Electronics and Computer Science
Faculty of Physical Sciences and Engineering University of
Southampton

Konstantinos Papadopoulos

2019 - 2020

Improving RNN performance for financial time series forecasting

Project supervisor:
Dr Gary Wills

Second examiner:
Dr Sebastian Stein

A project report submitted for the award of

BSc Computer Science

Abstract

This report presents a novel application of improving the performance of LSTM/GRU neural networks by assessing their optimal configuration and introducing transformation methods in order to remove short term noise from the input data.

It compares the results of training an optimally configured model on the initial history of 30 different asset prices against data processed through a Kalman Filter, a Fourier Transform or a Wavelet Transform. This model is optimized for stock price prediction but can be applicable for other asset classes as well.

It sequentially tries to determine the best network configuration for this task along with introducing three of the most popular transformation methods from the fields of Physics/Mathematics to optimize the accuracy of the network further. In order to highlight the practical difference between the unfiltered and filtered approach for time series forecasting, the notion of profit is introduced and backtesting is being performed to evaluate the historical returns generated from each method, assuming a simple backtesting strategy. Finally, the report presents a T-test to highlight the statistical significance of the returns achieved by both the unfiltered and filtered versions.

As a future work, the ideas presented in this project can be used and adapted to create an automated trading strategy that improves existing machine learning approaches by avoiding short term noise in time series forecasting. It can also be expanded to account for other asset classes, like derivates and commodities, or additional fundamental signals like EPS, P/E or EBITDA.

Acknowledgements

I would like to thank my first and second supervisor, Dr Gary Wills and Dr Sebastian Stein respectively, who supported me throughout my project with feedback and advice. I would also like to thank the university’s ECS department for providing me unlimited and free access to important resources.

Statement of Originality

- I have read and understood the ECS Academic Integrity information and the University's Academic Integrity Guidance for Students.
- I am aware that failure to act in accordance with the Regulations Governing Academic Integrity may lead to the imposition of penalties which, for the most serious cases, may include termination of programme.
- I consent to the University copying and distributing any or all of my work in any form and using third parties (who may be based outside the EU/EEA) to verify whether my work contains plagiarised material, and for quality assurance purposes.

You must change the statements in the boxes if you do not agree with them.

We expect you to acknowledge all sources of information (e.g. ideas, algorithms, data) using citations. You must also put quotation marks around any sections of text that you have copied without paraphrasing. If any figures or tables have been taken or modified from another source, you must explain this in the caption and cite the original source.

I have acknowledged all sources, and identified any content taken from elsewhere.

If you have used any code (e.g. open-source code), reference designs, or similar resources that have been produced by anyone else, you must list them in the box below. In the report, you must explain what was used and how it relates to the work you have done.

I have not used any resources produced by anyone else.

You can consult with module teaching staff/demonstrators, but you should not show anyone else your work
(this includes uploading your work to publicly-accessible repositories e.g. Github, unless expressly permitted by the module leader), or help them to do theirs. For individual assignments, we expect you to work on your own. For group assignments, we expect that you work only with your allocated group. You must get permission in writing from the module teaching staff before you seek outside assistance, e.g. a proofreading service, and declare it here.

I did all the work myself, or with my allocated group, and have not helped anyone else.

We expect that you have not fabricated, modified or distorted any data, evidence, references, experimental results, or other material used or presented in the report. You must clearly describe your experiments and how the results were obtained, and include all data, source code and/or designs (either in the report, or submitted as a separate file) so that your results could be reproduced.

The material in the report is genuine, and I have included all my data/code/designs.

We expect that you have not previously submitted any part of this work for another assessment. You must get permission in writing from the module teaching staff before re-using any of your previously submitted work for this assessment.

I have not submitted any part of this work for another assessment.

If your work involved research/studies (including surveys) on human participants, their cells or data, or on animals, you must have been granted ethical approval before the work was carried out, and any experiments must have followed these requirements. You must give details of this in the report, and list the ethical approval reference number(s) in the box below.

My work did not involve human participants, their cells or data, or animals.

Contents

Chapter 1 Introduction

Stock price prediction is a problem that has always occupied some of the brightest minds in the financial industry, ranging from hedge funds, portfolio managers, wall street traders and most recently high frequency trading firms. Technically, trading is a zero sum game [1] from an accounting point of view, meaning that all the participants are competing against each other in order to generate profit.

1.1 Problem

For decades investors were used to analyse all the information they were getting from the exchanges manually, which was taking a lot of workhours and was susceptible to human calculation errors while also limiting the spectrum of instruments they were able to analyse within this limited time. Nowadays, this is not the case anymore, since automated systems are responsible for processing terabytes of information and in some cases for making automated investing decisions based on them. The new unit of measurement is no longer the minute or second, but the nanosecond [2]. There are numerous trading strategies that have been developed that are taking advantage of the capabilities of modern hardware and machine learning in order to make accurate predictions regarding the future price of an instrument.

1.2 Goals

The goal of this project is designing an algorithm to predict the future price of an asset, given historical data of a US listed stock. It will evaluate the performance of recurrent neural network architectures for financial time series forecasting and assess potential improvements by applying methods such as Kalman filtering or wavelet transforms.

1.3. Objectives

The objectives of the project are:

- Developing a system that learns from historical data
- Applying a recurrent neural network architecture for future price prediction
- Evaluating the performance of the algorithm
- Researching filtering methods for further improvement
- Assessing its application to the financial industry

1.4 Scope

Historical data will be used to train and backtest the machine learning algorithm in order to measure its performance. Sentiment and news analysis are out of scope for this project. That is due to various reasons. First, getting historical data about news releases and the affect they had on various stock prices would be a very difficult task on its own. Also, news can create a very volatile environment in which market makers have an advantage, due to their advanced infrastructure which enables them to act faster than the rest of the participants [3]. Apart from that, news analysis would come with the overhead of having to filter the sources for their authenticity and detect fake news before executing a trade based on them. Therefore, I decided to focus on pricing history and other metrics that promise a more accurate prediction about the underlying price of an asset. Finally, if the results of the project turn out promising, the model could easily be adjusted and expanded for other markets and instruments, like currencies, options and bonds.

Chapter 2 Literature Review

In recent years, many techniques have been studied and applied from researchers and financial firms in order to facilitate the future price discovery of a stock. Some of those techniques are based on traditional methods that evaluate the underlying or intrinsic value of the asset to predict future price corrections. Others apply a more quantitative approach adopting some of the latest trends in statistical analysis and machine learning. This chapter aims to provide an overview of the most popular predicting models and the contribution they have made to this industry, focusing especially on the machine learning models that will contribute to the later development of this project.

2.1 Fundamental Analysis

According to the efficient market hypothesis [4], the price of a stock fully reflects all the information about it, since investors quickly act based on new data like earnings reports, press releases, insider selling etc. However, this hypothesis while theoretically true is widely disregarded by investors and portfolio managers who believe that markets are not efficient [5] and are trying to exploit those inefficiencies by finding and investing in undervalued stocks or short selling overvalued assets. A fundamental analyst will use the reports of a publicly traded company in order to evaluate its financial condition across three dimensions, concerning the economy, the industry and the firm itself [6]. They will use different financial ratios like price to earnings (PE), earnings per share (EPS), return on equity (ROE), market capitalization (MC), debt to equity (D/E) to assess the firm's growth, and as a result the future price of its share. There are various stock valuation models, that based on past data are trying to predict the future value of a share. For example, the popular Discounted Cash Flow (DCF) analysis values the company on basis of the net present value (NPV) of its future cash flows, discounted by an appropriate discount rate [7]. The DCF formula is described below.

$$NPV = \sum_{t=0}^{n} \frac{FCFt}{(1+r)^{t}}$$

Equation 1 Net Present Value equation

Here, the NPV or DCF is the sum of all future discounted cash flows that the investment is expected to produce. This is the fair or intrinsic value of the stock. FCF_t are all the cash flows the company is going to generate for any given year (meaning FCF_1 is the cash flows for year 1, FCF_2 for year 2 etc). Finally, r is the discount rate represented in decimal form, which is the target rate of return on the investment.

Fundamental investors also perform comparative analysis between firms of the same industry to find discrepancies along with considering qualitative fundamentals like a firm's business model, any competitive advantage, regulations and management decisions. This approach got widespread due to its adoption from prominent value investors, including Benjamin Graham and Warren Buffett [8].

2.2 Sentiment Analysis

Apart from fundamental ratios, financial news is considered to have a high impact on stock price returns. A positive article from Bloomberg or Reuters about a company's earnings, the optimistic opinion of a wall street analyst, even a simple tweet from a president regarding international trade developments can significantly shift investors' sentiment regarding either the overall economy or a specific industry/firm [9].

The applications of this field have been researched extensively in finance. Niederhoffer examined the correlation between news articles from the New York Times and the movement in stock prices, by classifying each article into a semantic domain of 20 categories [10]. Tetlock explored the interactions between media content and stock market activity, by analysing Wall Street Journal's articles corresponding to ether negative investor sentiment or risk aversion. He observed that trading volume increased after the release of a pessimistic report, which was also followed by a down trend and reversion of market prices [11]. Another study by Mittal and Goel found that an automated strategy predicting the Down Jones Industrial Average (DJIA) based on the results of capturing public sentiment from Twitter can be quite profitable [12].

While those approaches were proven quite successful, they worked in a quite simplistic way known as "bag-of-word" approaches, according to which news articles are analysed by a vector space model that translates each news piece into a vector of word statistical measurements, such as the number of occurrences, etc. According to Xiaodong Li et al [13], those approaches fail to capture the underlying sentiment behind each news article. Therefore, they studied the problem from another dimension according to which they would measure sentiment not only by word frequency but by decomposing each word into a vector of sentiment features (for example the word "accept" = "positive" + "submit" + "social relation"). They constructed a generic stock price prediction framework, into which they were later able to test and evaluate six different models. This framework worked like below.

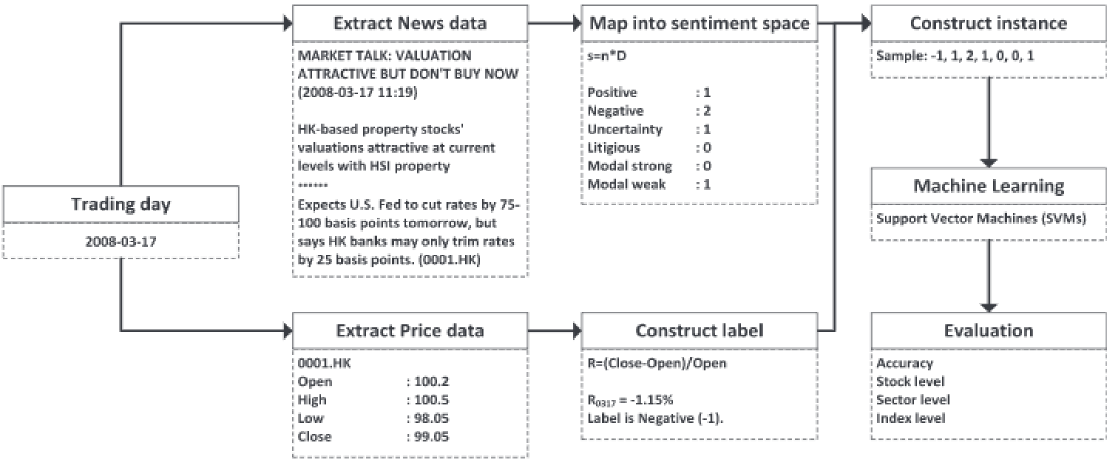


Figure 1 Mittal and Goel generic stock price prediction framework [13]

For each trading day, they collected a stock's price data from the Hong Kong exchange along with any relevant news articles. First, they pre-processed each price into a value R=(Close-Open)/Open, and then they translated each article n_i into a vector of term frequency values [t_1 t_2 … t_m]. Therefore, all those news articles combined would generate a matrix N like below.

$$N = \begin{bmatrix} n_1 \\ n_2 \\ \vdots \\ n_l \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} & \dots & t_{1m} \\ t_{21} & t_{22} & \dots & t_{2m} \\ \vdots & & \ddots & \vdots \\ t_{l1} & t_{l2} & \dots & t_{lm} \end{bmatrix}$$

Figure 2 Matrix N containing the articles as vectors of term frequency values [13]

They used a matrix D, where each row corresponded to one word and each column to a sentiment direction in order to map the matrix N to the sentiment space S as seen above. Then they labelled each row in S according to the daily return R calculated as shown above. So, if R was above a certain threshold, the label would be positive; otherwise negative. By applying support vector machines (SVMs) to train the dataset, they finally concluded that this approach outperformed the "bag-of-words" model and that simply using "positive" or "negative" as the sentiment dimensions is not enough for a satisfying accuracy; hence a wider sentiment space is needed [13]

2.3 Machine Learning Approach

2.3.1 Linear Regression

Linear and logistic regression are both some techniques which have been used in the past but are not really preferred anymore, due to the advancements of the neural network area, that offers more promising results. However, we are going to provide a brief overview on those methods.

Linear regression is a simple technique, according to which the output variable, which in our case is the price of the share depends on a list of independent features, like the daily volume, previous closing price, market capitalization etc. According to Nunno [14], a simplistic linear regression model trained on historical prices of AAPL stock although satisfying is not enough for portfolio optimization, since it depends greatly from the choice of the training window. Large training windows face the issue of overfitting as seen below.

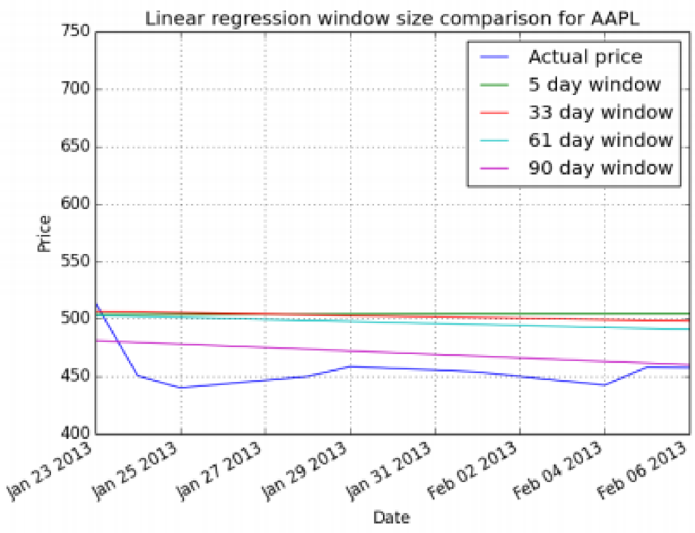
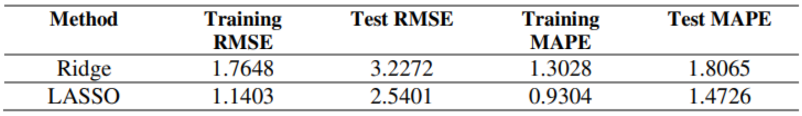


Figure 3 Linear Regression window size comparison for AAPL [14]

Sanjiban Sekhar Roy et al. [15] researched the application of a LASSO linear regression model in stock time series forecasting, benchmarking the results against those of a Ridge Regressor. Using the historical price dataset for the GS stock, they concluded that the LASSO regressor is a better choice according to the error metrics they used to assess the accuracy of the prediction.



Method	Training RMSE	Test RMSE	Training MAPE	Test MAPE
Ridge	1.7648	3.2272	1.3028	1.8065
LASSO	1.1403	2.5401	0.9304	1.4726

Figure 4 Comparison between Ridge and Lasso Regression for GS [15]

2.3.2 Support Vector Machines

Support vector machines (SVMs) are popular machine learning tools used for classification and regression first discovered by Vladimir Vapnik [16].

Mukherjee et al. [17], Tay and Cao [18] and Kyoung-jae [19] have examined the applications of SVMs in financial time series forecasting and have studied the advantages they offer over more traditional back propagation (BP) algorithms, proving their outperformance in terms of error metrics like mean squared error. SVMs try to solve the overfitting problems introduced by BP networks as well as the necessity of having to tune multiple parameters for a model to work. They implement the structural risk minimization principle according to which they try to minimize the generalization error rather than the training error.

In their study, Tay and Cao [18] extracted data of five future contracts from the Chicago Mercantile Market and compared the results of an SVM architecture to the classical BP network. From the pictures below, it seems like the Normalized Mean Square Error (NMSE) keeps decreasing for the BP network at a fast rate at the training set, however it decreases a bit before it starts increasing again for the validation set. That suggests that the model was susceptible to overfitting. In contrast, the SVM approach seems to keep the NMSE close to a constant value for both the training and test dataset.

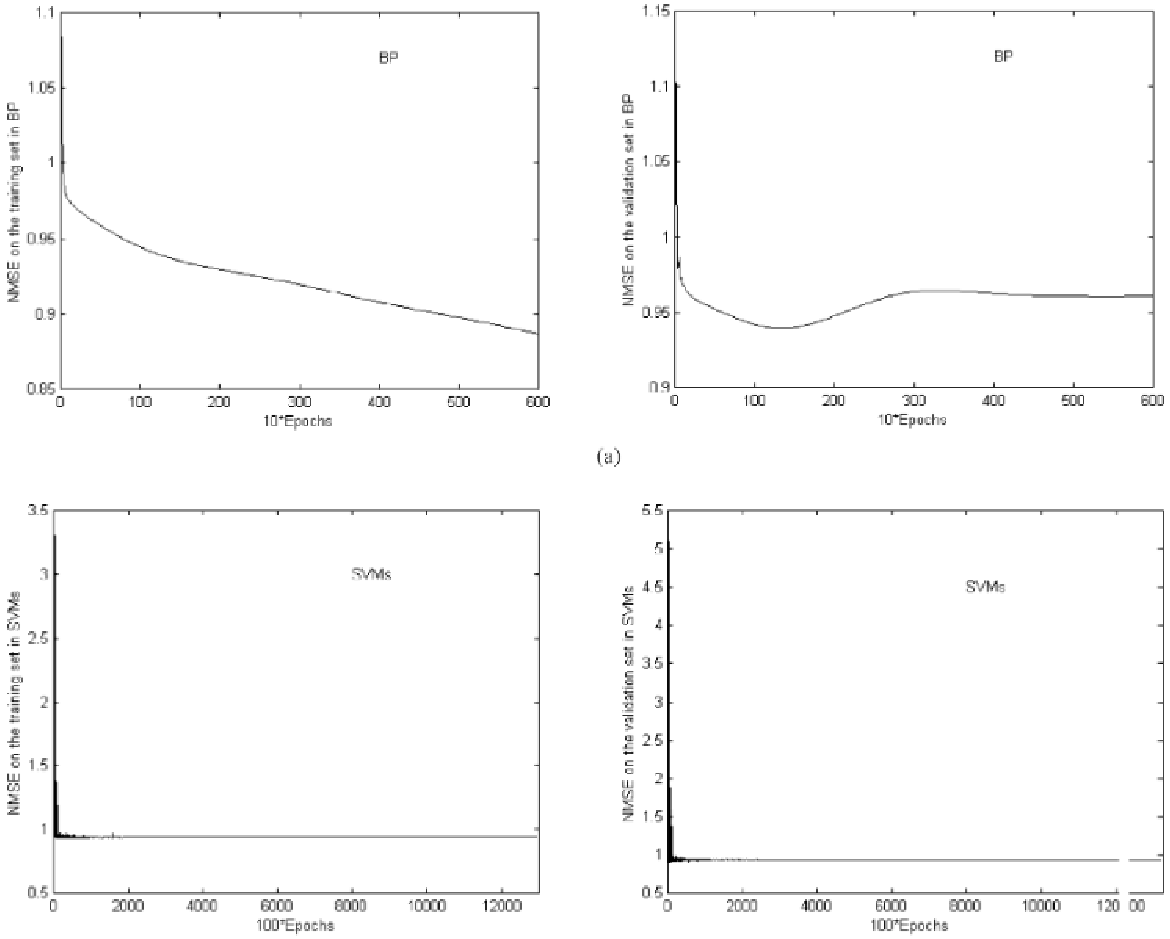


Figure 5 NMSE of training and validation set for BP network and SVM model [18]

2.3.3 Reinforcement Learning

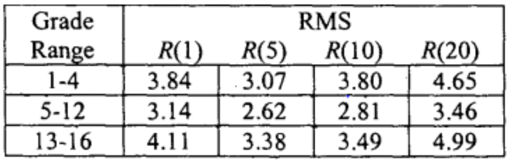
Reinforcement learning is the area of machine learning that studies the optimal way for an agent to make decisions inside an environment in order to maximise some reward value. Compared to the previous techniques of supervised learning which required a labelled input/output dataset, in reinforcement learning the focus is on finding the right balance between exploration (unknown environment) and exploitation (knowledge based on already explored environment).

Jae Won [20] believes that supervised learning, although powerful, is not sufficient for predicting stock price time series. Therefore, she experiments with reinforcement learning where she views the problem of predicting a company's share values as a Markov decision process (MDP). A Temporal Difference algorithm is implemented to calculate the future values of a given signal, while each state of the environment is defined as a "state vector" containing values like the opening/closing price of a share along with the daily volume and some technical indicators like MACD, volume oscillator etc. Finally, the reward of the TD model is defined as the daily return of the stock, calculated like below.

$$r_t = 100 \cdot \frac{y_c(t) - y_c(t-1)}{y_c(t-1)}$$

Figure 6 Daily return of the stock, as the reward of the TD model [20]

According to the conclusions of this research, the root mean squared error (RMS) between the predicted and the actual return tended to be higher at the lowest (1-4) and the highest grade range (13-16), meaning that the algorithm returned more promising results when applied to a day range of 5-9 or 9-12 days compared to 1-4/13-16 days.



Grade Range	RMS			
	$R(1)$	$R(5)$	$R(10)$	$R(20)$
1-4	3.84	3.07	3.80	4.65
5-12	3.14	2.62	2.81	3.46
13-16	4.11	3.38	3.49	4.99

Figure 7 Grade Range versus the Root Mean Squared Error of the TD model [20]

2.3.4 Neural Networks

Neural networks have been intensively researched regarding time series forecasting as one of their main benefits is calculating nonlinear models. More specifically, many researchers have argued that they can approximate the best functional form to describe any time series [21].

2.3.4.1 FFNs vs RRNs

A Feed-forward network (FFN) is an ANN, whose nodes do not form a cycle like in the case of a recurrent network. The simplest implementation of an FFN would be just a layer of output nodes, where the input is getting fed directly into. However, multi-layer FFNs also have one or more hidden layers of nodes, where each layer feeds the results to the next. They use common error functions like the MSE to calculate the prediction error in each case and usually use a backpropagation algorithm to learn from that error and adjust the weights for every input. Recurrent neural networks (RNNs) have similar architecture apart from the fact that they also have a context layer, which can get feedback from higher layers or from itself about previous activations of certain units. Therefore, RNNs can utilise their internal memory to process future inputs.

Kohara et al. [22] studied the effects of economic indicators and article headlines to stock price prediction by feeding the price of the Tokyo's stock exchange index, the USD/YEN exchange rate, the price of crude oil, a bond's interest rate and the average daily price of the Dow Jones index as inputs to both a FFN and a RNN.

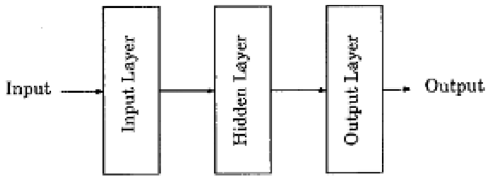


Figure 8 Feed Forward Network (FFN) [22]

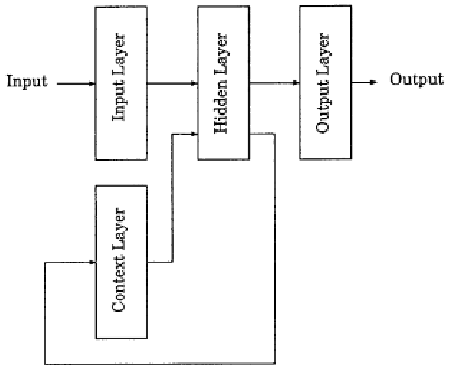
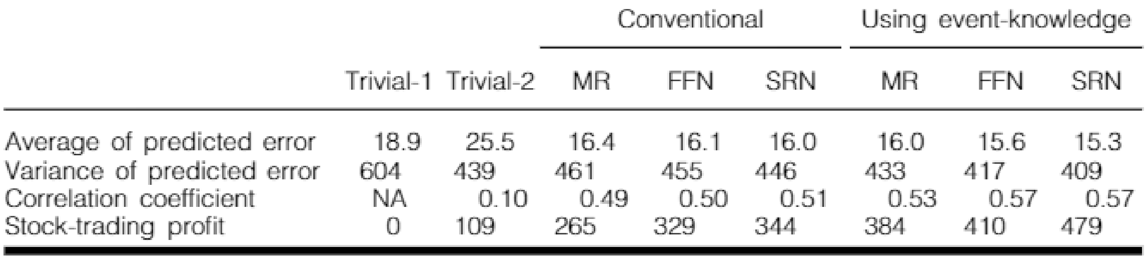


Figure 9 Recurrent Neural Network (RNN) [22]

They benchmarked the results of those networks against multiple regression analysis, arguing that their predictions outperformed by a 5% level of accuracy, while showing that neural networks are suitable for non-numerical input as well. Comparing between those two architectures of ANNs they found out that the simple recurrent network was slightly more profitable, as seen below.



			Conventional			Using event-knowledge		
	Trivial-1	Trivial-2	MR	FFN	SRN	MR	FFN	SRN
Average of predicted error	18.9	25.5	16.4	16.1	16.0	16.0	15.6	15.3
Variance of predicted error	604	439	461	455	446	433	417	409
Correlation coefficient	NA	0.10	0.49	0.50	0.51	0.53	0.57	0.57
Stock-trading profit	0	109	265	329	344	384	410	479

Figure 10 Error and profits generated by both the FFN and RNN models [22]

2.3.4.2 CNNs

Convolutional Neural Networks (CNNs) are also another type of ANNs that is being used especially for analysing high frequency time series. They are fully connected networks, where each neuron of each layer is connected to every neuron of the next layer. Although this architecture of multilayer perceptrons is mostly used in computer vision, Tsantekidis et al. [23] applied a CNN trained on high frequency limit order[1] (LOB) data to predict future price movements.

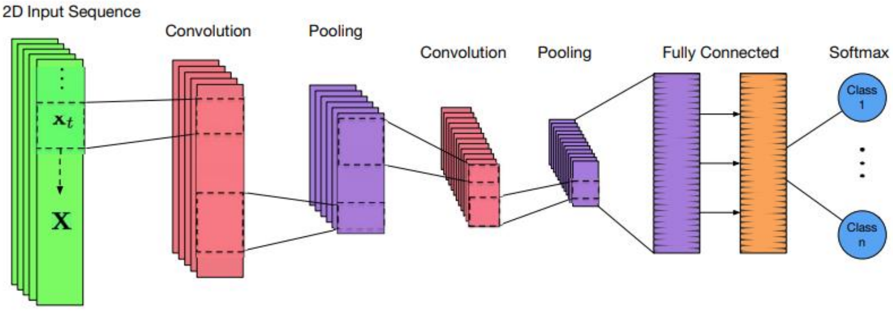


Figure 11 An example of a Convolutional Neural Network (CNN) [23]

They trained the model on over 4.5 million data points and according to their conclusions, CNNs proved more successful at predicting short term price movements compared to SVMs. [23]

2.3.4.3 LSTM

Long Short-Term Memory (LSTM) is an RNN model introduced by Hochreiter et al. [24] to reduce the time required to learn to store information over extended time intervals via recurrent backpropagation[2]. This ability of dealing with long term dependencies makes them suitable for long term stock price forecasting.

The architecture of a simple LSTM network consists of a *cell* and a *forget gate* apart from the usual input and output states. The cell consists of the long-term memory where all the long-term information is being stored. The forget gate is responsible for removing elements from the cell that are not improving the prediction's accuracy[25].

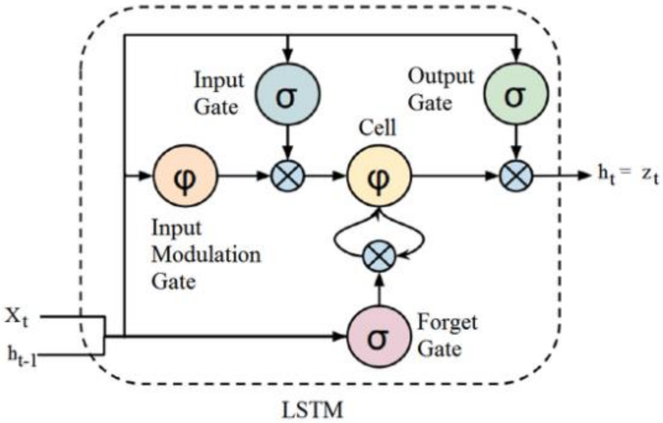


*Figure 12 - The architecture of an LSTM (https://miro.medium.com/max/542/1*ULozye1lfd-dS9RSwndZdw.png)*

[1] Limit order = a direction given to a broker to buy or sell a security at a specified price or better

[2] For a more intuitive explanation of the problem addressed by LSTMs visit https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Chun Yuan et al. [26] applied an LSTM model to predict the price of five Taiwan shares over the next five days given historical data of the stocks, using different technical indicators like the MACD[3] to evaluate them. They argued that the Relu activation function returned the most promising results with an MSE of just 1.9%.

2.3.4.4 GRU

Gated Recurrent Units (GRUs) describe a more recent type of RNNs discovered by Cho et al. [27] , which offer similar characteristics to LSTMs but have fewer parameters to adjust. They have two gates instead of three by combining the “forget” and “input” gates from above into a single “update” gate. The cell and hidden states are also merged into one, resulting in an overall simpler architecture, offering approximately the same results with an LSTM while being computationally more efficient, thus reducing the training time significantly.

More specifically, the activation h_t^j at time t depends on the activation h_{t-1}^j at time t-1 and the candidate activation $h_t^{\sim j}$:

$$h_t^j = \left(1 - z_t^j\right) h_{t-1}^j + z_t^j h_t^{\sim j}, \text{ where}$$

$$h_t^{\sim j} = tanh(Wx_t + \mathrm{U}[r_0 \circ h_{t-1}])^j, \text{ Hadamard product of } r_{0,} h_{t-1}$$

The update gate z_t^j controls the amount of content updated by the unit and is calculated like:

$$z_t^j = \sigma(W_z x_t + U_z h_{t-1})^j$$

Finally, the reset gate decides the amount of past information to be forgotten:

$$r_t^j = \sigma(W_r x_t + U_r h_{t-1})^j$$

According to research regarding the performance of GRU models on various fields, GRU networks outperformed traditional tanh and LSTM units for polyphonic music modelling [28], traffic flow analysis [29], automatic speech recognition [30] and many NLP tasks [31].

[3] MACD = Moving Average Convergence/Divergence

2.3.5 Pre-processing

In some cases, discovering correlations behind price movements requires filtering the input data in order to abstract away from short term noise and shift focus to long term dependencies. Quantitative finance has applied various filtering methods so far. [32]

2.3.5.1 Kalman filter

One of the most popular methods is Kalman filtering due to its high performance and simplicity. A Kalman filter uses noisy observations of a system over time to estimate its parameters and make future observations. It accepts as input:

1. A transition matrix, which is a mathematical model describing the changes in a system's state. For example, the movement of an object can be described with a kinematic equation whereas a more stable system can be modelled as a random walk
2. Covariance matrices of the transition noise and measurement noise
3. Estimate of the initial state and the estimate's error.

At every time step, the Kalman filter will use the transition matrix to estimate the current system's state x_t, take as input new measurements z_t and use the conditional probabilities of those measurements given the state to update the current state and the covariance matrix of the estimate P_t like seen below.

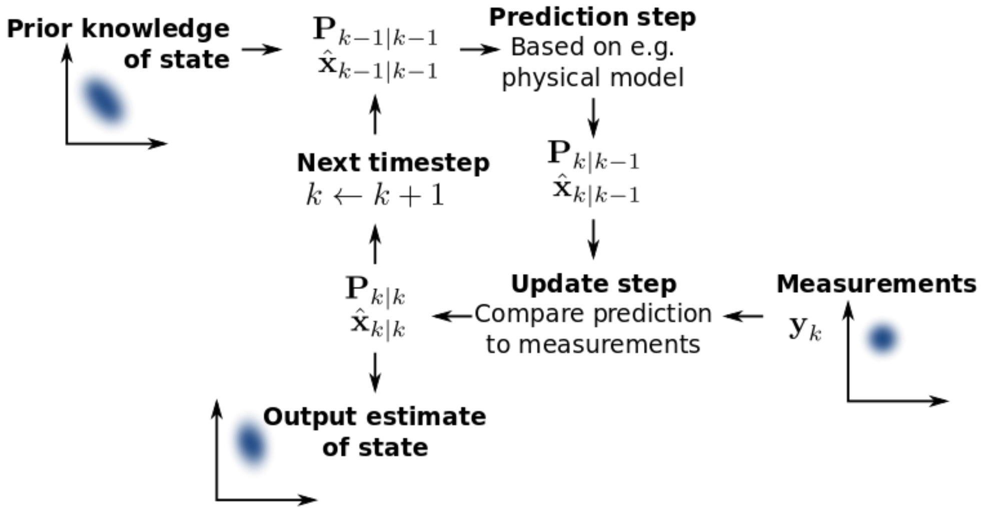


Figure 13 – Architecture of a Kalman filter (https://upload.wikimedia.org/wikipedia/commons/a/a5/Basic_concept_of_Kalman_filtering.svg)

A simple example [33] would be estimating the voltage value from a source, assuming that it has a constant value of **α** and 10 measurements are being observed above and below **α**. There are two equations:

- $x_k = x_{k-1} + w_k$ – the signal value is equal to linear combination of the previous signal plus some process noise – in this case the voltage value is constant so A=1 in $x_k = Ax_{k-1}$
- $z_k = x_k + v_k$ – the measurement value is a linear combination of the signal value plus some measurement noise.

Assuming the 10 observations are the following:

TIME	1	2	3	4	5	6	7	8	9	10
VALUE	0.39	0.50	0.48	0.29	0.25	0.32	0.34	0.48	0.41	0.45

For every time step, two sets of equations are considered.

Table 1 Kalman filter equations

Time update (prediction)	**Measurement update (correction)**
$\hat{x}_k^- = \hat{x}_{k-1}$, $P_k^- = P_{k-1} + Q$, *Figure 14 – Time update equation* [33]	$K_k = P_k^-(P_k^- + R)^{-1}$ $= \dfrac{P_k^-}{P_k^- + R}$, $\hat{x}_k = \hat{x}_k^- + K_k(z_k - \hat{x}_k^-)$, $P_k = (1 - K_k)P_k^-$. *Figure 15 – Measurement update equation* [33]

Table 2 Kalman filter example

k	**z**$_k$	$\hat{x}_{k-1}$	P_k^-	**Time update**	**Measurement update**	$\hat{x}_k$	P_k
1	0.390	0	1	$\hat{x}_k = \hat{x}_{k-1} = 0$ $P_k^- = P_{k-1} = 1$	K_k= 1 / (1 0.1) = 0.909 $\hat{x}_k$ = 0.909 . (0.390 - 0) = 0.35 P_k = (1 - 0.909) . 1 = 0.091	0.355	0.091
2	0.5	0.355	0.091	= 0.355 = 0.091	= 0.091 / (0.091 0.1) = 0.476 = 0.355 . 0.476 . (0.500 - 0.355) = 0.424 = (1 - 0.476) . 0.091 = 0.048	0.424	0.048
3	0.480	0.424	0.048			0.442	0.032

For the sake of simplicity, we only look at the first three iterations, but if we continued, we would see that the predictions would converge to the true voltage value.

2.3.5.2 Fourier Transform

The Fourier Transform is a transformation technique that converts a variable from one domain, usually time, to the frequency domain, by decomposing a function to its constituent frequencies.

$$\hat{f}(\xi) = \int_{-\infty}^{+\infty} f(x) e^{-2\pi i x \xi}\, dx\,, \xi \in \mathbb{R}$$

Equation 2 Fourier Transform

Using the Euler’s formula, it can filter any signal into a collection of circular paths, or complex sinusoids, as they are more formally known, and find the amplitude, frequency and phase angle of each component that needs to be described. Therefore, the set of all those different components describe the original signal.

$$e^{ix} = cos(x) + i\, sin(x)$$

Equation 3 Euler's formula

In Fourier's Analytical Theory of Heat [34], the inverse Fourier Transform was explained, according to which a signal can be generated from its Fourier coefficients.

$$f(x) = \int_{-\infty}^{+\infty} \hat{f}(\xi) e^{2\pi i x \xi}\, d\xi \,, x \in \mathbb{R}$$

Equation 4 Inverse Fourier Transform

The Fourier Transform has many variations which have been researched extensively in signal and image processing. Among many research papers, Todd. A. et al. [35] used hypercomplex numbers to define a Fourier Transform for colour images, Alan Marshall et al. [36] examined the underlying Fourier Transform ion cyclotron resonance mass spectrometry phenomena in idealized electromagnetic fields, while in the field of computational finance, Hurd and Zhou [37] introduced Fourier Transform analysis of the payoff function to forecast spread option pricing.

2.3.5.3 Wavelet Transform

Despite the advantages of the Fourier Transform, it is not suitable for analysing a non-stationary signal, because of the difficulty of interpreting the temporal information of the time-domain signal which is encoded in the frequency of the frequency domain signal. A solution to that would be the Short-time Fourier Transform (STFT) which divides the original non-stationary signal into stationary intervals of τ, like seen below.

$$X(t,f) = \int_{-\infty}^{+\infty} w(t-\tau) x(\tau) e^{-j^2 \pi f \tau}\, d\tau$$

Equation 5 Short-Time Fourier Transform

However, the problem of STFT is that it supports only fixed-sized windows, meaning that signals with both high and low frequency components are being poorly supported, since the window size may be too large for high frequency components or too small for low frequency ones.

The Wavelet Transform faces this issue by introducing variable sized windows, that can be reduced or expanded to capture both low and high frequency components.

$$X(\alpha, b) = \frac{1}{\sqrt{\alpha}} \int_{-\infty}^{+\infty} \psi\left(\frac{t-b}{a}\right) x(t)\, dt$$

Equation 6 Wavelet Transform formula

The function ψ is the orthonormal function used to define a Hilbert basis, where a is the scaling and b is the time. There are different types of Wavelets, with the most popular ones being Haar or Daubechies. There are many applications and research concerning wavelet transforms, ranging from image coding and compression [38] and analysis of peripheral blood flow oscillations [39] to modelling the future price of a stock market index [40].

Chapter 3 Analysis

This chapter focuses on analysing the requirements and constraints of the project, along with providing useful insight regarding some key strengths, opportunities and risks. The complete timeline of the whole project Is provided at the end, using the Gantt chart format.

3.1 Requirements

System requirements will be organized based on the MoSCoW prioritization [41], according to their level of significance (M=”Must”, S=”Should”, C=”Could”, W=”Won’t”). They will also get categorized as functional, non-functional and constraints, as seen below.

Functional Requirements

Table 3 Functional Requirements

ID	Description	Priority
F1	A recurrent neural network (e.g. LSTM) must be implemented and optimized for time series forecasting using keras/tensorflow	Must
F2	Different network architectures will be considered	Must
F3	The performance will be compared against a Kalman filter to reduce short term noise	Must
F4	The experiment will be expanded across 30 assets	Must
F5	Correlations between the history-target sizes and the performance will be examined	Should
F6	A T-test will be performed to analyse the statistical significance of the returns achieved the non-filtered against the filtered approach	Should
F7	GRU networks will be considered as well	Could
F8	More filtering methods will be considered (e.g. Fourier, Wavelets)	Could
F9	Backtesting can be performed to evaluate the profit returns of a simple strategy based on the model's predictions	Could
F10	Commissions will be applied to make the backtesting strategy more realistic	Could
F12	An automated trading strategy can be developed based on the results	Won't
F13	Sentiment analysis will be performed	Won't

Non-functional requirements

Table 4 Non-functional Requirements

ID	Description	Priority
NF1	The model should have a satisfying accuracy	Must
NF2	The model will not get affected by short term noise	Must
NF3	The model should predict both an increase and a fall in a share's price.	Should
NF4	The model will predict extreme price movements.	Could

Constraints

Table 5 Constraints

ID	Description	Reason
C1	Limited time will prevent the optimization of the LSTM's/GRU's structure	Finding the best model to fit the input data requires adjusting many parameters and trying different types of layers
C2	Data will be limited to 30 stocks and their daily historical prices	API calls have limits, however 30 shares from different industries should be enough to provide a proof of concept
C3	Not all variables can be considered	Stock prediction depends on more factors, like fundamentals, order book signals, sentiment analysis, regulatory environment but not everything can be taken into account

3.2 Risk Analysis

Below is a description of the risks identified for the project along with their severity and likelihood of happening.

Table 6 Risk Analysis

Risk	Description	Mitigation	Likelihood	Severity	Ranking
R1	Data loss	All data are backed up in cloud storage and every loss can be recovered	3	1	3
R2	Unsatisfying model results	Extending the scope with more data e.g. fundamentals/ using different model implementations	3	2	6
R3	Model overfitting	Collecting data from different industries and filter out short term noise	5	2	10
R4	Model develops long-side bias	Collecting full pricing history data to capture 2000's and 2008's stock market declines	5	3	15
R5	Unexpected economic developments deteriorating the results	The model might lag to factor in the results, but will be successful in the long term	4	5	10

3.3 SWOT Analysis

Below is a SWOT (Strengths – Weaknesses – Opportunities – Threats) analysis identifying our strong points along with areas of improvement.

Table 7 SWOT - Strengths

Strengths		
Experience in machine learning for quantitative analysis	Full daily pricing history provided	Data collected across many industries and well-established companies
Support by experienced individuals and academic papers	Experience with libraries for numerical processing and backtesting	

Table 8 SWOT - Weaknesses

Weaknesses		
Pricing history might be proved insufficient	Little experience regarding deep learning and neural networks	Global news and economic developments might cause the model to lag

Table 9 SWOT - Opportunities

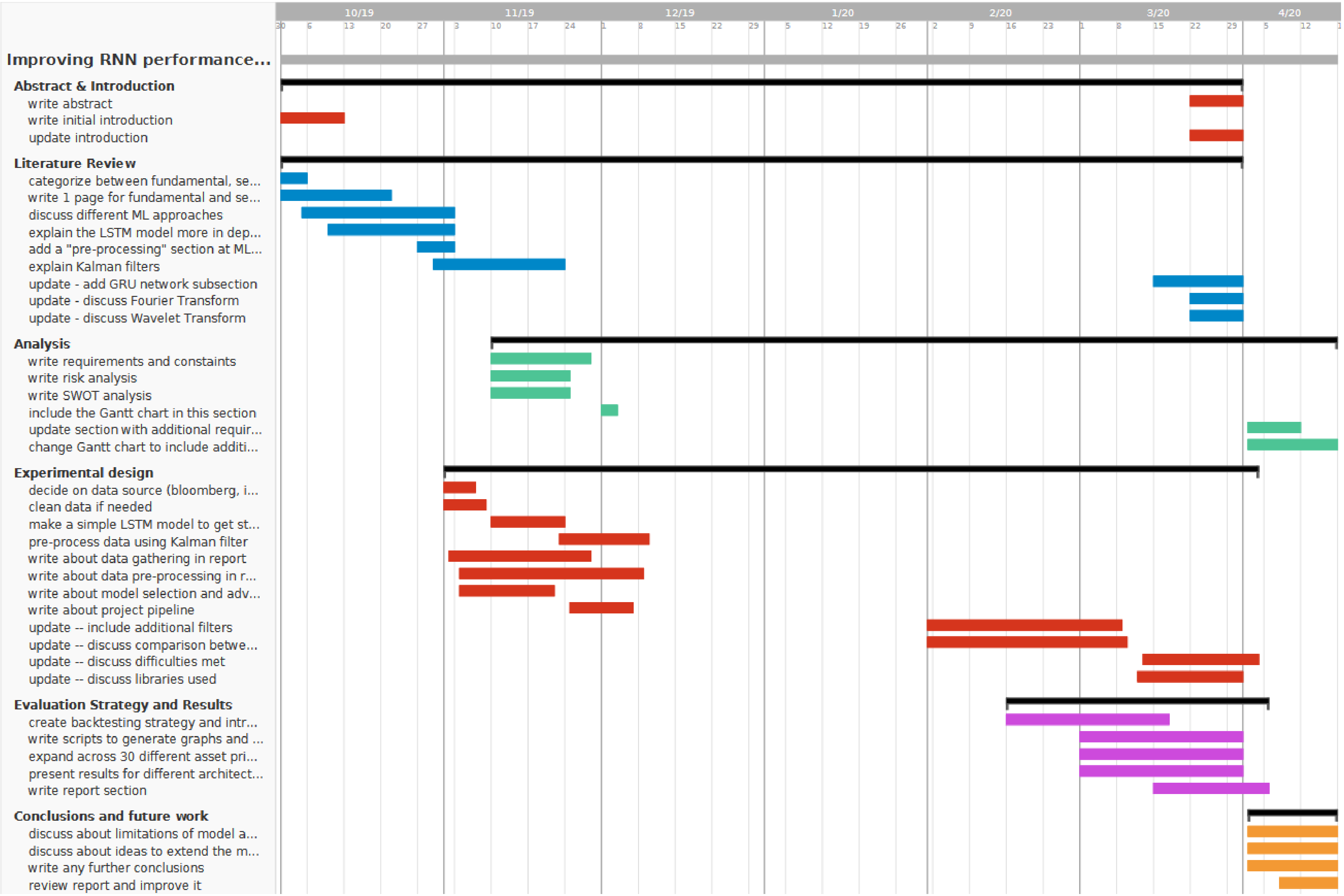
Opportunities		
LSTM/GRU seems promising for time series prediction	Application for other markets e.g. currencies, bonds, ETFs	Future development of a trading strategy based on the model's results
Recent developments regarding zero-commission trading		

Table 10 SWOT - Threats

Threats		
Exhaustive academic research regarding this topic	Competitive models have already achieved progress	Short term noise might affect the model's performance
Short time period might not be enough for achieving satisfying results	Zero commission trading will attract more competitors to this field	

3.4 Gantt chart

Below is a Gantt chart showing the schedule for both completed and future work.


10/19
11/19
12/19
1/20
2/20
3/20
4/20
30
6
13
20
27
3
10
17
24
1
8
15
22
29
5
12
19
26
2
9
16
23
1
8
15
22
29
5
12
1
Improving RNN performance...
Abstract & Introduction
write abstract
write initial introduction
update introduction
Literature Review
categorize between fundamental, se...
write 1 page for fundamental and se...
discuss different ML approaches
explain the LSTM model more in dep...
add a "pre-processing" section at ML...
explain Kalman filters
update - add GRU network subsection
update - discuss Fourier Transform
update - discuss Wavelet Transform
Analysis
write requirements and constaints
write risk analysis
write SWOT analysis
include the Gantt chart in this section
update section with additional requir...
change Gantt chart to include additi...
Experimental design
decide on data source (bloomberg, i...
clean data if needed
make a simple LSTM model to get st...
pre-process data using Kalman filter
write about data gathering in report
write about data pre-processing in r...
write about model selection and adv...
write about project pipeline
update -- include additional filters
update -- discuss comparison betwe...
update -- discuss difficulties met
update -- discuss libraries used
Evaluation Strategy and Results
create backtesting strategy and intr...
write scripts to generate graphs and ...
expand across 30 different asset pri...
present results for different architect...
write report section
Conclusions and future work
discuss about limitations of model a...
discuss about ideas to extend the m...
write any further conclusions
review report and improve it

Chapter 4 Experimental design

4.1 Data gathering

We have collected the full pricing history of 30 US companies' shares, operating in different industries, including MSFT, PG, XOM and JPM, in order to avoid overfitting and bias towards a specific sector. The data are in csv file format, like shown below.

timestamp	open	high	low	close	volume
28/10/2019	247.42	248.73	246.72	248.374	16218275
25/10/2019	243.16	246.73	242.88	246.58	18330500
24/10/2019	244.51	244.8	241.81	243.58	17318800

Table 11 List of shares considdered

TICKER	COMPANY NAME	TICKER	COMPANY NAME
AAPL	APPLE INC	**KO**	COCA COLA CO
AXP	AMER EXPRESS CO	**MCD**	MCDONALDS CORP
BA	BOEING CO	**MMM**	3M CO
CAT	CATERPILLAR INC	**MRK**	MERCK & CO INC
CSCO	CISCO SYSTEMS	**MSFT**	MICROSOFT CORP
CVX	CHEVRON CORP	**NKE**	NIKE INC-B
DD	DU PONT (EI) DE	**PFE**	PFIZER INC
DIS	DISNEY WALT	**PG**	PROCTER & GAMBL
GE	GENL ELECTRIC	**TRV**	TRAVELERS COS
GS	GOLDMAN SACHS	**UNH**	UNITEDHEALTH GP
HD	HOME DEPOT	**UTX**	UTD TECHS CORP
IBM	INTL BUS MACH	**V**	VISA INC-A
INTC	INTEL CORP	**VZ**	VERIZON COMM
JNJ	JOHNSON & JOHNS	**WMT**	WALMART INC
JPM	JPMORGAN CHASE	**XOM**	EXXON MOBIL CRP

A lot of thought has been given into choosing the best data source for this project. After a careful comparison of the available options, Alphavantage (https://www.alphavantage.co) was preferred in order to get the full pricing history of US stocks. As a future idea, this data can be combined with 5 years of fundamental data from Quandl (https://www.quandl.com), to introduce more variety at training the neural network.

Below is a comparison table of all the considered options along with their advantages and disadvantages, assuming a free subscription for Alphavantage and Quandl and a University of Southampton license for a Bloomberg terminal.

Table 12 Comparison of data vendors

	Alphavantage	Quandl	Quantopian	Bloomberg
Pricing data	Full pricing history of any stock	Limited to past 5 years for 30 stocks	Full pricing history of any stock	Full pricing history of any stock
Fundamental data	N/A	Limited to past 5 years for 30 stocks	Provided	Provided
Data format	Available in nice csv format	Available in nice csv format	Available directly as pandas DataFrames	Available as custom excel templates – would need pre-processing and cleaning before use
Other Limitations	N/A	N/A	Does not support tensorflow or keras.	N/A

Prioritizing the amount of pricing history and a clean data format, Alphavantage was our best option, since Quantopian's platform unfortunately does not support tensorflow/keras, which are the deep learning libraries we need to use for the development of our network.

4.2 Data pre-processing

As discussed during literature review, we are planning to experiment with Kalman filtering to smooth out our training data in order to avoid prediction errors due to short term noise. Daily stock prices can be very volatile however we are interested in identifying long term trends; therefore, applying our deep learning model on our filtered data is a promising method to reduce our loss function and achieve better validation results. Below is a 200-day graph of APPL stock compared to its equivalent Kalman filtered version. As you can see, the filter closely follows the underlying price movement without getting affected by momentary peaks and valleys.

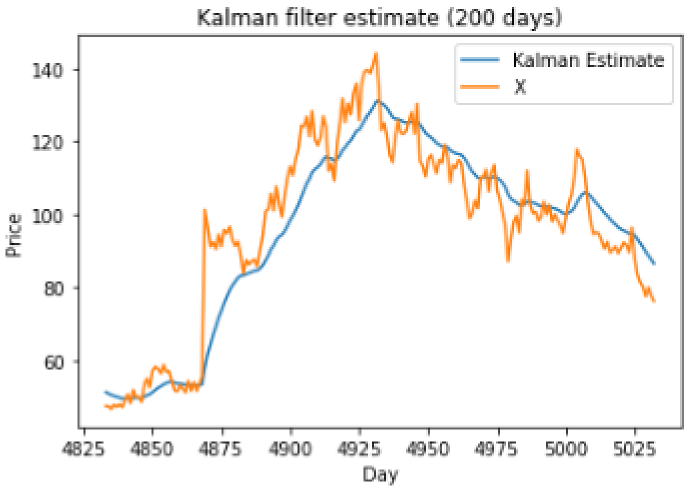


Figure 16 Kalman filter graph

If Kalman filtering achieves satisfying results, other data transformation methods will be examined, including Fourier and Wavelet transforms.

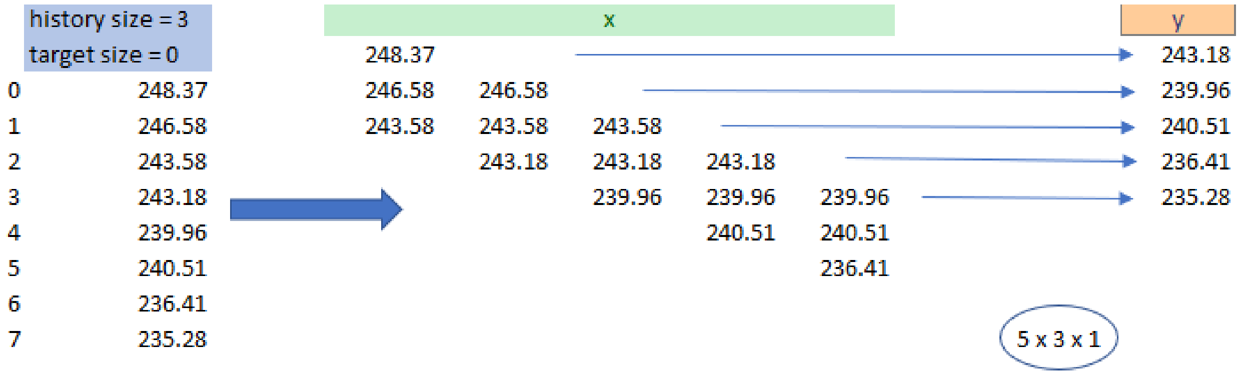
4.3 Model selection

This project is going to follow a quantitative approach by applying a gated RNN architecture in predicting future asset prices based on historical values. A *Long Short-Term Memory* (LSTM) model is a perfect fit for the scope of this experiment since:

- Its RNN structure allows it to utilise its internal memory for achieving better results.
- Its gated architecture allows the manipulation of the memory state, making it ideal for such regression problems.
- It is purpose-built for analysing long term dependencies, which is a problem normal RNNs do not successfully address.
- Small need for hyperparameter tuning such as input/output gate bias or learning rate. [24]
- The algorithm for updating each weight per time step has complexity of O(1), which is excellent considering the amount of data we are processing. [24]

However, GRU networks will also be considered since they offer a lot of the same benefits. They follow a similar but simpler gated architecture where instead of three gates (input, output and forget), they have two (reset and update gate), which generally reduces training time while also achieving similar performance compared to an LSTM unit.

Both architectures can predict a future time series value given a past window of information. Our data describe daily prices therefore a *history size* of 100 and *target size* of 5 would mean that we want to predict the price of the stock 5 days into the future given a window frame of the last 100 observations. Therefore, the model needs a threedimensional array as input describing all those possible combinations given the window and target size defined. The following picture will illustrate this clearly.



history size = 3 target size = 0	
0	248.37
1	246.58
2	243.58
3	243.18
4	239.96
5	240.51
6	236.41
7	235.28

x					y
248.37					243.18
246.58	246.58				239.96
243.58	243.58	243.58			240.51
	243.18	243.18	243.18		236.41
		239.96	239.96	239.96	235.28
			240.51	240.51	
				236.41	

Figure 17 Input/output data format for LSTM/GRU networks

For the sake of simplicity, the history size above is 3 and the target size is 0, so this 5x3x1 matrix is need as input in order to extract time series dependencies.

4.4 Project Pipeline

The project pipeline is shown below. After gathering and cleaning all the historical daily prices of the 30 different assets mentioned above, a filtering method is going to be applied and compared against the unfiltered results produced by an optimal configuration of an LSTM network for time series forecasting.

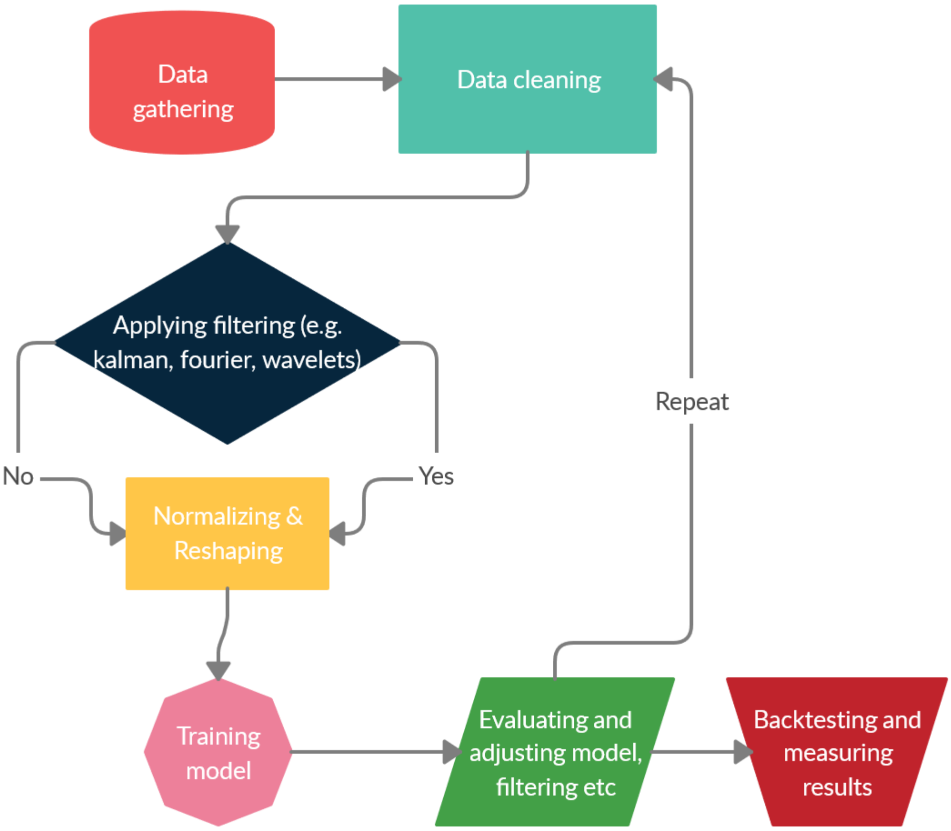

Data gathering
Data cleaning
Applying filtering (e.g. kalman, fourier, wavelets)
No
Yes
Repeat
Normalizing & Reshaping
Training model
Evaluating and adjusting model, filtering etc
Backtesting and measuring results

Figure 18 Project Pipeline

Chapter 5 Evaluation Strategy & Results

This section will evaluate all the steps taken to create and achieve an optimum configuration of a neural network model for financial time series analysis. The results presented in each subsection are based on the results of the previous subsections in a sequential manner.

Mean Squared Error (MSE) is selected as the loss function, which is formally defined using the following equation. It will ensure that the trained model does not have any outlier predictions with unexpectedly large errors, since it penalizes those data points more due to the squaring part of the function.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \widehat{y_i})^2$$

Equation 7 MSE equation

In the beginning, both an LSTM and a GRU architecture with various hidden layers and number of nodes are considered and assessed according to their validation loss and training time (in seconds). Then, the optimum training period is determined based on those results, along with a good combination of history and target size for future time series forecasting. Following that, different filtering signals are introduced as a pre-processing step to achieve a better accuracy and reduce the validation loss of the commonly used LSTM/GRU models. Finally, a back-testing strategy is constructed that will experiment with the practical application of those findings, by introducing the idea of profit to accurately compare the effectiveness of a simple gated architecture versus one combined with a filtering method. A T-test will be performed to determine if there is significant difference between the two results (unfiltered and filtered version).

Training time was a significant factor when choosing the various experimentation values below, since a nice balance is required between reducing the validation loss while keeping the training time to a satisfactory level.

5.1 Network Architecture

Both an LSTM and a GRU network were trained with different numbers of hidden layers and neurons per hidden layer to determine which one is the best configuration. Numbers in the range [1,10] were chosen for both the number of hidden layers (h) and the number of nodes per hidden layer (n) as they were considered appropriate in terms of both the dataset size and the training time required. For, example, for an LSTM network configuration of h=3, n=5, the following network architecture would be created.

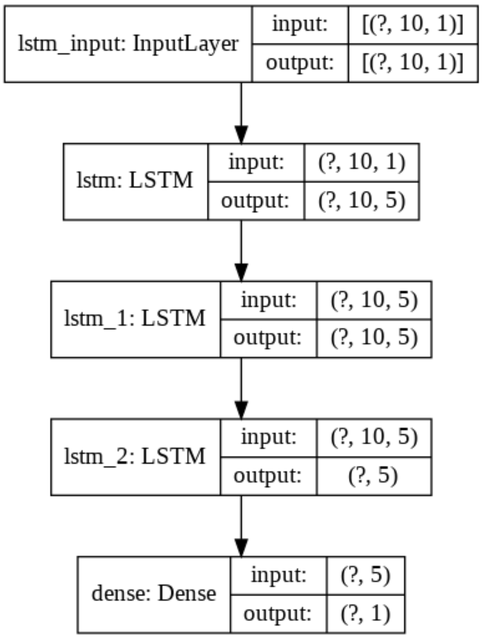


Figure 19 – LSTM architecture for (h=3, n=5)

The results below show that a configuration of (h=1, n=8) for a GRU architecture returns the minimum validation loss of 0.03739, within a satisfactory time limit of 93.3 seconds. However, the best values (h=1, n=3) for an LSTM network offer a validation loss of 0.03814; quite close to the optimum loss within only 65.9 seconds.

Table 13 Comparison of different architectures for LSTM/GRU

		LSTM		GRU	
hidden layers	nodes per layer	validation loss	training time (seconds)	validation loss	training time (seconds)
1	1	0.308834503	24.83551093	0.100520121	51.57224438
1	3	0.038148016	65.92009369	0.038324272	72.83415456
1	5	0.040174683	77.49370711	0.037965905	83.51088879
1	8	0.038517444	86.50880164	0.037394867	93.38708364
1	10	0.039581309	94.87026698	0.037489386	102.9761828
3	1	0.182476918	99.78945592	0.214383096	112.3895932
3	3	0.046751921	195.3144871	0.03924535	220.6500053
3	5	0.04783358	221.8021671	0.037711762	245.0554074
3	8	0.046419117	253.1148532	0.037465226	278.3107743
3	10	0.044369105	284.6826372	0.037552526	310.2061817
5	1	0.350702557	114.7518047	0.357979752	127.6174043
5	3	0.056923597	292.2250868	0.052684246	159.0075972
5	5	0.05157701	370.4334018	0.038256076	412.6588072
5	8	0.049661282	420.9229257	0.037717561	476.5845854
5	10	0.051303613	473.59055	0.037702145	515.0389204
8	1	0.075047786	419.1004646	0.08748922	477.5539804
8	3	0.064697625	520.4117569	0.044976149	598.4952081
8	5	0.066175957	529.7661476	0.041121779	652.7221331
8	8	0.075732698	291.7846827	0.037811243	751.664909
8	10	0.057310532	754.3566423	0.039749287	842.4404748
10	1	0.126164414	373.2446664	0.353920242	252.979563
10	3	0.068176577	577.1126912	0.043743286	747.6591855
10	5	0.06343007	734.7032618	0.03979856	843.6806108
10	8	0.065929453	876.585361	0.043783456	869.1385911
10	10	0.066760923	485.410351	0.041803398	1106.017163

According to Table 13, given the same configurations, GRU outperformed LSTM most of the time, apart from a few exceptions. However, that costed in terms of training time, which is always more for a GRU model. Another interesting observation is the fact that adding complexity to either one of the models, is not only increasing the training time as expected, but also the validation loss, which suggests that this task requires a smaller architecture with fewer number of hidden layers and neurons to perform well.

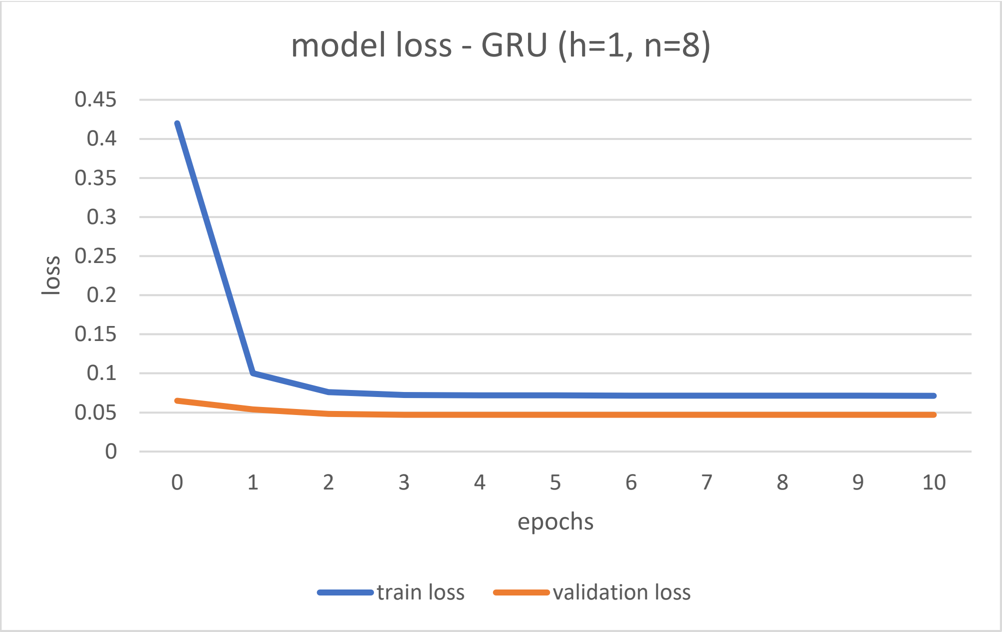


Figure 20 Train vs Validation Loss for GRU (h=1, n=8)

5.2 Training period

Another important hyper parameter that requires tuning is the number of epochs. Different values were tried and averaged across many runs for the optimal architecture found above of a GRU network with 1 hidden layer with 8 nodes. The final results are presented in the graph below in regard to the effect on validation loss and training time.

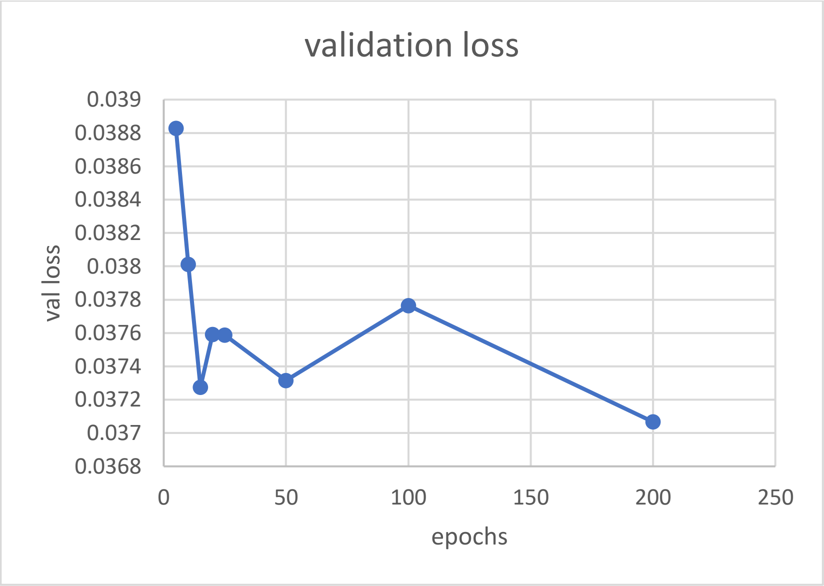


Figure 21 validation loss for epochs [0,200] GRU (h=1, n=8)

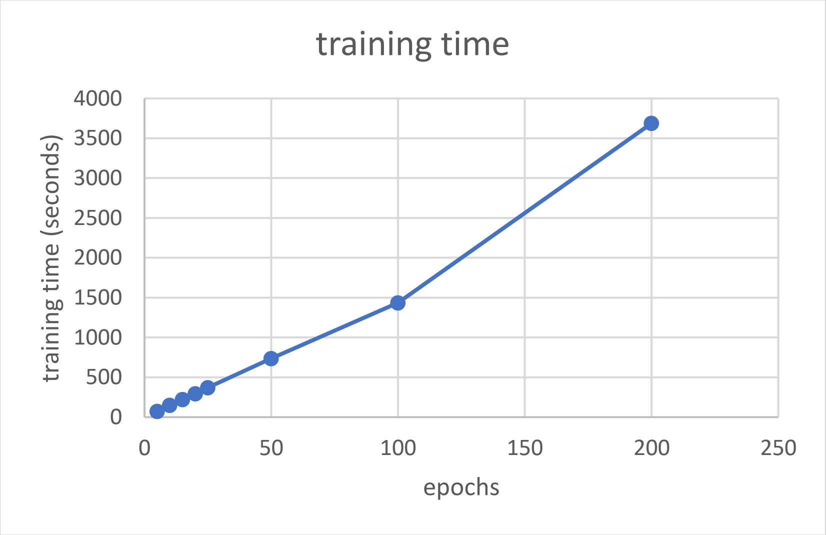


Figure 22 training time in seconds for epochs [0,200] GRU (h=1, n=8)

Although the validation loss does not follow a clear pattern after the first 20 epochs, it clearly converges below 0.0375 when the number of epochs becomes 15 and stays within the range 0.0370 - 0.0375 regardless of any more epochs added. Specifically, it even seems to increase temporarily for a value between 50 and 100, whereas 200 epochs do not seem to offer any particular improvement. However, the training time is increasing linearly as the number of training periods increase, making a choice of 15 epochs ideal for the task as a good balance between performance and time.

5.3 History and target size

An RNN used for time series forecasting requires historical data to learn from in order to predict future values. For the scope of this project, pricing data are collected daily in order to predict the future price of an asset *t* days into the future, if a rolling window of *h* historical data is known.

As expected, there are some differences in the accuracy of the model depending on the size of the historical period chosen as well as the future target size. If the history size is too small, the model fails to generalize and accurately predict future patterns. If it is too large, there is the risk of getting influenced a lot by older trends and fail to predict recent price movements, although that is one of the issues a gated architecture like a GRU/LSTM network promises to address.

Table 14 history and target size comparison for GRU (h=1, n=8, e=15)

History size	Target size	Validation loss	Training time	History size	Target size	Validation loss	Training time
3	1	0.014847	21.41083127	20	1	0.015033	31.38195002
3	2	0.017534	21.83473089	20	2	0.018132	45.81590911
3	5	0.025713	21.62576965	20	5	0.025053	46.22483324
3	10	0.036004	21.42670198	20	10	0.036117	45.03835174
3	25	0.061153	21.22386139	20	25	0.061618	46.37922469
3	50	0.081508	17.73633833	20	50	0.078967	32.9462721
5	1	0.016577	18.68513159	50	1	0.014791	97.93419925
5	2	0.018244	18.44492973	50	2	0.017485	97.41847865
5	5	0.025182	19.93363351	50	5	0.025364	63.80941383
5	10	0.036198	23.26300972	50	10	0.03552	99.25880983
5	25	0.060906	23.65716685	50	25	0.060429	96.1441115
5	50	0.078516	24.17702811	50	50	0.076567	86.19786126
10	1	0.014463	26.16797662	100	1	0.015666	170.6652388
10	2	0.018164	23.10039602	100	2	0.018282	193.9312609
10	5	0.025167	31.44622183	100	5	0.026388	188.2469731
10	10	0.035612	31.61951444	100	10	0.03833	145.5880652
10	25	0.062058	28.10374621	100	25	0.061649	193.1495406
10	50	0.078186	30.86968264	100	50	0.075949	198.5909909

From the above results, a history size of h=10 and a target size of t=1 results in the best validation loss of 0.014463 within only 26.16 seconds of training time on average. That means the model performs better when given 10 days of historical prices, tries to forecast the future price of the asset 2 days in the future (counting for t starts at 0). The training time in this case depends on the number of *<past window, future target>* partitions created to train the RNN.

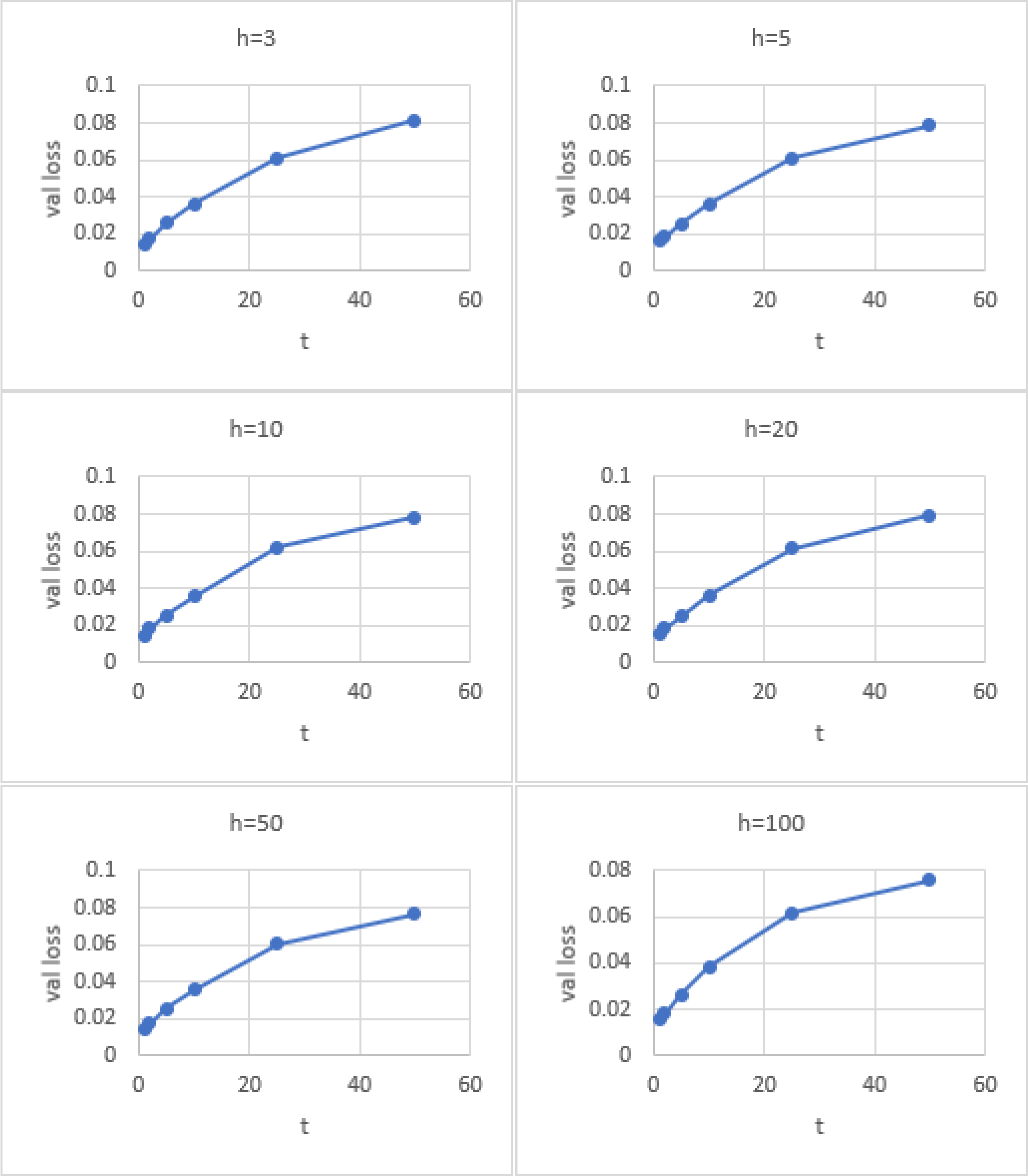


Figure 23 Comparison of validation loss increase In regards to history (h) and target (t) sizes for h,t ∈ [0,100]

According to the charts above, for every given history size, the validation loss is increasing at a logarithmic rate against the target size, meaning that although the model performs better at making short term predictions given a historical window of previous values, it is also suitable for longer term forecasting assuming some small accuracy decrease.

5.4 Filtering methods

There are many transformations commonly used in the fields of mathematics and physics for signal processing and time series analysis. Three of the most popular ones were described in the literature review section (Kalman filter, Fourier transform and Wavelet transform).

This project evaluates those transformations and their possible application in the field of quantitative finance, by using them as filtering methods for the dataset before training a gated neural network on it. The idea is that those filters will be able to smooth out the dataset by reducing short term noise; therefore, the model should be able to generalize better and discover longer term trends instead of getting affected by misleading price peaks and valleys.

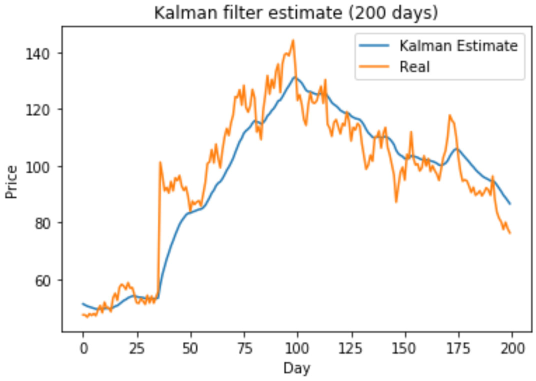


Figure 24 Kalman filter example for 200 days

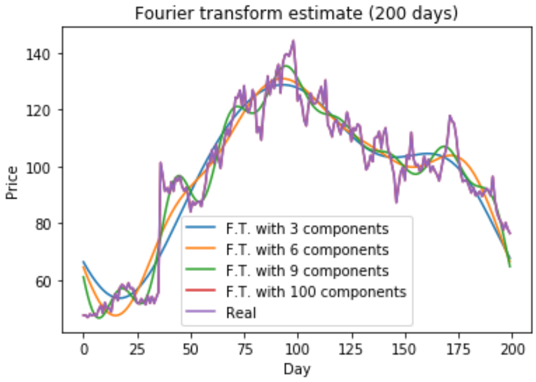


Figure 25 Fourier Transform example for 200 days

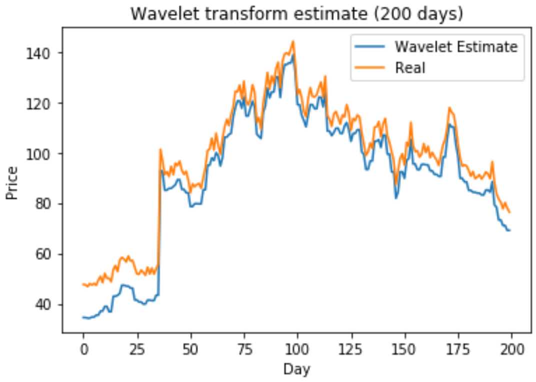


Figure 26 Wavelet transform example for 200 days

The results below record the validation loss and the training time (in seconds) of each filtering method along ten observations. The values stay within the same range across all runs, therefore no need for additional observations was necessary. The individual configurations of each filter such as the Kalman filter's state matrix or the number of components for the Fourier Transform were chosen after experimental observations about the values that result in the lowest validation loss for each method, while still generalizing well and avoiding overfitting. The model chosen here follows the best network architecture configuration as discussed in the previous subsections (GRU, hidden layers=1, nodes per layer=8, epochs=15, history size=10, target size=1).

Table 15 Comparison of different filtering methods

	Unfiltered		Wavelets		Fourier		Kalman	
Run	val loss	time	val loss	time	val loss	time	val loss	time
1	0.01496	22.833382	0.01511	9.16200383	0.002747	17.10673	0.004457	23.097168
2	0.01546	22.919901	0.015386	9.15196557	0.002867	17.09302	0.004218	23.266708
3	0.01545	22.356016	0.015587	9.15276689	0.002827	17.1376	0.004169	23.416081
4	0.01457	21.993748	0.015117	9.16052842	0.002604	17.14016	0.004317	23.335137
5	0.01473	22.474172	0.015325	9.15942008	0.002634	17.07381	0.004225	23.228566
6	0.01537	22.762209	0.015134	9.15104348	0.002863	17.06892	0.004131	23.280659
7	0.01513	22.050522	0.014762	9.15992984	0.002668	17.13301	0.004392	23.330843
8	0.01591	22.257689	0.015395	9.15642855	0.002899	17.07136	0.004169	23.122539
9	0.01594	22.363645	0.015602	9.15795456	0.002755	17.04805	0.004294	23.57536
10	0.01598	21.934891	0.01582	9.15099293	0.002901	17.03324	0.004109	23.809842

There are three important observations to be considered from the table above.

First, even though wavelets were proved to not be the best performing filtering method, since the corresponding validation loss fluctuates within the same range as the one produced by not filtering at all, the time required by applying wavelets is much less of only 9.15 seconds on average compared to 22.39 seconds of the initial (unfiltered) approach.

Kalman filtering was successful at reducing the validation loss of the initial approach by a significant 71.8022% from 0.01457 to 0.004109 with an average training time of 23.34 seconds, only 2 seconds more than unfiltered method.

Fourier Transforms were proven to be the most effective approach offering the overall best validation loss of 0.002604, a significant 82.1317% decrease from the best loss recorded without filtering, and a 36.6321% decrease from the best loss for Kalman filtering. The training time also got reduced significantly from 22.39 (unfiltered) and 23.34 (Kalman) seconds to only 17.09 on average for the Fourier Transform; 23.67% and 26.77% decrease respectively.

Below is a scatter plot comparing the results of the initial unfiltered approach versus the three transformations applied.

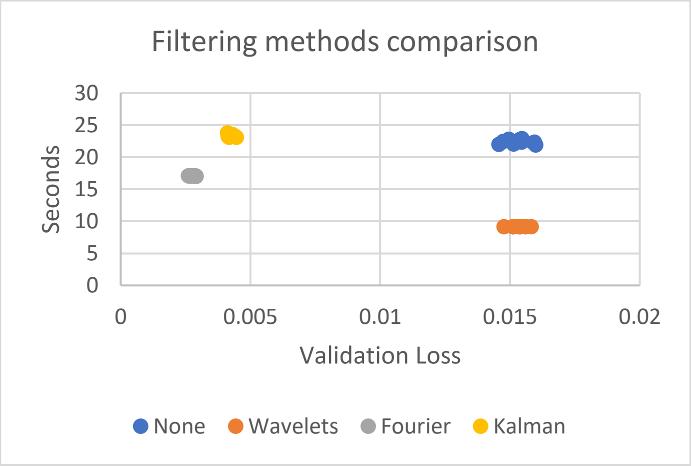


Figure 27 Scatter plot of different filtering methods

5.5 Back testing and profit

Quantitative analysts assess the viability of a trading strategy by evaluating its performance on historical data. That process is called backtesting and is necessary in order to determine whether a promising theoretical concept is successful if applied to a real-world scenario.

In the previous subsections, we discovered that a *Gated Recurrent Unit (GRU)* with *1 hidden layer*, *8 neurons per layer*, trained over *15 epochs* for a *history size of 10* and *target size of 2* days is the best configuration for our task. We then compared the validation loss of this network based on the initial dataset versus a filtered dataset using three different transformation methods, where the *Fourier Transform* was proved as the most successful one, as it reduced the validation loss the most.

For this section, a backtesting strategy was implemented and tested for both the initial (unfiltered) and the filtered approach, in order to determine which one, if any, can generate consistent profit over the testing sets of the 30 different assets collected.

Since the complexity of the trading strategy is not the main research focus of this project, a simple strategy was implemented according to which a buy order is executed whenever the network predicts the price is going to increase the next day; otherwise a sell order is executed, like shown in the pseudocode below.

```
testing_set;
network;
initial_balance;
final_balance;

for day in testing_set:
    current_price = network.predict(day) //predicted price for today
    next_price = network.predict(day+1) //predicted price for tomorrow
    if next_price > current_price:
        buy();
    else:
        sell();

print(final_balance);
```

The same trading strategy was tested across the historical prices of all 30 stocks, for both the initial datasets and their Fourier Transform filtered versions. The initial balance was set to 10,000$ and the testing datasets were produced according to an 80-20 split.

A lot research papers in the field of quantitative finance do not take commissions into account since they aim to provide an abstract proof of concept; however, we consider commissions as a vital characteristic of a proper backtesting strategy that aims to compare realistic profit results of different approaches. Therefore, the commission per trade was set to 4.95$, which was the industry standard before the recent trend of establishing 0$ commissions by most brokers.

Below are the results of the backtesting strategy for each different asset for both the unfiltered and the filtered approach. Filtering the datasets before applying the optimized GRU model from above, seems to return better results for the testing sets both in terms of validation loss and profit returns. In both tables, only three assets (highlighted in red) seemed to perform worse in terms of validation loss or profit return, though not necessarily the same ones. For example, MMM realized a worse validation loss by using Fourier Transform, however the profit return was much higher (from negative to positive) due to the fact that it generalized better and executed trades based on long term trends instead of short-term noise.

Table 16 Comparison of unfiltered vs Fourier Transform filtered returns for each asset

asset	unfiltered val loss	filtered val loss
AAPL	0.014531	0.002263
AXP	0.047751	0.028338
BA	3.009377	2.075655
CAT	0.179994	0.200934
CSCO	0.05836	0.020338
CVX	0.077055	0.008615
DD	0.09652	0.025775
DIS	0.087521	0.042765
GE	0.022393	0.023326
GS	0.094031	0.047743
HD	1.449762	1.115668
IBM	0.045837	0.00371
INTC	0.131704	0.013494
JNJ	0.511346	0.448553
JPM	0.988372	0.698516
KO	0.044234	0.005164
MCD	1.014139	0.606875
MMM	0.421255	0.614873
MRK	0.056051	0.00587
MSFT	0.201917	0.130694
NKE	0.064689	0.004217
PFE	0.053562	0.006288
PG	0.080656	0.046987
TRV	0.433814	0.330221
UNH	2.672781	1.737789
UTX	0.145629	0.055287
V	0.039079	0.008676
VZ	0.078704	0.011797
WMT	0.401553	0.272437
XOM	0.049515	0.003193

asset	unfiltered return	filtered return
AAPL	14.61338	-4.00029
AXP	-18.0218	43.10066
BA	49.73111	133.9159
CAT	-32.7697	18.11309
CSCO	-42.9635	-1.17678
CVX	-38.8526	13.29775
DD	-46.6599	16.19383
DIS	-48.0609	-67.0136
GE	-39.8444	-55.0973
GS	-22.6482	11.2975
HD	-38.5738	-24.0047
IBM	-45.3309	14.70211
INTC	-33.0048	19.47171
JNJ	-34.5412	25.14121
JPM	-30.0764	55.23641
KO	-41.9224	-14.9298
MCD	3.550801	57.54224
MMM	-35.4223	81.04591
MRK	-46.2009	30.26161
MSFT	15.53474	132.0099
NKE	-41.0533	-40.0778
PFE	-50.1319	-35.8404
PG	-25.5643	24.72001
TRV	-29.5682	17.85235
UNH	-9.36725	100.2018
UTX	-18.0935	-16.262
V	13.15395	50.83784
VZ	-43.9541	-31.4649
WMT	-18.1571	-3.45746
XOM	-50.5736	16.00458

The profit return distribution below makes clear that applying Fourier Transforms to the dataset before training a model on it, significantly improves the profit realized by a simple trading strategy, with 27/30 assets performing better and only 3/30 performing worse. There are also many cases where filtering helped in turning a loss into a profit. More specifically, in the unfiltered approach, only 5 out of 30 assets returned a positive profit, which is justified given the simplicity of the backtesting strategy and the 4.95$ commission per trade. However, filtering poses a substantial improvement where 19 out of 30 assets returned a positive profit; most having a quite big difference compared to their unfiltered counterpart.

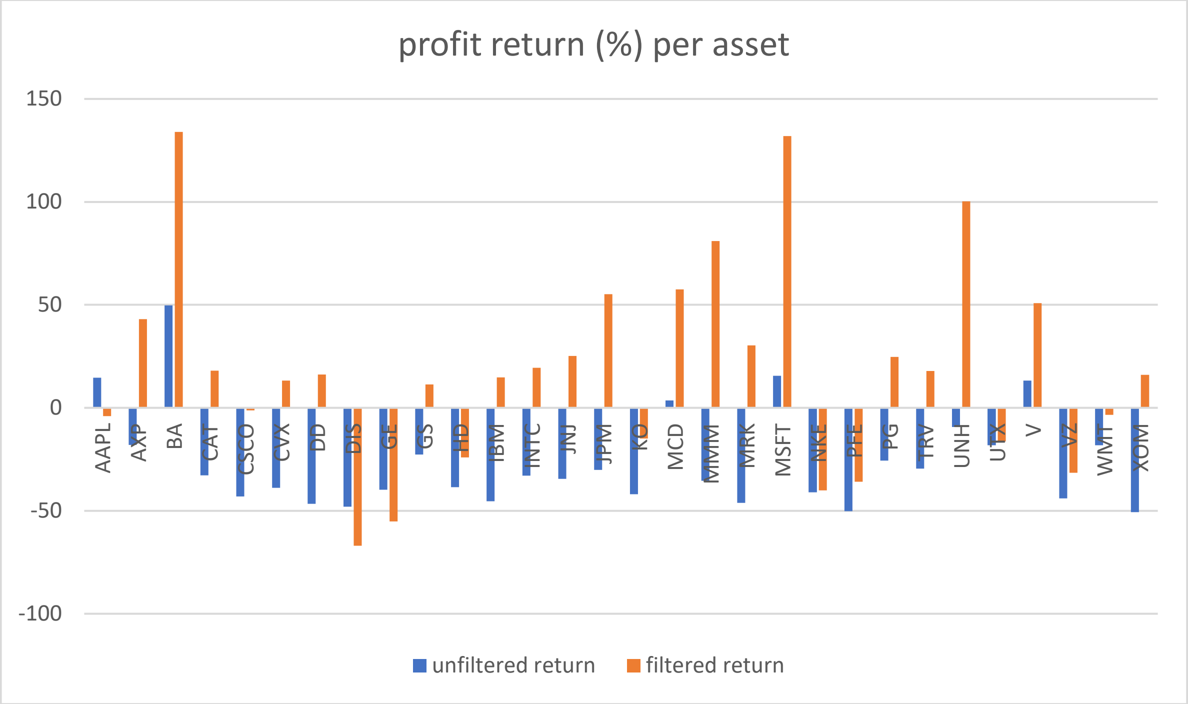


Figure 28 Graph of percentage return of each asset for both filtered and unfiltered methods

5.6 T-test

This section is going to apply a T-test as a type of inferential statistic to determine if there is a significant difference between the means of the two measurements from above. A T-test computes the t-statistic, the t-distribution values and the degrees of freedom to determine the statistical significance. The aim of a T-test is to reject or accept the null hypothesis, according to which the means of the two groups are equal.

Below are the results of the T-test performed over the measurements of the unfiltered against the filtered approach. One-tailed tests detect the differences only in one direction (unlike two-tailed tests), therefore the p-value of the two-tailed test is preferred for our results. Since the two-tailed p-value 4.56936E-05 is less than the standard significance level of 0.05, the null hypothesis can get rejected and it can be argued that there is significant statistical difference between the results of the two approaches.

Table 17 T-test results

	unfiltered	*filtered*
Mean	-26.15910273	18.92071683
Variance	578.059013	2370.949871
Observations	30	30
Hypothesized Mean Difference	0	
df	42	
t Stat	-4.546788533	
P(T<=t) one-tail	2.28468E-05	
t Critical one-tail	1.681952357	
P(T<=t) two-tail	4.56936E-05	
t Critical two-tail	2.018081703	

Chapter 6 Conclusions & future work

6.1 Comparison against goals

The initial goal of this project was researching an optimal configuration for an LSTM network that combines good results while avoiding overfitting for forecasting future asset prices of 30 different US listed share prices. Kalman filtering was meant to be implemented as a method to remove short term noise and extract the long-term pattern from a stock price, leading to better performance. This goal has been met and expanded, providing the following results:

- Research of optimal architectures and network configurations for both LSTM/GRU networks, along with a comparison of the two.
- Fourier Transform and Wavelet Transform explained and implemented as additional techniques to Kalman Filtering. Fourier proved to be the most promising method.
- Backtesting against both the unfiltered and filtered approaches to evaluate the profit returns.
- Perform T-test to assess the statistical significance between the unfiltered and filtered approach, concluding that filtering the initial pricing history with a simple Fourier Transform can have tremendous advantages for the vast majority of the 30 shares.

6.2 Challenges overcame

There were three key challenges that were faced and overcame to make this project feasible:

- The biggest challenge was the technical aspect of setting up the backtesting environment and simulating the profits and losses realized over a historical period according to the backtesting strategy. The strategy had to be simple, flexible and expandable enough to account for different datasets and commission rates, while returning accurate results.
- Another challenge was ensuring that the promising results returned from the various filtering methods used (fourier, wavelets, Kalman etc) were reliable and the adjusted models avoided overfitting. To overcome this, apart from comparing the training with the validation loss to verify their similarity, a variety of other methods was used, including expanding each model to 30 different assets to ensure that the average filtered returns were indeed better than the unfiltered version. Backtesting on the validation set was also used to ensure that reducing the loss resulted in better profits realized for the vast majority of stocks.

- The third main challenge was the time required to train hundreds of different neural networks of various configurations, architectures and filtering methods as well as running the backtesting strategy over many days for the unfiltered model against the fourier filtered one. The solution to this was developing some multithreaded python scripts and deploying them to the university servers as background processes that would run and save the results (validation loss, time in seconds, model architecture, profit/loss, commission rate etc) into csv files that would later get analysed and used for Chapter 5.

6.3 Limitations

However, this approach has some limitations when applicable in real world cases.

- Daily pricing history is not enough for forecasting future price values. Market makers usually utilise different methods, like analysing and extracting signals from the orderbook instead of looking at pricing values, which mostly reflect the results of those events. However, level1[4] and level2[5] data are very expensive to acquire and their events occur so frequently that would take lots of time to train a model on them.
- When aiming at forecasting the future value of a company, performing fundamental analysis and having a deep knowledge of the business model is crucial, which is a process that cannot be easily quantifiable unlike the machine learning approach.
- The inner workings of both the LSTM/GRU networks and the three filtering methods implemented function like a black box. Therefore, it is hard to reach a definite conclusion on the reason they do not seem to yield better results for a small portion out of the thirty shares. However, according to the T-test, filtering seems to provide a statistically significant difference across the thirty shares.

6.4 Future Work

In the requirements section there were some “Could’/”Won’t” ideas, that have not been fulfilled. Therefore, the following future work is proposed.

- Develop an automated trading strategy that utilizes the results of an optimal architecture for the GRU network, along with a Fourier transform for smoothing out the initial data.
- Increase the degrees of freedom, by collecting more data to train the model on. That can range from custom generated signals based on historical prices (e.g.

[4] Level1 data only show the top of the orderbook, meaning the best bid and best ask at every timestamp

[5] Level2 data show the whole depth of the orderbook, meaning all the orders placed in the orderbook at any given timestamp

rolling moving averages/Bollinger bands) to fundamental values like EPS, P/E, EBITDA etc).

- Expand the model to different asset classes as well, including options, futures and commodities.
- Instead of mid prices, adapt the model to use bid and ask prices, in order to a get a more accurate reflection on the effect of the spread into the final balance.

References

[1] L. Harris, “Trading and exchanges: Market microstructure for practitioners,” 2003.

[2] A. J. Menkveld, “High-Frequency Trading as Viewed through an Electron Microscope,” *Financ. Anal. J.*, vol. 74, no. 2, pp. 24–31, Apr. 2018.

[3] B. Biais, “High frequency trading,” 2011.

[4] M. S.- Rn and undefined 2011, “History of the efficient market hypothesis,” *cs.ucl.ac.uk*.

[5] M. J.-J. of financial economics and undefined 1978, “Some anomalous evidence regarding market efficiency,” *papers.ssrn.com*.

[6] I. K. Nti, A. F. Adekoya, and B. A. Weyori, “A systematic review of fundamental and technical analysis of stock market predictions,” *Artif. Intell. Rev.*, Aug. 2019.

[7] F. Steiger, “The Validity of Company Valuation Using Discounted Cash Flow Methods,” Mar. 2010.

[8] J. Kuhle, S. O.-J. of the A. of Business, and undefined 2010, “Teaching the Fundamental of Ben Graham and Warren Buffett.,” *search.ebscohost.com*.

[9] H. Saif, Y. He, and H. Alani, “Semantic Sentiment Analysis of Twitter,” 2012, pp. 508–524.

[10] V. N.-T. J. of Business and undefined 1971, “The analysis of world events and stock prices,” *JSTOR*.

[11] P. C. Tetlock, “Giving content to investor sentiment: The role of media in the stock market,” *J. Finance*, vol. 62, no. 3, pp. 1139–1168, Jun. 2007.

[12] A. Mittal and A. Goel, “Stock Prediction Using Twitter Sentiment Analysis.”

[13] X. Li, H. Xie, L. Chen, J. Wang, X. D.-K.-B. Systems, and undefined 2014, “News impact on stock price return via sentiment analysis,” *Elsevier*.

[14] L. Nunno, “Stock Market Price Prediction Using Linear and Polynomial Regression Models.”

[15] S. S. Roy, D. Mittal, A. Basu, and A. Abraham, “Stock market forecasting using LASSO linear regression model,” *Adv. Intell. Syst. Comput.*, vol. 334, pp. 371–381, 2015.

[16] S. R. Sain, “The Nature of Statistical Learning Theory,” *Technometrics*, vol. 38, no. 4, pp. 409–409, Nov. 1996.

[17] J. Principe, L. Gile, N. Morgan, and E. Wilson, “NEURAL NETWORKS FOR SIGNAL PROCESSING VII PROCEEDINGS OF THE 1997 IEEE SIGNAL PROCESSING SOCIETY WORKSHOP Seventh in a Series of Workshops Organized by the IEEE Signal Processing Society Neural Networks Technical Committee Edited by PTIC QUALITY INSPECTED 4,” 1997.

[18] F. E. H. Tay and L. Cao, “Application of support vector machines in ÿnancial time series forecasting,” 2001.

[19] K.-J. Kim, “Financial time series forecasting using support vector machines,” *Neurocomputing*, vol. 55, pp. 307–319, 2003.

[20] J. L.-I. 2001. 2001 I. I. S. on and undefined 2001, “Stock price prediction using reinforcement learning,” *ieeexplore.ieee.org*.

[21] G. Cybenko and G. Cybenkot, “Approximation by superpositions of a sigmoidal function. Math Cont Sig Syst (MCSS) 2:303-314 Mathematics of Control, Signals, and Systems Approximation by Superpositions of a Sigmoidal Function*,” *Math. Control Signals Syst.*, vol. 2, pp. 303–314, 1989.

[22] K. KOHARA, T. ISHIKAWA, Y. FUKUHARA, and Y. NAKAMURA, “Stock Price Prediction Using Prior Knowledge and Neural Networks,” *Int. J. Intell. Syst. Accounting, Financ. Manag.*, vol. 6, no. 1, pp. 11–22, Mar. 1997.

[23] A. Tsantekidis, N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis, “Forecasting Stock Prices from the Limit Order Book using Convolutional Neural Networks.”

[24] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[25] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, “TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS 1 LSTM: A Search Space Odyssey.”

[26] C. LAI, R. CHEN, and R. CARAKA, “PREDICTION AVERAGE STOCK PRICE MARKET USING LSTM,” *ir.lib.cyut.edu.tw*.

[27] T. Hornemann, S. Richard, M. F. Rütti, Y. Wei, and A. Von Eckardstein, “Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation Kyunghyun,” 2006.

[28] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical Evaluation of Gated

Recurrent Neural Networks on Sequence Modeling," Dec. 2014.

[29] R. Fu, Z. Zhang, L. L.-2016 31st Y. A. Annual, and undefined 2016, "Using LSTM and GRU neural network methods for traffic flow prediction," *ieeexplore.ieee.org*.

[30] S. Khandelwal, B. Lecouteux, L. Besacier, L. Besacier COMPARING GRU, and S. Khandelwal Benjamin Lecouteux Laurent Besacier, "COMPARING GRU AND LSTM FOR AUTOMATIC SPEECH RECOGNITION," 2016.

[31] W. Yin, K. Kann, M. Yu, and H. Schütze, "Comparative Study of CNN and RNN for Natural Language Processing," Feb. 2017.

[32] A. Javaheri, D. Lautier, and A. Galli, "Filtering in Finance," 2002.

[33] G. Welch and G. Bishop, "An Introduction to the Kalman Filter."

[34] "The Analytical Theory of Heat - Jean Baptiste Joseph baron Fourier - Google Books." [Online]. Available: https://books.google.gr/books?hl=en&lr=&id=3509AQAAMAAJ&oi=fnd&pg=PR1&dq=Fourier%27s+Analytical+Theory+of+Heat&ots=YzQ-bl155g&sig=Vpfs9uAlPcJxuakKQSf-JEP3gA4&redir_esc=y#v=onepage&q=Fourier's Analytical Theory of Heat&f=false.

[35] T. A. Ell and S. J. Sangwine, "Hypercomplex Fourier transforms of color images," *IEEE Trans. Image Process.*, vol. 16, no. 1, pp. 22–35, Feb. 2007.

[36] A. G. Marshall, C. L. Hendrickson, and G. S. Jackson, "Fourier transform ion cyclotron resonance mass spectrometry: A primer," *Mass Spectrom. Rev.*, vol. 17, no. 1, pp. 1–35, Jan. 1998.

[37] T. R. Hurd and Z. Zhou, "A fourier transform method for spread option pricing," *SIAM J. Financ. Math.*, vol. 1, no. 1, pp. 142–157, 2010.

[38] "Image coding using wavelet transform - Image Processing, IEEE Transactions on | Enhanced Reader.".

[39] "Application of the adaptive wavelet transform for analysis of blood flow oscillations in the human skin - IOPscience." [Online]. Available: https://iopscience.iop.org/article/10.1088/0031-9155/53/21/005/meta.

[40] Z. R. Struzik, "Wavelet methods in (financial) time-series processing," *Phys. A Stat. Mech. its Appl.*, vol. 296, no. 1–2, pp. 307–319, Jul. 2001.

[41] M. Vestola, "A Comparison of Nine Basic Techniques for Requirements Prioritization."

Appendix

Figures

Tables

Equations