MÄLARDALEN UNIVERSITY
SWEDEN

School of Education, Culture and Communication
Division of Applied Mathematics

MASTER THESIS IN MATHEMATICS/APPLIED MATHEMATICS

Forecasting the Stock Market - A Neural Network Approach

by

Magnus Andersson and Johan Palm

Magisterarbete i matematik/tillämpad matematik

MÄLARDALEN UNIVERSITY
UKK / TM
Box 883
SE-721 23 Västerås, Sweden

MÄLARDALEN UNIVERSITY
SWEDEN

School of Education, Culture and Communication
Division of Applied Mathematics

Master Thesis in Mathematics/Applied Mathematics

Date:
2009-03-04

Project name:
Forecasting the Stock Market - A Neural Network Approach

Authors:
Magnus Andersson and Johan Palm

Supervisor:
Prof. Kenneth Holmström

Examiner:
Prof. Kenneth Holmström

Comprising:
30 ECTS credits (hp)

Abstract

Forecasting the stock market is a complex task, partly because of the random walk behavior of the stock price series. The task is further complicated by the noise, outliers and missing values that are common in financial time series. Despite of this, the subject receives a fair amount of attention, which probably can be attributed to the potential rewards that follows from being able to forecast the stock market.

Since artificial neural networks are capable of exploiting non-linear relations in the data, they are suitable to use when forecasting the stock market. In addition to this, they are able to outperform the classic autoregressive linear models.

The objective of this thesis is to investigate if the stock market can be forecasted, using the so called error correction neural network. This is accomplished through the development of a method aimed at finding the optimum forecast model.

The results of this thesis indicates that the developed method can be applied successfully when forecasting the stock market. Of the five stocks that were forecasted in this thesis using forecast models based on the developed method, all generated positive returns. This suggests that the stock market can be forecasted using neural networks.

Acknowledgements

We would like to take this opportunity to thank Professor Kenneth Holmström for giving us the idea of this thesis subject and supplying us with stock data.

Furthermore, we would like to extend our thanks to Siemens AG, Corporate Technology Department, for allowing us to use the SENN application during this thesis.

March 2009, Eskilstuna, Sweden

Magnus Andersson
man04011@student.mdh.se

Johan Palm
johan@at-palm.se
jpm04001@student.mdh.se

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Forecasting the stock market is a very complicated task, it might even be impossible if the efficient market hypothesis is considered to be valid. The complexity of the problem can partly be attributed to the near random walk behavior of the stock price series. The problem is further complicated by noise, outliers and missing values, which are common in financial time series. Despite of this, the subject receives a fair amount of attention, which probably can be attributed to the potential rewards that follows from being able to forecast the stock market with some degree of accuracy.

Artificial neural networks are perfectly suitable to use as forecast models when predicting the stock market, especially since they, in most cases, are able to outperform the classical autoregressive linear models [GROT04, MCNE05]. One of the more commonly used network type in financial applications is the multi-layered perceptron (MLP) network, partly because of its ability to approximate any structure inside a data set [GROT04]. This universal approximation ability is a result of the MLP network trying to map input vectors to a corresponding output vector, giving it a pattern recognition approach to the problem of forecasting [GROT04]. There are however some drawbacks with using MLP networks for financial forecasting, one of them being the lack of included prior knowledge in the model. This means that the architectures of MLP networks have a very general structure, which is one reason to why it is so vulnerable to the problem of overfitting (a situation where the network also learns undesirable parts of the data).

This thesis focuses on the error correction neural network (ECNN), developed by Zimmermann et al. (2000) [ZIMM00]. The ECNN incorporates prior knowledge into the neural network forecast model, making it more resilient to the problem of overfitting. The prior knowledge consists of a view of the financial markets as dynamic systems, which transforms the forecasting problem into a system identification task. Thus the objective becomes to find the dynamic system that best explains the financial data that is to be predicted. The error correction neural network also uses the error of the previous predictions as additional input in order to help guide the model. Grothmann (2004, [GROT04]) found error correction neural network to be a promising solution to the problem of forecasting in financial markets. [GROT04, ZIMM00]

The aim of this thesis is to produce a method that can be used when developing forecast models, based on the error correction neural network, for the stock market. In order to achieve

this objective, an extensive literature study is performed, covering subjects such as properties of financial data, techniques to transform raw data into a more suitable format, evaluation methods, neural networks in general and the error correction network in specific. In addition to the literature study, different properties of the error correction neural network are examined through testing, using the SENN software package (a neural network simulation environment). The results from the literature study and investigation of the ECNN are then combined in order to derive a method that can be used when developing forecast models based on the ECNN.

Each chapter starts with a short introduction to the current subject and suggestions of further reading can be found throughout the report at appropriate places. This introductory chapter also states the problem formulation of the thesis and includes short summaries for each of the chapters.

1.1 Problem Formulation

Forecasting the stock market is a difficult task, even impossible if one believes the efficient market hypothesis (see Section 2.1). However, artificial neural networks have shown promising results and this thesis examines this further. Thus the main objective of this thesis is to investigate if the error correction neural network, with financial data as input, can be used to perform successful predictions in the stock market. In addition to this, a method is to be developed based on a literature and an empirical study, with the purpose of simplifying the design of forecast models for the stock market. The method shall accomplish this by splitting the forecast problem into a number of clearly defined issues that needs to be addressed, and then suggest solutions to these.

This is accomplished using the neural network simulation environment SENN (Siemens AG, see Section 6.2) when performing empirical tests. In addition to this, MATLAB (The MathWorks, Inc.) is used to perform necessary preprocessing and evaluation of the raw data and the network output. The financial data consists mainly of Swedish, but also some foreign, stocks, indexes, interest rates, etc.

Some issues that needs to be addressed when predicting the stock market, using neural networks, can be seen in the list below.

- Input Data: Selecting financial data to be used as input to the forecast model.
- Preprocessing: How to format the input data.
- Neural Network Architecture: Different architectural solutions, centered around the error correction neural network, to the forecasting problem.
- Learning Process: How to train the neural networks (i.e. optimization algorithms, error function etc.).
- Evaluation: Investigate the accuracy and reliability of performed predictions.

The following section contains a more detailed description of the thesis objective.

1.1.1 Thesis Objective

The objective of the thesis can be separated into three smaller tasks; a literature study, an empirical study and the development of a method based on these studies.

Literature Study

The first part of the objective is to perform an extensive literature study, covering subjects that are relevant when forecasting the stock market using the error correction neural network. These subjects can be categorized into three major groups; the selection and formatting of input data to the forecast model, selection and training of neural networks and evaluation of the performed forecasts.

Empirical Study

The second part of the objective is to perform an empirical study in order to investigate different aspects of neural network forecast models. The specific aspects to investigate shall be determined based on the knowledge gained during the literature study phase. At the very least, the potential of the error correction neural networks shall be examined when used as a forecast model. These tests shall be performed for stocks in at least two different industries.

Method Development

The third and final part of the thesis objective is to gather the knowledge gained during the literature and empirical study phases and summarize it into a method. The aim of this method shall be to simplify the development of suitable neural network forecast models based on the error correction neural network. This method shall then be tested and evaluated in order to ascertain its quality.

1.2 Chapter Summary

This section aims to give the reader a short outline of what subjects are covered in this thesis, by summarizing the different chapters.

The thesis can be separated into roughly three parts; a literature study (Chapter 2 through Chapter 5), an empirical study (Chapter 6) and a method part (Chapter 7). The main result of this thesis comprises of the developed method, while conclusions drawn from the tests can be found throughout the empirical study. Finally the main conclusions that can be drawn from this thesis are found in the conclusions chapter (Chapter 8).

Chapter 2: Financial Time Series

When predicting the stock market, financial time series serves as input and output to the artificial neural networks (in this thesis the error correction neural network). This chapter provides

relevant information regarding the properties of financial time series, the stock market and possible preprocessing of the data that can increase the forecasting ability of neural networks.

Thus issues like missing values, outliers, stationarity, input variable selection, range of the values in the time series and training, validation and generalization sets etc. are covered in this chapter.

Chapter 3: Neural Networks

This chapter provides an introduction to artificial neural networks in general and networks in the field of forecasting financial markets in specific. The aim of this chapter is to provide information needed for the following chapters, which covers the error correction neural network and the application of this network to the forecasting problem.

The presented information, concerning artificial neural networks, can be divided into three categories. First, basic information about the artificial neurons, the basic building block of neural networks. The second category covers common network architectures, e.g. feed-forward and recurrent networks. The third and last category concerns the learning process and covers different learning rules and techniques. In addition to these categories, a brief history of neural networks and different areas of applications are also provided.

In addition to the general information, this chapter focuses on time-delayed recurrent networks, on which the error correction network is based, the error correction learning rule and related issues (e.g. error back-propagation, vario-eta optimization, error functions, etc.). The problem of overfitting and different methods to avoid this issue are also discussed.

Chapter 4: Error Correction Neural Networks

This chapter describes the network of focus in this thesis, the error correction neural network. Information concerning finite unfolding in time and overshooting, two network modeling techniques used by the error correction networks, are also provided. Finally a brief description of some of the available extensions to the error correction neural network is provided.

Chapter 5: Evaluation

This chapter presents a number of methods that can be used when evaluating the performance of forecast models, using different performance measures and benchmarks.

The chapter can be divided into three categories, where one category discusses the subject of performance measures, which can be used to derive a measure of success for a forecast model. The second subject that is covered concerns benchmarks, which can be used to give the performance measures context. The third and final category discusses common mistakes that are made when developing and using forecast models.

This information is then used during the empirical study and the validation of the developed method in this thesis.

Chapter 6: Empirical Study

In this chapter, an empirical study is performed in order to study and try to find optimal configurations and training procedures when using the error correction neural network. Tests performed covers subjects such as determining the optimum number of neurons in the state clusters, amount of unfolding and overshooting, target multiplier etc.

During the tests, preprocessing and evaluation are performed using the developed 'Forecast Model Generation GUI', while the actual training and forecasting are performed in the SENN application. All of the performed tests are described in three sections; a method description, a results section and a conclusion section. In addition to this, brief descriptions of the 'Forecast Model Generation GUI', the SENN application and the characteristics of the financial data used are also included.

Chapter 7: Method to Select a Forecasting Model Based on ECNN

This chapter formulates a method that can be used to set up stock market forecast models based on the error correction neural network. The method is developed using the knowledge gained during the literature and empirical studies. In addition to this, the chapter also includes an evaluation of the method in order to determine its reliability.

Chapter 8: Conclusions

In this chapter, the final conclusions of the work as a whole are presented. In addition to this, suggestions of further work that can be performed, based on this thesis, are also discussed.

Appendix A: Formulas

In this appendix, a number of mathematical formulas that are used throughout this thesis, but not sufficiently defined, are briefly described (for a more thoroughly description, see some of the cited sources in connection to the formulas).

Appendix C: Glossary

This appendix contains a glossary over the more common and important technical concepts used in this thesis, for which a short description is provided.

Appendix B: Terminology

In this appendix a brief description of the terminology used in this thesis is provided.

Appendix D: Stock Selection

This appendix lists the different long and short lists of stock data that are supplied as input to the neural networks, used in the empirical study and the evaluation of the developed forecast method.

Appendix E through L: Results - Empirical Study

These appendixes contains tables with more complete results of the tests performed in the empirical study, than what is listed in the empirical study section.

Appendix M through N: Results - Validation of Method

These appendixes contains a more complete description of the results obtained during the evaluation of the developed method.

Chapter 2

Financial Time Series

A financial time series is a sequence of economical observations over time, e.g. daily stock prices, daily exchange rates, yearly profits etc. The task of predicting a time series is to try and estimate how it evolves in the future, what the future observations will be. [MAKR98]

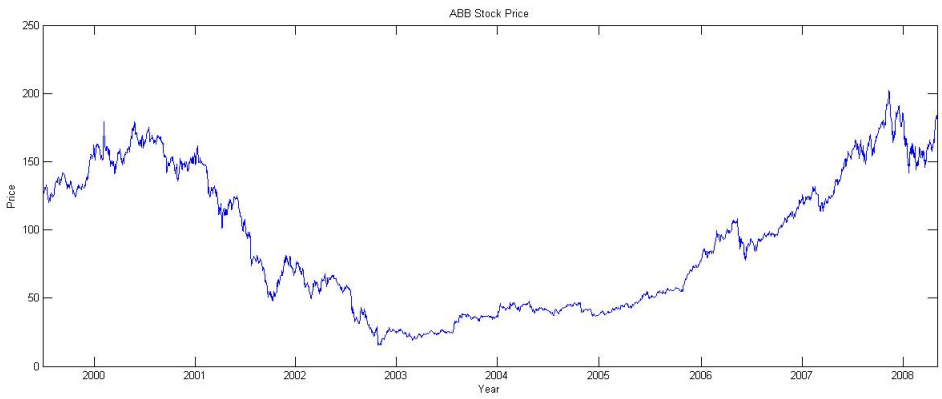

Figure 2.1: This figure shows an example of how a price time series could evolve, in this case the price of ABB stocks.

Predicting stocks is commonly viewed as a very complicated task, partly because of the near random walk process behavior of the stocks price series [HEL98a]. In addition to this, financial time series are often noisy, contains outliers, have missing values and are non-stationary [GROT04]. When using artificial neural networks in a forecast model, additional issues concerning the data arises, like the range (of each value) and size of the time series [MCNE05].

In order to increase the forecasting abilities of a model, these issues should be addressed. One way to do this is to include preprocessing of the data in the model. However, it is also important to perform the preprocessing carefully, since it otherwise might lead to the removal of useful information.

For a more comprehensive description of the stock market; see e.g. Hellström (1998) [HEL98a], and for time series; see e.g. Tsay (2005) [TSAY05] and Wei (1990) [WEIW90].

2.1 The Efficient Market Hypothesis

In short, the efficient market hypothesis (EMH) expresses that the current market price is the result of all available information and if new information becomes available, the market price quickly adjusts to reflect this change. There are three forms of the efficient market hypothesis which can be seen below. [HEL98a]

- Weak form: The weak form of EMH only considers the past prices.
- Semistrong form: The semistrong EMH considers all publicly available information (e.g. volume, sales forecasts etc.).
- Strong form: The strong form of EMH considers all data, including private information.

While the weak form of the efficient market hypothesis rules out any predictions based on past price information, the strong form rules out predictions altogether (of future prices) [HEL98a, HEL98b]. If it is assumed that the efficient market hypothesis is valid, this implies that the stock prices follows a random walk [HEL98a].

Even though the efficient market hypothesis is well supported there still exists some arguments against it. For example most of the market actors believes that they can predict the market well enough to make a profit. There also exists some research papers that suggests that non-linear methods (e.g. neural networks) can be applied to make successful predictions. The most common arguments against the EMH refers to a time delay, between an event happening and information of this event reaching the whole market, during which the market price does not reflect all available information. [HEL98c]

2.2 Data for Stock Prediction

There are different kinds of financial data available when predicting stocks; there are the actual stock data (also known as technical data) and the fundamental data. The fundamental data relates to the situation of the market and the condition of some company, while the stock data basically is time series with past stock information. These two categories are covered briefly in the following sections. [HEL98a]

2.2.1 Stock Data

For a stock there are different types of data available (for each trading day), which are listed below [HEL98a].

- Open: The opening price of the stock during a day, P_o.
- Close: The closing price of the stock during a day, P.
- High: The highest price of the stock during a day, P_h.

- Low: The lowest price of the stock during a day, P_l.
- Volume: The total number of stocks traded during a day, V.

These time series are usually non-stationary (see Section 2.3.2), which makes them undesirable to use in their raw form when forecasting [GROT04, HEL98a]. Instead information is derived from these series using preprocessing, usually on the closing price (e.g. asset return, see Section 2.4.1) [HEL98a].

During a normal week, Monday through Friday are trading days [HEL98a]. It is usually assumed that there are 252 trading days per year, 63 trading days per quarter and 21 trading days per month (in the USA) [TSAY05].

2.2.2 Fundamental Data

Fundamental data refers to information concerning a company's financial situation and activities. This information can be used to try and determine the 'true' value of the company's stock. To determine this there are usually three types of information that are of interest, which can be seen in the list below. [HEL98a]

- The general economy:

  Inflation, interest rates and trade balances etc., can be used as indicators of the general economy.

- The condition of the industry:

  Indexes, related commodity prices and competitors' stock values can be used as indicators of the condition of the industry.

- The condition of the company:

  A company's dept ratio, prognoses of future profits and sales, net profit margin etc., can be used as indicators of a company's condition.

Important to notice is that this information is not always available to the public [HEL98a]. If fundamental data (e.g. indexes, interest rates etc.) are used when forecasting, it might be necessary to preprocess it in order to transform it into a more suitable format.

2.2.3 Aggregating Data

Sometimes there is a need to aggregate the data, e.g. use daily data in order to get weekly or monthly data. This can be the case when making weekly or monthly forecasting.

One approach to aggregating daily data to a weekly time series is to use the closing value for a specific day. The high and low series can then be retrieved by finding the highest and the lowest values during the past week. The volume for the aggregated weekly time series can be extracted by taking the sum of the volume for each of the trading days during the past

week. This approach can of course be generalized and applied when aggregating to other time periods, e.g. monthly or yearly data from a daily time series.

Another approach to aggregating data is to specify the days of the week for which the data will be used. This is done through deciding a starting day and the total number of days (in sequence) to be used during the week. Since weekends are not trading days, daily stock data can be seen as having Monday as a starting day and then a total number of five days per week (i.e. Monday through Friday). There are no changes in the data, the desired days are simply inserted into the aggregated series, thus ignoring the other days.

2.3 Financial Time Series

Financial time series are often noisy, contains outliers, have missing values and are non-stationary [GROT04]. These properties will be discussed in the following sections.

2.3.1 Time Series and Patterns

It is important to understand what a pattern is and how it relates to time series. Suppose that there are three time series that will be used as input to a forecast model, as seen in Table 2.1. Each row in this table corresponds with a specific time, denoting when the values will be fed into the model, and each column with a specific time series. A pattern is the input supplied to a forecast model at some point, which corresponds with a row in the table. In this case the input pattern will consist of three values from the same row, one from each column, in the table. This can be stated more generally as that a pattern consists of one value from each input time series and where all values corresponds with the same input time.

		Time Series		
		Volvo B	DJ Industrial	OMX
Patterns	t-3	30.19	8364.29	511.37
	t-2	29.81	8538.46	510.58
	t-1	52.00	8528.15	516.45
	t	28.65	Missing	509.85
	t+1	28.36	8445.91	504.52
	t+2	28.65	8295.34	513.24
	t+3	Missing	Missing	Missing

Table 2.1: The columns represents different time series and each row is a point in time when the values (in the row) are used as input, thus the values in each row represents a pattern. There are some missing values and the entry at time $t-1$ for Volvo B could be interpreted to be an outlier.

2.3.2 Stationarity

Stationarity is an important property of time series, e.g. in the case of forecasting, where weak stationarity enables the ability to predict [TSAY05]. Non-stationary time series can vary greatly over larger time periods (see Figure 2.2) and contain inflationary trends [GROT04, HEL98a]. In general two types of stationarity are of interest, the strict stationarity and the weak stationarity (also known as wide sense stationarity or covariance stationarity) [TSAY05, WEIW90]. Proving that a time series fulfills the strong conditions of strict stationarity is hard, making the less constricting weak stationarity more commonly used [TSAY05, WEIW90].

Weak stationarity requires that the mean and variance of a time series are constant through time (i.e. time invariant), while the covariance and correlation only depends on the time difference (see Appendix A) [WEIW90]. A time series that is normally distributed and weakly stationary will also be strictly stationary [TSAY05, WEIW90]. A commonly used test for weak stationarity is the Dickey-Fuller test [GROT04], see e.g. McNelis (2005) [MCNE05].

Weak stationarity is a common assumption in time series analysis, unfortunately financial time series does not often fulfill this condition. However, a time series can be turned into a weakly stationary time series using differencing, see the next section. [GROT04]

Differencing

A non-stationary time series can be transformed into a weakly stationary time series by differencing it, as can be seen in Equation 2.1 [GROT04, TSAY05]. In some cases a time series needs to be differenced more than once (e.g. two-step differencing, see Equation 2.2) in order for it to become weakly stationary [GROT04]. Notice that the step length k (usually one) determines the size of the period (number of entries in the time series) that will be covered by the differencing.

$$\hat{y}_t = y_t - y_{t-k} \tag{2.1}$$

$$\tilde{y}_t = \hat{y}_t - \hat{y}_{t-k} = y_t - 2y_{t-k} + y_{t-2k} \tag{2.2}$$

For financial time series it is usually enough with a one-step differencing for it to become weakly stationary (see Figure 2.2). However, if the time series contains a non-linear trend, more than one differencing is necessary. [GROT04]

2.3.3 Outliers

Financial time series often contains outliers, which is a problem that should be addressed since they can have a negative effect on the performance of forecast models [GROT04]. An outlier is a value that differs a lot from the other values in the time series and can be the result of for example 'information shocks' (e.g. unexpected news) or 'unexpected shocks' (e.g. economic or political crises) [GROT04]. In the real world, the determination of what is an outlier is usually a very subjective decision [STRA01].

Since outliers can have a negative effect on forecasting models, steps should be taken to reduce their impact on the prediction performance [WEIW90]. This can be done both

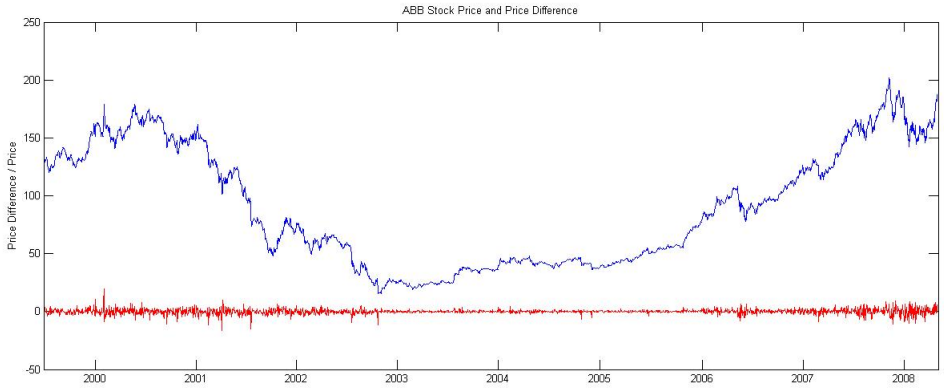

Figure 2.2: In this figure, the ABB stock price series (top one) is used to demonstrate the effect of a one-step differencing (bottom one).

through the use of preprocessing functions and the use of neural network models (e.g. choice of architecture, learning rule etc.) that are less sensitive to outliers [ENGE02]. Examples of techniques that are covered in this report, that reduces the effect of outliers, can be seen in the list below.

- Scaling function (preprocessing), see Section 2.5.2.
- log-return (preprocessing), see Section 2.4.1.
- Hyperbolic tangent activation function (architecture), see Section 3.1.1.
- The robust error function $\ln\cosh$ (learning), see Section 3.3.4.
- Cleaning with noise (learning), see Section 3.5.4.

2.3.4 Missing Values

Missing values in the data is a very common problem in real world applications and should never be ignored, since it can both reduce the performance of a prediction model and create unwanted bias [ENGE02, HEL98a]. One source of missing values is weekends and holidays, since these are usually not trading days. Another source can be technical or human errors that leads to unregistered values.

During the preprocessing of input data to a forecast model, missing values in the time series should be handled. There exists a number of different methods to do this, and three different approaches to this problem are listed below.

- Replace missing values:

  A simple way of handling missing values in a time series is to replace them with the last known previous value. Another choice is to replace the missing values with the average of the time series to which it belongs [ENGE02].

There are two important issues to consider when using an average value to replace an missing value. The first issue concerns the use of future data when calculating the mean, which is obviously wrong (i.e. off-by-one error, see Section 5.1). The other issue concerns non-stationary time series (see Section 2.3.2), since these series have a mean that is dependent on time and thus might vary for different parts of the series. Thus using the mean of a time series to replace missing values should be avoided when the series is non-stationary.

- Additional inputs:

  Another method to deal with this issue is to have a forecast model that accepts missing values [HEL98a]. When neural networks are used, this can be accomplished by supplying information of which input nodes currently have a missing value as input, using additional input nodes [ENGE02]. Through the use of this method, the impact of missing values on the performance can also be determined [ENGE02].

- Remove input patterns with missing values:

  A third approach for handling the missing value issue is to remove patterns that contains missing data completely [ENGE02]. This clearly handles the missing value problem, but may lead to information loss and to small data sets [ENGE02, HEL98a].

2.4 Derived Data

Derived data is used for a number of reasons, e.g. to get weakly stationary time series, better representation of the information and to emphasize the important information in the raw data. There are a number of ways that data can be derived from its raw time series (e.g. stock prices, volume etc.) and some of them will be covered in the following sections. Besides those covered below, there also exists other methods (e.g. technical indicators).

2.4.1 Asset Return

The asset return series is derived from the raw price series, which has some undesirable qualities (e.g. non-stationarity). This is one of the reasons for using asset return series instead of price series, since they are commonly assumed to be weakly stationary. Another reason for using the asset return is that it gives a complete and scale free description of the asset. [TSAY05]

In addition to this, asset returns are usually treated as continuous random variables, especially if it is low frequency data (e.g. stock and index returns) [TSAY05].

There exists a number of different types of asset returns, the simple return (common in the trading community) and the continuous compounded return (commonly used by academics) are covered in this section [HEL98a].

Simple Return

Simple return, also known as rate of return and momentum, includes a one-step differencing (see Section 2.3.2) and is derived according to Equation 2.3 [GROT04, TSAY05]. This is the most common version of the asset return in the trading community [HEL98a].

$$R_t = \frac{P_t - P_{t-1}}{P_{t-1}} \tag{2.3}$$

The k-period simple net return, see Equation 2.4, computes the net return over a period with the length k (where $k = 1$ gives the simple return) [TSAY05].

$$R_t^k[k] = \frac{P_t - P_{t-k}}{P_{t-k}} \tag{2.4}$$

To calculate the gross return (multiperiod simple return) for holding the asset during a period $t-k,...,t$, Equation 2.5 can be used [TSAY05].

$$R_t^G[k] = \prod_{i=0}^{k-1}(1+R_{t-i}) - 1 \tag{2.5}$$

Continuously Compounded Return

The continuously compounded return, also known as the log-return, is derived according to Equation 2.6 [TSAY05]. An advantage of the log-return compared to the simple return is that it is better at handling outliers in the data, thus reducing their impact on a forecasting model [HEL98a].

$$R_t^{log} = \log\left(\frac{P_t}{P_{t-1}}\right) \tag{2.6}$$

The gross return, when using the continuously compounded return, for holding an asset during a period $t-k,...,t$ can be computed using Equation 2.7 [TSAY05].

$$R_t^{G-log}[k] = \sum_{i=0}^{k-1} R_t^{log} \tag{2.7}$$

Equation 2.8 shows the relationship between the simple return and the continuously compounded return [TSAY05].

$$R_t = 100\left(e^{R_t^{log}/100} - 1\right) \tag{2.8}$$

2.4.2 Volume: Rate of Change and Gaussian Volume

The volume might be a good source of information since an increase in trading is an indicator of new information reaching the market [HEL98a]. The raw volume data can be transformed into the volumes rate of change using Equation 2.9.

$$V_t^R = \frac{V_t - V_{t-1}}{V_{t-1}} \tag{2.9}$$

The volume data can also be transformed into a Gaussian volume, using the scaling function (see Section 2.5.2) with a sliding window technique, which should reduce non-stationarity in the volume data [HEL98a, NYGR04]. Thus the mean μ_t^n and standard deviation σ_t^n are calculated using a sliding window with the size n (a common window size is 30), for more information see Appendix A. To transform a volume into a Gaussian volume, Equation 2.10 can be used [HEL98a].

$$V_t^G = \frac{V_t - \mu_t^n}{\sigma_t^n} \tag{2.10}$$

2.4.3 Volatility

Volatility describes the variability of a price series, i.e. how much it moves around its mean, and can be used when estimating the risk (or profit opportunity) of investing in an asset [HEL98a, CORN07]. This since the predictability of an asset and its volatility are connected [HEL98a].

Although there exists several definitions of volatility, the standard definition can be seen in Equation 2.11, where μ is the mean value and R_t^{log} is the continuous compounded return. In its standard definition, the volatility is the same as the standard deviation (see Appendix A.4) of the log return series [HEL98a].

$$\sigma_v = \sqrt{\frac{1}{T-1}\sum_{t=1}^{T}\left(R_t^{log} - \mu\right)^2} \tag{2.11}$$

Volatility can be used both as input to and output from a prediction model (i.e. volatility can be predicted). Usually a sliding window technique is used when determining volatility that will be used as input to a forecast model. [HEL98a].

2.4.4 Trends

The k-step simple net return (see Equation 2.4) can be seen as a trend for a price series, which is an important element in times series and can be either linear or non-linear [HEL98a, WONN90]. Hellström (1998) suggests dividing the k-step simple net return with its step length in order to enable easy comparison between trends with different step lengths, see Equation 2.12 [HEL98a].

$$T_t^k[k] = \frac{R_t^k[k]}{k} \tag{2.12}$$

2.4.5 Turning Points

In stock price series, turning points can indicate that supply and demand have reached an equilibrium. Thus these positions can be seen as more important than the information in between the turning points. [HEL98a]

The force may be used to characterize the turning points of a time series. In Equation 2.13 a generalized k-step version of the force transformation equation can be seen (where k refers to the period over which the force is calculated). [GROT04]

The force transformation includes a two-step differencing, which usually is enough to transform a financial time series into a weakly stationary series (see Section 2.3.2).

$$F_t^k = \frac{y_t - 2y_{t-k} + y_{t-2k}}{y_{t-k}} \tag{2.13}$$

2.5 Scaling

Scaling can be used to reduce the range of the values in a time series [HEL98a]. When using neural networks, scaling of the input variables to the active domain of the activation function (see Section 3.1.1) can improve performance greatly [ENGE02, MCNE05].

Using the hyperbolic tangent activation function as an example, input values that have exceeded the activation function's saturation levels will simply generate an output with a value close to 1 or -1. Thus, using values that lies outside the active domain of the activation function leads to a loss of information that is undesirable. Reasonable ranges when scaling are $[0,1]$ for the logistic activation functions and $[-1,1]$ for the hyperbolic tangent activation functions. Also note that the time series used as a target (i.e. desired output, see Section 3.3.2) needs to be scaled to the range of the output neurons' activation function. [MCNE05]

2.5.1 Linear Scaling

Linear scaling makes use of the minimum and maximum values of the time series that is to be scaled. Equation 2.14 scales the time series to the range $[0,1]$, while Equation 2.15 scales it to the range $[-1,1]$ [MCNE05].

$$\hat{y}_t = \frac{y_t - \min(y_t)}{\max(y_t) - \min(y_t)} \tag{2.14}$$

$$\hat{y}_t = 2\left[\frac{y_t - \min(y_t)}{\max(y_t) - \min(y_t)}\right] - 1 \tag{2.15}$$

2.5.2 Mean and Variance Scaling

Mean scaling (also known as mean centering, see Equation 2.16), calculates the mean of a time series and then subtracts it from each term, which is suitable for data without bias. Variance scaling (see Equation 2.17) on the other hand is more suitable when several time series with

different units (e.g. price, volume etc.) are used as input to a model. For information regarding the mean and standard deviation, see Appendix A. [ENGE02]

$$\hat{y}_t = y_t - \mu_y \tag{2.16}$$

$$\hat{y}_t = \frac{y_t}{\sigma_y} \tag{2.17}$$

Both mean and variance scaling can be used simultaneously, which is referred to as the 'scaling function', see Equation 2.18 [ENGE02, HEL98a]. Grothmann (2004) states "that a scaling of the data fits best to the numerics of hyperbolic tangent squashing functions" [GROT04, p. 27]. Thus it might be a good idea to use the scaling function on data that will serve as input to a neural network that uses the hyperbolic tangent activation function [GROT04]. In addition to this, the scaling function can be used to reduce the impact of outliers (see Section 2.3.3) on the forecast model [TIET08, ENGE02].

$$\hat{y}_t = \frac{y_t - \mu_y}{\sigma_y} \tag{2.18}$$

All three of these scaling functions can be used when the minimum and maximum values (i.e. the range) of the data are unknown [ENGE02]. Also note that, when calculating the mean and standard deviation, this may only be done using data in the training and validation sets, never on the data in the generalization set (see Section 2.8).

The scaling function can also be used to reduce non-stationarity (see Section 2.3.2) in a time series, through the use of a sliding window technique [HEL98a]. This means that the mean and standard deviation used in the scaling function are calculated using the past n values (i.e. $t-n,...,t$) [HEL98a]. As long as the mean and standard deviation only are calculated on past values, it can be applied to the whole data set, including the generalization set.

2.6 Scaled Momentum and Force

Grothmann (2004) proposes the use of two rather simple transformations that are able to describe the underlying dynamics of the input time series, scaled momentum and force (see Equations 2.19 and 2.20) [GROT04].

$$u_t = \text{scale}\left(\frac{y_t - y_{t-k}}{y_{t-k}}\right) \tag{2.19}$$

$$u_t = \text{scale}\left(\frac{y_t - 2y_{t-k} + y_{t-2k}}{y_{t-k}}\right) \tag{2.20}$$

The momentum extracts information concerning the rate of change in a raw time series (e.g. simple return, see Section 2.4.1). However, a drawback with only using the momentum is that it leads to trend following forecast models. In order to rectify this, the force transformation is also used, which gives information concerning the turning points (see Section 2.4.5) in the

raw time series. Both of these transformations includes differencing, which usually leads to weak stationarity when used on financial time series (see Section 2.3.2). [GROT04]

The scaling of the momentum and force (using the scaling function, see Section 2.5.2), leads to transformed time series that better fits the numerics of the hyperbolic tangent function [GROT04]. In addition to this, scaling also reduces the effect that outliers in the data have on a forecast model. [TIET08, ENGE02]

2.7 Dimensional Reduction

When predicting financial time series, there are often a large number of time series available that can be used as input to a forecast model [MCNE05]. However, when using neural networks to forecast, the use of a large set of input variables, instead of a small set, will not necessarily lead to a higher forecast performance [PISS02]. This is the result of a phenomenon known as the 'curse of dimensionality', which refers to that the size of the training set needed to train a model to a certain level of accuracy increases exponentially with the addition of new input variables [MCNE05, PISS02].

In addition to this, two input variables that are highly correlated supplies roughly the same information to a forecast model [REED99]. This means that adding a variable to a model, which is highly correlated with an already used input variable, will add no or little new useful information and thus not improve the forecast performance. As discussed in Section 2.3), financial time series are often noisy and contains outliers etc., which can have a negative impact on forecast models. Thus it stands to reason that the addition of a variable that is highly correlated with an input variable might even reduce the performance of a forecast model. Highly correlated input variables might also disturb the training of a neural network [TIET08].

Dimensional reduction of the input variables to a forecast model can be performed in several different ways, three of which is covered briefly below.

- Aggregating several time series:

  One approach that reduces the input dimensionality of a model is to aggregate the data from several different time series into one time series [HEL98a].

- Carefully selection of input variables:

  By being careful when selecting input variables to a forecast model, the dimensionality of the input can be kept low. This can be accomplished through the use of expert knowledge, selection criterions etc., and will be discussed further in Section 2.7.1.

- Principle Component Analysis (PCA):

  Principle component analysis can be used to reduce a high dimensional input data set into a low dimensional output data set [ENGE02, MCNE05]. PCA accomplishes this by trying to find a small set of principle components (linear combinations of the input data) that explains a large portion of the variation in the input data set [MCNE05]. This smaller set of principle components can then be used as input to a forecast model.

When forecasting time series, PCA can be used to separate variants in the data (i.e. parts of the data that changes over time) and invariants (i.e. parts of the data that are constant over time). Since the invariant parts of the data remains constant through time there is no need to predict this portion of the data. Thus a forecast model only needs to predict the variants in the data, which should have a lower dimensionality than the original data set. The predictions of the variants are then recombined with the invariants to form the forecast of the original (high dimensional) data set. [GROT04]

Principle component analysis can be performed using a bottleneck network, for more information see Section 3.2.5.

2.7.1 Input Variable Selection

As discussed earlier in Section 2.7, high dimensional input data sets might lead to forecasting models with a lower ability to predict than if a low dimensional input set is used. The selection of input variables to a forecast model is a very difficult task and should be done with great care, if possible with expert knowledge of the market [MAKR98, YANG07].

Makridakis et al. (1998) suggests using a "long list" and a "short list" of input variables [MAKR98, p. 275-277]. The "long list" contains a first preliminary set of input variables (stock, fundamental and derived data etc.), preferably assembled with the help of expert knowledge. This list is then reduced to a "short list", which is used as input to a forecast model. [MAKR98]

There are several methods to choose from and combine when reducing the "long list" to the "short list", and some of them are discussed below.

- Input-Input Correlation Check:

  As discussed in Section 2.7, two or more input variables that are highly correlated will supply more or less the same information and can even lead to a reduction of the forecasting performance. This can be avoided through the use of a correlation check on the input variables, where one of the variables is removed if it is highly correlated with another variable (i.e. all but one of a set of highly correlated variables are removed) [TIET08, MAKR98].

- Input-Target Correlation Check:

  Another way to shorten the list of input variables is to make a correlation check between the input variables and the target variables. Input variables that have a high correlation with the target should be favored to remain in the "short list". This since high correlation indicates that the input variable have a strong positive influence on the forecasting performance. [TIET08]

  Notice that the correlation is a measure of linear dependency and might thus miss non-linear associations, although this is not common in practice [MAKR98, DANI99].

- Quality of Input Data:

A third way to reduce the variable list is to remove input variables with low quality data. The quality of the data is affected by e.g. missing values, outliers and noise (see Sections 2.3.4 and 2.3.3). The amount of available data can also be considered a quality issue since short data sets might not be enough to properly train and evaluate a forecast model. In addition to this, quality issues can arise if the preprocessing of the raw input series is not done properly.

2.8 Training, Validation and Generalization Set

In general when developing a forecast model, the data used is divided into three sets; a training, a validation and a generalization set. A visual representation of these sets can be seen in Figure 2.3, and a short description of them follows.

- Training set:

  The training set is used during the training of the forecast model and when using neural networks the patterns in this set affects the weights in the network [TIET08].

- Validation set:

  A number of patterns can be removed from the training set and added to the validation set instead [TIET08]. This set can then be used to evaluate the training of the forecast model, enabling the detection of e.g. overfitting [TIET08, HEL98a].

  Although patterns can be picked in any order from the training set, when using financial time series the most recent data should be used for the validation set. This since it enables the ability to evaluate the stability of the model over time. [NEUN98]

- Generalization set:

  The generalization set contains patterns that are not included in the other two sets [TIET08]. When forecasting time series, these patterns shall be chosen so that the generalization set only contains more recent data than the training and validation sets (see Figure 2.3) [TIET08, NEUN98]. This set is used to evaluate the generalization performance of the model [TIET08, HEL98a].

  During the selection and training of the model, the generalization set is assumed not to exist (i.e. this data is in the future). Accordingly, all decisions that affects the model must be based solely on data that is not included in the generalization set [TIET08]. In the case of time series, this can be expanded to not including patterns that are more recent in time then the data in the generalization set.

  This means that, when characteristics of the data are used for preprocessing (e.g. mean, variance etc.), these are always calculated based on the patterns in the training and validation sets, never the generalization set [TIET08].

Notice that no pattern can belong to more than one of these sets [NEUN98]. In addition to this, it is always a good idea to define the sets explicitly, so that it is always clear to which of the training, validation and generalization set a pattern belongs [TIET08].

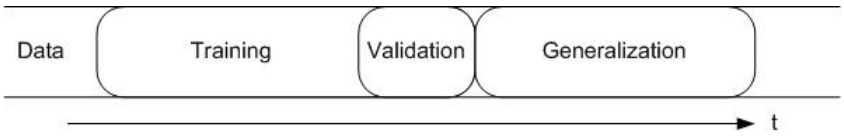

Figure 2.3: This figure visualizes how the data should be divided into a training, a validation and a generalization set when using time series with a forecasting model.

When using artificial neural networks to forecast financial time series, the separation of data into sets is very important and can have a large impact on the performance of the model [HEL98a]. Often the training set needs to be larger than for most linear models, since networks with complex architectures have many free weights that needs to be estimated [MCNE05, WALC01]. This might lead to old data (in relation to the generalization data) being used during training, which can be a problem since this data might not reflect the market during the generalization period very well [MCNE05, WALC01].

Walczak (2001) discusses the time series recency effect, which states that using training data that is closer in time to the generalization set results in better performing forecast models. His research, where feed-forward networks are used, indicates that there is a critical amount of training data that produces the optimum forecast, usually a maximum of two years of daily data. It is also indicated that adding additional training data to the critical amount will not lead to an increase, it might even lead to a decrease, of the forecasting performance. [WALC01]

Chapter 3

Neural Networks

The human brain is capable of processing a wide variety of data, such as interpreting and detecting nuances in speech and recognize different visual objects. The capabilities of the brain can be summarized into pattern recognition, perception and motor control [HAYK94]. In addition to this it is able to learn, memorize and generalize [ENGE02]. Several of these tasks can be done simultaneously and faster by the brain compared to a digital computer [ENGE02]. With this in mind the artificial neural network was developed as an effort to imitate the human brain [ENGE02].

The brain is composed of several neural cells (biological neurons), each consisting of a cell body, dendrites and an axon (see Figure 3.1). These neurons are interconnected and thus creates a larger biological neural system. The connections, called synapses, are made between the dendrite of one neuron and the axon of another. When a neuron receives a signal through the axon, it either inhibit or excite this signal and passes it on through its dendrites to all connected neurons. [ENGE02]

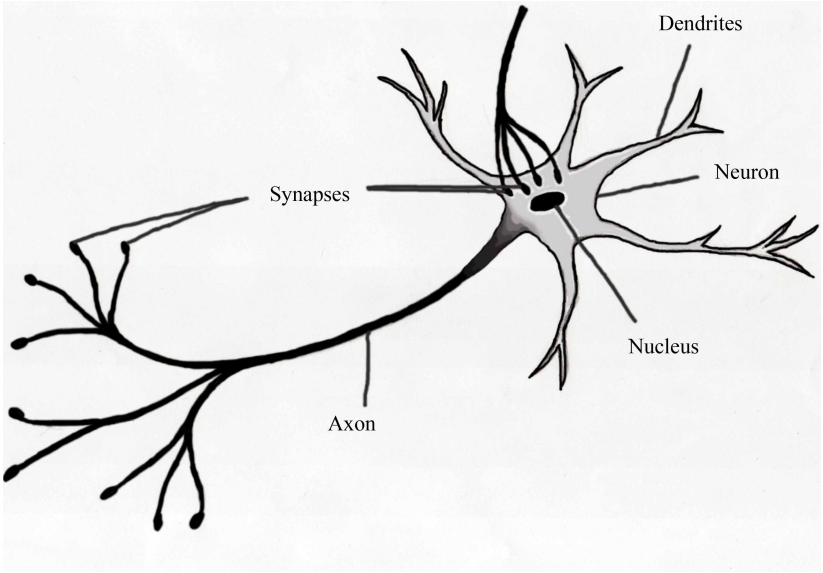

Figure 3.1: Biological neuron showing the cell body, axon and dendrites.

The concept of artificial neural networks (ANN) is to model the neural network of the brain. To this end, the ANN contains a set of interconnected artificial neurons that models the biological neurons. [ENGE02]

Compared to silicon logic gates the human neurons are slower, operating at speeds of the magnitude milliseconds (10^{-3} s) while the logic gates operates at speeds of the magnitude nanoseconds (10^{-9} s) [HAYK94]. Still the vast number of neurons and synapses in the human brain makes it very powerful [HAYK94]. Estimations indicates that the human brain contains approximately 10-500 billion neurons and 60 trillion synapses, structured into about 1 000 main modules, each having around 500 neural nets [ENGE02].

The current state of ANN modeling and technology allows for moderate sized problems with a single objective to be solved. This is far from the capabilities of the human brain, which is able to solve several problems simultaneously. The main obstacles for creating ANN as powerful as the human brain are the lack of computing power and storage space. [ENGE02]

3.1 Artificial Neurons

The artificial neuron (see Figure 3.2) is an essential part of neural networks and generates a mapping from the input space to the output space, usually in the interval $[0,1]$ or $[-1,1]$ (depending on the activation function used) [ENGE02, HAYK94]. It has three elements; synapses, adder and an activation function [HAYK94].

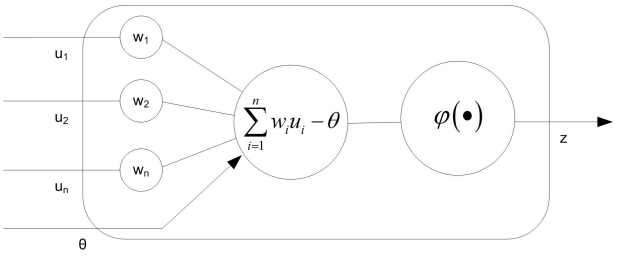

Figure 3.2: Artificial neuron showing the synapses, adder and activation function [GROT04].

Synapses are weighted connections through which input signals are received. The weight of the synapse determines the strength of the input signal and can be positive or negative. The weighted input signal can then be calculated by multiplying the input signal with the associated synapse weight (u_i and w_i in Figure 3.2), i.e. $u_i w_i$. [GROT04]

The adder then calculates the net input signal n by summing the weighted input signals (see Equation 3.1) [HAYK94]. The net input signal is then lowered by subtracting the bias term θ [GROT04]. Sometimes a threshold term is used instead of the bias; the threshold term is in fact the negative bias term [HAYK94].

$$n = \sum_{i=1}^{n} w_i u_i \tag{3.1}$$

When the net input, lowered with the bias, reaches a certain activation level the artificial neuron emits an output signal (see Equation 3.2). This is controlled by the activation function

$\varphi(\cdot)$ (see Section 3.1.1). [GROT04]

$$z = \varphi\left(\sum_{i=1}^{n} w_i u_i - \theta\right) \tag{3.2}$$

3.1.1 Activation Functions

The activation function, also known as the squashing function, determines the output (i.e. the strength of the firing) of the neuron based on the net input and bias (see Section 3.1) [HAYK94, ENGE02]. The range of the output is limited to some interval by the activation function [GROT04].

There exists several different kinds of activation functions, some of which are the threshold function (see below) and sigmoid functions (a group of functions) etc. The sigmoid functions are the most common type of activation functions and they are monotonically increasing, continuous and differentiable (e.g. the logistic function and the hyperbolic tangent function, see the following sections) [GROT04].

Threshold Function

The threshold function (see Equation 3.3) is a binary valued function with the range $[0,1]$ and neurons using this activation function are often referred to as McCulloch-Pitts model [HAYK94]. There also exists threshold functions with other ranges (e.g. $[-1,1]$) [ENGE02].

$$\varphi(u) = \begin{cases} 1 & \text{if } u \geq 0 \\ 0 & \text{if } u < 0 \end{cases} \tag{3.3}$$

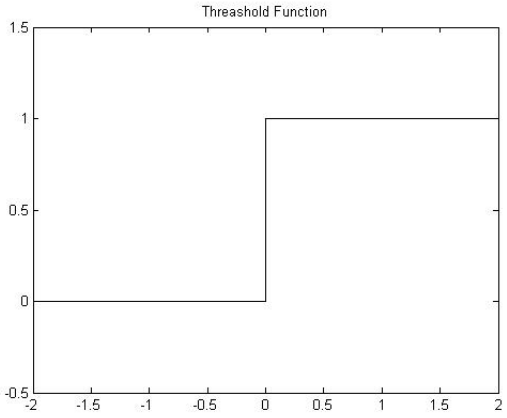

Figure 3.3: The graph of the threshold function shows how it, at a certain activation level, changes the output from zero to one.

Linear Function

The linear activation function multiplies the net input with a constant value k to produce the neuron output, which can be seen in Equation 3.4. Also note that the range of the linear activation function is $(-\infty, \infty)$. [ENGE02]

$$\varphi(u) = ku \tag{3.4}$$

Logistic Function

The logistic function (see Equation 3.5) is a sigmoid function with the range $(0, 1)$ [HAYK94, ENGE02]. The parameter a affects the slope of the function and when a goes towards infinity, the logistic function becomes a threshold function [GROT04].

$$\varphi(u) = \frac{1}{1 + e^{-au}} \tag{3.5}$$

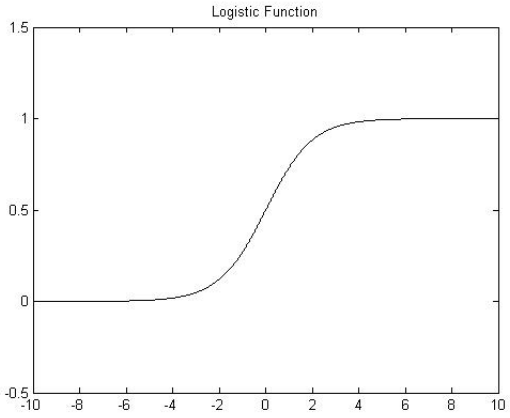

Figure 3.4: The graph of the logistic function shows the output value (between zero and one) for certain activation levels. In this graph a has been assigned a value of one.

Hyperbolic Tangent Function

The hyperbolic tangent function (see Equation 3.6) is also a sigmoid function, with the range $(-1, 1)$ [HAYK94, ENGE02].

$$\varphi(u) = \tanh\left(\frac{u}{2}\right) \tag{3.6}$$

The hyperbolic tangent function is almost linear close to zero, which means that input values close to zero passes almost unchanged. On the other hand, large input values are squeezed to the limits of the hyperbolic tangent function (i.e. towards 1 or -1). This means that the hyperbolic tangent function has the ability to reduce the effect of outliers in the data (assuming that the data in general is centered around zero). [GROT04]

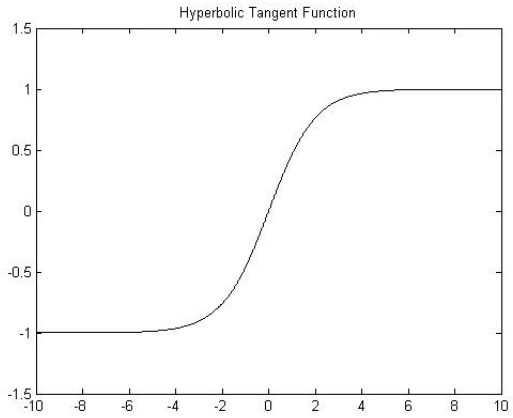

Figure 3.5: Graph of the hyperbolic tangent function. It can be seen how values close to zero passes through almost unchanged, while large values are squeezed towards 1 or -1 [GROT04].

The hyperbolic tangent activation function may cause numerical difficulties when used in large neural networks with the error back-propagation algorithm. This problem is however avoided when the vario-eta optimization algorithm is used (see Section 3.3.5). For more information, see Grothmann (2004). [GROT04]

3.2 Neural Network Architecture

The architecture of a neural network describes how their neurons are connected to each other [HAYK94]. It is important to notice that the architecture of a neural network and the learning rule (see Section 3.3) used to train it are closely related [HAYK94]. There are many types of neural networks, for example feed-forward neural networks, functional link neural networks, product unit neural networks, recurrent neural networks, time-delay neural networks and lattice structures [ENGE02, HAYK94]. Some of these architectures will be covered briefly in the following sections.

A neural network can be divided into several layers, where the layer can be one of the three different types that are listed below.

- Input layer:

  Contains source nodes that gathers information from the outside world and passes it on to the rest of the neural network [HAYK94, GROT04].

- Hidden layer:

  Contains neurons (i.e. computational nodes) and is located between the input and output layers of the neural network [GROT04].

- Output layer:

  In addition to having neurons, the output layer also provides the response of the neural network to the outside world [HAYK94, GROT04].

There is only one input layer and one output layer, while the number of hidden layers may vary (including no hidden layer). Since no computation takes place in the input nodes, the input layer is ignored when determining the total number of layers in a neural network (e.g. a two layered network has one input, one hidden and one output layer) [FAUS94].

3.2.1 Feed-Forward Networks

A network is feed-forward when the connections in the network are directed forward, from the input layer towards the output layer and never in the other direction (see Figure 3.6) [HAYK94]. This means that a feed-forward neural network has no feedback loops (see Section 3.2.2) or connections within the same layer [GROT04]. Usually the input to neurons in a layer is obtained from the output of neurons in the immediately preceding layer [HAYK94].

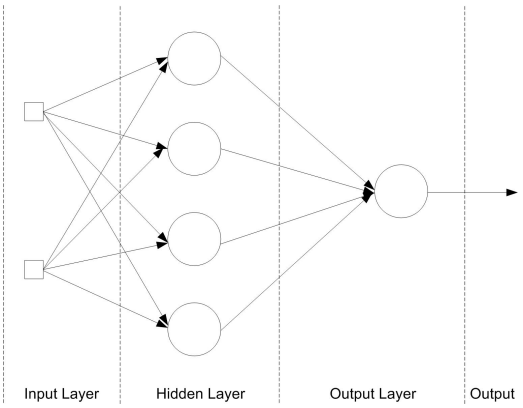

Figure 3.6: Simple multilayered feed-forward neural network with two input nodes, four hidden neurons and one output neuron.

If the input layer is directly projected onto the output layer (i.e. there is no hidden layer) the network is said to be a single-layer network, but if there is one or more hidden layers the network is a multi-layered network [HAYK94]. Adding more hidden layers to the feed-forward neural network might enable it to solve more complex problems, since higher order statistics may be extracted [HAYK94, MCNE05].

A feed-forward neural network can be fully connected, where all neurons (or nodes) in every layer are connected to every neuron in the next layer, or partially connected where some of the connections are missing (i.e. some of the connections of a fully connected feed-forward network are removed) [HAYK94].

Multi-Layered Perceptrons (MLP) Networks

Multi-layered feed-forward neural networks, where each neuron uses a smooth (i.e. differentiable everywhere) non-linear activation function (e.g. the logistic or hyperbolic tangent functions, see Section 3.1.1), are also known as the multi-layered perceptron (MLP) network [MCNE05, HAYK94].

It has been proven that if the multi-layered perceptron network (with one or more hidden layers) has a sufficiently large number of hidden neurons, it is in principle able to approximate any continuous function [GROT04, ENGE02].

The multi-layer perceptron network with one hidden layer is the most commonly used neural network type in financial applications. In addition to this the multi-layered perceptron network is a good choice as an alternative to the linear forecasting models when forecasting. [MCNE05]

3.2.2 Feedback Loop

A feedback loop refers to when the output of a neuron influence the input of that very same neuron, either directly or indirectly through other preceding neurons [HAYK94]. The single-loop feedback is an example of when a neuron is affected directly by the feedback loop and can be seen in Figure 3.7. For an example of when a neuron is affected indirectly by the feedback loop, see Figure 3.8.

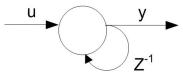

Figure 3.7: This figure shows a single-loop feedback which feeds the output of the neuron back to the same neuron using an unit delay operator z^{-1} [HAYK94].

Referring to Figure 3.7, the output from the unit delay operator z^{-1} is delayed one time unit with respect to its input. The unit delay operator gives the neural network a non-linear dynamic behavior, which plays a key role in the network's ability to retain memory. [HAYK94]

3.2.3 Recurrent Networks

If a neural network has one or more feedback loops (see Section 3.2.2) it is called a recurrent neural network (see Figure 3.8) [HAYK94]. Recurrent neural networks have the ability to retain memory, which enables them to learn temporal characteristics of the data [HAYK94, ENGE02]. This makes recurrent networks suitable to use with data that has a time dimension (e.g. financial time series) [MCNE05].

In addition to this, feedback loops greatly improves the learning ability and performance of neural networks [HAYK94]. However, a problem with recurrent neural networks is that they tend to focus on the most recent data, thus lowering their ability to learn temporal structures [GROT04].

There exists several different types of recurrent neural networks and three of them are listed below.

- Elman's:

  The feedback loops originates from neurons in the hidden layer, before the signal has been squashed by the activation function. This information is then used as input to the hidden layer. [MCNE05]

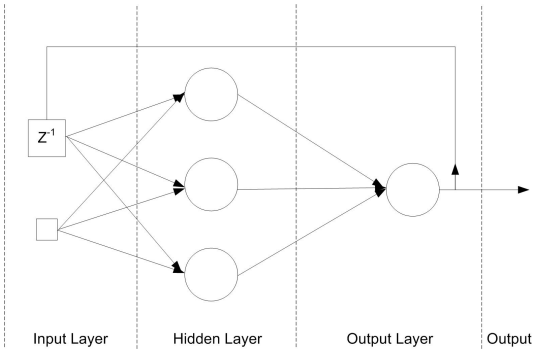

Figure 3.8: Jordan's recurrent neural network with one input node, three hidden neurons and one output neuron.

- Jordan's:

  In Jordan's recurrent networks the feedback loops originates from neurons in the output layer, see Figure 3.8 [ENGE02].

- Time-Delayed:

  Time-delayed networks are another type of recurrent networks, see Section 3.2.4 for more details.

3.2.4 Time-Delay Recurrent Networks

According to Haykin (1994) "A dynamical system is a system whose state varies with time" [HAYK94, p. 539]. This is true for many financial markets (e.g. stock and foreign exchange markets). Time-delayed recurrent neural networks (see Figure 3.10) can be used to model dynamical systems [GROT04].

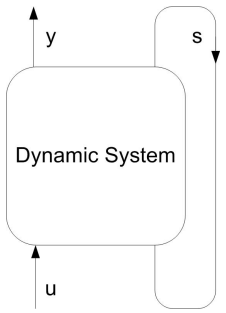

Figure 3.9: A dynamic system, where s is the current state, u the input and y the output [GROT04]

A dynamic system can be described recurrently with a set of equations, one that maps the current state from the previous state and the input (see Equation 3.7), while the other equation (see Equation 3.8) provides the output of the model. [GROT04, ZIMM00]

$$s_t = f(s_{t-1}, u_t) \tag{3.7}$$
$$y_t = g(s_t) \tag{3.8}$$

The task of finding a specific dynamic system can thus be accomplished by finding the set of functions (see Equations 3.7 and 3.8) that minimizes the difference between the model output and the desired output, which corresponds with the dynamic system that is to be found. This can be accomplished through the use of a time-delayed neural network, where the task of finding a set of functions is transformed into the task of finding a set of weights. The network that accomplishes this can be seen in Figure 3.10. [GROT04, ZIMM00]

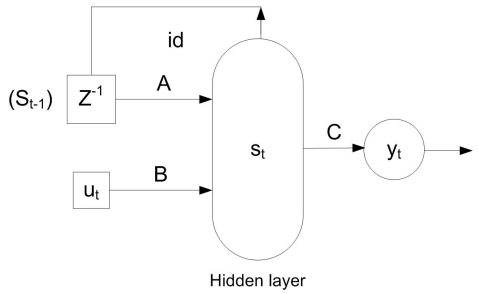

Figure 3.10: The time-delayed recurrent network has neurons in the hidden layer s_t that receives data from the input neurons u_t and sends information to the output neurons y_t. In addition to this, it also has feedback loops from the state neurons (in the hidden layer). Notice that the input node can represent several nodes, the state neuron several neurons and so on. This means that the weights (A, B, C) are matrices representing several synaptic weights, one for each connection between single neurons. [GROT04]

When a time-delayed recurrent neural network is used to model a dynamic system, the set of equations (see Equations 3.7 and 3.8) are transformed into Equations 3.9 and 3.10, where A, B and C are the weight matrices that are to be found. Notice that the output neurons (see Equation 3.10) have a linear activation function, while the hidden neurons (see Equation 3.9) have a hyperbolic tangent activation function. [GROT04, ZIMM00]

$$s_t = \tanh(As_t + Bu_t) \tag{3.9}$$
$$y_t = Cs_t \tag{3.10}$$

An interesting property of time-delayed recurrent neural networks is that they have relatively few free weights, which makes them very resistant to the problem of overfitting (see Section 3.5). [GROT04]

3.2.5 Auto-Associative Networks

An auto-associative network, also referred to as a compression-decompression network or a bottleneck network, can be used to perform principle component analysis and variant-invariant

separation on an input vector (see Section 2.7). The auto-associative network is best illustrated in Figure 3.11.

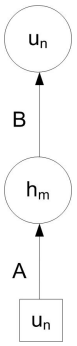

Figure 3.11: An auto-associative neural network, where n and m denotes the number of input units and/or neurons and where $n > m$. The n dimensional input space is transformed into a m dimensional space, and then reconstructed to a n dimensional space. The weight matrix A is referred to as a compression matrix and the matrix B as a decompression matrix. [GROT04, MCNE05]

Although non-linear activation functions may be used in the hidden layer (h_m), linear activation functions are usually preferred since the resulting neural networks only have one global minimum and are thus easy to train [REED99, PISS02]. Note that the compression matrix A in Figure 3.11 may be used to extract the principle components of the input data [PISS02]. If there is a desire to reconstruct the data at some point, the decompression matrix B may be used (e.g. the variant-invariant separation procedure described in Section 2.7). There also exists auto-associative neural networks for non-linear principle component analysis, for more information see e.g. McNelis (2005) [MCNE05].

3.3 Learning Process

A neural network has the ability to learn from its environment, thus improving its performance. The learning is carried out through iterative changes in the synaptic weights and biases, in accordance with some learning rule. One of these iterations is often referred to as an epoch (where the whole training set has been used to update the weights, not only a single pattern, see Sections 3.3.1 and 2.8). [HAYK94]

The following discussion of the learning process focuses on the error correction learning rule, for more information on alternative learning rules see e.g. Haykin (1994) [HAYK94], Engelbrecht (2002) [ENGE02] or Grothmann (2004) [GROT04].

3.3.1 Learning Categories

There exists several different learning rules, which can be divided into three main learning categories; supervised, unsupervised and reinforcement learning. A short list over some of the learning rules are listed below [HAYK94, ENGE02].

- Supervised Learning
  - Error Correction Learning Rule
    Usually uses error back-propagation and some optimization algorithm, e.g.:
    - Gradient Descent Optimization
    - Scaled Conjugate Gradient Optimization
    - Pattern by Pattern Optimization
    - Vario-eta Optimization
  - Boltzman Learning Rule
- Reinforcement Learning
  - Reinforcement Learning Rule
- Unsupervised Learning
  - Hebbian Learning Rule
  - Competitive Learning Rule

Supervised Learning Rules

In supervised learning there is a teacher present that has knowledge of the environment in the format of input - desired output pairs [HAYK94]. The teacher proceeds by presenting this knowledge to the neural network, thus teaching it a mapping between the input and desired output [HAYK94, ENGE02].

This training proceeds until the neural network reflects the teacher (i.e. it has learned all that it can from the teacher), after which the teacher can be removed. The neural network will then be able to work within the environment in an unsupervised manner. [HAYK94]

A drawback when using supervised learning is that the neural network is unable to learn anything outside the scope of the teachers knowledge (i.e. outside the training set) [HAYK94].

Reinforcement Learning Rules

Reinforcement learning rules have their basis in the study of animal learning behavior [HAYK94]. The basic idea is to learn through trial and error, where actions leading to good performance are reinforced and other actions are weakened [HAYK94, ENGE02]. The reinforcement or weakening of some action will then encourage or discourage the neural network to take the same action again [HAYK94].

Unsupervised Learning Rules

Unsupervised learning (also known as self-organized learning) tries to find patterns, basically by performing clustering (categorizing the input) without the presence of a teacher [HAYK94, ENGE02].

3.3.2 Error Correction Learning Rule

When using the error correction learning rule, the neural network is trained with input data for which some desired output is known. The difference between the actual neural network output y_t and the desired output y_t^d is called an error (or error information). This error is then used to find the network error function (see Section 3.3.4). The objective of the error correction learning rule is to minimize the error function (i.e. minimize the difference between the network output y_t and the desired output y_t^d), which is an optimization problem. [HAYK94]

The learning can be divided into two steps, a forward pass and a backward pass. In the forward pass the input is given to the neural network, which computes the network output with unchanged synaptic weights. In the backward pass, the error function (for the current output) is used to adjust the weights in the network. [HAYK94]

The backward pass can be divided into several smaller steps, where the first step is to find the error information. The error back-propagation algorithm (see Section 3.3.3) then uses this error information to find the gradient of the network error function, with respect to the synaptic weights. Finally this gradient is used by a optimization algorithm (see Section 3.3.5) to adjust the synaptic weights in the neural network. [GROT04]

Another important part of error correction learning is how the patterns are presented to the neural network during training, for more information see Section 3.3.6.

The error correction learning rule is suitable to use with neural networks when forecasting time series, since these time series can be used both as input and desired output when training the network [HAYK94].

3.3.3 Error Back-Propagation

As stated in Section 3.3.2, the objective of the error correction learning rule is to minimize the network error. To be able to do this the credit assignment problem must be solved [GROT04].

The credit assignment problem is the problem of determining which internal decision to blame or credit when some action (where the action depends on several internal decisions) leads to a worse or better result [HAYK94]. The problem arises when using supervised learning with neural networks that have hidden neurons [GROT04].

In practice the error information is used to update the synaptic weights in the neural network. The problem lies in that the error information is only available in the output layer, not to neurons in the hidden layers. This means that the error information needs to be propagated back through the network, from the output layer to the hidden layers. [GROT04]

The error back-propagation algorithm (also known as back-propagation) solves this issue by providing the first partial derivatives of the error function, with respect to the synaptic weights, to the whole neural network in an efficient way. These partial derivatives, which forms the gradient of the neural network's error function, can then be used by a optimization algorithm in order to find the weights that minimizes the error function (see Section 3.3.5). [GROT04]

As discussed in Section 3.4, prior knowledge can be introduced in the neural network architecture by using shared weights (e.g. when using finite unfolding in time, see Section 4.1). The standard error back-propagation algorithm can not handle shared weights, making it nec-

essary to use a version with a shared weight extension, for more information see e.g. Grothmann (2004) [GROT04].

3.3.4 Error Function

The error function, also known as the cost function, is used by the error correction learning rule when training the neural networks [HAYK94]. There exists several different error functions, which in general can be written as shown in Equation 3.11, where E_t represent the specific error function and T is the number of training patterns used [NEUN98]. Two different error functions will be covered in this section.

$$E = \frac{1}{T}\sum_{t=1}^{T} E_t \tag{3.11}$$

One of the more widely used error function is the mean squared error (MSE) function, see Equation 3.12 [HAYK94, GROT04]. An important property of the MSE function is that it requires that the data have certain distribution properties, e.g. weak stationarity (see Section 2.3.2) [HAYK94, NEUN98]. This means that outliers, which are common in financial data, is a large problem (see Section 2.3.3) [GROT04]. Outliers disturbs the training of neural networks, thus affecting their forecasting abilities negatively [GROT04, NEUN98].

$$E_t = \frac{1}{2}\left(y_t - y_t^d\right)^2 \tag{3.12}$$

In order to better handle these outliers, Grothmann (2004) suggests using the robust error function $\ln\cosh$ (see Equation 3.13). The parameter a is usually chosen so that $a \in [3,4]$. [GROT04]

$$E_t = \tfrac{1}{a}\ln\cosh\left(a\left(y_t - y_t^d\right)\right) \quad \text{where } a > 1 \tag{3.13}$$

For small errors, the $\ln\cosh$ error function acts similar to the mean squared error function (see Figure 3.12), while simultaneously reducing the impact of larger errors (caused by e.g. outliers) [GROT04].

Also note that the network output y_t can be thought of as a function value ($y_t = NN(u_t)$), where the neural network function is determined by the network architecture (i.e. the synapses, weights and activation functions etc.). As an example, a one layered feed-forward network with two input nodes and one hidden neuron, using the hyperbolic tangent function, can be described mathematically as shown in Equation 3.14.

$$y_t = NN(u_t) = \tanh\left(\sum_{i=1}^{2}(w_i u_i - \theta)\right) \tag{3.14}$$

3.3.5 Optimization Algorithms

When the error back-propagation algorithm (see Section 3.3.3) has found the gradient of the network's error function (see Section 3.3.4), an optimization algorithm can be used to mini-

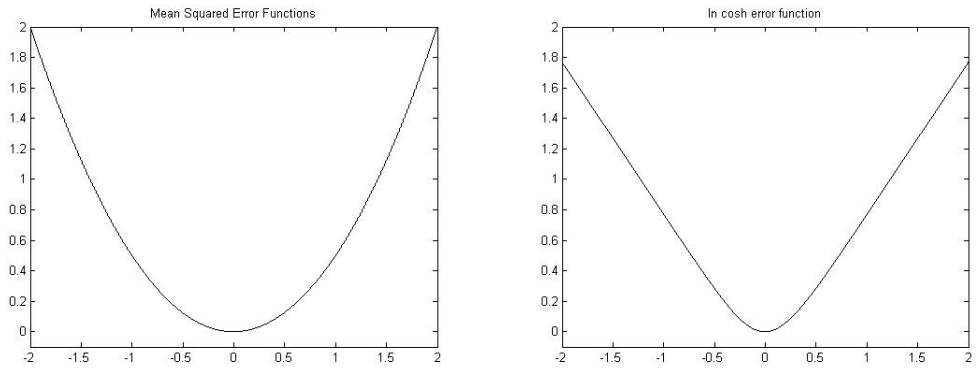

Figure 3.12: Left: The MSE error function. Right: The ln cosh error function, with a assigned a value of three. It can be seen that for small values the functions acts similar.

mize the error. Let Δw_t be the search direction determined by some optimization algorithm (e.g. gradient descent, vario-eta etc.) and η the step length along that search direction. Then the weights are updated iteratively according to Equation 3.15. [GROT04]

$$w_{t+1} = w_t + \eta \Delta w_t \tag{3.15}$$

There are a number of issues related to the optimization process that are good to keep in mind:

- Global vs. Local Minimum

  When optimizing non-linear functions, there is always a risk of finding a local minimum instead of a global. Since the error functions of neural networks are highly non-linear (i.e. they have a lot of local minima), there is a high risk that the solution found by an optimization algorithm, during a training session, only is a local minimum. This has the effect that, when training a network several times with random initial weights (see Section 3.3.7), different solutions are found by the optimization algorithm. Thus the accuracy of a network model may vary somewhat for different training sessions. [MCNE05]

- Step length (learning rate)

  The step length can either be constant, usually in the interval $[0.01, 0.1]$, or determined by a line search algorithm [TIET08, GROT04]. When using a line search algorithm to find the step length, a starting value and a precision value usually needs to be supplied [TIET08].

  The step length affects the convergence behavior of the optimization algorithm, where large step sizes might lead to it jumping over minima [MCNE05]. On the other hand, a too small step size leads to a slow convergence and a greater risk of getting stuck in an undesirable local minimum [ENGE02]. Thus the step length should be chosen sufficiently small so that the optimization is able to find local minima, thus also being able to converge [TIET08, GROT04].

Since the error function of a network usually is highly non-linear with a lot of local minima, there might be a desire to force the optimization algorithm not to settle on the first found local minimum. This can be accomplished using a larger step size, which enables the optimization algorithm to jump over local minima, thus examining a larger search space. As the training proceeds, the step length can be reduced in order to find better local minima and to guarantee convergence. [TIET08]

- Algorithm properties

  Different optimization algorithms have different properties concerning accuracy, memory and computational requirements. There is often a trade off between these properties when choosing an optimization algorithm, e.g. an algorithm with a low computational requirement might not be very accurate.

There exists a lot of different optimization algorithms and in this section two of them will be discussed briefly. For more information, see e.g. Grothmann (2004) [GROT04] or Engelbrecht (2002) [ENGE02].

Gradient Descent Optimization

When using gradient descent optimization, the search direction is the same as the negative gradient, which is supplied by the back-propagation algorithm [GROT04]. The negative gradient always points in the direction of largest decrease of the error function [CORN07]. Typically the gradient descent algorithm is used in full batch mode (see Section 3.3.6) [GROT04].

A drawback with the gradient descent algorithm is that it usually finds the minimum that is closest to the starting point, which often is an undesirable solution [GROT04].

Vario-Eta Optimization

In addition to using the negative gradient, vario-eta has the ability to scale the step length individually for each weight. This rescaling both makes vario-eta faster than gradient descent and more suitable to use when training large neural networks (especially if the hyperbolic activation function is used, see Section 3.1.1). This is also the reason to why the vario-eta algorithm behaves like a stochastic approximation of the quasi-Newton algorithms. [GROT04]

Experiments by Neuneier et al. (1998) indicates that the vario-eta algorithm performs best when used with small batch sizes, using less than 20 patterns per batch. In addition to this, the step length should be chosen as large as possible in order to enforce the rescaling properties, which can be seen as a penalty, of the vario-eta algorithm. [NEUN98]

3.3.6 Pattern Presentation

When training a neural network using the error correction learning rule (or another supervised learning rule) the patterns needs to be presented to the network [HAYK94, ENGE02]. This means that there is a need to decide which and how many of the patterns that are to be presented to the network, which corresponds to the forward pass of the error correction learning rule (see

Section 3.3.2), before the weights are updated. The list below describes three different ways in which the patterns can be presented to the network.

- Batch mode:

  In batch mode, also referred to as off-line learning, all the patterns in the training set are presented to the network before the weights are updated [HAYK94, ENGE02].

- Stochastic mode:

  In stochastic mode, also referred to as on-line learning, the weights are updated after each pattern presentation. The patterns are drawn randomly from the training set. [HAYK94, ENGE02]

- Stochastic batch mode:

  There also exists a method that combines the stochastic and batch methods mentioned above, where a batch may include more than one and less than all patterns in the training set. The weights in the network are updated each time all the patterns in a batch have been presented to the network. [TIET08]

  There are several ways in which the patterns in a batch can be selected from the training set; e.g. randomly, permute or sequential. When using the permute selection strategy, the patterns are selected randomly from the training set, excluding those patterns already included in a previous batch. The major difference between the permute and random selection strategies is that the latter may also choose patterns that already have been included in previous batches. [TIET08]

3.3.7 Weight Initialization

When an artificial neural network is to be trained using the error correction learning rule (see Section 3.3.2), the first step is to initialize the weights to some starting value. There are two major issues that should be addressed when initializing the weights; saturation of the neurons activation functions and symmetry of the weights [REED99]. The main benefit of a carefully chosen set of initial weights is a speed up of the learning process [REED99].

The problematic of the first issue lies in that large initial weights might result in a high internal activation level and thus an immediate saturation of the neuron's activation function (see Sections 3.1 and 3.1.1). This leads to small gradients during training, which in turn leads to small weight updates and thus slow learning. This problem can obviously be avoided through the initialization of the weights to small values centered around zero. [REED99]

The problem of symmetry occurs if the neurons have identical weights, which has the effect that the neurons in the network will behave in the same way, including the way that their weights are updated. This is obviously not a desirable effect and is easily avoided by not initializing the neurons to identical weights. [REED99]

In addition to these issues, very small weight values leads to small error values when they are propagated back through the network using the error back-propagation algorithm

(see Section 3.3.3), which leads to a slow learning process. This since the partial derivatives calculated by the error back-propagation algorithm includes weight terms. [REED99]

There are two main categories of weight initialization methods; random and non-random initialization, which will be covered briefly in the following sections. For more information concerning weight initialization see e.g. Reed et al. (1999). [REED99]

Random Weight Initialization

Random weight initialization, which is the most commonly used category of weight initialization methods, assigns random values to the weights in a neural network. The fact that the weights are initialized to random values means that the issue of symmetry is solved. To avoid immediate saturation of the activation functions, the range of the weight values can be limited so that they are small and centered around zero. There are several different suggestions of ranges that can be used, one of which can be seen in Equation 3.16: [REED99]

$$\left[-\frac{A}{\sqrt{N}}, \frac{A}{\sqrt{N}}\right] \tag{3.16}$$

The range in Equation 3.16 is dependent on the number of synapses N connected to the neuron and a constant value A (suggestions of the value of A ranges between one and three) [ENGE02, REED99]. In general, too small initial values are usually preferable compared with too large initial values [REED99].

Non-Random Weight Initialization

The second category of weight initialization includes non-random methods, where the weights are initialized to values that have been previously determined by some non-random method, e.g. the weights obtained during a previous training session of the network. The non-random method can speed up the training process of neural networks, provided that the estimated initial values have a high quality. The drawback with these methods is that an extra step is needed, before the actual training of the network starts, to determine these weight values. [REED99]

3.4 Bias - Variance Dilemma

When forecasting there usually is a difference between the predicted and actual value, which is referred to as an error $(y_t - y_t^d)$. Using the mean squared error function, this error can be rewritten into terms of bias and variance (see Equation 3.17) [HEL98a]. For more details, see e.g. Hellström (1998) [HEL98a] or Haykin (1994) [HAYK94].

$$\left(y_t - y_t^d\right)^2 \rightarrow \sigma^2 + \left(\mu - y_t^d\right)^2 \tag{3.17}$$

The bias term $\mu - y_t^d$ describes how much the average prediction differs from the actual value and the variance σ^2 tells how much the predictions varies [HEL98a].

In general the optimum scenario would be to have both a low bias and a low variance, this is however hard to achieve due to the bias - variance dilemma [HAYK94]. The bias - variance

dilemma can be explained using the extreme cases of complexity in the forecasting models [HEL98a]. The complexity of a neural network is determined by the number of free weights that it has, where a large number leads to a highly complex network and a low number to a less complex network [HEL98a].

Neural networks with a low complexity have an architecture that is unable to fit the data when trained. These types of network models are unable to express the finer details of the data, which means that the variance will be low and the bias high. On the other hand, highly complex networks have an architecture that is able to fit the data when trained and are thus able to express the finer details of the data. This leads to a high variance and a low bias. [HEL98a]

However, the complexity of the model can be lowered with the use of prior knowledge (i.e. 'adding bias'), thus reducing the bias (since the added bias is harmless) without affecting the variance [HEL98a]. It is important to make sure that the introduced knowledge actually is harmless in the area where the model will be applied [HAYK94]. Also note that prior knowledge can improve a model in one area of application, while worsen it in other areas [HAYK94].

In neural networks, prior knowledge can be introduced in several different ways, e.g. using shared weights (i.e. several synapses shares the same weight), partial connected networks and specialized architectures [HAYK94].

3.5 Overfitting

In Section 3.4 it is discussed how neural networks with complex architectures (i.e. many free weights) can be trained to fit the data. When the data set used to train a neural network with a complex architecture is small, noisy and contains outliers, which is common in financial data, the network will not only learn the underlying function but also the noise in the data. This phenomenon is known as overfitting and results in a loss of generalization performance. [GROT04]

Also note that the complexity of a neural network can not be made arbitrary low, since this might lead to the network being unable to learn the underlying function from the data [GROT04].

There are several methods to reduce the risk of overfitting, some of which can be seen below.

- Larger training sets

  Larger training sets, at least 10 times as many patterns as there are free parameters in the network, reduces the risk of overfitting. However, it does not guarantee that overfitting will not occur. [GROT04]

  An alternative, if only small training sets are available, is to use artificial noise. This since a small training set can be made infinitely large by adding a random noise term to a pattern each time it is presented to the network. [NEUN98]

- Reduced complexity

A reduction of the number of free weights (i.e. complexity) in the neural network also reduces the risk of overfitting. This can be accomplished in several different ways, e.g. using node or weight pruning (see Section 3.5.3), penalty methods or incorporating prior knowledge into the network architecture (i.e. 'adding bias', see Section 3.4). [GROT04]

The penalty methods can be used to reduce the complexity of the network by adding a penalty term to the network's error function. There exists several different algorithms, e.g. weight decay. [ENGE02]

An example of a network type with few free parameters is the time-delayed recurrent neural network and the unfolded time-delayed network (since it uses shared weights, see Sections 3.2.4 and 4.1). [GROT04]

- Stoping criterion

  A stopping criterion (early and late stopping, see Sections 3.5.1 and 3.5.2) can be used, possibly in combination with methods to refine the neural network architecture (e.g. pruning and penalty methods, see Section 3.5.3), to reduce the problem of overfitting. [GROT04]

3.5.1 Early Stopping

One of the simplest methods of avoiding overfitting is to observe the error on the validation set during training. When the error starts to increase this is usually a sign of overfitting, and thus the training is stopped. [GROT04]

An extension to this method is to, after the training has been stopped, use a weigh pruning method (see Section 3.5.3) to reduce part of the neural network model. The training is then restarted and continues until the method described above stops it again, at which point the procedure repeats itself. This iteration continues until it results in a trained network without overfitting. [GROT04]

A drawback with the early stopping method is that it may lead to network models with low complexity, since the early stopping method sometimes stops the training after just a few epochs [GROT04].

3.5.2 Late Stopping

Late stopping, opposite to early stopping, trains the neural network until the error (on the training set) has been minimized, which most of the time means that overfitting has occurred. As a result of the extensive training, the maximum amount of information can be extracted from the training data. The generalization ability is then increased by using a weight pruning method (see Section 3.5.3) to reduce the neural network model. Training is then restarted and the procedure repeats until a network model without overfitting has been found. [GROT04]

The late stopping method can be modified by adding noise to the input or using a penalty term in the error function, both of which delays the stopping point [GROT04].

3.5.3 Network Pruning

Usually, a first initial network architecture is chosen based on assumptions made of the problem that is to be solved [NEUN98]. This often leads to networks that are more complex than they need to be and thus vulnerable to the problem of overfitting [ENGE02].

A solution to this problem is to use a pruning technique, where the idea is to start with a network that is more complex than necessary, train it and then if necessary apply a pruning algorithm (during or after the training) to reduce the complexity of the network [REED99]. There are several ways in which the complexity of a network can be reduced; removing weights (i.e. weight pruning), hidden neurons or input nodes (i.e. node pruning) [ENGE02].

There exists many different pruning algorithms, that can be used to decide which nodes or weights to remove based on some measure of significance. The objective of all pruning algorithms is to find the least complex network that is able to accurately learn the underlying function. There are several benefits with finding a less complex network, e.g. lower sensitivity to the problem of overfitting, shorter training sessions and smaller training sets. [ENGE02]

A few of the pruning algorithms that can be used are: stochastic pruning, instability pruning, inverse-kurtosis, early brain damage etc. [GROT04, REED99]. For more information, see e.g. Reed et al. (1999) [REED99], Neuneier et al. (1998) [NEUN98], Grothmann (1994) [GROT04] and Engelbrecht (2002) [ENGE02].

3.5.4 Cleaning with Noise

As discussed in Section 3.4, a highly complex neural network can be trained to fit the data. Since financial time series usually are noisy, a complex network will not only learn the important structures of the data, but also the corrupt parts. This is known as the problem of overfitting (see Section 3.5) and one possible remedy to this problem is using the cleaning technique [NEUN98]. The cleaning technique stands out from the other techniques used to overcome the problem of overfitting, since it actually assumes that the data is corrupt [NEUN98].

Thus, the cleaning technique approaches the problem of overfitting by trying to find the errors, for each pattern in the training set, that can be attributed to the corrupt parts of the data. These errors are then stored in a correction vector Δu_t, which is used to clean the input, as can be seen in Equation 3.18 [NEUN98].

$$u_t^c = u_t + \Delta u_t \tag{3.18}$$

Important to notice is that, in order for the cleaning technique to find the correction vector, the target values must be known. Thus, the correction vector can only be calculated for the training data. This means that the network can be trained using a cleaned training set, while the validation and generalization sets still are noisy. This is obviously an undesirable effect, since the data used to train the network and the data that is to be predicted will have different properties. A remedy to this problem is using the cleaning with noise technique, which adds a noise term to the training data. [NEUN98]

By adding a random noise term to the input, the training data can be expanded infinitely, thus forcing the network to concentrate on the underlying structures in the data and ignore the

corrupt parts. If the noise terms are drawn randomly from the correction vector (estimated using the cleaning technique), they will have the same distribution as the noise in the training set. This means that the network will be trained on a data set that have a general description of the noise in the original data. Since the noise in the original training data is less general than the added noise terms, they might not correspond as well with the noise in the generalization set. Thus a network that uses the cleaning with noise technique should have a better description of the noise in the validation and generalization sets and thus a better generalization performance than when not using the technique. [NEUN98]

If k is a random number such that $k \in [1, T]$ and T is the number of patterns in the training set, the corrected input with added noise can be determined using Equation 3.19 [NEUN98]

$$u_t = u_t^d + \Delta u_t - \Delta u_k \tag{3.19}$$

For more information concerning the cleaning method, including how the correction vector is determined, see e.g. Neuneier et al. (1998) [NEUN98].

3.6 Thick Modeling

When training an artificial neural network several times with random initial weights (see Section 3.3.7), the obtained results will be different [ENGE02, MCNE05]. This since the different starting points in the search space means that the optimization algorithm used ends up in different local minima (see Section 3.3.5).

Thick modeling, also known as ensemble networks, is based on this phenomenon. The basic idea is to train several networks, either with the same or different architectures, and then derive the output of the thick model from the several included networks [ENGE02, MCNE05]. There are several different approaches in which the output of the thick model can be derived, some of which can be seen in the list below [ENGE02].

- Selection

  The output from the internal neural network with the best performance is selected and used as output from the thick model.

- Average

  The average of the internal neural network outputs is calculated and used as output of the thick model.

- Weighted

  A weight is assigned to the different internal neural networks based on some measure of quality. The weighted sum of the internal networks' output is then used as output from the thick model.

A drawback with thick models is that they are more time and resource consuming, since instead of one network to train, there are several. Also note that there is no need to train the networks in parallel since the final result of the thick model can then be calculated 'off-line'.

3.7 History

This section gives a brief overview of some events in the history of neural networks. For a more complete description of the neural network history, the reader is referred to e.g. Haykin (1994) [HAYK94], Engelbrecht (2002) [ENGE02], Grothmann (2004) [GROT04] and Fausett (1994) [FAUS94].

- 1943: Model of a neuron, by Warren McCulloch and Walter Pitts

  In 1943 McCulloch and Pitts started to use mathematics to describe the human brain mechanisms [GROT04]. This enabled them to model logical operators (e.g. AND, OR or NOT) with artificial neural networks [GROT04]. Their work can be seen as the beginning of the modern era in the artificial neural network field [HAYK94, FAUS94].

  McCulloch and Pitts developed a model of an artificial neuron (known as the McCulloch and Pitts model), which has a simple threshold activation function where the output is either zero or one [GROT04]. This model of a neuron is still widely used [FAUS94].

- 1949: Hebian learning rule, by Donald Hebb

  Hebb suggested that individuals learns through the continuous modification of synapses (i.e. their connectivity) in the brain [HAYK94, GROT04]. This led to the development of the first learning rule for artificial neural networks, where the modification of synapses is carried out through changes in their synaptic weights [GROT04, FAUS94].

- 1958: Rosenblatt's perceptron, by Frank Rosenblatt

  In 1958 Rosenblatt investigated how the brain is able to distinguish between different types of stimuli, which led him to develop the perceptron. The perceptron, using a single McCulloch and Pitts neuron, is a pattern classifier that can assign input patterns to one of two different classes. By adding neurons to the perceptron, the number of classes, to which an input pattern can be assigned, also increases. [GROT04]

- 1969: Critic of perceptrons, by Marvin Minsky and Saymour Papert

  In 1969 Minsky and Papert released a book, entitled "Perceptrons", in which they criticized the perceptron [HAYK94]. In their book they pointed at two fundamental flaws of the perceptrons, one of which was the inability of single layered perceptrons to solve classification problems which are not linear separable [HAYK94, GROT04].

  The second flaw they pointed out is known as the credit assignment problem, which indicated that it would be impossible to train multilayered perceptrons [GROT04]. The release of their book led to a drop in the research concerning artificial neural networks [HAYK94, GROT04].

- 1982: Self-organizing maps, by Teuvo Kohonen

  Influenced by studies of the brain (i.e. brain maps) Kohonen developed an artificial neural network type called self-organizing maps. These self-organizing maps uses an

unsupervised method of learning with a lattice structured architecture to map high dimensional input to a low dimensional representation. Kohonen's network is one of the more popular unsupervised artificial neural networks. [HAYK94, GROT04]

- 1982: Hopfield networks, by John Hopfield

  John Hopfield connected the artificial neural networks to the field of physics in 1982 [HAYK94, GROT04]. The Hopfield networks are constructed in such a way that they start in a random state and then moves to a final stable state, mimicking the behavior of dynamic systems [GROT04].

- 1985: The Boltzmann machine, by Ackley, Hinton and Sejnowski

  The Boltzmann machine was the first neural network with hidden layers that was successfully trained, thus proving that the Minsky and Papert speculation, concerning the ability to train multilayered perceptrons, was wrong [GROT04].

  The network can be described as a generalization of a Hopfield neural network, inspired by the optimization algorithm simulated annealing [GROT04].

- 1974, 1986: The standard error back-propagation algorithm, by Paul Werbos, David Rumelhart, et al.

  In 1974 Werbos solved the credit assignment problem for multilayered perceptrons with the invention of the back-propagation algorithm [GROT04].

  Rumelhart et al. then reinvented the back-propagation algorithm in 1986. As an example, based on the idea that feed-forward neural networks can be used to represent any recurrent neural network, they solved the credit assignment problem for time-delayed neural networks using unfolding in time and shared weights. [GROT04]

  With the invention of the back-propagation algorithm, the research concerning neural networks became important again and has since been one of the larger areas in computer science [ENGE02, GROT04].

- 1988: Radial basis function networks, by Broomhead and Lowe

  In 1988 Broomhead and Lowe introduced a new class of feed-forward networks, that came to be an alternative to the multilayered perceptron, called the radial basis function network [HAYK94, GROT04]. Radial basis function networks are universal function approximators and can be applied in areas such as pattern classification, function approximation or regularization [GROT04].

- 1990: Elman's Recurrent Neural Networks, by Jeff Elman

  Elman developed a recurrent neural network, often referred to as a simple recurrent network, which supplies unsquashed information from the hidden neurons, through the use of a context unit, as additional input to the network. Since the context unit delays the information for one time unit, the simple recurrent network is a time-delayed network. Similar networks were also developed simultaneously by several other researchers (e.g. Jordan's recurrent network). [GROT04]

- 1990's: Overfitting

  As a result of the neural networks' universal approximation abilities, a new problem was discovered in the 1990's known as overfitting, which is a loss of generalization performance. Overfitting occurs when neural networks starts to learn the noise in the data and not only the relevant information. [GROT04]

  There exists some techniques to avoid the problem of overfitting, some of which are pruning of the network architecture, special learning algorithms (early and late stopping) and careful selection of the neural network architecture (i.e. activation function, error function etc.). Another solution is provided by Zimmermann et al. who suggests integrating prior knowledge of the dynamic system into the neural network model. [GROT04]

3.8 Neural Networks Applications

At present, neural networks are used to perform tasks in several different areas, some of which are: medicine (e.g. diagnoses), speech and pattern recognition, data mining, composing music, signal and image processing, forecasting, control (e.g. robot), credit approval, classification, planning game strategies, and compression [ENGE02, FAUS94]. Some of these are briefly described below. For a more detailed information; see e.g. Haykin (1994) [HAYK94] and Fausett (1994) [FAUS94].

- Speech Recognition

  Understanding speech is a very difficult task for a computer system, this due to the fact that pronunciation, moods and personality varies heavily [HAYK94, FAUS94]. Still, that very same task is performed easily and automatically by the human brain [HAYK94]. Thus an effective method for building computer systems that can recognize speech is to use neural networks, e.g. Kohonen's self-organizing maps [HAYK94, FAUS94].

- Optical Character Recognition

  Optical character recognition (OCR) is a technique where a computer system translates handwritten text into digital characters [HAYK94]. The recognition of handwritten text is a pattern recognition problem, for which neural networks are very suitable [HAYK94, FAUS94].

- Medicine

  Neural networks can be used for medical purposes, e.g. diagnosis and treatment suggestions. Input to the neural network can be symptoms, from which the neural network makes a diagnosis and suggests a treatment. [FAUS94]

- Control

  One possible application of neural networks are as control systems, for example in robots or as an assistant to a driver of a docking trailer [FAUS94].

- Signal Processing

  An example of signal processing where neural networks can be used is noise reduction on a telephone line for long distant calls. This was in fact one of the first public application of neural networks. [FAUS94]

Chapter 4

Error Correction Neural Networks

The error correction neural network, developed by Zimmermann et al., introduces prior knowledge into neural networks. This in order to reduce the problem of overfitting when forecasting financial markets (see Section 3.5). This is accomplished partly by basing the error correction network on a time-delayed recurrent neural network (see Section 3.2.4) which, in addition to the external data also considers the previous states of the model in order to derive the output. [GROT04]

Thus, opposite to the more common multi-layered perceptron (see Section 3.2.1) which has a pattern recognition approach, the error correction network instead views the forecasting problem as a system identification problem, where the task is to find the dynamic system that best explains the data. Thus, the prior knowledge introduced into the model is that the financial markets can be seen as dynamical systems, with a large autonomous part (i.e. dynamics not driven by external influences). In addition to this the error correction neural network uses the last model error as an additional input to the network, which serves as an indicator of the correctness of the model. This means that the error information can be used to guide the model dynamics and thus preventing the autonomous part of the model to learn false temporal relations. [GROT04]

In order to solve the credit assignment problem, the error correction neural network is unfolded (see Section 4.1), after which the network can be trained using the error back-propagation algorithm with a shared weights extension (see Section 3.3.3). In addition to this, the unfolding introduces a temporal structure into the network architecture.

The technique of overshooting (see Section 4.2) is also introduced as a way to force the network to concentrate on the autoregressive parts of the dynamics in the data during training. [GROT04]

For more information concerning the error correction neural network, see e.g. Zimmermann et al. (2000) [ZIMM00] and Grothmann (2004) [GROT04].

4.1 Finite Unfolding in Time

Finite unfolding in time is an architectural solution to the credit assignment problem (see Section 3.3.3) for time-delayed recurrent neural networks (see Section 3.2.4) and was developed

by Rumelhart et al. The solution has its basis in the idea that any recurrent neural network can be represented by an identically behaving feed-forward network. [GROT04]

The time-delayed recurrent neural network (see Figure 3.10) is transformed into a feed-forward network by unfolding it for a finite amount of time steps using shared weights, which is best illustrated in Figure 4.1. [GROT04, ZIMM00]

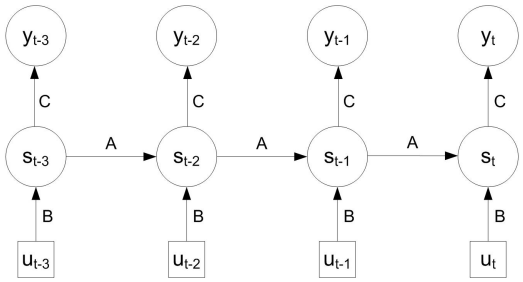

Figure 4.1: Comparing with Figure 3.10 at time step t, the recurrent loop is replaced with a state cluster (with input and output) that represents the previous state (s_{t-1}). This is repeated for a finite number of time steps using the shared weights A, B and C. [GROT04, ZIMM00]

Thus finite unfolding in time can be used to represent a time-delayed recurrent neural network as a feed-forward network, where the credit assignment problem can be solved using the error back-propagation algorithm with a shared weights extension (see Section 3.3.3) [GROT04].

The shared weights means that the weight matrices A, B and C (see Figure 4.1) are the same at all occurrences in the unfolded network (i.e. they share the same memory). This means that the unfolded time-delayed network will have the same amount of free weights as the original recurrent time-delayed network. This makes the unfolded network more resistant to the problem of overfitting (see Section 3.5) than many of the other feed-forward networks. [GROT04, ZIMM00]

A major issue when unfolding the time-delayed neural network is the decision of how many time steps that should be present in the resulting feed-forward network. This issue is discussed in the following section.

4.1.1 Maximum Inter-temporal Connectivity (MIC)

Grothmann (2004) suggests using the maximum inter-temporal connectivity when determining for how many time steps a time-delayed recurrent neural network should be unfolded. The maximum inter-temporal connectivity denotes how many historic values that contributes to the estimation of the output value at the current time (t). [GROT04]

Suppose that a time-delayed neural network is unfolded for the time steps $t-\tau,...,t$, then the individual error for each of these time steps can be observed. Usually the error decreases from the most historic time step until some minimum has been reached, which is illustrated in Figure 4.2. [GROT04]

As illustrated in Figure 4.2, the MIC can be determined by first unfolding the network a number of time steps such that the minimum error occurs before the current time step (i.e. e_{min}

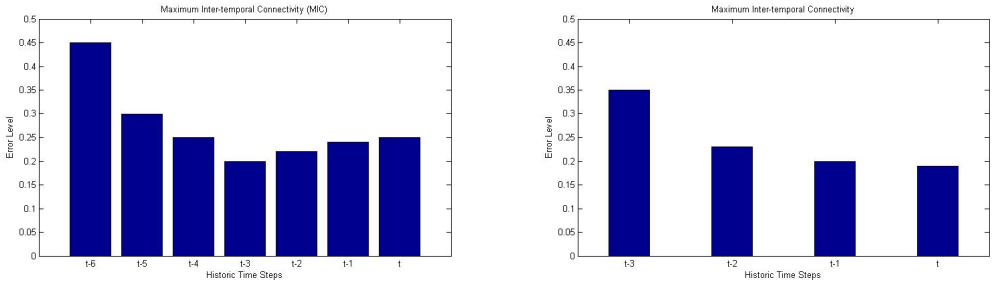

Figure 4.2: Left: The individual error for each time step can be seen, where the leftmost error is the most historic step. It can be seen that the error level decreases from the most historic value at time step $t-6$ until a minimum error has been reached at time step $t-3$. Thus the MIC is four time steps. Right: Here the error decreases from the most historic time step $t-3$ until the minimum has been reached at time step t. The MIC can not be determined based on this graph alone sice the minimum occurs at time t (i.e. either the MIC is four or greater than four). If it is assumed that both graphs represents the same network, the right graph will also have a MIC of four and thus a correct length of the unfolding.[GROT04]

such that $min < t$). The MIC will then be the number of time steps between the most historic value and the time step where the minimum error occurs (i.e. $\tau - min + 1$). When the MIC has been determined, the final network is unfolded with the number of historic time steps that was determined by the MIC. [GROT04]

4.2 Overshooting

A problem when modeling dynamical systems with the unfolded time-delayed neural network (see Section 4.1) is that the error back-propagation algorithm (see Section 3.3.3) tends to focus on the relations between the most recent data and the output (e.g. u_t and y_t, see Figure 4.3). This means that the unfolded network is biased towards learning the parts of the dynamics that are dependent on the most recent data. This problem can be solved using overshooting, which forces the network to learn autonomous dynamics and relations between older data and the output (e.g. u_{t-3} and y_t, see Figure 4.3). [GROT04]

Overshooting is added to the unfolded time-delayed network through the addition of one or more future predictions, as illustrated in Figure 4.3. The overshooting has the effect that more useful error information can be propagated back to the network, while the number of free weights remains unchanged (since the overshooting part of the network uses the same shared weights as the original unfolded network). In addition to this, if the extra predictions achieves a low error during training, more useful information is generated as output from the neural network. Interesting to note is that, if the network is able to learn the additional future prediction (y_{t+k}), the error levels for the preceding predictions ($y_t, ..., y_{t-k-1}$) usually also decreases. [GROT04, ZIMM00]

It is obvious that the extension into the future is limited by the maximum inter-temporal connectivity (see Section 4.1.1). Grothmann (2004) suggest using the following procedure to

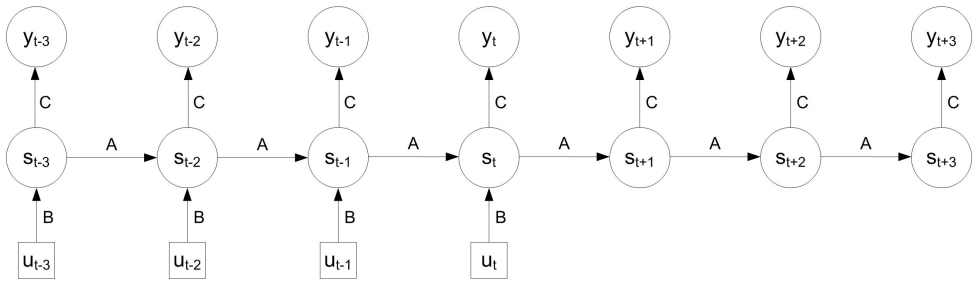

Figure 4.3: Overshooting is added to the unfolded time-delayed neural network through additional future predictions, in this case three. The overshooting part of the network receives no input and uses the shared weights (A and C) of the original unfolded network. [GROT04, ZIMM00]

determine the amount of future outputs that the overshooting should encompass: If the neural network shows signs of overfitting (see Section 3.5) when trained to convergence, an additional future prediction (y_{t+k}) is added, provided that the MIC is not exceeded. The procedure is then repeated until the MIC is reached, the network is unable to learn the new future prediction or the network error starts to increase. [GROT04]

4.3 Error Correction Neural Networks (ECNN)

The error correction neural networks approaches the forecasting problem by viewing it as the task of identifying the dynamic system that best describes the data. For this reason, the error correction network is based on the time-delayed recurrent network. Thus a large portion of the discussion in this section is based on previous discussions relating to the time-delayed recurrent network in Section 3.2.4. It is strongly recommended to review the information relating to the time-delayed recurrent network before continuing with this section. [GROT04]

It was previously determined that the Equations 4.1 and 4.2 can be used to describe a dynamic system recurrently. [GROT04, ZIMM00]

$$s_t = f(s_{t-1}, u_t) \tag{4.1}$$

$$y_t = g(s_t) \tag{4.2}$$

In addition to viewing the forecasting problem as the task of finding a dynamic system, the error correction neural network uses the last modeled error ($y_{t-1} - y^d_{t-1}$) as an additional input to the model, thus the Equations 4.1 and 4.2 are revised to include the error in Equations 4.3 and 4.4. [GROT04, ZIMM00]

$$s_t = f\left(s_{t-1}, u_t, y_{t-1} - y^d_{t-1}\right) \tag{4.3}$$

$$y_t = g(s_t) \tag{4.4}$$

Following the discussion concerning time-delayed neural networks, the task of identifying the dynamic system that best describes the data can be accomplished using neural networks, which results in the Equations 4.5 and 4.6. [GROT04, ZIMM00]

$$s_t = \tanh\left(As_{t-1} + Bu_t + D\left(Cs_{t-1} - y^d_{t-1}\right)\right) \tag{4.5}$$

$$y_t = Cs_t \tag{4.6}$$

There are however some issues in the neural network described by the Equations 4.5 and 4.6, which results in numerical problems. This since the relation between s_t and s_{t-1} in Equation 4.5 can be coded in both matrix A and DC. In order to solve this issue, Grothmann (2004) suggests using the hyperbolic tangent function (see Section 3.1.1). In addition to solving the numerical issue, the hyperbolic tangent function also scales the error information so that it varies around zero in the range $(-1,1)$, which is suitable for neural networks that uses the hyperbolic tangent activation function. Thus the final set of equations that describes the error correction neural network can be seen in Equations 4.7 and 4.8. [GROT04]

$$s_t = \tanh\left(As_{t-1} + Bu_t + D\tanh\left(Cs_{t-1} - y^d_{t-1}\right)\right) \tag{4.7}$$

$$y_t = Cs_t \tag{4.8}$$

The error correction neural network is then unfolded following the procedure described in Section 4.1. The resulting network architecture can be seen in Figure 4.4. [GROT04, ZIMM00]

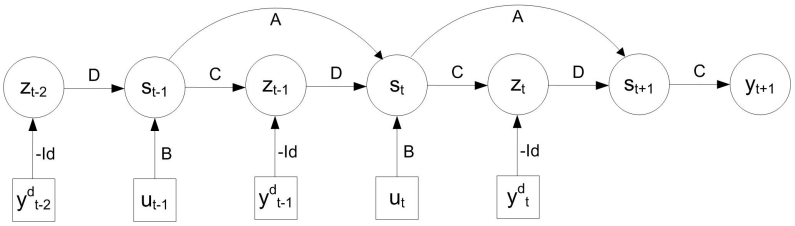

Figure 4.4: This figure illustrates the unfolded error correction neural network where the weight matrices A, B, C and D are shared weights and $-Id$ is the negative identity matrix [GROT04, ZIMM00].

Finally the technique of overshooting is applied to the error correction neural network, using the method described in Section 4.2. The resulting network architecture can be seen in Figure 4.5. Overshooting has the additional advantage, when used with error correction neural networks, of forcing the network to concentrate on the autoregressive parts of the data. [GROT04, ZIMM00]

Refereing to Figure 4.5, five different types of clusters can be observed; state clusters s_t, input clusters u_t, target input clusters y^d_t, error output clusters z_t and output clusters y_t. These clusters are connected to each other using the shared weights A, B, C, D and the negative

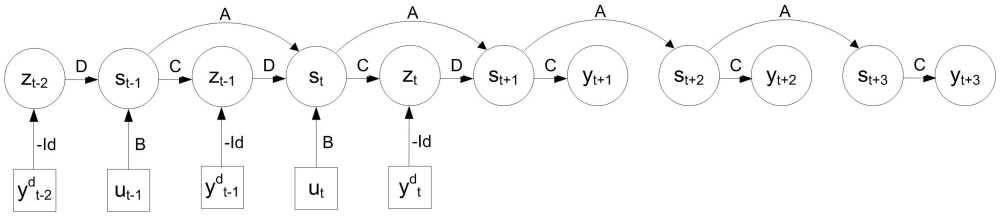

Figure 4.5: This figure illustrates the unfolded error correction neural network with a overshooting part [GROT04, ZIMM00].

identity matrix $-Id$. Each of these clusters contains a number of neurons, which is determined by the number of inputs (clusters u_t), the number of targets (clusters y_t, y_t^d and z_t) or defined when designing the model (clusters s_t). The different clusters can also be traced back to different parts of Equations 4.7 and 4.8, which is illustrated in Equations 4.9, 4.10 and 4.11. [GROT04, ZIMM00]

$$y_t = Cs_t \tag{4.9}$$

$$z_t = \tanh\left(Cs_{t-1} - y_{t-1}^d\right) \tag{4.10}$$

$$s_t = \tanh\left(As_{t-1} + Bu_t + Dz_t\right) \tag{4.11}$$

From Equations 4.9, 4.10 and 4.11, it can easily be seen that the output neurons have a linear activation function, while the error output neurons and the state neurons have the hyperbolic tangent activation function (the input nodes have no activation functions). [GROT04, ZIMM00]

Both the error output neurons and the output neurons produces the network's output values, which are used by the error correction learning rule (see Section 3.3.2) in order to determine the optimum weights during training. The output neurons have desired target values that are determined by the target series that is to be forecast. Since the error output neurons have the previous error, squashed by the hyperbolic tangent function, as output (see Equation 4.10), its target value is obviously zero (i.e. the desired error is zero). [GROT04, ZIMM00]

Since shared weights are used for the unfolded and overshooting parts of the network, the error correction neural network has relatively few free weights. The additional network outputs (from the unfolding and overshooting) also produces more error information during training (i.e. more information is available when determining the optimum weight values). This makes the error correction neural network very resilient to the problem of overfitting compared to other feed-forward networks. The shared weights also requires that a shared weight extension of the error back-propagation algorithm (see Section 3.3.3) is used during training. [GROT04, ZIMM00]

It was previously discussed that the state and error output neurons uses the hyperbolic tangent activation function, which means that the error correction neural network will contain long chains of the hyperbolic tangent function. As discussed in Section 3.1.1, this can cause numerical problems when using the error back-propagation algorithm. This since each pass through the derivative of the hyperbolic tangent function may shrink the error information.

This problem can however be avoided through re-normalization of the gradient information after each step of the back-propagation. Thus the vario-eta optimization algorithm is very suitable to use with the error correction neural network, since it is able to perform this re-normalization. [GROT04]

The error correction network has some additional advantages, which includes handling missing values in the target series by automatically generating replacement values. The error correction mechanism also enables the network to handle the initializing phase (when there is no accumulated memory in the historic parts of the network). [GROT04, ZIMM00]

In addition to this, since the architecture of the error correction network has a temporal structure, there is no need for complicated preprocessing of the input data. The relatively simple scaling and transformation using scaled momentum and scaled force (see Section 2.6) are able to capture the dynamics of the system in a way that is suitable for the error correction networks. This can be compared to other feed-forward networks which usually requires complicated preprocessing procedures in order to capture the temporal characteristics of the data. [GROT04]

4.3.1 Extensions to ECNN

There exists several extensions to the error correction neural network, a few of which can be seen in the list below. For more extensive information see e.g. Zimmermann et al. (2000) [ZIMM00] or Grothmann (2004) [GROT04].

- Variants-Invariants Separation

  When forecasting high dimensional systems (i.e. several target series), the task can be simplified by separating the variants and invariants using an auto-associative network (see Sections 2.7 and 3.2.5). A principle component analysis is performed in order to separate the variants and invariants, thus reducing the dimensionality of the system that needs to be forecasted (since the invariants are constant through time and thus needs no prediction). The predicted value is then recombined with the invariants to form the final forecast of the model.

- State Space Reconstruction

  The aim of this extension of the error correction neural network is to force the state transitions to evolve more smoothly using a penalty function (and a specialized network architecture). In addition to a more smooth transition of the states, there also is an improvement of velocity control and noise reduction. [GROT04]

- Undershooting

  Undershooting can be used when there is a need to have models with a finer time grid then the supplied data (i.e. the models uses shorter time periods than the supplied data). Using undershooting can provide a deeper understanding of the dynamics of the system that is to be identified by the ECNN. [GROT04]

- Alternating Errors

  The idea of the alternating error extension to the error correction neural network is to avoid trend following behavior. This since models usually tends to overestimate downward trends, underestimate upward trends and detects trend reversals to late. [GROT04]

Chapter 5

Evaluation

When creating a forecast model, a very important step is to evaluate its prediction performance. This can be accomplished through the use of performance measures, e.g. hit rate and return on investment (see Section 5.2). However, in order for these measures to make any sense, there is also a need to define what a good performance measure is. This can be accomplished through the use of benchmarks, e.g. naive prediction, linear regression models or indexes (see Section 5.3). When using benchmarks, the performance measures of the forecast model are compared with a suitable measure of a benchmark.

In addition to evaluating the model through the use of performance measures and benchmarks, the forecast model and evaluation method should also be inspected for correctness and obvious mistakes, as discussed in Section 5.1.

5.1 Correctness of Model and Evaluation

A first step when evaluating a forecast model is to make sure that no obvious mistakes have been made, some of the more common problems can be seen in the list below.

- Future observations:

  A fairly common mistake when creating forecast models is the inclusion of future observations as input to the model (also known as 'off-by-one' errors). This is a devastating problem since the future, which is to be predicted, obviously is unknown. [HEL98a]

- Data snooping:

  Another major problem is data snooping, which denotes a situation where the generalization set (see Section 2.8) is used before the model evaluation phase. This occurs if the data in the generalization set is used during, or affects the e.g. input variable selection, model selection or model training. [HEL98a]

- Forecasting future:

  When forecasting financial time series, it is important to note that as the forecasts moves further into the future, the prediction performance often worsens. Thus it might be a

good idea to divide the generalization set (see Section 2.8) into several smaller sets, spanning e.g. a year or a month. The model can then be evaluated with respect to each of these smaller sets, giving the ability to detect how far into the future the forecasting model generates reliable predictions. [HEL98a]

- Several forecasts:

  When evaluating a prediction model, it should be based on as many forecasts as possible. This since it is obvious that an evaluation based on several forecasts will produce more reliable results than if only one or a few forecasts are used. [HEL98a]

- Overfitting:

  In the field of forecasting using neural networks the problem of overfitting is well known (see Section 3.5). When using neural networks as forecast models, there usually is a great deal of effort made to avoid overfitting. Thus the forecast models tendency to overfit might be considered a performance measure.

  Overfitting can be detected by observing the error on the training set and the validation set during the training of the network. If the error starts to increase on the validation set at the same time as it decreases for the training set, it is usually a sign that overfitting has occurred.

- Predicting total random walk series:

  One way to test the forecasting model and evaluation method is to try and predict a random walk process. This since successful predictions of a random walk process indicates that there is something wrong with the forecasting model or evaluation method. [HEL98a]

5.2 Performance Measures

There exists a large number of different performance measures, e.g. hitrate, Sharpe ratio, return on investment, realized potential and annualized return [GROT04, HEL98a]. Some of these performance measures are described in the following sections.

5.2.1 Root Mean Squared Error

The root mean squared error (see Equation 5.1 [HEL98a]) can be used as a measure of the average error between the predicted value and the actual value.

$$RMSE = \sqrt{\frac{1}{T}\sum_{t=1}^{T}\left(y_t - y_t^d\right)^2} \tag{5.1}$$

5.2.2 Hit Rate

The hit rate is a measure of the directional correctness of the predictions, which is defined as the percentage of predictions that are correct in sign [HEL98a]. This ratio is calculated as the number of correct non-zero predictions divided by the total number of non-zero predictions and can be seen in Equation 5.2 [HEL98c]. A modified version of the hitrate can be seen in Equation 5.3, which counts zero values as hits.

$$HR = \frac{|\left\{t|y_t y_t^d > 0, t = 1,...,N\right\}|}{|\left\{t|y_t y_t^d \neq 0, t = 1,...,N\right\}|} \tag{5.2}$$

$$HR = \frac{|\left\{t|y_t y_t^d \geq 0, t = 1,...,N\right\}|}{|\left\{t|t = 1,...,N\right\}|} \tag{5.3}$$

5.2.3 Trading Strategy

A trading strategy is a rule that decides when to invest and when not to invest in an asset. There are two simple strategies that can be used in different performance measures, e.g. return on investment and realized potential. Notice that the following trading rules assumes that the predicted time series is a return series.

The first trading rule can be seen in Equation 5.4, where a one represents a buy signal and a zero a sell signal [HEL98a]. If the forecast model predicts a positive change (e.g. positive return) a buy signal is issued and if a negative change (e.g. negative return) is predicted, a sell signal is issued. If no change is predicted (i.e. $y_t = 0$) the previous signal remains unchanged (i.e. no action is taken).

$$T_t^g = \begin{cases} 1 & \text{if } y_t > 0 \\ T_{t-1}^g & \text{if } y_t = 0 \\ 0 & \text{if } y_t < 0 \end{cases} \tag{5.4}$$

The second trading rule can be seen in Equation 5.5, where a one represents a long position and a minus one represents a short position [GROT04, SIEK99]. The long position is taken when the the prediction indicates a positive change (e.g. positive return) and a short position when the prediction indicates a negative change (e.g. negative return) [GROT04]. If no change is predicted (i.e. $y_t = 0$) the previous signal remains unchanged (i.e. no action is taken).

$$T_t^{ls} = \begin{cases} 1 & \text{if } y_t > 0 \\ T_{t-1}^{ls} & \text{if } y_t = 0 \\ -1 & \text{if } y_t < 0 \end{cases} \tag{5.5}$$

The trading rules can be expanded to also consider trading costs, which is an administrative cost that occurs when an asset is bought (and possible also when sold). Note that a trading cost can only occur when there is a change in the trading decision which results in an action, for example from a sell to buy signal or a long to short position and not e.g. a buy to buy signal.

5.2.4 Return on Investment

The return on investment (ROI), also known as the accumulated return on investment, not only considers the directional correctness of the prediction but also the impact it will have in terms of return (i.e. it also considers the quantity of the return) [GROT04, NYGR04]. The ROI, as seen in Equation 5.6, uses a trading rule to take long and short positions based on the predictions (i.e. the directional correctness of the prediction) and the simple return (see Section 2.4.1) of the target series to determine the impact of this prediction [SIEK99].

$$ROI = \sum_{t=1}^{T} \left(\frac{y_t^d}{y_{t-1}^d} - 1 \right) T_t^{ls} \tag{5.6}$$

5.2.5 Realized Potential

The realized potential, as seen in Equation 5.7, calculates the ratio between the accumulated return on investment and the maximum possible accumulated return on investment [GROT04, NYGR04].

$$RP = \frac{\sum_{t=1}^{T} \left(\frac{y_t^d}{y_{t-1}^d} - 1 \right) T_t^{ls}}{\sum_{t=1}^{T} \left| \frac{y_t^d}{y_{t-1}^d} - 1 \right|} \tag{5.7}$$

5.2.6 Gross Return

The gross return, see Equation 5.8, can be used together with a trading rule based on the predictions of the forecast model (or benchmark) to calculate how much profit (or loss) the model generates. [TSAY05]

$$R^G = \left[\prod_{t=1}^{T} \left(\frac{y_t^d}{y_{t-1}^d} \right) T_t^g \right] - 1 \tag{5.8}$$

5.2.7 Annualized Return

The annualized return can be used to compute the average return during a year, based on the gross return, enabling easier comparison between forecasts with different time periods (i.e. when and for how long a stock is forecasted). Equation 5.9, can be used to compute the annualized return, where k denotes the number of years that are forecasted. [TSAY05]

$$AR = \left[\prod_{t=1}^{T} \left(\frac{y_t^d}{y_{t-1}^d} \right) T_t^g \right]^{1/k} - 1 \tag{5.9}$$

5.3 Benchmarks

When evaluating a forecast model, performance measures are usually not enough. This since a forecast model with low performance measures can still be the best available alternative (i.e. even if it loses money, it might lose less than the other alternatives). Thus by comparing the performance measures of the prediction model with benchmarks, the evaluation becomes more meaningful. [HEL98a]

An example of an easy to use benchmark is the naive prediction, where a forecast is based on the previous value (of the time series, see Section 5.3.1) [MAKR98]. More advanced prediction models can also be used as benchmarks, e.g. linear regression models or multi-layered perceptron networks (see Section 3.2.1). Another possibility is to use a suitable index as a benchmark [BAIN96].

Obviously, when comparing a forecast model with a benchmark, it is very important that the performance measures are based on the same time periods (i.e. the same generalization set) and have the same scale. In addition to this, when computing a performance measure that requires a trading strategy for a index benchmark, the buy-and-hold strategy is suitable to use.

5.3.1 Naive Prediction

The naive prediction is a trivial benchmark that predicts the next value to be the same as the current value, as can be seen in Equation 5.10 [HEL98a].

$$y_{t+h} = y_t \quad h \geq 1 \tag{5.10}$$

Since the stock prices are commonly considered to be close to a random walk process, the naive predictions can be seen as a best estimate of the next price. Hellström (1998) also proposes to use the naive prediction on stock return series. [HEL98a]

5.4 Comparison of Performance Measures

When comparing the hitrate or the root mean squared error of a forecast model with a benchmark, it can be helpful to calculate the quotient between them, as done in Equations 5.11 and 5.12 [HEL98a].

$$HR_q = \frac{HR_F}{HR_B} \tag{5.11}$$

$$RMSE_q = \frac{RMSE_F}{RMSE_B} \tag{5.12}$$

When using the return on investment or gross return as performance measures of a forecast model and a benchmark (e.g. an index or another forecast model), it might be a good idea to compare them by plotting them in a graph (see Figure 5.1).

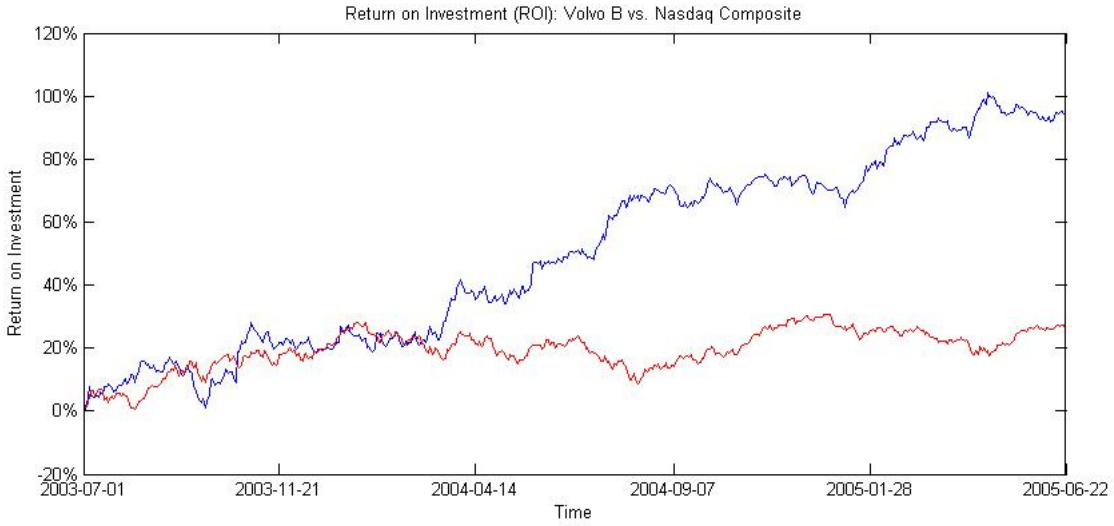

Figure 5.1: This figure shows an example of how the return on investment can be compared to an index when using a graph. This example compares the return on investment for a forecast of the Volvo B stock with the Nasdaq Composite index.

Chapter 6

Empirical Study

This chapter contains a number of tests aimed at investigating different techniques, such as cleaning with noise and missing value replacement, and different aspects of using forecasting models based on the error correction neural network, such as determining the number of neurons in the state clusters, unfolding and overshooting time steps. The results from these tests are also interpreted in terms of potential of the ECNN, giving an indication of what performance might be expected when using these networks in forecast models.

The chapter starts with a brief description of different characteristics of the stock data, the SENN application (used to train the networks) and the Forecast Model Generator (used to generate and evaluate the forecast models). In addition to this, a general test method that describes the key aspects of the test method used when performing the different tests, can be found.

Finally, each test has a detailed description of the method used, results that were obtained and conclusions that could be drawn.

6.1 Stock Data

The data, used when performing the tests in this thesis, is available in three different databases and includes e.g. stocks, indexes, interest rates, foreign exchange rates and raw material costs etc., mainly from the Swedish market. Each database entry includes, when applicable, information concerning closing, opening, high and low prices and volume. The quality of the data has also been examined, including making sure that the inequality $high \geq close \geq low$ is fulfilled when possible.

These three databases have been inserted into one single database, in which each entry (i.e. stocks, indexes etc.) includes between one and three tables. The first table contains the original data, the second table contains additional data and the third table contains merged data. The third table was created, when an entry was present in more than one database, had overlapping data and a high cross-correlation, through merging of the two tables (i.e. one table from each entry). The second table was used, if the entry existed in more than one database but had no overlapping data (or a low cross-correlation), to store the additional data. The final database, containing the information from the three original databases, is the only one that is

used in this empirical study.

An investigation has also been performed on approximately 30 entries in the database, checking the rate of missing values and the cross-correlation of the different time series (i.e. the open, high, low, close and volume series) that are present. This investigation indicates that the open price series (P_o) have a large rate of missing values, while it varies a lot between different entries for the volume series (V). The high (P_h), low (P_l) and closing (P) price series on the other hand seems to have a low rate of missing values and a high cross-correlation to each other. The cross-correlation between the open (P_o) and the other series were low, while the cross-correlation between the volume (V) series and the other series were even lower. The mean of the cross-correlation for all the entries in the database can be seen in Table 6.1. A correlation check between A and B shares of stocks from the same company indicates a high cross-correlation between these.

Mean Internal Cross-Correlation					
	Open:	High:	Low:	Close:	Volume:
Open:	1.0000	0.5716	0.5749	0.5732	0.0932
High:	0.5716	1.0000	0.9879	0.9918	0.0811
Low:	0.5749	0.9879	1.0000	0.9953	0.0611
Close:	0.5732	0.9918	0.9953	1.0000	0.0739
Volume:	0.0932	0.0811	0.0611	0.0739	1.0000

Table 6.1: This table shows the mean values of the internal cross-correlation of all entries in the database.

This indicates that the open price series should be avoided due to their high rate of missing values and that the rate of missing values should be checked carefully for the volume series before using them as input to forecast models. In addition to this, the high cross-correlation between the high, low and close series suggests that only one of these, for each stock, should be used as input.

6.2 SENN

The SENN (Simulation Environment for Neural Networks) software supplies an environment where artificial neural networks can be created for forecasting and classification applications. SENN was created by Siemens AG, Germany, for more information regarding the SENN application, see the SENN user manual, Tietz (2008). [TIET08]

SENN supports a variety of different network types, e.g. multi-layered perceptron (MLP) networks, radial basis function (RBF) networks, etc. In addition to this it has a wide range of available optimization algorithms, e.g. vario-eta, gradient descent, conjugate gradient, etc., which can be used during the training of neural networks. [TIET08]

Each model is specified using two text files, the specification file and the topology file. In addition to these two files, additional files can be used to supply the data, initial weight values and settings. The specification file supplies the settings of the input and target data, including preprocessing and output signals. The topology file specifies the network architecture of the

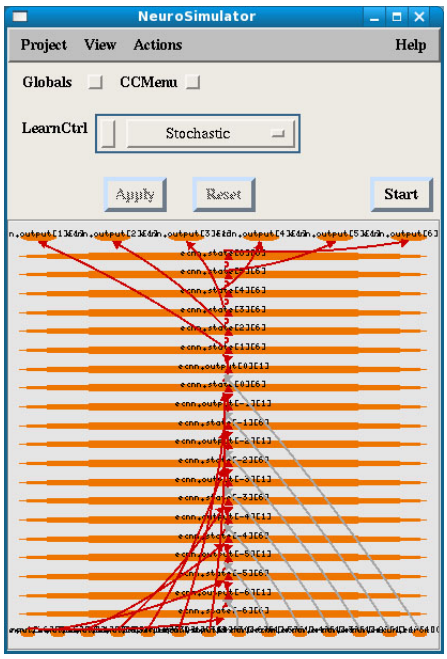

Figure 6.1: This figure shows the main window of the SENN application.

model, such as neurons, cluster of neurons and how these are connected to each other using synapses. It also contains a configuration portion in which a subset of the SENN application settings can be modified, e.g. initial weight values, neuron activation functions, optimization algorithm etc. [TIET08]

The specification and topology files can both be created using a text editor, after which they can be loaded into the SENN application where the network is trained and evaluated using its graphical user interface. [TIET08]

Step Length Control

The SENN application has a functionality that allows the user to reduce the step size automatically during training. In addition to the initial step length, a reduction factor and an interval are supplied. The interval specifies the number of epochs that passes between each reduction of the step length. The new step length is calculated by multiplying the old step length with the reduction factor. [TIET08]

Signals

Signals can be defined in the specification file in order to simplify the evaluation of neural network models. These signals can be configured to calculate performance measures based on the network output. Thus more information, in addition to the network error levels, can be extracted during a training session. As an example, a signal can be formulated to calculate the hitrate according to Equation 5.3. [TIET08]

Output: Graphical and Files

The error levels, different signals, neuron outputs, etc. can be displayed graphically for the training, validation and generalization sets during a training session. These results can also be extracted to tabulated text files using the following menu commands: [TIET08]

- Print to file

  The results that are displayed graphically in a window can also be extracted to a file using this menu command.

- Protocol

  Using this command during a training session, the results displayed graphically can be extracted to a text file.

- Save Weights

  The currently active weights in a neural network can be stored (and loaded later) to a text file.

6.3 Forecast Model Generator GUI

In order to facilitate easy set up of forecast models a number of functions were created, using MATLAB, with which preprocessing, selection of an ECNN architecture, generation of necessary files for the SENN application (see Section 6.2) and evaluation of forecasts can be performed. A graphical user interface (GUI) (see Figure 6.2) allows easy access to the more common functions used during the set up phase of the forecast model.

The first window of the GUI allows the user to select financial data from the database and then divide this data into training, validation and generalization sets. In the following window (see Figure 6.2), scaling and transformation functions, strategy to replace missing values and aggregation of data can be selected. In addition to this, information concerning input-input correlation, input-output correlation and quality of data can be extracted from the selected financial data. This information can be used when performing input variable selection, as described in Section 2.7.1. In the final window (see Figure 6.2), the configuration of the error correction neural network can be selected, including unfolding and overshooting time steps, number of neurons in the state clusters, target multiplier and whether to use the cleaning with noise technique or not.

When all selections have been made, the specification, topology and data files, used by the SENN application, can be generated. In addition to these files, a file with random weight values is generated, which SENN can use to initialize the network's weights.

In addition to generating the SENN files, there also exists functions that can be used to evaluate the forecasts, assuming that the result of the predictions has been extracted in ASCII format from the SENN application. The evaluation functions can be used to calculate and store (in ASCII, MATLAB binary or figure file formats), among others, the hitrate, return on investment and the realized potential. In addition to calculating these performance measures,

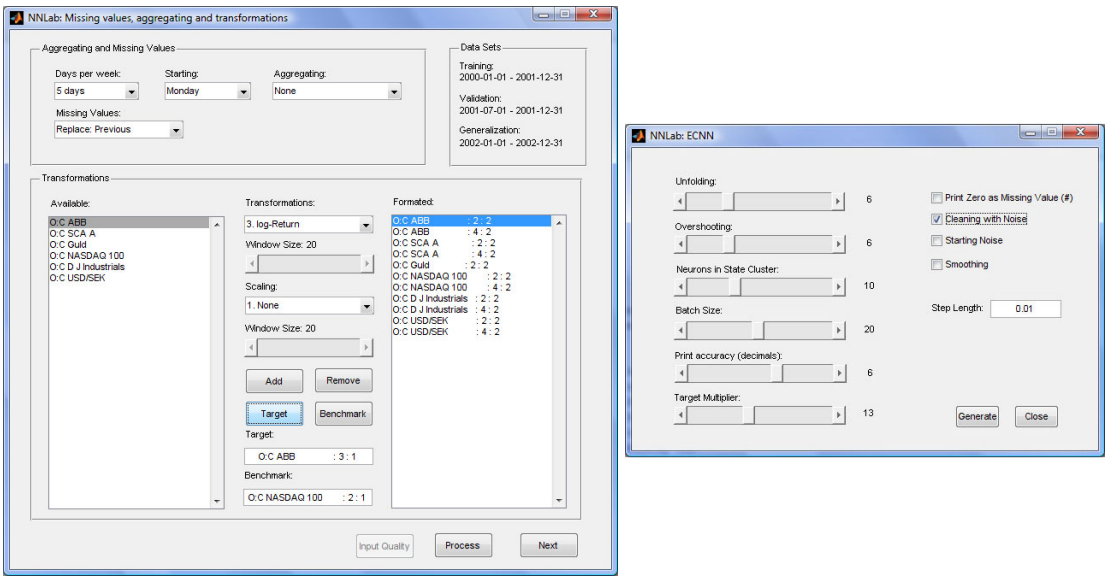

Figure 6.2: This figure shows the second and third window of the Forecast Model Generator GUI.

the evaluation functions can also be used to compare them with benchmarks (e.g. the naive prediction and indexes).

6.4 General Testing Method

The general test method described in this section will be applied to the different tests when applicable.

There are two major types of tests that are performed, one where the training lasts for 10 000 epochs and a second where a stopping criterion is used in order to determine for how many epochs the network should be trained. For all tests, the hitrate (see Equation 5.3) and error levels are extracted for each epoch during the training session, covering the training and validation sets. This allows the network to be examined for different effects, e.g. overfitting, both during and after completion of the training session. In addition to this, when the network is trained for the predetermined amount of 10 000 epochs, the hitrate and error levels are also extracted for each epoch covering the generalization set. This allows the potential of different network configurations, e.g. different number of neurons in the state clusters, different input data, etc., to be examined. The potential of an ECNN configuration is defined to be the highest hitrate, for the generalization set, achieved during a training session. Since the amount of epochs is known prior to the training of the network, knowledge of the generalization performance will not affect the training.

When several different network configurations are trained for the fixed amount of 10 000 epochs, their performances will not be affected by a stopping criterion, making it easier to compare the different network configurations. In addition to this, the performance during each epoch of the training, validation and generalization sets can be investigated in order to

find relations between these sets. This can be used to find a suitable stopping criterion and when examining how different network configurations affects the efficiency of this criterion.

In addition to this, the networks are trained for five separate training sessions using different initial weights. As a result of the multiple training sessions, the potential of the ECNN can be determined with a higher level of accuracy (using the mean and standard deviation). The effect of different initial weights in combination with the optimization algorithm's inability to find the same solution repeatedly should also be observable (see Section 3.3.5).

When discussing the general and selected architectures in the following tests, the general architecture refers to an error correction network with seven unfolding and six overshooting time steps. The selected architectures refers to when the number of unfolding and overshooting time steps have been determined using the maximum inter-temporal connectivity and error levels of future predictions, as described in Sections 4.1 and 4.2.

The stocks that are forecasted during these tests are the Volvo B, Scania B and SE Banken A stocks, each divided into three sets as can be seen in Table 6.2. The input variables used when forecasting the different stocks, and how they are scaled and transformed, can be seen in Appendix D. The target series is transformed and scaled using the log-return function and a target multiplier. The target multiplier is a term used to denote a constant factor with which the values in the target series are multiplied in order to increase its range.

Training, Validation and Generalization Sets						
	Training		Validation		Generalization	
	Starting	Ending	Starting	Ending	Starting	Ending
Volvo B	2001-02-01	2003-06-30	2003-02-01	2003-06-30	2003-07-01	2005-06-30
Volvo B*	2001-02-01	2003-12-31	2003-02-01	2003-12-31	2004-01-01	2005-06-30
Scania B	2001-02-01	2003-06-30	2003-02-01	2003-06-30	2003-07-01	2005-06-30
SE Banken A	1990-01-01	1992-06-30	1992-01-01	1992-06-30	1992-07-01	1994-06-30

Table 6.2: This table shows the time spans that are covered by the training, validation and generalization sets when forecasting the Volvo B, Scania B and SE Banken A stocks.

In addition to the performance measures mentioned in Chapter 5, three additional measures are used to characterize the performance of a stopping point. The first one, called stopping hitrate, denotes the mean hitrate of the generalization set for the ten closets epochs to the stopping epoch. This can be calculated as shown in Equation 6.1, where HR_s is the stopping hitrate, HR_e is the hitrate for epoch e and s is the stopping epoch.

$$HR_s = \frac{1}{11} \sum_{e=s-5}^{s+5} HR_e \tag{6.1}$$

The second measure is referred to as the gap, which can be seen as a benchmark and can be used to calculate the difference between the stopping hitrate and the potential achieved during a training session, in percent. Equation 6.2 shows how the gap can be calculated, where HR_p is the potential achieved during a training session.

$$G = \frac{HR_p - HR_s}{HR_p} \tag{6.2}$$

The final measure, the mean window hitrate (MWH) is used to denote the mean hitrate of the ten closest epochs to the epoch that has the highest hitrate on the generalization set. The MWH can be seen as a measure of difficulty of finding the epoch with the highest hitrate on the generalization set. The MWH is calculated using Equation 6.3, where p denotes the epoch that has the highest hitrate on the generalization set.

$$MWH = \frac{1}{11} \sum_{e=p-5}^{p+5} HR_e \tag{6.3}$$

Finally, a more complete description of the results obtained during the different tests can be found in Appendixes E through L.

6.5 Potential of ECNN and Training Procedure

The purpose of this test is to investigate the potential of the error correction neural networks and suitable configurations that may facilitate easier training. Thus the issues in the list below are examined more closely.

- Determining the number of neurons in the state clusters

  The ECNN has a number of state clusters (see Section 4.3) where the number of neurons needs to be determined. This will affect the overfitting behavior and the complexity of the network, in extension its ability to learn form the training data.

- Determining if the stopping criterion should be based on the hitrate or the error level

  During the training session different data are available, including hitrates and error levels. This data can be used during the training to determine a suitable stopping point. The choice of a stopping point can potentially have a large impact on the final result of a forecast.

- Determining if the stopping criterion should be based on the training set or validation set

  As mentioned in the previous item, different data is available during the training session. This data is available both for the training and validation sets. Thus there exists a need to determine which set that is most suitable to use when determining a stopping point.

- Investigate how the size of the validation set affects the training process

  If the stopping criterion is based on the validation set, it should be investigated how its size affects the ability to find a suitable stopping point. It could be argued that a larger validation set better reflects the data in the generalization set. At the same time a larger validation set means that the training data is farther away in time from the data that is to be predicted, the generalization set.

- Investigate the potential of the ECNN

  The potential of the ECNN can be investigated by finding the maximum hitrate during a training session for the training, validation and generalization sets. These hitrates can be seen as an indicator of the networks' forecasting abilities, overfitting behavior and how they are affected by different initial weights.

- Investigate if the cleaning with noise technique improves the performance of ECNN

  If the network is prone to overfitting behavior, the technique of cleaning with noise (see Section 3.5.4) can be applied. The effect of this technique should be investigated both with respect to the network's ability to predict and to how it affects the stopping criterion.

Test Method

For the purpose of this investigation, the Swedish stocks Volvo B and Scania B are forecasted. The input variables are selected as described in Section 2.7.1 and the resulting long and short lists can be seen in Appendix D. The available data is divided into three sets, covering the time periods which can be seen in Table 6.2.

The input data are preprocessed by replacing missing values with the last existing previous value. The preprocessing also includes transformation and scaling of the raw time series using momentum and force as specified in Tables D.2 and D.4 (Volvo B and Scania B). The target series is transformed and scaled using the log-return function and a target multiplier, a constant factor of thirteen

The test includes a number of different network configurations, each using the general architecture, where the amount of neurons in the state clusters varies between 3, 6, 10, 15 and 20 neurons. In addition to this, the network configurations are also tested with and without the cleaning with noise technique.

The different network configurations are trained using the error correction learning rule with the vario-eta optimization algorithm and a step size of 0.01. Each network configuration is trained five separate times, with different initial weights, for 10 000 epochs.

During the training sessions, the hitrates and error levels are extracted, thus allowing the potential of the different ECNN configurations to be examined and to find a suitable stoping criterion.

Results

In Figure 6.3 the potential for the different configurations of the error correction neural network can be seen. Interesting to note is that the maximum hitrate for the training set increases with the number of neurons in the state cluster. This can be explained by the increase of complexity in the network as the number of neurons increases. This since more complex networks are able to extract more information from the training set than a network with a low complexity (see Section 3.4).

At the same time, looking at the validation and generalization sets, their maximum hitrates also starts to increase with the number of neurons, but only up to a point. This since the

network configurations with a low number of neurons have a too low level of complexity in order to extract all relevant information from the training data. Thus, looking at Figure 6.3, the network configurations with only three neurons have (in general) a too low level of complexity. When the number of neurons increases beyond the point where the maximum hitrates also increases, the network has reached a level of complexity that allows it to not only learn the relevant parts of the data, but also noise, which is known as overfitting. Also interesting to note is that, even though the hitrates starts to decrease for the validation and generalization sets, it still increases for the training set. Thus a high hitrate on the training set could be interpreted as a sign of overfitting behavior.

Since cleaning with noise is a technique that can be used to reduce the problem of overfitting, it might be expected that more complex networks can be used without encountering the problem of overfitting (compared to the same network configuration without cleaning with noise). It might also be expected that these networks would have a better generalization performance. However, observing the test data in Figure 6.3, no pattern can be found that leads to any conclusions regarding whether using the cleaning with noise technique results in a higher performance or not.

Observing the plots in Figure 6.3, it can be seen that the number of neurons that performs best on the validation set is close to the number of neurons that performs best on the generalization set. Since the performance on the generalization set is not available during the selection of the network architecture, it is a reasonable approach to use the number of neurons that have the highest performance on the validation set.

Observing Figure 6.4, the hitrate on the generalization set is shown for the different stopping criteria that are investigated, which can be seen in the list below.

- Stopping on the highest hitrate on the training set.
- Stopping on the lowest error level on the training set.
- Stopping on the highest hitrate on the validation set.
- Stopping on the lowest error level on the validation set.

In general, when viewing Figure 6.4, it can be seen that using the validation set when determining the best stopping point seems to be the best choice. When comparing using a stopping criterion based on the best hitrate or the best error level, using the best error level seems to lead to the highest performance. Thus a suitable stopping criterion is to stop the training session close to the epoch with the lowest error levels on the validation set.

In Table 6.3, the mean stopping hitrates for the different network configurations can be seen. It can also be seen that the network configurations using the optimum number of neurons in the state clusters, as determined previously, will not always lead to the highest hitrate when using a stopping criterion. However, the difference to the network configuration with the best result is relatively small. No obvious difference in the performance can be detected when using the stopping criterion, between network configurations using and not using the cleaning with noise technique.

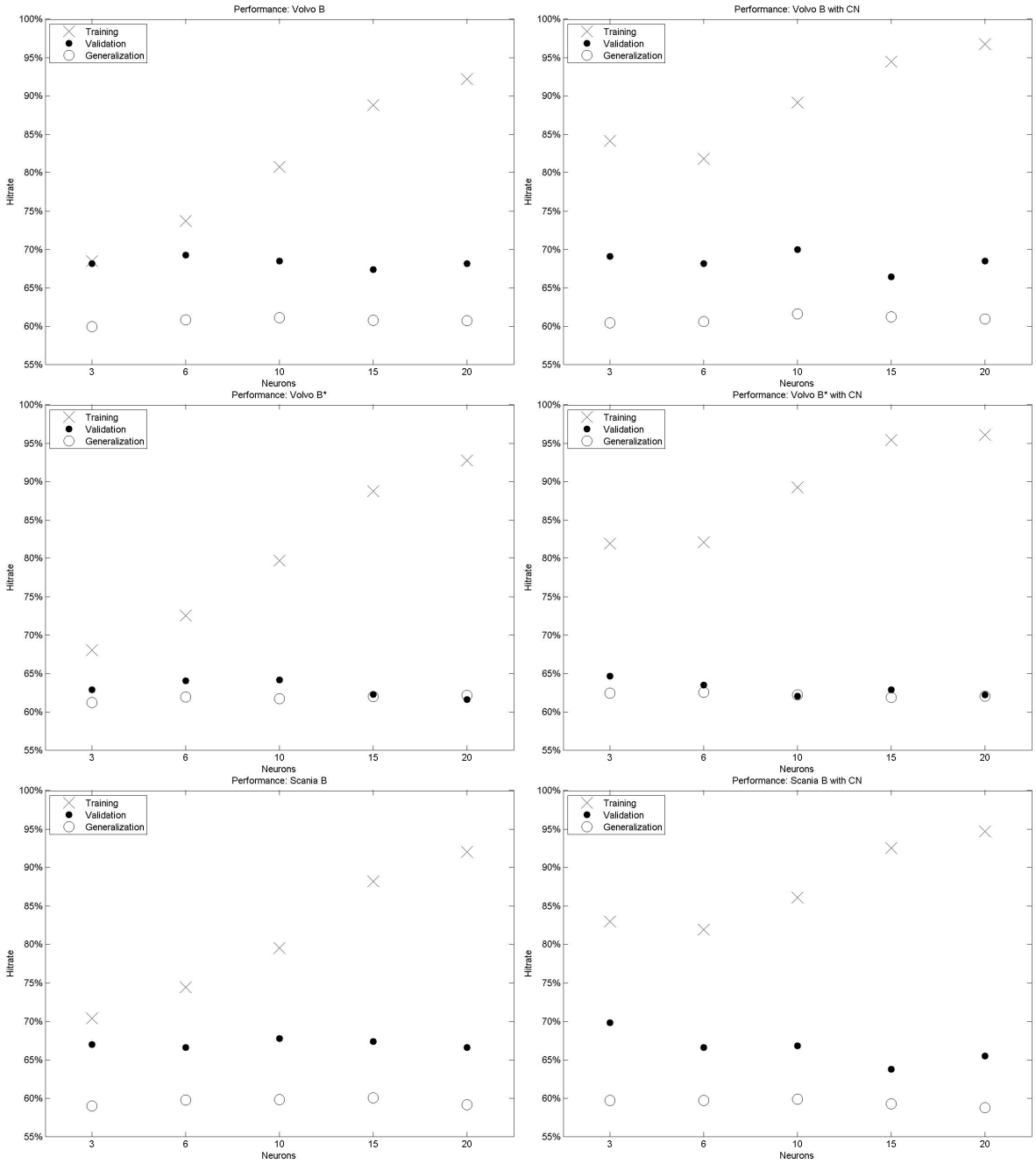

Figure 6.3: This figure shows the potential of the error correction neural networks. The left column shows the potential for the ECNN configurations without cleaning with noise, while the right column shows it with cleaning with noise. (CN: Cleaning with noise).

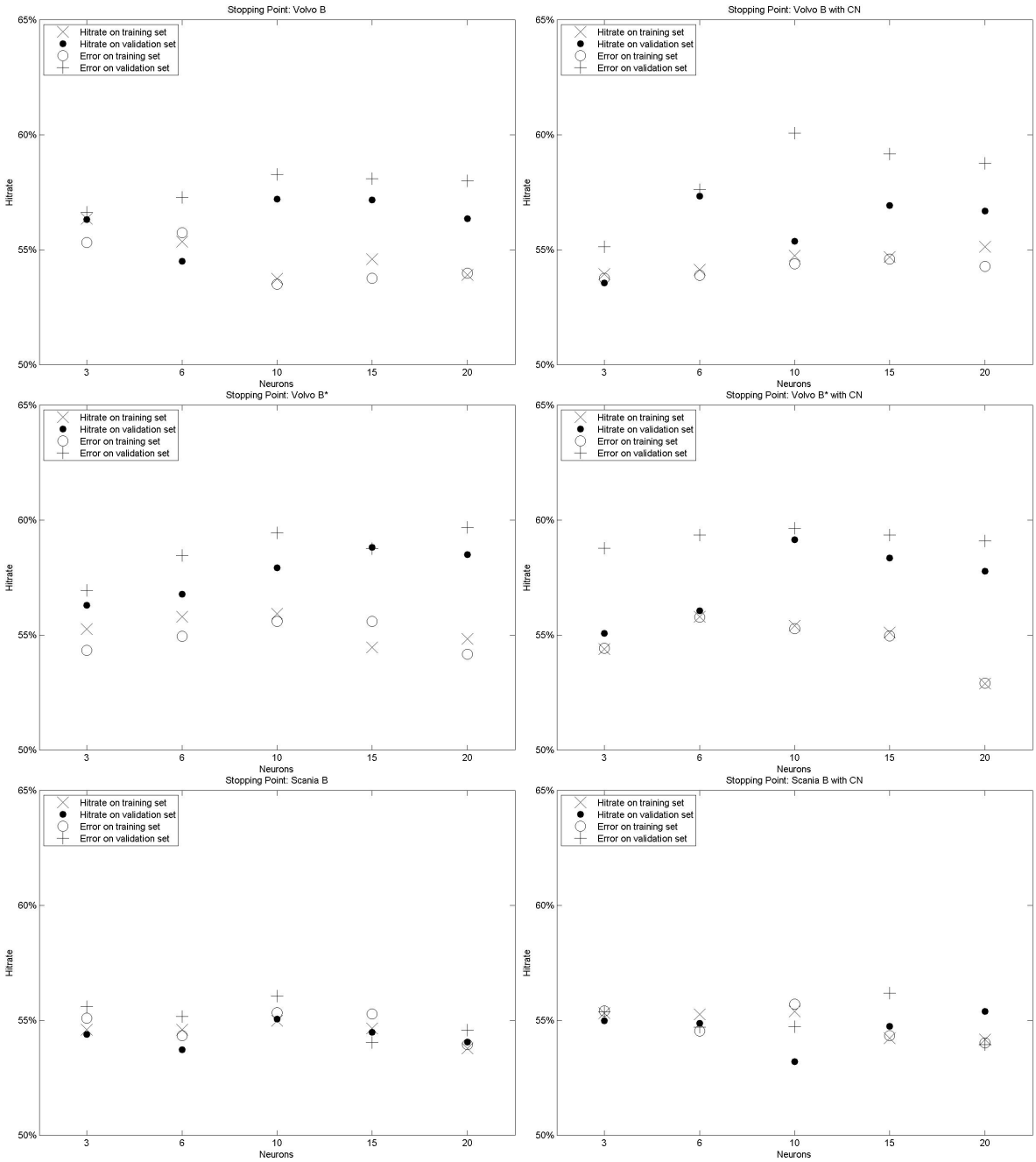

Figure 6.4: This figure shows the hitrates for different choices of stopping points, based on the training or validation set using the hitrate or the error level. The left column shows network configurations without cleaning noise, while the right column shows configurations with cleaning noise. (CN: Cleaning with noise).

Stopping Performance										
	3 Neurons		6 Neurons		10 Neurons		15 Neurons		20 Neurons	
Mean	Hitrate	Gap	Hitrate	Gap	Hitrate	Gap	Hitrate	Gap	Hitrate	Gap
Volvo B	56.63%	5.50%	57.29%	5.78%	58.29%	4.56%	58.11%	4.34%	58.01%	4.42%
Volvo B with CN	55.15%	8.76%	57.62%	4.90%	60.08%	2.43%	59.18%	3.30%	58.77%	3.46%
Scania B	55.59%	5.69%	55.17%	7.65%	56.06%	6.27%	54.05%	9.94%	54.58%	7.67%
Scania B with CN	55.38%	7.21%	54.71%	8.31%	54.74%	8.54%	56.19%	5.14%	53.96%	8.16%
Volvo B*	56.95%	6.94%	58.48%	5.55%	59.45%	3.67%	58.78%	5.16%	59.70%	3.90%
Volvo B* CN	58.79%	5.83%	59.37%	5.09%	59.66%	4.06%	59.36%	4.06%	59.10%	4.72%

Table 6.3: This table shows the generalization performance when using the lowest error on the validation set as a stopping criterion. The mean of the stopping hitrates and gaps for the five different training sessions can be seen. (CN: Cleaning with noise).

Figure 6.5 shows the gap when using the stopping criterion previously discussed, for the Volvo B stock with a short and a long validation set. Observing the figure, the differences between using a long and a short validation set seems to be relatively small, although when using the cleaning with noise technique it has a greater impact.

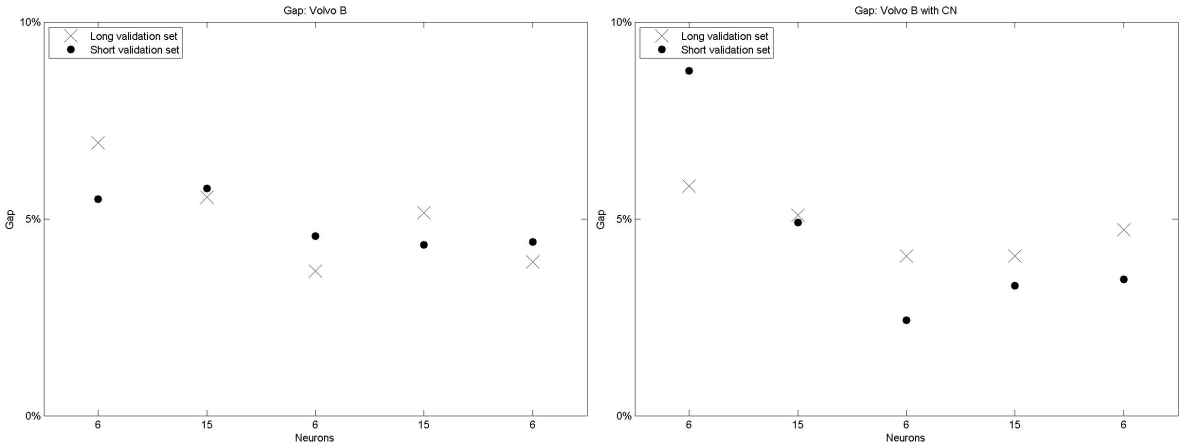

Figure 6.5: This figure shows the gap for the Volvo B stock using a long and a short validation set. The left plot shows the result of networks trained without the cleaning with noise technique, while the right plot shows the results of networks that used it. (CN: Cleaning with noise).

Conclusions

From the results of the 'Potential of ECNN and Training Procedure' test, the following relevant conclusions can be drawn:

- Neurons in State Cluster:

  The number of neurons that achieves the highest performance on the validation set should also be used in the final architecture (i.e. the final forecasting model).

- Stopping Criterion:

  The training session should be stopped in the vicinity of the epoch which has the lowest error level on the validation set.

- Cleaning with Noise:

  From the performed tests, no conclusion can be drawn whether or not to use the cleaning with noise technique.

- Long or Short Validation Set:

  No real conclusions can be drawn concerning whether to use a short or a long validation set. The best choice of validation set size is most likely strongly influenced by the time period during which the forecasting is performed. That is, choosing the validation set so that it best reflects the market, during the time period which is to be predicted, will most likely lead to better forecasting performance. In order to do this, expert knowledge of the market is required.

6.6 Determining MIC, Unfolding and Overshooting

As described in Chapter 4, concerning error correction neural networks, the optimum network architecture needs to be determined. A step in this process is to determine the maximum inter-temporal connectivity (MIC), which is the same as the number of unfolding time steps that should be used. In addition to the MIC, the size of the overshooting network also needs to be determined.

Test Method

The Swedish stocks Volvo B and Scania B are forecasted during this investigation, where the input variables are selected as described in Section 2.7.1. The resulting long and short lists can be seen in Appendix D. The selected data is divided into three sets, covering the time periods which can be seen Table 6.2.

The input data are preprocessed by replacing missing values with the last existing previous value. In addition to this, the raw input time series are also transformed and scaled using momentum and force, as specified in Tables D.2 and D.4 (Volvo B and Scania B). The target series is scaled and transformed using the log-return function (see Equation 2.6) and a target multiplier, a constant factor of thirteen.

The investigation is divided into two stages, where the first stage is to find the MIC and thus the number of unfolding time steps, (see Section 4.1), during which the general architecture is used. In the second stage of the investigation, the number of overshooting steps is to be determined (see Section 4.2). In order to determine the optimum number of overshooting time steps, a number (the same as the MIC) of network architectures, with the overshooting steps ranging from one to the MIC, needs to be tested. Each of these network configurations have the unfolding length that was determined in stage one of this test. For the Volvo B stock, six neurons are used in the state clusters, while ten neurons are used for the Scania B stock.

The different network configurations are trained using the error correction learning rule with the vario-eta optimization algorithm and a step size of 0.01. Each configuration is trained for 200 epochs, 10 000 epochs and using a stopping criterion, where the training is interrupted close to the epoch which has the lowest error level on the validation set.

The error levels and hitrate for the epoch where the training session was halted are extracted, making it possible to investigate how the number of unfolding and overshooting time steps can be determined in a suitable way.

Results

In Figure 6.6 the error levels of the unfolding time steps, for the stopping epoch, can be seen for each training session. It can easily be deduced that training for a fixed amount of epochs is a poor strategy when determining the MIC, which is especially clear when looking at the 10 000 epoch training sessions. This even though the strategy might find the correct result, e.g. 200 epochs for Volvo B. However, when using the stopping criterion, the number of unfolding time steps (MIC) can clearly be determined from the error levels for both the Volvo B and Scania B stocks, which are two and four unfolding times steps.

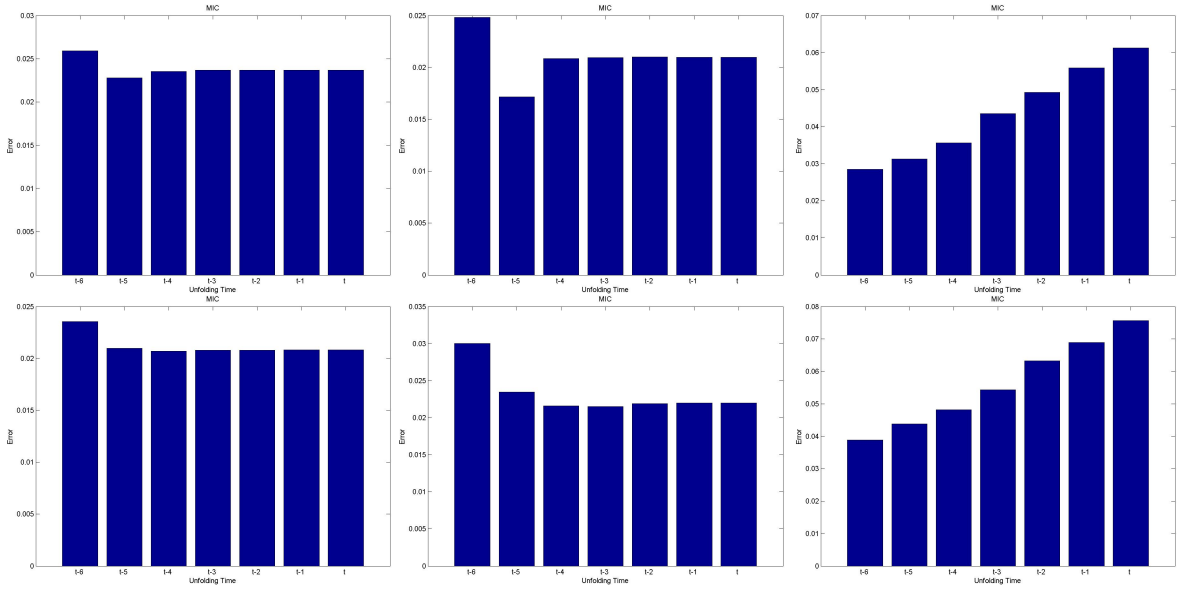

Figure 6.6: This figure shows the error levels for the unfolding time steps for Volvo B (top) and Scania B (bottom) network configurations. The left plots shows the configurations that were trained for 200 epochs, the center plots configuration using a stopping criterion and the right plots configurations trained for 10 000 epochs.

In the second stage of this test, the number of overshooting time steps can be determined from the error levels that are extracted when training the different network configurations. The resulting error levels, from the training sessions where a stopping criterion was used, can be seen in Table 6.4.

Looking at Table 6.4, it can be seen that the lowest error level for time step $t+1$ occurs when the network architecture has two (Volvo B) and four (Scania B) overshooting time steps,

Error Levels					
	Overshooting	t+1	t+2	t+3	t+4
Volvo B	1	0.2680			
	2	0.2665	0.2760		
Scania B	1	0.2248			
	2	0.2229	0.2314		
	3	0.2236	0.2318	0.2327	
	4	0.2205	0.2288	0.2324	0.2328

Table 6.4: This table shows the error levels for the different available future time steps and network configurations.

and should thus be chosen as the size of the overshooting network. This finding is further confirmed when looking at the hitrate (see Table 6.5) for the generalization set, since the highest hitrate is achieved for these network configurations.

Hitrate						
		From:	20030701	20030701	20040101	20040701	20050101
	Overshooting	To:	20050628	20031231	20040630	20041231	20050628
Volvo B	1		53.83%	48.78%	55.00%	52.42%	59.48%
	2		53.94%	50.41%	54.17%	52.42%	59.13%
Scania B	1		53.18%	51.33%	49.58%	60.66%	50.85%
	2		51.38%	48.67%	45.38%	60.66%	50.43%
	3		51.91%	49.56%	42.86%	61.48%	53.45%
	4		53.73%	57.52%	43.70%	62.30%	51.30%

Table 6.5: This table shows the hitrate for the different network configurations at time step $t+1$. The first column shows the hitrate for the entire generalization set, while the other columns shows the hitrate for the subsets of the generalization set.

However, a problem could also be observed when determining the number of overshooting time steps. Looking at Figure 6.7 it can clearly be seen that, even though the maximum inter-temporal connectivity was previously determined to be two (Volvo B) and four (Scania B), a new MIC has been found (one and two). The reason for this behavior is unknown and seems to contradict the outlined method for determining the MIC, as discussed in Section 4.1.1.

Similar tests were also performed for the Volvo B stock, with the modification that different numbers of neurons were used in the state clusters. The results indicates that the number of neurons in the state cluster does not affect the determination of the unfolding and overshooting time steps.

Conclusions

In addition to what has been described in Sections 4.1 and 4.2, the following conclusions can be drawn from the 'Determining MIC, Unfolding and Overshooting' test:

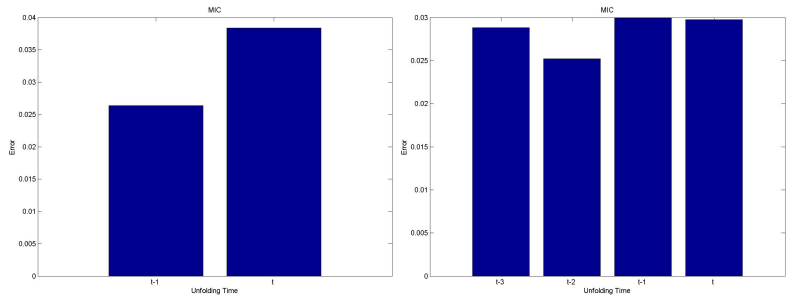

Figure 6.7: This figure shows the maximum inter-temporal connectivity that was found during the determination of the overshooting network size.

- Training Strategy:

  When determining the maximum inter-temporal connectivity and the numbers of unfolding and overshooting time steps, the network should be trained using a stopping criterion.

- Number of Neurons in the State Clusters:

  The test also indicated that the number of neurons in the state clusters does not affect the determination of the maximum inter-temporal connectivity and the numbers of unfolding and overshooting time steps.

- Trouble Determining the MIC:

  A contradiction with the outlined method (see Section 4.1.1) to determine the maximum inter-temporal connectivity could be observed when determining the size of the overshooting network. This since, even though the MIC was determined previously using the outlined method, a new MIC was found when the number of overshooting time steps was determined.

6.7 Short Forecasting Periods

In previous tests, the whole generalization set, two years of data for each stock, has been forecasted using the same training set. In this test, the performance is investigated while only forecasting half a year into the future. Thus, the two years of generalization data are forecasted using four different training sets, each corresponding with half a year of generalization data.

Test Method

During this investigation, the Swedish stocks Volvo B and Scania B are forecasted. The input variables are selected as described in Section 2.7.1 and the resulting long and short lists can be seen in Appendix D. The available data is divided into a training, a validation and a generalization set, which is done once for each of the four forecast periods, as can be seen in Table 6.6.

Training, Validation and Generalization Sets: Short Forecast Period							
		Training		Validation		Generalization	
		Starting	Ending	Starting	Ending	Starting	Ending
Volvo B	Period 1:	2001-02-01	2003-06-30	2003-02-01	2003-06-30	2003-07-01	2003-12-31
	Period 2:	2001-08-01	2003-12-31	2003-08-01	2003-12-31	2004-01-01	2004-06-30
	Period 3:	2002-02-01	2004-06-30	2004-02-01	2004-06-30	2004-07-01	2004-12-31
	Period 4:	2002-08-01	2004-12-31	2004-08-01	2004-12-31	2005-01-01	2005-06-31
Scania B	Period 1:	2001-02-01	2003-06-30	2003-02-01	2003-06-30	2003-07-01	2003-12-31
	Period 2:	2001-08-01	2003-12-31	2003-08-01	2003-12-31	2004-01-01	2004-06-30
	Period 3:	2002-02-01	2004-06-30	2004-02-01	2004-06-30	2004-07-01	2004-12-31
	Period 4:	2002-08-01	2004-12-31	2004-08-01	2004-12-31	2005-01-01	2005-06-31

Table 6.6: This table shows the training, validation and generalization sets for the Volvo B and Scania B stocks, once for each forecast period.

The input data are preprocessed by replacing missing values with the last existing previous value. The preprocessing also includes transformation and scaling of the raw time series using momentum and force as specified in Tables D.2 and D.4 (Volvo B and Scania B). The target series is transformed and scaled using the log-return function (see Equation 2.6) and a target multiplier, a constant factor of thirteen.

The selected architectures used in this test contains two (Volvo B) and four (Scania B) unfolding time steps and the same amount of overshooting steps, as determined in a previous test (see Section 6.6). Six neurons are used for the Volvo B stock, while ten neurons are used for the Scania B stock, in the state clusters.

The different network configurations are trained using the error correction learning rule with the vario-eta optimization algorithm and a step size of 0.01. Each network configuration is trained five separate times, with different initial weights, for 10 000 epochs.

During the training sessions, the hitrate and error levels are extracted for each epoch, thus allowing the potential of the error correction neural network to be examined when using different forecasting periods. In addition to the previously mentioned performance measures, the mean window hitrate (MWH) is also used.

Results

In Table 6.7, the results for each of the forecast periods can be seen, including their mean which corresponds with the entire two years of generalization data. In addition to this, results from previously performed tests ('Potential of ECNN and Training Procedure', see Section 6.5, and 'Determining MIC, Unfolding and Overshooting', see Section 6.6), with the same network configuration except for the size of the forecasting period and that the configurations corresponding with the column 'General' uses the general architecture can be seen.

Very interesting to notice is the relatively high potential that was achieved when short forecast periods were used, compared to when the entire two years of generalization data was forecasted in one training session. In regard to this, it is somewhat surprising to find that the stopping hitrate had not improved, in fact it even worsened slightly. Examining the mean

Performance: Short Forecast Periods								
	From:	20030701	20040101	20040701	20050101	20030701	General	Selected
	To:	20031231	20040630	20041231	20050630	20050630		
Volvo B	Hitrate	63.54%	64.06%	64.15%	69.84%	65.40%	60.81%	61.69%
	MWH*					60.77%	59.47%	59.68%
	Stopping	54.21%	56.88%	57.45%	57.78%	56.58%	57.29%	57.15%
	Gap	14.65%	11.14%	10.31%	17.18%	13.32%	5.78%	7.30%
Scania B	Hitrate	64.69%	63.33%	68.44%	64.96%	65.36%	59.81%	59.58%
	MWH*					61.47%	57.67%	57.96%
	Stopping	56.63%	52.84%	56.51%	57.64%	55.91%	56.06%	55.30%
	Gap	12.41%	16.44%	17.39%	11.20%	14.36%	6.27%	7.17%

Table 6.7: This table shows the potential (hitrate), mean window hitrate, stopping hitrate and gap for each of the forecast periods. In addition to this, the mean of these measures are also shown, which corresponds with the results for the entire two years of generalization data. Corresponding results for the entire generalization set when using the general and selected architectures are also shown.

window hitrate (MWH), it can be seen that there is a relatively large difference between this value and the potential, while for the previous tests, this difference is rather small.

A MWH value far from a forecast's potential indicates that the best epoch is somewhat of an isolated event, since the mean hitrate of the ten closest epochs (excluding the best epoch) is even farther away from the potential than the MWH. Opposite to this, a MWH value close to the potential might indicate a training that evolved more smoothly towards the optimum solution.

A possible explanation for this behavior might be found in the small generalization set, since this increases the chance of an arbitrary epoch of the training session leading to a high hitrate in the generalization set, while not for the validation set. This would mean that there is no relation between the performance in the validation and generalization sets that can be exploited, by a stopping criterion, in order to achieve a stopping hitrate closer to the higher potential that was generated when using short forecast periods. This increase in chance can be attributed to the relatively few points, opposite to when a large generalization set is used, that needs to be a hit in order for the hitrate to increase.

Even though there is a rather large difference between the stopping hitrate and the potential when using short forecasting periods, the stopping hitrate is still only slightly lower than when forecasting a two year period. Thus the test indicates that there is no major difference in performance when using short forecast periods compared to when using longer periods.

Conclusions

The conclusion that can be drawn from the 'Short Forecasting Periods' test is that using short forecast periods seems to decrease the efficiency of using a stopping criterion, even though the potential increases. Thus, when using a stopping criterion, short forecast periods should be avoided in order to achieve the best possible performance.

6.8 State Cluster Neurons

In a previously performed test (see Section 6.5), the number of neurons to use in the state clusters was determined using a general network architecture. This test investigates if the number of neurons also can be determined using a selected architecture.

Test Method

For the purpose of this investigation, the Swedish stocks Volvo B and Scania B are forecasted. The input variables are selected as described in Section 2.7.1 and the resulting long and short lists can be seen in Appendix D. The available data is divided into three sets, covering the time periods which can be seen in Table 6.2.

The input data are preprocessed by replacing missing values with the last existing previous value. The preprocessing also includes transformation and scaling of the raw time series using momentum and force as specified in Tables D.2 and D.4 (Volvo B and Scania B). The target series is transformed and scaled using the log-return function (see Equation 2.6) and a target multiplier, a constant factor of thirteen.

The selected architectures used contains two (Volvo B) and four (Scania B) unfolding and overshooting time steps. The test includes a number of different network configurations where the amount of neurons in the state clusters varies between 6, 10, 15, 20 and 25 neurons.

The different network configurations are trained using the error correction learning rule with the vario-eta optimization algorithm and a step size of 0.01. Each network configuration is trained five separate times, with different initial weights, for 10 000 epochs.

During the training sessions, the hitrate and error levels are extracted for each epoch, thus allowing the potential of the different ECNN configurations to be examined and to find a suitable amount of neurons to use in the state clusters.

Results

In Figure 6.8 the hitrates for the different network configurations and sets can be seen. As in the previous test (see Section 6.5), an appropriate approach when selecting the number of neurons in the state cluster is to use the amount that achieves the highest performance on the validation set.

The performance of the network configurations with the selected architecture, using the number of neurons that resulted in the highest hitrate on the validation set as determined using both the selected (this test) and the general (‘Potential of ECNN and Training Procedure’ test, see Section 6.5) architecture, can be seen in Table 6.8. In addition to this, the performance of the network configurations using the general architecture with the number of neurons determined in the ‘Potential of ECNN and Training Procedure’ test, can also be seen. When comparing the performance of the different configurations in Table 6.8, they seem to be fairly similar even though the number of neurons are quite different.

This can partly be explained when examining the standard deviation of the potential for the network configurations where only the number of neurons varies. In Table 6.9 the standard deviation can be seen for the Volvo B and Scania B stocks, with and without the cleaning with

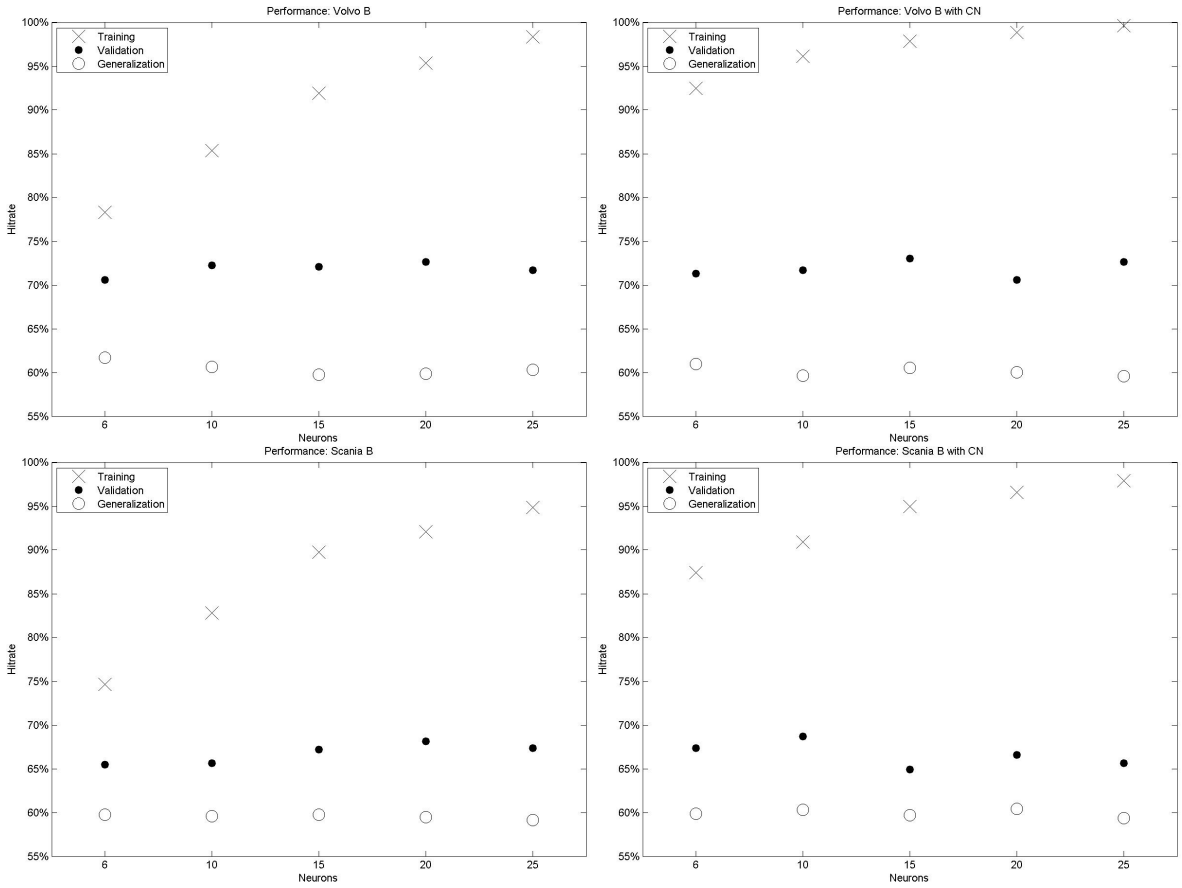

Figure 6.8: This figure shows the performance of the error correction neural network when using the selected as determined in the 'Determining MIC, Unfolding and Overshooting' test (see Section 6.6). (CN: Cleaning with noise).

noise technique and using the general and selected architectures. As can be seen in Table 6.9, the standard deviation is quite low, indicating that the choice of the amount of neurons in the state clusters has a very low impact on the potential of the error correction neural network. This could also indicate that the networks only need a low number of neurons in the state clusters in order to learn all relevant information in the training data. That is, if a network has learned all that it can from the data, increasing the number of neurons used will not have any noticeable effect on the potential of that network.

However, when examining the stopping hitrate the general architecture seems to be the best choice (see Table 6.8), which is somewhat unexpected. This since the selected architecture uses the maximum inter-temporal connectivity to determine how much historic information that is relevant when forecasting, and thus included as input to the network. Thus, a network architecture that includes more historic information should not be able to obtain more relevant information from the data, in fact its forecasting abilities should be lowered since the additional inputs also provides additional noise. This could also be related to the problem of determining the MIC as discussed in Section 6.6.

Performance: State Cluster Neurons											
	Specific Architecture								General Architecture		
	Neurons	Hitrate	Stopping	Gap	Neurons*	Hitrate	Stopping	Gap	Hitrate	Stopping	Gap
Volvo B	20	59.85%	57.03%	4.70%	6	61.69%	57.15%	7.30%	60.81%	57.29%	5.78%
Volvo B with CE	15	60.54%	57.16%	5.57%	10	59.65%	56.83%	4.72%	61.59%	60.08%	2.43%
Scania B	20	59.46%	54.99%	7.52%	10	59.58%	55.30%	7.17%	59.81%	56.06%	6.27%
Scania B with CE	10	60.31%	56.46%	6.34%	3	-	-	-	59.69%	55.28%	7.21%

Table 6.8: This table shows the performance of the network configurations with the selected architecture, using the number of neurons that was determined to be optimum using the selected architecture and the number found using the general architecture. In addition to this, the performance for the general architecture, using the number of neurons that was determined to be optimum when using the general architecture, can also be seen.

Standard Deviation: State Cluster Neurons				
	Selected Architecture		General Architecture	
	Mean	Std	Mean	Std
Volvo B	60.45%	0.78%	60.65%	0.43%
Volvo B with CE	60.15%	0.59%	60.94%	0.47%
Scania B	59.52%	0.26%	59.54%	0.47%
Scania B with CE	59.93%	0.43%	59.44%	0.45%

Table 6.9: This table shows the mean hitrate and standard deviation of the network configurations where only the number of neurons in the state cluster changes.

The fact that the MIC is hard to determine might suggest that the time between an event and information of it reaching the market is short, which might be interpreted as a validation of the efficient market hypothesis. However, since the general architecture seems to achieve the same or a higher performance than the selected architecture, this indicates that the MIC in reality is larger than what was suggested when the selected architecture was determined in a previous test (see Section 6.6). Thus the problem of determining the MIC can not be interpreted as a validation of the EMH, only that it is hard to find the MIC in the stock market.

In addition to this, it can clearly be seen in Table 6.9 that there is no noticeable difference in performance, for neither the general or selected architectures, with or without the cleaning with noise technique.

Conclusions

From the results of the 'State Cluster Neurons' test, the following relevant conclusions can be drawn:

- Determining the Number of Neurons in the State Cluster:

  No real conclusions can be drawn concerning whether or not to determine the number of neurons in the state clusters using a general or selected architecture. This since, even

though the number of optimum neurons varies for the different network configurations, their potential are roughly the same, which also is true for the stopping hitrate.

- Using a General or Selected Architecture:

  When examining the potential of ECNN using a general and a selected architecture, using the general architecture seems to lead to the best results, even though the difference between the architectures is very small. Since the test indicates that the MIC is hard to determine in the stock market, this is an additional reason to use the general architecture. Another benefit with using a general architecture is that it takes fewer steps to determine the final network configuration. At the same time, it also takes longer time to train the networks with a general architecture.

- Cleaning with Noise:

  The test also indicates that using the technique of cleaning with noise leads to no noticeable difference in the potential of the error correction networks. Since it takes longer time to train a network that employs this technique, it is advisable not to use it.

6.9 Target Multiplier

When preprocessing the target series using the log-return function, the range of the values in the resulting transformed series becomes very small. In addition to this, the error correction neural networks uses linear output neurons, which means that there is no need to limit the range of the target series.

However, using time series with small ranges can lead to numerical problems during the training of the network. A remedy could be to multiply the target series with a constant factor in order to increase its range. Thus the effect of using different target multipliers will be examined in relation to a network's potential, overfitting behavior and efficiency of the stoping criterion.

Test Method

During this investigation, the Swedish stocks Volvo B and Scania B are forecasted. The input variables are selected as described in Section 2.7.1 and the resulting long and short lists can be seen in Appendix D. The available data is divided into three sets, covering the time periods which can be seen in Table 6.2.

The input data are preprocessed by replacing missing value with the last existing previous value. The preprocessing also includes transformation and scaling of the raw time series using momentum and force as specified in Tables D.2 and D.4 (Volvo B and Scania B). The target series is transformed and scaled using the log-return function (see Equation 2.6) and then multiplied with a target multiplier, a constant factor. Five different target multipliers, the constant factors 1, 7, 13, 20 and 27, are tested in this investigation. The network configurations in this test uses the general architecture with six (Volvo B) and ten (Scania B) neurons in the state clusters.

The different network configurations are trained using the error correction learning rule with the vario-eta optimization algorithm and a step size of 0.01. Each network configuration is trained five separate times, with different initial weights, for 10 000 epochs.

During the training sessions, the hitrate and error levels are extracted for each epoch, thus allowing the potential of the different ECNN configurations to be examined. In addition to this, signs of overfitting behavior can be detected both during and after the training session.

Results

It is important to understand that the error correction neural network uses the error of its previous predictions as an additional input. Examining Equation 4.10, it can easily be seen that the errors are derived through the subtraction of the desired target values from the previous predictions. This has the effect that, if the range of the target series is small, so will the errors be. Thus, if the range of the target series is increased, the range of the predicted values will also increase (assuming that the predictions are relatively close to their targets) and by extension the range of the error. This assumption can best be explained by examining the errors that are used as additional inputs to ECNN, see Equation 6.4, where y^d is the target value and y is the predicted value.

$$e = y^d - y \tag{6.4}$$

If a constant target multiplier M is applied to the target value, the network will make a different prediction and thus the error will also change, according to Equation 6.5 where e_n and y_n are the new error and prediction respectively.

$$e_n = My^d - y_n \tag{6.5}$$

The new prediction can be described in terms of the target multiplier, the original prediction and an error as can be seen in Equation 6.6.

$$y_n = My - \Delta_e \tag{6.6}$$

This allows Equation 6.5 to be rewritten so that the new prediction is eliminated, according to Equation 6.7. Notice that, since the target multiplier is a scaling factor, the relative error for the original and new predictions is the same if Δ_e is zero (even though e_n is M times larger than e). Thus Δ_e is a measure of the prediction performance due to the target multiplier (compared to when no multiplier is used). This means that if $\Delta_e < 0$ holds, the prediction performance (in terms of error levels) when using the target multiplier is better than when not using it.

$$e_n = My^d - My + \Delta_e = M\left(y^d - y\right) + \Delta_e = Me + \Delta_e \tag{6.7}$$

Equation 6.7 implies that, if $\Delta_e > (1-M)\,e$, the range of the error input to the error correction neural network increases when the target multiplier M is applied to the target value. This makes the error correction neural network more resilient to numerical issues due to small error values, potentially allowing it to learn more of the underlying dynamics in the training data. However, it is important to notice that using a target multiplier does not necessarily lead

to a smaller relative error, since Δ_e might be positive and thus adds to the error obtained when no target multiplier was used.

In addition to this, the error correction neural network uses the hyperbolic tangent function to scale down the error information (while the target multiplier scales it up). Thus the optimum range of the error values is the same as the (approximately) linear part of the hyperbolic tangent function's input range (i.e. close to zero, see Sections 2.5 and 3.1.1).

Also note that the choice of a target multiplier not only affects the range of the error values used as input to ECNN, but also the performance of the optimization algorithm. This since it uses the error of the network to determine new optimum weight values. Thus small error values leads to numerical issues for the optimization algorithm as well. In addition to affecting the optimization algorithm, the target multiplier also has an impact on the optimum step length. This since the target multiplier in a sense scales the search space, which obviously affects the optimum step length. A reasonable approach when selecting a new step length is to scale the previously optimum length using the same constant factor as the target multiplier.

The error levels for the different epochs and network configurations can be seen in Figure 6.9. Examining the plots for when no target multiplier is used, it seems as though the network has numerical problems. This since the network only is able to achieve a certain level of accuracy in the beginning of the training session, after which no major changes occurs in the error level. Examining the plots for when a suitable target multiplier is used, it can be deduced from the variations in the error levels that these networks are better at extracting information from the training data. It can also easily be seen that these network configurations exhibits signs of overfitting behavior, since the error level of the validation set starts to increase while it is still decreasing for the training set.

These findings are further confirmed in Table 6.10, which shows the hitrates for a number of network configurations with different target multipliers. As can be seen, all the examined target multipliers leads to a better performance than when no target multiplier is used (i.e. when a constant factor of one is used).

Perfromance: Target Multiplier						
		1	7	13	20	27
Volvo B	Hitrate	57.91%	60.77%	60.81%	62.01%	61.16%
	Stopping	54.27%	58.00%	57.29%	59.42%	57.10%
	Gap	6.23%	4.57%	5.78%	4.19%	6.68%
Scania B	Hitrate	58.68%	59.57%	59.81%	59.30%	59.23%
	Stopping	55.16%	54.85%	56.06%	54.57%	53.23%
	Gap	6.00%	7.88%	6.27%	7.95%	10.09%

Table 6.10: This table shows the hitrates, stopping hitrates and gaps for the Volvo B and Scania B stocks when using different target multipliers.

A reasonable approach when selecting a suitable target multiplier is to, similar to how the number of neurons in the state clusters is selected, train the network using several different target multipliers and then select the one that achieves the highest performance on the validation set. When examining Figure 6.10, this approach seems to be sound. Suitable target

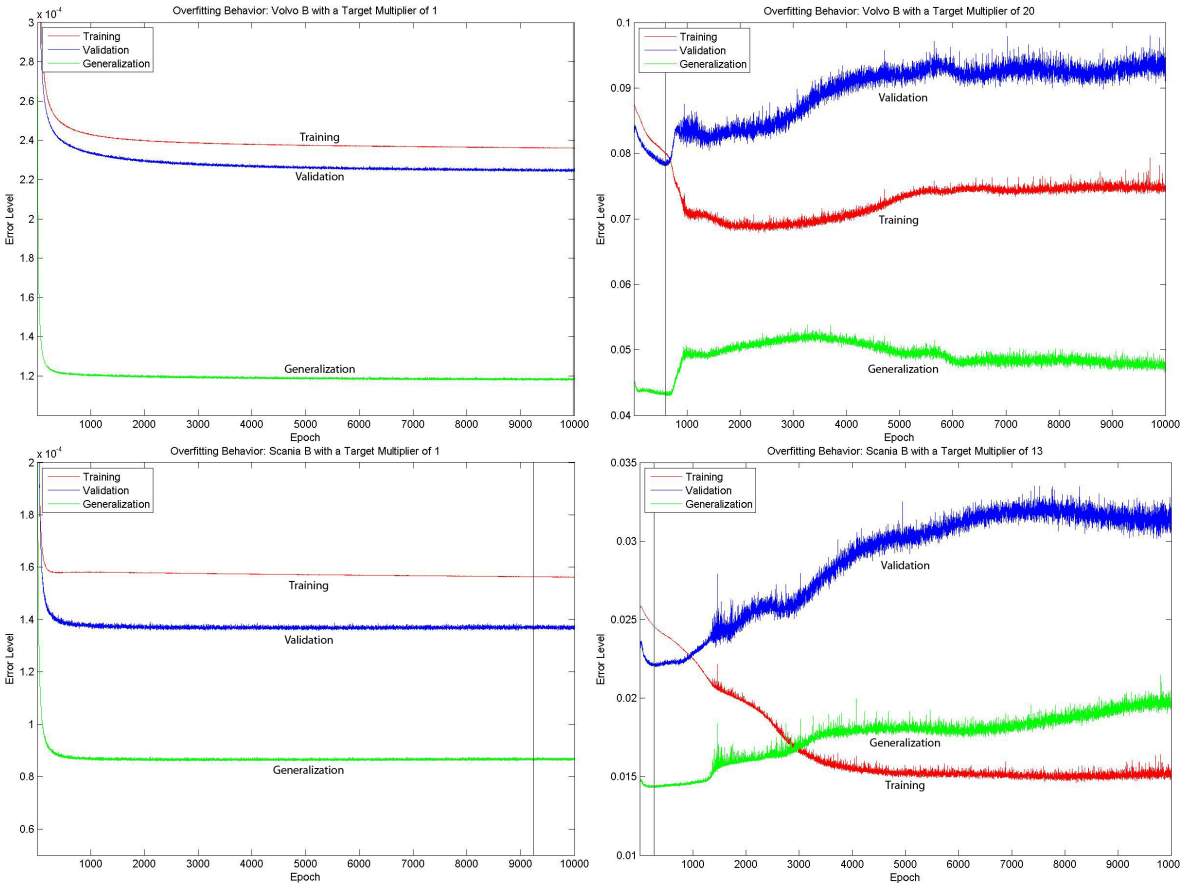

Figure 6.9: The different plots shows the error level during each epoch of a training session. The upper two plots covers the Volvo B stock using no target multiplier (left) and one with the constant factor of twenty (right). The two lower plots covers the Scania B stock using no target multiplier (left) and one with a constant factor of thirteen (right). The error levels are shown for the training, validation and generalization sets. The vertical lines indicates which epoch that resulted in the lowest error for the generalization set.

multipliers for the Volvo B and Scania B stocks are twenty and thirteen.

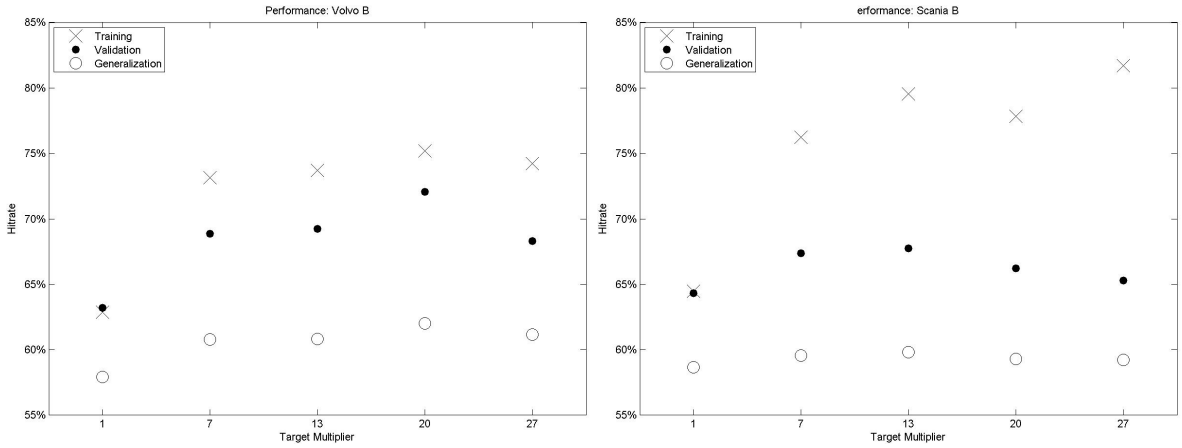

Figure 6.10: This figure shows the potential when different target multipliers are used for the training, validation and generalization sets. The left plot covers the Volvo B stock while the right plot covers the Scania B stock.

Conclusions

From the results of the 'Target Multiplier' test, the following relevant conclusions can be drawn:

- Numerical Problems:

  As mentioned in the introduction to and confirmed by this test, small ranges of the target series leads to numerical issues, which have an adverse effect on the performance during forecasting. It was deduced that a small range of the target values might lead to even smaller ranges for the error values. Since these errors are used both as input to the error correction neural network and during training by the optimization algorithm, they have a strong influence on the network's performance. In addition to this, the range of the error values also affects the optimum step length.

  It was also deduced that a target multiplier can be used on the target series in order to increase the range of the error values, thus providing a solution to the numerical issues.

- Selecting a Target Multiplier:

  A reasonable approach when selecting a target multiplier is to train several networks using different target multipliers. The target multiplier that led to the best performance on the validation set should then be selected for the final network configuration.

- Selecting a Step Length:

  When selecting a step length for a network that uses a target multiplier, a reasonable starting point is to multiply the same constant factor (i.e. the selected target multiplier) with the optimum step length for the network where no target multiplier was used.

6.10 Replacing Missing Values

Missing values in time series is a common quality issue when using stock data (e.g. stock prices and volumes) and can have a negative effect on the performance of a forecast model. There are several methods in which missing values can be handled, the goal of which should be to enable the network to learn the underlying dynamics of the data, or at the very least not hinder it.

This test investigates two simple methods, where the first one replaces the missing values with the last existing previous value. A problem with this method is that, when some transformations are used (e.g. simple return), this can result in zero values in the transformed series (i.e. if the price is the same during $t-1$ and t, the one-step simple return for t is zero). The other method investigated uses the mean of the last five values to replace the missing value. This should reduce the risk of zero values occurring in the transformed series as a result of missing values.

Test Method

During this investigation, the Swedish stocks Volvo B and Scania B are forecasted. The input variables are selected as described in Section 2.7.1 and the resulting long and short lists can be seen in Appendix D. The available data is divided into three sets, covering the time periods which can be seen in Table 6.2.

The input data are preprocessed by replacing missing values using two different methods. The first method replaces the missing values with the last existing previous value, while the second method replaces it with the mean of the last five values. The preprocessing also includes transformation and scaling of the raw time series using momentum and force as specified in Tables D.2 and D.4 (Volvo B and Scania B). The target series is transformed and scaled using the log-return function (see Equation 2.6) and then multiplied with a target multiplier, a constant factor of thirteen.

The network configurations in this test uses the general architecture with six, ten and fifteen neurons in the state clusters. In addition to this, the networks are also tested with and without the cleaning with noise technique.

The different network configurations are trained using the error correction learning rule with the vario-eta optimization algorithm and a step size of 0.01. Each network configuration is trained five separate times, with different initial weights, for 10 000 epochs.

During the training sessions, the hitrate and error levels are extracted for each epoch, thus allowing the potential of different ECNN configurations to be examined. Since the SENN hitrate (see Equation 5.3) counts zero values as hits, the choice of method to replace missing values can have an impact on it (i.e. improvements in the hitrate might not be the result of improvements of the performance). This means that comparing the hitrates for forecasts that uses different methods to replace missing values might be misrepresenting. In order to solve this issue, the hitrates extracted during training are normalized using Equation 6.8, where N is the number of patterns and Z is the number of zero values, in the set of target values for which the hitrate is calculated. Normalizing the hitrates removes the effect that zero values in the target series have on the hitrate.

$$HR_n = \frac{HR_s N - Z}{N - Z} \tag{6.8}$$

Results

It was expected that the mean replacing method would be better suited to use in combination with the cleaning with noise technique than the replace with previous value method. This since the mean replacing method should lead to a more smooth target series, which in turn would lead to more smooth error values. The cleaning with noise technique uses these errors to find cleaning terms, which in turn are used to generate noise which is added to the input data. Thus it would stand to reason that a replacing method that generates more smooth error values should enable the cleaning with noise technique to perform better. However, the results in Table 6.11 indicates that this was not the case, which could mean that the assumption was wrong or that the mean replacing method does not lead to smoother error values than the replace with previous value method.

Performance: Replacing Missing Values									
		Replacing with Mean				Replacing with Previous			
	Neurons	6	10	15	Mean	6	10	15	Mean
Volvo B	Hitrate	57.56%	57.40%	57.48%	57.48%	58.13%	57.95%	57.57%	57.88%
	Stopping	53.65%	53.86%	54.19%	53.90%	53.85%	54.93%	54.73%	53.84%
Volvo B with CN	Hitrate	57.77%	57.16%	56.96%	57.30%	57.40%	58.49%	58.07%	57.99%
	Stopping	54.23%	53.30%	53.99%	53.84%	54.20%	56.87%	55.89%	54.99%
Scania B	Hitrate	55.97%	55.23%	55.39%	55.53%	55.52%	55.56%	55.81%	55.63%
	Stopping	50.73%	50.79%	51.14%	50.89%	50.44%	51.41%	49.20%	50.35%
Scania B with CN	Hitrate	55.23%	56.83%	55.19%	55.75%	55.43%	55.60%	54.92%	55.32%
	Stopping	50.05%	49.79%	52.10%	50.65%	49.92%	49.95%	51.56%	50.48%

Table 6.11: This table shows the performance of the error correction neural network when replacing missing values with the last existing previous value and when replacing it with the mean of the five previous values. Also note that the hitrates have been normalized according to Equation 6.8 and can not be compared with hitrates from other tests. (CN: Cleaning with noise).

The results in Table 6.11 also indicates that neither of the methods leads to a better performance than the other, thus it is of little consequence which of the two methods is used.

Also note that, since the hitrates in Table 6.11 are corrected for zero values in the target series (see Equation 6.8), it represents a more correct measure of the forecast performance. This since there is no way to determine whether a prediction is correct or not when the target value is zero.

Conclusions

From the results of the 'Replacing Missing Values' test, the following relevant conclusions can be drawn:

- Cleaning with Noise:

  The test indicated that using the mean replacing method did not affect the performance of the cleaning with noise technique in any noticeable way (compared to the previous value replacing method).

- Missing Value Replacing Method:

  No difference in the performance worth mentioning could be observed when comparing using the previous value replacing method and mean replacing method.

6.11 Using Volume as Input Variable

Volume series for a stock contains information concerning the amount of stocks traded during some time period. Potentially these volume series contains a lot of useful information when forecasting. This will be investigated in this test by adding the volume series, of the stock that is predicted, to the input variables.

Test Method

During this investigation, the Swedish stocks Volvo B and Scania B are forecasted. The input variables are selected as described in Section 2.7.1 and the resulting long and short lists can be seen in Appendix D. In addition to these input variables, the volume series for the stock that is forecasted is also added to the input variables. The available data is divided into three sets, covering the time periods which can be seen in Table 6.2.

The input data are preprocessed by replacing missing values with the last existing previous value. The preprocessing also includes transformation and scaling of the raw time series using momentum and force as specified in Tables D.2 and D.4 (Volvo B and Scania B). The volume series is transformed and scaled using both the momentum and force, while the target series is scaled and transformed using the log-return function (see Equation 2.6) and a target multiplier, a constant factor of thirteen.

The network configurations in this test uses the general architecture with six, ten, fifteen and twenty neurons in the state clusters.

The different network configurations are trained using the error correction learning rule with the vario-eta optimization algorithm and a step size of 0.01. Each network configuration is trained five separate times, with different initial weights, for 10 000 epochs.

During the training sessions, the hitrate and error levels are extracted for each epoch, allowing the potential of the error correction neural network to be examined when adding volume information to the input variables.

Results

In Table 6.12 the results from the test can be seen with a calculated mean of the network configurations, with varying numbers of neurons in the state clusters, for each stock. The mean is also shown for when the volume was not included. Investigating the differences, it can be seen that the mean potential is slightly higher when the volume is included. However, this is not true for the stopping hitrate, which is higher when using the volume for the Scania B stock and worse for the Volvo B stock.

Performance: Volume									
		Volume						No Volume	
	Neurons	6	10	15	20	Mean	Std	Mean	Std
Volvo B	Hitrate	61.08%	61.47%	61.59%	60.50%	61.16%	0.49%	60.83%	0.43%
	Stopping	57.58%	58.85%	59.40%	57.60%	53.36%	0.91%	56.32%	1.10%
	Gap	5.72%	4.24%	3.56%	4.78%	4.58%	0.91%	4.77%	0.67%
Scania B	Hitrate	58.99%	59.23%	58.92%	59.27%	59.10%	0.17%	59.69%	0.47%
	Stopping	54.38%	53.80%	54.99%	54.66%	54.46%	0.50%	54.34%	0.50%
	Gap	7.84%	9.15%	6.65%	7.75%	7.85%	1.02%	7.88%	1.64%

Table 6.12: This table shows the performance of the different network configurations when the volume was added to the input variables and for comparison when it was not included.

Since the difference in potential is small and the lack of pattern in the results for the stopping hitrate, no real conclusions can be drawn from this test. The best approach when deciding whether to include the volume or not is most likely to treat it as any other input variable during the input variable selection process (see Section 2.7.1).

Conclusions

The closest to a conclusion that can be drawn from the 'Using Volume as Input Variable' test is that the volume should be treated as any other input variable during the input variable selection process (see Section 2.7.1).

6.12 Verification of Findings: SE Banken A

During previous tests, different approaches of how to configure the error correction neural network in order to obtain the best possible forecast performance. This test aims to verify some of the conclusions that were drawn during those tests, when applied simultaneously, using the SE Banken A stock as the target series.

Test Method

During this investigation, the Swedish stock SE Banken A is forecasted. The input variables are selected as described in Section 2.7.1 and the resulting long and short lists can be seen in

Appendix D. The available data is divided into three sets, covering the time periods which can be seen in Table 6.2.

The input data are preprocessed by replacing missing values with the last existing previous value. The preprocessing also includes transformation and scaling of the raw time series using momentum and force as specified in Table D.6 (SE Banken A). The target series is transformed and scaled using the log-return function (see Equation 2.6) and a target multiplier, a constant factor of 7, 13, 20 or 27. Which target multiplier to use in the final network configuration will be determined during the test, through comparison of the forecast performances for the different target multipliers.

Both the general and selected architectures are tested during this investigation, where the selected architecture is determined during the test, using the maximum inter-temporal connectivity and error levels of future predictions as described in Sections 4.1 and 4.2. Both architectures are tested using six, ten and fifteen, the specific architecture also tests 20 and 25, neurons in the state clusters. The results are then used to determine the optimum amount of neurons to use in the state clusters for the two different architectures. Also note that the target multiplier is tested using the optimum amount of neurons in the state clusters.

The different network configurations are trained using the error correction learning rule with the vario-eta optimization algorithm and a step size of 0.01. Each network configuration is trained five separate times, with different initial weights, for 10 000 epochs.

During the training sessions, the hitrate and error levels are extracted for each epoch, allowing the potential of the different network configurations to be examined in order to evaluate the conclusions drawn in previous tests.

Results

In Figure 6.11, the results from the investigation, when the general architecture was used, can be seen. According to these results, based on previously drawn conclusions, ten neurons will be used in the state clusters and a constant factor of thirteen will be used as a target multiplier.

For the selected architecture, the first step is to determine the number of unfolding and overshooting time steps to use, based on conclusions drawn in previous tests. In Figure 6.12, the MIC has been determined to be two, which is the number of unfolding time steps that will be used, while from Table 6.13 it can be deduced that two overshooting time steps is to be used.

Error Levels			
	Overshooting	t+1	t+2
SE Banken A	1	0.2226	
	2	0.1982	0.2099

Table 6.13: This table shows the error levels for the different future predictions of the SE Banken A stock and can be used when determining the overshooting size of the ECNN network.

In Figure 6.13, the results from when investigating the selected architecture can be seen.

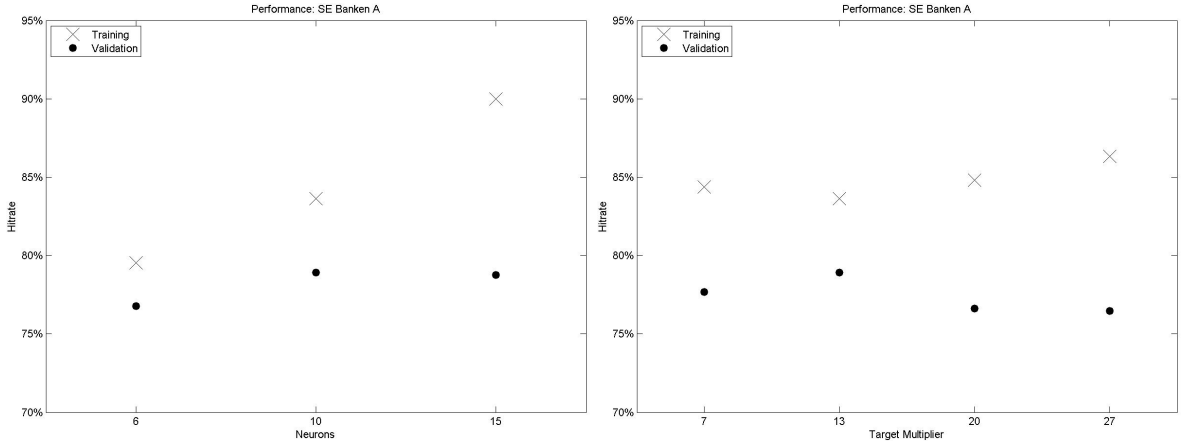

Figure 6.11: This figure shows the performance for the SE Banken A stock on the training and validation sets when using the general architecture for different number of neurons in the state clusters (left) and different target multipliers (right).

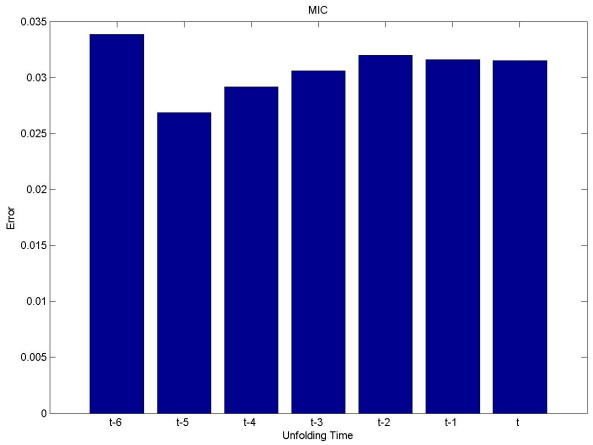

Figure 6.12: This figure shows the maximum inter-temporal connectivity of the SE Banken A stock.

From these results, based on conclusions drawn in previous tests, twenty neurons should be used in the state clusters with a target multiplier of thirteen.

Table 6.14 shows the potential for two of the network configurations, one with a general and one with a selected architecture, both using the optimum target multiplier and number of state cluster neurons, as determined previously in this test.

Even though the results indicates, contradictory to results in previous tests, that the selected architecture leads to the highest performance, the difference between the potentials and stopping hitrates of the selected and general architecture is very small. In addition to this, the gap is still higher for the selected architecture than for the general architecture, indicating that networks using a selected architecture are harder to train when using a stopping criterion. This in combination with results from previous tests, indicating that it is hard to find the correct MIC, suggests that it still is more suitable to use the general architecture when forecasting.

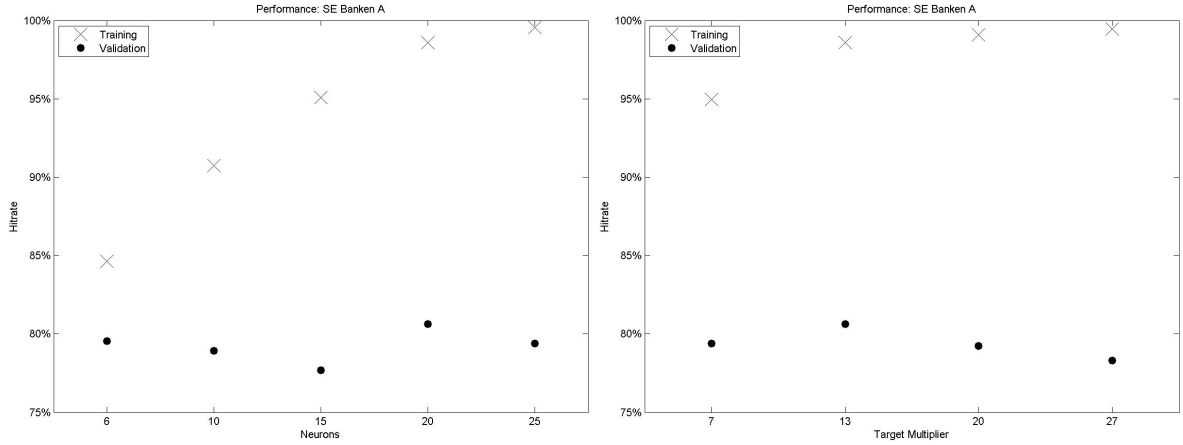

Figure 6.13: This figure shows the performance for the stock SE Banken A on the training and validation sets, when using the selected architecture, for different numbers of neurons in the state clusters (left) and target multipliers (right).

Performance		
	General	Selected
Hitrate	65.23%	65.42%
Stopping	62.18%	62.21%
Gap	4.67%	4.91%

Table 6.14: This table shows the performance of two network configurations, one with a general and one with a selected architecture, that uses the optimum amount of state cluster neurons and target multiplier as determined previously in this test.

Conclusions

It seems as though the findings from previous tests, that were investigated in the 'Verification of Findings: SE Banken A' test, still are valid for the stock SE Banken A. This suggests that the findings in the previous tests might generalize to stocks other than the Volvo B, Scania B and SE Banken A stocks.

6.13 Verification of Findings: Forecasting

Applying the knowledge gained during previous tests, this test aims to investigate the performance of the ECNN when using a stopping criterion, which also provides an opportunity to evaluate the potential of this knowledge. In addition to this the test will also investigate the potential of using thick modeling, where several similar networks are trained and based on their performances on the validation set, one is selected as the thick model output.

Test Method

During this investigation, the Swedish stocks Volvo B, Scania B and SE Banken A are forecasted. The input variables are selected as described in Section 2.7.1 and the resulting long and short lists can be seen in Appendix D. The available data is divided into three sets, covering the time periods which can be seen in Table 6.2.

The input data are preprocessed by replacing missing values with the last existing previous value. The preprocessing also includes transformation and scaling of the raw time series using momentum and force as specified in Tables D.2, D.4 and D.6 (Volvo B, Scania B and SE Banken A). The target series are transformed and scaled using the log-return function (see Equation 2.6) and then multiplied with a target multiplier, which can be seen in Table 6.15.

Network Architectures					
	Architecture	Neurons	TM	Unfolding	Overshooting
Volvo B	General	6	20	7	6
	Selected	20	20	2	2
Scania B	General	10	13	7	6
	Selected	20	13	4	4
SE Banken A	General	10	13	7	6
	Selected	20	13	2	2

Table 6.15: This table shows the network configurations that are used in this test.

Two different network architectures are tested for each stock during this investigation, a general and a selected architecture. The different network configurations (i.e. the number of neurons in the state clusters, unfolding and overshooting steps), as determined in previous tests, can be seen in Table 6.15.

The different network configurations are trained using the error correction learning rule with the vario-eta optimization algorithm and a step size of 0.01. Each network configuration is trained for five separate times, with different initial weights, using the stopping criterion based on the lowest error on the validation set.

In addition to this, the potential of thick modeling is investigated, using the five separate training sessions of each network configuration as input. The thick model output is determined using a selection criterion, based on the performance on the validation set, where the network with the highest performance is selected as output of the thick model. Two different performance measures are investigated in order to determine which is suitable to use as a selection criterion, the hitrate and realized potential.

The predicted values at the stopping epoch are extracted and investigated thoroughly in order to determine the forecast performance. This is accomplished using a number of performance measures (hitrate, return on investment, realized potential and annualized return) and benchmarks (naive prediction and the NASDAQ Composite index).

Results

The highest hitrates and realized potentials achieved during the five training sessions of each network configuration, in addition to the mean hitrates, can be seen in Table 6.16. The hitrate and realized potential of the trained networks that were selected as output of the thick model can also be seen. Notice that only the results of one selection criterion can be seen, even though two different selection criteria were tested. This since they selected the same trained networks as output of the thick model. Thus, the choice between using the hitrate or realized potential on the validation set, as a selection criterion, has no or little influence on the performance of the thick model. This also indicates that there is a strong connection between the hitrate and realized potential achieved during a training session.

In addition to this, it was also noted during the training sessions that they sometimes reached their stopping point after only a few epochs. It seems unlikely that a network is able to achieve good generalization performance after only a few epochs and the low error level on the validation set is most likely due to random chance (e.g. using a random initial vector that achieves a good performance on the validation set) with no relation to the performance on the generalization set. Thus forecast models that are stopped after only a few epochs should be rejected or retrained.

Performance: Stopping Criterion									
	Architecture	Max Hitrate		Mean Hitrate		Max RP	Selection Criterion		
		SENN	Norm	SENN	Norm		Hitrate	RP	AR
Volvo B	General	59.77%	56.58%	58.53%	55.24%	17.22%	58.61%	13.39%	20.90%
	Selected	57.58%	54.15%	56.74%	53.24%	5.69%	57.20%	5.69%	11.49%
Scania B	General	59.77%	55.56%	55.51%	50.85%	10.78%	55.13%	7.49%	23.79%
	Selected	57.42%	52.88%	55.92%	51.22%	12.00%	57.42%	7.81%	21.07%
SE Banken A	General	65.70%	58.16%	63.41%	55.37%	13.85%	63.57%	12.24%	54.30%
	Selected	63.65%	55.63%	62.62%	54.37%	13.27%	62.31%	9.74%	36.96%

Table 6.16: This table shows the performance for the Volvo B, Scania B and SE Banken A stocks when using a stopping criterion. The SENN hitrates corresponds with Equation 5.3, while the Norm hitrate corresponds with Equation 5.2. In addition to this the results of the thick model can also be seen.

As can be seen in Table 6.16, the thick models based on the general architecture achieves a higher performance than when they are based on the selected architecture, except for the Scania B stock, where the selected architecture leads to a higher hitrate and realized potential. Notice that for the Scania B stock, the realized potential is almost the same when using the general and selected architecture, while the annualized return is higher for the general architecture. For this reason in combination with previous test results which indicates that it is hard to determine a suitable selected architecture, it is still a good idea to use the general architecture.

In addition to this, it can also be seen that the hitrates of the thick model outputs are very close to the mean hitrates of the five training sessions (performed for each network configuration). Thus using thick models must be considered a safer choice since, while it might not

lead to the highest possible performance, it is also less likely that it leads to a much worse performance than the average.

In Table 6.17, a more complete presentation of the results from the thick models, based on the general architecture, can be seen. This includes hitrates for four half year periods of the generalization set and a comparison with the naive benchmark.

Performance: Stopping Criterion with Thick Modeling						
		Hitrate		Norm	SENN	Naive
Volvo B	Period 1:	54.55%	Hitrate:	55.32%	58.61%	1.16
	Period 2:	56.92%	RP:	13.39%		-3.76%
	Period 3:	59.85%	AR:	20.90%		
	Period 4:	63.41%				
Scania B	Period 1:	51.52%	Hitrate:	50.43%	55.13%	1.11
	Period 2:	50.77%	RP:	7.49%		5.39%
	Period 3:	61.36%	AR:	23.79%		
	Period 4:	56.91%				
SE Banken A	Period 1:	68.94%	Hitrate:	55.56%	63.57%	1.13
	Period 2:	58.91%	RP:	12.24%		10.78%
	Period 3:	62.88%	AR:	54.30%		
	Period 4:	63.41%				

Table 6.17: This table shows the performance of the thick models based on the general architecture. The SENN hitrates corresponds with Equation 5.3, while the Norm hitrate corresponds with Equation 5.2. There is also a comparison with naive prediction (see Section 5.3.1), where the hitrate quota corresponds with Equation 5.11. Each period corresponds with half a year of data in the generalization set, in the same order in which they are listed.

Figure 6.14 shows the return on investment for the different stocks, including the NASDAQ Composite index which serves as a benchmark. The return on investment is calculated using a trading strategy (taking long and short positions based on if the stock is predicted to increase or decrease in value, see Section 5.2.3) based on the thick models for each stock and a buy and hold strategy for the NASDAQ Composite index benchmark.

Conclusions

From the results of the 'Verification of Findings: Forecasting' test, the following relevant conclusions can be drawn:

- Thick Modeling:

  This investigation indicates that using thick models leads to an average performance and can thus be considered a safer choice. This since, even though it might not lead to the highest performance, it is also less likely that the performance is much worse than the average, compared to when not using thick models (i.e. when only training the network once).

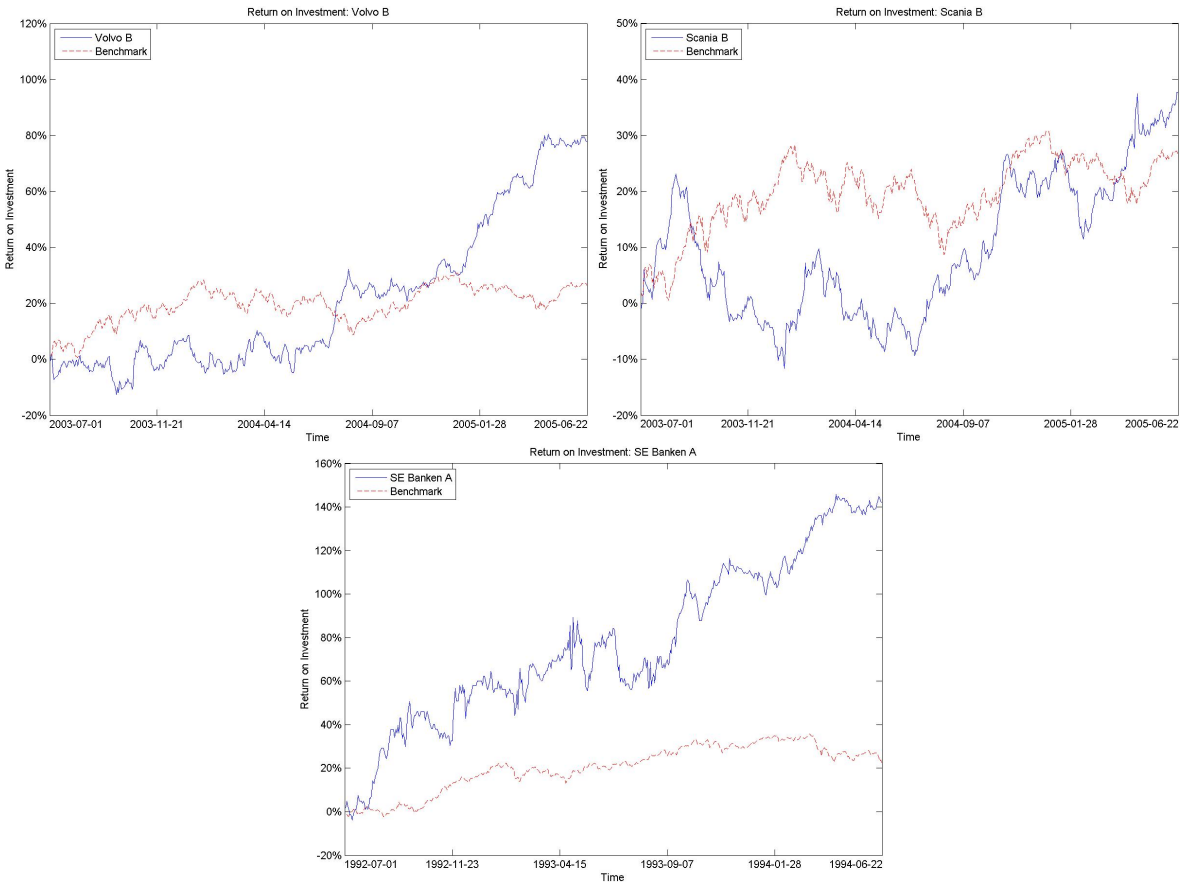

Figure 6.14: This figure shows the return on investment for the Volvo B (left), Scania B (right) and SE Banken A (bottom) stocks, including the NASDAQ Composite index benchmark.

- Selection Criterion:

  This test investigated two different selection criteria, used by the thick model to determine its output, where one uses the hitrate and the other the realized potential, on the validation set. The investigation indicates that both selection criteria leads to the same performance, making the choice between them trivial.

- Stopping Point:

  It was noticed that a few training sessions were stopped after only a few epochs. It was deduced that this stopping point most likely is due to random chance with no relation to the networks' generalization performance, thus networks that are stopped after only a few epochs should be rejected or retrained.

- General vs. Selected Architectures:

  When comparing the performance between networks using the general and selected architecture, only small differences could be detected. This indicates that the choice between using a general and a selected architecture is trivial, but since previous tests

showed that it is hard to determine a suitable selected architecture, it is still more appropriate to use the general architecture.

- Performance:

  Even though it might be desirable with higher hitrates than what was achieved with the thick model for the different stocks, the absence of negative realized potential and annualized return should be considered a success. In addition to this, the return on investment for all stocks performed better than the NASDAQ Composite index benchmark.

Chapter 7

Method to Select a Forecasting Model Based on ECNN

This chapter contains the method, developed based on the literature and empirical studies. The aim of the method is to help developed forecast models for the stock market using the error correction neural network. It does this by dividing the forecasting problem into a number of issues and then suggests solutions to these. Since the method is based on previous parts of the thesis, the details of different solutions is not included, instead the reader is referred to appropriate sections of the thesis.

The chapter also contains a verification of the method, in which forecast models for the Ericsson B and Atlas Copco B stocks are developed. These models are trained and evaluated in order to be able to evaluate the performance of the method.

7.1 Method Description

This method describes how neural network forecast models can be designed when forecasting the stock market using the error correction neural network. This is accomplished by dividing the forecast problem into a number of issues that needs to be solved and suggest possible solutions to these in a suitable order. It is based on the information gathered during the literature study, Chapters 2, 3, 4 and 5, and the empirical study, Chapter 6.

Target Selection

As a first step, the stock that is to be forecasted shall be selected, since it serves as the target for the neural network forecast model. Of the different available stock data, the closing price is used as the target series, since it represents a fixed value of the stock at a specific time. It is important to be aware of that the quality of the target series (i.e. rate of missing values, outliers, etc.) can affect the forecasting performance, since it is used by the error correction learning rule during the training process. In addition to this, the quality of the target series can also affect the evaluation process, since it often is used by different performance measures and benchmarks (e.g. realized potential and the naive benchmark etc.). This means that

target series with low quality can have a negative impact on the forecasting performance and evaluation procedure and should thus be avoided.

Input Selection: Long List

The next step is to select a long list of potential input variables to the forecast model, which is accomplished using expert knowledge or at least a best effort. These variables can include stock data (such as closing prices and volume), and fundamental data (such as indexes, interest rates, commodity prices and foreign exchange rates).

When selecting the variables, they should have some perceived relation to the target stock and company. In addition to this, an obvious limiting factor is the availability and cost of the data. Some examples of possible input variables can be seen in the list below.

- Stock data of the predicted stock.
- Stock data of companies in the same industry as the company whose stock is to be forecasted.
- Indexes of the same industry as the company whose stock is to be forecasted.
- Foreign exchange rates of currencies used by the company whose stock is to be forecasted.
- Prices of commodities used as raw material by the company whose stock is to be forecasted.

For more information, see Sections 2.2 and 2.7.1.

Training, Validation and Generalization Sets

In this step of the model selection, the data is explicitly divided into a training, a validation and a generalization set, as described in Section 2.8. This in order to ensure that the data in the generalization set is unable to affect the remaining steps of the model selection process, which is important since it represents unknown future values.

The sizes of these sets also needs to be determined and based on Walczaks (2001, [WALC01]) research, we propose to use no more than two years of daily data (corresponding to approximately 500 data points, if using e.g. weekly data) in the training set.

Determining the size of the validation set is a balance between training the network with data that is as close as possible in time to the generalization set and the ability to detect overfitting behavior during the training processes. In addition to this, it is important that the data in the validation set represents the state of the market during the time period covered by the generalization set. This in order for the stopping criterion (see Section 'Training and Forecasting') to be efficient when determining the stopping point during a training session. We propose to use expert knowledge of the market, or at least a best effort, to find the validation set that best reflects the stock during the time period when it is forecasted, but no larger than half the size of the training set (i.e. at most a year, or correspondingly 250 data points).

Our empirical study indicates that the size of the generalization set has no major impact on the performance of the forecasting model. In order to facilitate a more detailed evaluation of the forecast model, the generalization set should also be divided into several subsets, spanning no more than a year of daily data each (corresponding with 250 data points). This will enable the evaluation of the forecast model in relation to how far into the future the predictions are made.

For more information, see Sections 2.8 and 6.7.

Handling Missing Values

There are several ways in which missing values can be handled, we propose to use a rather simple strategy. When choosing the final input variables in the short list (see Section 'Input Selection: Short List'), time series with a high rate of missing values shall be avoided. This shall be done in combination with replacing missing values with the last known previous value.

For more information, see Sections 2.3.4 and 6.10.

Aggregating Data

In some cases there might be a wish to use aggregated data instead of the more common daily data, e.g. when making weekly forecasts. In Section 2.2.3 two different methods are discussed that can be used to aggregate daily data. When aggregating to weekly data, it is suggested to use the first method (i.e. the closing price of a chosen week day, the highest and lowest prices during the past week and the total volume during the past week becomes closing, high, low and volume for the weekly data).

Input Preprocessing

The next step is to perform preprocessing on the input time series in order to turn them into a more suitable format. There are two main issues that needs to be addressed by this preprocessing, one of them being the stationarity of the time series. When forecasting, a time series should at least be weakly stationary, which raw financial time series usually not are. Non-stationary financial time series can usually be turned into weakly stationary time series using a one step differencing, which is accomplished when deriving data using e.g. simple return, log-return or the force etc. In addition to this, the scale function can also be used with a sliding window technique in order to derive a weakly stationary time series from non-stationary financial time series.

The other issue concerns the range of the values in the time series, which should be scaled to suit the activation function that is used by the network. In the case of the error correction neural network it is the hyperbolic tangent function, for which task Grothmann (2004) determined that the scale function is suitable [GROT04].

In addition to these issues, steps should be taken to handle outliers in the data, since these can have a negative effect on the forecasting performance. The scale function, in addition to scaling the data, is also able to reduce the effect that outliers have on the forecasting performance.

We propose, based on Grothmanns (2004, [GROT04]) suggestion, that the scaled momentum (i.e. scaled simple return) and the scaled force should be used as preprocessing functions (both on each series).

For more information, see Sections 2.3, 2.4, 2.5 and 2.6.

Target Preprocessing

It is important to notice that preprocessing the target series is at least as important as the preprocessing of the input variables. This since the target series is used by the error correction learning rule during training of the network. Thus the same issues (i.e. stationarity, range and outliers) arises when preprocessing the target series as when preprocessing the input variables.

Usually the target series should be scaled to fit the range of the output neurons (i.e. its output range), but since the error correction neural network uses linear activation function, there is no obvious suitable range. Since the ECNN uses the errors of the previous predictions as additional inputs, which are scaled using the hyperbolic tangent function, one aim could be to scale the target series so that the range of the error values falls within the linear input region of this function. Also important to notice is that if the range of the target series is too small, this can cause numerical issues both for the ECNN, since it uses the error values as additional inputs, and the optimization algorithm used to train the network.

Another issue that needs to be addressed when preprocessing the target series is what the transformed target values represents (e.g. prices, returns, volatility etc.), since the output of the forecast model can be thought of as having the same transformation. Thus the transformation and scaling used on the target series should either be in the desired format or using functions that can be inverted (i.e. injective functions) in order to be able to transform the predictions into the desired format.

We propose to preprocess the target series using the log-return function, since it reduces the impact of outliers on the forecasting model, represents asset returns, transforms the target series into a weakly stationary series and is invertible. In addition to this, a target multiplier (i.e. a constant factor that the target series is multiplied with) shall be used in order to scale the transformed target series to a suitable range. The target multiplier can be determined through the training of several networks and then selecting the one that achieves the highest performance (hitrate) on the validation set. This should be done after the network architecture, number of neurons and training procedure have been selected, using the input variables in the short list (see following sections).

When determining the target multiplier to use, the networks shall be trained for a fixed amount of epochs, where the hitrate on the validation set is extracted for each epoch. This enables easy comparison between the performance of the networks with different target multipliers. The amount of epochs should be selected so it at the very least includes the epoch with the highest performance on the validation set (which also should include the epoch with the highest performance on the generalization set, although this is unknown during the selection process).

For more information, see Sections 2.4.1 and 6.9.

Input Selection: Short List

In this step, the final input variables in the short list are selected, following the procedure described in Section 2.7.1. This is accomplished by examining the rate of missing values, input-input and input-output correlation. The rate of missing values should be examined based on the raw time series, while the correlation should be determined based on the preprocessed time series. The input-target correlation should be determined using lagged values for the input series, where a suitable number of lagged values corresponds roughly with the unfolding size of the network architecture (see Section 'Network Architecture Selection'). Based on the information gathered by these three tests, the input variables in the long list can be ranked, where the best input variables are left in the short list.

We propose using the procedure described above to rank the input variables and determine which will be added to the short list as described below:

- High Input-Target Correlation and Low Rate of Missing Values

  These input variables have a high quality of the data and a strong observable relation to the target series and will receive a high ranking. Thus these variables will be added to the short list.

- High (Low) Input-Target Correlation and High (Low) Rate of Missing Values

  These input variables have a low (high) quality and a high (low) observable relation to the target series and will thus receive a medium ranking. The question of adding them to the short list or not needs to be a balanced decision based on their ranking and of how many other better input variables there are that can be added to the short list.

- Low Input-Target Correlation and High Rate of Missing Values

  These input variables have a low quality of the data and a low observable relation to the target series and will thus receive a low ranking. These variables will not be added to the short list.

- High Input-Input Correlation

  When two or more input variables have a high correlation to each other, all but one of them shall be removed. The decision of which input variable to keep will be based on their internal ranking, determined as described in the three previous items.

For more information, see Sections 2.3.4 and 2.7.1, and Appendix A.

Network Architecture Selection

The error correction neural network, as described in Chapter 4, will be used to perform the forecasts. When using the ECNN there is a need to determine the amount of unfolding and overshooting time steps, in addition to the number of neurons in the state clusters, that will be used. It is also important to notice that the error output neurons have the desired target values of zero.

When selecting the architecture (i.e. number of unfolding and overshooting time steps), there are two different approaches, using a general or a selected architecture. The general architecture is determine using expert knowledge or at least a best effort, while the selected architecture is determined through using the maximum inter-temporal connectivity and error levels of future predictions (see Sections 'Maximum Inter-temporal Connectivity (MIC)', 'Unfolding' and 'Overshooting').

We propose to use a general architecture with approximately seven unfolding and six overshooting time steps. It is also possible to use the selected architecture, however our empirical study indicates that this does not improve the forecasting performance. In addition to this, we propose to train several networks with different numbers of neurons, and then select the number that achieves the highest performance (hitrate) on the validation set to the final forecast model. Our empirical study also indicated that the number of neurons should be selected using the network architecture that will be in the final forecast model (i.e. the number of unfolding and overshooting time steps).

When determining the number of neurons to use, the networks shall be trained for a fixed amount of epochs, where the hitrate on the validation set is extracted for each epoch. This enables easy comparison between the performance of the networks with different numbers of neurons. The amount of epochs should be selected so it at the very least includes the epoch with the highest performance on the validation set (which also should include the epoch with the highest performance on the generalization set, although this is unknown during the selection process).

For more information, see Sections 6.5, 6.6 and 6.8.

Maximum Inter-temporal Connectivity (MIC)

In order to determine the amount of unfolding in time that should be used in the selected architecture of the ECNN, the maximum inter-temporal connectivity needs to be determined. This can be accomplished following the procedure described in Section 4.1.1.

This procedure entails first guessing the the amount of unfolding (i.e. the number of time steps included in the unfolded architecture) and then training the network using a stopping criterion (see Section 'Training and Forecasting'). The maximum inter-temporal connectivity can then be determined by looking at the error levels on the training set of the outputs that corresponds with different points in time.

Unfolding

As described in Section 4.1, the amount of unfolding (i.e. how many historic values) that should be included in the selected architecture of the ECNN is determined by the maximum inter-temporal connectivity (i.e. the amount of unfolding is the same as the MIC), which was determined in the previous step of the model selection.

Overshooting

The next step is to determine the amount of overshooting (i.e. how many future predictions) that should be included in the selected architecture of the ECNN. This is accomplished using the procedure described in Section 4.2.

This procedure entails an iterative search for the optimum overshooting architecture through the training of the networks until convergence using a stopping criterion (see Section 'Training and Forecasting'). At this point the error levels on the training set are evaluated to determine if the optimum overshooting architecture has been found (signs of overfitting, worsening error levels or inability to learn a future prediction).

Thick Modeling

The basic idea of thick modeling, in relation to forecasting, is to include several forecast models and then choose the output of the thick model from these, using a selection criterion. There are several different approaches to selecting the output, e.g. a single forecast model, the average of all included forecast models or a weighted sum of all included forecast models can be used as the thick model's output. Based on our empirical study, using thick models seems to limit both the lowest and highest level of performance and can thus be seen as a safer choice than when not using it.

We propose to use a selection criterion that chooses a single forecast model using the performance measure hitrate. At least five different forecast models should be included, using the same configuration but with different initial weights for each training session.

For more information, see Sections 3.6 and 6.13.

Learning Rule

The next step in the model building process is to determine which learning rule to use when training the network. The error correction neural network uses the error correction learning rule, which includes the error back-propagation algorithm with a shared weight extension. In addition to the back-propagation algorithm, an optimization algorithm, an error function and a step length needs to be selected.

When selecting an optimization algorithm, care should be taken to ensure that the numerical problems that might occur in the back-propagation algorithm, caused by the long chains of hyperbolic tangent functions in the error correction network, are avoided. The choice of step length is closely related to the selected optimization algorithm and can be set either statically or using a line search method. Also note that the choice of target multiplier affects the optimum step length. Concerning the error function, it is advantageous to choose one that has the ability to reduce the impact of outliers on the training process.

We propose to use the vario-eta optimization algorithm, which is able to handle the numerical issues of large networks using the hyperbolic tangent function, in combination with the $\ln\cosh$ error function to suppress the effect of outliers in the data. The learning rate is set statically to a relatively large size at the beginning of the learning procedure, which suits the vario-eta algorithm. The step size can then be reduced as the training progresses in order to increase the chance of finding the optimum solution.

For more information, see Section 3.3.

Pattern Presentation

In this step a decision is made regarding how the patterns in the training set are to be presented to the network. The patterns can be presented using either the stochastic or the batch procedure. The stochastic procedure presents the patterns one by one, randomly selected from the training set. In contrast, the batch procedure presents the patterns in batches to the network, selecting the patterns from the training set randomly (with replacement), randomly (without replacement, permute) or sequential.

We propose to use the batch procedure with a batch size of twenty patterns that are selected randomly without replacement (permute) from the training set.

For more information, see Section 3.3.6.

Weight Initialization

The next step is to find a set of initial weight values. This can be done in one of two ways, using previous determined weight values (e.g. weights determined in a pervious training session of the network) or randomly selected within some range.

We propose to initialize the weights randomly within the ranges described in the list below. Let N be the number of state neurons used in one state cluster and M the number of input variables.

- State neurons:
$$\left[-\frac{1}{\sqrt{M+1}}, \frac{1}{\sqrt{M+1}}\right]$$
- Error output neurons:
$$\left[-\frac{1}{\sqrt{N+1}}, \frac{1}{\sqrt{N+1}}\right]$$
- Output neurons:
$$\left[-\frac{1}{\sqrt{N}}, \frac{1}{\sqrt{N}}\right]$$

For more information, see Section 3.3.7.

Overfitting

Overfitting is a problem that occurs when too complex networks are used during forecasts, which means that they are able to learn the training data by heart, including the noise. This leads to a loss of generalization performance and can be observed during training since the error level for the validation set starts to increase, while it still is decreasing for the training set.

Steps should be taken in order to increase the network's ability to resist the problem of overfitting. There exists several different techniques that can be used, some of which can be seen in the list below:

- Larger training sets.
- Early Stopping with Weight Pruning: Section 3.5.1.
- Late Stopping with Weight Pruning: Section 3.5.2.
- Cleaning with Noise: Section 3.5.4.

Our empirical study indicates that using the cleaning with noise technique leads to no increase in performance worth mentioning.

For more information, see Sections 3.4 and 3.5.

Training and Forecasting

At this point, the forecast model is ready to be trained, thus producing the forecasts. The forecast model is trained until the stopping point is reached, which is determined by a stopping criterion.

We propose to use a stopping criterion based on the error level for predictions made one time step into the future. The training should be stopped within at least five epochs to the one with the lowest error level on the validation set. If the stopping point occurs after only a few epochs, the forecast model shall be retrained, if necessary with different initial weights.

For more information, see Sections 6.5 and 6.13.

Performance Measure Selection

The next step is to select a number of performance measures which can be used to determine the accuracy of the forecasts. These performance measures can also be used in combination with different benchmarks (see Section 'Benchmark Selection'). There are a number of different performance measures, some of which can be seen in the list below. For more detailed information, see Section 5.2.

- Root Mean Squared Error (RMSE): Section 5.2.1
- Hit Rate: Section 5.2.2
- Return on Investment: Section 5.2.4
- Realized Potential: Section 5.2.5
- Gross Return: Section 5.2.6
- Annualized Return: Section 5.2.7

The RMSE performance measure is used when the number of unfolding and overshooting time steps are determined for the selected architecture.

When calculating the return on investment or the gross return for a stock, the trading strategy using long and short positions should be used.

Benchmark Selection

When the performance measures have been picked, the next step is to select a number of benchmarks. These benchmarks are used to give the performance measures a context. This since a forecast model with a low performance still might outperform all available alternative models. It is important to make sure that the performance measures that are compared represents the same time period. Some benchmarks that can be used are listed below, for more information see Section 5.3.

- Naive Prediction: see Section 5.3.1

  The naive prediction is a relatively simple and commonly used benchmark.

- Alternative Forecasting Model

  When predicting the same target series using different forecasting models, their performance measures can be compared in order to determine which of the models that achieves the highest performance. This is useful when there is a desire to examine how small changes in a forecast model affects the forecasting performance.

- Index

  When using the return on investment or gross return performance measures, it might be a good idea to compare them to an index, which in general indicates the average performance of a certain market, industry, etc. When calculating the return on investment or the gross return for the index, a buy and hold strategy should be used.

Evaluation

The performance of the forecasting model is finally evaluated on the generalization set, using a number of performance measures and benchmarks. The model should also be checked for obvious mistakes (e.g. off-by-one errors, data snooping etc.). In addition to this, detecting signs of overfitting behavior can also be considered as a part of the evaluation process.

We propose to use the following performance measures; hitrate, return on investment, realized potential and annualized return. The return on investment shall be used in combination with the index benchmark in order to compare the forecasting performance to how the market evolved, which can easily be visualized using a graph. The naive prediction benchmark can be used for comparisons of the forecasting hitrate. The forecast model shall also be evaluated for signs of overfitting behavior.

For more information, see Chapter 5.

7.2 Verification of Method

The aim of this test is to investigate the performance and validity of the developed method (see Section 7.1). During this investigation two forecast models are developed, based on the method described in Section 7.1, in order to forecast the Swedish stocks Ericsson B and Atlas

Copco B. These stocks have intentionally been selected so that they have not been included in any of the previous tests performed in order to increase the validity of this investigation. The long and short lists of the selected input variables can be seen in Appendix D (including how they are preprocessed), while the time periods covered by their training, validation and generalization sets can be seen in Table 7.1.

Training, Validation and Generalization Sets						
	Training		Validation		Generalization	
	Starting	Ending	Starting	Ending	Starting	Ending
Ericsson B	2001-02-01	2003-06-30	2003-02-01	2003-06-30	2003-07-01	2005-06-30
Atlas Copco B	1990-01-01	1992-06-30	1992-01-01	1992-06-30	1992-07-01	1994-06-30

Table 7.1: This table shows the time spans that are covered by the training, validation and generalization sets when forecasting the Ericsson B and Atlas Copco B stocks.

The number of neurons and the target multiplier are determined, as described in the method, using the general architecture. The forecast models are trained five separate times, using different initial weights, and then used as input to a thick model. The performance of the thick model is analyzed using different performance measures and benchmarks, the results of which can be seen in the following two sections.

The forecast model is configured, generated and evaluated using the 'Forecast Model Generator GUI' (see Section 6.3), while the SENN application (see Section 6.2) is used to train them. The input and target series are selected from the database described in section 6.1.

7.2.1 Ericsson B

In Figure 7.1, the results from when the number of neurons and the target multiplier were determined can be seen. It can easily be determined from this figure that the forecast model shall use fifteen neurons and a target multiplier with the constant factor of twenty.

In Table 7.2, the performance of the forecast model for the Ericsson B stock can be seen (the rows 'Generalization Set' and the different 'Periods'), in addition to the maximum and mean hitrates of the five training sessions (notice that the hitrate is calculated using Equation 5.3). The realized potential of the naive predictions, in addition to a comparison between its hitrate and that of the forecast model (see Equation 5.11), can also be seen.

Finally, in Figure 7.2, the return on investment generated by the Ericsson B forecast model can be seen, including the NASDAQ Composite benchmark index.

7.2.2 Atlas Copco B

In Figure 7.3, the results from when the number of neurons and the target multiplier were determined can be seen. It can easily be determined from this figure that the forecast model shall use three neurons and a target multiplier with the constant factor of twenty.

In Table 7.3, the performance of the forecast model for the Atlas Copco B stock can be seen (the rows 'Generalization Set' and the different 'Periods'), in addition to the maximum

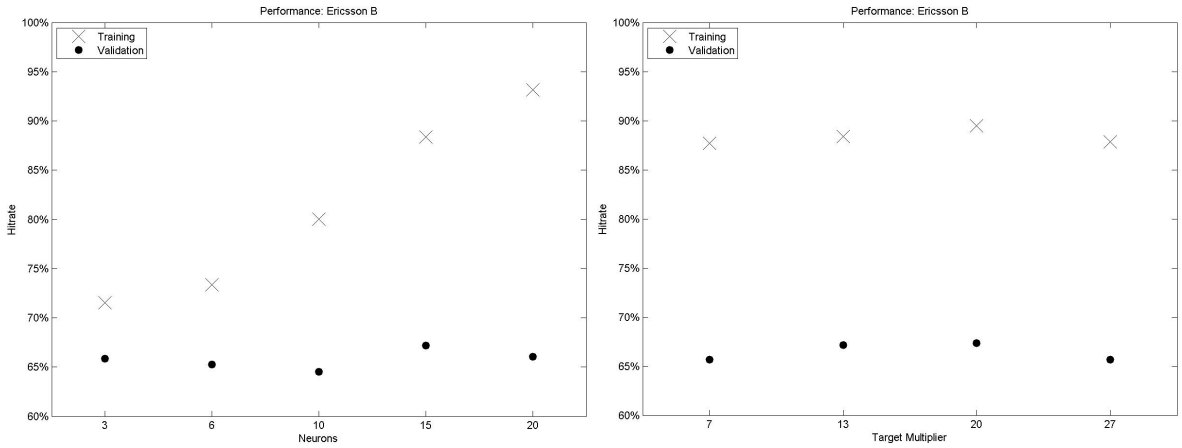

Figure 7.1: This figure shows the performance for the Ericsson B stock when different number of neurons and target multipliers are used.

Performance: Ericsson B			
	Hitrate	RP	AR
Max:	62.28%	14.73%	
Mean:	56.94%		
Generalization Set:	57.83%	4.28%	51.03%
Naive Prediction:	1.02	-2.67%	
Period 1:	56.82%	-0.98%	56.20%
Period 2:	59.23%	2.02%	88.81%
Period 3:	57.58%	17.51%	44.53%
Period 4:	57.72%	-1.44%	22.06%

Table 7.2: This table shows the performance of the forecast model, developed using the 'Method to Select a Forecasting Model Based on ECNN' method, for the Ericsson B stock. Each period corresponds with half a year of data in the generalization set, in the same order in which they are listed.

and mean hitrates of the five training sessions (notice that the hitrate is calculated using Equation 5.3). The realized potential of the naive predictions, in addition to a comparison between its hitrate and that of the forecast model (see Equation 5.11), can also be seen.

Finally, in Figure 7.4, the return on investment generated by the Atlas Copco B forecast model can be seen, including the NASDAQ Composite benchmark index.

7.2.3 Conclusions

From the verification of the method, it can be seen that the developed method was applied successfully to a forecast problem when designing the forecast models for the Ericsson B and Atlas Copco B stocks. The results in Tables 7.2 and 7.3 indicates that the forecast models are able to predict the stocks with some level of accuracy. However, the most interesting

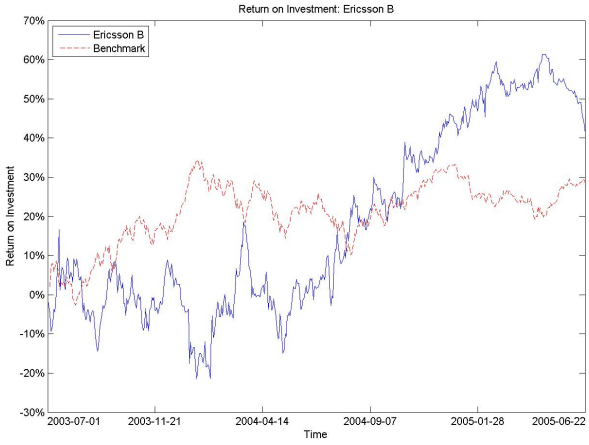

Figure 7.2: This figure shows the return on investment when using the forecast model for the Ericsson B stock. In addition to this the index benchmark NASDAQ Composite can be seen.

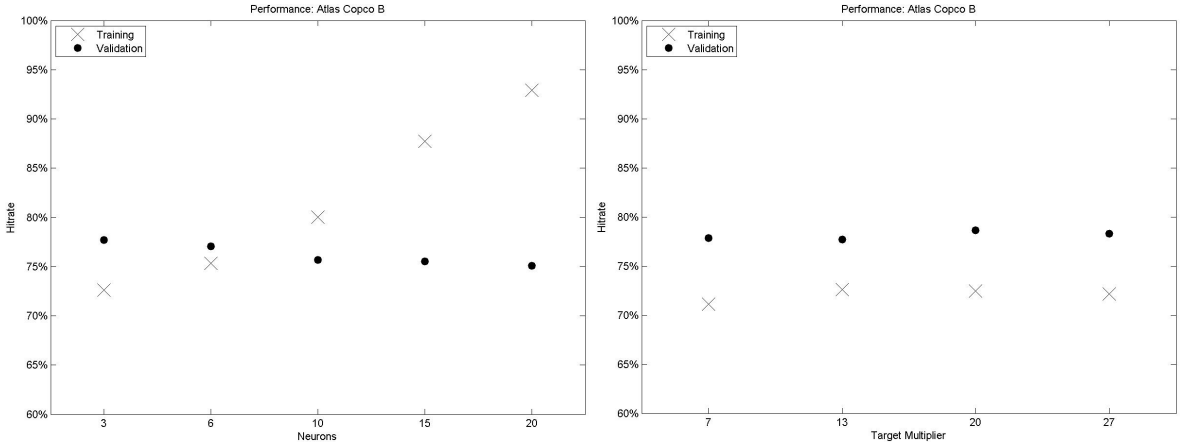

Figure 7.3: This figure shows the performance for the Atlas Copco B stock when different number of neurons and target multipliers are used.

result, which also can be observed for the 'Verification of Findings: Forecasting' test (see Section 6.13), is that none of the models lead to loses of money. These results must be considered a success since a first goal when investing must be not to lose money, which the method seems to achieve very well.

If the first goal when investing is not to lose money, a close second goal is to maximize the earnings. Although the method produces forecast models that are able to earn money, looking at the realized potentials, a slightly higher performance might be desirable.

To summarize, the developed method fulfills the objective of being a tool that can be used when developing forecast models based on the error correction neural network in the stock market. In addition to this, it provides a solid foundation when investigating the problem of forecasting. Since it divides the problem into several issues, the method is easy to build on or modify, in order to generate even better performing forecast models.

Performance: Atlas Copco B			
	Hitrate	RP	AR
Max:	60.66%	13.44%	
Mean:	59.11%		
Generalization Set:	59.11%	11.28%	21.14%
Naive Prediction:	1.15	25.05%	
Period 1:	56.06%	19.32%	40.16%
Period 2:	55.04%	-1.67%	-8.75%
Period 3:	63.64%	4.48%	19.39%
Period 4:	61.79%	18.09%	41.04%

Table 7.3: This table shows the performance of the forecast model, developed using the 'Method to Select a Forecasting Model Based on ECNN' method, for the Atlas Copco B stock. Each period corresponds with half a year of data in the generalization set, in the same order in which they are listed.

Figure 7.4: This figure shows the return on investment when using the forecast model for the Atlas Copco B stock. In addition to this the index benchmark NASDAQ Composite can be seen.

Chapter 8

Conclusions

The aim of this thesis was to investigate if it is possible to forecast the stock market and if so, develop a method that can be used when designing forecast models based on the error correction neural network. This was accomplished by performing a literature and an empirical study focused on the different aspects of the forecasting problem. Based on these studies the method 'Method to Select a Forecasting Model Based on ECNN' was developed, which can be seen as the main result of this thesis. This method can be found in Chapter 7.

In addition to incorporating the more important conclusions that were drawn during the empirical study in the method, each test is also accompanied with a conclusions section, which summarizes the results of the test in question. These conclusions will not be repeated in this section, instead the reader is referred to the empirical study found in Chapter 6.

In this thesis, five different forecast models were designed and evaluated based on the method, one for each of the Volvo B, Scania B, SE Banken A, Ericsson B and Atlas Copco B stocks. A summary of the performance achieved by these forecast models can be seen in Table 8.1.

Forecast Performance					
	Forecast Model			Naive Prediction	
	Hitrate	RP	AR	HQ	RP
Volvo B	58.61%	13.39%	20.90%	1.16	-3.76%
Scania B	55.13%	7.49%	23.79%	1.11	5.39%
SE Banken A	63.57%	12.24%	54.30%	1.13	10.78%
Ericsson B	57.83%	4.28%	51.03%	1.02	-2.67%
Atlas Copco B	59.11%	11.28%	21.14%	1.15	25.05%

Table 8.1: This table summarizes the performances of the five different forecast models. The hitrate, realized potential and annualized return can be seen for the different forecast models, while the hitrate quotient (see Equation 5.11) and realized potential can be seen for the naive prediction benchmark.

As mentioned previously in this thesis, the Efficient Market Hypothesis (see Section 2.1) suggests that it is impossible to predict stocks. There are however some research that contra-

dicts this hypothesis mainly referring to a time difference, that occurs between an event and information of this event reaching the whole market, that can be exploited.

Even though the results from the five forecast models (see Table 8.1) are not exceptionally good, there are two circumstances that suggests that it is possible to predict stocks. The first one being the fact that none of the forecast models generates negative results (i.e. loses money). Where it can be argued that a single successful forecast might be due to pure chance, the fact that all of the forecast models generated positive results (i.e. earned money) should be considered a strong indication that stocks can be forecasted. The second circumstance is that a majority (all but the realized potential of the Atlas Copco B stock) of the forecast models outperformed both the naive prediction benchmark and the index benchmark. Thus, the results of this thesis indicates that stocks can be forecasted successfully, contradicting the efficient market hypothesis.

When investing in the stock market, a first goal must be not to lose money which, as discussed previously, none of the forecast models did. A closely following second goal is to maximize the earnings of the investment and, even though all the forecast models generated positive results, this leaves some to be desired of the method's performance. Worth noting is that the forecast models for the Volvo B, Scania B and Ericsson B stocks used data from a time period during which the market might have been slightly harder to forecast than normally. This since the so called 'dot-com bubble' collapsed in the beginning of the 21th century. To summarize, the results of this thesis indicates that the developed method can successfully be applied when forecasting stocks using the error correction neural network. Thus it fulfills the objective of being a tool that can be used when developing forecast models based on the error correction neural network in the stock market.

There is still an issue with the method that needs to be addressed concerning how the amount of unfolding and overshooting time steps in the forecast model should be determined. In addition to this issue, there are also suggestions of further work in Section 8.1 that might improve the performance of the method. There are two suggestions of further work that might be especially interesting to investigate further, one of them being the state space reconstruction extension to the error correction neural network. The other suggestion concerns using weekly data instead of daily data in the forecast models. In any case the method provides a solid foundation that can be built on or used as an introduction to the problem of forecasting stocks using neural networks.

8.1 Further Work

An obvious extension to the work in this thesis is to further investigate the performance of the developed method and how it generalizes when used to forecast additional stocks. In addition to this, the sections below describes further work that can be performed in order to improve the method.

Input Selection

When the literature study was performed it became obvious that little research had been performed in the area of how to select the input variables of a forecast model based on neural networks. Most research found were aimed at linear models, where the correlation coefficient often is used to detect linear relations between input and target variables. Since neural networks can exploit non-linear relations, the correlation measure might be insufficient.

Another issue that needs to be addressed is how to determine the number of input variables to use, which is a balance between the additional noise of an extra variable and the relevant information that it provides. One approach that could simplify this issue is using a bottleneck network in combination with the ECNN to perform principle component analysis. Thus a large number of input variables can be used, from which the bottleneck network extracts a lower dimensional set of principle components. These principle components can then be used as input to the neural network that performs the actual predictions (i.e. the error correction neural network).

It might be suitable to use non-linear principle component analysis in combination with neural networks. This might even prove to be a good supplement to the correlation measure, since it should be able to remove principle components with low non-linear relations.

For more information concerning principle component analysis when using neural networks, see e.g. McNelis (2005) [MCNE05] and Reed et al. (1999) [REED99].

Training, Validation and Generalization Sets

Our empirical study indicated that using short forecasting periods did not result in any noteworthy increase in generalization performance. However, this can be investigated further since we focused on relatively long forecast periods, half a year of generalization data. It might be interesting to investigate the network's performance during only, for example, a week of generalization data. This might even provide some insight to the workings of the error correction neural network.

Since the ECNN solves the forecasting problem through viewing it as a system identification task, it might be expected that the performance should be evenly distributed over a time period where the state of the market is fairly similar (i.e. the training, validation and hopefully the generalization set). Thus, if the state of the market is similar during a longer period, using a shorter forecasting period should (in general) not lead to any higher performance. It might also be interesting to compare the difference in performance when using long and short forecasting periods, between the error correction neural network and a common multi-layered feedforward network.

Handling Missing Values

It is no doubt that the quality of the data, e.g. the rate of missing values, can have a great impact on the performance of the forecast models. In the developed method, the rather simple strategy of replacing the missing values with the last known pervious value is used. There are several ways in which the handling of missing values can be improved, one way being

the use of weekly time series aggregated from daily data (see Section 2.2.3). This way, the closing prices of the weekly series can be chosen from the daily series in such a way that the missing values are minimized (i.e. taking the closing price of the day which contains the fewest missing values). Another approach could be to use Kalman filters or Markov Chain Monte Carlo (MCMC) methods, for more information see e.g. Tsay (2005) [TSAY05].

It could be argued that, since the error correction neural network tries to identify the dynamics of the training data, a sound approach when replacing the missing values is to do so in such a way that the dynamics of the non-missing data are preserved. It is also important to make sure that the evaluation process is not biased as a result of the missing values or their replacement values.

However, the obviously best approach is to only use data of the highest quality, lacking missing values altogether.

Aggregating Data

Previously performed theses suggests that forecasting weekly instead of daily time series might lead to a higher performance, see e.g. Nygren (2004) [NYGR04] and Kurtagic (2001) [KURT01]. Since this thesis focuses on daily data, using aggregated weekly time series can be investigated further. This would entail finding the optimum method to use when aggregating daily time series into weekly series, and then examining whether the developed method ('Method to Select a Forecasting Model Based on ECNN') can be applied successfully for these weekly time series.

The impact of the so called 'day of the week' effect should also be investigated and steps taken to incorporate this knowledge in the aggregation method. In addition to this, the daily time series should be aggregated in such a way that the derived weekly time series have the highest possible quality.

For more information concerning the 'day of the week' effect, see e.g. Hellström (1998) [HEL98a].

Network Architecture Selection

One major issue in the developed method is the determination of the optimum amount of unfolding and overshooting time steps in the forecast model. The empirical study indicates that the maximum inter-temporal connectivity, in extension the amount of unfolding time steps, is hard if not impossible to determine when forecasting the stock market. Instead the method suggests using a general architecture where the amount of unfolding and overshooting time steps are determined trivially. This part of the method needs to be improved, which can be accomplished using one of two approaches.

The first approach is to investigate the concept of the MIC and the methods used to determine the number of unfolding and overshooting time steps, as described in Sections 4.1 and 4.2, further.

The alternative approach is to develop a new method that can be used to determine the optimum amount of unfolding and overshooting time steps. A first step could be to determine a maximum and minimum amount of unfolding and overshooting time steps, after which all

possible combinations of these are tested. The combination that leads to the highest performance can then be used in the forecast model. This would entail investigating how the maximum and minimum amount of unfolding and overshooting time steps should be determined, which performance measure used when determining the optimum combination (of unfolding and overshooting time steps) and on which set this measure should be applied.

In addition to determine the optimum amount of unfolding and overshooting time steps to use, there are also some interesting extensions to the ECNN architecture (see Section 4.3.1). Especially interesting is the state space reconstruction extension, since an investigation performed by Grothmann (2004, [GROT04]) suggests that it can improve the forecasting performance of the error correction neural network.

Thick Modeling

The developed method incorporates thick modeling in the forecast model, using the rather simple selection criterion of choosing the network, out of at least five different networks, that has the highest hitrate on the validation set. Thus the inclusion of a more complex thick model in the method can be investigated further. This would entail examining different selection criterions, whether to allow the output to be selected from different networks for each pattern and if the included networks should have different architectures (e.g. different amounts of unfolding steps, overshooting steps, neurons etc.).

In addition to this, since the ECNN produces future predictions for each overshooting time step, it would be very interesting to base the selection criterion partly on these additional predictions.

Training and Forecasting

A subject that can be investigated further is how to select and apply the step size, used by the optimization algorithm during the training session, and how this affects the performance of the forecast model. This should include determining the optimum step size, whether to train the network in several steps using different step sizes and if a line search algorithm should be used.

Evaluation

In this thesis, only the predictions made one step into the future are evaluated. Since the ECNN produces future predictions for each overshooting time step used by the network architecture, it might be interesting to investigate the performance of these additional predictions.

Another aspect that would be very interesting to investigate further, from a real world point of view, is the transaction costs. The simplest way would be to count the number of times that the the trading rule changes between the short and long positions. A more real world approach would be to apply a penalty cost each time this change occurs. It would be especially interesting to compare the number of transactions, and the influence of transaction costs on the results, between daily and weekly forecast models. If the transaction costs are

high, the possibility to modify the trading rule in order to reduce these costs might also be examined.

References

[BAIN96] Bain William G., *Investment performance measurement*, Woodhead, 1996

[CORN07] Cornuejols Gerard and Tütüncü Reha, *Optimization Methods in Finance*, Cambridge University Press, 2007

[DANI99] Daniels H. and Kamp B., *Application of MLP Networks to Bond Rating and House Pricing*, Neural Computing & Applications, (1999)8, p. 226-234, Springer-Verlag London, 1999

[ENGE02] Engelbrecht Andries P., *Computational Intelligence - An Introduction*, John Wiley & Sons, South Africa, 2002

[FAUS94] Fausett Laurene, *Fundamentals of Neural Networks Architectures, Algorithms, and Applications*, Prentice-Hall, 1994

[FREN02] Frennelius Arne, *Sannolikhetslära och statistisk inferens för tekniska utbildningar*, 5:e upplagan, Västerås Statistikutbildning, Sweden, 2002

[GROT04] Grothmann Ralph, *Multi-Agent Market Modeling Based On Neural Networks*, Siemens, 2004

[HAYK94] Haykin Simon, *Neural Networks. A Comprehensive Foundation*, Macmillan College, New York, 1994

[HEL98a] Hellström Thomas, *A Random Walk through the Stock Market*, Umeå University, Sweden, 1998

[HEL98b] Hellström Thomas and Holmström Kenneth, *Predictable Patterns in Stock Returns*, Mälardalen University, Sweden, 1998

[HEL98c] Hellström Thomas and Holmström Kenneth, *Predicting the Stock Market*, Mälardalen University, Sweden, 1998

[KIJI03] Kijima Masaaki, *Stochastic Processes with Applications to Finance*, Chapman & Hall/CRC, 2003

[KURT01] Kurtagic Arman, *Stock Market Prediction with Error Correction Neural Networks*, Mälardalen University, Sweden, 2001

[MAKR98] Makridakis Spyros, Wheelwright Steven C. and Hyndman Rob J., *Forecasting Methods and Applications*, Third Edition, John Wiley & Sons, 1998

[MCNE05] McNelis Paul D., *Neural Networks in Finance*, Elsevier, 2005

[NEUN98] Neuneier Ralph and Zimmermann Hans Georg, *How to Train Neural Networks*, Siemens, 1998

[NYGR04] Nygren Karl, *Stock Prediction - A Neural Network Approach*, Royal Institute of Technology, Sweden, 2004

[PISS02] Pissarenko Dimitri, *Neural Networks For Financial Time Series Prediction: Overview Over Recent Research*, BSc (Hons) Computer Studies, 2001-2002

[REED99] Reed Russell D. and Marks Robert J. II, *Neural Smithing - Supervised Learning in Feedforward Artificial Neural Networks*, Massachusetts Institute of Technology, 1999

[RICE07] Rice John A., *Mathematical Statistics and Data Analysis*, Third Edition, Thomson Brooks/Cole, 2007

[SIEK99] Siekmann Stefan, Neuneier Ralph, Zimmermann Hans Georg and Kruse Rudolf, *Neuro Fuzzy Systems for Data Analysis*, Siemens, 1999

[STRA01] Strandberg Fredrik, *Tails and outliers in financial time series*, Royal Institute of Technology, Sweden, 2001

[TIET08] Tietz Christoph, *SENN V3.1 User Manual*, Siemens, 2008

[TSAY05] Tsay, Ruey S., *Analysis of Financial Times Series*, Second Edition, Wiley-Interscience, 2005

[WALC01] Walczak Steven, *An Empirical Analysis of Data Requirements for Financial Forecasting with Neural Networks*, Journal of Management Information Systems, Vol. 17, No. 4, p. 203-222, 2001

[WEIW90] Wei William W. S., *Time Series Analysis - Univariate and Multivariate Methods*, Addison-Wesley, 1990 (1994)

[WONN90] Wonnacott Thomas H. and Wonnacott Ronald J., *Introductory Statistics For Business and Economics*, Fourth Edidtion, John Wiley & Sons, 1990

[YANG07] Yang Hui-Ling and Wu Wei-Pang, *Forecasting New Taiwan Dollar/United States Dollar Exchange Rate Using Neural Network*, The Business Review, Cambridge, Vol. 7, Num. 1, 2007

[ZIMM00] Zimmermann Hans-Georg, Neuneier Ralph and Grothmann Ralph, *Modeling of Dynamic Systems by Error Correction Neural Networks, Modeling and Forecasting Financial Data - Techniques of Nonlinear Dynamics*, Kluwer Academic, 2000

Appendix A

Formulas

This appendix briefly describes important mathematical formulas, used when performing preprocessing on the input data and evaluating the results of a forecast model.

A.1 Mean

If $\{x_1,...,x_T\}$ is a random sample of X, a random vector, then the (sample) mean is defined in Equation A.1 [TSAY05, WEIW90].

$$\mu = \frac{1}{T}\sum_{t=1}^{T} x_t \tag{A.1}$$

A moving average (using the past n values) of a time series $\{x_1,...,x_T\}$ is defined in Equation A.2 [TIET08].

$$\mu_{t,n} = \frac{1}{n}\sum_{i=0}^{n-1} x_{t-i} \tag{A.2}$$

A.2 Variance

If $\{x_1,...,x_T\}$ is a random sample of a random vector and where μ is the (sample) mean (see Equation A.1), then the variance is defined in Equation A.3 [TSAY05].

$$\sigma^2 = \frac{1}{T-1}\sum_{t=1}^{T} (x_t - \mu)^2 \tag{A.3}$$

A moving variance (using the past n values) of a time series $\{x_1,...,x_T\}$ is defined in Equation A.4 [TIET08].

$$\sigma^2_{t,n} = \frac{1}{n-1}\sum_{i=0}^{n-1} (x_{t-n} - \mu_{t,n})^2 \tag{A.4}$$

A.3 Covariance

Let $\{x_1,...,x_T\}$ and $\{y_1,...,y_T\}$ be two random samples of X and Y (two random vectors) and where μ_x and μ_y are the (sample) means (see Equation A.1), then the (sample) covariance is defined in Equation A.5 [CORN07, TIET08, TSAY05].

$$\Sigma_{x,y} = \frac{1}{T-1}\sum_{t=1}^{T}(x_t-\mu_x)(y_t-\mu_y) \tag{A.5}$$

A.4 Standard Deviation

If $\{x_1,...,x_T\}$ is a random sample of X (a random vector) and where μ is the (sample) mean (see Equation A.1), then the standard deviation is defined as the square root of the variance, as seen in Equation A.6 [RICE07, WONN90].

$$\sigma = \sqrt{\sigma^2} = \sqrt{\frac{1}{T-1}\sum_{t=1}^{T}(x_t-\mu)^2} \tag{A.6}$$

A moving standard deviation (using the last n values) of a time series $\{x_1,...,x_T\}$ is defined in Equation A.7.

$$\sigma_{t,n} = \sqrt{\frac{1}{n-1}\sum_{i=0}^{n-1}(x_{t-n}-\mu_{t,n})^2} \tag{A.7}$$

A.5 Correlation

Suppose that $\{x_1,...,x_T\}$ and $\{y_1,...,y_T\}$ are two random samples from X and Y (two random vectors), then the linear dependency between them is given by the correlation coefficient ρ (i.e. correlation measures the linear dependency) [TSAY05]. When $|\rho|$ is close to one it indicates a high correlation, and when it is close to zero a low correlation (the inequality $-1 \leq \rho \leq 1$ always holds) [FREN02].

When measuring the correlation between two different random samples, the (sample) cross-correlation equation can be used (see Equation A.8). When dealing with time series, there also exists the (sample) autocorrelation equation (see Equation A.9), which determines the correlation between lagged values of the same time series (i.e. x_t is compared to x_{t-l}) [HEL98a]. Also note that the cross- and autocorrelation can be modified with a sliding window technique (i.e. moving correlation) [TIET08].

A.5.1 Cross-Correlation

If $\{x_1,...,x_T\}$ and $\{y_1,...,y_T\}$ are two random samples of X and Y (random vectors) and μ is the (sample) mean, then the cross-correlation is defined in Equation A.8 [TSAY05].

$$\rho_{x,y} = \frac{\Sigma_{x,y}}{\sigma_x \sigma_y} = \frac{\sum_{t=1}^{T} (x_t - \mu_x)(y_t - \mu_y)}{\sqrt{\left(\sum_{t=1}^{T} (x_t - \mu_x)^2\right)\left(\sum_{t=1}^{T} (y_t - \mu_y)^2\right)}} \tag{A.8}$$

A.5.2 Autocorrelation

If $\{x_1, ..., x_T\}$ is a random sample of X (a random vector), l is the lag-value and μ is the (sample) mean, then the autocorrelation is defined in Equation A.9 [TSAY05].

$$\rho_l = \frac{\Sigma_{x_t, x_{t-l}}}{\sigma^2} = \frac{\sum_{t=1}^{T} (x_t - \mu)(x_{t-l} - \mu)}{\sum_{t=1}^{T} (x_t - \mu)^2} \tag{A.9}$$

Appendix B

Terminology

In this appendix commonly used terms are briefly described.

Neural Networks: Signals and Neurons

u_t	Neural network input cluster at time t (or pattern t)
y_t	Network output cluster at time t (or pattern t)
y_t^d	Desired output (target output) at time t (or pattern t)
s_t	Current state cluster (output from a hidden state neuron)
z_t	Error input cluster at time t (or pattern t)
h_m	Hidden layer cluster (m neurons)
θ	Bias term

Neural Networks: Functions

$NN(\cdot)$	Neural network function
n	Net input
$\varphi(\cdot)$	Activation function
z^{-1}	Unit delay operator
E_t	Network error function

Neural Networks: Training

w_t	Weight at time t
E	Error of a set of patterns $(1,...,T)$
η	Step length

Stock Data

V	Volume
P	Price (open, high, low indicated by indexes o, h, l)
y_t	Time series value at time t

Time Series: Derived Data

R_t	Asset return t
V_t	Volume rate of change or Gaussian volume (at time t)
σ_v	Volatility
T_t	Trend
F_t	Force

Time Series: Properties

μ	Mean
μ_t	Moving average
σ	Standard deviation
σ_t	Moving standard deviation
σ^2	Variance
σ_t^2	Moving variance
$\rho_{x,y}$	Cross-correlation
ρ_l	Autocorrelation

Performance Measures

HR	Hitrate
HR_s	Stopping hitrate
HR_n	Normalized hitrate
G	Gap
MWH	Mean Window Hitrate
T_t^g, T_t^{ls}	Trading strategies
ROI	Return on investment
RP	Realized potential
R^G	Gross return
AR	Annualized Return
$RMSE$	Root mean squared error

Appendix C

Glossary

Active Domain The active domain of a function includes the values in the domain, for which changes leads to noticeable changes in the function's value [ENGE02].

Activation Level The activation level refers to an input value for the activation function that results in a certain output. For example, when using the McCulloch and Pitts neurons, the threshold function has the activation level $u \geq 0$, which gives the output 1, and $u < 0$, which gives the output 0 [HAYK94].

Autonomous System An autonomous system is a system without any external inputs [GROT04]. Thus the current state of the system is only dependent on the previous states of the system (and not influenced by external inputs as is the case of dynamical systems, see below).

Batch A batch, when training neural networks, refers to a set of patterns from the training set, while true batch denotes a batch that contains all the patterns in the training set [TIET08]. The patterns in a batch are used during training of the network to compute weight changes. The weights are updated after all the patterns in the batch have been presented to the network.

Cluster A cluster represents a group of neurons, which have the same properties. This means that the neurons have the same amount of weights, are connected to the same neurons and have the same activation function.

Complexity of Neural Networks A neural network with a complex architecture has many free weights that needs to be estimated [MCNE05]. If a neural network is too complex, there is a risk that the network fits the noise in the data, instead of only learning the desired structures [GROT04]. However, too low complexity might lead to the neural network being unable to learn the underlying structures in the data [GROT04].

Credit Assignment Problem The credit assignment problem denotes the problem of assigning blame or credit to a set of internal decisions, that in unison resulted in an action that led to an improvement or worsening of a result [HAYK94]. This problem occurs when training neural networks with one or more hidden layers [GROT04]. There are however

several solutions to this problem, including the error back-propagation algorithm and finite unfolding in time [GROT04].

Dynamic System A dynamic system is a system for which the output is dependent on both the internal (autonomous part, previous states) and external influences (input), i.e. the current state is the result of the previous states and external inputs. [GROT04].

Epoch An epoch denotes a learning iteration, during which all the patterns in the training set are presented to the network, new weight values are calculated and the weights are updated. [ENGE02, REED99].

Lattice structure Another type of feed-forward neural networks is a lattice structure, where the output neurons are arranged in arrays (rows and columns) [HAYK94].

Long and Short positions In finance the difference between long and short positions is that for a long position the asset is actually owned, compared to a short position, where an asset is borrowed and then sold [TSAY05].

This has the effect that, when taking the long position, profit is earned if the asset has a positive rate of return. Opposite to this, when taking a short position, profit is earned if the asset has a negative rate of return.

Noise A time series that is noisy have disturbing influences included in the series. There is also a possibility to add (artificial) noise in order to avoid the problem of overfitting (see Section 3.5.4).

Pattern An input vector to a neural network is often referred to as a pattern (activation pattern), which is the information supplied to the input nodes of a neural network at some specific time [GROT04].

Penalty Term Penalty terms can be used in addition to the error function, during the training procedure, in order to force the network towards a preferable solution. Usually the preferred solution of a network training session is to minimize the network error (i.e. the difference between the network output and target value). The penalty term can thus be used in cases where the preferable solution cannot be expressed in terms of this error. [REED99]

Prediction Horizon A prediction horizon is a time period (future values) for which predictions are performed, e.g. daily, weekly, monthly and so on [GROT04].

Pruning Pruning is a technique used on neural networks in order to change its architecture. This is accomplished by removing either weights or nodes in the network, thus making it less complex. Since less complex networks are more resilient to the problem of overfitting, this technique is often used on networks that exhibits signs of overfitting during training.

Random Walk Process A random walk process can be described as a process where the direction of the next step is either up or down with some probability [KIJI03]. Mathematically a random walk process can be stated as $y_t = y_{t-1} + a_t$, where $\{y_t\}$ is a time series, y_0 is the starting value and $\{a_t\}$ is a white noise series (i.e. a sequence of random variables, which are independent and identically distributed (iid), with a finite mean and variance) [TSAY05].

Saturation Level Changes in values that exceeds the saturation level of a function gives no noticeable change in the function value.

Sliding Window Sliding window is a technique that can be used when computing certain properties of a time series, e.g. average, variance, correlation, etc. Usually these properties are computed using the entire time series, which results in a single value that is a measure of the property in question. When using a sliding window technique, the property is computed for a specific time period using only a subset of the time series. Suppose that w is the window size and that the property is to be computed for the time t, then the values that corresponds to the time period $t-w+1,...,t$ in the time series will be used. This computation is then repeated for the time $t+1,t+2,...$, which corresponds with the sliding part of the technique. This method will thus provide several measures of the property in question, each corresponding with a smaller part of the time series. If the property varies a lot for different parts of the time series, this will be more accurately reflected when using the sliding window technique.

Trading Costs Trading costs (transaction costs) are expenses that occurs when buying or selling, e.g. stocks, in financial markets.

Underfitting If the complexity of the neural network architecture (i.e. the number of free weights) is too low, it will be unable to learn the underlying function in the data, a situation referred to as underfitting. Important to note is that underfitting also can occur as a result of a too small training set (i.e. the network is unable to learn the underlying structure since it is not completely represented by the data in the training set). [REED99]

Universal Approximation The universal approximation theorem states that a continuous function f can be approximated using certain types of non-constant, bounded and monotonic increasing continuous functions $\varphi(\cdot)$, for details of these functions and the theorem see e.g. Haykin (1994). The theorem can be applied to the multi-layered perceptron networks (see Section 3.2.1) to prove their approximation abilities. [HAYK94]

Variants-Invariants Time variants are structures that changes over time, and thus needs to be predicted. Time invariants are the opposite, i.e. structures that are constant through time, hence needs no predictions. [GROT04]

Appendix D

Stock Selection

In this appendix the different input variables used in the empirical study (see Chapter 6) are listed in the following tables. The reduction to a short list was performed using correlation measures based on the momentum and force of the closing price series of the different stocks listed in the long list.

Volvo B: Long List		
Stocks		
Volvo A	Scania A	Haldex
Volvo B	Scania B	Bilia A
Volvo B (US)	Caterpillar Inc.	General Motors C
Saab B	Goodyear Tire &	
Indexes		
OMX	SX O-Listan	SX-2510 Bil *
DJ Industrials	OMX S30	SX-4010 Bank
Nasdaq Comp.	SX-20 Industri	SX-2030 Transport
S&P 500	SAX	SOX Generalindex
Interest Rates		
Sobl 5 år	Rta US Bond 30Y	SSV 90
Sobl 10 år		
Commodities		
Koppar	Aluminium	Gasolja
West Texas I.Oil		
Foreign Exchange Rates		
USD/SEK	EUR/SEK	

Table D.1: Volvo B Long List: The Volvo B long list is selected with forecasting of the Volvo B stock in mind, using best effort as described in Section 2.7.1. Data is available during the period 2001-01-05 through 2005-10-18. When applicable, *merged series.

Volvo B: Short List		
Stocks		
Volvo B	Scania A	Haldex
Volvo B (US)	Caterpillar Inc.	
Indexes		
OMX	SX O-Listan ***	SX-2510 Bil *
Nasdaq Comp. ***	SX-20 Industri	SOX Generalindex **
S&P 500 **		
Interest Rates		
Sobl 5 år		
Commodities		
West Texas I.Oil	Gasolja	
Foreign Exchange Rates		
USD/SEK	EUR/SEK	

Table D.2: Volvo B Short List. Data is available during the period 2001-01-05 through 2005-10-24. When applicable, *merged series, **only momentum, ***only force (otherwise both momentum and force).

Scania B: Long List		
Stocks		
Volvo A	Scania A	Haldex
Volvo B	Scania B	Bilia A
Volvo B (US)	Caterpillar Inc.	General Motors C
Saab B	Goodyear Tire &	
Indexes		
OMX	SX O-Listan	SX-2510 Bil *
DJ Industrials	OMX S30	SX-4010 Bank
Nasdaq Comp.	SX-20 Industri	SX-2030 Transport
S&P 500	SAX	SOX Generalindex
Interest Rates		
Sobl 5 år	Rta US Bond 30Y	SSV 90
Sobl 10 år		
Commodities		
Koppar	Aluminium	Gasolja
West Texas I.Oil		
Foreign Exchange Rates		
USD/SEK	EUR/SEK	

Table D.3: Scania B Long List: The Scania B long list is selected with forecasting of the Scania B stock in mind, using best effort as described in Section 2.7.1. Data is available during the period 2001-01-05 through 2005-10-18. When applicable, *merged series.

Scania B: Short List		
Stocks		
Volvo B	Scania B	General Motors C
Volvo B (US)	Caterpillar Inc.	Saab B
Indexes		
DJ Industrials **	SX O-Listan	SX-4010 Bank
Nasdaq Comp. ***	SX-20 Industri	SOX Generalindex ***
Interest Rates		
Sobl 5 år **	SSV 90 ***	
Commodities		
Aluminium	Gasolja	
Foreign Exchange Rates		
USD/SEK	EUR/SEK	

Table D.4: Scania B Short List. Data is available during the period 2001-01-05 through 2005-10-24. When applicable, **only momentum, ***only force (otherwise both momentum and force).

SE Banken A: Long List		
Stocks		
SE Banken A	Investor B	Kinnevik B*
SE Banken A****	Industrivärlden A	Midway Holding B
SE Banken C	American Express	
SE Banken C****	J.P. Morgan	
Indexes		
DJ Composite	OMX S30	AFV Bankindex
Nasdaq Composite	SX O-Listan	AFV Generalindex
S&P 500*	SX OMX	
Interest Rates		
UDS 30 Year Bond		
Commodities		
Guld	Silver	
Foreign Exchange Rates		
USD/SEK	USD/JPY	

Table D.5: SE Banken A Long List: The SE Banken A long list is selected with forecasting of the SE Banken A stock in mind, using best effort as described in Section 2.7.1. Data is available during the period 1989-10-13 through 1995-06-30. When applicable, *merged series, ****volume.

SE Banken A: Short List		
Stocks		
SE Banken A	SE Banken C	J.P. Morgan
SE Banken A****	American Express	
Indexes		
DJ Composite	Nasdaq Composite	S&P 500*
SX O-Listan	SX OMX	
Interest Rates		
UDS 30 Year Bond		
Commodities		
Guld	Silver	
Foreign Exchange Rates		
USD/SEK	USD/JPY	

Table D.6: SE Banken A Short List. Data is available during the period 1989-07-27 through 2006-06-27. When applicable, *merged series, ****volume series.

Ericsson B: Long List		
Stocks		
Ericsson A *	TeliaSonera	Tele 2 B
Ericsson B	Glocalnet	Ericsson (US)
Nokia A	Tele 2 A	
Indexes		
OMX	SAX	Nikkei 225
S&P 500	SOX Generalindex	SX-50 Telekom
SX O-List	DJ Composite	SX-4520 Teknologi
OMX S30	Nasdaq Telecom	
Interest Rates		
Rta US Bond 30Y	Sobl 5år	Sobl 10år
SSV90		
Commodities		
Koppar	Guld	Aluminium
Foreign Exchange Rates		
USD/SEK	EUR/SEK	JPY/SEK

Table D.7: Ericsson B Long List: The Ericsson B long list is selected with forecasting of the Ericsson B stock in mind, using best effort as described in Section 2.7.1. Data is available during the period 2001-01-05 through 2005-10-18. When applicable, *merged series.

Ericsson B: Short List		
Stocks		
Ericsson B	TeliaSonera	Nokia A
Ericsson (US)		
Indexes		
OMX	DJ Composite	Nikkei 225
OMX S30	Nasdaq Telecom	
Interest Rates		
SSV90 **	Sobl 5år ***	
Commodities		
Koppar	Guld	
Foreign Exchange Rates		
USD/SEK	EUR/SEK	JPY/SEK

Table D.8: Ericsson B Short List. Data is available during the period 2000-06-13 through 2005-10-24. When applicable, **only momentum, ***only force (otherwise both momentum and force).

Atlas Copco B: Long List		
Stocks		
Atlas Copco A	Sandvik	Avesta Sheffield
Atlas Copco A****	SSAB A	
Atlas Copco B	SSAB B	
Atlas Copco B****	Seco Tools B	
Indexes		
DJ Industrials	S&P 500*	SX OMX
NASDAQ Composite	SX O-Listan	AFV Generalindex
OMX S30	Nikkei 225	
Interest Rates		
UDS 30 Year Bond		
Commodities		
Guld	Aluminium	Koppar
Foreign Exchange Rates		
USD/SEK	USD/JPY	

Table D.9: Atlas Copco B Long List: The Atlas Copco B long list is selected with forecasting of the Atlas Copco B stock in mind, using best effort as described in Section 2.7.1. Data is available during the period 1989-08-28 through 1995-06-30. When applicable, *merged series, ****volume series.

Atlas Copco B: Short List		
Stocks		
Atlas Copco B	Atlas Copco A**	Avesta Sheffield
Atlas Copco B****	Sandvik	SSAB A
Indexes		
DJ Industrials	S&P 500*	SX OMX
NASDAQ Composite	SX O-Listan	
Interest Rates		
UDS 30 Year Bond		
Commodities		
Guld	Aluminium	
Foreign Exchange Rates		
USD/SEK	USD/JPY	

Table D.10: Atlas Copco B Short List. Data is available during the period 1989-08-25 through 2001-02-23. When applicable, *merged series, ****volume series.

Appendix E

Results: Potential of ECNN and Training Procedure

This appendix contains a more detailed description of the results obtained during the 'Potential of ECNN and Training Procedure' test, which can be seen in Section 6.5. Each table, one for each stock with and without the cleaning with noise technique, contains information concerning the maximum hitrates, stopping hitrates and gaps that were achieved during the test.

Volvo B								
	Highest Hitrate			Stopping on Lowest Error				
Neurons: 3	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	69.57%	66.98%	59.57%	56.32%	0.47%	57.06%	55.51%	5.460%
Test 2	68.22%	66.98%	59.57%	57.15%	0.51%	58.22%	56.48%	4.073%
Test 3	69.77%	69.81%	60.15%	56.16%	0.69%	57.06%	54.55%	6.635%
Test 4	66.47%	68.87%	59.38%	54.72%	1.21%	56.48%	52.61%	7.847%
Test 5	68.22%	67.92%	60.93%	58.80%	1.24%	60.35%	56.67%	3.492%
Mean	68.45%	68.11%	59.92%	56.63%				5.50%
Std	1.32%	1.23%	0.63%	1.49%				1.06%
Neurons: 6	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	74.22%	70.75%	58.80%	56.11%	0.61%	57.45%	55.51%	4.575%
Test 2	73.45%	68.87%	61.12%	59.22%	0.89%	61.12%	58.41%	3.107%
Test 3	72.67%	69.81%	60.73%	54.02%	1.05%	55.51%	52.22%	11.060%
Test 4	75.78%	66.98%	61.12%	59.89%	0.72%	60.93%	58.99%	2.014%
Test 5	72.48%	69.81%	62.28%	57.22%	1.07%	58.22%	54.93%	8.131%
Mean	73.72%	69.25%	60.81%	57.29%				5.78%
Std	1.34%	1.43%	1.27%	2.38%				2.30%
Neurons: 10	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	81.20%	70.75%	60.93%	58.17%	0.37%	58.80%	57.64%	4.531%
Test 2	77.71%	68.87%	60.93%	59.22%	0.31%	59.77%	58.61%	2.799%
Test 3	80.62%	63.21%	61.70%	57.99%	0.34%	58.61%	57.25%	6.013%
Test 4	82.95%	72.64%	61.51%	58.15%	0.33%	58.61%	57.64%	5.460%
Test 5	81.20%	66.98%	60.35%	57.92%	0.82%	59.19%	56.67%	4.021%
Mean	80.74%	68.49%	61.08%	58.29%				4.56%
Std	1.90%	3.63%	0.54%	0.53%				0.79%
Neurons: 15	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	89.34%	67.92%	61.12%	58.48%	0.33%	59.19%	58.22%	4.315%
Test 2	90.31%	67.92%	60.73%	57.69%	1.06%	59.57%	56.48%	5.009%
Test 3	87.60%	67.92%	59.77%	55.27%	0.63%	56.87%	54.55%	7.532%
Test 4	88.57%	66.04%	61.70%	59.73%	0.39%	60.15%	58.99%	3.192%
Test 5	88.18%	66.98%	60.35%	59.36%	0.78%	60.35%	58.61%	1.632%
Mean	88.80%	67.36%	60.74%	58.11%				4.34%
Std	1.06%	0.84%	0.74%	1.78%				1.31%
Neurons: 20	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	92.25%	62.26%	60.93%	59.68%	0.70%	60.93%	58.80%	2.049%
Test 2	93.22%	68.87%	60.35%	59.54%	0.41%	60.15%	58.99%	1.340%
Test 3	92.64%	72.64%	62.09%	57.53%	0.39%	58.41%	56.87%	7.335%
Test 4	91.86%	69.81%	59.19%	55.78%	0.48%	56.87%	55.13%	5.764%
Test 5	91.09%	66.98%	60.93%	57.52%	0.75%	59.19%	56.67%	5.599%
Mean	92.21%	68.11%	60.70%	58.01%				4.42%
Std	0.80%	3.86%	1.05%	1.63%				1.59%

Table E.1: This table shows the performance achieved when the Volvo B stock was examined. In addition to this, the effectiveness of using the error level on the validation set as a stopping criterion is also examined.

Volvo B with Cleaning Noise								
	Highest Hitrate			Stopping on Lowest Error				
Neurons: 3	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	91.67%	70.75%	61.12%	56.00%	1.60%	58.03%	52.03%	8.372%
Test 2	79.07%	66.98%	60.54%	57.06%	0.82%	58.80%	56.09%	5.751%
Test 3	72.67%	68.87%	59.57%	50.48%	0.86%	51.84%	49.13%	15.260%
Test 4	87.60%	67.92%	60.93%	56.62%	1.42%	58.99%	54.93%	7.071%
Test 5	89.73%	70.75%	59.96%	55.57%	1.82%	58.61%	53.38%	7.331%
Mean	84.15%	69.06%	60.43%	55.15%				8.76%
Std	8.02%	1.69%	0.65%	2.67%				2.21%
Neurons: 6	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	79.26%	71.70%	58.99%	54.09%	0.77%	55.51%	52.80%	8.316%
Test 2	76.94%	66.98%	61.51%	56.92%	1.08%	58.22%	54.55%	7.461%
Test 3	90.31%	66.04%	61.32%	59.72%	0.41%	60.35%	59.19%	2.610%
Test 4	77.13%	68.87%	60.54%	57.96%	0.72%	58.99%	56.67%	4.270%
Test 5	85.27%	66.98%	60.54%	59.42%	0.44%	60.54%	58.80%	1.859%
Mean	81.78%	68.11%	60.58%	57.62%				4.90%
Std	5.84%	2.25%	0.99%	2.27%				1.72%
Neurons: 10	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	85.85%	67.92%	62.09%	60.08%	0.39%	60.54%	59.57%	3.229%
Test 2	88.76%	69.81%	60.93%	60.10%	0.49%	60.73%	59.19%	1.356%
Test 3	90.70%	68.87%	61.32%	60.10%	0.88%	61.32%	58.99%	1.979%
Test 4	89.73%	71.70%	62.28%	60.01%	1.18%	61.70%	58.41%	3.642%
Test 5	90.70%	71.70%	61.32%	60.12%	0.27%	60.35%	59.57%	1.950%
Mean	89.15%	70.00%	61.59%	60.08%				2.43%
Std	2.01%	1.69%	0.57%	0.04%				0.61%
Neurons: 15	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	95.16%	66.04%	59.77%	57.90%	0.73%	58.80%	56.48%	3.119%
Test 2	92.44%	66.98%	62.48%	60.21%	0.76%	61.51%	58.99%	3.631%
Test 3	96.32%	67.92%	61.90%	60.05%	0.45%	60.54%	58.99%	2.983%
Test 4	95.54%	66.98%	61.51%	60.38%	0.73%	61.51%	59.57%	1.830%
Test 5	92.83%	64.15%	60.35%	57.36%	0.95%	58.80%	56.09%	4.953%
Mean	94.46%	66.42%	61.20%	59.18%				3.30%
Std	1.72%	1.43%	1.12%	1.43%				0.68%
Neurons: 20	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	96.51%	72.64%	59.96%	58.31%	0.66%	59.38%	57.06%	2.757%
Test 2	96.71%	68.87%	60.73%	59.36%	0.95%	60.73%	58.03%	2.258%
Test 3	96.71%	69.81%	62.28%	57.45%	0.43%	58.22%	56.87%	7.764%
Test 4	97.67%	65.09%	61.51%	59.75%	0.46%	60.54%	58.99%	2.859%
Test 5	96.12%	66.04%	59.96%	58.96%	0.68%	59.96%	58.03%	1.672%
Mean	96.74%	68.49%	60.89%	58.77%				3.46%
Std	0.57%	3.03%	1.01%	0.91%				1.54%

Table E.2: This table shows the performance achieved, while using the cleaning with noise technique, when the Volvo B stock was examined. In addition to this, the effectiveness of using the error level on the validation set as a stopping criterion is examined.

Scania B								
	Highest Hitrate			Stopping on Lowest Error				
Neurons: 3	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	70.93%	66.04%	58.22%	55.92%	0.71%	57.45%	55.13%	3.957%
Test 2	69.77%	68.87%	59.19%	56.27%	0.50%	56.87%	55.51%	4.932%
Test 3	70.93%	67.92%	59.38%	54.39%	0.48%	55.32%	53.58%	8.410%
Test 4	70.16%	66.04%	57.83%	55.06%	1.23%	57.06%	53.38%	4.804%
Test 5	70.16%	66.04%	60.15%	56.34%	1.56%	59.57%	54.55%	6.343%
Mean	70.39%	66.98%	58.96%	55.59%				5.69%
Std	0.52%	1.33%	0.93%	0.85%				1.06%
Neurons: 6	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	74.22%	66.04%	58.61%	54.05%	0.53%	55.13%	53.19%	7.771%
Test 2	75.78%	66.04%	59.57%	54.53%	0.71%	56.29%	53.77%	8.471%
Test 3	73.84%	66.98%	61.32%	54.81%	0.48%	55.32%	54.16%	10.611%
Test 4	75.58%	67.92%	61.32%	56.57%	1.65%	59.57%	54.35%	7.743%
Test 5	72.67%	66.04%	58.03%	55.92%	0.37%	56.67%	55.51%	3.636%
Mean	74.42%	66.60%	59.77%	55.17%				7.65%
Std	1.29%	0.84%	1.52%	1.04%				1.59%
Neurons: 10	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	79.65%	68.87%	58.80%	56.08%	0.58%	57.45%	55.51%	4.635%
Test 2	79.65%	65.09%	59.96%	57.85%	1.07%	59.77%	56.29%	3.519%
Test 3	77.71%	66.98%	58.61%	53.51%	0.80%	55.13%	52.61%	8.701%
Test 4	81.59%	66.04%	60.54%	56.39%	1.09%	58.22%	55.13%	6.854%
Test 5	79.07%	71.70%	61.12%	56.46%	1.16%	58.03%	54.16%	7.624%
Mean	79.53%	67.74%	59.81%	56.06%				6.27%
Std	1.39%	2.62%	1.09%	1.58%				1.28%
Neurons: 15	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	88.95%	71.70%	59.38%	54.49%	0.58%	55.51%	53.77%	8.232%
Test 2	86.05%	68.87%	59.77%	55.11%	1.26%	57.25%	53.77%	7.796%
Test 3	88.57%	66.98%	62.09%	53.86%	0.65%	55.51%	53.00%	13.254%
Test 4	86.63%	67.92%	59.19%	52.59%	0.79%	53.77%	51.26%	11.141%
Test 5	90.70%	61.32%	59.77%	54.21%	0.43%	54.93%	53.58%	9.297%
Mean	88.18%	67.36%	60.04%	54.05%				9.94%
Std	1.87%	3.81%	1.17%	0.94%				1.46%
Neurons: 20	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	89.92%	62.26%	59.57%	54.21%	0.64%	55.13%	53.38%	9.002%
Test 2	94.38%	68.87%	60.35%	55.88%	0.66%	56.87%	54.55%	7.401%
Test 3	91.47%	69.81%	56.67%	55.04%	0.59%	55.90%	53.77%	2.885%
Test 4	91.67%	65.09%	58.80%	53.74%	0.89%	55.71%	52.61%	8.612%
Test 5	92.83%	66.98%	60.35%	54.05%	0.85%	55.90%	52.61%	10.431%
Mean	92.05%	66.60%	59.15%	54.58%				7.67%
Std	1.66%	3.03%	1.53%	0.87%				1.77%

Table E.3: This table shows the performance achieved when the Scania B stock was examined. In addition to this, the effectiveness of using the error level on the validation set as a stopping criterion is also examined.

Scania B with Cleaning Noise								
	Highest Hitrate			Stopping on Lowest Error				
Neurons: 3	**Training**	**Validation**	**Generalization**	**Mean**	**Std**	**Max**	**Min**	**Gap**
Test 1	83.14%	69.81%	59.19%	55.57%	1.44%	57.64%	53.38%	6.120%
Test 2	85.27%	69.81%	59.38%	54.05%	0.63%	54.74%	52.61%	8.972%
Test 3	84.30%	68.87%	61.12%	55.74%	1.44%	58.03%	52.61%	8.803%
Test 4	77.52%	67.92%	59.19%	54.99%	0.84%	56.29%	53.58%	7.100%
Test 5	84.69%	72.64%	59.57%	56.57%	0.80%	57.64%	55.13%	5.047%
Mean	82.98%	69.81%	59.69%	55.38%				7.21%
Std	3.15%	1.76%	0.82%	0.93%				1.04%
Neurons: 6	**Training**	**Validation**	**Generalization**	**Mean**	**Std**	**Max**	**Min**	**Gap**
Test 1	85.66%	65.09%	59.19%	55.28%	1.62%	57.06%	53.00%	6.595%
Test 2	80.62%	69.81%	58.61%	55.51%	0.61%	56.87%	54.93%	5.281%
Test 3	78.29%	64.15%	60.54%	51.64%	0.92%	53.97%	50.48%	14.697%
Test 4	88.95%	66.98%	61.12%	55.58%	0.66%	57.25%	54.93%	9.062%
Test 5	75.97%	66.98%	58.99%	55.51%	0.90%	57.25%	54.35%	5.902%
Mean	81.90%	66.60%	59.69%	54.71%				8.31%
Std	5.33%	2.17%	1.08%	1.72%				2.38%
Neurons: 10	**Training**	**Validation**	**Generalization**	**Mean**	**Std**	**Max**	**Min**	**Gap**
Test 1	83.91%	66.98%	58.99%	55.23%	1.27%	57.64%	53.58%	6.379%
Test 2	87.79%	65.09%	59.57%	54.51%	0.62%	55.32%	53.38%	8.501%
Test 3	87.02%	72.64%	60.35%	51.36%	0.85%	52.61%	49.71%	14.889%
Test 4	86.43%	64.15%	58.99%	53.44%	1.06%	55.13%	52.03%	9.419%
Test 5	85.27%	65.09%	61.32%	59.15%	0.76%	60.35%	57.83%	3.527%
Mean	86.09%	66.79%	59.85%	54.74%				8.54%
Std	1.52%	3.43%	0.99%	2.87%				2.54%
Neurons: 15	**Training**	**Validation**	**Generalization**	**Mean**	**Std**	**Max**	**Min**	**Gap**
Test 1	92.83%	65.09%	60.15%	58.57%	0.72%	59.77%	57.25%	2.631%
Test 2	91.86%	62.26%	58.22%	55.39%	1.01%	57.25%	53.58%	4.863%
Test 3	93.99%	63.21%	59.19%	56.22%	0.41%	56.87%	55.32%	5.021%
Test 4	92.44%	66.04%	58.41%	54.72%	0.24%	54.93%	54.16%	6.321%
Test 5	91.47%	62.26%	60.15%	56.04%	0.65%	57.25%	55.13%	6.840%
Mean	92.52%	63.77%	59.23%	56.19%				5.14%
Std	0.97%	1.71%	0.92%	1.46%				0.97%
Neurons: 20	**Training**	**Validation**	**Generalization**	**Mean**	**Std**	**Max**	**Min**	**Gap**
Test 1	96.51%	69.81%	58.61%	54.42%	0.58%	55.32%	53.58%	7.141%
Test 2	94.38%	69.81%	58.80%	53.24%	0.15%	53.38%	53.00%	9.450%
Test 3	93.60%	62.26%	57.83%	53.95%	0.98%	55.90%	52.42%	6.719%
Test 4	95.35%	61.32%	57.83%	52.87%	1.31%	55.51%	51.26%	8.574%
Test 5	93.60%	64.15%	60.73%	55.32%	0.72%	56.29%	53.97%	8.917%
Mean	94.69%	65.47%	58.76%	53.96%				8.16%
Std	1.25%	4.09%	1.19%	0.97%				0.74%

Table E.4: This table shows the performance achieved, while using the cleaning with noise technique, when the Scania B stock was examined. In addition to this, the effectiveness of using the error level on the validation set as a stopping criterion is examined.

Volvo B*								
	Highest Hitrate			Stopping on Lowest Error				
Neurons: 3	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	66.47%	61.76%	61.82%	57.85%	0.57%	58.44%	56.88%	6.417%
Test 2	67.83%	62.60%	61.30%	56.67%	1.135	57.92%	53.77%	7.550%
Test 3	65.70%	62.60%	60.78%	55.70%	0.90%	56.62%	54.29%	8.353%
Test 4	68.41%	64.29%	60.78%	57.71%	1.15%	60.00%	56.36%	5.050%
Test 5	71.71%	63.03%	61.30%	56.81%	0.69%	58.44%	55.84%	7.319%
Mean	68.02%	62.86%	61.19%	56.95%				6.94%
Std	2.32%	0.92%	0.43%	0.87%				0.7%
Neurons: 6	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	71.32%	65.55%	61.56%	57.95%	1.00%	59.74%	56.88%	5.869%
Test 2	75.78%	65.13%	61.56%	59.41%	0.89%	60.52%	57.92%	3.491%
Test 3	71.51%	64.71%	61.30%	59.53%	0.78%	60.26%	58.18%	2.889%
Test 4	72.48%	64.29%	62.34%	55.56%	1.67%	57.92%	53.51%	10.871%
Test 5	71.51%	60.50%	62.86%	59.95%	0.70%	61.04%	58.96%	4.621%
Mean	72.52%	64.03%	61.92%	58.48%				5.55%
Std	1.88%	2.03%	0.65%	1.80%				1.99%
Neurons: 10	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	79.26%	61.34%	59.48%	57.24%	0.82%	58.44%	56.10%	3.771%
Test 2	77.13%	64.71%	61.82%	58.58%	0.54%	59.74%	57.92%	5.233%
Test 3	80.62%	64.29%	63.64%	61.28%	0.49%	62.08%	60.52%	3.711%
Test 4	78.49%	65.97%	61.56%	59.83%	0.67%	61.30%	58.70%	2.800%
Test 5	82.95%	64.29%	62.08%	60.31%	0.53%	61.56%	59.74%	2.853%
Mean	79.69%	64.12%	61.71%	59.45%				3.67%
Std	2.22%	1.70%	1.49%	1.57%				0.61%
Neurons: 15	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	88.57%	64.29%	62.86%	60.57%	0.67%	61.56%	59.74%	3.644%
Test 2	87.60%	61.76%	62.86%	60.05%	1.02%	61.30%	58.44%	4.470%
Test 3	89.53%	62.60%	62.08%	55.89%	0.88%	57.14%	54.55%	9.966%
Test 4	88.76%	61.34%	61.30%	59.15%	0.88%	60.78%	58.18%	3.505%
Test 5	89.34%	61.34%	60.78%	58.23%	0.45%	58.70%	57.40%	4.196%
Mean	88.76%	62.27%	61.97%	58.78%				5.16%
Std	0.76%	1.24%	0.93%	1.84%				1.69%
Neurons: 20	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	93.60%	60.08%	62.34%	60.97%	0.65%	62.08%	59.74%	2.197%
Test 2	91.67%	60.92%	63.64%	60.64%	0.97%	63.38%	59.74%	4.712%
Test 3	92.25%	61.76%	61.30%	58.84%	0.68%	60.52%	57.92%	4.006%
Test 4	93.60%	60.92%	60.52%	58.13%	0.97%	59.74%	57.14%	3.941%
Test 5	92.64%	64.29%	62.86%	59.93%	1.23%	61.82%	57.14%	4.658%
Mean	92.75%	61.60%	62.13%	59.70%				3.90%
Std	0.85%	1.62%	1.24%	1.20%				0.65%

Table E.5: This table shows the performance achieved when the Volvo B stock, with a large validation set, was examined. In addition to this, the effectiveness of using the error level on the validation set as a stopping criterion is also examined.

Volvo B* with Cleaning Noise								
	Highest Hitrate			Stopping on Lowest Error				
Neurons: 3	**Training**	**Validation**	**Generalization**	**Mean**	**Std**	**Max**	**Min**	**Gap**
Test 1	86.05%	65.13%	64.16%	58.80%	3.37%	62.60%	50.91%	8.355%
Test 2	80.62%	65.13%	62.34%	58.84%	2.14%	61.82%	56.10%	5.606%
Test 3	75.78%	63.87%	62.34%	59.36%	0.71%	60.78%	58.44%	4.773%
Test 4	90.12%	64.71%	61.30%	58.37%	0.79%	60.00%	56.88%	4.777%
Test 5	77.13%	64.29%	62.08%	58.58%	0.65%	59.74%	57.40%	5.629%
Mean	81.94%	64.62%	62.44%	58.79%				5.83%
Std	6.05%	0.55%	1.05%	0.37%				0.99%
Neurons: 6	**Training**	**Validation**	**Generalization**	**Mean**	**Std**	**Max**	**Min**	**Gap**
Test 1	77.52%	68.49%	62.08%	60.71%	0.76%	62.08%	59.48%	2.206%
Test 2	79.26%	60.50%	63.38%	61.63%	0.49%	62.34%	60.78%	2.757%
Test 3	88.37%	62.18%	62.60%	57.66%	0.55%	58.70%	57.14%	7.884%
Test 4	85.47%	61.76%	61.56%	56.77%	0.48%	57.92%	56.10%	7.787%
Test 5	79.84%	64.29%	63.12%	60.07%	0.53%	60.78%	59.22%	4.826%
Mean	82.09%	63.45%	62.55%	59.37%				5.09%
Std	4.60%	3.13%	0.74%	2.07%				1.66%
Neurons: 10	**Training**	**Validation**	**Generalization**	**Mean**	**Std**	**Max**	**Min**	**Gap**
Test 1	91.67%	61.76%	59.74%	56.91%	0.95%	58.70%	56.10%	4.743%
Test 2	86.43%	62.18%	62.86%	60.52%	0.74%	61.82%	59.48%	3.719%
Test 3	88.76%	62.60%	63.38%	61.23%	0.59%	62.08%	60.00%	3.390%
Test 4	89.53%	60.50%	63.38%	61.25%	0.58%	62.08%	60.52%	3.353%
Test 5	89.92%	63.03%	61.56%	58.42%	0.41%	58.96%	57.66%	5.102%
Mean	89.26%	62.02%	62.18%	59.66%				4.06%
Std	1.91%	0.97%	1.55%	1.93%				0.45%
Neurons: 15	**Training**	**Validation**	**Generalization**	**Mean**	**Std**	**Max**	**Min**	**Gap**
Test 1	95.74%	64.71%	63.12%	61.06%	0.68%	62.34%	60.52%	3.255%
Test 2	95.74%	62.18%	61.56%	58.51%	0.31%	59.22%	58.18%	4.948%
Test 3	96.32%	62.60%	62.86%	59.48%	0.58%	60.52%	58.96%	5.372%
Test 4	94.19%	61.76%	60.78%	59.08%	0.87%	60.26%	57.66%	2.797%
Test 5	95.16%	63.03%	61.04%	58.65%	0.40%	59.48%	57.92%	3.907%
Mean	95.43%	62.86%	61.87%	59.36%				4.06%
Std	0.81%	1.14%	1.06%	1.03%				0.69%
Neurons: 20	**Training**	**Validation**	**Generalization**	**Mean**	**Std**	**Max**	**Min**	**Gap**
Test 1	96.12%	61.34%	60.26%	58.02%	1.12%	59.74%	56.88%	3.722%
Test 2	95.93%	64.29%	62.86%	60.73%	0.66%	61.82%	59.74%	3.381%
Test 3	96.32%	63.03%	62.08%	56.06%	0.60%	57.40%	55.32%	9.699%
Test 4	95.16%	60.92%	62.60%	59.62%	1.00%	61.04%	58.18%	4.753%
Test 5	96.90%	61.34%	62.34%	61.06%	0.36%	61.56%	60.52%	2.046%
Mean	96.09%	62.18%	62.03%	59.10%				4.72%
Std	0.63%	1.42%	1.03%	2.08%				1.83%

Table E.6: This table shows the performance achieved, while using the cleaning with noise technique, when the Volvo B stock with a large validation set was examined. In addition to this, the effectiveness of using the error level on the validation set as a stopping criterion is examined.

Appendix F

Results: Short Forecast Periods

This appendix contains a more detailed description of the results obtained during the 'Short Forecasting Periods' test, which can be seen in Section 6.7. Each table, one for each stock, contains information concerning the maximum hitrates, stopping hitrates and gaps that were achieved during the test.

Volvo B								
	Highest Hitrate			Stopping on Lowest Error				
Period: 1	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	76.97%	67.92%	66.92%	56.71%	2.58%	61.54%	53.08%	15.256%
Test 2	82.53%	74.53%	66.92%	55.80%	1.30%	56.92%	53.85%	16.614%
Test 3	80.61%	72.64%	60.00%	51.40%	1.49%	53.08%	47.69%	14.336%
Test 4	79.27%	75.47%	60.77%	51.96%	1.16%	53.85%	49.23%	14.499%
Test 5	76.20%	71.70%	63.08%	55.17%	1.98%	56.92%	51.54%	12.528%
Mean	79.12%	72.45%	63.54%	54.21%				14.65%
Std	0.03%	2.94%	3.29%	2.38%				1.31%
Period: 2	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	79.65%	65.14%	63.28%	55.18%	0.63%	56.25%	53.91%	12.795%
Test 2	81.00%	63.30%	65.63%	56.75%	1.76%	59.38%	53.91%	13.528%
Test 3	80.23%	67.89%	60.94%	58.45%	1.39%	60.16%	55.47%	4.079%
Test 4	79.65%	63.30%	64.84%	55.75%	1.22%	57.03%	53.91%	14.020%
Test 5	77.35%	59.63%	65.63%	58.24%	5.36%	65.63%	46.88%	11.255%
Mean	79.58%	63.85%	64.06%	56.88%				11.14%
Std	0.01%	3.01%	1.99%	1.46%				2.71%
Period: 3	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	79.23%	64.81%	61.54%	53.15%	0.54%	53.85%	52.31%	13.636%
Test 2	79.23%	65.74%	66.92%	55.80%	2.56%	60.00%	52.31%	16.614%
Test 3	79.42%	63.89%	61.54%	59.93%	0.54%	60.77%	59.23%	2.614%
Test 4	80.96%	67.59%	66.92%	57.34%	0.72%	58.46%	56.15%	14.316%
Test 5	77.12%	69.44%	63.85%	61.05%	1.39%	62.31%	59.23%	4.381%
Mean	79.19%	66.30%	64.15%	57.45%				10.31%
Std	0.01%	2.23%	2.70%	3.18%				4.24%
Period: 4	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	79.85%	65.45%	69.53%	59.16%	1.40%	60.94%	56.25%	14.913%
Test 2	80.04%	67.27%	71.88%	57.17%	2.09%	60.16%	53.13%	20.455%
Test 3	78.31%	68.18%	66.41%	58.59%	0.99%	60.16%	57.03%	11.765%
Test 4	79.46%	67.27%	70.31%	55.89%	1.86%	58.59%	52.34%	20.505%
Test 5	80.61%	69.09%	71.09%	58.10%	0.94%	59.38%	57.03%	18.282%
Mean	79.65%	67.45%	69.84%	57.78%				17.18%
Std	0.01%	1.35%	2.11%	1.28%				2.93%

Table F.1: This table shows the performance when using short forecast periods for the Volvo B stock. In addition to this, the effectiveness of using the error level on the validation set as a stopping criterion can be seen.

Scania B								
	Highest Hitrate			Stopping on Lowest Error				
Period: 1	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	79.96%	66.98%	64.84%	58.17%	1.01%	59.38%	57.03%	10.296%
Test 2	82.85%	64.15%	64.06%	54.33%	1.86%	57.81%	52.34%	15.188%
Test 3	82.85%	67.92%	65.63%	54.83%	0.77%	56.25%	53.91%	16.450%
Test 4	82.85%	68.87%	65.63%	55.89%	0.41%	56.25%	55.47%	14.827%
Test 5	79.38%	66.04%	63.28%	59.94%	1.11%	61.72%	57.81%	5.275%
Mean	81.58%	66.79%	64.69%	56.63%				12.41%
Std	0.02%	1.81%	1.02%	2.37%				3.05%
Period: 2	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	83.24%	69.72%	65.08%	53.10%	1.09%	54.76%	51.59%	18.404%
Test 2	82.85%	64.22%	61.11%	54.62%	1.62%	56.35%	50.79%	10.626%
Test 3	84.01%	66.05%	61.11%	51.15%	0.90%	52.38%	50.00%	16.293%
Test 4	82.66%	63.30%	62.70%	53.97%	1.37%	55.56%	52.38%	13.924%
Test 5	81.70%	64.22%	66.67%	51.37%	1.42%	53.97%	50.00%	22.944%
Mean	82.89%	65.50%	63.33%	52.84%				16.44%
Std	0.01%	2.56%	2.47%	1.54%				3.34%
Period: 3	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	82.82%	61.11%	69.53%	56.96%	1.90%	60.16%	54.69%	18.080%
Test 2	81.08%	63.89%	66.41%	54.69%	1.52%	57.03%	53.13%	17.647%
Test 3	83.20%	62.04%	67.19%	59.80%	1.12%	62.50%	58.59%	10.994%
Test 4	83.78%	63.89%	69.53%	58.81%	1.05%	60.16%	57.03%	15.424%
Test 5	83.98%	59.26%	69.53%	52.27%	2.22%	57.03%	47.66%	24.821%
Mean	82.97%	62.04%	68.44%	56.51%				17.39%
Std	0.01%	1.96%	1.52%	3.07%				3.57%
Period: 4	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	83.62%	66.36%	59.20%	53.82%	0.72%	54.40%	52.00%	9.091%
Test 2	82.85%	67.27%	66.40%	58.55%	0.79%	60.00%	57.60%	11.829%
Test 3	82.66%	68.18%	65.60%	57.67%	1.81%	60.00%	55.20%	12.084%
Test 4	83.43%	65.45%	65.60%	59.78%	0.72%	60.80%	58.40%	8.869%
Test 5	79.58%	66.36%	68.00%	58.40%	1.19%	60.00%	56.00%	14.118%
Mean	82.43%	66.73%	64.96%	57.64%				11.20%
Std	0.02%	1.04%	3.37%	2.27%				1.72%

Table F.2: This table shows the performance when using short forecast periods for the Scania B stock. In addition to this, the effectiveness of using the error level on the validation set as a stopping criterion can be seen.

Appendix G

Results: State Cluster Neurons

This appendix contains a more detailed description of the results obtained during the 'State Cluster Neurons' test, which can be seen in Section 6.8. Each table, one for each stock with and without the cleaning with noise technique, contains information concerning the maximum hitrates, stopping hitrates and gaps that were achieved during the test.

Volvo B								
	Highest Hitrate			Stopping on Lowest Error				
Neurons: 6	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	77.16%	67.92%	60.27%	57.18%	0.38%	57.97%	56.62%	5.124%
Test 2	80.04%	72.64%	64.11%	56.78%	0.56%	57.77%	56.05%	11.432%
Test 3	77.35%	69.81%	60.27%	55.82%	0.75%	56.81%	54.51%	7.383%
Test 4	77.93%	72.64%	59.88%	57.28%	0.41%	57.77%	56.62%	4.341%
Test 5	79.08%	69.81%	63.92%	58.66%	0.73%	60.27%	57.77%	8.217%
Mean	78.31%	70.57%	61.69%	57.15%				7.30%
Std	0.01%	2.05%	2.13%	1.03%				1.88%
Neurons: 10	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	85.80%	70.75%	60.08%	56.46%	0.55%	57.39%	55.28%	6.012%
Test 2	86.18%	71.70%	60.46%	56.69%	0.46%	57.39%	55.85%	6.234%
Test 3	84.26%	75.47%	61.80%	58.58%	0.75%	59.69%	57.39%	5.223%
Test 4	85.41%	72.64%	60.27%	55.75%	0.39%	56.43%	55.09%	7.499%
Test 5	85.22%	70.75%	60.65%	58.02%	1.69%	60.65%	55.47%	4.344%
Mean	85.37%	72.26%	60.65%	57.10%				5.86%
Std	0.01%	1.96%	0.68%	1.16%				0.70%
Neurons: 15	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	90.40%	70.75%	59.12%	54.11%	0.43%	54.89%	53.36%	8.471%
Test 2	92.51%	69.81%	60.84%	55.99%	0.72%	57.58%	54.89%	7.972%
Test 3	92.32%	73.58%	59.12%	55.54%	0.57%	56.43%	54.89%	6.051%
Test 4	93.28%	71.70%	58.73%	55.84%	0.52%	56.62%	54.89%	4.932%
Test 5	91.17%	74.53%	61.04%	57.39%	0.69%	58.93%	56.24%	5.975%
Mean	91.94%	72.08%	59.77%	55.77%				6.68%
Std	0.01%	1.96%	1.08%	1.17%				0.90%
Neurons: 20	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	93.67%	71.70%	61.04%	57.15%	0.45%	57.77%	56.43%	6.375%
Test 2	95.01%	71.70%	59.69%	58.47%	0.85%	59.50%	57.01%	2.046%
Test 3	95.78%	77.36%	60.65%	58.07%	0.96%	59.88%	56.62%	4.258%
Test 4	96.74%	73.58%	59.88%	56.01%	0.62%	57.01%	54.89%	6.468%
Test 5	95.59%	68.87%	57.97%	55.45%	0.75%	56.62%	54.13%	4.335%
Mean	95.36%	72.64%	59.85%	57.03%				4.70%
Std	0.01%	3.13%	1.19%	1.29%				1.11%
Neurons: 25	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	98.85%	71.70%	60.08%	56.76%	0.65%	57.77%	55.66%	5.518%
Test 2	98.85%	70.75%	60.65%	57.60%	0.81%	58.35%	56.05%	5.035%
Test 3	97.31%	72.64%	59.88%	55.54%	0.67%	56.62%	54.51%	7.255%
Test 4	98.27%	70.75%	60.08%	55.77%	0.67%	57.20%	54.70%	7.174%
Test 5	98.46%	72.64%	60.84%	57.51%	0.69%	58.73%	56.62%	5.477%
Mean	98.35%	71.70%	60.31%	56.64%				6.09%
Std	0.01%	0.94%	0.42%	0.96%				0.61%

Table G.1: This table shows the performance achieved for the Volvo B stock when examining the state cluster neurons. In addition to this, the effectiveness of using the error level on the validation set as a stopping criterion can be seen.

Volvo B with Cleaning Noise								
	Highest Hitrate			Stopping on Lowest Error				
Neurons: 6	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	92.51%	71.70%	60.08%	55.16%	0.86%	56.81%	53.55%	8.191%
Test 2	93.09%	67.92%	60.65%	56.97%	0.78%	58.35%	55.66%	6.070%
Test 3	93.28%	72.64%	59.12%	57.15%	0.97%	58.93%	55.85%	3.335%
Test 4	92.90%	70.75%	63.92%	56.03%	0.62%	57.01%	55.09%	12.340%
Test 5	90.59%	73.58%	61.04%	57.08%	0.47%	57.77%	56.43%	6.489%
Mean	92.48%	71.32%	60.96%	56.48%				7.29%
Std	0.01%	2.17%	1.80%	0.87%				2.18%
Neurons: 10	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	96.16%	75.47%	59.88%	57.35%	1.12%	58.54%	54.51%	4.225%
Test 2	96.35%	73.58%	60.46%	56.36%	0.76%	57.39%	55.09%	6.782%
Test 3	95.39%	70.75%	60.46%	57.91%	0.97%	59.50%	56.24%	4.214%
Test 4	96.74%	69.81%	58.54%	55.63%	0.83%	56.81%	54.32%	4.978%
Test 5	96.16%	68.87%	58.93%	56.92%	0.44%	57.58%	56.05%	3.405%
Mean	96.16%	71.70%	59.65%	56.83%				4.72%
Std	0.00%	2.75%	0.88%	0.88%				0.79%
Neurons: 15	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	98.08%	73.58%	59.69%	55.77%	0.58%	56.62%	54.89%	6.577%
Test 2	97.50%	73.58%	60.84%	57.97%	0.64%	58.93%	57.01%	4.732%
Test 3	97.89%	76.42%	61.80%	57.13%	0.45%	58.16%	56.81%	7.566%
Test 4	97.70%	68.87%	60.84%	58.24%	1.46%	60.84%	55.09%	4.273%
Test 5	98.08%	72.64%	59.50%	56.69%	0.38%	57.39%	56.24%	4.721%
Mean	97.85%	73.02%	60.54%	57.16%				5.57%
Std	0.00%	2.72%	0.95%	1.00%				0.89%
Neurons: 20	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	99.23%	68.87%	60.46%	56.90%	0.65%	57.77%	55.85%	5.888%
Test 2	98.66%	69.81%	59.31%	55.75%	0.57%	56.24%	54.32%	6.002%
Test 3	99.04%	70.75%	60.46%	59.01%	0.71%	59.88%	57.58%	2.395%
Test 4	98.46%	69.81%	59.69%	56.71%	0.42%	57.39%	55.85%	4.999%
Test 5	98.85%	73.58%	60.08%	57.15%	0.44%	57.77%	56.62%	4.879%
Mean	98.85%	70.57%	60.00%	57.10%				4.83%
Std	0.00%	1.81%	0.50%	1.19%				0.86%
Neurons: 25	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	99.81%	71.70%	60.27%	57.09%	0.77%	58.54%	56.24%	5.269%
Test 2	99.42%	72.64%	58.73%	56.41%	0.53%	57.01%	55.47%	3.951%
Test 3	99.62%	72.64%	59.88%	55.82%	1.04%	57.97%	54.51%	6.789%
Test 4	99.62%	72.64%	60.84%	58.61%	1.20%	60.65%	56.05%	3.671%
Test 5	99.62%	73.58%	58.16%	55.03%	0.61%	56.05%	54.32%	5.371%
Mean	99.62%	72.64%	59.58%	56.59%				5.01%
Std	0.00%	0.67%	1.11%	1.36%				0.75%

Table G.2: This table shows the performance achieved, while using the cleaning with noise technique, for the Volvo B stock when examining the state cluster neurons. In addition to this, the effectiveness of using the error level on the validation set as a stopping criterion can be seen.

Scania B								
	Highest Hitrate			Stopping on Lowest Error				
Neurons: 6	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	76.69%	64.15%	57.61%	53.90%	0.81%	55.11%	52.79%	6.446%
Test 2	75.92%	65.09%	60.12%	54.00%	0.50%	55.11%	53.37%	10.169%
Test 3	72.64%	66.98%	59.15%	53.30%	0.70%	54.72%	52.22%	9.890%
Test 4	73.60%	64.15%	60.31%	57.10%	0.80%	58.38%	56.26%	5.315%
Test 5	74.37%	66.98%	61.46%	54.30%	0.55%	55.68%	53.76%	11.656%
Mean	74.64%	65.47%	59.73%	54.52%				8.70%
Std	0.02%	1.43%	1.44%	1.49%				1.68%
Neurons: 10	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	82.27%	66.04%	59.54%	53.39%	0.52%	54.72%	52.79%	10.327%
Test 2	82.85%	66.04%	59.15%	55.32%	0.32%	55.88%	54.72%	6.485%
Test 3	84.20%	66.98%	60.12%	55.32%	0.63%	56.84%	54.72%	7.984%
Test 4	80.54%	65.09%	58.00%	54.67%	0.39%	55.49%	54.14%	5.738%
Test 5	84.20%	64.15%	61.08%	57.82%	0.39%	58.38%	57.03%	5.334%
Mean	82.81%	65.66%	59.58%	55.30%				7.17%
Std	0.02%	1.08%	1.14%	1.61%				1.22%
Neurons: 15	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	87.67%	70.75%	58.77%	54.67%	0.66%	55.68%	53.95%	6.975%
Test 2	89.40%	65.09%	61.27%	54.51%	0.24%	55.11%	54.34%	11.035%
Test 3	91.91%	67.92%	59.15%	55.23%	0.55%	56.07%	54.14%	6.633%
Test 4	89.40%	66.98%	61.08%	54.41%	0.64%	55.49%	53.37%	10.926%
Test 5	90.37%	65.09%	58.38%	57.10%	0.30%	57.42%	56.65%	2.190%
Mean	89.75%	67.17%	59.73%	55.18%				7.55%
Std	0.02%	2.35%	1.35%	1.12%				2.27%
Neurons: 20	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	91.91%	68.87%	59.54%	55.14%	0.45%	55.68%	54.34%	7.385%
Test 2	92.29%	69.81%	59.15%	55.11%	0.51%	55.68%	54.14%	6.840%
Test 3	93.45%	64.15%	58.19%	52.93%	0.64%	54.14%	51.83%	9.031%
Test 4	90.94%	69.81%	59.92%	56.16%	0.74%	57.23%	54.53%	6.285%
Test 5	91.91%	67.92%	60.50%	55.63%	1.02%	57.23%	54.34%	8.049%
Mean	92.10%	68.11%	59.46%	54.99%				7.52%
Std	0.01%	2.35%	0.87%	1.23%				0.60%
Neurons: 25	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	94.80%	66.04%	59.73%	56.73%	0.90%	58.00%	55.30%	5.015%
Test 2	94.99%	66.98%	58.77%	55.32%	0.86%	56.45%	53.56%	5.872%
Test 3	94.60%	63.21%	58.19%	55.02%	0.53%	55.49%	53.95%	5.448%
Test 4	95.38%	70.75%	61.08%	56.59%	1.22%	58.19%	53.76%	7.342%
Test 5	94.60%	69.81%	57.80%	53.06%	0.43%	53.95%	52.60%	8.212%
Mean	94.87%	67.36%	59.11%	55.34%				6.38%
Std	0.00%	3.03%	1.32%	1.49%				0.79%

Table G.3: This table shows the performance achieved for the Scania B stock when examining the state cluster neurons. In addition to this, the effectiveness of using the error level on the validation set as a stopping criterion can be seen.

Scania B with Cleaning Noise								
	Highest Hitrate			Stopping on Lowest Error				
Neurons: 6	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	87.67%	67.92%	58.38%	53.20%	0.70%	54.53%	52.22%	8.881%
Test 2	88.05%	67.92%	61.08%	53.60%	0.44%	54.34%	52.99%	12.245%
Test 3	87.48%	62.26%	58.96%	55.75%	0.73%	56.84%	54.91%	5.437%
Test 4	89.60%	69.81%	61.85%	56.72%	0.72%	58.38%	55.88%	8.298%
Test 5	84.20%	68.87%	58.96%	55.14%	1.28%	57.80%	53.76%	6.476%
Mean	87.40%	67.36%	59.85%	54.88%				8.27%
Std	0.02%	2.95%	1.52%	1.47%				1.64%
Neurons: 10	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	86.51%	65.09%	61.27%	55.30%	0.30%	55.88%	54.91%	9.748%
Test 2	89.98%	67.92%	60.31%	57.21%	1.07%	58.77%	55.30%	5.141%
Test 3	93.64%	75.47%	59.34%	57.09%	0.47%	58.19%	56.65%	3.808%
Test 4	90.94%	66.98%	59.34%	58.15%	0.54%	58.96%	57.23%	2.007%
Test 5	93.45%	67.92%	61.27%	54.55%	0.34%	55.11%	54.14%	10.978%
Mean	90.91%	68.68%	60.31%	56.46%				6.34%
Std	0.03%	3.97%	0.96%	1.49%				2.39%
Neurons: 15	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	94.41%	62.26%	58.57%	53.09%	0.65%	54.34%	51.83%	9.360%
Test 2	94.41%	68.87%	59.92%	55.75%	0.63%	56.45%	54.72%	6.957%
Test 3	95.38%	63.21%	60.50%	54.77%	0.31%	55.30%	54.14%	9.467%
Test 4	94.22%	66.04%	60.69%	56.58%	0.45%	57.61%	55.88%	6.782%
Test 5	96.34%	64.15%	58.77%	55.65%	0.76%	57.23%	54.72%	5.306%
Mean	94.95%	64.91%	59.69%	55.17%				7.57%
Std	0.01%	2.62%	0.98%	1.33%				1.08%
Neurons: 20	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	96.92%	64.15%	59.54%	54.69%	0.33%	55.11%	54.14%	8.149%
Test 2	97.11%	66.98%	59.34%	55.70%	0.81%	56.65%	54.34%	6.139%
Test 3	96.53%	66.98%	61.27%	56.52%	0.16%	56.84%	56.26%	7.747%
Test 4	95.18%	66.98%	62.04%	55.98%	0.26%	56.45%	55.68%	9.769%
Test 5	97.11%	67.92%	59.92%	52.88%	1.03%	54.34%	51.83%	11.751%
Mean	96.57%	66.60%	60.42%	55.16%				8.71%
Std	0.01%	1.43%	1.18%	1.44%				1.31%
Neurons: 25	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	99.04%	64.15%	58.77%	55.32%	0.39%	56.07%	54.72%	5.872%
Test 2	98.84%	68.87%	60.12%	55.53%	0.33%	56.26%	55.11%	7.634%
Test 3	96.72%	66.98%	59.73%	55.96%	0.31%	56.45%	55.49%	6.305%
Test 4	97.88%	64.15%	58.57%	52.95%	1.09%	54.91%	51.64%	9.599%
Test 5	97.11%	64.15%	59.73%	55.74%	0.27%	56.07%	55.11%	6.686%
Mean	97.92%	65.66%	59.38%	55.10%				7.22%
Std	0.01%	2.17%	0.67%	1.22%				0.86%

Table G.4: This table shows the performance achieved, while using the cleaning with noise technique, for the Scania B stock when examining the state cluster neurons. In addition to this, the effectiveness of using the error level on the validation set as a stopping criterion can be seen.

Appendix H

Results: Target Multipliers

This appendix contains a more detailed description of the results obtained during the 'Target Multiplier' test, which can be seen in Section 6.9. Each table, one for each stock, contains information concerning the maximum hitrates, stopping hitrates and gaps that were achieved during the test.

Volvo B								
	Highest Hitrate			Stopping on Lowest Error				
TM: 1	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	64.73%	65.09%	56.67%	53.54%	0.73%	55.32%	52.80%	5.523%
Test 2	59.88%	62.26%	59.57%	54.35%	1.17%	56.09%	52.80%	8.766%
Test 3	61.82%	57.55%	56.09%	53.89%	0.23%	54.16%	53.38%	3.918%
Test 4	66.09%	67.92%	57.83%	55.32%	0.37%	55.90%	54.93%	4.348%
Test 5	61.82%	63.21%	59.38%	54.26%	0.85%	55.90%	53.00%	8.617%
Mean	62.87%	63.21%	57.91%	54.27%				6.23%
Std	0.02%	3.83%	1.56%	0.67%				1.44%
TM: 7	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	71.90%	67.92%	61.12%	59.89%	0.93%	61.12%	57.45%	2.014%
Test 2	73.64%	67.92%	60.35%	57.97%	0.62%	59.38%	57.06%	3.934%
Test 3	74.42%	67.92%	61.70%	57.85%	0.32%	58.41%	57.45%	6.241%
Test 4	75.19%	69.81%	60.54%	59.05%	0.29%	59.38%	58.61%	2.469%
Test 5	70.54%	70.75%	60.15%	55.21%	0.72%	56.67%	53.97%	8.214%
Mean	73.14%	68.87%	60.77%	58.00%				4.57%
Std	0.02%	1.33%	0.63%	1.76%				1.58%
TM: 13	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	74.22%	70.75%	58.80%	56.11%	0.61%	57.45%	55.51%	4.575%
Test 2	73.45%	68.87%	61.12%	59.22%	0.89%	61.12%	58.41%	3.107%
Test 3	72.67%	69.81%	60.73%	54.02%	1.05%	55.51%	52.22%	11.060%
Test 4	75.78%	66.98%	61.12%	59.89%	0.71%	60.93%	58.99%	2.014%
Test 5	72.48%	69.81%	62.28%	57.22%	1.07%	58.22%	54.93%	8.131%
Mean	73.72%	69.25%	60.81%	57.29%				5.78%
Std	0.01%	1.43%	1.27%	2.38%				2.30%
TM: 20	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	75.00%	69.81%	62.67%	61.19%	0.65%	61.90%	59.96%	2.357%
Test 2	76.16%	74.53%	63.25%	61.14%	0.45%	61.70%	60.15%	3.336%
Test 3	74.22%	68.87%	60.54%	57.89%	0.69%	58.80%	56.48%	4.386%
Test 4	73.84%	71.70%	61.70%	59.49%	0.49%	60.15%	58.22%	3.591%
Test 5	76.74%	75.47%	61.90%	57.38%	0.95%	58.41%	55.51%	7.301%
Mean	75.19%	72.08%	62.01%	59.42%				4.19%
Std	0.01%	2.88%	1.03%	1.78%				1.15%
TM: 27	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	75.00%	70.75%	60.54%	57.92%	1.18%	59.19%	55.13%	4.328%
Test 2	75.00%	66.04%	61.51%	57.62%	0.88%	58.80%	56.29%	6.318%
Test 3	70.35%	65.09%	61.90%	59.80%	1.15%	61.70%	58.41%	3.381%
Test 4	77.33%	69.81%	59.96%	51.52%	0.80%	53.00%	50.10%	14.076%
Test 5	73.45%	69.81%	61.90%	58.62%	0.77%	59.38%	57.25%	5.284%
Mean	74.22%	68.30%	61.16%	57.10%				6.68%
Std	0.03%	2.55%	0.87%	3.23%				2.54%

Table H.1: This table shows the performance achieved, and the effectiveness of using the error level on the validation set as a stopping criterion, for the Volvo B stock when using different target multipliers.

Scania B								
	Highest Hitrate			Stopping on Lowest Error				
TM: 1	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	66.09%	65.09%	59.77%	57.29%	0.97%	58.99%	56.09%	4.148%
Test 2	64.34%	66.04%	59.77%	57.43%	0.93%	59.19%	56.09%	3.913%
Test 3	64.73%	61.32%	59.77%	53.40%	0.83%	54.74%	52.42%	10.650%
Test 4	62.98%	65.09%	56.09%	53.23%	1.02%	55.13%	51.84%	5.110%
Test 5	64.15%	64.15%	58.03%	54.44%	0.45%	55.13%	53.58%	6.182%
Mean	64.46%	64.34%	58.68%	55.16%				6.00%
Std	0.01%	1.81%	1.63%	2.06%				1.66%
TM: 7	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	78.29%	67.92%	60.54%	54.77%	1.07%	57.83%	53.77%	9.527%
Test 2	71.71%	65.09%	60.54%	54.58%	0.99%	56.87%	53.77%	9.846%
Test 3	75.19%	67.92%	59.38%	54.21%	0.37%	54.74%	53.58%	8.706%
Test 4	79.65%	66.04%	57.64%	56.18%	0.53%	56.87%	55.13%	2.532%
Test 5	76.36%	69.81%	59.77%	54.53%	0.49%	55.51%	53.97%	8.767%
Mean	76.24%	67.36%	59.57%	54.85%				7.88%
Std	0.03%	1.84%	1.19%	0.77%				1.85%
TM: 13	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	79.65%	68.87%	58.80%	56.08%	0.58%	57.45%	55.51%	4.635%
Test 2	79.65%	65.09%	59.96%	57.85%	1.07%	59.77%	56.29%	3.519%
Test 3	77.71%	66.98%	58.61%	53.51%	0.80%	55.13%	52.61%	8.701%
Test 4	81.59%	66.04%	60.54%	56.39%	1.09%	58.22%	55.13%	6.854%
Test 5	79.07%	71.70%	61.12%	56.46%	1.16%	58.03%	54.16%	7.624%
Mean	79.53%	67.74%	59.81%	56.06%				6.27%
Std	0.01%	2.62%	1.09%	1.58%				1.28%
TM: 20	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	73.64%	65.09%	59.38%	56.39%	0.74%	57.45%	55.32%	5.034%
Test 2	77.91%	64.15%	58.99%	53.93%	0.27%	54.35%	53.38%	8.584%
Test 3	79.46%	68.87%	58.41%	54.86%	0.77%	56.09%	53.77%	6.081%
Test 4	78.88%	68.87%	60.73%	52.08%	0.35%	52.61%	51.64%	14.244%
Test 5	79.26%	64.15%	58.99%	55.57%	0.42%	56.48%	54.93%	5.812%
Mean	77.83%	66.23%	59.30%	54.57%				7.95%
Std	0.02%	2.44%	0.87%	1.66%				2.32%
TM: 27	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	80.62%	66.04%	60.35%	52.52%	1.41%	55.71%	50.87%	12.966%
Test 2	83.33%	64.15%	59.38%	53.17%	0.38%	53.58%	52.42%	10.453%
Test 3	84.30%	64.15%	58.41%	53.75%	0.49%	54.93%	53.00%	7.977%
Test 4	80.04%	66.98%	58.41%	54.74%	0.58%	56.29%	54.16%	6.291%
Test 5	80.23%	65.09%	59.57%	51.96%	0.67%	53.58%	51.26%	12.780%
Mean	81.71%	65.28%	59.23%	53.23%				10.09%
Std	0.02%	1.23%	0.83%	1.08%				1.82%

Table H.2: This table shows the performance achieved, and the effectiveness of using the error level on the validation set as a stopping criterion, for the Scania B stock when using different target multipliers.

Appendix I

Results: Replacing Missing Values

This appendix contains a more detailed description of the results obtained during the 'Replacing Missing Values' test, which can be seen in Section 6.10. Each table, one for each stock with and without the cleaning with noise technique, contains information concerning the maximum hitrates, stopping hitrates and gaps that were achieved during the test.

Volvo B								
	Highest Hitrate			Stopping on Lowest Error				
Neurons: 6	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	67.65%	65.69%	56.56%	55.02%	0.92%	55.96%	52.74%	2.715%
Test 2	69.07%	64.71%	57.77%	53.07%	0.66%	54.35%	52.13%	8.134%
Test 3	69.47%	66.67%	59.58%	54.16%	1.04%	55.75%	52.13%	9.084%
Test 4	69.47%	69.61%	56.56%	52.10%	0.67%	53.34%	51.13%	7.887%
Test 5	68.06%	66.67%	57.36%	53.91%	0.57%	54.75%	53.14%	6.024%
Mean	68.74%	66.67%	57.56%	53.65%				6.77%
Std	0.84%	1.83%	1.24%	1.11%				1.50%
Neurons: 10	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	79.78%	68.63%	58.77%	53.36%	1.31%	55.96%	51.33%	9.208%
Test 2	79.58%	70.59%	56.36%	52.32%	0.82%	53.34%	50.73%	7.169%
Test 3	81.00%	66.67%	58.57%	56.45%	0.55%	57.36%	55.15%	3.621%
Test 4	76.75%	64.71%	56.96%	53.34%	0.48%	54.15%	52.74%	6.355%
Test 5	75.74%	66.67%	56.36%	53.82%	1.28%	55.35%	52.34%	4.509%
Mean	78.57%	67.45%	57.40%	53.86%				6.17%
Std	2.22%	2.24%	1.18%	1.55%				1.30%
Neurons: 15	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	85.04%	71.57%	57.36%	55.72%	0.35%	56.36%	55.15%	2.868%
Test 2	87.67%	70.59%	58.77%	55.68%	0.60%	56.56%	54.35%	5.257%
Test 3	86.66%	67.65%	56.96%	51.99%	0.70%	53.34%	50.93%	8.731%
Test 4	87.87%	63.73%	57.77%	53.03%	0.50%	53.94%	52.34%	8.197%
Test 5	84.64%	65.69%	56.56%	54.51%	0.42%	55.15%	53.74%	3.620%
Mean	86.37%	67.84%	57.48%	54.19%				5.73%
Std	1.48%	3.28%	0.85%	1.65%				1.52%

Table I.1: This table shows the performance achieved when replacing missing values with the mean of the five previous values for the Volvo B stock. The effectiveness of using the error level on the validation set as a stopping criterion can also be seen.

Volvo B with Cleaning Noise								
	Highest Hitrate			Stopping on Lowest Error				
Neurons: 6	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	76.35%	67.65%	58.17%	55.74%	0.86%	57.56%	54.95%	4.180%
Test 2	87.67%	67.65%	56.76%	50.85%	0.90%	52.13%	48.72%	10.404%
Test 3	77.96%	66.67%	56.96%	54.13%	0.34%	54.95%	53.74%	4.975%
Test 4	82.01%	64.71%	59.78%	56.67%	0.81%	57.97%	55.75%	5.200%
Test 5	79.38%	64.71%	57.16%	53.78%	0.28%	54.15%	53.14%	5.917%
Mean	80.67%	66.27%	57.77%	54.23%				6.14%
Std	4.43%	1.49%	1.25%	2.23%				1.37%
Neurons: 10	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	89.49%	67.65%	57.97%	48.99%	0.61%	49.92%	48.31%	15.486%
Test 2	86.05%	64.71%	56.16%	53.38%	0.96%	54.75%	52.34%	4.949%
Test 3	90.70%	64.71%	56.96%	53.29%	0.60%	54.35%	52.54%	6.452%
Test 4	91.10%	69.61%	57.36%	55.59%	0.44%	56.36%	54.95%	3.092%
Test 5	88.68%	67.65%	57.36%	55.28%	0.60%	56.36%	54.55%	3.633%
Mean	89.20%	66.86%	57.16%	53.30%				6.72%
Std	2.01%	2.13%	0.67%	2.63%				2.95%
Neurons: 15	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	93.33%	65.69%	56.96%	54.40%	0.97%	56.76%	52.94%	4.494%
Test 2	92.52%	67.65%	57.77%	55.15%	1.09%	56.56%	53.54%	4.526%
Test 3	95.15%	65.69%	56.96%	53.60%	0.52%	54.75%	52.74%	5.906%
Test 4	96.36%	66.67%	57.36%	53.58%	0.75%	54.95%	52.34%	6.598%
Test 5	94.74%	64.71%	55.75%	53.20%	0.49%	54.15%	52.54%	4.591%
Mean	94.42%	66.08%	56.96%	53.99%				5.22%
Std	1.52%	1.12%	0.75%	0.79%				0.57%

Table I.2: This table shows the performance achieved when replacing missing values with the mean of the five previous values for the Volvo B stock, while using the cleaning with noise technique. The effectiveness of using the error level on the validation set as a stopping criterion can also be seen.

Scania B								
	Highest Hitrate			Stopping on Lowest Error				
Neurons: 6	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	71.00%	58.76%	55.68%	52.79%	0.45%	53.42%	52.19%	5.193%
Test 2	67.19%	62.89%	56.50%	50.49%	1.62%	52.60%	48.29%	10.631%
Test 3	68.67%	59.79%	57.52%	51.01%	0.66%	52.60%	50.14%	11.317%
Test 4	68.89%	62.89%	55.88%	48.65%	0.87%	49.93%	47.06%	12.951%
Test 5	67.19%	56.70%	54.24%	50.73%	0.28%	51.16%	50.34%	6.465%
Mean	68.59%	60.21%	55.97%	50.73%				9.31%
Std	1.57%	2.69%	1.20%	1.48%				1.92%
Neurons: 10	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	73.12%	59.79%	54.24%	51.24%	0.83%	52.40%	49.73%	5.537%
Test 2	75.87%	55.67%	56.70%	51.43%	1.11%	52.81%	49.93%	9.310%
Test 3	70.16%	59.79%	54.65%	49.80%	1.01%	51.57%	48.09%	8.874%
Test 4	73.75%	58.76%	56.09%	49.75%	1.15%	52.19%	48.70%	11.308%
Test 5	77.78%	61.86%	54.45%	51.76%	0.52%	52.60%	50.55%	4.934%
Mean	74.13%	59.18%	55.23%	50.79%				7.99%
Std	2.88%	2.26%	1.10%	0.95%				1.55%
Neurons: 15	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	83.91%	58.76%	56.91%	49.93%	0.84%	51.98%	48.91%	12.259%
Test 2	86.24%	55.67%	55.47%	52.64%	0.74%	54.04%	51.78%	5.111%
Test 3	85.61%	60.82%	55.88%	50.34%	1.25%	52.81%	47.88%	9.914%
Test 4	86.24%	59.79%	55.06%	51.16%	1.16%	53.01%	49.52%	7.080%
Test 5	83.91%	58.76%	53.63%	51.63%	0.55%	52.40%	50.55%	3.722%
Mean	85.18%	58.76%	55.39%	51.14%				7.62%
Std	1.19%	1.93%	1.20%	1.07%				2.02%

Table I.3: This table shows the performance achieved when replacing missing values with the mean of the five previous values for the Scania B stock. The effectiveness of using the error level on the validation set as a stopping criterion can also be seen.

Scania B with Cleaning Noise								
	Highest Hitrate			Stopping on Lowest Error				
Neurons: 6	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	71.43%	61.86%	55.88%	51.69%	1.04%	53.22%	49.93%	7.511%
Test 2	75.24%	59.79%	54.45%	50.23%	0.68%	51.16%	49.32%	7.743%
Test 3	83.91%	64.95%	55.27%	50.10%	1.79%	54.24%	48.09%	9.349%
Test 4	79.68%	58.76%	55.68%	49.93%	0.97%	52.40%	48.70%	10.319%
Test 5	76.08%	55.67%	54.86%	48.31%	0.52%	49.73%	47.88%	11.936%
Mean	77.27%	60.21%	55.23%	50.05%				9.37%
Std	4.73%	3.47%	0.59%	1.20%				1.01%
Neurons: 10	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	85.18%	57.73%	54.65%	50.04%	0.83%	51.37%	48.91%	8.431%
Test 2	85.40%	60.82%	60.40%	49.60%	0.86%	51.37%	48.50%	17.883%
Test 3	79.89%	56.70%	56.29%	51.22%	0.66%	52.60%	50.55%	9.013%
Test 4	84.34%	57.73%	54.65%	48.38%	0.60%	49.73%	47.27%	11.468%
Test 5	82.64%	64.95%	58.14%	49.71%	0.72%	50.55%	47.88%	14.502%
Mean	83.49%	59.59%	56.83%	49.79%				12.26%
Std	2.28%	3.37%	2.46%	1.02%				2.57%
Neurons: 15	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	88.36%	61.86%	56.70%	51.11%	0.62%	51.98%	50.14%	9.869%
Test 2	89.42%	59.79%	53.83%	51.09%	0.43%	51.78%	50.34%	5.094%
Test 3	90.90%	59.79%	55.06%	50.40%	0.19%	50.75%	50.14%	8.469%
Test 4	93.65%	58.76%	55.88%	54.56%	0.58%	55.68%	53.83%	2.370%
Test 5	89.21%	60.82%	54.45%	53.37%	0.38%	53.63%	52.40%	1.987%
Mean	90.31%	60.21%	55.19%	52.10%				5.56%
Std	2.08%	1.18%	1.14%	1.77%				2.00%

Table I.4: This table shows the performance achieved when replacing missing values with the mean of the five previous values for the Scania B stock, while using the cleaning with noise technique. The effectiveness of using the error level on the validation set as a stopping criterion can also be seen.

Appendix J

Results: Using Volume as Input Variable

This appendix contains a more detailed description of the results obtained during the 'Using Volume as Input Variable' test, which can be seen in Section 6.11. Each table, one for each stock, contains information concerning the maximum hitrates, stopping hitrates and gaps that were achieved during the test.

Volvo B								
	Highest Hitrate			Stopping on Lowest Error				
Neurons: 6	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	74.81%	68.87%	60.35%	58.71%	0.43%	59.38%	58.03%	2.710%
Test 2	73.26%	73.58%	62.09%	56.16%	1.20%	58.41%	53.77%	9.544%
Test 3	74.42%	67.92%	62.28%	59.98%	0.93%	61.70%	58.61%	3.698%
Test 4	71.51%	71.70%	60.15%	54.18%	0.79%	55.51%	53.00%	9.939%
Test 5	74.81%	67.92%	60.54%	58.89%	1.14%	60.54%	57.45%	2.730%
Mean	73.76%	70.00%	61.08%	57.58%				5.72%
Std	0.01%	2.53%	1.02%	2.36%				2.26%
Neurons: 10	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	82.36%	66.98%	61.12%	59.29%	0.52%	60.35%	58.41%	2.992%
Test 2	80.23%	70.75%	62.86%	60.44%	0.60%	61.32%	59.57%	3.860%
Test 3	82.56%	64.15%	61.90%	57.64%	0.76%	59.38%	57.06%	6.875%
Test 4	81.01%	65.09%	59.57%	58.20%	0.68%	58.80%	57.06%	2.302%
Test 5	79.65%	70.75%	61.90%	58.70%	0.69%	59.38%	57.06%	5.170%
Mean	81.16%	67.55%	61.47%	58.85%				4.24%
Std	0.01%	3.10%	1.23%	1.07%				1.14%
Neurons: 15	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	87.98%	66.04%	61.70%	60.35%	0.75%	61.70%	58.80%	2.194%
Test 2	91.09%	66.04%	61.70%	60.44%	0.48%	61.32%	59.77%	2.052%
Test 3	87.60%	66.98%	61.51%	58.94%	1.03%	60.73%	57.25%	4.174%
Test 4	87.98%	68.87%	61.70%	60.07%	0.38%	60.73%	59.38%	2.650%
Test 5	88.37%	74.53%	61.32%	57.20%	0.91%	58.22%	55.90%	6.711%
Mean	88.60%	68.49%	61.59%	59.40%				3.56%
Std	0.01%	3.57%	0.17%	1.37%				1.19%
Neurons: 20	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	95.93%	70.75%	62.28%	59.45%	0.63%	60.54%	58.61%	4.545%
Test 2	92.44%	67.92%	59.96%	55.88%	0.91%	57.25%	54.55%	6.803%
Test 3	93.41%	70.75%	60.54%	55.65%	0.93%	57.25%	54.35%	8.074%
Test 4	93.80%	71.70%	60.73%	58.64%	0.57%	60.15%	57.83%	3.445%
Test 5	92.64%	70.75%	58.99%	58.38%	0.42%	58.99%	57.64%	1.043%
Mean	93.64%	70.38%	60.50%	57.60%				4.78%
Std	0.01%	1.43%	1.20%	1.72%				1.67%

Table J.1: This table shows the performance achieved when using the volume of the Volvo B stock as an additional input variable, when examining the Volvo B stock. In addition to this, the effectiveness of using the error level on the validation set as a stopping criterion can be seen.

Scania B								
	Highest Hitrate			Stopping on Lowest Error				
Neurons: 6	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	74.61%	63.21%	58.41%	51.75%	0.58%	52.42%	51.06%	11.409%
Test 2	74.61%	66.04%	58.61%	53.54%	0.87%	55.13%	52.22%	8.641%
Test 3	73.84%	66.98%	59.38%	57.46%	0.74%	58.80%	56.48%	3.228%
Test 4	75.19%	63.21%	59.57%	54.02%	0.85%	55.90%	52.61%	9.327%
Test 5	72.29%	67.92%	58.99%	55.11%	0.49%	55.90%	54.35%	6.587%
Mean	74.11%	65.47%	58.99%	54.38%				7.84%
Std	0.01%	2.17%	0.49%	2.11%				1.81%
Neurons: 10	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	82.17%	68.87%	59.38%	52.03%	0.34%	52.42%	51.45%	12.378%
Test 2	78.29%	66.04%	61.51%	56.55%	1.93%	60.35%	54.93%	8.062%
Test 3	78.10%	66.04%	58.22%	53.46%	0.68%	54.55%	52.22%	8.185%
Test 4	75.58%	66.04%	57.64%	54.18%	0.76%	55.32%	53.19%	6.010%
Test 5	79.46%	70.75%	59.38%	52.77%	1.07%	54.74%	51.26%	11.134%
Mean	78.72%	67.55%	59.23%	53.80%				9.15%
Std	0.02%	2.17%	1.48%	1.73%				1.55%
Neurons: 15	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	85.66%	65.09%	58.61%	55.02%	0.81%	57.25%	54.35%	6.121%
Test 2	87.60%	67.92%	58.41%	55.44%	0.38%	56.09%	54.93%	5.087%
Test 3	84.50%	66.04%	60.35%	56.59%	0.56%	57.45%	55.90%	6.236%
Test 4	86.24%	66.04%	60.73%	55.25%	0.58%	56.09%	54.35%	9.033%
Test 5	90.31%	67.92%	56.48%	52.66%	0.67%	53.38%	51.84%	6.756%
Mean	86.86%	66.60%	58.92%	54.99%				6.65%
Std	0.02%	1.27%	1.71%	1.43%				0.94%
Neurons: 20	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	90.50%	68.87%	59.77%	54.93%	0.62%	55.51%	53.38%	8.091%
Test 2	92.25%	65.09%	61.32%	55.92%	0.75%	56.67%	54.55%	8.804%
Test 3	91.67%	64.15%	58.61%	53.77%	0.69%	55.32%	52.61%	8.251%
Test 4	89.73%	65.09%	59.38%	55.21%	1.00%	56.67%	53.38%	7.018%
Test 5	93.22%	69.81%	57.25%	53.47%	0.88%	54.93%	52.61%	6.603%
Mean	91.47%	66.60%	59.27%	54.66%				7.75%
Std	0.01%	2.55%	1.50%	1.02%				0.63%

Table J.2: This table shows the performance achieved when using the volume of the Scania B stock as an additional input variable, when examining the Scania B stock. In addition to this, the effectiveness of using the error level on the validation set as a stopping criterion can be seen.

Appendix K

Results: SE Banken A

This appendix contains a more detailed description of the results obtained during the 'SE Banken A' test, which can be seen in Section 6.12. Each table contains information concerning the maximum hitrates, stopping hitrates and gaps achieved during the test.

SE Banken A								
	Highest Hitrate			Stopping on Lowest Error				
Neurons: 6	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	80.04%	76.15%	65.89%	63.14%	0.63%	64.73%	62.40%	4.171%
Test 2	79.84%	78.46%	65.31%	63.34%	0.37%	63.76%	62.60%	3.021%
Test 3	78.68%	74.62%	66.67%	63.30%	0.61%	64.53%	62.40%	5.048%
Test 4	79.26%	76.92%	66.86%	64.80%	0.47%	65.89%	64.34%	3.083%
Test 5	79.84%	77.69%	64.34%	61.42%	1.14%	63.57%	60.27%	4.545%
Mean	79.53%	76.77%	65.81%	63.20%				3.97%
Std	0.56%	1.48%	1.03%	1.20%				0.59%
Neurons: 10	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	83.91%	76.92%	66.86%	63.37%	0.68%	64.53%	62.40%	5.217%
Test 2	84.11%	79.23%	65.70%	63.21%	0.36%	63.76%	62.79%	3.781%
Test 3	82.95%	79.23%	64.15%	62.90%	0.56%	63.95%	62.21%	1.950%
Test 4	83.91%	80.77%	64.92%	60.08%	0.53%	60.85%	59.30%	7.463%
Test 5	83.33%	78.46%	64.53%	61.36%	0.43%	62.60%	61.05%	4.914%
Mean	83.64%	78.92%	65.23%	62.18%				4.67%
Std	0.49%	1.40%	1.08%	1.42%				1.32%
Neurons: 15	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	91.47%	79.23%	66.47%	63.11%	0.25%	63.37%	62.60%	5.062%
Test 2	94.57%	78.46%	66.67%	62.93%	0.20%	63.18%	62.60%	5.603%
Test 3	87.21%	76.92%	64.73%	63.11%	0.68%	64.34%	62.21%	2.504%
Test 4	88.95%	80.00%	65.70%	62.19%	0.28%	62.60%	61.82%	5.337%
Test 5	87.79%	79.23%	66.09%	64.24%	0.66%	66.09%	63.57%	2.799%
Mean	90.00%	78.77%	65.93%	63.11%				4.26%
Std	3.04%	1.17%	0.77%	0.73%				1.00%

Table K.1: This table shows the performance achieved when determining the optimum number of neurons for the the general architecture and the SE Banken A stock. In addition to this, the effectiveness of using the error level on the validation set as a stopping criterion can be seen.

SE Banken A								
	Highest Hitrate			Stopping on Lowest Error				
TM: 7	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	82.36%	78.46%	65.70%	63.05%	0.38%	63.57%	62.21%	4.023%
Test 2	85.27%	78.46%	65.12%	63.00%	0.46%	64.34%	62.79%	3.247%
Test 3	83.91%	79.23%	64.73%	61.54%	0.48%	62.21%	60.85%	4.927%
Test 4	85.85%	76.92%	65.70%	63.23%	0.42%	63.95%	62.60%	3.754%
Test 5	84.50%	75.38%	67.25%	63.74%	0.71%	64.53%	62.02%	5.214%
Mean	84.38%	77.69%	65.70%	62.91%				4.23%
Std	1.35%	1.54%	0.96%	0.82%				0.56%
TM: 13	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	83.91%	76.92%	66.86%	63.37%	0.68%	64.53%	62.40%	5.217%
Test 2	84.11%	79.23%	65.70%	63.21%	0.36%	63.76%	62.79%	3.781%
Test 3	82.95%	79.23%	64.15%	62.90%	0.56%	63.95%	62.21%	1.950%
Test 4	83.91%	80.77%	64.92%	60.08%	0.53%	60.85%	59.30%	7.463%
Test 5	83.33%	78.46%	64.53%	61.36%	0.43%	62.60%	61.05%	4.914%
Mean	83.64%	78.92%	65.23%	62.18%				4.67%
Std	0.49%	1.40%	1.08%	1.42%				1.32%
TM: 20	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	84.30%	76.15%	65.70%	63.78%	0.80%	65.50%	62.98%	2.923%
Test 2	85.47%	77.69%	65.70%	63.25%	0.40%	64.15%	62.79%	3.728%
Test 3	85.27%	78.46%	66.09%	63.83%	0.30%	64.15%	62.98%	3.412%
Test 4	87.40%	74.62%	65.31%	63.90%	0.28%	64.34%	63.37%	2.158%
Test 5	81.59%	76.15%	67.05%	62.68%	0.30%	62.98%	62.02%	6.516%
Mean	84.81%	76.62%	65.97%	63.49%				3.75%
Std	2.12%	1.50%	0.67%	0.52%				1.13%
TM: 27	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	85.85%	76.92%	66.47%	65.91%	0.41%	66.47%	65.12%	0.848%
Test 2	85.66%	76.92%	66.09%	63.58%	0.22%	63.95%	63.18%	3.786%
Test 3	87.02%	76.15%	65.50%	64.11%	0.75%	65.12%	62.21%	2.125%
Test 4	87.60%	75.38%	65.89%	64.76%	0.61%	65.89%	63.95%	1.711%
Test 5	85.47%	76.92%	65.70%	63.87%	0.40%	64.73%	63.37%	2.789%
Mean	86.32%	76.46%	65.93%	64.45%				2.25%
Std	0.94%	0.69%	0.37%	0.93%				0.73%

Table K.2: This table shows the performance achieved when determining the optimum target multiplier for the general architecture and the SE Banken A stock. In addition to this, the effectiveness of using the error level on the validation set as a stopping criterion can be seen.

SE Banken A								
	Highest Hitrate			Stopping on Lowest Error				
Neurons: 6	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	84.07%	77.69%	65.58%	61.77%	0.62%	62.88%	60.96%	5.812%
Test 2	84.26%	79.23%	65.38%	61.33%	0.84%	62.50%	59.81%	6.203%
Test 3	85.60%	78.46%	65.00%	61.26%	0.53%	62.12%	60.58%	5.756%
Test 4	83.88%	81.54%	65.19%	61.52%	0.99%	63.27%	59.62%	5.632%
Test 5	85.41%	80.77%	66.15%	62.48%	1.24%	65.19%	61.15%	5.550%
Mean	84.64%	79.54%	65.46%	61.67%				5.79%
Std	0.80%	1.60%	0.44%	0.49%				0.16%
Neurons: 10	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	89.44%	79.23%	66.54%	61.47%	0.43%	62.12%	60.77%	7.620%
Test 2	91.17%	79.23%	65.77%	63.81%	0.71%	64.81%	62.31%	2.977%
Test 3	91.55%	76.92%	66.35%	63.29%	0.78%	64.42%	61.73%	4.611%
Test 4	90.02%	81.54%	66.54%	63.85%	0.48%	64.23%	62.69%	4.046%
Test 5	91.55%	77.69%	65.19%	62.52%	0.37%	63.27%	61.92%	4.103%
Mean	90.75%	78.92%	66.08%	62.99%				4.67%
Std	0.96%	1.77%	0.59%	1.00%				1.18%
Neurons: 15	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	95.59%	77.69%	65.58%	63.71%	1.70%	65.58%	60.96%	2.853%
Test 2	95.78%	76.92%	65.58%	60.00%	0.85%	61.15%	58.27%	8.504%
Test 3	94.43%	76.92%	63.65%	59.13%	1.05%	60.77%	57.69%	7.113%
Test 4	95.59%	78.46%	66.15%	63.30%	0.61%	64.23%	62.50%	4.308%
Test 5	94.05%	78.46%	64.42%	62.03%	0.40%	62.88%	61.54%	3.718%
Mean	95.09%	77.69%	65.08%	61.63%				5.30%
Std	0.79%	0.77%	1.01%	2.01%				1.55%
Neurons: 20	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	98.46%	80.00%	66.92%	61.92%	1.38%	64.23%	59.42%	7.471%
Test 2	98.46%	82.31%	64.42%	60.89%	0.63%	61.92%	59.81%	5.482%
Test 3	97.70%	79.23%	64.42%	61.22%	0.65%	61.92%	60.00%	4.966%
Test 4	98.66%	81.54%	65.77%	60.61%	1.63%	65.00%	59.04%	7.842%
Test 5	99.62%	80.00%	63.85%	58.88%	0.81%	60.00%	57.50%	7.777%
Mean	98.58%	80.62%	65.08%	60.71%				6.71%
Std	0.69%	1.26%	1.25%	1.13%				0.93%
Neurons: 25	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	99.62%	76.92%	65.96%	62.31%	0.71%	63.08%	60.77%	5.539%
Test 2	100.00%	79.23%	64.62%	60.12%	0.46%	61.15%	59.62%	6.953%
Test 3	99.42%	80.00%	65.77%	60.66%	0.94%	62.12%	59.04%	7.762%
Test 4	99.23%	80.77%	65.00%	60.66%	0.67%	62.12%	59.62%	6.670%
Test 5	99.42%	80.00%	65.58%	60.94%	0.97%	62.12%	59.62%	7.065%
Mean	99.54%	79.38%	65.38%	60.94%				6.80%
Std	0.29%	1.48%	0.56%	0.82%				0.53%

Table K.3: This table shows the performance achieved when determining the optimum number of neurons for the selected architecture and the SE Banken A stock. In addition to this, the effectiveness of using the error level on the validation set as a stopping criterion can be seen.

SE Banken A								
	Highest Hitrate			Stopping on Lowest Error				
TM: 7	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	96.74%	81.54%	65.58%	62.80%	0.96%	64.04%	60.77%	4.239%
Test 2	93.86%	78.46%	65.38%	63.04%	0.54%	63.65%	62.31%	3.583%
Test 3	94.24%	78.46%	66.15%	63.83%	0.49%	64.42%	62.88%	3.515%
Test 4	95.59%	80.77%	66.15%	62.83%	0.36%	63.27%	62.31%	5.021%
Test 5	94.43%	77.69%	65.19%	62.41%	0.73%	63.65%	60.96%	4.264%
Mean	94.97%	79.38%	65.69%	62.98%				4.12%
Std	1.18%	1.67%	0.44%	0.52%				0.41%
TM: 13	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	98.46%	80.00%	66.92%	61.92%	1.38%	64.23%	59.42%	7.471%
Test 2	98.46%	82.31%	64.42%	60.89%	0.63%	61.92%	59.81%	5.482%
Test 3	97.70%	79.23%	64.42%	61.22%	0.65%	61.92%	60.00%	4.966%
Test 4	98.66%	81.54%	65.77%	60.61%	1.63%	65.00%	59.04%	7.842%
Test 5	99.62%	80.00%	63.85%	58.88%	0.81%	60.00%	57.50%	7.777%
Mean	98.58%	80.62%	65.08%	60.71%				6.71%
Std	0.69%	1.26%	1.25%	1.13%				0.93%
TM: 20	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	98.46%	80.77%	65.38%	62.66%	1.10%	65.19%	61.35%	4.171%
Test 2	98.66%	78.46%	65.96%	62.71%	0.65%	63.85%	61.54%	4.930%
Test 3	99.23%	77.69%	64.42%	62.40%	1.15%	63.85%	60.77%	3.148%
Test 4	99.62%	78.46%	66.15%	61.68%	0.81%	63.27%	60.19%	6.765%
Test 5	99.42%	80.77%	65.19%	61.59%	1.16%	63.46%	59.04%	5.524%
Mean	99.08%	79.23%	65.42%	62.21%				4.91%
Std	0.50%	1.44%	0.69%	0.54%				0.92%
TM: 27	Training	Validation	Generalization	Mean	Std	Max	Min	Gap
Test 1	99.81%	79.23%	61.92%	59.37%	1.70%	61.35%	54.81%	4.122%
Test 2	99.42%	78.46%	66.15%	62.47%	1.14%	64.04%	60.58%	5.576%
Test 3	99.62%	79.23%	65.00%	59.81%	0.78%	61.35%	59.04%	7.988%
Test 4	99.23%	76.92%	64.23%	61.78%	0.52%	62.50%	60.58%	3.811%
Test 5	99.23%	77.69%	64.62%	61.24%	1.02%	63.46%	59.81%	5.222%
Mean	99.46%	78.31%	64.38%	60.93%				5.34%
Std	0.25%	1.00%	1.55%	1.31%				1.11%

Table K.4: This table shows the performance achieved when determining the optimum target multiplier for the selected architecture and the SE Banken A stock. In addition to this, the effectiveness of using the error level on the validation set as a stopping criterion can be seen.

Appendix L

Results: Forecasting

This appendix contains a more detailed description of the results obtained during the 'Forecasting' test, which can be seen in Section 6.13. Each table contains information concerning the hitrates, realized potentials and annualized returns that were obtained for each forecast model.

Volvo B: General Architecture						
	Hitrate				Hitrate (SENN)	
Generalization	Norm	SENN	RP	AR	Training	Validation
Period 1:	50.41%	53.79%	0.49%	29.06%	65.43%	63.21%
Period 2:	60.00%	63.08%	10.97%	40.32%		
Period 3:	54.84%	57.58%	4.31%	7.86%		
Period 4:	58.04%	61.79%	16.31%	47.87%		
Period 1-4:	55.74%	58.99%	7.68%	30.37%		
	Hitrate				Hitrate (SENN)	
Generalization	Norm	SENN	RP	AR	Training	Validation
Period 1:	52.85%	56.06%	11.90%	52.43%	63.50%	62.26%
Period 2:	58.33%	61.54%	18.91%	58.24%		
Period 3:	50.00%	53.03%	15.83%	24.76%		
Period 4:	58.04%	61.79%	18.76%	51.91%		
Period 1-4:	54.70%	58.03%	16.21%	46.22%		
	Hitrate				Hitrate (SENN)	
Generalization	Norm	SENN	RP	AR	Training	Validation
Period 1:	51.22%	54.55%	3.89%	34.48%	66.40%	64.15%
Period 2:	53.33%	56.92%	-0.51%	17.02%		
Period 3:	57.26%	59.85%	18.94%	30.46%		
Period 4:	59.82%	63.41%	37.35%	92.57%		
Period 1-4:	55.32%	58.61%	13.39%	41.01%		
	Hitrate				Hitrate (SENN)	
Generalization	Norm	SENN	RP	AR	Training	Validation
Period 1:	55.28%	58.33%	16.58%	66.82%	67.20%	57.55%
Period 2:	52.50%	56.15%	5.53%	29.42%		
Period 3:	59.68%	62.12%	16.45%	26.74%		
Period 4:	58.93%	62.60%	33.46%	84.17%		
Period 1-4:	56.58%	59.77%	17.22%	49.83%		
	Hitrate				Hitrate (SENN)	
Generalization	Norm	SENN	RP	AR	Training	Validation
Period 1:	55.28%	58.33%	-3.37%	21.76%	65.27%	61.32%
Period 2:	50.00%	53.85%	0.82%	20.17%		
Period 3:	58.06%	60.61%	6.38%	11.68%		
Period 4:	51.79%	56.10%	6.13%	30.77%		
Period 1-4:	53.86%	57.25%	2.05%	20.90%		

Table L.1: This table shows the performance achieved when forecasting the Volvo B stock, while using the general architecture.

Volvo B: Selected Architecture						
	Hitrate				Hitrate (SENN)	
Generalization	Norm	SENN	RP	AR	Training	Validation
Period 1:	50.41%	53.79%	-12.15%	4.99%	66.67%	66.98%
Period 2:	53.33%	56.92%	7.73%	33.67%		
Period 3:	54.03%	56.82%	-5.12%	-4.12%		
Period 4:	58.26%	62.20%	7.59%	34.27%		
Period 1-4:	53.94%	57.39%	-0.72%	15.94%		
	Hitrate				Hitrate (SENN)	
Generalization	Norm	SENN	RP	AR	Training	Validation
Period 1:	52.03%	55.30%	-11.74%	5.73%	68.26%	66.04%
Period 2:	47.50%	51.54%	-3.62%	11.90%		
Period 3:	48.39%	51.52%	-13.18%	-13.71%		
Period 4:	59.13%	62.99%	5.68%	30.97%		
Period 1-4:	51.66%	55.28%	-5.91%	7.53%		
	Hitrate				Hitrate (SENN)	
Generalization	Norm	SENN	RP	AR	Training	Validation
Period 1:	56.91%	59.85%	11.50%	52.95%	69.70%	69.81%
Period 2:	51.67%	55.38%	3.41%	24.92%		
Period 3:	47.58%	50.76%	-3.68%	-2.77%		
Period 4:	59.13%	62.99%	10.39%	38.81%		
Period 1-4:	53.73%	57.20%	5.69%	26.72%		
	Hitrate				Hitrate (SENN)	
Generalization	Norm	SENN	RP	AR	Training	Validation
Period 1:	56.10%	59.09%	-2.78%	22.06%	69.06%	65.09%
Period 2:	53.33%	56.92%	-3.62%	11.69%		
Period 3:	46.77%	50.00%	-7.45%	-7.39%		
Period 4:	60.87%	64.57%	14.09%	46.50%		
Period 1-4:	54.15%	57.58%	-0.22%	16.62%		
	Hitrate				Hitrate (SENN)	
Generalization	Norm	SENN	RP	AR	Training	Validation
Period 1:	50.41%	53.79%	-10.46%	7.85%	66.67%	59.43%
Period 2:	54.17%	57.69%	4.01%	26.26%		
Period 3:	51.61%	54.55%	-4.44%	-3.15%		
Period 4:	54.78%	59.06%	-2.66%	17.17%		
Period 1-4:	52.70%	56.24%	-3.43%	11.49%		

Table L.2: This table shows the performance achieved when forecasting the Volvo B stock, while using the selected architecture.

Scania B: General Architecture						
	Hitrate				Hitrate (SENN)	
Generalization	Norm	SENN	RP	AR	Training	Validation
Period 1:	43.36%	51.52%	-6.02%	-2.64%	68.81%	64.15%
Period 2:	46.22%	50.77%	-0.59%	23.73%		
Period 3:	58.20%	61.36%	25.50%	41.47%		
Period 4:	53.51%	56.91%	13.49%	24.52%		
Period 1-4:	50.43%	55.13%	7.49%	20.70%		
	Hitrate				Hitrate (SENN)	
Generalization	Norm	SENN	RP	AR	Training	Validation
Period 1:	46.02%	53.79%	-4.38%	0.49%	60.61%	59.43%
Period 2:	47.90%	52.31%	-5.63%	15.97%		
Period 3:	54.10%	57.58%	6.20%	13.42%		
Period 4:	38.60%	43.09%	-24.64%	-20.15%		
Period 1-4:	46.79%	51.84%	-6.96%	1.36%		
	Hitrate				Hitrate (SENN)	
Generalization	Norm	SENN	RP	AR	Training	Validation
Period 1:	51.33%	58.33%	9.93%	20.26%	65.76%	56.60%
Period 2:	41.18%	46.15%	-19.48%	-3.60%		
Period 3:	56.56%	59.85%	12.62%	22.07%		
Period 4:	49.12%	52.85%	8.66%	18.16%		
Period 1-4:	49.57%	54.35%	2.29%	13.72%		
	Hitrate				Hitrate (SENN)	
Generalization	Norm	SENN	RP	AR	Training	Validation
Period 1:	53.98%	60.61%	10.43%	20.75%	60.13%	55.66%
Period 2:	49.58%	53.85%	-3.97%	18.31%		
Period 3:	59.84%	62.88%	15.76%	26.17%		
Period 4:	58.77%	61.79%	23.21%	39.51%		
Period 1-4:	55.56%	59.77%	10.78%	25.93%		
	Hitrate				Hitrate (SENN)	
Generalization	Norm	SENN	RP	AR	Training	Validation
Period 1:	53.98%	60.61%	12.35%	24.31%	65.76%	61.32%
Period 2:	42.86%	47.69%	-15.29%	2.28%		
Period 3:	61.48%	64.39%	31.95%	53.54%		
Period 4:	49.12%	52.85%	9.90%	20.29%		
Period 1-4:	51.92%	56.48%	8.92%	23.79%		

Table L.3: This table shows the performance achieved when forecasting the Scania B stock, while using the general architecture.

Scania B: Selected Architecture						
	Hitrate				Hitrate (SENN)	
Generalization	Norm	SENN	RP	AR	Training	Validation
Period 1:	46.90%	54.55%	1.99%	8.10%	66.08%	52.83%
Period 2:	40.34%	45.38%	-16.71%	-0.12%		
Period 3:	59.02%	62.12%	25.69%	42.32%		
Period 4:	49.57%	53.60%	8.68%	16.67%		
Period 1-4:	49.04%	53.95%	4.11%	15.71%		
	Hitrate				Hitrate (SENN)	
Generalization	Norm	SENN	RP	AR	Training	Validation
Period 1:	51.33%	58.33%	16.05%	29.43%	66.56%	64.15%
Period 2:	45.38%	50.00%	-12.79%	5.04%		
Period 3:	58.20%	61.36%	25.24%	41.37%		
Period 4:	54.78%	58.40%	22.72%	37.66%		
Period 1-4:	52.45%	57.03%	12.00%	27.54%		
	Hitrate				Hitrate (SENN)	
Generalization	Norm	SENN	RP	AR	Training	Validation
Period 1:	48.67%	56.06%	5.88%	13.24%	67.04%	60.38%
Period 2:	47.90%	52.31%	1.71%	27.43%		
Period 3:	57.38%	60.61%	24.54%	39.84%		
Period 4:	48.70%	52.80%	5.78%	12.08%		
Period 1-4:	50.75%	55.49%	9.14%	22.64%		
	Hitrate				Hitrate (SENN)	
Generalization	Norm	SENN	RP	AR	Training	Validation
Period 1:	49.56%	56.82%	10.20%	20.10%	66.56%	55.66%
Period 2:	42.02%	46.92%	-16.47%	0.14%		
Period 3:	59.84%	62.88%	24.22%	39.65%		
Period 4:	52.17%	56.00%	17.02%	28.67%		
Period 1-4:	50.96%	55.68%	7.91%	21.25%		
	Hitrate				Hitrate (SENN)	
Generalization	Norm	SENN	RP	AR	Training	Validation
Period 1:	44.25%	52.27%	-6.22%	-2.20%	72.96%	65.09%
Period 2:	51.26%	55.38%	6.39%	36.49%		
Period 3:	67.21%	69.70%	41.66%	71.86%		
Period 4:	47.83%	52.00%	-9.17%	-6.36%		
Period 1-4:	52.88%	57.42%	7.81%	21.07%		

Table L.4: This table shows the performance achieved when forecasting the Scania B stock, while using the selected architecture.

SE Bankan A: General Architecture						
	Hitrate				Hitrate (SENN)	
Generalization	Norm	SENN	RP	AR	Training	Validation
Period 1:	58.59%	68.94%	20.07%	3.99%	77.09%	73.85%
Period 2:	51.38%	58.91%	3.86%	169.63%		
Period 3:	55.05%	62.88%	11.19%	172.27%		
Period 4:	57.55%	63.41%	18.41%	20.90%		
Period 1-4:	55.56%	63.57%	12.24%	74.30%		
	Hitrate				Hitrate (SENN)	
Generalization	Norm	SENN	RP	AR	Training	Validation
Period 1:	54.55%	65.91%	0.33%	-40.01%	74.77%	70.77%
Period 2:	45.87%	54.26%	-5.81%	83.26%		
Period 3:	51.38%	59.85%	9.43%	156.39%		
Period 4:	59.43%	65.04%	16.18%	16.09%		
Period 1-4:	52.72%	61.24%	3.14%	34.50%		
	Hitrate				Hitrate (SENN)	
Generalization	Norm	SENN	RP	AR	Training	Validation
Period 1:	64.65%	73.48%	27.46%	34.04%	76.01%	73.08%
Period 2:	48.62%	56.59%	-6.61%	76.17%		
Period 3:	57.80%	65.15%	20.63%	254.35%		
Period 4:	62.26%	67.48%	23.42%	32.35%		
Period 1-4:	58.16%	65.70%	13.85%	82.42%		
	Hitrate				Hitrate (SENN)	
Generalization	Norm	SENN	RP	AR	Training	Validation
Period 1:	58.59%	68.94%	19.77%	3.55%	76.63%	73.08%
Period 2:	52.29%	59.69%	-4.22%	100.33%		
Period 3:	54.13%	62.12%	11.80%	175.46%		
Period 4:	61.32%	66.67%	17.93%	19.30%		
Period 1-4:	56.50%	64.34%	9.57%	61.59%		
	Hitrate				Hitrate (SENN)	
Generalization	Norm	SENN	RP	AR	Training	Validation
Period 1:	59.60%	69.70%	29.18%	37.66%	75.54%	69.23%
Period 2:	54.13%	61.24%	8.33%	213.78%		
Period 3:	46.79%	56.06%	-13.03%	37.97%		
Period 4:	55.66%	61.79%	5.96%	-4.89%		
Period 1-4:	53.90%	62.21%	8.03%	54.30%		

Table L.5: This table shows the performance achieved when forecasting the SE Banken A stock, while using the general architecture.

SE Banken A: Selected Architecture						
	Hitrate				Hitrate (SENN)	
Generalization	Norm	SENN	RP	AR	Training	Validation
Period 1:	51.52%	63.64%	2.43%	-38.80%	78.80%	76.92%
Period 2:	55.96%	62.79%	21.29%	418.86%		
Period 3:	50.46%	59.09%	0.30%	96.84%		
Period 4:	57.80%	63.78%	12.27%	5.06%		
Period 1-4:	53.99%	62.31%	9.74%	60.08%		
	Hitrate				Hitrate (SENN)	
Generalization	Norm	SENN	RP	AR	Training	Validation
Period 1:	53.54%	65.15%	14.69%	-7.87%	76.96%	74.62%
Period 2:	56.88%	63.57%	0.01%	130.59%		
Period 3:	55.05%	62.88%	11.88%	176.60%		
Period 4:	50.46%	57.48%	1.52%	-15.01%		
Period 1-4:	53.99%	62.31%	6.98%	49.49%		
	Hitrate				Hitrate (SENN)	
Generalization	Norm	SENN	RP	AR	Training	Validation
Period 1:	54.55%	65.91%	4.97%	-33.90%	75.58%	75.38%
Period 2:	52.29%	59.69%	1.22%	136.65%		
Period 3:	52.29%	60.61%	-1.97%	88.18%		
Period 4:	52.29%	59.06%	5.41%	-8.13%		
Period 1-4:	52.82%	61.35%	2.09%	28.24%		
	Hitrate				Hitrate (SENN)	
Generalization	Norm	SENN	RP	AR	Training	Validation
Period 1:	57.58%	68.18%	18.45%	1.30%	74.65%	69.23%
Period 2:	55.05%	62.02%	8.74%	211.03%		
Period 3:	53.21%	61.36%	11.51%	176.09%		
Period 4:	55.96%	62.20%	16.89%	14.38%		
Period 1-4:	55.40%	63.46%	13.27%	77.61%		
	Hitrate				Hitrate (SENN)	
Generalization	Norm	SENN	RP	AR	Training	Validation
Period 1:	53.54%	65.15%	-0.09%	-43.78%	79.11%	73.85%
Period 2:	60.55%	66.67%	0.51%	134.00%		
Period 3:	54.13%	62.12%	16.00%	201.88%		
Period 4:	54.13%	60.63%	3.88%	-11.40%		
Period 1-4:	55.63%	63.65%	4.74%	36.96%		

Table L.6: This table shows the performance achieved when forecasting the SE Banken A stock, while using the selected architecture.

Appendix M

Results: Ericsson B

This appendix contains a more detailed description of the results obtained during the 'Ericsson B' test, which is used to verify the developed model and can be seen in Section 7.2. The table contains information concerning the hitrates, realized potentials and annualized returns that were obtained for each forecast model.

Ericsson B						
	Hitrate				Hitrate (SENN)	
Generalization	Norm	SENN	RP	AR	Training	Validation
Period 1:	45.38%	50.76%	-16.77%	0.00%	59.81%	51.89%
Period 2:	47.37%	53.85%	-21.36%	-0.05%		
Period 3:	49.58%	54.55%	0.63%	0.00%		
Period 4:	42.57%	52.85%	-17.49%	-3.67%		
Period 1-4:	46.36%	53.00%	-14.08%	-0.94%		
	Hitrate				Hitrate (SENN)	
Generalization	Norm	SENN	RP	AR	Training	Validation
Period 1:	48.74%	53.79%	-4.02%	43.73%	64.47%	54.72%
Period 2:	51.75%	57.69%	-5.75%	52.80%		
Period 3:	56.30%	60.61%	23.20%	64.96%		
Period 4:	41.58%	52.03%	-12.44%	3.57%		
Period 1-4:	49.89%	56.09%	0.67%	39.18%		
	Hitrate				Hitrate (SENN)	
Generalization	Norm	SENN	RP	AR	Training	Validation
Period 1:	61.34%	65.15%	18.37%	168.54%	62.38%	60.38%
Period 2:	58.77%	63.85%	8.37%	122.46%		
Period 3:	57.14%	61.36%	26.14%	72.55%		
Period 4:	49.50%	58.54%	2.29%	27.14%		
Period 1-4:	56.95%	62.28%	14.73%	90.27%		
	Hitrate				Hitrate (SENN)	
Generalization	Norm	SENN	RP	AR	Training	Validation
Period 1:	50.42%	55.30%	-3.34%	46.86%	64.47%	55.66%
Period 2:	50.88%	56.92%	-5.37%	54.52%		
Period 3:	52.10%	56.82%	0.60%	-1.11%		
Period 4:	42.57%	52.85%	-12.51%	3.66%		
Period 1-4:	49.23%	55.51%	-4.44%	23.50%		
	Hitrate				Hitrate (SENN)	
Generalization	Norm	SENN	RP	AR	Training	Validation
Period 1:	52.10%	56.82%	-0.98%	56.20%	66.24%	61.32%
Period 2:	53.51%	59.23%	2.02%	88.81%		
Period 3:	52.94%	57.58%	17.51%	44.53%		
Period 4:	48.51%	57.72%	-1.44%	22.06%		
Period 1-4:	51.88%	57.83%	4.28%	51.03%		

Table M.1: This table shows the performance achieved when forecasting the Ericsson B stock, while using the general architecture.

Appendix N

Results: Atlas Copco B

This appendix contains a more detailed description of the results obtained during the 'Atlas Copco B' test, which is used to verify the developed model and can be seen in Section 7.2. The table contains information concerning the hitrates, realized potentials and annualized returns that were obtained for each forecast model.

Atlas Copco B						
	Hitrate				Hitrate (SENN)	
Generalization	Norm	SENN	RP	AR	Training	Validation
Period 1:	59.17%	62.88%	22.08%	48.02%	70.90%	70.00%
Period 2:	50.00%	58.14%	3.55%	-2.53%		
Period 3:	49.53%	59.09%	-12.14%	11.69%		
Period 4:	57.80%	62.60%	33.43%	47.79%		
Period 1-4:	54.28%	60.66%	13.44%	24.23%		
	Hitrate				Hitrate (SENN)	
Generalization	Norm	SENN	RP	AR	Training	Validation
Period 1:	51.67%	56.06%	19.32%	41.00%	69.35%	76.92%
Period 2:	46.30%	55.04%	-1.67%	-7.73%		
Period 3:	55.14%	63.64%	4.48%	31.09%		
Period 4:	56.88%	61.79%	18.09%	23.87%		
Period 1-4:	52.48%	59.11%	11.28%	20.56%		
	Hitrate				Hitrate (SENN)	
Generalization	Norm	SENN	RP	AR	Training	Validation
Period 1:	57.50%	61.36%	17.05%	37.09%	69.35%	73.08%
Period 2:	45.37%	54.26%	-10.15%	-14.82%		
Period 3:	51.40%	60.61%	-7.61%	16.82%		
Period 4:	49.54%	55.28%	-1.72%	-0.26%		
Period 1-4:	51.13%	57.95%	1.26%	8.00%		
	Hitrate				Hitrate (SENN)	
Generalization	Norm	SENN	RP	AR	Training	Validation
Period 1:	55.83%	59.85%	14.65%	32.71%	68.58%	69.23%
Period 2:	47.22%	55.81%	1.49%	-4.28%		
Period 3:	43.93%	54.55%	-20.81%	2.53%		
Period 4:	60.55%	65.04%	33.45%	47.50%		
Period 1-4:	52.03%	58.72%	8.71%	17.73%		
	Hitrate				Hitrate (SENN)	
Generalization	Norm	SENN	RP	AR	Training	Validation
Period 1:	53.33%	57.58%	18.68%	40.16%	70.43%	74.62%
Period 2:	45.37%	54.26%	-3.14%	-8.75%		
Period 3:	52.34%	61.36%	-5.15%	19.39%		
Period 4:	58.72%	63.41%	29.48%	41.04%		
Period 1-4:	52.48%	59.11%	11.45%	21.14%		

Table N.1: This table shows the performance achieved when forecasting the Atlas Copco B stock, while using the general architecture.