipruto, Mung'atu, Orwa & Gathimba (2018) used a NN to forecast exchange rates of the Kenyan currency (KES) against the USD, EUR, GPB and JPY. The authors used one hidden layer with a sigmoid activation function and an output layer with a linear transfer function. Then they evaluated the models based on the MAE, MAPE and RMSE.

A more complex set-up was studied by Bao, Yue & Rao (2017). They presented a deep learning architecture which consists of 3 stages and applied this on different stock indexes. First, the stock data is pre-processed using a wavelet transformation in order to get rid of the noise; Then, a stack of autoencoders (fully connected layers) is applied on the denoised data; and finally, the output is generated by a delayed LSTM. The authors decided to benchmark their model against a traditional RNN, an LSTM, and a WLSTM (combination of wavelet transform and LSTM) and measured their performance by three indicators: R, MAPE and Theil U. Based on the performance measures, the complex set-up, as discussed above, seems to be quite promising on the prediction of daily closing prices of Forex data.

Borovykh, Bohte & Oosterlee (2018) created a similar wavelet architecture based on a convolution neural network (CNN). Their model is applied on different time series, including S&P500 and several exchange rates, and outperformed a vector autoregressive model (VAR) while competing with an LSTM. However, the training speed of their model is much faster than the one of the LSTM. In their paper, the recommendation was made to apply their model on intraday data, as the correlations between data points are stronger on an intraday basis.

Another set-up to predict forex trends via CNN's was proposed by Tsai, Chen & Wang (2018). In this paper, the raw quantitative data was first pre-processed into images. These images, containing price and moving averages (5, 10 & 20), were used to train several deep CNN architectures. The authors concluded, however, that none of their models produced the expected performance.

In table 1, the literature discussed is summarized into an overview. Although most of the research papers mentioned above were not able to back up the applicability of Deep Learning Techniques on financial time series data, the more complex and recent models as discussed by Bao et al. (2017) and Borovykh et al. (2018), did show some promising results. However, the applicability of these models on intraday Forex data still needs to be investigated. Overall two set-ups seem to be the most promising for predicting financial time series: the LSTM, as discussed by Bao et al. (2017) and the CNN, as discussed by Borovykh et al. (2018).

Table 1: Overview of literature

SOURCE	TITLE	MODEL USED	DOMAIN
TENTI (1996)	FORECASTING FOREIGN EXCHANGE RATES USING RECURRENT NEURAL NETWORKS	RNN	FOREX
GILES, LAWRENCE & TSOI (2001)	NOISY TIME SERIES PREDICTION USING RECURRENT NEURAL NETWORKS AND GRAMMATICAL INFERENCE	RNN & SOM (LSTM)	FOREX
ASHOK & MITRA (2002)	FORECASTING DAILY FOREIGN EXCHANGE RATES USING GENETICALLY OPTIMIZED NEURAL NETWORKS	HYBRID: NN & GA	FOREX
DUNIS & WILLIAMS (2002)	MODELLING AND TRADING THE EUR/USD EXCHANGE RATE: DO NEURAL NETWORK MODELS PERFORM BETTER?	NN	FOREX: EUR/USD
AHMED, ATIYA, EL GAYAR & EL-SHISHINY (2010)	AN EMPIRICAL COMPARISON OF MACHINE LEARNING MODELS FOR TIME SERIES FORECASTING	DIFFERENT ML MODELS	TIME SERIES (M3 COMPETITION)
PACELLI & BEVILACQUA (2010)	AN ARTIFICIAL NEURAL NETWORK MODEL TO FORECAST EXCHANGE RATES	HYBRID: NN & GA	FOREX: EUR/USD
BAO, YUE & RAO (2017)	A DEEP LEARNING FRAMEWORK FOR FINANCIAL TIME SERIES USING STACKED AUTOENCODERS AND LONGSHORT TERM MEMORY	WSAES - LSTM	STOCKS
BOROVYKH, BOHTE & OOSTERLEE (2018)	CONDITIONAL TIME SERIES FORECASTING WITH CONVOLUTIONAL NEURAL NETWORKS	CNN	FOREX, S&P500, …
KIPRUTO, MUNG'ATU, ORWA & GATHIMBA (2018)	APPLICATION OF ARTIFICIAL NEURAL NETWORK (ANN) IN MODELING FOREIGN CURRENCY EXCHANGE RATES	NN	FOREX: KENYAN (KES)

TSAI, CHEN & WANG (2018)	PREDICT FOREX TREND VIA CONVOLUTIONAL NEURAL NETWORKS	CNN	FOREX

3 Methods for forex prediction

In this section, we will briefly explain the theory behind the methods used throughout this paper.

3.1 Traditional methods

3.1.1 Persistence model (Naïve approach)

The persistence model (Brownlee, 2016) is also known as the naïve forecasting strategy. This model only uses the most recent value of the time series to predict the value at the next step. We will distinguish between two different ways of implementing this model: price-based and difference-based.

3.1.1.1 Price-based

The first persistence model uses the series as-is to make predictions in the future, hence the name price-based. As one would expect a persistence model to behave, the predictions are made in a rathe naïve way: $\hat{P}_{t+1} = P_t$.

The forecasted value, $\hat{P}_{t+1}$, is the predicting closing price for the next tick bar. When using a price-based persistence model, the forecasted value is equal to P_t , the actual closing price at tick bar t. Note that this model assumes a constant price over time and thus always predicts that the time series will remain unchanged. Therefore, a trading strategy based on a price-based persistence model will never open a position.

3.1.1.2 Difference-based

Whereas a price-based persistence model works with the time series as-is, the difference-based model works with the time series in difference. Each timepoint then reflects the difference in price. Again, the predictions are done in a naïve way: $\hat{D}_{t+1} = D_t$.

The forecasted value, $\hat{D}_{t+1}$, presents the predicted difference for the next tick bar. This is then computed by just taking the value of D_t, the actual differentiated closing price at tick bar t. Unlike the price-based persistence model, the price does not remain constant over time, but the difference does, this means that this model just uses the trend of the last two datapoints and extrapolates it for the forecasted horizon. Therefore, a trading strategy based on the difference-based persistence model, would rely on the presence of a trend-following characteristic of the Forex market.

3.1.2 ARIMA model

A widely used traditional forecasting model is the Auto Regressive Integrated Moving Average, or in short ARIMA. The ARIMA model has three components, the AR, which uses previous values as inputs, and MA component, which uses previous errors as inputs, and an extra component that defines the number of differentiations, being the I component. These first two former components have been tuned based on the partial autocorrelation function (PACF) and the autocorrelation function (ACF). These two plots show an indication of the number of significant parameters for respectively the moving average (MA) and auto regressive (AR) components. For more information on the ARIMA model, Nau (2014) gives a quick run-through.

3.2 Deep Learning methods

3.2.1 Artificial Neural Networks (Multi-Layer Perceptron)

Deep learning is a popular subject. However, before diving into the theory of the models implemented in this thesis, a foundation needs to be built. The simplest deep learning model is called a Multi-Layer Perceptron (MLP). This architecture consists of at least three layers: an input layer, one or multiple hidden layers and finally, an output layer. Inputs are passed along the different layers, each with its own weights and non-linear activation function until the end of the neural network is reached. In the simplest case, all layers are fully connected, meaning that every output of the previous layer is used for the next one. The way this model is trained goes as follows, every batch of data, the gradient of the loss function is calculated using backpropagation. Using this gradient, the weights are updated in an iterative way for each batch until the total number of epochs is reached. However, some techniques do not require the total number of epochs to be completed, but one of these will be discussed further on. Figure 1 shows an example of an MLP architecture. For more information on MLP's, Hernane Spatti et al. (2017) give a good and very thorough explanation.

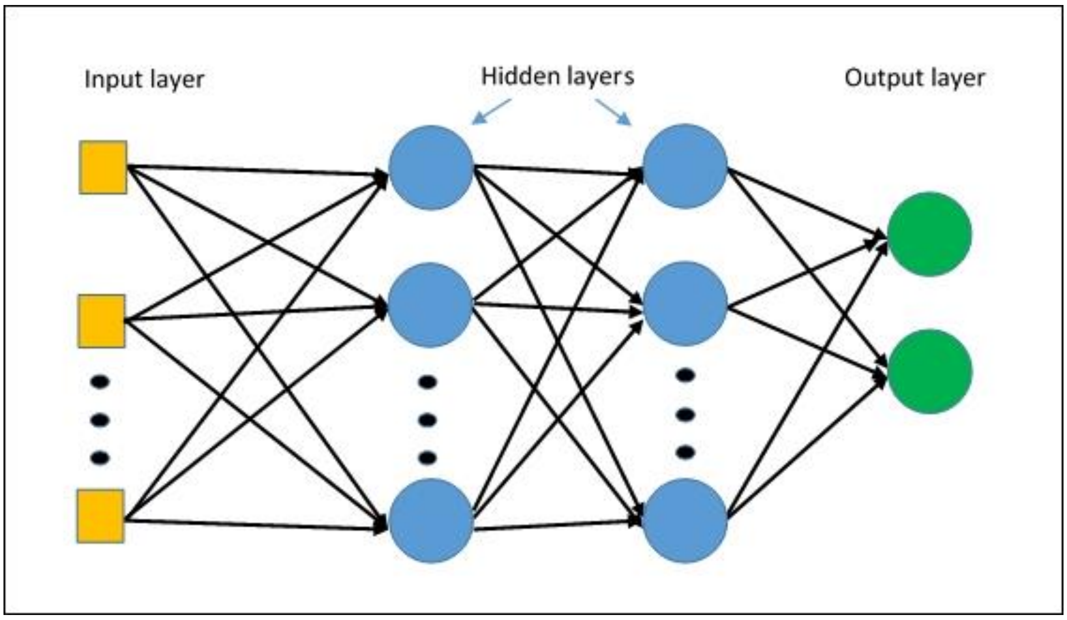


Figure 1: Example of a Multi-Layer Perceptron

https://www.oreilly.com/library/view/getting-started-with/9781786468574/ch04s04.html

3.2.2 Convolutional Neural Networks

A slightly more complex architecture, is the Convolutional Neural Network (CNN). The main idea behind this architecture lies within the use of its so-called filters or kernels. These then perform convolutions, actions that are known to detect certain structures in data. Therefore, CNNs are mainly used for image recognition. The reason for the applicability of CNNs for financial data is its power to detect certain patterns in the preceding datapoints, and thus giving an indication of what the direction of the movement in price will be. For more in-depth information on the working of a CNN, Borovykh et al. (2018) gives a brief but thorough explanation.

3.2.3 Recurrent Neural Networks & LSTM

A widely used branch of Deep Learning for time series analysis is the domain of Recurrent Neural Networks (RNN). The main difference with standard deep neural networks is given by their feedback loops used within the neural network. However, one disadvantage of RNNs, is the problem of vanishing gradients, which is well explained by Pascanu et al. (2012). One branch of the RNN models offers a specific solution to this problem by introducing a new model architecture: Long Short-Term Memory (LSTM). This is done by the introduction of the concept 'forget-gate' by Gers et al. (2000).

3.3 Machine Learning Concepts

3.3.1 Grid Search

A difficult, and often arbitrary part of creating a Machine Learning model is the choice of hyperparameters. However, one way to solve this issue, is to run through the search space of hyperparameters to see which kind of models fit the data best. One way to implement this, is by using a so-called Grid Search. Here, different values are passed along for the different hyperparameters that are present in the grid. These are then enumerated in order to find the model that best learns how to fit the data. More information on hyperparameter searches is given by Claesen & De Moor (2015).

3.3.2 Cross-validation

One way to validate the fitted model is by using the so-called cross-validation. As where a standard training procedure uses one training and one validation set are used, cross validation does this several times, based on the cross-validation parameter, which can be arbitrarily chosen. One extension to this is the so-called stratified cross-validation. The only difference here is that the mean value for the output are set equal, in order to not have too difference between the datasets. When performing a Grid Search, as discussed in the above, a stratified cross-validation is used as well.

3.3.3 Early stopping

One of the most prominent questions when fitting a model on data is whether or not the model is over- or underfit. One technique that helps solving the over- and underfitting issues is early stopping. The concept is simple; Whenever the loss function stops declining for the validation set after a chosen number of training epochs, the training is immediately stopped, hence the name. This prevents the model from overfitting on the training data without improving (or even worsening) the performance of the validation data. One downside of this method is the arbitrarily chosen number of epochs after which the training might be terminated. For more information on this topic, Prechelt (2012) can be consulted.

3.3.4 Dropout

Another way to deal with overfitting, is by using dropout. As discussed by Srivastava et al. (2014), this regularization technique randomly drops nodes at a certain dropout rate when training the model. For example, with a dropout rate of 0.5, half of the trained nodes, along with their connections and output value, are dropped during training. The intuition behind dropout goes as follows, when certain nodes can be dropped, the network cannot rely on certain nodes during training and thus will spread out the weights over different nodes. However, when using dropout, one thing needs to be considered. As dropped nodes during training are present in the final model, weights are changed by a factor of one minus the dropout rate, ensuring that the outputted test values resemble the trained output values. Figure 2 gives a visual explanation of how dropout works.

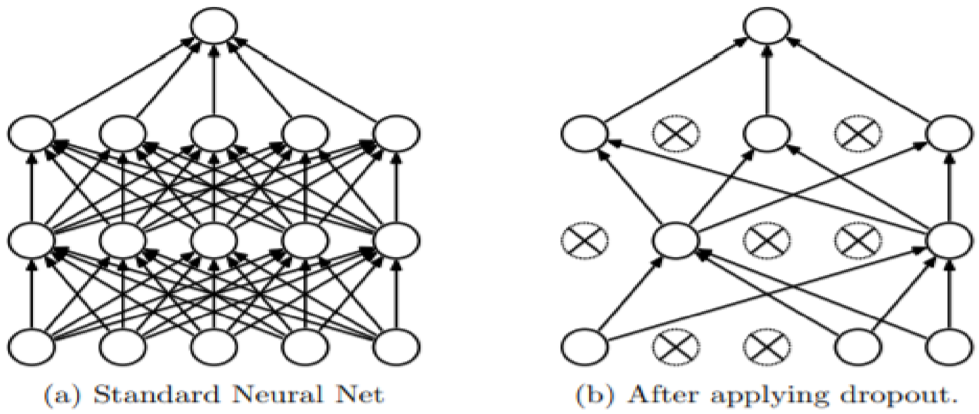


Figure 2: Visual representation of a NN before (a) and after (b) applying dropout (Srivastava et al., 2014)

3.3.5 Batch Normalization

Another way to increase the performance of a deep learning model, is by applying batch normalization. When using this technique, the output values of the activation function for the different nodes in the network are then normalized, i.e. scaled between 0 and 1. After this normalization, two parameters are added to the change the mean and standard deviation of these outputs. By doing this, two advantages are created. First, the training of the model can be done a lot quicker and secondly, some large weights are avoided as this batch normalization is implemented in the process of calculating the gradients when updating the weights of the model. Therefore, batch normalization can be seen as a replacement of dropout for avoiding an overfit of the training data.

4 Methodology

Creating a Deep Learning model can be a long and tedious process. Therefore, the framework for machine learning projects presented by Yufeng (2017) can offer some guidance. The presented framework consists of 7 steps:

1. Data Collection
2. Data Preparation
3. Choosing a Model
4. Training the Model
5. Evaluating the Model
6. Finetuning the Hyperparameters
7. Making Predictions

However, throughout this project, steps 2 through 6 will be looped over when experimenting with certain features and model architectures. The entire process will be executed using Python in Google Colab.

4.1 Data Collection

The first step in any machine learning project is gathering the necessary data. In this case the forex data of the EUR/USD market from 2016 to 2018 was gathered. When taking a closer look at the data, a tick-by-tick granularity can easily be observed. This is the case for 3 features: the bid, the ask and the point in time of the concerned tick. This data was made available for this research by Double Duty NV.

4.2 Data Preparation

4.2.1 Aggregation

The first step in preparing the above-mentioned data, is by aggregating it. Due to its high granularity and the time it would take to open and close a position, a model on highly granular data would be a waste of computational power as our goal is not to scrape profits by performing lightning fast trades. However, note that this is exactly what high-frequency trading (HFT) does (Gomber & Haferkorn, 2013).

The aggregation of the data has been done based on the number of ticks (tick bars) instead of the column indicating the time. By aggregating this way, the main problem of time bars, being that low-activity periods are oversampled and vice versa, is avoided. More information on this topic is given by Lopez de Prado (2018).

The number of ticks per bar can be arbitrarily chosen, but traditionally one relies on the Fibonacci numbers (Vonko, 2019). That is why tick bars have been constructed in batches of 610 and 2584. Depending on the volatility, these granularities represent bars for every minute, 610 ticks in a highly volatile period, to even an hour, 2584 ticks in a period with a rather low volatility. For every batch of tick bars, the open, high, low and close features are constructed based on the averages of the bid and the ask for every datapoint in the batch. Next to these, the latest point in time is passed along as well.

4.2.2 Data exploration

Before using data to train predictive models, it is important to understand the data. In figure 3, a plot of the closing prices is given. Three years of data are shown. Note that the data of 2017 has a different trend, compared to the year before and after.

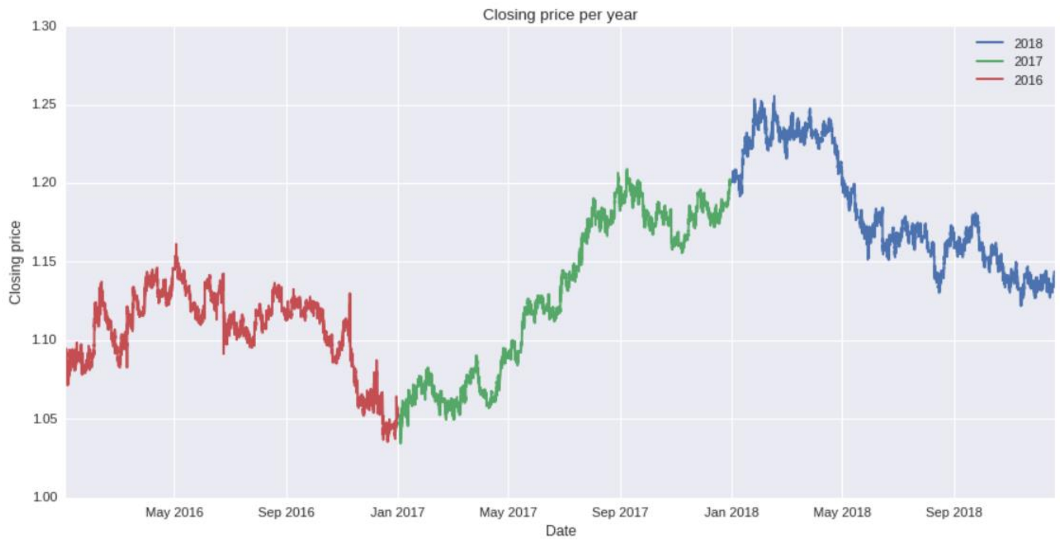


Figure 3: Closing price per year

Furthermore, table 2 reports some summary statistics for each year of our data. These figures are constructed out of the aggregated tick bars of size 610. Firstly, the number of values for the three years of data varies a lot. Due to the use of tick bars, this means that a year with a higher count, such as 2016, has a higher number of transactions, whereas 2018 clearly shows a lower number of transactions. Also, interesting to see, is that the kurtosis varies significantly from year to year. Here, a lower value for the kurtosis of the year 2016 indicates a less peaked version of the probability density function, compared to the ones of 2017 and 2018.

Table 2: Summary statistics of closing prices per year

YEAR	COUNT	MEAN	STD	MIN	MAX	SKEWNESS	KURTOSIS
2016	202022	1.105855	0.026543	1.035300	1.113395	1.16126	-0.833590
2017	124940	1.106802	0.047312	1.034335	1.089155	1.20893	0.585335
2018	66003	1.184923	0.037221	1.121840	1.171865	1.25546	0.284131

4.2.3 Differentiating

When performing time series analysis on financial data, the prices are rarely used as inputs for the concerned model. This is due to the non-stationary characteristics of the price over time. Instead, the differenced time series, i.e. the time series of the changes in price, is used. Figure 4 shows a plot of the same data, but now in differences. From this figure can be seen that the volatility is higher in 2016 than in e.g. 2018. In addition, we can see that the search space is much different compared to the one we had with figure 3. The differenced time series helps us to try and find different patters than the trend in figure 3.

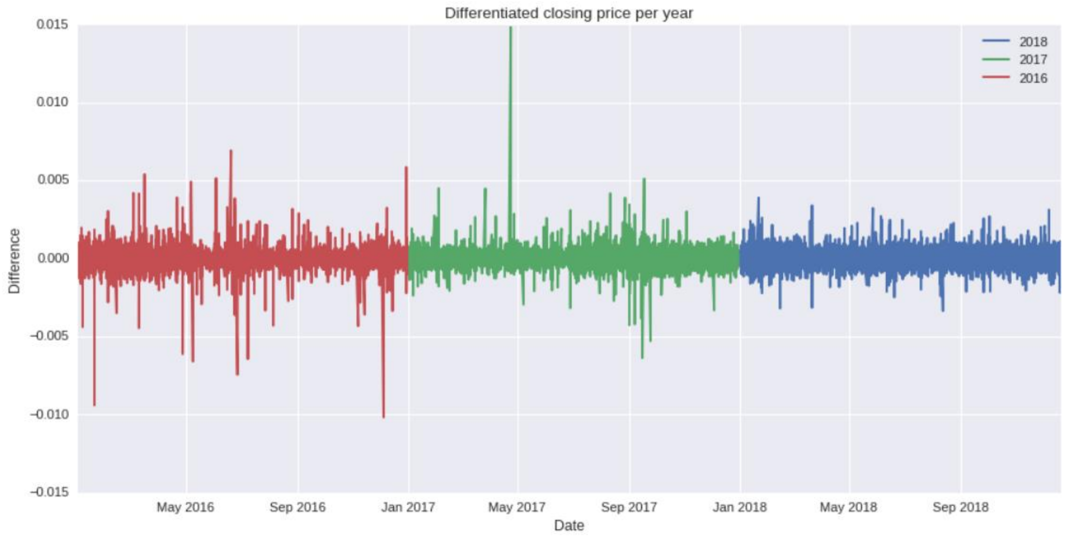


Figure 4: Differentiated closing price per year

Table 3 shows some basic statistics of the time series in differences. One statistic that stands out compared to the statistics of the undifferentiated time series, is the kurtosis. The much higher value is due to the inherent characteristic that differenced time series are much spikier because of the elimination of the trend.

Table 3: Summary statistics of differentiated closing prices per year

YEAR	COUNT	MEAN	STD	MIN	MAX	SKEWNESS	KURTOSIS
2016	202021	-1.682003E-07	0.000235	-0.010185	0.006920	-0.705251	69.629981
2017	124939	1.183578E-06	0.000240	-0.006380	0.014815	1.954202	135.982658
2018	66002	-9.470168E-07	0.000343	-0.003370	0.003890	0.101151	2.777154

4.2.4 Train/test split

The training set will consist out data from years 2016 and 2017, while the test set will be the data from 2018. This gives a test set of 66000 instances, which is about 17% of our available data.

4.3 Choosing a Model and Architecture

As concluded in the literature review, two types of deep learning architectures are currently perceived as state-of-the-art for financial time series prediction: CNNs and LSTMs. That is why those 2 architectures will be considered in this research.

4.3.1 Convolutional Neural Networks

The structure of our CNN will consist of different one-dimensional convolutional layers, followed by a flatten layer, that flattens the data to a one-dimensional array. Next, two fully connected layers are added that give a number of outputs, based on the forecasting horizon, which can predict a movement in price.

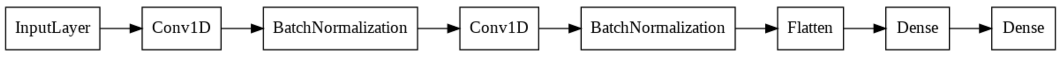


Figure 5: Representation of CNN model

4.3.2 Long Short-Term Memory

The architecture of our LSTM model consists of several LSTM blocks, followed by two fully connected layers, resulting a number of outputs, again reflecting the movements in price for our forecasting horzion. As an LSTM block can ‘forget’, the addition of extra blocks is not harmful for the quality of the predictions. Note that this was not the case for the architecture of the CNNs discussed above.

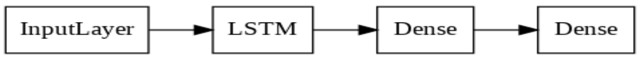


Figure 6: Representation of LSTM model

4.4 Training the Model

Both the LSTM and CNN models are trained on the differentiated time series data of 2016 and 2017. Twenty percent of the training data is left out as a validation set. This data will act as a way of checking whether the initial 80% of the data is overfit or not. When training our models, some Deep Learning techniques such as early stopping, dropout and batch normalization were used. For a more in-depth explanation on these approaches, chapter 3 can be consulted.

4.4.1 Input features

In table 4, an overview of the four different input features sets can be found. In set A, only the closing prices will be used to predict closing prices in the future. Set B expands on this by adding the high and the low for each tick bar. In set C, the closing price will be used together with moving averages of length 5, 10 and 20. Lastly, set D will combine all features together.

Table 4: Overview of input feature sets

INPUT FEATURE	SET A	SET B	SET C	SET D
CLOSE	X	X	X	X
HIGH		X		X
LOW		X		X
MOVING AVERAGE 5			X	X
MOVING AVERAGE 10			X	X
MOVING AVERAGE 20			X	X

4.5 Evaluating the model

Knowing whether these above-mentioned performance metrics are good or not is hard to say based on the values itself. That is why the performance of the different models will be benchmarked against the performance of two different persistence models and an Auto Regressive Integrated Moving Average model (ARIMA). This benchmark will be done for the data of 2018, based on some performance measures.

In time series analysis, 5 performance measures are commonly used: mean absolute error (MAE), mean absolute percentage error (MAPE), mean squared error (MSE), root mean squared error (RMSE) and R^2 (RSQ). A full discussion for these metrics can be found in Hanke and Reitsch (1998) and Pindyck and Rubinfeld (1998). However, when comparing a CNN model trained on the differenced time series against a price-based persistence model, a transformation must be done, before a performance metric such as the MAPE can be used, as this metric uses the percentage-wise errors.

4.5.1 Persistence model

The first kind of benchmark model used to test the performance of our Deep Learning models, is the persistence model. Both its variants, price-based and difference-based, are explained in the previous chapter.

4.5.2 ARIMA model

The second type of benchmark model is the ARIMA model. The process for choosing the appropriate ARIMA model consist of starting with a less complex model and gradually increasing the number of parameters. While doing so, the Aikaike Information Criteria (AIC) is used as a goodness of fit measure. The lower this value, the better the goodness of fit. An indication of the order for the AR and MA component can be indicated by the Autocorrelation function (ACF) and the Partial Autocorrelation function (PACF).

However, as can be seen from the first set of the ACF and PACF, interdependencies occur between the autocorrelations. This can be resolved by taking the first order differences of the time series (I component of 1) and consequently plotting the same two functions. This conclusion can be backed by the test statistic score of 0.817 of the augmented dickey fuller test (Dickey & Fuller, 1979) which clearly does not reject the null hypothesis of a unit root and thus, indicates that the time series is not stationary. Note that these plots are made based on the closing price of the 610-tick bars, but the same conclusion could be drawn for the different aggregated tick bars.

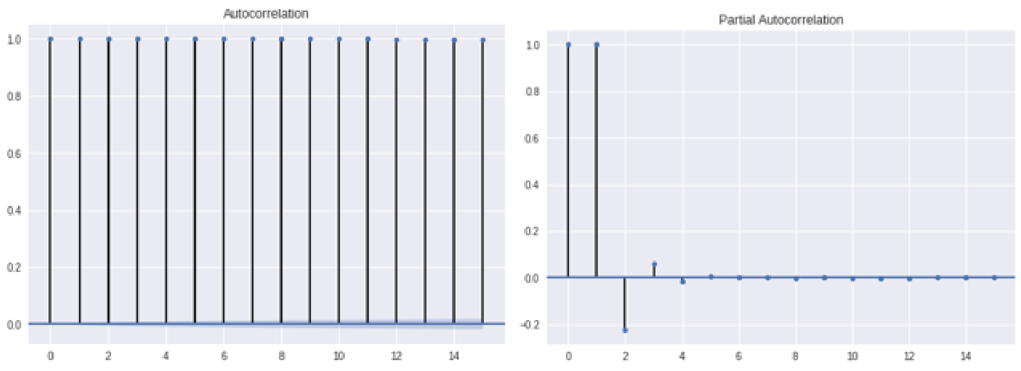


Figure 7: ACF and PACF for closing price of the 610-tick bars

When taking the first differences of the previous mentioned time series, an augmented Dickey-Fuller test now clearly shows that the time series is stationary with a test statistic of 7.96.10e-30. The second set of plots now shows the ACF and the PACF of the differenced time series. Whereas the previous set of plots showed some very high correlations due to the trend, this is no longer the case. This might indicate that the obtained ARIMA models will not be as significant as hoped for. Table X and X below show an overview of the different tested orders and their AIC-value. Note that an augmented Dickey-Fuller test was performed on the residuals of any given model and that the conclusion was identical for every model, confirming the validity of the constructed models.

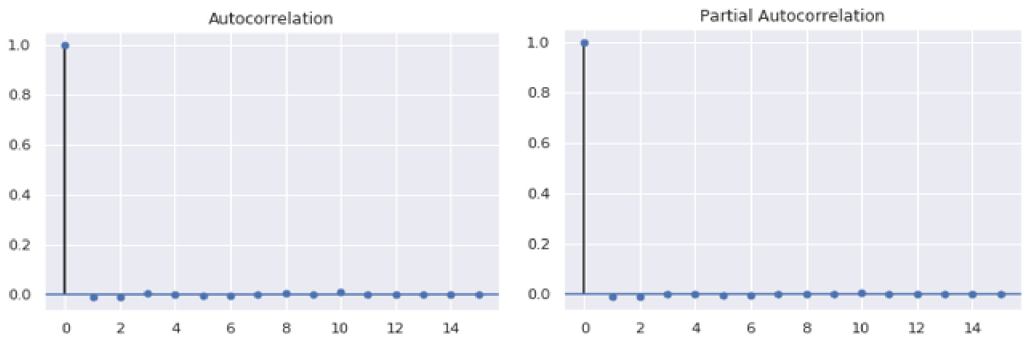


Figure 8: ACF and PACF of the closing price of the 610-tick bars after taking first differences

As can be seen in the two tables below, the lowest AIC was obtained by the ARIMA-model of order (0,1,2) for the 610-tick bars and order (1,1,1) for the 2584-tick bars. For more information on this model, McKenzie (2010) can be consulted. Furthermore, the residuals behaved like a random walk, which was backed by the Dickey-Fuller test score.

Nau (2014) and Armstrong et al. (2015) confirmed that the orders of the AR and MA component of an ARIMA should be kept quite small, which backs the low-order models used during this research.

Table 5: AIC values for different order ARIMA models

610-TICK BARS		**2584-TICK BARS**	
ORDER	**AIC**	**ORDER**	**AIC**
(1,1,0)	-4530815.72	(1,1,0)	-962646.50
(0,1,1)	-4530817.71	(0,1,1)	-962647.46
(0,1,2)	-4530872.20	(0,1,2)	-962650.00
(1,1,1)	-4530863.42	(1,1,1)	-962652.10
(2,1,1)	-4530869.89	(2,1,1)	-962647.61
(3,1,1)	-4530867.88	(3,1,1)	-962648.08
(1,1,2)	-4530870.30	(1,1,2)	AR- COMPONENT IS NOT STATIONARY
(2,1,0)	-4530871.86	(2,1,0)	-962649.52

4.6 Tuning the hyperparameters

When working with deep learning techniques, the performance of the models also relies on the correct choice of hyperparameters. One way to deal with this choice, is by running a grid search. This test different pre-defined combinations of hyperparameters for a small number of iterations (epochs). The most promising values are then chosen to be included in the final model.

Table 6: Tune grid for the CNN and LSTM

CNN		LSTM	
PARAMETER	**GRID SEARCH VALUES**	**PARAMETER**	**GRID SEARCH VALUES**
# FILTERS	16, 32, 64, 128	# LSTM UNITS	5, 10, 20 ,50
KERNEL SIZE	2, 4, 8, 16	DROPOUT RATE	0.05, 0.10, 0.20
# NODES IN DENSE LAYERS	20, 50, 100	# NODES IN DENSE LAYERS	20, 50, 100
# CONVULUTIONAL LAYERS	1, 2, 3	/	/
# HIDDEN DENSE LAYERS	1, 2, 3	# HIDDEN DENSE LAYERS	1, 2, 3
OPTIMIZER	ADAM, ADAGRAD, ADADELTA	OPTIMIZER	ADAM, ADAGRAD, ADADELTA
LEARNING RATE	0.001, 0.01, 0.1, 0.15, 0.2, 0.3	LEARNING RATE	0.001, 0.01, 0.1, 0.15, 0.2, 0.3

During this project, several grid searches have been performed in order to find optimal values for hyperparameters such as the learning rate, the optimal number of neurons in the first fully connected layer, or even the specific built of the convolutional layers or LSTM blocks. Table 6 summarizes the tune grids for both models. When performing the grid search, a standard stratified 3-fold cross validation was used. Based on the mean and the standard deviation of these results, the optimal set of hyperparameters were chosen.

4.7 Making Predictions

4.7.1 One step ahead

The final step in the process discussed by Yufeng (2017) consists of the deployment of the model or the actual predictions as he calls it. This means that a model can be implemented on real time data. Therefore, the validity of the training and test data must be true. This means, that at the time of prediction, all data concerning that prediction should be known and that the prediction interval is large enough for predictions to be made and implemented. In this case, a prediction can be made every batch of ticks, depending on which model that is being used. At that point in time, the aggregated groups of ticks are then used to predict the average closing price of the next batch of ticks. This can offer guidance in determining the trend by looking at the difference between the predicted value and the closing price of the last batch ticks.

4.7.2 Multiple step ahead

Another way to tackle this problem, is to perform a multiple step ahead forecast. In this case, multiple timepoints are predicted at once. Here, the intuition is that a possible upward or downward trend can be revealed over a longer period of time. This is particularly referring to the Deep Learning models, as the naïve approaches are not that appropriate for this task. During this thesis, a 5-step ahead forecast was used to test the relevancy of our models.

5 Results

Throughout this chapter, the results of all models will be reported individually. The discussion and comparison of these results will be done in chapter 6. The assessment of the models will be based on the MSE.

5.1 One-step forecasting

Here, we will briefly summarize the main results for the one-step ahead case. An overview of the results can be found in table 6.

5.1.1 Persistence model

5.1.1.1 Price-based

In figure 9, it can be seen that the predicted values are equal to the true values with a lag of one period. This is the most simple and naïve model and acts purely as a baseline. For the tick bars of size 610, an MSE of 1.176E-07 was found, while the 2584 sized tick bars showed an MSE of 4.742E-07.

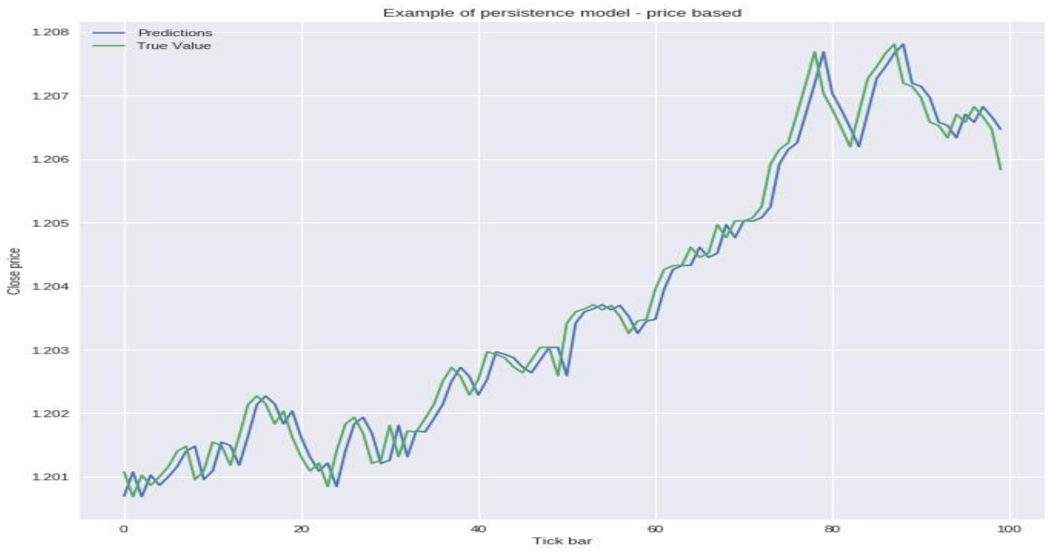


Figure 9: A sample of the true and predicted values for the price-based persistence model

5.1.1.2 Difference-based

Figure 10 demonstrates a sample of the predictions versus actuals in the case of a difference-based persistence model. Here as well, the model purely serves as a baseline. An MSE of 2.392E-07 and 9.591E-07 was obtained for the tick bars of size 610 and 2584 respectively.

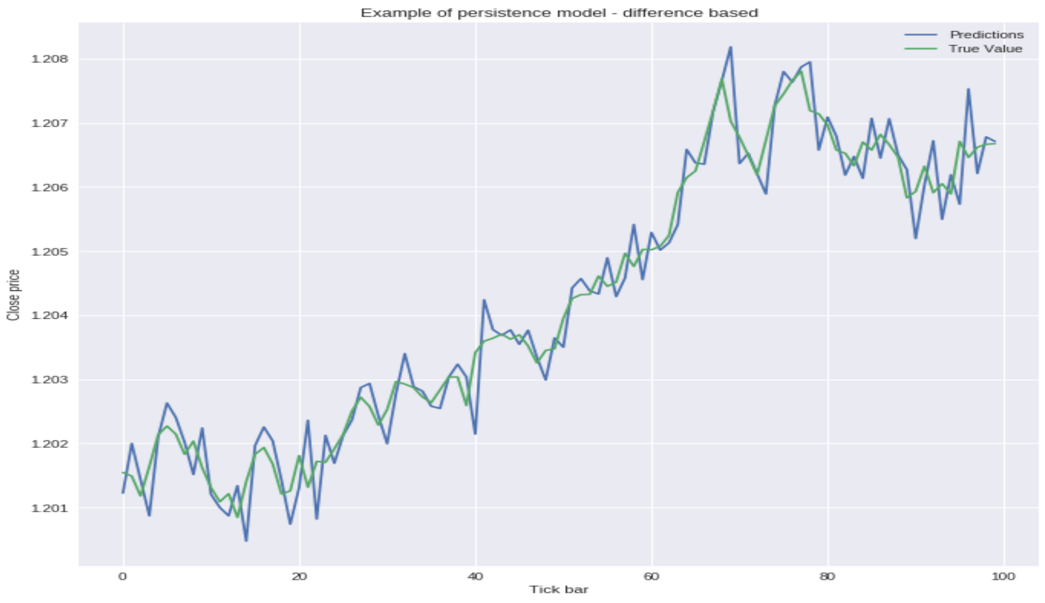


Figure 10: A sample of the true and predicted values for the difference-based persistence model

5.1.2 ARIMA

5.1.2.1 610 ticks

A third and final benchmark model which was constructed is the ARIMA. As shown in the methodology, section 4.5.2, the order for the 610-tick model was decided to be (0,1,2), where the MA-components have a value of respectively -1.537E-2 and -1.333E-2. Figure 11 plots the predictions versus actuals. Note that the similarity to the plot of the price-based persistency model can be explained by the absence of the AR- component and the small values for both Ma-components.

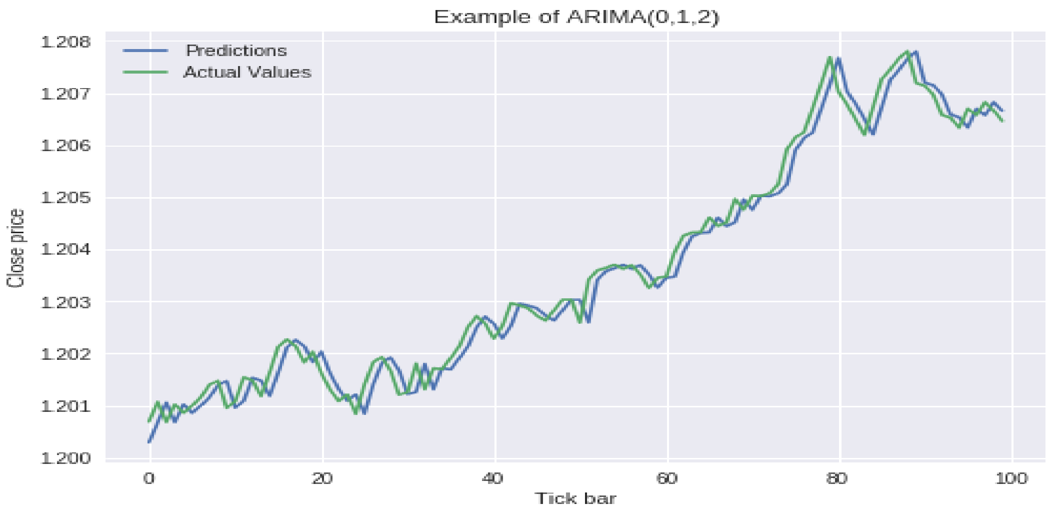


Figure 11: A sample of the true and predicted values for the ARIMA model

The MSE for this one-step ARIMA models was found to be 1.180E-07 and 4.740E-07.

5.1.2.2 2584 ticks

As was already mentioned, the order for the 2584-tick ARIMA model was decided to be (1,1,1), where the AR-component has a value of 2.575E-1 and the MA-component a value of 2.859E-1. Figure 12 shows the predictions versus actuals. Even though both components have at least one value, the plot still doesn't show any significant predictions, due to the fitted parameters to be so small.

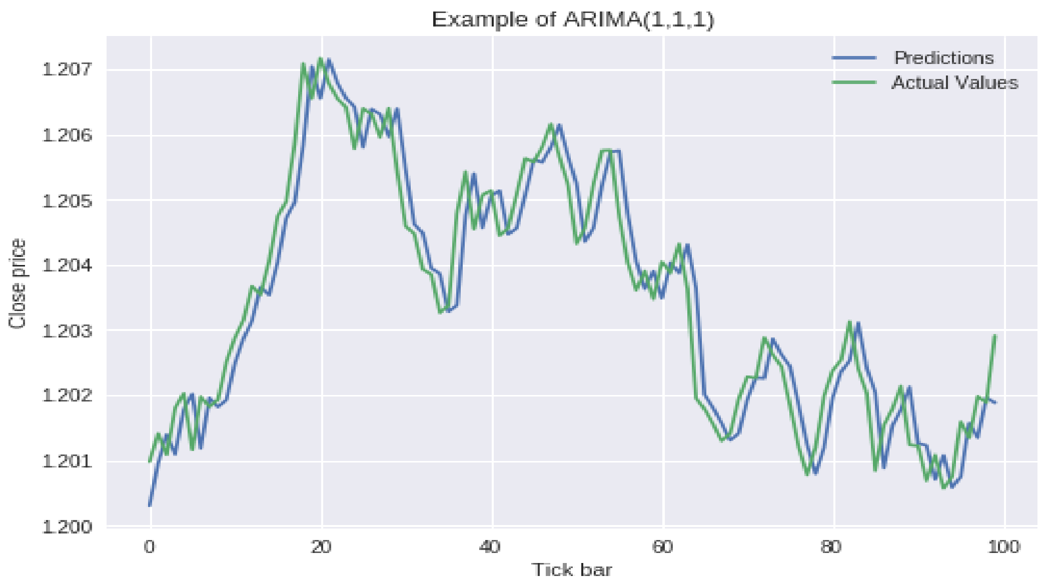


Figure 12: A sample of the true and predicted values for the ARIMA model

The MSE for this one-step ARIMA models was found to be 4.740E-07.

5.1.3 CNN

5.1.3.1 610 ticks

MSE for set-up A, B, C and D are respectively 1.178E-07, 1.196E-07, 1.180E-07, 1.300E-07. It is noteworthy to mention that for all of the set-ups, the CNN predicts a small constant value for the whole test set. Therefore, as can be seen in figure 13, the results look very similar to the results of the persistence model, something which will be further discussed in the next chapter.

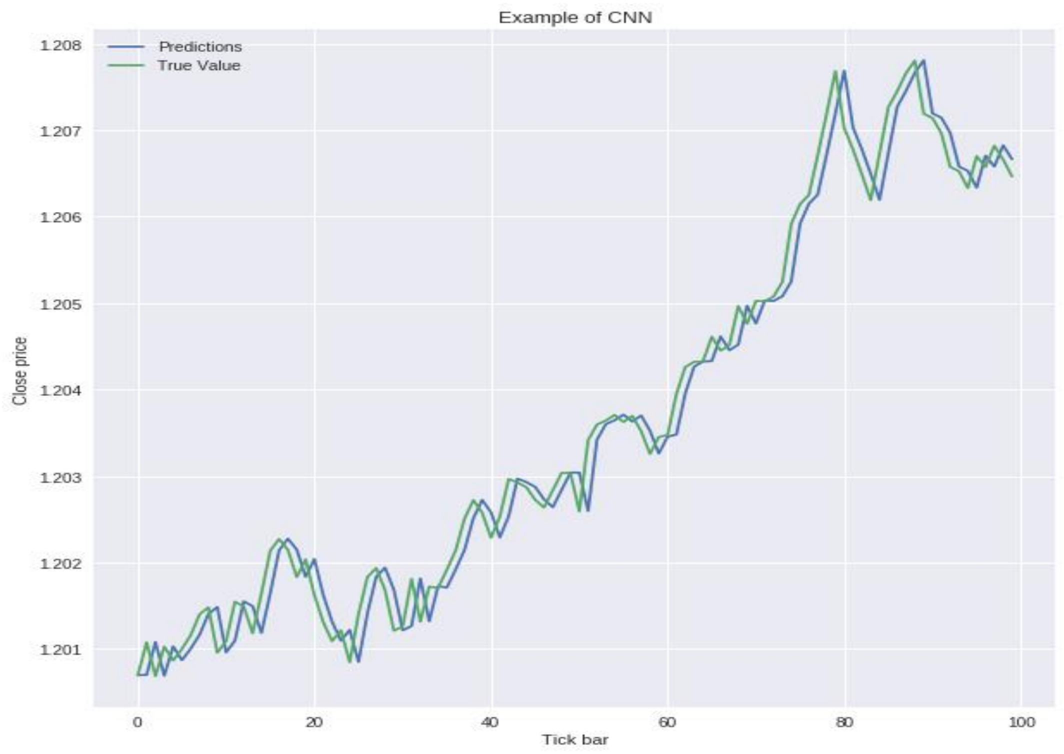


Figure 13: A sample of the true and predicted values for the CNN model – 610 ticks

5.1.3.2 2584 ticks

In contrast to the effect of constant predictions for the tick bars of size 610, the CNN resulted in a model which actually did make some interesting predictions. However, the MSE for the different set-ups are found to be higher than for the price-based persistence model: 5.749E-07 for set-up A, 6.224E-07 for set-up B, 5.155E-07 for set-up C, and 6.465E-07 for set-up D. Figure 14 demonstrates a sample of the true and predicted prices.

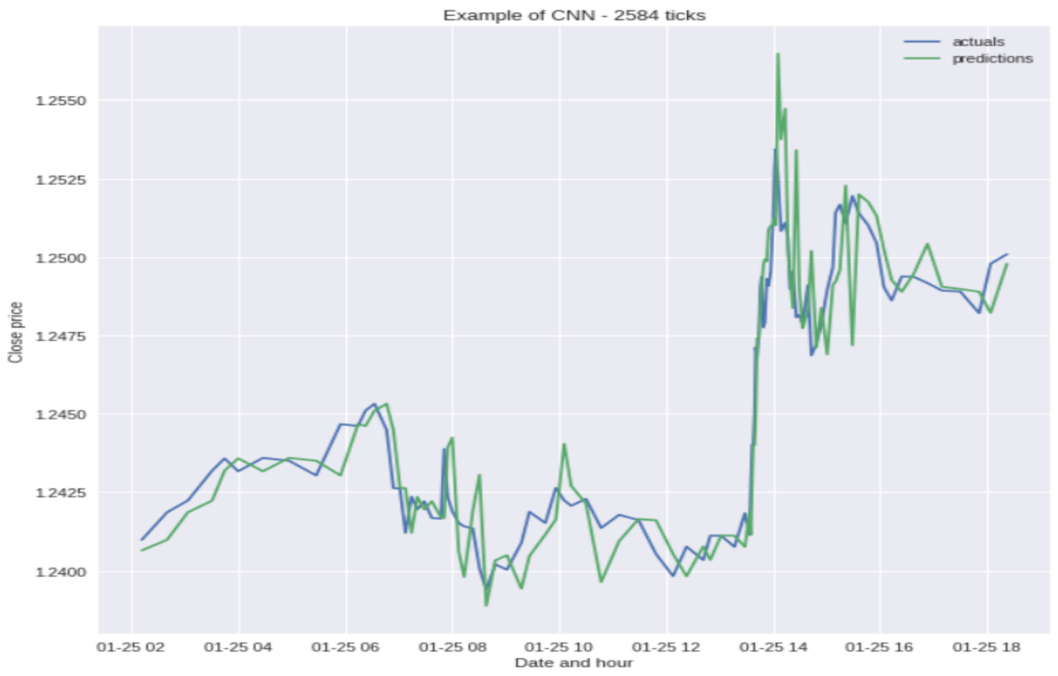


Figure 14: A sample of the true and predicted values for the CNN model – 2584 ticks

5.1.4 LSTM

5.1.4.1 610 ticks

Same as for the CNN, the LSTM predicts here a very small (nearly) constant value. The MSE's are respectively 1.176E-07, 1.194E-07, 1.416E-07, and 1.234E-07 for the different set-ups. A plot would be comparable to figure 13.

5.1.4.2 2584 ticks

In this case, the MSE's are respectively 4.745E-07, 4.745E-07, 4.753E-07, and 4.746E-07. The LSTM applied on the tick bars of size 2584 resulted in predictions which are not constant over time, though very small compared to the actuals. This will be further discussed during the next chapter.

5.2 Multi-step predictions

The results for each model and set-up for the multi-step prediction case can be found in table 6. These results will be discussed throughout the next chapter.

Table 7: Overview of results MSE

		ONE-STEP		MULTI-STEP (5)	
MODEL	SET-UP	610 TICKS	2584 TICKS	610 TICKS	2584 TICKS

PERSISTENCE PRICE	A	**1.176E-07**	4.742E-07	**2.367E-07**	**1.396E-06**
PERSISTENCE DIFF	A	2.392E-07	9.591E-07	4.653E-07	1.885E-06
ARIMA	A	1.180E-07	**4.740E-07**	3.390E-07	**1.396E-06**
CNN	A	1.178E-07	5.749E-07	3.430E-07	1.452E-06
	B	1.196E-07	6.224E-07	3.437E-07	1.454E-06
	C	1.180E-07	5.155E-07	3.434E-07	1.523E-06
	D	1.300E-07	6.465E-07	3.429E-07	1.443E-06
LSTM	A	**1.176E-07**	4.745E-07	3.412E-07	**1.396E-06**
	B	1.194E-07	4.745E-07	3.434E-07	1.397E-06
	C	1.416E-07	4.753E-07	3.430E-07	1.400E-06
	D	1.234E-07	4.746E-07	3.476E-07	1.400E-06

6 Discussion

In table 6, the MSEs for the different models and set-ups can be found. Based on this metric, it seems that both the CNN and LSTM did not manage to learn valuable information in order to predict future values. Below, a closer look is taken at the predictions of the fitted models.

One remarkable effect that could be observed when using tick bars of size 610 is that the predictions, both for the CNN as for the LSTM, are a constant value. This effect was first observed in the setting of one-step ahead forecasting, after which the decision was made to increase the complexity of the set-up. However, as can be seen in figure 15, the effect did not change.

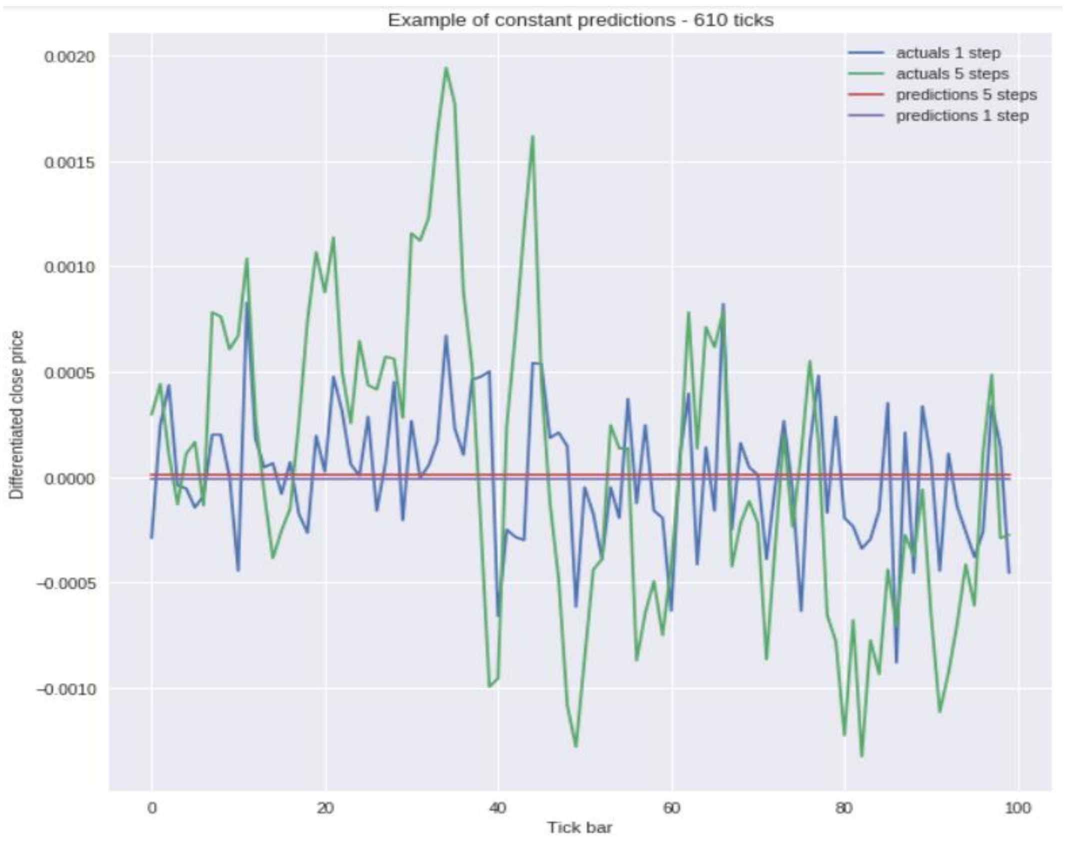


Figure 15: Example of the constant predictions - 610-tick bars

One possible explanation for why these predictions are constant, might be the granularity. By aggregating the data to 610-tick bars, the initial correlation between the adjacent closing prices on tick-by-tick basis are lost. This decision, however, was made to differentiate our approach to the approach of HFT as discussed by Gomber & Haferkorn (2013). However, when aggregating even further, some of the (unexplainable) volatility is eliminated, which might cause our models to perform better.

Therefore, the decision was made to look into forecasting with lower granularities, i.e. more ticks included per bar, by constructing 2584-tick bars. These delivered some useful predictions, which will now be discussed one by one.

6.1 CNN

Even though the MSE's for every CNN input feature set is worse than for the naïve baseline model, it seems like the model did learn to recognize some patterns. Figure 16 demonstrates the predicted differences versus actuals for the CNN.

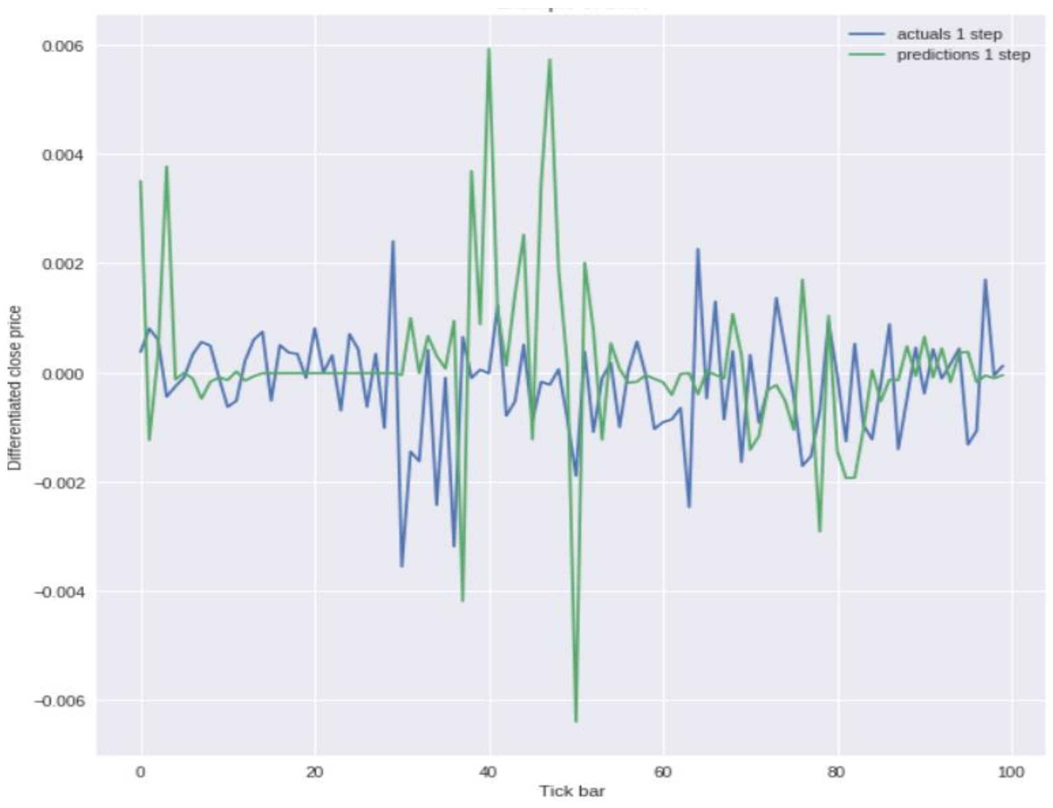


Figure 16: A sample of the predicted differences vs. actuals for the CNN – 2584-tick bars

This might at first not look like trustworthy predictions but could in fact give a robust forecast when applied on bigger timespans.

When comparing the performance between the different sets of input features, we can observe a slightly better result for input feature set C, where moving averages are taken into account.

6.2 LSTM

Overall, the LSTM seemed to perform better compared to the CNN in terms of MSE. The predictions, however, are again significantly smaller – but not constant. This can be observed in figure 17 and 18, where only the predicted differences are plotted, and then put into perspective by adding the actual values.

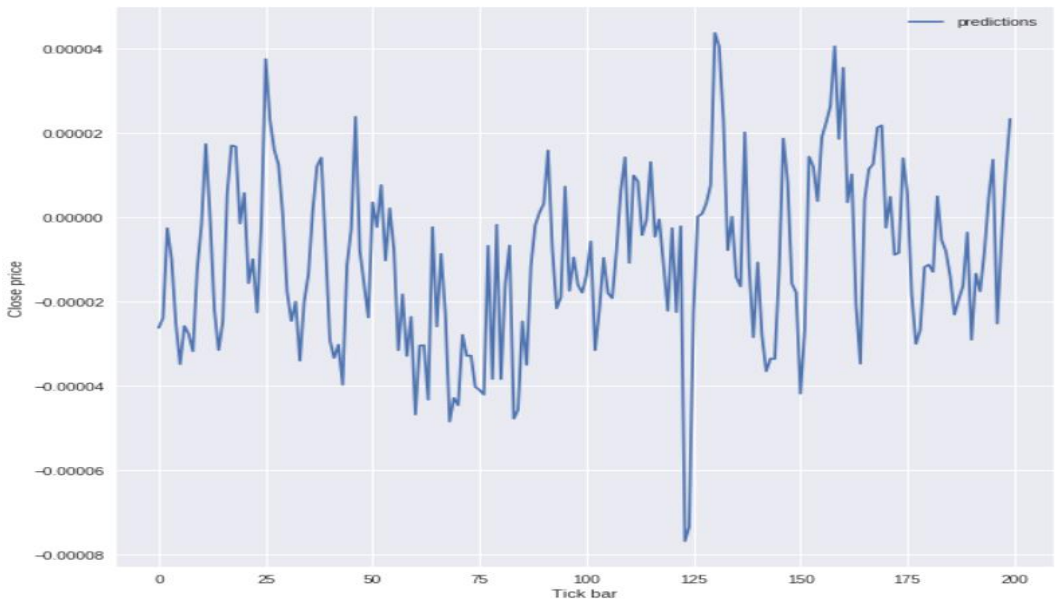


Figure 17: A sample of the predicted differences for the LSTM – 2584-tick bars

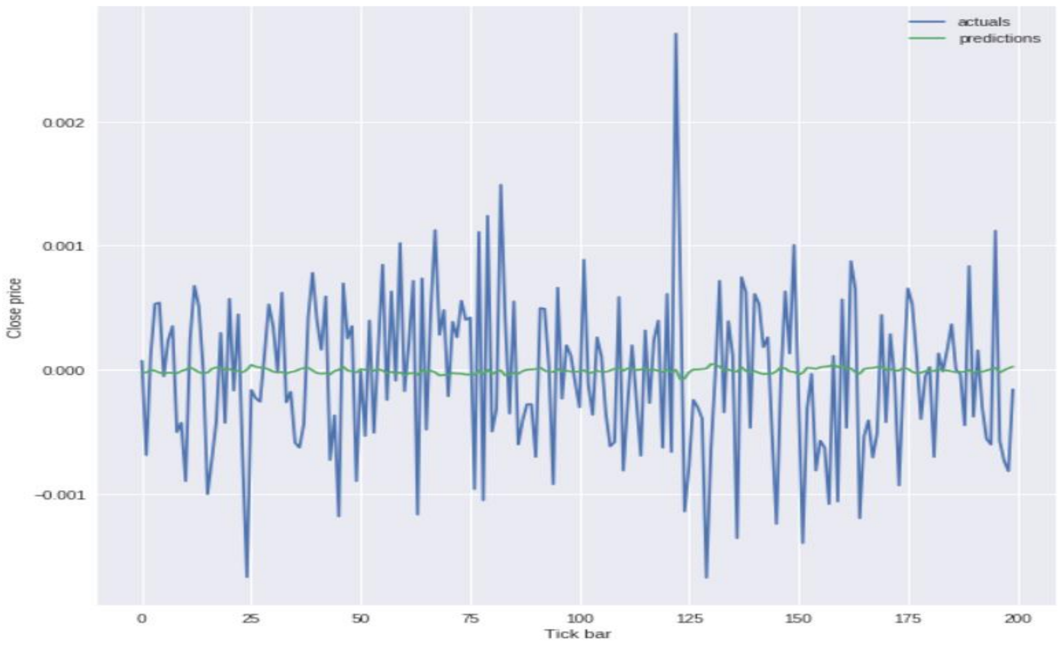


Figure 18: the predictions for the LSTM put into perspective

Although these predictions are not that significant compared to the actual differences, these predictions might contain some useful information regarding the sign of the forecasted differences, i.e. whether price will go up or down. This, however, is something that still needs to be investigated. Additionally, the non-constant learned outputs might suggest that something can be learned from the training data or that some patterns might be hidden in the data.

6.3 Comparison and remarks

Overall, it can be seen that input feature set A, which only uses the closing price as input feature, is in most cases the best performing. This result was quite surprising, since we expected the lows and highs of closing prices (set B), or the moving averages (set C) to contain additional information that could increase the performance of the model. In contrast, with the results obtained, it is not possible to conclude such statements. In fact, if we only look at MSE, we would have to conclude that the simplest model in most cases performs best.

Moreover, it seems that both the CNN and LSTM did not manage to outperform a naïve, price-based persistency model. However, the CNN did manage to learn some patterns and make significant predictions, while the LSTM did find some small values for the predicted differences. The exact reason for the significant differences in outcomes between the between the CNN and LSTM is still unclear. It would be interesting to see what would happen with the outcomes once the granularity is decreased somewhat more.

As we observed it, both the LSTM and CNN resulted in more valuable predictions after decreasing the granularity and increasing the number of steps predicted ahead. This is, as we believe, where deep learning starts to look more promising for predicting forex time series.

7 Conclusions

We set out to fill the research gap concerning the application of deep learning on intraday forex rates. To that end, we constructed two models based on state-of-the-art machine learning techniques for time series and compared their performance with both a naïve and a traditional approach. Our results suggest that predicting intraday forex returns, without scalping slight profits in huge volumes (i.e. high-frequency trading according to Gomber & Haferkorn, 2013), remains a challenging task. More specifically, we found that both a CNN and an LSTM did not manage to outperform a naïve model. In addition, the obtained results are not able to prove that adding additional informative input features would improve the predictive performance. To this extent, we found that our univariate models, which only take closing prices as an input, often perform better than their multivariate versions which took lows and highs or moving averages into the set of input features. Lastly, we found that lowering the granularity of input data and increasing the number of steps predicted ahead is beneficial for the usefulness of predictions based on our deep learning models.

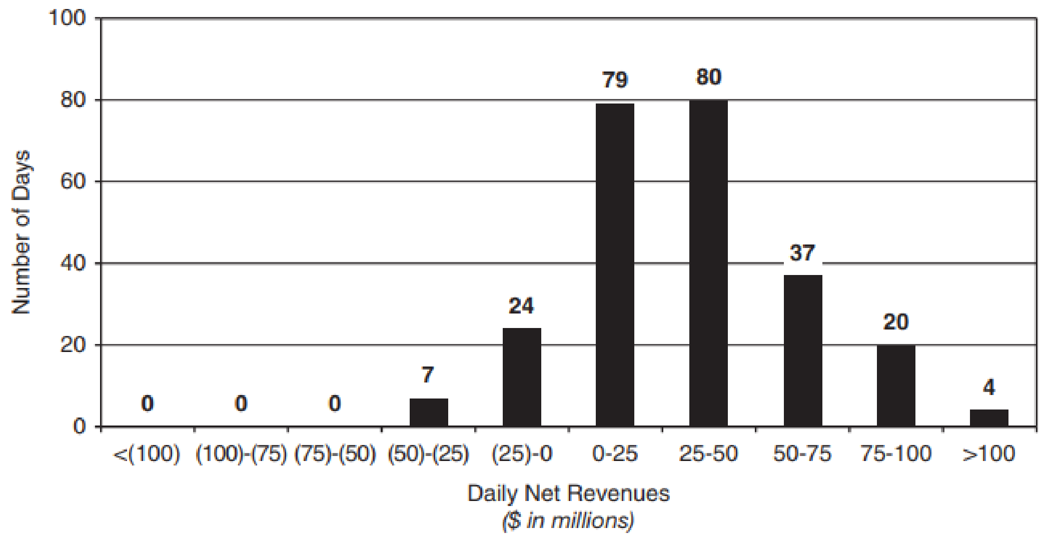
There are three additional things that could be tried in order to improve our models. Firstly, the models could be fitted for larger forecast horizons and lower granularity of input data. Secondly, it would be interesting to see what the effect is of using time bars instead of tick bars. And thirdly, it might be possible that increasing the complexity of the models could improve performance, in the hope that the increased complexity would be able to detect certain patterns which our models were not able to find.

Overall, the usefulness of Deep Learning to predict intraday foreign exchange rates has not been proven yet. Therefore, it would not be beneficial thus far to construct a trading robot for this task. However, when expanding the forecast horizon and lowering the granularity of the tick bars, the deep learning approaches start to seem more valuable. This would be an interesting path for future research.

An important question to be answered in future research is whether our conclusion remains valid on other currency pairs, or whether adding other informative data such as bond yields would increase the performance of intraday predictive models.

Appendices

Appendix 1: Daily net revenues of Goldman Sachs in 2017



Daily net revenues of Goldman Sachs in 2017 (Goldman Sachs 2017 Form 10-K, p.95)

Sources

Ahmed, N., Atiya, A., El Gayar & N., El-Shishiny, H. (2010). An empirical comparison of machine learning models for time series forecasting. Econometric Revieuws, Vol. 29, Ro. 5-6, pp. 594-621.

Armstrong, J & Green, Kesten & Graefe, Andreas. (2015). Golden Rule of Forecasting: Be Conservative. SSRN Electronic Journal. 10.2139/ssrn.2643546.

Bao W, Yue J, Rao Y (2017) A deep learning framework for financial time series using stacked autoencoders and long-short term memory. PLoS ONE 12(7): e0180944. https://doi.org/10.1371/journal.pone.0180944

Borovykh, A. W., Bohte, S., & Oosterlee, C. (2017). Conditional time series forecasting with convolutional neural networks. Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 10614, 729-730.

Brock, W. A., D. A., Hsieh, and B. LeBaron. 1991. Nonlinear dynamics, chaos, and instability. Cambridge, Mass.:MIT Press.

Brooks C (1996) Testing for non-linearity in daily sterling exchange rates. Appl Financ Econ 6(4): pp 307–317. doi:10.1080/096031096334105

Chiarella C, Peat M, Stevenson M (1994) Detecting and modelling nonlinearity in flexible exchange rate time series. Asia Pac J Manag 11(2):159–186. doi:10.1007/BF01739197

Claesen, M., & De Moor, B. (2015). Hyperparameter Search in Machine Learning.

Cont, R, Empirical properties of asset returns: Stylized facts and statistical issues, (2001)

Deng S, Yoshiyama K, Mitsubuchi T, Sakurai A (2015) Hybrid method of multiple kernel learning and genetic algorithm for forecasting short-term foreign exchange rates. Comput Econ 45(1):49–89. doi:10.1007/s10614-013-9407-6

De Grauwe, P., H. Dewachter, and M. Embrechts. 1993. Exchange rate theory. Chaotic models of foreign exchange markets. London: Blackwell.

Dickey, D., & Fuller, W. (1979). Distribution of the Estimators for Autoregressive Time Series With a Unit Root. Journal of the American Statistical Association, 74(366), 427-431.

Dunis, C. & Williams, M. (2002). Modelling and trading the euro/US dollar exchange rate: Do neural network models perform better?, Derivatives use, trading and regulation. Vol.8(3), pp.211-239

Etzkorn, M. & Schwager, D., (2016). A Complete Guide to the Futures Market: Fundamental Analysis, Technical Analysis, Trading, Spreads and Options. Somerset : John Wiley & Sons, Incorporated.

Fang, H., K. S. Lai, and M. Lai. 1994. Fractal structure in currency futures price dynamics. Journal of Futures Markets 14:169±181.

Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. European Journal of Operational Research, 270(2), 654-669. doi:10.1016/j.ejor.2017.11.054

Folger, J. (2013) A closer look at price chart choices. Futures, Vol.42(7), pp.30-32

Gers, F. A., Schmidhuber, J., and Cummins, F. (2000). Learning to Forget: Continual Prediction with LSTM. Neural Computation, 12(10):2451–2471

Giles, C. L., Lawrence, S., Tsoi, A. C., 2001. Noisy time series prediction using recurrent neural networks and grammatical inference. Machine learning 44 (1), 161-183.

Gomber, P., & Haferkorn, M. (2013). High-Frequency-Trading. Business & Information Systems Engineering, 5(2), 97-99.

Gooijer, J. G., & Hyndman, R. J. (2006). 25 years of time series forecasting. International Journal of Forecasting, 22(3), 443-473. doi:10.1016/j.ijforecast.2006.01.001

Hamilton, J. D. (1994) Time series analysis, vol. 2, Princeton university press Princeton.

Heaton, J., Polson, N., & Witte, J. (2016). Deep Learning for Finance: Deep Portfolios. SSRN Electronic Journal. doi:10.2139/ssrn.2838013

Hernane Spatti, D., Andrade Flauzino, R., Liboni, L. H. B., & Dos Reis Alves, S. F.. (2017). Artificial Neural Networks: A Practical Course. Cham: Springer International Publishing : Imprint: Springer.

Joseph, A., Larrain, M., & Singh, E. (2011, 1 1). Predictive Ability of the Interest Rate Spread Using Neural Networks. Procedia Computer Science, 6, 207-212.

Kipruto, G., Mung'atu,J., Orwa, G. & Gathimba, N. (2018). Application of Artificial Neural Network (Ann) in Modeling Foreign Currency Exchange Rates. International Journal of Scientific Research and Management. 6.

Liu, C., Hou, W., & Liu, D. (2017, 12). Foreign Exchange Rates Forecasting with Convolutional Neural Network. Neural Processing Letters, 46(3), 1095-1119.

Lopez de Prado, M. (2018) Advances in financial machine learning. New Jersey: Wiley.

McKenzie, E., Gardner, E. (2010). Damped trend exponential smoothing: A modelling viewpoint. International Journal of Forecasting. 26(4), 661-665.

Mehta, M. (1995), 'Foreign Exchange Markets', 176-198, in A. N. Refenes [ed.], Neural Networks in the Capital Markets, John Wiley, Chichester.

Pacelli, V., Bevilacqua, V., & Azzollini, M. (2011). An Artificial Neural Network Model to Forecast Exchange Rates. Journal of Intelligent Learning Systems and Applications, 03(02), pp 57-69. doi:10.4236/jilsa.2011.32008

Pascanu, R., Mikolov, T., & Bengio, Y. (2012). Understanding the exploding gradient problem. CoRR, abs/1211.5063, 2.

Prechelt, L. (2012). Early stopping - But when? Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 7700, 53-67.

Sattler, K.-U., Schallehn, E. (2001). A Data preparation framework based on a multidatabase language. Proceedings of the International Symposium on Database Engineering & Applications, pp. 219–228.

Srivastava, N. et al., 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal Of Machine Learning Research, 15, pp.1929–1958.

Taylor, S. J. 1986. Modelling financial time series. New York: Wiley

Tenti, P. (1996) Forecasting foreign exchange rates using recurrent neural networks, Applied Artificial Intelligence, 10:6, 567-582, DOI: 10.1080/088395196118434

Tsai, Y., Chen, J. & Wang, J. (2018). Predict Forex Trend via Convolutional Neural Networks. Journal of Intelligent Systems, 0(0), pp. -. Retrieved 13 Apr. 2019, from doi:10.1515/jisys-2018-0074

Internet

Brownlee, J., (2016, December 26). How to Make Baseline Predictions for Time Series Forecasting with Python Retrieved from https://machinelearningmastery.com/persistence-time-series-forecasting-with-python/

Burns, S., (2018, May 10). 15 Interesting Facts About the Forex Market – Infographic. *New Trader U*, Retrieved from http://www.newtraderu.com/2018/05/10/15-interesting-facts-about-the-forex-market-infographic/

CME Institute (sd). Understanding Futures Expiration & Contract Roll. *CME Group*, Retrieved from https://institute.cmegroup.com/courses/introduction-to-futures-html/modules/understanding-futures-expiration-contract-roll

Gardner, E. & McKenzie, E. (2009, October 22). Why the damped trend works. Retrieved from https://www.bauer.uh.edu/gardner/docs/pdf/Why-the-damped-trend-works.pdf

Kumar, N. (2017, December 5). How AI will invade every corner of Wall Street. *Bloomberg*, Retrieved from https://www.bloomberg.com/news/features/2017-12-05/how-ai-will-invade-every-corner-of-wall-street

Nau, R. (2014, October 30). Notes on nonseasonal ARIMA models. Retrieved from http://people.duke.edu/~rnau/Notes_on_nonseasonal_ARIMA_models--Robert_Nau.pdf

Turner, M. (2016, January 21). Goldman Sachs: We're investing deeply in artificial intelligence. *Business Insider*, Retrieved from http://uk.businessinsider.com/goldman-sachs-investing-in-artificial-intelligence-2016-1?IR=T&r=US

UNITED STATES SECURITIES AND EXCHANGE COMMISSION: Form 10-K. *Goldman Sachs*, Retrieved from https://www.goldmansachs.com/investor-relations/financials/current/10k/2017-10-k.pdf

Vonko, D. (2019, January 24). Understanding Fibonacci Numbers and Their Value as a Research Tool. *Investopedia*, Retrieved from https://www.investopedia.com/articles/trading/06/fibonacci.asp

Yufeng, D. (2017, August 31). The 7 Steps of Machine Learning. *Towards Data Science*, Retrieved from https://towardsdatascience.com/the-7-steps-of-machine-learning-2877d7e5548e