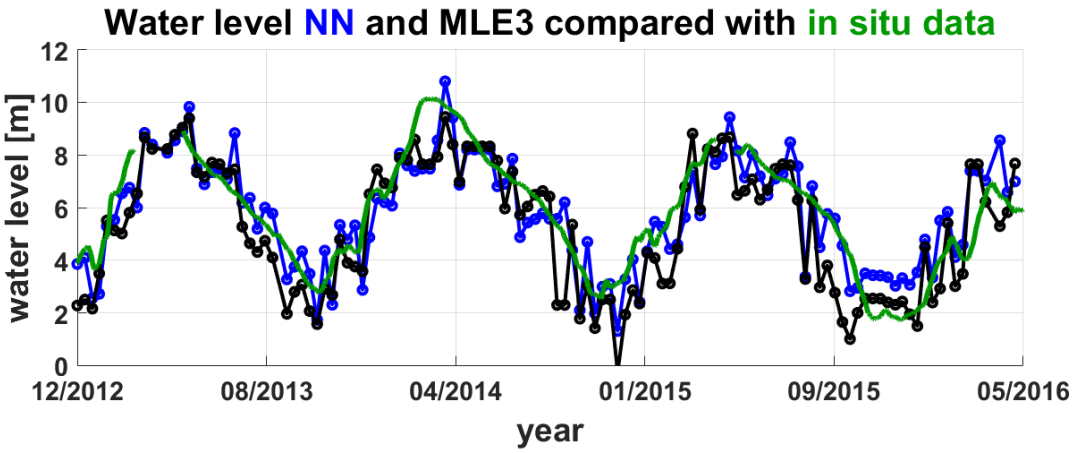
Universität Stuttgart
Geodätisches Institut

Analysis of waveforms in the satellite altimetry by using neural networks



Masterarbeit im Studiengang
Geodäsie & Geoinformatik
an der Universität Stuttgart

Dennis Frederic Mattes

Stuttgart, April 2019

Betreuer: M.Sc. Omid Elmi
Universität Stuttgart

Prof. Dr.-Ing. Nico Sneeuw
Universität Stuttgart

Erklärung der Urheberschaft

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form in keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Ort, Datum Unterschrift

The in situ data of inland water bodies is only limited and is declining lately. At the same time, it is more and more important to monitor the inland water bodies, since the climate is changing rapidly.

To handle this problem, space born sensors are used more and more. One of the possibilities is to use satellite altimetry, which was previously designed for measurements over the oceans. Thereby, the satellite is transmitting a radar signal towards the earth surface at nadir. This signal is reflected by the ground back to the satellite. By doing so, it estimates the surface height with the runtime of the signal. However, caused by the fast changing terrain over the inland, more noise is included and lead to errors in the height estimation. To solve this, retracker are applied which analyse the received signal and estimate the correct runtime.

In this thesis, a new approach will be presented which aims to use neural networks for the retracking purpose. The advantage is that neural networks can learn the characteristic pattern of the signals and then find this pattern during the retracking process. Thereby two approaches are developed, one which uses solely a neural network and a second one, which uses the results of the neural network as an input for an algorithm. They are then applied to different study areas to analyse their performance.

It could be shown that the neural networks can estimate the water height well so that a reasonable water height time series can be created. Thereby, the neural network approach shows better results than the algorithm. At the end also the transferability of the neural networks could be shown. Thus, one can use a trained neural network also on other water bodies as which are used for training.

Contents

1 Introduction **1**
1.1 Introduction . . . 1
1.2 Motivation . . . 2
1.3 Objectives . . . 3
1.4 Overview of the analysed water areas . . . 3
1.4.1 The Cupari river . . . 3
1.4.2 Lago do Madabá Grande . . . 3
1.4.3 Lake Tana . . . 6
1.4.4 Data . . . 8
1.5 Outline of the thesis . . . 9

2 Introduction in satellite altimetry **11**
2.1 Satellite altimetry . . . 11
2.1.1 Basics of satellite altimetry . . . 11
2.1.2 Applied corrections . . . 14
2.2 Overview of the common retracking algorithms . . . 20
2.2.1 Empirical waveform retracking . . . 21
2.2.2 Physically based waveform retracking . . . 25

3 Neural networks **31**
3.1 From the biological to the artificial neuron . . . 31
3.2 The network . . . 33
3.3 A machine learning perspective . . . 35
3.4 How neural networks learn . . . 36
3.4.1 The backpropagation . . . 37
3.4.2 The gradient descent . . . 40
3.4.3 Changing the weights of the network . . . 41
3.4.4 The output of a neural network . . . 44
3.5 The implementation of a neural network . . . 44
3.5.1 Forward propagation of the signals . . . 44
3.5.2 Changing the weights with matrices . . . 45
3.5.3 Defining initial values . . . 46

4 New retracking approaches **49**
4.1 A neural network approach . . . 49
4.1.1 Processing the data . . . 49
4.1.2 Defining the network parameter . . . 51
4.1.3 First test of the approach . . . 53
4.2 Subwaveform classification by using neural networks . . . 63
4.2.1 A new retracking algorithm . . . 63

4.2.2 Comparing the results with in situ data 66
4.3 Conclusions 69

5 Results **71**
5.1 Analysing the results 71
5.1.1 The Cupari river 71
5.1.2 Lago do Madabá Grande 81
5.1.3 Lake Tana 86
5.1.4 The final test 91

6 Conclusion and outlook **97**
6.1 Overview of the work and conclusions 97
6.2 Outlook 99

A Appendix **XXI**
A.1 Parameter selection for the standard neural network XXI
A.2 Parameter selection for the neural network in the algorithm XXIII

List of Figures

1.1 Location of the Cuapri river (source: Greene (2019)) 4
1.2 Area in which the Jason-2 satellite crosses the Cuapri river, in red the chosen area is marked (source: https://landlook.usgs.gov/viewer.html) 4
1.3 Position of the Lago do Madabá Grande (source: Greene (2019)) 5
1.4 Track of Jason-2 across Lago do Madabá Grande and in red the chosen area is marked (source: https://landlook.usgs.gov/viewer.html) 5
1.5 Two images of the floodplain taken by Sentinel 2 at the 7th of July and 12th of December 2016 respectively (source: https://landlook.usgs.gov/viewer.html) . . 6
1.6 Location of Lake Tana in Ethiopia (source: Greene (2019)) 7
1.7 Jason-2 track across Lake Tana and in red the choen area is marked (source: https://landlook.usgs.gov/viewer.html) 7
1.8 Jason-2 satellite in the orbit (Source: https://directory.eoportal.org/web/eoportal/satellite-missions/j/jason-2) 8

2.1 Creation of the waveform (Tourian, 2012) 12
2.2 Schematic graphic of the parts of a waveform over ocean surface (Tourian, 2012) 12
2.3 Basic principle of the satellite altimetry 12
2.4 Schematic graph of needed corrections in the radar altimetry (www.altimetry.info) 15
2.5 Principle of retracking with the nominal gate (tracking gate) and the retracked gate in red (Tourian, 2012) 20
2.6 Schematic diagram of the OCOG retracker (Tourian, 2012) 21
2.7 Threshold retracker with a typical waveform from ice-sheet altimeter measurements. It is to mention that the threshold is referenced to a defined percentage of the maximum waveform amplitude above the DC level in front of the leading edge (Davis, 1997) 23
2.8 Sketches of the 5β and 9β fitting algorithm which are applied to single-ramp and double-ramp waveforms respectively (Roohi, 2015) 24

3.1 Schematic representation of a biological neuron (Source: Source: https://askabiologist.asu.edu) 32
3.2 Example of a step function 32
3.3 Example of a sigmoid funciton 33
3.4 Artificial Neuron with three input signals 33
3.5 Single layer feed foreward neural network 34
3.6 Data set which can be separated by using a linear function 35
3.7 Neural Network without hidden layer 35
3.8 Data set which is not possible to separate with one linear function 36
3.10 First step of the backpropagation 37
3.9 Each connection has its own weight 37
3.11 Closer look at a neuron in the output layer 38
3.12 Correction of the weights with the distributed error 38

3.13 Backpropagation to the hidden layers . . . 39
3.14 Gradient descent towards the minimum . . . 40
3.15 Graphical representation of the error function depending on two weights (Gardner and Dorling, 1998) . . . 41

4.1 Normalised waveforms with marked maximum peak . . . 50
4.2 Examples of separation of the leading edge . . . 52
4.3 L2 (a), L1 (b) and Huber (c) loss function versus the predictions . . . 53
4.4 L1 loss function for 100 test data . . . 54
4.5 Error rate for 100 test data . . . 55
4.6 Time series of the first test with 600 waveforms for training . . . 56
4.7 Accuracy of the neural network with 30 runs . . . 57
4.8 Accuracy of the neural network with different numbers of hidden neurons (a) and learning rate (b) . . . 58
4.9 Change in the accuracy depending on the number of epochs . . . 58
4.10 Loss function and error rate for the third run of the neural network . . . 59
4.11 Time series of the second test with 600 waveforms (a) for training and the residuals of it (b) . . . 61
4.12 Time series and residuals after the retracking which was done by a neural network with 720 and 900 training waveforms respectively . . . 61
4.13 Scatter plot of the retracked water hight with 600, 720 and 900 waveforms against the in situ data . . . 62
4.14 Examples of waveforms with the search window of 30 bins . . . 63
4.15 Diagramm of the developed algorithm . . . 65
4.16 Time series and residuals after using 10 neurons in the input layer and using 600 (a), (c) and 900 (b), (d) waveforms for training respectively . . . 67
4.17 Time series and residuals after using 30 neurons in the input layer and using 600 (a), (c) and 900 (b), (d) waveforms for training respectively . . . 68

5.1 Radargram of the Cupari river (a) where the waveforms of one track are plotted along the y-axis, whereas the longitude of the satellite track is along the x-axis. The power level of the waveforms is indicated by the colour bar. An overview of the study area is shown in (b) . . . 72
5.2 Time series retracked by the neural network and the MLE3 retracker (a) and the threshold retracker (b) compared to the in situ data respectively by using 900 waveforms for training at the Cupari river . . . 73
5.3 Histogram of the residuals of the time series created by the neural network and the MLE3 retracker (a) and the threshold retracker (b) regarding the in situ data at the Cupari river . . . 73
5.4 Chosen area in the time series and also the detailed plot of the results . . . 74
5.5 The first two waveforms and the retracked leading edges of the data set . . . 75
5.6 Chosen area in the time series and also the detailed plot of the results with good performance of MLE3 . . . 75
5.7 The waveforms and the retracked leading edges of the data set . . . 78
5.8 Time series, retracked by the algorithm and by the MLE3 retracker (a) and by the threshold retracker (b) compared with the in situ data respectively at the Cupari river . . . 78

5.10 Chosen area in the time series and the detailed plot of the results of algorithm, MLE3 and the network in comparison 79
5.11 The waveforms and the retracked leading edges of the algorithm (left) compared with the network result (right) 80
5.12 Radargram of the track with the chosen area (a) and an overview of the area (b) at Lago do Madabá Grande 81
5.9 Histogram of the residuals of the time series created by the algorithm and the MLE3 retracker (a) and the threshold retracker (b) regarding the in situ time series respectively at the Cupari river 83
5.13 Comparison of the time series, based on the neural network and the MLE3 retracker (a) and the threshold retracker (b) with the DAHITI time series respectively at Lago do Madabá Grande 83
5.14 Histograms of the residuals of the time series based on the neural network, compared to the one of the MLE3 retracker (a) and the threshold retracker (b) at Lago do Madabá Grande 85
5.15 Comparison of the time series, based on the algorithm and the MLE3 retracker (a) and the threshold retracker (b) with the DAHITI time series respectively at Lago do Madabá Grande 85
5.17 Radargram of the track with the chosen area (a) and an overview of the area (b) at Lake Tana 86
5.16 Histograms of the residuals of the time series based on the algorithm, compared to the one of the MLE3 retracker (a) and the threshold retracker (b) at Lago do Madabá Grande 87
5.18 Comparison of the time series, based on the neural network and the MLE3 retracker (a) and the threshold retracker (b) with the DAHITI time series respectively at Lake Tana 87
5.19 Waveforms measured over lake Tana 89
5.20 Comparison of the time series, based on the neural network and the MLE3 retracker (a) and the threshold retracker (b) with the DAHITI time series respectively at Lake Tana by using a threshold retracker for classification 89
5.21 Histograms of the residuals of the time series based on the neural network, compared to the one of the MLE3 retracker (a) and the threshold retracker (b) at Lake Tana 92
5.22 Comparison of the time series, based on the algorithm and the MLE3 retracker (a) and the threshold retracker (b) with the DAHITI time series respectively at Lake Tana by using a threshold retracker for classification 92
5.23 Histograms of the residuals of the time series based on the algorithm, compared to the one of the MLE3 retracker (a) and the threshold retracker (b) at Lake Tana 94
5.24 Comparison of the time series, based on the neural network, which is trained with data of another study area, and the MLE3 retracker (a) and the threshold retracker (b) with the in situ time series respectively at the Cupari river 94
5.25 Histograms of the residuals of the time series based on the neural network, which is trained with data of another study area, compared to the one of the MLE3 retracker (a) and the threshold retracker (b) at the Cupari river 95

6.1 Time series of the smoothed neural network after it was trained with data of another area compared with in situ data 98

A.1 Accuracy of the neural network which is trained with 120 training data sets with different numbers of hidden neurons (a) and learning rate (b) XXI
A.2 Changes in the accuracy by increasing the number of epochs with 720 training waveforms XXII
A.3 Accuracy of the neural network which is trained with 150 training data sets with different numbers of hidden neurons (a) and learning rate (b) XXII
A.4 Changes in the accuracy by increasing the numbre of epochs with 900 training waveforms XXIII
A.5 Accuracy of the neural network which uses ten input neurons with different numbers of hidden neurons with 600 (a) and 900 (c) waveforms respectively and also the learning rate with 600 (b) and 900 (d) waveforms respectively XXIV
A.6 Accuracy of the neural network which uses ten input neurons by increasing the number of epochs for 600 (a) and 900 (b) waveforms respectively XXV
A.7 Accuracy of the neural network which uses 30 input neurons with different numbers of hidden neurons with 600 (a) and 900 (c) waveforms respectively and also the learning rate with 600 (b) and 900 (d) waveforms respectively XXVI
A.8 Accuracy of the neural network which uses 30 input neurons by increasing the number of epochs for 600 (a) and 900 (b) waveforms respectively XXVI

List of Tables

1.1 Orbit parameter of OSTM/Jason 2 8
1.2 Used tracks and time frame for the current study 9

4.1 Number of the used waveforms 51
4.2 Mean and RMSE of the first test of the neural network 57
4.3 Changes of the network parameters during the second correction 59
4.4 Mean and RMSE of the first test of the neural network 59
4.5 Set of parameter which is used with 600, 720 and 900 training waveforms 60
4.6 Standard deviation, median and RMSE for tests with 600, 720 and 900 training waveforms 62
4.7 Neural network parameters for a network with 10 input neurons 66
4.8 The accuracy, mean and RMSE for tests with 600 and 900 training waveforms with 10 input neurons 67
4.9 Neural network parameters for a network with 30 input neurons 68
4.10 The accuracy, mean and RMSE for tests with 600 and 900 training waveforms with 30 input neurons 69

5.1 Comparison of the MLE3 and the neural network approach with 900 trainings waveforms 74
5.2 Comparison of the algorithm approach with 900 trainings waveforms and 30 bin input data, the MLE3 and threshold retracker as well as the previous test with a neural network and 900 training waveforms 77
5.3 Comparison of the neural network, MLE3 and threshold retracker at the Lago do Madabá Grande 82
5.4 Comparison of the algorithm, MLE3 and the neural network approach at the Lago do Madabá Grande 84
5.5 Comparison of the statistical values of two neural network approaches and the MLE3 and threshold retracker 90
5.6 Comparison of the standard deviation and the mean values of the algorithm, MLE3 retracker and neural network approach regarding the DAHITI database . 91
5.7 Parameter of the neural network for transferable results 93
5.8 Statistical data of the neural network, the MLE3 and threshold retracker of the Cupari river, after training the network with data of another area 95

A.1 Starting parameters for the neural network with 10 bin window size XXIII

Chapter 1

Introduction

1.1 Introduction

Freshwater is the basic need of any form of life, which makes it one of the most important factors in the biosphere (Fekete and Vörösmarty, 2007). This gets even more obvious if one regards the fact that river basins are one of the highest populated regions (Kuo and Kao, 2011). The reasons for this are manifold and range from easy access to water, irrigation of agriculture to the transport of goods and people. However, water is not only crucial for our daily life, but it is also significant for our environment. Hence, Fekete and Vörösmarty (2007) show that the hydrological cycle is important to regulate the climate system of the earth. This hydrological cycle contains numerous components, but river discharge is the component which can be measured with one of the highest accuracies (Hagemann and Dümenil, 1997). Here, river discharge is the volume of water flowing through the river at any given point of it.

Despite the importance of surface freshwater, only limited information is available about the spatial and temporal dynamics of river discharge and the changes in the global storage of freshwater (Alsdorf et al., 2007). Alsdorf et al. (2007) mention that existing in situ networks, which measure the height and discharge of inland water bodies at so-called gauging stations, are not able to provide global knowledge regarding the volume of water stored and flowing in inland water bodies. Also, monitoring inland water bodies becomes more and more difficult, since the number of gauging stations is decreasing (Group et al., 2001). Thus, to tackle this lack of knowledge, many studies are conducted to improve the possibilities to apply satellite altimetry over inland water areas, such as (Birkett, 1995, 1998; Alsdorf et al., 2007). Though, with now 29 years of satellite altimetry, they proved that altimetry data could be used as a complementary data source (Calmant et al., 2008). As first altimetry missions only focused on the application on the ocean, they are now also providing high-quality water height measurements on inland water areas, which is more and more important (Da Silva et al., 2010; Lambin et al., 2010). As the most repeat periods of altimeter satellites are around 10, 17 or 35 days, it also enables the monitoring of interannual seasonality (Birkett and Beckley, 2010). Therefore, short term variations can be observed to estimate water height time series, the discharge or floods monitoring. This shows that satellite altimetry offers great potential to tackle the problem of declining gauging stations.

The first work on satellite altimetry already happened in 1969 (Kaula, 1969), in which long term plans were specified and also requirements, regarding the used technology, which is needed to fulfil these goals. 1972 the so-called Earth and Ocean Physics Application program

(EOPAP) report was published with the same aim as the previous one. Both focused on the development of a satellite altimeter, which reaches an accuracy of 10 cm, by using at least a spatial resolution of 1° (100 km). To achieve the EOPAP goals, the first satellite altimetry mission was flown with Skylab 1974 (McGOOGAN, 1975), followed by GEOS-3 1975 which could show the potential of altimetry missions over the ocean as well as over land areas and was then followed by SEASAT 1978 (Barrick and Swift, 1980). As these missions were focused on measure surface sea height over the ocean, they already reached good results over large inland water areas, such as lakes. Unfortunately, results over smaller water areas, like rivers, was not satisfying (Da Silva et al., 2010). This problem was solved in 1992 with the launch of Topex/Poseidon, which was widely used to measure inland water bodies. However, with Jason-1, launched in 2001, the scientists faced a setback because it showed big problems to measure inland water areas, as already mentioned in the previous chapter. This problem was then solved with the launch of Jason-2 (Seyler et al., 2013).

In this study, the Jason-2 satellite is used. According to Lambin et al. (2010), the primary objective of this mission is the generation of a climate record over several decades. Which would enable scientists to study the rise of global sea levels and improve the understanding of the relationship between ocean circulation and changing the climate. This is possible since TOPEX/-Poseidon, which was launched 1992 and the Jason-1 mission, launched 2001 have the same ground track which enables the continuous measurement over the same areas (Lambin et al., 2010). Caused by the previous bad measurement quality of Jason-1 over inland water areas (Seyler et al., 2013), several modifications were conducted to improve coverage over inland water areas. Also, previous weak systems of the Jason-1 satellite were corrected (Lambin et al., 2010). With OSTM/Jason-2 also EUMETSAT (the European Organisation for the Exploitation of Meteorological Satellites) and NOAA (the National Oceanic and Atmospheric Administration) become a partner of this project (NASA, 2017).

1.2 Motivation

Regarding the importance of the river observation to monitoring the hydrological cycle and the importance of our daily life, the declining number of gauging stations in rivers lead to difficulties in the estimation of the hydrological system. To cope with this problem, a high amount of altimetry data has to be processed to estimate the water level and river discharge. To improve these results, one can improve the measuring accuracy of the satellites, but also the post-processing of the altimetry data. Thereby, the height information of the water body is received by using a two-way time measurement with a radar pulse which is then used to compute the range of the satellite. Over inland water areas, the satellite altimeter has problems to measure the correct height which is caused by influences of the atmosphere and the surface area on the radar signal mentioned above. Thus, to receive accurate height information, the so-called retracking algorithms are applied to estimate the correct height. To compare the different retracking approaches, several studies are conducted in the past such as Tourian (2012) or Kuo and Kao (2011), and also new retracker are developed which focused on inland water bodies Troitskaya et al. (2012). However, the new methods are often time-consuming as they require additional processing steps, which makes it difficult to process a high number of data with these approaches.

1.3 Objectives

In opposite to the previous retracking approaches, a so called neural network is used in this study to estimate the correct water level, which is measured by the Jason-2 satellite. To do so, the objectives of this thesis are as follow:

- Two neural network approaches are developed.
- The developed approaches are applied to three different case studies to see the performance compared to another retracker.
- Testing the transferability of the approach by training the network over one water body and applying it to another one.

1.4 Overview of the analysed water areas

1.4.1 The Cupari river

The Cupari river is located in the Amazon river basin and flows into the Tapajos river, which is a major trinutary of the Amazon river. The location of the river can be seen in Figure 1.1.

The Cupari river is on both sides hemmed by forests with up to 30 metre high trees, which could be influence the measurement results. Interesting is the fact, that even 853 kilometres away from the river mouth, tides from the ocean are still present in the dry season. This is explained by the extreme flatness of the Amazon basin (Bates, 1863). At the crossing point of the satellite track with the water body, the river has a width of 4 km. To avoid the influence of land contamination to the measured signals, a virtual station in the middle of the river is chosen. Hence, only measurements are chosen, which cross the middle area of the river. This can be seen in Figure 1.2.

Figure 1.2 shows, that there are also several small islands located within the river. However, because of the big footprint of the radar signal, these islands have only a small influence on the measurements of the satellite altimeter. This area was chosen because it can be assumed that it creates good-natured results which can be used to test the developed algorithms.

1.4.2 Lago do Madabá Grande

The next study area is a floodplain inside the Amazon basin with the name Lago do Madabá Grande. It is chosen because it is more challenging compared to the previous study area, but at the same time inside the same basin. The exact location of the floodplain can be seen in Figure 1.3.

One can see in Figure 1.3 (a), that it consists of several flooded areas which are separated by land. The connection to the Amazon river happens due to the Nhamundá River and several channels which connect the floodplains with the main river. The satellite track, as well as the chosen area, can be seen in the next figure. As it can be seen, we chose the water body called Lago do Matapi, which is inside the floodplain as virtual station. In Figure 1.4 one can see the location of the virtual station.

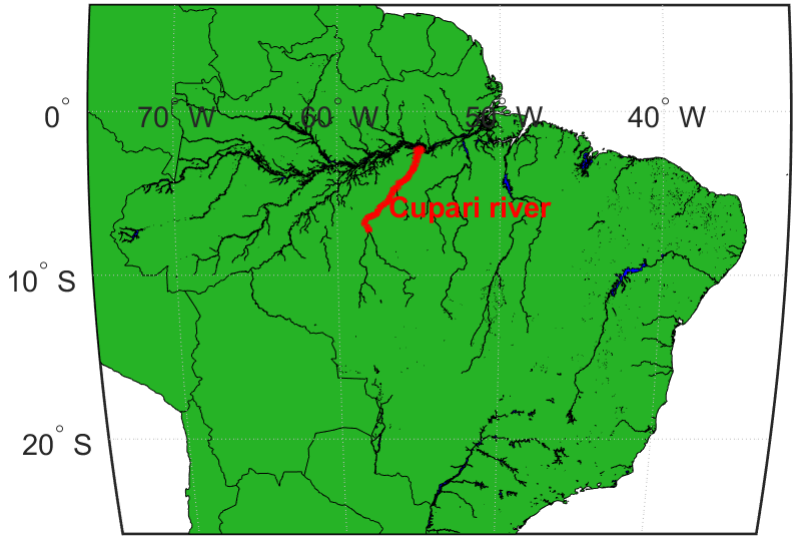


Figure 1.1: *Location of the Cuapri river (source: Greene (2019))*

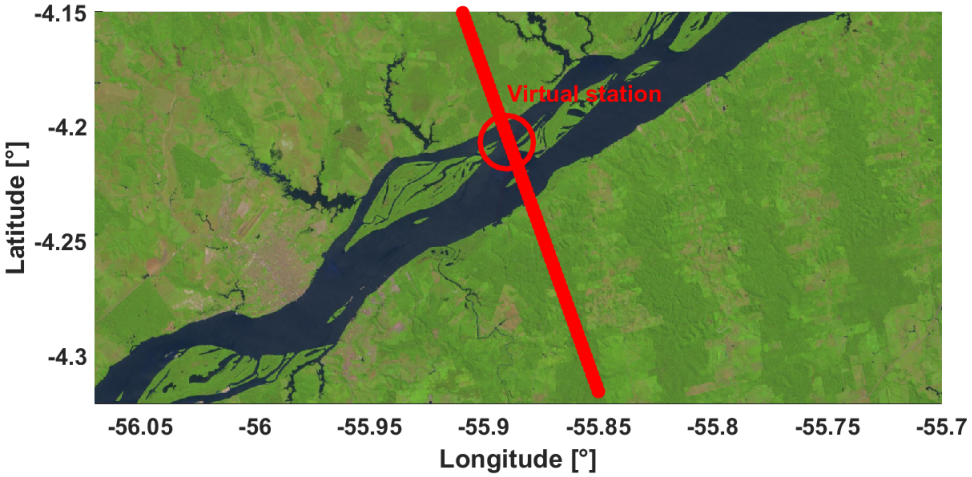


Figure 1.2: *Area in which the Jason-2 satellite crosses the Cuapri river, in red the chosen area is marked (source: https://landlook.usgs.gov/viewer.html)*

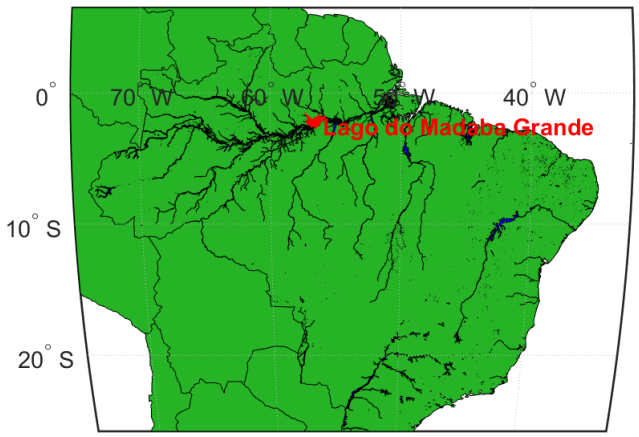


Figure 1.3: *Position of the Lago do Madabá Grande (source: Greene (2019))*

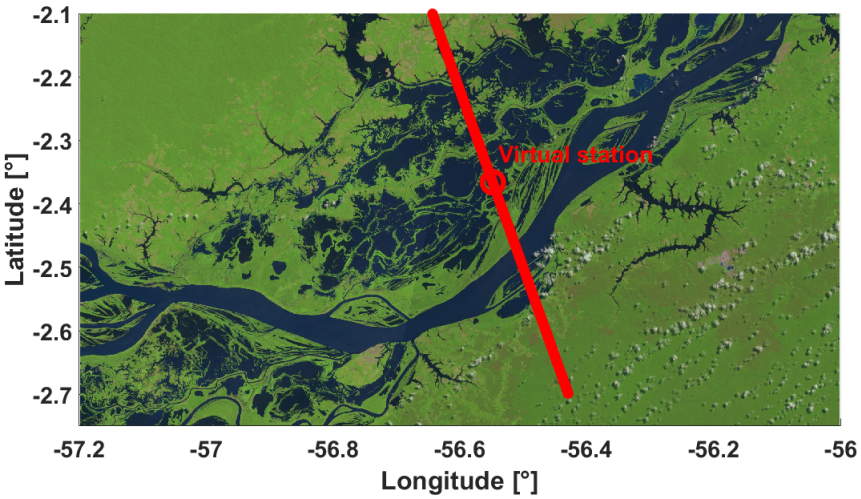


Figure 1.4: *Track of Jason-2 across Lago do Madabá Grande and in red the chosen area is marked (source: https://landlook.usgs.gov/viewer.html)*

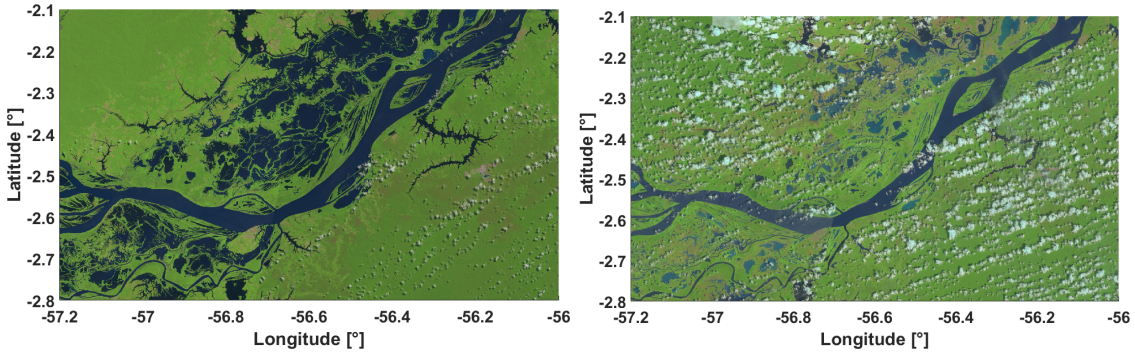


Figure 1.5: *Two images of the floodplain taken by Sentinel 2 at the 7th of July and 12th of December 2016 respectively (source: https://landlook.usgs.gov/viewer.html)*

As it can be seen in figure 1.4, is the track across the whole lake system and also over the Nhamundá river north of the floodplain. However, to see the performance in the flooded areas, the red marked area over Lago do Matapi was chosen. An analysis of Landsat Images was conducted and showed, that during the autumn and winter month the floodplain contains very less water, while the area is flooded during spring and summer month. However, water coverage varies. In Figure 1.5 one can see two examples, which show the water coverage of the study area in 2016.

The Figures 1.5 show, that a lot of water areas occur in the region during the summer month, whereas during autumn and winter the areas containing water are reduced. However, as Lago do Matapi is one of the biggest water areas inside the floodplain, it can be assumed that it contains for the most time water.

1.4.3 Lake Tana

Lake Tana is located in northern Ethiopia and is chosen as a study area because it differs a lot from the previous study areas and can show the performance of the algorithm in this area. The lake is located in the "tropical highland monsoon" region, which only sees one wet season from June to September. Whereas, between October and May the region faces a dry period. While the northern and western part is surrounded by forest, the eastern and southern regions are urban areas. Thereby, the lake is located at a height around 1788 m, which is in significant contrast regarding the previous test areas. In Figure 1.6 one can see the location of the lake.

Lake Tana has a surface area between 3000 to 3500 km^2 and a maximal depth of around 15 m. The main tributaries are Gigel Abay, Reb and Gumara and Megech but overall there are seven large rivers and 40 small seasonal rivers which flow into the lake. However, Lake Tana is the source of the Blue Nile. In Figure 1.7, one can see the track of the satellite over the lake.

Figure 1.7 shows, that the satellite crosses Lake Tana in the eastern area. To ensure that there is no land contamination to the measurements, the virtual station is chosen at the middle of satellite track.

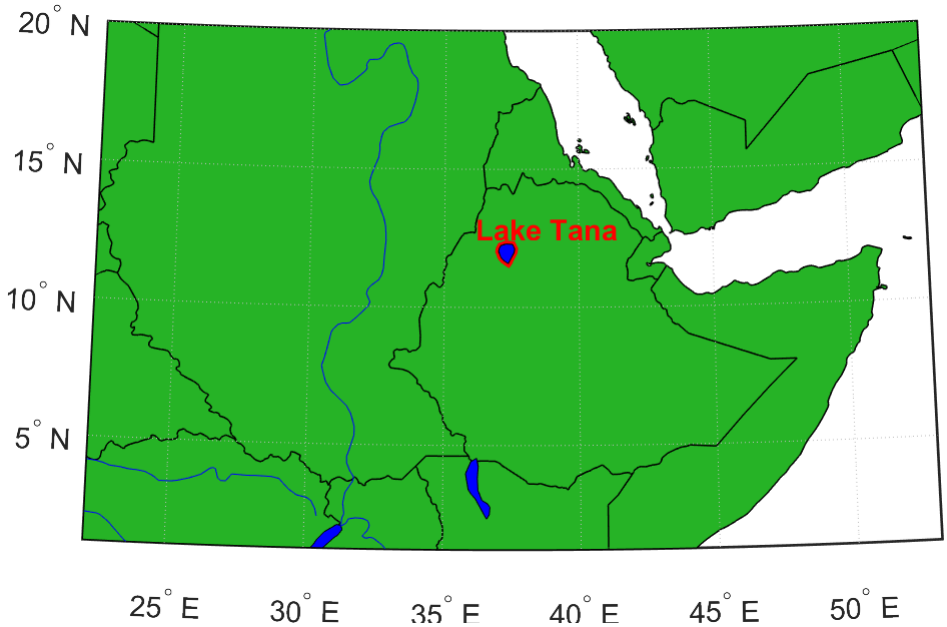


Figure 1.6: *Location of Lake Tana in Ethiopia (source: Greene (2019))*

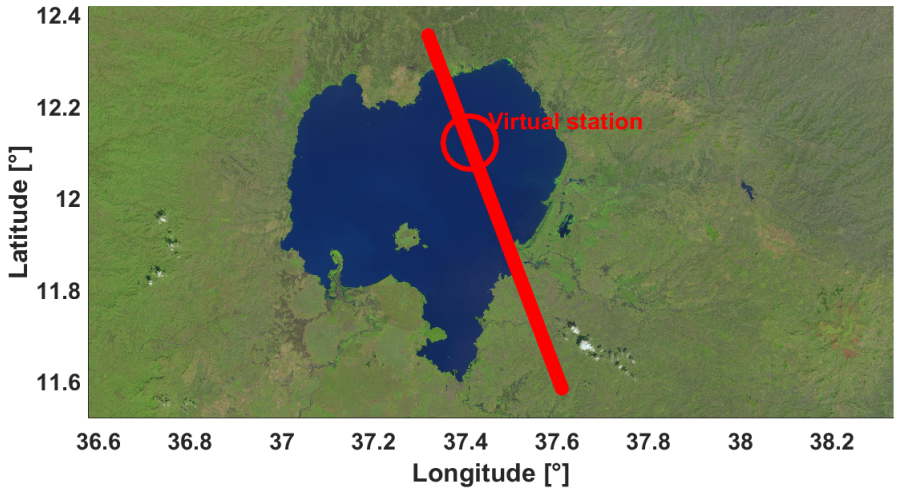


Figure 1.7: *Jason-2 track across Lake Tana and in red the choen area is marked (source: https://landlook.usgs.gov/viewer.html)*

1.4.4 Data

In this study data of the OSTM/Jason-2 satellite is used, which can be seen in Figure 1.8. The mission of the Jason-2 satellite is called Ocean Surface Topography Mission (OSTM), the reason why it is often referred to the satellite as OSTM/Jason-2 (NASA, 2017). The satellite was launched 12th July 2008 to an orbit with the following parameters:

Table 1.1: *Orbit parameter of OSTM/Jason 2*

Parameter	Value
Altitude	1339 km
Inclination	66.042°
Repeat period	9.9156 days
Equatorial cross track distance	315 km

Figure 1.8: *Jason-2 satellite in the orbit (Source: https://directory.eoportal.org/web/eoportal/satellite-missions/j/jason-2)*

The mission objective is, to measure the sea surface height, with 1 Hz measurements, with an accuracy of 3.4 cm or better (NASA, 2017). To do so, the altimeter payload of the satellite contains basically three elements Lambin et al. (2010):

- Poseidon-3, a dual frequency (Ku- and C- band) radar altimeter, which measure the distance between the satellite and the Earth surface at nadir. At the same time it gain also informations about the ionospheric delay
- Advanced Microwave Radiometer (ARM), which measures data regarding the wet tropospheric propagation delay. This causes big errors during altimeter range measurements
- Three systems for precise orbit determination, which are DORIS (Doppler Orbitography and Radiopositonning Integrated by Satellite), a GPS receiver and a Laser Reflector Array (LRA)

Thus, the range measurements are conducted by the Poseidon-3 altimeter which operates in Ku-band at 13.575 GHz and C-band at 5.3 GHz (NASA, 2017). The receiving signals are accumulated in 3.125 ns gates (Gommenginger et al., 2011). Regarding the altimetry data, the most significant change was the introduction of the so-called MLE3 retracker, additionally to the already used MLE4 retracker, which improves the quality of the coastal data (NASA, 2017). As mentioned in Thibaut et al. (2010), the Maximum Likelihood Estimator (MLE) retracking algorithm ground processing approaches are based on the Hayne model, which will be together with the MLE3, and MLE4 retracker explained in detail in 2.2.2.

In this study, the sensor geophysical data records (SGDR) data is used, which includes the so-called measured waveforms are included. These waveforms represent the received measurement signal and will be explained later. Thereby, the 20 Hz data is used, which is together with the previously mentioned 1 Hz data also provided. In the following table, the track number, as well as the time frame of the used data, is provided:

Table 1.2: *Used tracks and time frame for the current study*

Study area	track number	time frame
Cupari river	228	8.2009 - 06.2016
Lago do Madabá Grande	228	8.2009 - 09.2016
Lake Tana	94	05.2008 - 09.2016

1.5 Outline of the thesis

The basics of satellite altimetry and so-called waveform retracking algorithms are discussed in the second chapter. The theory of the neural networks are explained in the third chapter and also the developed approaches are shown in detail. In the fourth chapter, the algorithms are applied to the three study areas, and the results are discussed. Finally, the conclusions and outlook for further studies are presented in chapter six.

Chapter 2

Introduction in satellite altimetry

2.1 Satellite altimetry

2.1.1 Basics of satellite altimetry

Satellite altimetry missions provide continuous observations since 1992. These measurements can be used to see the long term variations of the oceans and inland water basins. The principle of Satellite altimetry is based on run time measurement, to estimate the height. Hence, the satellite is using a nadir pointing antenna to transmit a radar pulse towards the Earth surface. This signal is moving downwards and is reflected by the Earth surface and heading back to the satellite. Because of using a so-called pulse limited radar, only the earliest returns of the signal are recorded by the satellite. The probability density function (PDF) of reflected signals are resulting in a tractable form which can be averaged to a waveform (Quartly et al., 2001). The creation of such a waveform can be seen in Figure 2.1.

At the beginning of the transmission ($0 < t < t_0$), the altimeter antenna is emitting a pulse limited electromagnetic signal which propagates in a spherical wavefront. Reaching point $t = t_0$, the apex encounters with the Earth surface which is directly beneath the satellite. It illuminates this point, and the signal is reflected back to the altimeter, thereby it is creating the so-called leading edge of the waveform. After this first encounter, the wavefront descends more downwards, and the area of encounter propagate from the nadir point away while forming a ring. At point $t = t_1$, the rear of the pulse reaches the Earth surface, and the ring is changing to a ring. This is also the point at which the reflected energy is reaching the maximum. With $t > t_1$, the energy of reflected signals is declining which creates the so-called trailing edge (Guo et al., 2009; Quartly et al., 2001). In Figure 2.2, the three parts of a waveform can be seen.

The difficulty for each altimeter system is now, to keep the waveform inside the predefined time window (Birkett, 1998). In the ideal case, the altimeter also fixes the half point of the leading edge at a predefined position inside this window, at the so-called tracking gate. The location of that point is then tracked and gives, regarding the transmitted pulse, the information about the measured range (R) (Marth et al., 1993). By using the precise altitude of the satellite (S) over the reference ellipsoid, it is then possible to calculate the sea surface height (SSH):

$$SSH = S - R + \text{corr} \tag{2.1}$$

For a better understanding one can take a look at Figure 2.3.

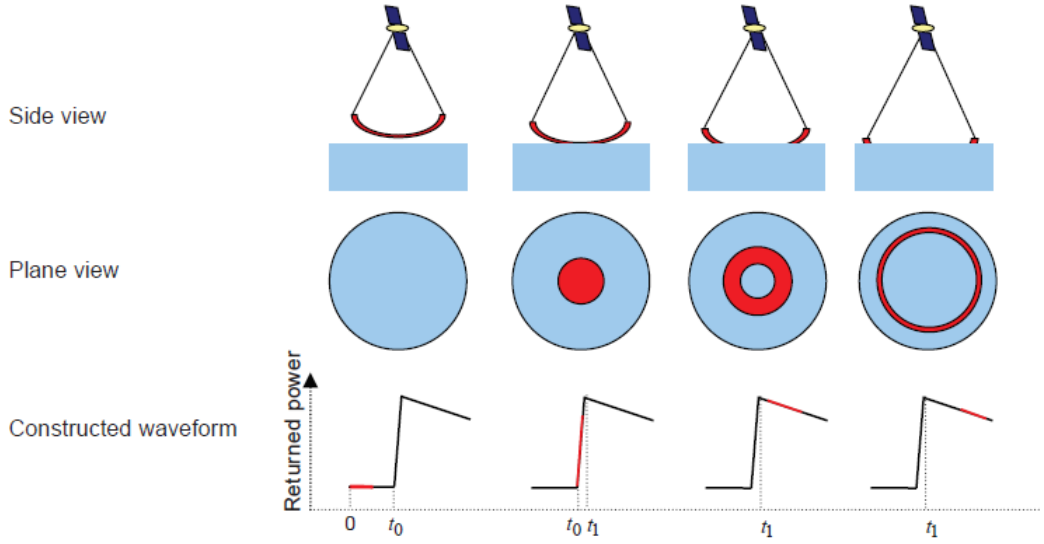


Figure 2.1: *Creation of the waveform (Tourian, 2012)*

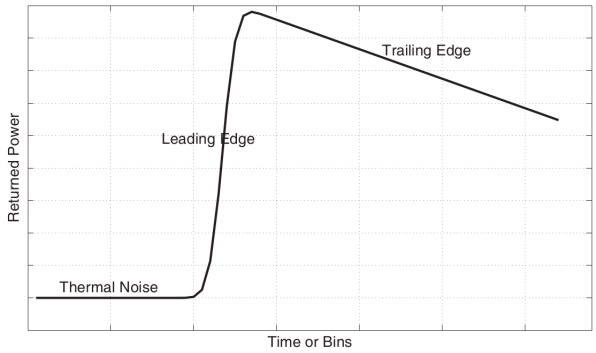


Figure 2.2: *Schematic graphic of the parts of a waveform over ocean surface (Tourian, 2012)*

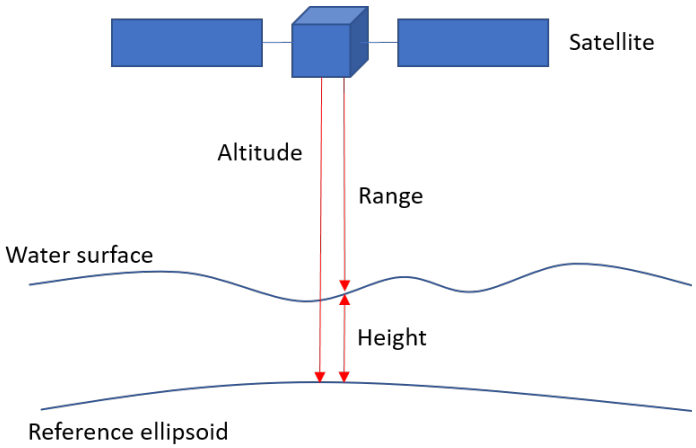


Figure 2.3: *Basic principle of the satellite altimetry*

The corrections (corr) in Equation (2.1) are used to cope with propagation delays of the radar signal and other variations caused by the Earth atmosphere and reflecting surface. In section 2.1.2 the parameters to correct the measurements are explained in more detail. To receive water level within a terrestrial reference frame, the orbital trajectory of the satellite has to be measured with GPS, DORIS (Doppler Orbitography and Radiopositioning Integrated by Satellite) or also satellite laser ranging (SLR) (Luthcke et al., 2003). With this, the correct position of the satellite is known within a reference frame, and one can estimate the water level according to it. To reach this goal, Jason-2 is flying at an altitude of 1336 km, which lead to better orbit determination because atmospheric drag is reduced. Also, the influence of the Earth gravitational field and tides are reduced, which lead to a reduction of orbit perturbations (Lambin et al., 2010). With this, it is now possible to determinate the orbit precisely which lead to an accurate determination of the water level. However, the main difficulty over inland altimetry is, that there are fast changes in the elevation profile, which do not occur over the ocean. With these fast changes, the altimeter can lose lock. Thus it is not able to keep the waveform inside the window. This problem occurred for example at the Jason-1 altimeter, so that there were nearly no useful measurements over inland water bodies (Da Silva et al., 2010).

If one applies satellite altimetry over inland water bodies, several problems will be faced:

- The shape of the waveforms over inland water bodies is highly variable and differs from the consistent waveforms, received over the ocean (Calmant et al., 2008). The reason is that the radar signal is now not only reflected by water surface, but also by the surrounding terrain which creates noise (Gommenginger et al., 2011). Hence, the onboard retracker are only designed to retrack the waveforms received over ocean measurements (Smith, 1997). Thus, the estimated range which is based on the received radar signal can be erroneous. In the worst case, the altimeter may lose tracking thus that the subsequent echoes are lost. However, this can be solved by using several altimeters and tuned filters which lead to high-quality data (Calmant et al., 2008).

- The off-nadir reflections, notably the hooking effect creates errors. This effect occurs if a water area at the edge of the ground footprint reflects more energy to the antenna than the area at nadir of the antenna. Thus, the stronger signal travelled a longer skewed path which leads to an error in the range estimation by the satellite. This happens because the satellite assumes that the target is at nadir below the satellite. Because of that, it has to be taken into account during the analysis of the data, as it is not uncommon in continental waters (Calmant et al., 2008).

- Also, the slope of the river, which wants to be measured, affect the measurement of the height. As the footprint of the radar signal is several kilometres wide, the height of the river can change significantly. As before, also this effect changes the shape of the waveforms and influences the range determination (Calmant et al., 2008).

- Another problem Calmant et al. (2008) stated is the air density, thus the amount of water vapour and electrons in the ionosphere. They influence the travel time of the radar signals through the atmosphere. Whereas the electron content in the ionosphere and the air pressure are obtained by independent data sets, the water vapour is estimated by using microwave radiometers onboard the satellite. However, over continents, the signature of atmospheric water vapour is mixed with that of the ground wetness. Because of that, the current microwave radiometers fail to estimate the amount of water vapour correctly

which lead to errors between some centimetres to tens of centimetres. The corrections can therefore only be estimated by using large scale global datasets, which can also suffer from large errors (Calmant et al., 2008).

2.1.2 Applied corrections

The corrections which are applied to the measurements can be summarised into three different groups.

- The first group are geophysical corrections, which are tides of the ocean, solid Earth and the poles as well as tidal loading. One can see that this group of effects change the height of the surface itself so that the resulting measurement does not reflect the true height of the surface and therefore have to be corrected.

- The second group are called propagation corrections, which include the ionosphere as well as the wet and dry troposphere. Thus, these effects occur as the signal is propagating through the atmosphere which causes delays.

- In the last group are surface corrections summarised, which are the inverse barometer effect and also electromagnetic bias. These two effects appear relatively close to the surface and again interact with the radar signal.

Figure 2.4 shows the different corrections, which are applied to the measurements.

Inverse barometer

This effect describes the static response of the oceans to the atmospheric pressure. Thus, if the atmospheric pressure is increasing and decreasing, the surface water bodies fall and rise respectively during the hydrostatic response. It is also to mention that a change in the dry tropospheric corrections affects the inverse barometric corrections because the surface atmospheric pressure corresponds to the dry tropospheric corrections. Hence, if the atmospheric pressure decrease about 1 mbar, the surface of the sea is depressed by about 1 cm.(NASA, 2017). Thus, the immediate ionospheric corrections can be computed by using the surface atmospheric pressure (P_{atm}):

$$\Delta IB = -9.948 \cdot (P_{\text{atm}} - P) \tag{2.2}$$

where P is the time-varying mean of the global surface atmospheric pressure over the ocean. Whereas the scale factor 9.948 is based on the empirical value of the inverse barometric response at mid-latitude. To determine the atmospheric pressure, the European Center for Medium Range Weather Forecasting (ECMWF) numerical weather prediction model is used for Jason-2. Thereby, the uncertainty of the ECMWF atmospheric pressure products depends on the location (NASA, 2017).

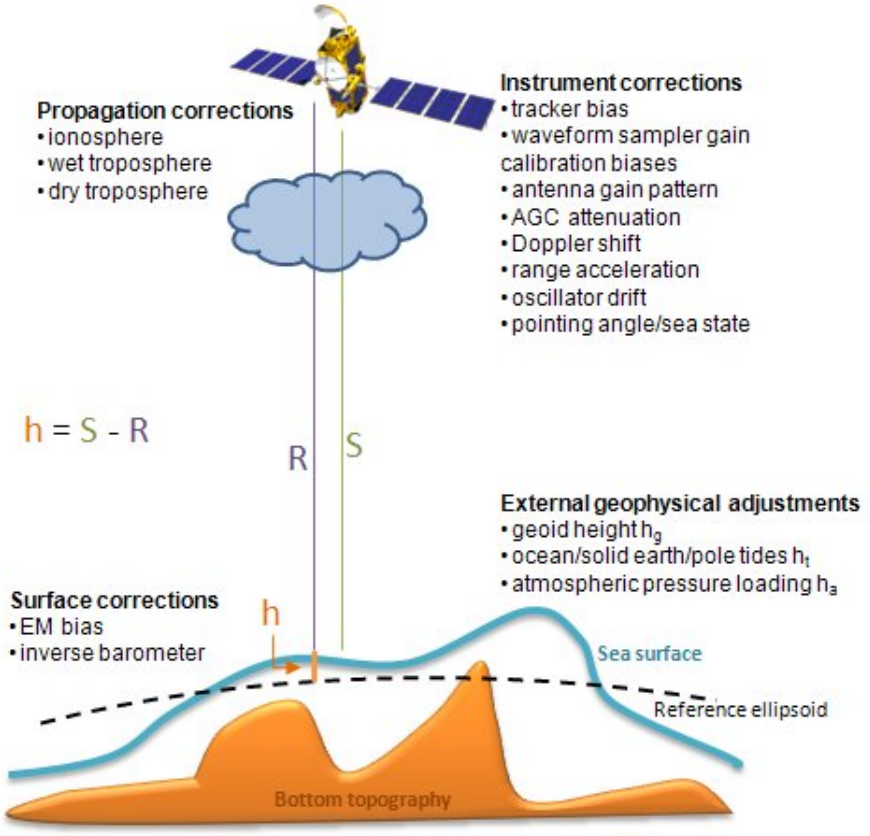


Figure 2.4: *Schematic graph of needed corrections in the radar altimetry (www.altimetry.info)*

Sea state bias

The sea state bias is an error in the satellite ranging measurement which is caused by ocean waves and tides on the surface. This correction adjusts the measured range at the cm level to improve the satellite radar estimation of the mean sea level (Tran et al., 2010). The sea state bias contains the electromagnetic bias, the skewness bias and the tracker bias. Because the troughs of the waves tend to reflect the radar signal better than the crests, the centre of the mean reflecting surface is shifted away from the mean sea level to the troughs of the waves. This leads to an overestimation of the range measurement (NASA, 2017). According to Rodríguez et al. (1992), the electromagnetic bias is between 1 % and 4 % of the surface water height. The median reflecting surface is lower than the mean sea level, which is caused by the skewness in the sea height distribution over the satellite footprint. This is caused by the stronger backscattering of the wave troughs over the crests (Scharroo and Lillibridge, 2005). Whereas the inherent instrumental and measurement errors cause the tracker bias. These are associated with the range estimation as well as the wave height and backscatter from the returned signal (Fu and Cazenave, 2000).

If satellite altimetry is applied over inland water bodies, the sea state bias is normally set as approximately zero because the average waves are cancelled out along track. Thus, over smaller lakes or narrow rivers, the correction is negligible. However, in case of great lakes, the application of the sea state bias corrections can slightly reduce the lake level height anomaly variance at crossovers (Zhao, 2018).

Ionosphere

If the radar signals of the altimetry measurement propagate through the ionosphere, they are affected by the free ions and electrons in the ionosphere. These particles are in an area from about 50 to roughly 2000 km above the earth's surface in such a high abundance that they slow down the radar pulse (Rush, 1986). Thereby, the retardation of the velocity of the radar pulse is inversely proportional to the frequency squared (NASA, 2017). The ionospheric correction can be derived from the integral over the height of the ionospheric refractivity $N_{ion}(z)$, which is proportional to the density $n_e(z)$, with this one receive the following formula:

$$\begin{aligned} \Delta Ion(f) &= 10^{-6} \int_0^R N_{ion}(z)dz \\ &= \frac{40.3 \cdot 10^6}{f^2} \int_0^R n_e(z)dz \end{aligned} \tag{2.3}$$

where n_e has the unit electrons$/m^3$, f is in Hz and ΔIon is in metres. The second integral in Equation (2.3) represents the total electron content (TEC), which is the integrated electron density along the signal path with the unit TECU (TEC Units) (Marković, 2014) with 1 TECU = 10^{16}electrons$/\mathrm{m}^2$. With this, one can write the ionospheric path delay as:

$$\Delta Ion(f) = \frac{k\mathrm{TEC}_{\mathrm{alt}}}{f^2} \tag{2.4}$$

where $k = 0.4025\mathrm{m} \cdot \mathrm{GHz}^2 \cdot \mathrm{TECU}^{-1}$, f is expressed in GHz and $\mathrm{TEC}_{\mathrm{alt}}$ is the total electron content below the satellite altimeter. In case of the Ku-band with 13.6 GHz, 1 TECU would result in 2.18 mm path delay (Schreiner et al., 1997). Because the ionospheric delay depends on the frequency, it can be measured by using dual-frequency altimeter e.g. Jason-1, Jason-2 and Envisat. Thereby, $\mathrm{TEC}_{\mathrm{alt}}$ can be calculated as

$$\mathrm{TEC}_{\mathrm{alt}} = \frac{f_{\mathrm{Ku}^2 f_{\mathrm{C}^2}}}{f_{\mathrm{Ku}}^2 - f_{\mathrm{C}}^2} \frac{R_{\mathrm{C}} - R_{\mathrm{Ku}}}{k} \tag{2.5}$$

where R_{Ku} and R_{C} are the ranges measured in the Ku-band and C-band respectively. The C-band has a frequency of 5.3 GHz for TOPEX and Jason-1/2 satellites. Thus, by using the range measurements it is now possible to estimate the ionospheric influence on the radar signal propagation in the ionosphere. Over the ocean, the Ku-band ionospheric range correction can each an accuracy of 0.5 cm (NASA, 2017). However, it is also possible to calculate the corrections by using dual frequency radar signals as GPS and DORIS which are provided in models like the Global Ionosphere Maps (GIM). This model is provided for Jason-2 asa backup ionospheric correction (NASA, 2017).

Dry troposphere

The dry gases in the Earth troposphere slow down the velocity of the radar pulse. The resulting height errors are approximate -2.3 m, and therefore they are responsible for the most significant range errors. The dry tropospheric corrections (ΔDTC) are equal to the surface pressure and multiplied by -0.0022768 m/mbar and with a small adjustment to reflect the small dependency on the latitude(NASA, 2017) and can be computed using the modified Saastamoinen model (Davis et al., 1985):

$$\Delta DTC = -\frac{0.0022768 \cdot p_s}{1 - 0.00266cos2\phi - 0.28 \cdot 10^{-6} h_s} \tag{2.6}$$

where p_s is the surface pressure in hPa, ϕ is the latitude, h_s is the surface height above the geoid and ΔDTC is in metres. The ECMWF numerical weather prediction model determines the surface pressure. The error of this model is depending on the location and vary from 1 mbar at the northern Atlantic Ocean to a few mbars at the Pacific Ocean. where 1 mbar error in the atmospheric pressure results in a 2.3 mm error in the dry tropospheric correction (NASA, 2017).

Wet troposphere

Beside the dry gasses in the atmosphere, also the water vapour is delaying the radar signal propagation. However, in opposition to the previous correction, the height dependence of the water vapour is difficult to model as it is high time dependence (Fernandes et al., 2015). Also

Fernandes et al. (2013) state, that in order to receive a sub-cm accuracy the pressure has to be known better than 5 hPa. Such pressure accuracy can be obtained by using microwave radiometers (MWR), which are onboard the altimetry missions. These instruments measure the radiation from the earth at two or three frequencies in which the absorption by water vapour and liquid water occurs. Thereby, measurements of the atmospheric brightness temperature close to the water vapour line at 22.2356 GHz and removal of the background is needed. However, it is to mention, that the computed corrections of the algorithms are only valid over homogenous water surfaces, as they are based on sea surface emissivity models (Desportes et al., 2007). Thus, if one wants to apply these corrections over inland water bodies, other options should be used.

To estimate the wet tropospheric correction without relying on those measurements, Kouba (2008) proposed the following empirical expression:

$$\Delta WTC(h_s) = \Delta WTC(h_0) e^{-(\frac{h_0 - h_s}{2000})} \tag{2.7}$$

where h_s and h_0 are the ellipsoidal heights of the measurement station and the sea surface respectively. Thus, height errors of 1 cm and 5.6 cm will result in a correction of 20 cm. In Jason-2 products, an improved near land wet path delay algorithm is applied, which improves the performance. In case of sun glint, land contamination, or anomalous sensor behaviour, the measurements of the MWR are unusable. For these situations, the ECMWF numerical weather prediction model also provides values for the wet tropospheric corrections (NASA, 2017).

Pole tide

The variations in the geocentric location of the pole of the Earth's instantaneous rotation axis, or polar motion, is introducing a differential centrifugal force which causes displacements of the solid Earth and oceans. These displacements are referred to as pole tides (Munk and MacDonald, 1961). The reason for this naming is that they can be treated as similar as the luni-solar tides. Regarding the centrifugal potential, it is temporarily determined by periodic alternations in polar motions, primarily at the Chandler wobble with seasonal periods of around 14 and 12 months (Desai et al., 2015). Thereby, the displacement of the solid Earth, thus the body pole tide, have amplitudes up to 10 mm (Wahr, 1985), which depends on the location and time-varying amplitude of the Chandler wobble. Desai (2002) shows, that the displacements of the ocean surface with respect to the ocean bottom, hence the ocean pole tide, have a similar amplitude. Because the satellite radar altimeters observe the geocentric sea surface height (SSH), it contains the total sum of the body, ocean, and pole tides which leads to amplitudes up to 20 mm (Desai et al., 2015).

In order to compute the pole tide displacements, one can apply tidal Love numbers to the differential centrifugal potential, whereas the potential is derived from polar motion observations. Wahr (1985) describe the computation of the pole tides ΔPT as:

$$\Delta PT = A \cdot sin2\phi \cdot [(x - x_{\text{avg}}) \cdot cos\lambda - (y - y_{\text{avg}}) \cdot sin\lambda] \tag{2.8}$$

where ΔPT is expressed in mm, A is the scaled amplitude factor, which is calculated by the Love number in metre. λ and ϕ are the longitude and latitude of the measurement, whereas

x and y are the nearest previous pole location data which is relative to the altimeter time and x_{avg} and y_{avg} are the averaged pole coordinates. The modelling requires also a time series of perturbations to the Earth rotation axis, which is now measured with space techniques. In case of Jason-2, the pole tides are computed with a more precise time series of the Earth rotation axis (NASA, 2017).

Ocean tides

It refers to the rise and fall of the sea, which is caused by the gravitational attraction of the sun and moon. Thus, the tidal variations of the surface represent even more than 80% of the ocean surface variability (Fu and Cazenave, 2000). Therefore, ocean tides have to be removed from the altimeter signal as it requires undisturbed water levels. To compute the ocean tide ΔOT, the products of TOPEX/Poseidon are based on the GOT99.2 model, which can be written as the sum of N tidal constituents h_i (D. Ray, 1999):

$$\Delta OT = \sum_{i=1}^{N} F_i(t) \cdot [A_i(\phi, \lambda) \cdot cos\psi_i + B_i(\phi, \lambda) \cdot sin\psi_i] \tag{2.9}$$

with

$$\psi_i = \sigma_i \cdot t + X_i(t) + U_i(t) \tag{2.10}$$

where F_i is the tidal coefficient of the amplitude nodal correction, X_i is the tidal astronomical value. U_i is the nodal correction of the tidal phase. t is the altimeter time, and ϕ and λ are the latitude and longitude respectively. $A_i(\phi, \lambda)$ and $B_i(\phi, \lambda)$ are harmonic coefficients which are at the altimeter location (ϕ, λ) bilinearly interpolated, with the input harmonic coefficient map which is given by the GOT99.2 model by D. Ray (1999). Whereas σ_i is the tidal frequency. The standard deviations of the tidal variations in the open ocean are between 10 and 60 cm with even larger values near the coastal regions. However, by using satellite altimetry missions, it is now possible to estimates tides with an accuracy of 2-3 cm (Fu and Cazenave, 2000).

Earth tides

Similar to the ocean tides, also the Earth tides are caused by responses of the solid Earth to the gravitational forces of the sun and moon. Thereby, the response of the Earth to these gravitational forces is even fast enough, that it can be considered to be in balance with the gravitational forces. In that case, the surface is parallel with the equipotential surface, and the height of the tide is proportional to the potential (NASA, 2017). The two proportionality constants are the Love numbers, which are mainly frequency independent (Wahr, 1985).

For Jason-2, the solid Earth tides are computed as a purely radial elastic response of the solid Earth to the tidal potential. Thereby, the adopted tidal potential is the Cartwright and Tayler (1971) and Cartwright and Edden (1973) tidal potential which is extrapolated to the 2000 era and includes coefficients of degree two and three of the tidal potential (NASA, 2017).

2.2 Overview of the common retracking algorithms

The onboard retracker tries to adjust the midpoint of the leading edge at the predefined so-called nominal tracking gate. As an example, for the 60 Topex/Poseidon gates, the tracking gate is located at bin 29.5 and for the 104 sample waveform of Jason-2 the nominal tracking gate is at position 31.0 (Gommenginger et al., 2011). In Figure 2.5 one can see the basic retracking principle.

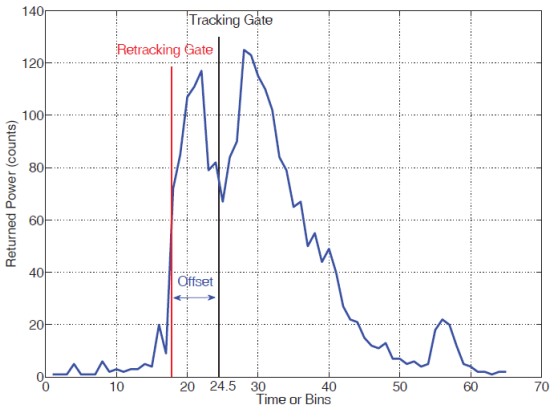


Figure 2.5: *Principle of retracking with the nominal gate (tracking gate) and the retracked gate in red (Tourian, 2012)*

Based on Figure 2.5 the following formula can be derived:

$$\Delta R_{\text{ret}} = (\text{bin}_{\text{ret}} - \text{bin}_{\text{nom}}) \cdot \tau \cdot \frac{c}{2} \tag{2.11}$$

where ΔR_{ret} is the retracked range, bin_{ret} is the bin of the retrackeding gate, bin_{nom} is the bin of the nominal tracking gate. τ is the pulse width, which for Jason-2 is 3.125 ns. With this corrected range Equation (2.1) changes to:

$$SSH_{\text{corr}} = S - (R + \Delta R_{\text{ret}}) + \text{corr} \tag{2.12}$$

With this Equation (2.12), one can finally receive the corrected sea surface height. It can be seen, that the accuracy of estimated ranges depends on the retracking algorithm which is used. In the past, several different retracking algorithms are developed which all focus on different types of waveforms. To provide an overview, several common retracker are discussed in the following.

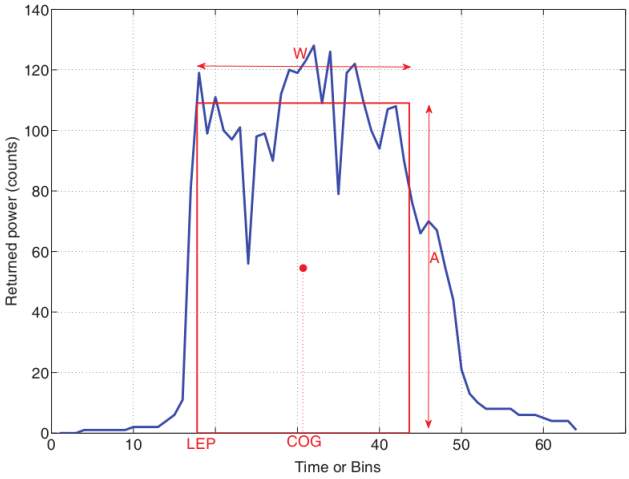


Figure 2.6: *Schematic diagram of the OCOG retracker (Tourian, 2012)*

2.2.1 Empirical waveform retracking

Offset Center of Gravity (OCOG) retracker

Wingham et al. (1986) developed a robust retracking algorithm which is based on a statistical approach and is called the offset centre of gravity (OCOG) retracking. As mentioned, the shape of the waveforms is varying a lot over inland water areas. Thus, the OCOG algorithm is estimating the shape of waveforms in a first step. Thereby it uses the power level of the gates to find the centre of gravity *COG* of the waveform. Hence, a rectangular box is calculated which is defined by the amplitude A and the width W of the waveform which is used to determine the retracking gate. This can be seen in figure 2.6.

One can use the following formulas to calculate the parameters of the rectangular box which is shown in Figure 2.6:

$$COG = \frac{\sum_{i=1+n_1}^{N-n_2} iP_i^2(t)}{\sum_{i=1+n_1}^{N-n_2} P_i^2(t)}, \tag{2.13}$$

$$A = \sqrt{\frac{\sum_{i=1+n_1}^{N-n_2} P_i^4(t)}{\sum_{i=1+n_1}^{N-n_2}, P_i^2(t)}} \tag{2.14}$$

$$W = \frac{\left(\sum_{i=1+n_1}^{N-n_2} P_i^2(t)\right)^2}{\sum_{i=1+n_1}^{N-n_2} P_i^4(t)}. \tag{2.15}$$

P_i is the power of the waveform at the ith bin, N is the total number of samples and $n_1 = n_2 = 4$ are the numbers of pins which are affected by the aliasing at the beginning and the end of the waveform. After calculating this, the leading edge position (LEP) can be calculated with the following formula:

$$LEP = COG - \frac{W}{2} \tag{2.16}$$

The OCOG is easy to implement, but it also faces drawbacks. If there are surface undulations and a leading edge with a small slope, it leads to a waveform which cannot be retracked correctly by the OCOG (Deng, 2003). Another problem is that in case of a low signal to noise ratio (SNR) notable noise energy is located on the left side of the first arrived return. In such a case, OCOG cannot distinguish between this noise and the signal, and if the noise level is strong enough, OCOG is estimating the leading edge on the left side of the window. This leads to the effect, that the retracker 'run away' with the noise (Wingham et al., 1986).

Threshold retracker

The threshold retracker was developed by Davis (1997) with the primary goal to measure ice-sheet elevations. The main benefits of this algorithm are the smooth implementation and that it shows good results in respect of repeatability (Davis, 1995). In this case, repeatability describes the consistency of the retracker in selecting a retracking point Davis (1997). Thereby, 10%, 20% and 50% of the maximum amplitude of the waveform as thresholds are chosen. Thereby the 10% threshold showed the highest repeatability and the 20% threshold is useful in case of surface and volume scattering by measuring over ice sheets. The 50% threshold is only recommended if there is a dominating surface scattering in the waveform (Davis, 1997). To receive the retracking gate, a linear interpolation is conducted between the neighbouring bins at the position in which the threshold value crosses the leading edge. By defining WD_n as a data sample of waveforms with n as the number of the range gate, the maximum amplitude of the waveform array $A_{\max}$ can be calculated with the formula below:

$$A_{\max} = \max(WD_n) \tag{2.17}$$

In the next step, the DC level in front of the leading edge can be calculated by using the first five waveform samples which are not aliased:

$$DC = \frac{1}{5} \sum_{n=\tilde{n}}^{\tilde{n}+4} WD_n \tag{2.18}$$

where $\tilde{n}$ is the location of the first waveform sample which is not aliased. After that one can get the threshold level by using the next formula:

$$TL = DC + T_{\text{coeff}}(A_{\max} - DC) \tag{2.19}$$

with T_{coeff} as the percentage of the maximum waveform amplitude which is above the DC level. Next, the location on the leading edge bin_{ret} can be calculated:

$$bin_{\text{ret}} = (\hat{n} - 1) + \frac{TL - WD_{\hat{n}-1}}{WD_{\hat{n}} - WD_{\hat{n}-1}} \tag{2.20}$$

in which $\hat{n}$ is the first bin which exceeds the defined threshold level. Of course, it is to mention that in case of $WD_{\hat{n}} = WD_{\hat{n}-1}$, Equation (2.20) is not defined. In such a case the term

$$bin_{\text{ret}} = \hat{n} - 1 \tag{2.21}$$

can be used. To receive now the correct range, the next formula can be used:

$$\Delta R_{\text{ret}} = (bin_{\text{ret}} - bin_{\text{nom}}) \cdot \tau \cdot \frac{c}{2} \tag{2.22}$$

In the following sketch the mentioned parameter can be seen more clearly:

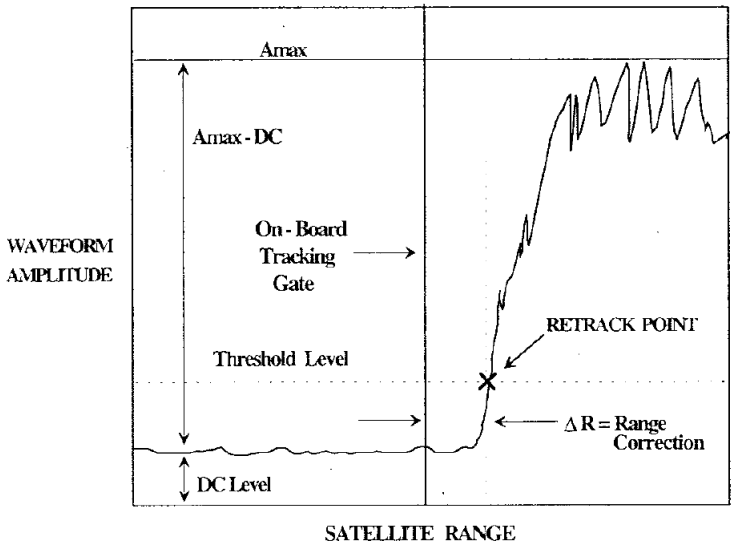
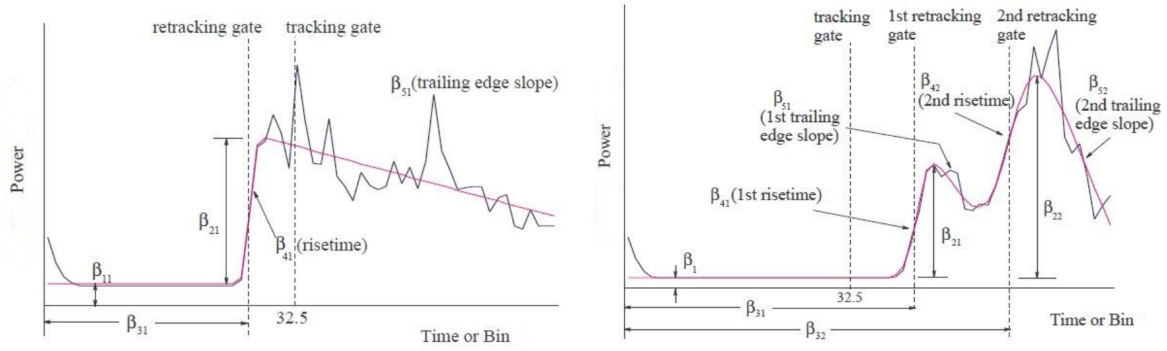


Figure 2.7: *Threshold retracker with a typical waveform from ice-sheet altimeter measurements. It is to mention that the threshold is referenced to a defined percentage of the maximum waveform amplitude above the DC level in front of the leading edge (Davis, 1997)*

This retracker performs well if one is interested in repeatable results (Davis, 1997). However, the problem is that depending on the threshold, one receive different heights. Thus, differences of 0.5 - 4.5 m are typical between the 10 and 50% threshold levels (Davis, 1997). Therefore, this retracker is only limited useful if one wants to measure the accurate height.



(a) (b)

Figure 2.8: *Sketches of the 5β and 9β fitting algorithm which are applied to single-ramp and double-ramp waveforms respectively (Roohi, 2015)*

β-parameter retracker

Martin et al. (1983) developed the first retracker focusing on continental ice sheets by fitting a function of multi-parameters to the measured waveform. The algorithm can be used with 5 or 9 parameters which it then fit to the waveform. Thereby, one receive geophysical parameters from the fitted function. The 5β parameter function is applied to so-called single-ramp returns, which are waveforms with one leading edge and can be seen in Figure 2.8 (a). However, the 9β parameter algorithm is focusing on so-called double-ramp returns which can be seen in Figure 2.8 (b). These returning signals generated during the penetration of radar signals through the surface. In this case, two signals are received, one from the reflection at the surface and the other from the reflection at the inner surface. Resulting in the creation of two smaller waveforms. In the case of the 5β parameter algorithm, the common formulas are given as (Martin et al., 1983):

$$y(t) = \beta_1 + \beta_2(1 + \beta_5 Q)P\left(\frac{t - \beta_3}{\beta_4}\right), \tag{2.23}$$

in which

$$Q = \begin{cases} t - (\beta_3 + 0.5\beta_4), & for\ t \geq \beta_3 + 0.5\beta_4 \\ 0, & for\ t < \beta_3 + 0.5\beta_4 \end{cases}, \tag{2.24}$$

$$P(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} \exp\left(\frac{-q^2}{2}\right) \mathrm{d}q \tag{2.25}$$

Thereby, the unknown parameter can be described as follow:

- β_1: thermal noise level of the waveform
- β_2: returned signal amplitude of the waveform
- β_3: position of the mid point on the leading edge, which is then used as the tracking gate

- β_4: rise time of the waveform
- β_5: slope of the trailing edge of the waveform

It is also possible to replace the trailing edge with an exponential decay term. This can be used to fit the function onto a waveform with fast decaying trailing edge (Zwally, 2017). In case of the 9 parameter function with an exponential trailing edge, one can write the formulas by using (Deng and Featherstone, 2006)

$$y(t) = \beta_1 + \sum_{i=1}^{2} \beta_{2i}(1 + \beta_{5i})Q_i)P\left(\frac{t - \beta_{3i}}{\beta_{4i}}\right), \tag{2.26}$$

with:

$$Q = \begin{cases} t - (\beta_{3i} + 0.5\beta_{4i}), & for\ t \geq \beta_{3i} - 2\beta_{4i} \\ 0, & for\ t < \beta_{3i} - 2\beta_{4i} \end{cases}, \tag{2.27}$$

$$P(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} \exp\left(\frac{-q^2}{2}\right) \mathrm{d}q \tag{2.28}$$

With these formulas, it is now possible to estimate the β parameters by using the least squares algorithm. To do so, parameters are fitted to the waveform. However, because of equation 2.26, one has to linearise the term. This can be done by using the linear least squares parametric adjustment iteratively to estimate the wanted parameters. The initial values can then be calculated by using the already described OCOG retracker. After the parameters are received, it is then possible to correct the range measurement by calculating the difference between the estimated position of the midpoint of the leading edge (β_3 parameter) and the nominal onboard retracker gate, as explained previous, which lead to a formula similar to equation 2.11:

$$\Delta R_{\mathrm{ret}} = (\beta_3 - bin_{\mathrm{nom}}) \cdot \tau \cdot \frac{\mathrm{c}}{2} \tag{2.29}$$

in which τ is the sampling rate of the corresponding bin of β_3, while c is the speed of light. As Deng and Featherstone (2006) pointed out, there is no relation between the estimated β parameters and the physical properties. However, Tourian (2012) showed, that the 5β retracker has good results in the case of so-called quasi Brown waveforms. But if the waveforms are for example specular, this algorithm fails to fit the function on the waveform. Thus, the performance of this algorithm depends on the shape of the waveforms.

2.2.2 Physically based waveform retracking

Brown-Hayne theoretical model

The previous retracking algorithms used either statistical properties of the received waveforms or they fit an empirical function to it (Gommenginger et al., 2011). But it could be seen, that these models face problems if the actual waveform does not fit into the empirical model. The retracking algorithms explained in this chapter are using now the knowledge of scattering

microwave signals at nadir to retrack the signals (Gommenginger et al., 2011).

Barrick and Lipa (1985) showed, that it is possible to express the average returned power from a rough scattering surface, which is a function of time delay t, as a convolution as followed

$$\mathrm{W(t)} = \mathrm{FSSR(t)} * \mathrm{PTR(t)} * \mathrm{PDF(t)} \tag{2.30}$$

where FSSR is the response of the flat sea, PTR is the response of the radar target point and PDF is the probability density function at specular points of the ocean surface. Thereby, the PTR function is a so-called sinc function which is defined as $\left(\frac{\sin x}{x}\right)$. This function can be approximated by using the Gaussian function:

$$\mathrm{PTR(t)} \approx \exp\left(\frac{-t^2}{2\sigma_p^2}\right) \tag{2.31}$$

with

$$\sigma_p = \frac{1}{2\sqrt{2ln2}} r_t = \approx 0.425 r_t \tag{2.32}$$

as the width of the target response function of the radar point (Brown, 1977), while r_t represents the time resolution. Hayne (1980) showed, that the assumption of a Gaussian form of the PTR can be expressed with:

$$\mathrm{PTR(t)} = \frac{1}{\sqrt{2\pi}\sigma_s}\left[1 + \frac{\lambda_s}{6} H_3(t/\sigma_s)\right] \exp\left[-\frac{1}{2}(t/\sigma_s)^2\right] \tag{2.33}$$

in which σ_s is the RMS wave height of the surface, λ_s is the skewness and H_3 is a Hermite polynomial. Furthermore, if the height distribution of the sea surface is regarded to be a Gaussian, it can be concluded that the surface water height (SWH) is four times the RMS wave height. Therefore it is related to σ_s by

$$\mathrm{SWH} = 4(\mathrm{c}/2)\sigma_s \tag{2.34}$$

where c/2 is used to convert from ranging time to surface elevation. As Hayne (1980) pointed out, the equation 2.33 results only from taking the first two terms in a general Gram-Charlier series (Kendall, 1943) beside recent work by Huang and Long (1980) suggested that this equation is inadequate to model the surface elevation density. Hence, it is necessary to use the four-term series which is

$$\mathrm{PTR(t)} = \left[1 + \frac{\lambda_s}{6} H_3(t/\sigma_s) + \frac{\kappa_s}{24} H_4(\mathrm{t}/\sigma_s)\right] \times \frac{1}{\sqrt{2\pi}\sigma_s} \exp\left[-\frac{1}{2}(\mathrm{t}/\sigma_s)^2\right] \tag{2.35}$$

where κ_s is the kurtosis with other quantities as already defined, and the Hermite polynomials for the argument z which are used in the Equations (2.33) and (2.35) are

$$H_3(z) = z^3 - 3z \tag{2.36}$$

$$H_4(z) = z^4 - 6z^2 + 3 \tag{2.37}$$

With this, one receive the waveform $\mathrm{W}(t)$ as

$$\mathrm{W(t)} = \frac{A}{6}\exp\left[-d\left(\tau + \frac{d}{2}\right)\right]\sum_{n=0}\left(\frac{1}{n!}\right)^2\left(\frac{\beta^2\sigma}{4}\right)^n C_n(\mathrm{t}) \tag{2.38}$$

where

$$C_n(t) = C_{n0} + \kappa C_{n1} + \lambda^2 C_{n2}, \tag{2.39}$$

$$C_{n0} = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\tau}(\tau - z)^n[6 + \lambda H_3(z+d)]e^{-z^2/2}\mathrm{dz}, \tag{2.40}$$

$$C_{n1} = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\tau}(\tau - z)^n[H_4(z+d)]e^{-z^2/2}\mathrm{dz}, \tag{2.41}$$

and

$$C_{n2} = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\tau}(\tau - z)^n[H_6(z+d)]e^{-z^2/2}\mathrm{dz}, \tag{2.42}$$

with

$$\tau = \frac{\mathrm{t} - \mathrm{t}_0}{\sigma} - d, \tag{2.43}$$

$$d = \delta\sigma \tag{2.44}$$

and

$$\delta = \left(\frac{4}{\gamma}\right)\left(\frac{\mathrm{c}}{\mathrm{h}}\right)cos(2\xi) \tag{2.45}$$

with c as the speed of light in vacuum, h is the satellite altitude, γ is the antenna beam width parameter defined like in Brown's equation (Brown, 1977) by using a Gaussian approximation to the antenna gain and ξ is the absolute of nadir pointing angle (Hayne, 1980).

Ice-2 retracker

The Ice-2 retracker was developed for retracking waveforms which are received by the Ku- and C-band over continental ice sheets. The model is, like in the retracker before, derived from the Brown model. Legrésy and Remy (Legrésy and Remy, 1997; Rémy et al., 1999) fitted an error function to the leading edge by using a least square estimation while an exponential decrease function is used to span the trailing edge

$$y_k(\mathrm{t}) = \frac{P_\mathrm{u}}{2}\left[1 + \mathrm{erf}\left(\frac{k - \mathrm{bin}_{ret}}{\sigma_\mathrm{L}}\right)\right]\exp[s_\mathrm{T}(k - \mathrm{bin}_\mathrm{ret})] + P_n \tag{2.46}$$

in which $\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-k^2}\mathrm{d}k$, while k is the number of samples in a waveform, σ_L is the width of the leading edge, P_u is the amplitude of the returned waveform, s_T is the slope of the logarithm of the returned waveform at the trailing edge and P_n is the thermal noise level of the received data.

Maximum likelihood estimator (MLE)

The MLE retracker is a statistical method which fit the theoretical model return power (u_i) to the measured return power ($\hat{u}_i$) from the altimeter (Gommenginger et al., 2011). The theoretical model is thereby derived from the Brown model (Thibaut et al., 2010). Because gate-to-gate and pulse-to-pulse independence are assumed, it is possible to make convenient assumptions about the (multiplicative and negative exponential) noise on the measurements. Thereby, one can express the measure of misfit χ as (Challenor and Srokosz, 1989; Tokmakian et al., 1994):

$$\chi^2 = \sum_{i=1}^{n}\left(-N\frac{\hat{u}_i}{u_i} - N\ln u_i\right)^2 \tag{2.47}$$

with N as the number of individual echoes averaged to form the measured waveform and n as the number of gates. The aim is, to optimise the value of the estimated parameters (θ_j) in order to minimise the measure of misfit χ^2. Where the partial derivatives of χ^2 against θ_j is (Gommenginger et al., 2011):

$$\frac{\partial\chi^2}{\partial\theta} = N\sum_{i=1}^{n}\frac{\partial u_i(\theta_j)}{\partial\theta_j}\left(\frac{\hat{u}_i}{u_i^2} - \frac{1}{u_i}\right) \tag{2.48}$$

where θ_j represents the estimated parameters, e.g. epoch or significant wave height. By using the MLE estimation, one receives asymptotically, the minimum variance unbiased estimators of the wanted parameters (Gommenginger et al., 2011). The advantage is that it is possible to derive the variance-covariance matrix to gain a measure of the error on the estimates of the retrieved parameters (Tokmakian et al., 1994). The variance-covariance matrix $\mathbf{V}$ can be computed with the inverse of the Fishers information matrix $\mathbf{F}$ (Challenor and Srokosz, 1989):

$$\mathbf{V} = \mathbf{F}^{-1} \tag{2.49}$$

with

$$\mathbf{F} = \{N \sum_{i=1}^{n} \frac{1}{u_i^2} \frac{\partial u_i}{\partial \theta_j} \frac{\partial u_i}{\partial \theta_k}\} \tag{2.50}$$

where $j, k = 1, ..., npar$, with $npar$ as the number of parameters to be received.

In case of Jason-2, the MLE3 and MLE4 retracker are used, which derive 3 and 4 parameters respectively. The MLE 4 retracker is fitting from a 2nd order Brown analytical model to gain 4 parameters, which are (NASA, 2017):

- Epoch ⇒ altimeter range
- Composite sigma ⇒ surface water height
- Amplitude ⇒ σ_0
- Square of mispointing angle (only Ku-band, for C-band a null value is used as input)

As also mentioned in NASA (2017), the MLE3 retracker is fitting from a 1st order of the Brown analytical model. Thus, MLE3 simultaneously retrieve 3 parameters which can be inverted from the waveforms and are listed as follows:

- Epoch ⇒ altimeter range
- Composite sigma ⇒ surface water height
- Altitude ⇒ σ_0

Also, instrumental corrections will be offered for both retracker. However, in our analysis we will focus on the MLE3 which is offered by AVISO (CNES) and which shows good results by waveforms which match with the shape of the Brown model.

In chapter 5, the threshold retracker, as well as the MLE3 retracker, are used for the comparison. With this, an empirical retracker and a retracker, based on the Brown analytical model are used to see their performance compared to the developed approaches can be shown.

Chapter 3

Neural networks

Neural networks are more and more important these days. Thus, it may be surprising that the first works on this topic are done in the 1940s. That time it was called cybernetics. However, the name 'neural network' suggests, that they are based on biological neurons. In fact, the brain-inspired the structure of the neural networks but they are no realistic models of them (Goodfellow et al., 2016). The theory behind the neural networks will at first be explained using a biological point of view and based on that, be transferred to the artificial approach. The descriptions of the neural network in the following sections are based mostly on the German version of the book 'Make your own neural network' from Rashid (2016). The main reason is that the networks used in the thesis are based on the network, which is described in this book, all other sources are named explicitly.

3.1 From the biological to the artificial neuron

In Figure 3.1, a neuron is shown as it is also represented in the human brain. Taking a look at their basic structure, one can see at the beginning, so-called dendrites. They are connected to other neurons and receive signals from them. These signals are then forwarded to the corpus of the neuron in which the actual processing takes place. After this calculation step, the signal is propagated forward along the axons which then transmit it to the dendrites of the neuron in the next layer (Rashid, 2016). Roughly spoken, the signal is received, applied to a short processing step and then forwarded to the next one. However, regarding the brain, the neurons are organised in different layers. This structure is also very important for artificial neural networks. But before the organisation of layers are discussed, the calculations which are conducted inside the neuron bodies are analysed more closely.

Given a big network of neurons, they propagate signals through it, but thereby each neuron has to decide if it transmits the signal to the next neuron or not. The reason behind this is that organs of perception receive signals continuously and transfer them to our brain. But of course not all signals are relevant, and one is mostly focusing on the strongest signals. Therefore, it has to be decided which signals are important enough to be transported forward and which are not. At first, one can assume to use a step function for this purpose, as it can be seen in Figure 3.2, in which x_i is the strength of the input signal. By using this approach, it can already be distinguished between signals which have to be forwarded and which not. However, the problem by using this approach is, that there are only two possibilities, namely forwarding the signal or not. In contrast, humans can distinguish between the intensity of a signal, which is not possible by using this approach. The reason is, that there is no possibility to regulate the

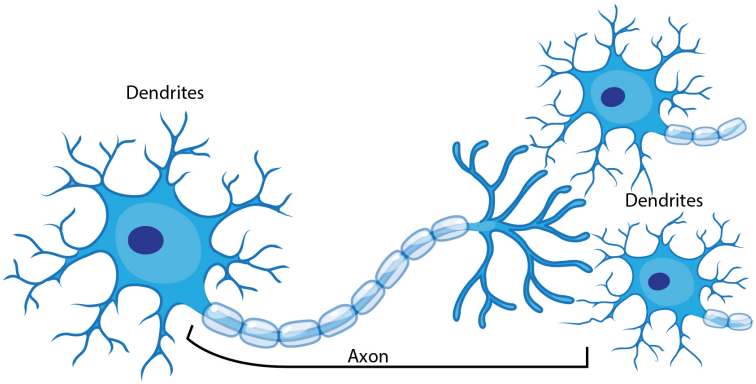


Figure 3.1: *Schematic representation of a biological neuron (Source: Source: https://askabiologist.asu.edu)*

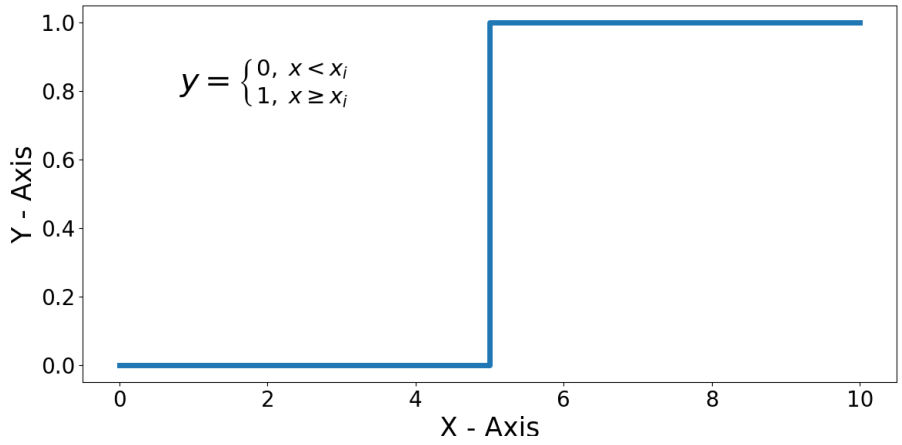


Figure 3.2: *Example of a step function*

intensity of the forwarding signal. Thus, the approach has to be modified towards a smooth increase of the signal. This is done by the so-called activation function.

An activation function creates a smooth increase of the output value depending on the input value. For neural networks a wide range of these activation functions are used, each of them has its advantages and disadvantages. In the current case, the most common one, the so-called sigmoid function, is applied, which is shown in Figure 3.3. As it can be seen there, the result improved compared to the step function approach as there is a smooth increase of the forwarding signal. With this, it is now possible to distinguish even between small variations of the input signal. Thus, small signals would be blocked, and with increasing strength, they are propagated forward with increasing power. By doing so, the output values are limited between zero and one.

Regarding Figure 3.1, one can see that there are several neurons connected and to simulate these structures, one has to use weights which are then multiplied with the signal which is transferred with the connection. These weights are used to regulate the power of the signal which is propagated to the neuron. The reason behind them is explained later when it comes

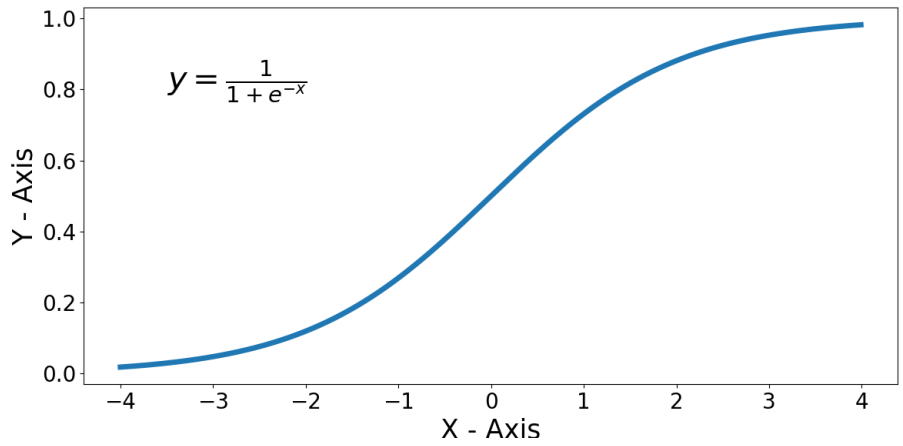


Figure 3.3: *Example of a sigmoid funciton*

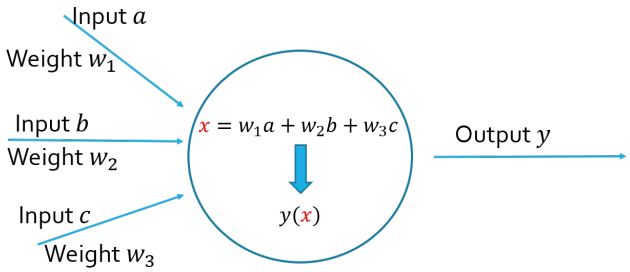


Figure 3.4: *Artificial Neuron with three input signals*

to the learning process of the neural network. However, for now, it is important to know, that the signals from the previous neurons are regulated with them. After that multiplication, the sum of the resulting input signals is calculated by the neuron. In Figure 3.4 one can see a sketch of an artificial neuron which has three input signals s_i from the previous neurons which are multiplied by the corresponding weights w_i. It illustrates the needed calculations inside an artificial neuron and contains the following two steps:

- Calculate the sum of the weighted signals: $x = \sum_i w_i \cdot s_i$
- Use the sigmoid function to calculate the output value with the sum x: $y = \frac{1}{1+e^{-x}}$

After the functionality of an artificial neuron is explained, one can combine them to create a network, which is later used in the retracking approach.

3.2 The network

Biological neurons are organized in layers which propagate the signals. This can be also transferred to the artificial neurons by creating a so-called artificial neural network. The basic structure of a neural network, as it is used in this thesis, can be seen in Figure 3.5. This is a so-called

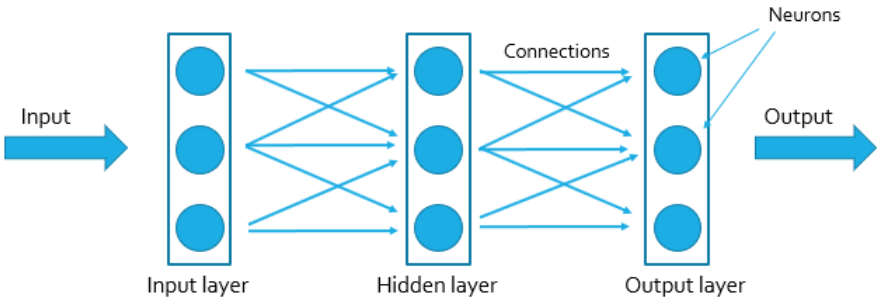


Figure 3.5: *Single layer feed foreward neural network*

single layer feed-forward neural network. Hence, there is only one hidden layer used in the neural network. Besides that, the flow of information is strictly from the input layer towards the output layer and thus in one direction, namely forward.

The functions of the layers can be explained as the following:

Input layer: This layer can be seen on the left side of Figure 3.5. It is used to receive the data, which has to be analysed with the network. Also, the number of neurons inside the input layer is fixed because it has to be equal with the number of variables which have to be processed.

Hidden layer: Inside this layer, the main process is conducted. In opposite to the input layer, this layer is equipped with activation functions, which is in this study the sigmoid function. By using this function, the intensity of the signal, which has to be propagated to the next neuron, is regulated. In the current example, only one hidden layer is included, but there is no limitation. However, it is even possible that each layer has another activation function (Gardner and Dorling, 1998).

Output layer: In this layer one receive the resulting values of the neural network. The number of neurons inside this layer is equivalent to the number of classes in which the dataset has to be separated. Thus, the output value of each neuron of the output layer is the probability of the class it represents.

The neurons of this network are fully connected. Thus it is a so-called fully connected neural network. The connections also have a significant role inside the neural network which will be explained in the following chapter. Because the sigmoid function, which is used in the hidden layer, has only values between zero and one, also the input data has to be in that range. That means one has to normalise the input data. However, because the aim of the neural network is the classification, the next section is describing the network from a machine learning perspective.

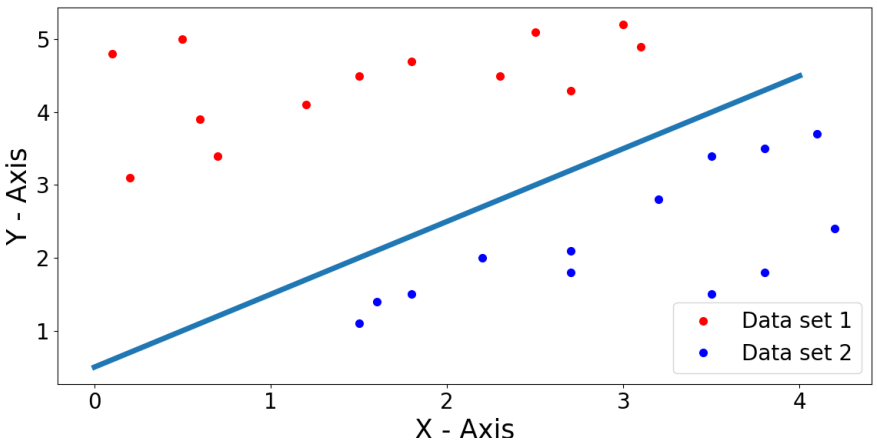


Figure 3.6: *Data set which can be separated by using a linear function*

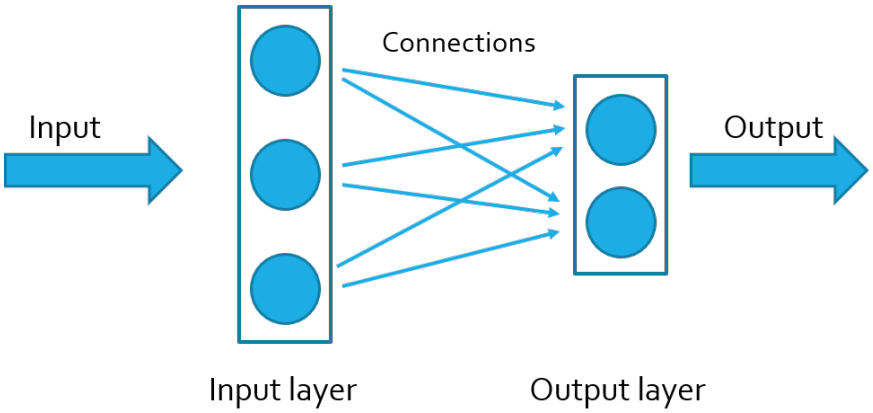


Figure 3.7: *Neural Network without hidden layer*

3.3 A machine learning perspective

One of the primary tasks of machine learning is, to separate a given dataset in several classes, which is also the aim of a neural network. If only the input and the hidden layer are used, they are combined linearly with each other because no activation function is used. With them, the data separation in Figure 3.6 can be conducted. As it can be seen, only one line is necessary to divide two sets. Thus, it can be separated by using a linear function and therefore, no hidden layer is needed. The neural network for this task can be seen in Figure 3.7.

Unfortunately, in most cases, the data sets cannot be separated by a linear function. Therefore, one has to apply a hidden layer to separate the two classes correctly. As it can be seen in Figure 3.8, it needs two lines to separate them which can be achieved by using one hidden layer, as it is used in Figure 3.8. The next question is, how to decide which number of neurons inside the hidden layer is needed? There are several rules of thumb on how to define the number of neurons, such as (Heaton, 2017):

- The number should be between the number of neurons in the input and output layer.

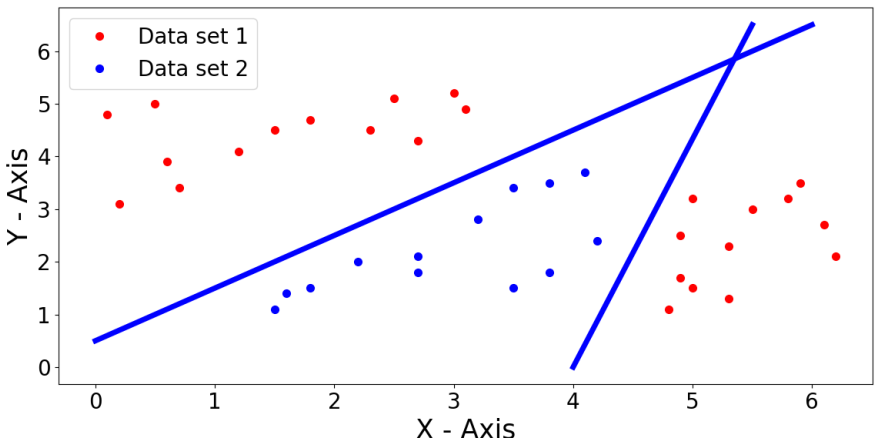


Figure 3.8: *Data set which is not possible to separate with one linear function*

- The number of the neurons in the hidden layer should be 2/3 of the number of neurons in the input layer plus the number of neurons in the output layer
- The number should be less than twice the number of neurons in the input layer

However, in the next chapter, the question will be discussed in more details on how to find the best number of neurons in the case of this study. The number of hidden neurons is important, as it has a big influence on the overfitting and underfitting. That means, fewer neurons lead to the problem, that the neural network would only focus on the most prominent patterns. This could lead to a very rough assumption of the problem thus in many cases the neural network can not classify all test data correctly. On the other hand, a too big number of neurons in the hidden layer result in the fact, that the neural network would be able to learn every small variation during the training. Therefore, the neural network would be focused on the training data set, and if the test data, which have to be classified, are introduced to the network, the network would again face difficulties to assign the data to the correct class. One can see in Figure 3.4, that there are only two calculations conducted inside the body of the neuron. However, to enable the network to learn patterns it needs more than that. In the next section, it is discussed, how the connections between the layer can be used for learning.

3.4 How neural networks learn

It is again possible to draw parallels between the learning process in biological neurons and artificial ones. The human brain is learning by building connections between neurons. Thus, the same is done in case of artificial neural networks. As already mentioned, the current neural network is a so-called fully connected network, meaning, all neurons of one layer are connected to the neurons in the next layer. Hence, one can not create new connections between them during the learning process. However, it is possible to apply weights to the connections. These weights are then multiplied with the signal the connection is transporting. By doing so, it is possible to adjust the strength of the propagated signals. The importance of the strength of a signal becomes relevant if one remembers how a neuron works, more specific, how the sigmoid

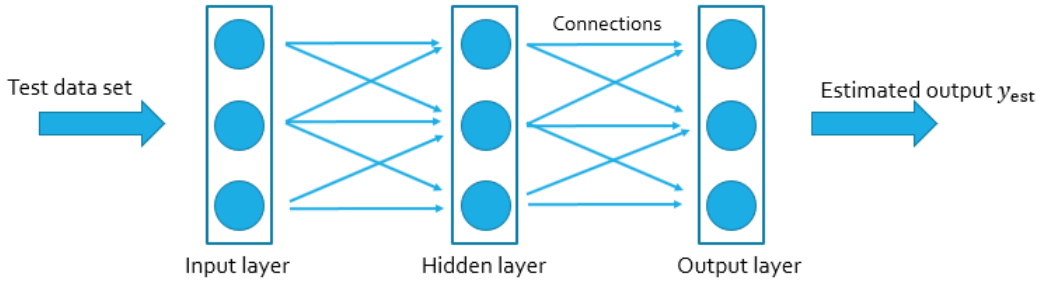


Figure 3.10: *First step of the backpropagation*

function work. Because the strength of the output value directly influences the output value, which can be seen in Figure 3.9. Regarding the changes in Figure 3.9 also the calculation of the input value x for each neuron has to be adjusted as follow:

$$x_l = \sum_i w_{i,l} \cdot s_{i,l} \tag{3.1}$$

In this case, x_l is the resulting input signal of the actual neuron l, w and s are the weights and the signals which come from the neuron i from the previous layer and go to the neuron l in the current layer. With this formula, it is now possible to compute input values depending on the weights of the connections. In the next step, the learning process can be discussed, which change the weights of the connections. This is done by using the so-called backpropagation.

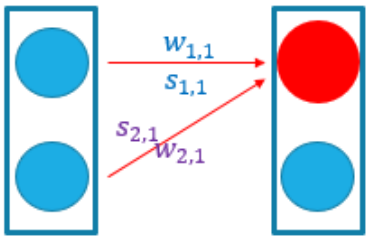


Figure 3.9: *Each connection has its own weight*

3.4.1 The backpropagation

In machine learning algorithms, one has usually two steps, first the training phase in which the algorithm is learning the characteristic pattern by using data, in which all samples are already labelled. The second phase is the testing phase, in which the algorithm has to label unknown data sets. Thus, now the first step will be discussed in which known samples are used to train the neural network. Hence, in the beginning, the weights are set with random values and will later be corrected towards the best value. By applying the test data, one receives a first estimated output y_{est} of the neural network, as it is shown in Figure 3.10.

These estimated values y_{est} have to be compared now with the given correct solution y_{true}:

$$e = y_{\text{true}} - y_{\text{est}} \tag{3.2}$$

This leads to the error e, which is used to correct the weights of the connections. In Figure 3.11, an example neuron of the output layer is given with two connections.

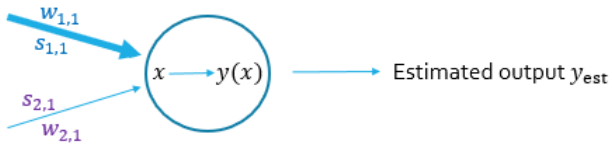


Figure 3.11: *Closer look at a neuron in the output layer*

As it can be seen in Figure 3.11, the weights change because of the usage of random values during the first run. In this case, the upper weight $w_{1,1}$ is bigger than the lower one, which means that the signal transported by this connection has a more significant influence on the result and hence also on the error. Thus, this weight has to be corrected more than the other weight $w_{2,1}$, which only has a small contribution to the result. It means the error has now be distributed to the connections depending on the actual weight of it. This can be done by using the following formula:

$$e_{i,l} = \frac{w_{i,l}}{\sum_i w_{i,l}} \cdot e_l \tag{3.3}$$

With this Formula (3.3), it is now possible to distribute the error to the connections:

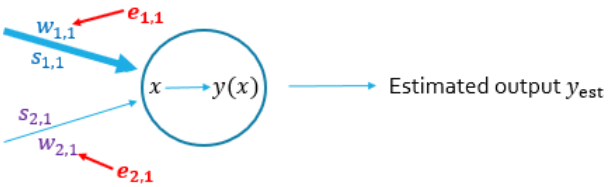


Figure 3.12: *Correction of the weights with the distributed error*

Of course, this error propagation has to be done through the whole neural network backwards. In the next step, one has to find out, how to calculate the error values of the hidden layer outputs, because no reference values are given, as they are available for the output layer. This is solved by combining the distributed errors to create the total error of the output for each neuron in the hidden layer. In the following figure, it has to be regarded that the indices have changed for the errors. Now it is important to note, that the first layer in the graphic is the hidden layer and the second one the output layer. Therefore, the error values are named $e_{o,i}$, o stands for the output layer, and i is the number of the output neuron. Thus, $e_{o,1}$ is the error of the first neuron in the output layer:

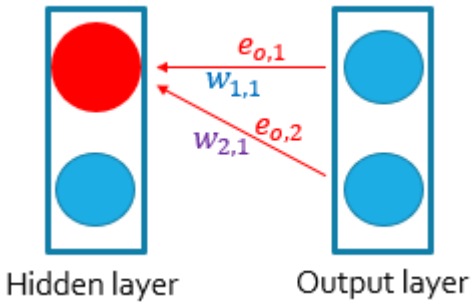


Figure 3.13: *Backpropagation to the hidden layers*

Regarding the example in Figure 3.13, the first neuron in the hidden layer is now used as an example to calculate the error. Which lead to the following formula:

$$e_{h,1} = e_{o,1} \cdot \frac{w_{1,1}}{w_{1,1} + w_{2,1}} + e_{o,2} \cdot \frac{w_{1,2}}{w_{1,1} + w_{2,1}} \tag{3.4}$$

With this formula it is now possible to propagate the error through the whole network and therefore, to adjust all weights. After all weights are corrected, one starts again from the beginning and uses the training data to calculate the estimated output y_{est}, which then can be used again to calculate the error by using the Formula (3.2). However, until now it was just described how to correct the weights depending on the errors of the output layer. When it comes to the correction of the weights between input and hidden layer, one has to regard the activation function as well as the summation of all input signals. Thus, if one wants to adjust the weight between these two layers in a way, that a specific result is reached during the training phase one need the following formula:

$$o_k = \frac{1}{1 + e^{-\sum_{j=1}^{3} \left(w_{j,k} \cdot \frac{1}{1 + e^{-\sum_{j=1}^{3} \left(w_{i,j} \cdot x_i \right)}} \right)}} \tag{3.5}$$

Regarding Formula (3.5) the weights $w_{i,j}$ connect the input layer with the hidden layer and the weights $w_{j,k}$ connect the hidden layer with the output layer. The variable x_i stands for the input signal at the beginning of the neural network and o_k is the variable for the output value of the neuron k in the output layer. Thus, the relationship between the variables are very complex and a neural network with even more layers and neurons in each layer it would become even more difficult. Hence, another solution is needed to adjust the weights depending on the errors. This can be done by the so-called gradient descent algorithm.

Introduction of the gradient descent

In the following, an algorithm called gradient descent will be introduced, which minimises the sum of squared errors comparable to the ordinary least squares algorithm. The main difference is that the gradient descent algorithm calculates the sum of squared errors iteratively to find

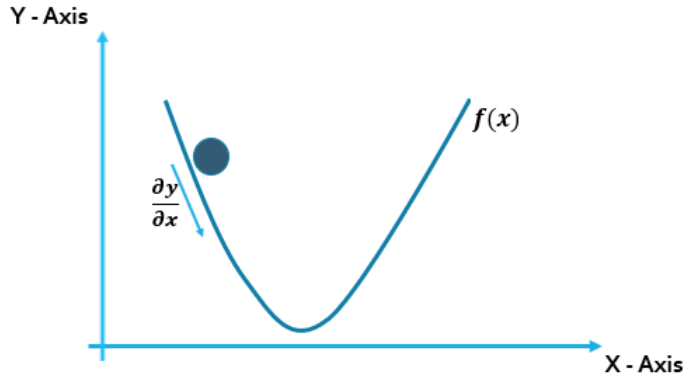


Figure 3.14: *Gradient descent towards the minimum*

the global minimum error. At each step, the partial derivative of the function is calculated, though the gradient. This gradient is used to see, in which direction one has to go if one wants to reach the global minimum. This can be reached if one follows the direction of the negative gradient. The step size, in which the parameters are changed is called learning rate, which is predefined. Thus, the learning rate defines the rate of change for each parameter during one iteration. The errors are then minimized until a global minimum is reached, by iteratively adjusting the parameters. The main advantage of the gradient descent is the performance if one wants to estimate the values of a high number of parameters.

3.4.2 The gradient descent

The aim is now, to get the combination of weights which lead to the minimum global error. Thus, the algorithm called gradient descent is used to estimate the minimum of the error by searching the lowest point in the function. To have a picture in mind, it can be said, that having a function $f(x)$ and one want to find the minimum by calculating the gradient $\frac{\partial x}{\partial y}$. By doing it we can find the way towards the minimum, which can be seen in Figure 3.14.

The first step is now to find the most useful function to apply. This function has to represent the errors which have to be minimised by this algorithm. Regarding Equation (3.2) it can be seen that the error is easy to calculate. The problem with this function is, that if for example the error in one neuron is -0.2 and in another neuron, the error is 0.1, the overall error would be just -0.1 which of course is not correct if during the calculations one wants to control the size of all errors together. Hence, another formula is needed which is also continuous. One approach is the following formula:

$$E = \sum_{i}^{n} (y_{\texttt{true},i} - y_{\texttt{est},i})^2 \tag{3.6}$$

So it can be seen in Equation (3.6) that the error function E is the summation of the quadratic difference between the correct value and the estimated one, whereat the sum is over all n output neurons. Another important property of this function is that towards the minimum the gradient becomes smaller. This is very important because the aim is, metaphorically speaking,

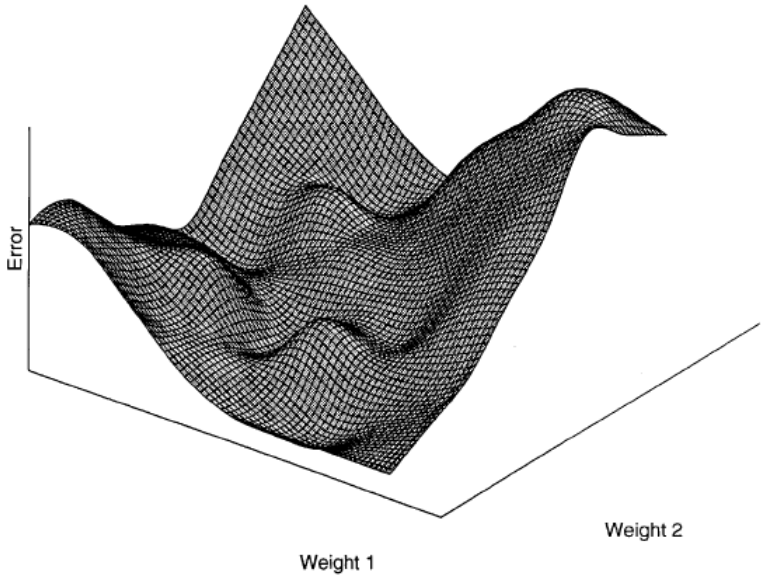


Figure 3.15: *Graphical representation of the error function depending on two weights (Gardner and Dorling, 1998)*

to walk down the function towards a minimum. It can be seen as hiking in the mountains and the only orientation to find a way down is the gradient. Thus it is important to choose smaller steps when it comes towards the minimum. Graphic 3.15 shows, that the analogy matches very well. As it can be seen, the error value depends in this case only on two weights, which makes it equivalent with graphic 3.12.

The problem in this approach is the decision of the correct step size. The main aim is to find the global minimum, but if the step size is 2, but the minimum is only 0.7 elements away it would be not possible to reach the minimum. Thus, the property of E is now essential, because the gradient decreases if it is closer to the minimum. Taking a look at Figure 3.15, one can see that there is one global minimum, but also several local minima. By using only the gradient to find the global minima, only a local minimum may be found. This would also reduce the error, but it would not result in the best solution which is possible. One option to avoid this is, to chose at the beginning bigger step sizes which have a high potential to miss the smaller local minima. However, it is also common to repeat this process several times and use always different weights at the beginning. With this, it can be secured to use different paths downwards and avoid choosing the same local minimum. The next step is now to show the mathematical aspect of gradient descent.

3.4.3 Changing the weights of the network

In this chapter Equation (3.6) and the discussed gradient descend will be combined to correct the weights between the layers properly. At first, one has the error e which is depending on the weights and can be seen in Figure 3.15. To start, the gradient of the error function which is used to adjust the weighs between the hidden layer and the output layer, is discussed. This gradient looks like the following:

$$\nabla E_{i,k} = \frac{\partial E}{\partial w_{i,k}} \tag{3.7}$$

A small simplification of Equation (3.6) can be done. Because the output of a neuron just depends on the weights which are connected to this neuron, the sum over all output neurons is not necessary. This has the advantage that the sum can be removed, which results in the following formula:

$$\nabla E_{i,k} = \frac{\partial}{\partial w_{i,k}} (y_{\text{true},k} - y_{\text{est},k})^2 \tag{3.8}$$

In the first step of derivation one can apply the chain rule to the error function:

$$\nabla E_{i,k} = \frac{\partial E}{\partial y_{\text{est},k}} \cdot \frac{\partial y_{\text{est},k}}{\partial w_{i,k}} \tag{3.9}$$

Next, the first part of the term can be derived which lead to:

$$\nabla E_{i,k} = -2 \cdot (y_{\text{true},k} - y_{\text{est},k}) \cdot \frac{\partial y_{\text{est},k}}{\partial w_{i,k}} \tag{3.10}$$

The second part is a bit more difficult to derive, because of the sigmoid function:

$$\nabla E_{i,k} = -2 \cdot (y_{\text{true},k} - y_{\text{est},k}) \cdot \frac{\partial}{\partial w_{i,k}} \frac{1}{1 + e^{-\sum_i w_{i,k} \cdot y_{\text{est},i}}} \tag{3.11}$$

In Equation (3.11) $y_{\text{est},k}$ is the output value of the neuron in the output layer and $y_{\text{est},j}$ is the output value of the neuron in the previous hidden layer. The advantage is, that the derivative of the sigmoid function can be expressed as:

$$\frac{\partial}{\partial x} \frac{1}{1 + e^{-x}} = \frac{1}{1 + e - x} \cdot \left(1 - \frac{1}{1 - e^{-x}}\right) \tag{3.12}$$

With this, the gradient can be written as:

$$\nabla E_{i,k} = \frac{\partial E}{\partial w_{i,k}} = -2 \cdot (y_{\text{true},k} - y_{\text{est},k}) \cdot \frac{1}{1 + e^{-\sum_i w_{i,k} \cdot y_{\text{est},i}}} \cdot \left(1 - \frac{1}{1 + e^{-\sum_i w_{i,k} \cdot y_{\text{est},i}}}\right) \cdot \frac{\partial}{\partial} \sum_i w_{i,k} \cdot y_{\text{est},i} \tag{3.13}$$

$$\nabla E_{i,k} = \frac{\partial E}{\partial w_{i,k}} = -2 \cdot (y_{\text{true},k} - y_{\text{est},k}) \cdot \frac{1}{1 + e^{-\sum_i w_{i,k} \cdot y_{\text{est},i}}} \cdot \left(1 - \frac{1}{1 + e^{-\sum_i w_{i,k} \cdot y_{\text{est},i}}}\right) \cdot y_{\text{est},i} \tag{3.14}$$

With this, the gradient of the error function is now calculated, regarding the weights between the hidden and output layer. Formula (3.14) can be now analysed a bit more closely. The first part $(y_{\text{true},k} - y_{\text{est},k})$ is the common error which was already mentioned in Equation (3.2). This is then multiplied by the sigmoid functions which sum up the output of the neurons from the output layer. At the end of the formula, the output of the hidden layer is multiplied to the term. In the last step, the gradients of the error function regarding the weights between the input layer and the output layer will be calculated. Hence, Equation (3.14) can be used and modified as followed:

Error calculation: as already mentioned, no reference values for the calculations of the error of the hidden layer is available. However, with Formula (3.4) it is possible to calculate these errors

Sigmoid functions: in sigmoid functions, nearly the same summations can be used, but now outputs of input layer neurons and the weights between input and hidden layer are regarded.

Output value: as before, the output value of the first of the two layers, which are regarded in the calculations, is multiplied. In the current case, the output of the input layer is multiplied.

This new gradient of the error function can be seen in the next equation:

$$\nabla E_{j,i} = \frac{\partial E}{\partial w_{j,i}} = -2 \cdot e_{h,i} - \cdot \frac{1}{1 + e^{-\sum_i w_{j,i} \cdot y_{\text{est},j}}} \cdot \left(1 - \frac{1}{1 + e^{-\sum_i w_{j,i} \cdot y_{\text{est},j}}}\right) \cdot y_{\text{est},j} \tag{3.15}$$

With this, the weights can now be corrected, depending on the gradients:

$$w_{\text{new,j,i}} = w_{\text{old,j,i}} - \alpha \cdot \frac{\partial E}{\partial w_{j,i}} \tag{3.16}$$

$$w_{\text{new,i,k}} = w_{\text{old,i,k}} - \alpha \cdot \frac{\partial E}{\partial w_{i,k}} \tag{3.17}$$

In Equation (3.16) the weight adjustment between input and the hidden layer can be seen, and in Equation (3.17) the adjustment of the weights between hidden and the output layer is shown. Also, a new variable is introduced, which is α. This is the so-called learning rate and is used to moderate the change caused by the gradient. As it can be seen in both formulas, the old weight is subtracted by the gradient. The reason behind it is, that one wants to move in the opposite direction of the gradient. With these formulas, it is now possible to train the weights in a way, that the neural network can learn. The next section is explaining the output of the neural network.

3.4.4 The output of a neural network

As shown in Figure 3.3, the sigmoid function is in a range between zero and one. Because a sigmoid function is also included inside the output layer, the output of the neural network is between these two values. Thus, these values can be interpreted as the probability of each label. If the signal of the first output neuron is strong, the sigmoid function will create a high output value, which is a value close to 1, or in other words, close to 100%. Thus, one has a high probability of this label. The question, how the network can assign a probability to a label is solved, by defining an output vector with the size of the searched labels. For example, to classify three labels (red, green and blue) the output vector will contain three elements:

$$\mathbf{O}_o = \begin{pmatrix} o_{o,1} \\ o_{o,2} \\ o_{o,3} \end{pmatrix} \tag{3.18}$$

Because the output of each neuron is equal to the probability of the label, one has to search the position of the maximum value inside this vector. With the position, one has also received the label which is, to a certain probability, the label. For example an output layer like the following one

$$\mathbf{O}_o = \begin{pmatrix} 0.0025 \\ 0.9612 \\ 0.0063 \end{pmatrix} \tag{3.19}$$

is pointing out, that the second label has the highest possibility and therefore the label 2 is the searched one. After this, all parts of the neural network is explained and also the learning process regarding backpropagation and gradient descent. In the next section, the implementation of neural networks will be discussed.

3.5 The implementation of a neural network

3.5.1 Forward propagation of the signals

In this chapter, the implementation of the previously discussed calculations will be shown. However, the implementation of the forward propagation is easy to conduct by using matrices and vectors for the calculations. Thereby, each layer is represented by a vector, and matrices represent the weights between the layer. Hence, the input layer would look like the following:

$$\mathbf{I} = \begin{pmatrix} i_1 \\ i_2 \\ i_3 \end{pmatrix} \tag{3.20}$$

While the weights between the input layer and the hidden layer can be shown as:

$$\mathbf{W}_{\mathrm{i,h}} = \begin{pmatrix} w_{1,1} & w_{2,1} & w_{3,1} \\ w_{1,2} & w_{2,2} & w_{3,2} \\ w_{1,3} & w_{2,3} & w_{3,3} \end{pmatrix} \tag{3.21}$$

With these two, Equation (3.1) can be realized by using the matrix multiplication for the combination of (3.20) and (3.21)

$$\mathbf{X}_{\mathrm{h}} = \mathbf{W}_{\mathrm{i,h}} \cdot \mathbf{I} \tag{3.22}$$

Now all values needed of the hidden layer are calculated, which depends on the weights of the connections between input and hidden layer. Next, the sigmoid function is applied as an activation function to calculate the output of the neurons. This can be done by applying the sigmoid function to every element of the hidden layer:

$$\mathbf{O}_{\mathrm{h}} = \frac{1}{1 + e^{-\mathbf{X}_{\mathrm{h}}}} = \mathrm{sigmoid}(\mathbf{X}_{\mathrm{h}}) \tag{3.23}$$

The next two calculations are like the previous ones, meaning that first the input of the output layer is calculated by using the weight matrix between the hidden layer and output layer which are then multiplied with the output of the hidden layer (see Equation (3.23)):

$$\mathbf{X}_{\mathrm{o}} = \mathbf{W}_{\mathrm{h,o}} \cdot \mathbf{O}_{\mathrm{h}} \tag{3.24}$$

In the last step, again the sigmoid function is used to calculate the output of the neural network

$$\mathbf{O}_{\mathrm{o}} = \frac{1}{1 + e^{-\mathbf{X}_{\mathrm{o}}}} = \mathrm{sigmoid}(\mathbf{X}_{\mathrm{o}}) \tag{3.25}$$

After that, the forward propagation could be implemented by only using basic matrix calculations. The next step is a bit more difficult since the backpropagation has to be computed and also the weights between the layers have to be adjusted by using the gradient descent method.

3.5.2 Changing the weights with matrices

The calculation of the errors of the output layer, as it was shown in (3.2), has now to be changed from scalar values to vectors

$$\mathbf{e}_{\mathrm{o}} = \mathbf{y}_{\mathrm{true}} - \mathbf{O}_{\mathrm{o}} \tag{3.26}$$

The next step is to propagate the error through the network as done in (3.4). Realising this formula by using matrices, one gets a result as

$$\mathbf{e}_{\mathrm{h}} = \begin{pmatrix} \frac{w_{1,1}}{w_{1,1}+w_{2,1}+w_{3,1}} & \frac{w_{1,2}}{w_{1,2}+w_{2,2}+w_{3,2}} & \frac{w_{1,3}}{w_{1,3}+w_{2,3}+w_{3,3}} \\ \frac{w_{2,1}}{w_{1,1}+w_{2,1}+w_{2,1}} & \frac{w_{2,2}}{w_{1,2}+w_{2,2}+w_{3,2}} & \frac{w_{2,3}}{w_{1,3}+w_{2,3}+w_{3,3}} \\ \frac{w_{3,1}}{w_{1,1}+w_{2,1}+w_{2,1}} & \frac{w_{3,2}}{w_{1,2}+w_{2,2}+w_{3,2}} & \frac{w_{3,3}}{w_{1,3}+w_{2,3}+w_{3,3}} \end{pmatrix} \cdot \begin{pmatrix} e_{\mathrm{o},1} \\ e_{\mathrm{o},2} \\ e_{\mathrm{o},3} \end{pmatrix} \tag{3.27}$$

as it can be seen, a 3×3 matrix is used in this example, which is still possible to implement. However, using bigger matrices in later calculations, the implementation would face computational difficulties. However, a closer look at the elements of the matrix shows, that there is a way to simplify it for the implementation. The point is that the denominator can be regarded as a scale factor which can be neglected. Of course, in the previous section, it was stressed that it is essential to scale the correction of the error depending on the sum of the weights, which is for sure correct. However, the implementation shows, that the most important element is the numerator of the fraction, which is the actual weight of the connection. Also, the network is regulating a weight which was changed too high or too low in the next iteration of the learning process. By doing so, the equation is reduced to

$$\mathbf{e}_{\mathrm{h}} = \begin{pmatrix} w_{1,1} & w_{1,2} & w_{1,3} \\ w_{2,1} & w_{2,2} & w_{2,3} \\ w_{3,1} & w_{3,2} & w_{3,3} \end{pmatrix} \cdot \begin{pmatrix} e_{\mathrm{o},1} \\ e_{\mathrm{o},2} \\ e_{\mathrm{o},3} \end{pmatrix} \tag{3.28}$$

$$\mathbf{e}_{\mathrm{h}} = \mathbf{W}_{\mathrm{h,o}}^{T} \cdot \mathbf{e}_{\mathrm{o}} \tag{3.29}$$

with this equation, it is easy to propagate the error towards the next layer by transpose the error matrix. Now, all informations are computed to adjust the weights. Thereby, Equation (3.15) can be used as a model and can be changed for the usage of matrices to

$$\Delta \mathbf{W}_{\mathrm{i,k}} = \alpha \cdot \mathbf{e}_{\mathrm{k}} \cdot \mathbf{O}_{\mathrm{k}}(1 - \mathbf{O}_{\mathrm{k}}) \cdot \mathbf{O}_{\mathrm{i}}^{T} \tag{3.30}$$

which is the correction of the weights between the output layer and the hidden layer. With this, the implementation of the neural network is finished.

3.5.3 Defining initial values

Before the calculation can start, the implementation of initial values for the weights between the layers are needed. If one wants to optimise the results, it becomes more tricky to find the best ones. The first important point which has to be considered is, that too high starting values would lead to a saturation of the sigmoid function which can be seen in Figure 3.3. It can be seen, that the gradient of the sigmoid function in case of high values is smaller than by smaller values. However, because using the gradient of Function (3.16), a small gradient is reducing the learning capability. Hence, it is helpful to use random and uniformly distributed values between -1.0 to $+1.0$. However, more advanced solutions which would improve the results even more are possible. A rule of thumb can be used, which shows that randomly chosen values which are within

$$-\frac{1}{\sqrt{n}} \leq w_i \leq +\frac{1}{\sqrt{n}} \tag{3.31}$$

with n as the number of connections to one neuron. In the current case with three connections to one node the range would be between

$$-\frac{1}{\sqrt{3}} = -0.58 \leq w_i \leq +\frac{1}{\sqrt{3}} = +0.58 \tag{3.32}$$

The reason behind it is on the one hand that a high value of weights would lead to the saturation mentioned above of the sigmoid function which would reduce the learning ability. On the other hand, if there are a lot of connections towards one neuron, they will be accumulated which lead again to high values, if the weight of each connection is not small enough. The discussed theory will be important in the next chapter in which algorithms are developed to retrack waveforms of the satellite altimetry.

Chapter 4

New retracking approaches

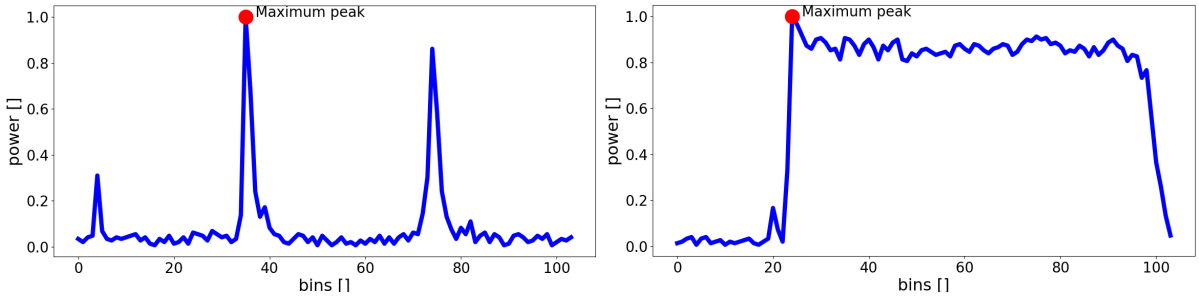
In this chapter two different approaches to retrack, the waveforms will be discussed. In a first approach, a neural network will be used as it was described in the previous chapter. After that, a more sophisticated algorithm will be applied which uses the output of the neural network to estimate the leading edge position.

4.1 A neural network approach

4.1.1 Processing the data

The first approach can be seen as a standard approach for classification. Thereby, the neural network will be trained to estimate the position of the correct peak with the leading edge. However, before the description of the neural network start, the used data has to be discussed, because the choice of parameters depends directly on it. All values inside the neural network have to be between zero and one. One reason is the used sigmoid function as an activation function which creates also output values within that range. However, the other important reason is, that scaling differences have to be eliminated. For example, a waveform with small values but a clear leading edge should get the same influence to the learning process than a waveform with high maximum values. The reason is that the calculated error of a waveform with big numbers is higher than the error of a waveform with small values. Hence, the correction of the weights would mostly be influenced by the waveforms with big values which can create a bias. To avoid this, the normalisation is an important step to prepare the data, which will be used in the neural network. Thus, the first step is the rescaling between zero and one that all waveforms have the same influence during the learning process without creating any bias.

After scaling the waveforms, the neural network has to be trained at which part of the waveform the leading edge is located. This is done by labelling the waveforms. The question is, with which criteria the correct peak and leading edge have to be chosen. Thereby it should be focused on the usability of the detection because hundreds of waveforms of training data sets have to be labelled. However, it has to be mentioned that neural networks can cope with a number of wrong labels as long as they remain on a low percentage. A visual analysis of the waveforms shows that the leading edges go in the most cases hand in hand with the highest peak in the waveform. Thus, one can define the position of the highest peak of the waveform with the neural network and use this information to calculate the leading edge position. To



(a) (b)

Figure 4.1: *Normalised waveforms with marked maximum peak*

get the information about the accuracy of the neural network, all used data sets are labelled, with the difference that the network only get the label of the training data sets as additional input. The labels of the validation and test data are solely used to estimate the accuracy of the network. Thus, the following steps of data preparation have to be conducted to receive the waveforms in Figure 4.1:

1. Normalisation of the waveforms

2. Labelling of the waveforms

As a next step, one has to decide which data is to use for training, validation and testing:

- **Training data:** this data set is used to train the neural network that it can adjust the weights of the connections. Thus, this data set is important as the network learn based on these informations how to estimate the leading edge.

- **Validation data:** if one wants to see, if the gradient descent method during the training data phase showed well results, an independent data set is needed. Because the network already used the training data for the learning process, it is familiar with these waveforms. Hence, a validation test with the training data would not show the performance of the network in case of unknown data correctly. Thus, a new set of data is needed, namely the validation data set.

- **Test data:** This set of data is applied to the network after the training process. It contains all waveforms which have to be analysed.

However, the more training data is used, the better is the results because the network can learn more pattern which improves the ability to identify the searched patterns. However, if this approach is later used in analyses, one can assume that not always a high amount of data is available. Thus only a smaller number of data can be used for training. Because of this, multiple scenarios are tested to show the performance of networks with different numbers of training data sets. In this chapter, the data of the Cupari river is used. There, an amount of 305 tracks are available which contain always 6 waveforms. Therefore, the following combinations of data sets are used:

Table 4.1: *Number of the used waveforms*

	Training	Validation	Testing
Example 1	600	60	870
Example 2	720	60	870
Example 3	900	60	870

In Table 4.1 one can see, that always the same number of validation and test data is used and just the number of training data changes. The reason is that with this all results are comparable and show the same time frame. In the next chapter, the chosen network parameters are discussed. This means that for example the first 600 waveforms are used to train the network, the next 60 waveforms are used to validate it and the last 870 waveforms are used to test it. Hence, the data of a couple of years is used during training and testing.

Terminology of neural networks

In the following chapters, some new vocabulary is used, which are explained in this section, and also some previously explained elements of the neural network are explained shortly regarding the current case:

- Learning rate: the learning rate is related to the gradient descent algorithm and describes the step size of the algorithm to "walk down" the function. The size is important as a big step size could lead to the problem, that one misses the global minimum, and if the step size is too small, one could choose a local minimum, as it is explained in Section 3.4.2.
- Epoch: in the case of neural networks, epochs represent the number of iterations in which the training data is applied to the network. In most cases, it is not enough to train the network with the training data only once, because the number of parameters which have to be adjusted are too high. Therefore, the training data has to be applied to the neural network multiple times that the weights can be appropriately adjusted.

4.1.2 Defining the network parameter

Before a network can be created, the following questions have to be answered:

1. Which sizes have the input and output layer?
2. How many hidden layers are needed and how many neurons should they include?
3. Which learning rate and number of epochs should be chosen?

To answer the first question, one has to remember that the number of neurons containing in the input and output layer is determined by the number of input data and labels respectively. In the case of Jason-2, the data set contains 104 bins which are then also the number of input neurons. Regarding the output layer, the whole waveform has to be labelled in peak or no peak. Hence, also the output layer contains 104 neurons.

The second question is more difficult to answer. In chapter 3.3 it is explained, that the number of hidden layers depends on the pattern of the data sets and how they can separate the classes. In Figures 3.6 and 3.8 examples are shown, how one can conclude the number of hidden layers by analysing the best way to separate the different classes. In the current case, there are two classes present, the leading edge and the rest of the waveform which is regarded as noise. Thus, the leading edge has to be separated from the rest of the waveform. Hence, one can use the example waveforms 4.1 to test, how the leading edge can be separated in the best way:

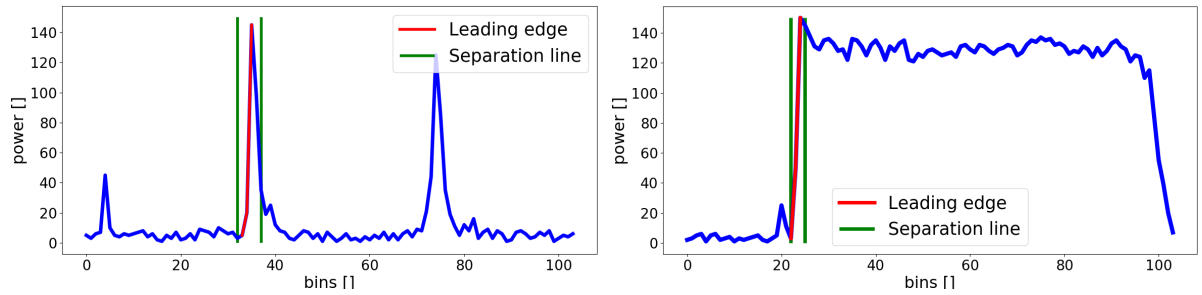


Figure 4.2: *Examples of separation of the leading edge*

As it can be seen in Figure 4.2 at least two lines are needed to separate the leading edge from the other areas of the waveform, which is equivalent to Figure 3.8. Thus, at least one hidden layer is needed in the neural network because the separation cannot be conducted with a linear function. The question, how many neurons are needed for the hidden layer is more difficult to answer, and some of the possibilities are mentioned in chapter 3.3. Regarding the rule of thumb, some possibilities will be tested during the implementation to find the optimal solution.

The last question can also be solved mostly by testing different combinations of learning rates and epochs. The risk of increasing the number of epochs is that the network is focusing too much on this data, which also influences the previously mentioned overfitting effect. If a waveform looked different compared to the training data, the neural network would face difficulties to estimate the correct peak. This problem has also to be taken into account when it comes to a decision about how many epochs should be chosen. To get more information about the learning process, one can use the information gained from the validation data. With this data, it is possible to calculate the so-called loss function which basically shows the amount of errors during the tests with the validation data, which is explained more detailed in the next section.

The loss function

During the training phase, the neural network is creating a functional model. Hence, one needs to find the minimum value of it which is done by using the gradient descent function. With the loss function, it is possible to visualise this function by walking it down which is helpful to see the performance of the network. It is even possible to literally see if one walks in the right direction, namely downwards, or if it is decreasing. To do so, a wide number of loss functions are developed the most prominent are (Natekin and Knoll, 2013):

Mean square error (L2 loss):

$$L2 = \frac{\sum_{i=1}^{n} (y_i - y_i^p)^2}{n} \tag{4.1}$$

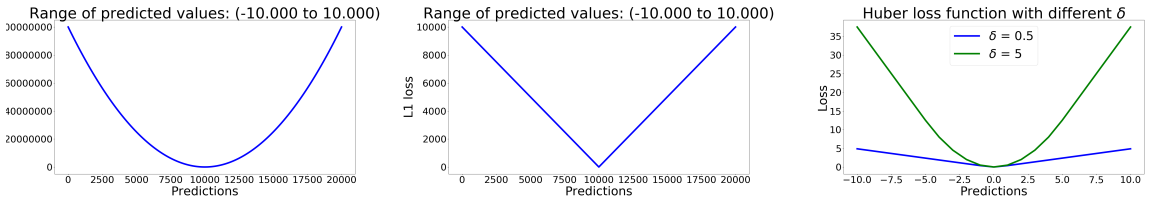
Mean absolute error (L1 loss)

$$L1 = \frac{\sum_{i=1}^{n} |y_i - y_i^p|}{n} \tag{4.2}$$

Huber loss function

$$L(y,f)_\delta = \begin{cases} \frac{1}{2}(y-f)^2, & |y-f| \leq \delta \\ \delta(|y-f| - \frac{\delta}{2}), & |y-f| > \delta \end{cases} \tag{4.3}$$

Where y_i^p and f are the estimated values and y_i and f are the true values. As heartbeat pointed out, the L1 loss function, which is shown in Figure 4.3 (a), is more robust against outliers but the derivation is not continuous. However, the L2 loss function, which can be seen in Figure 4.3 (b), is the whole time continuous which makes it more stable. The L2 loss also penalises big errors and neglect smaller errors (Natekin and Knoll, 2013). On the other hand, the Huber loss function is a parametrised loss function and includes the so-called cutting edge parameter δ (Natekin and Knoll, 2013) which is used to adjust the choice of the error. This function is plotted in Figure 4.3 (c). For smaller values, it is identical to the L2, and for bigger, it is similar to the L1 loss. In the following, plots of the functions versus the predictions can be seen:



(a) (b) (c)

Figure 4.3: *L2 (a), L1 (b) and Huber (c) loss function versus the predictions*

Because the amount of outliers during the validation process of the neural network is not known, the L1 loss will be used for the visualisation. In the next section, the neural network will be discussed. Thereby, also some time series will be shown to see the performance, which will be analysed in detail in chapter 5.

4.1.3 First test of the approach

Three data sets with different numbers of training data are prepared, to see the performance of the network also with low numbers of training data, which are shown in Table 4.1. To get a better understanding, how the parameters of the neural network are chosen, the first case with 600 training data, 100 training sets, will be shown in more detail. The basic structure of this neural network is discussed in section 4.1.2, which means that in both the input and the output layer 104 neurons are used. However, some parameters still have to be chosen:

- How many neurons are need for the hidden layer
- Which learning rate is needed
- How many epochs are needed to learn properly

To start with the rule of thumb, it can be said that the number of hidden neurons should be between the number of input and output neurons. Because they are the same in this case, one can set 100 hidden neurons for the beginning, which can be changed depending on the results. As a starting value for the learning rate 0.01 can be a proper value, and the number of epochs can start with 100. By using the loss function, it can be seen how the model behaves during the epochs. Hence, if the values are increasing or decreasing. With these parameters, the neural network can be run for a first test, which creates an accuracy for the test data of 73.9%. Whereas the accuracy *acc* is calculated by:

$$acc = \frac{n_{\text{true}}}{n} \tag{4.4}$$

with the number of correct retracked waveforms n_{true} and the total number of test data n, where the correct labels are determined by comparing the results with the labelled data. The loss function, which is calculated with the Formula (4.3) (L1 Loss function) is now computed for the validation data as well as the training data to see the differences:

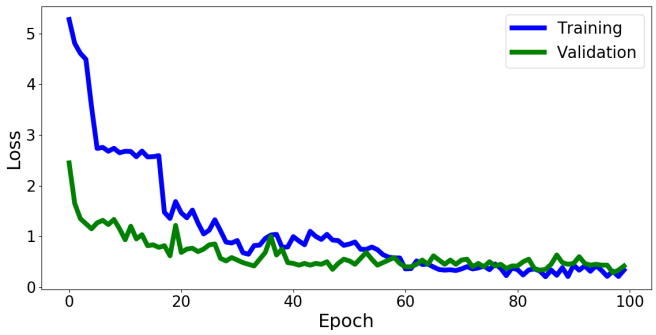


Figure 4.4: *L1 loss function for 100 test data*

In Figure 4.4 the errors decrease fast in the beginning, but after epoch 30, the level of decrease slows down. Regarding the validation data, it is after that epoch more or less constant while the loss function of the training data decreases a bit more. However, after some epochs, a saturation is reached, hence a change of the weights, as explained in Equation (3.16) and (3.16), has only limited improvements. But, it is interesting to see, that at the beginning the error of the training data is bigger than the error of the validation data. Besides the training data should be already known by the neural network and therefore be lower than the error of the, for the neural network unknown, validation data. So how can this change occur? The reason is that the loss function shows the amount of all errors, not how many errors occur. To make this more visible, a formula similar to the loss function can be used, which solely shows the number of errors:

$$errors_\gamma = \frac{1}{n}\sum_{i=1}^{n}\gamma_i \tag{4.5}$$

with

$$\gamma = \begin{cases} 1 & \text{correct labeled leading edge} \\ 0 & \text{wrong labeled leading edge} \end{cases} \tag{4.6}$$

By using this, the number of errors is shown as a value between zero and one, where n is the number of waveforms which were used for the test. Again one can apply this to the training data as well as to the validation data:

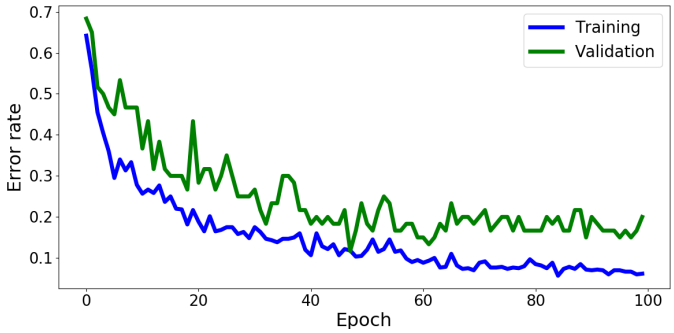


Figure 4.5: *Error rate for 100 test data*

As it can be seen in Figure 4.5, the number of errors is indeed higher for the validation data than for the training data. Thus one can conclude, that the training data has fewer but higher errors. It can be assumed, that the training data, which are 600 waveforms have a higher probability to contain significant errors than the validation data which are 60 waveforms. This problem can be compensated by taking both graphics, the loss function and the error rate, into account. One other question could occur, why the number of errors does not descent smooth but with peaks, which show that the network is even worse in a higher epoch. This can be answered by remembering the way how the weights are adjusted. Because the gradient descent is used, the overall minimum is searched inside a function, which represents the weights of the connection between the layers. Thereby, the starting point of the gradient descent algorithm changes in each epoch. Thus it can happen that only a local minima was found, which lead to a higher error compared to the previous epoch. Thus, the weights of the connections do not fit to the model and the network has problems to estimate the correct peak. However, by using more iterations, the number of errors decreases. As a result, one receives the position of the maximum peak which can be used to calculate the position of the leading edge. To do so, in a first step the DC bias level is computed which can be seen as bias by using 10 bins in front of the assumed foot point of maximum peak position bin_k, which is assumed 3 bins in front of it:

$$DC = \frac{1}{n} \sum_{i=3}^{13} \text{bin}_{k-i} \tag{4.7}$$

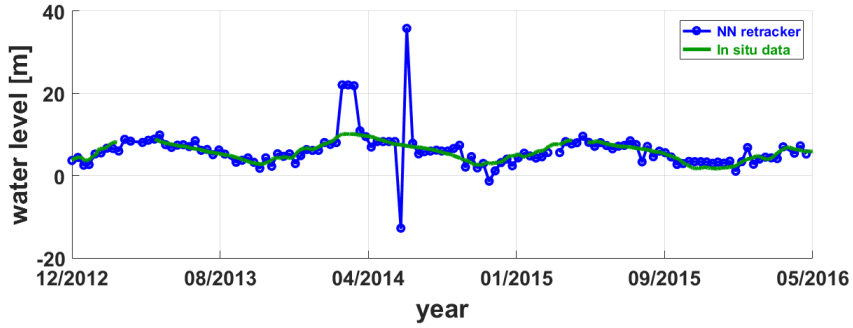
As it can be seen in Equation (4.7) the noise level is calculated from the 13th bin in front of the peak to the third peak in front of the peak position bin_k. Now, the power level P_m of the middle point bin_m on the leading edge can be estimated by using

$$P_m = \frac{P_k - DC}{2} \tag{4.8}$$

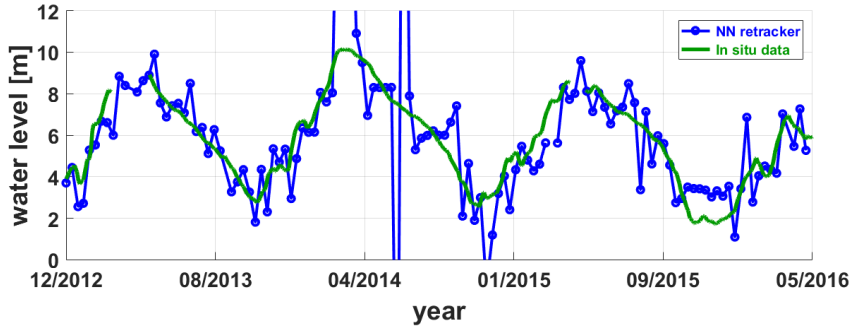
With P_m as the power level of the bin_m. It is then possible to interpolate the position of the leading edge with the following formula

$$\text{bin}_m = \text{bin}_k + \frac{\text{bin}_f - \text{bin}_k}{P_f - P_k} \cdot (P_m - P_k) \tag{4.9}$$

where bin_k is the bin at the footpoint of the leading edge, bin_f is the bin at the top of the leading edge and P_f and P_k is the power of the waveform at the position of the top and at the foot point of the leading edge respectively. Thus, after Equation (4.9) the position of the leading edge is calculated. Thus, it is now possible to use Equation (2.12) to calculate the water height time series. Because the aim of this study is, to show the performance of the neural network, no blunder detection or smoothing is conducted during the processing of the data. The calculated time series can be seen in Figure 4.6 (a). However, several huge outliers can be seen there. To make also smaller variations visible, the y-axis is limited in the next plot between 0 and 11 meters, which is shown in Figure 4.6 (b).



(a)



(b)

Figure 4.6: *Time series of the first test with 600 waveforms for training*

in Figure 4.6 (b), one can see that most of the time series matches quite well to the in situ time series. The main problem occurs in the first full seasonal peak in which all four main outliers occur. After that, the other seasonal peaks match well and only close to the end of the time series more outliers occur. By looking at the statistical data, one gets the following results:

Table 4.2: *Mean and RMSE of the first test of the neural network*

	NN [m]
Mean	1.49
RMSE	3.56

Regarding Table 4.1.3, it can be seen that the results are not very good and besides in some cases, the estimated time series match well with the in situ data, the several outliers worsen the result. This can be also explained if one has a look at the accuracy of the neural network which is only 75.8%. However, it could already be shown that the seasonal variations of the in situ data could be reproduced by using the neural network, besides the various outliers. Therefore, this is a promising result and will be improved in the following. However, before continuing with this, another point will be discussed more close. In Figure 4.4 and 4.5 it can be seen that the results are varying and even get worse, because of the gradient descent. To make this more visible, in the following the accuracy of the neural network with the current set of parameters is shown for multiple runs:

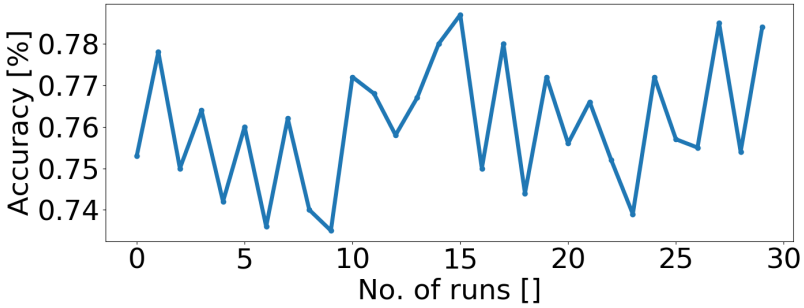


Figure 4.7: *Accuracy of the neural network with 30 runs*

As it can be seen in Figure 4.7, the accuracy is varying during the runs, because the final chosen global minima are different for every run. Therefore, it is recommended to run the neural network multiple times to find the optimal result. However, now we can think about how to improve the network parameters to improve the results. Therefore the network analysis methods like loss function, error rate and accuracy will be taken into account for the improvement. One can conclude, that if one increases the number of neurons in the hidden layer, there are more weights the neural network can adjust. This enables the network to learn more pattern and improve the results. However, the question is, which number of neurons in the hidden layer is the best one? Moreover, is it necessary to change also the other parameters? To answer this, the influence of the parameters will be discussed next.

A closer look at the parameters

Until now, only the loss and error curve was regarded, which give an indication of the changes during the learning process. In the next step, it will be focused on the reached accuracy by changing one parameter besides the other parameters remain the same. At first, the number of neurons in the hidden layer and learning rate will be varied, and the changes in the accuracy of

the neural network will be plotted. The resulting curves indicate if the network is improving or not. Beside this processing is time-consuming, it is the best way to find the ideal set of parameters:

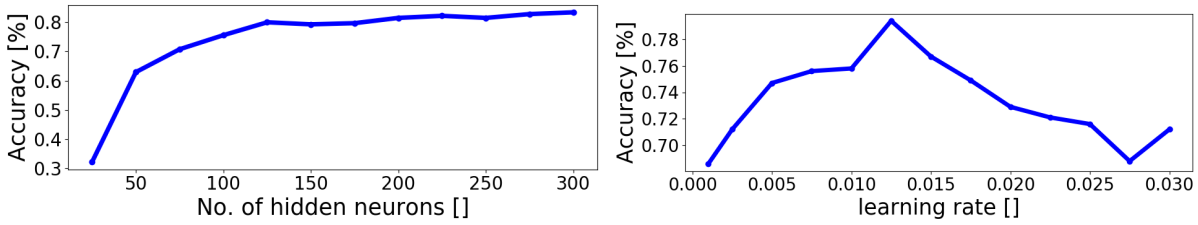


Figure 4.8: *Accuracy of the neural network with different numbers of hidden neurons (a) and learning rate (b)*

The variations of the two parameters are very interesting. As it can be seen in Figure A.1 (a), there is a big increase in the accuracy as the number of hidden neurons increase at the beginning. However, at a certain point, which is in this case around 200, a change in the number cause less increase in the accuracy. Which is also very important here is, that the number of hidden neurons has a significant influence on the neural network need to run during the training. Therefore, it is beneficial that there is an only limited improvement at a certain point. Thus, one can choose a lower number and still reach an appropriate accuracy with a usable runtime. However, in Figure A.1 (b) it can be seen, that there is a distinct value regarding the learning rate, at which the network has a maximum accuracy. This indicates, that the learning rate has to be increased until 0.0125 to improve the accuracy. Next, the number of epochs have to be chosen which will be analysed in the same manner. The resulting graph can be seen in the following plot:

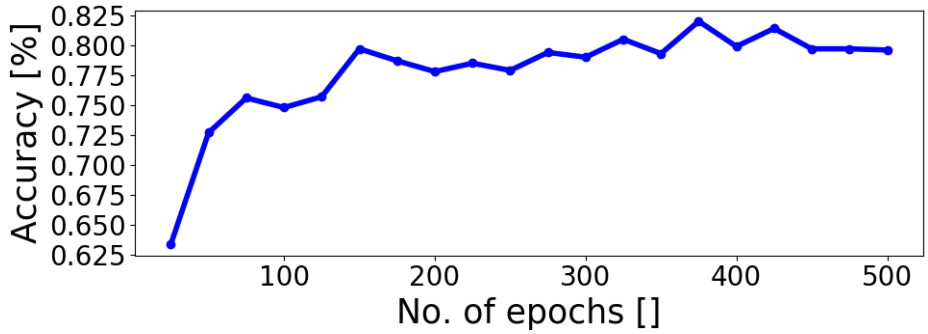


Figure 4.9: *Change in the accuracy depending on the number of epochs*

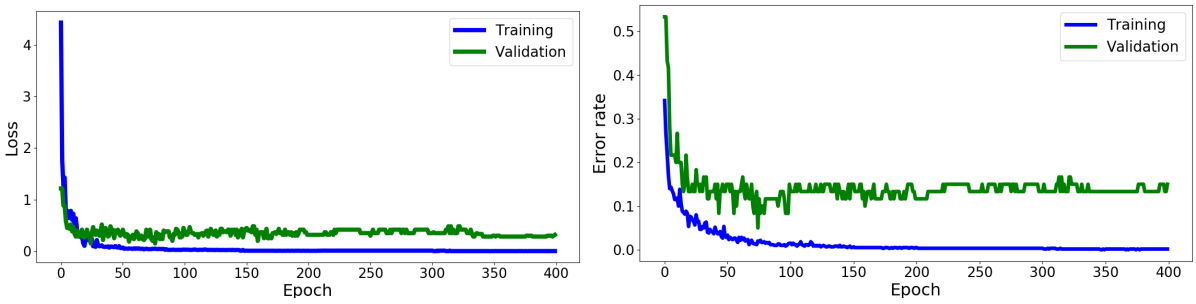
In Figure 4.9 the same stagnation of the increase of the accuracy can be seen, which also happened at the neurons of the hidden layer. In the beginning, one can see an increase in accuracy if one increases the number of epochs. But starting at epoch 350, the accuracy is only varying, but no significant increase can be seen. However, the effect of overfitting can occur if the number of epochs is too high. Also, the number of epochs has a high influence on the run time. Thus it is again a good effect, that the accuracy is not changing a lot after reaching a certain level. With this, it is now possible to improve the neural network parameters to reach the opti-

mal combination by maintaining a reasonable run time. In the following, the chosen parameter can be seen, compared with the second run:

Table 4.3: *Changes of the network parameters during the second correction*

Parameters	First run	Second run
Hidden neurons	100	250
Learning rate	0.01	0.0125
Epochs	100	400

By rerunning the network, it can be tested if there is an improvement in the results. By doing so, an accuracy of 81.9% is reached which is already an increase. As done before, also the loss function and error rate can be shown, to see, if there is also an improvement:



(a) (b)

Figure 4.10: *Loss function and error rate for the third run of the neural network*

It can be seen in Figure 4.10 (a), that the decrease in the loss function is very smooth, regarding the training data. However, also the validation data shows only smaller variations. The error rate in Figure 4.10 (b) shows a similar result, indicating that the training data decrease fast, as expected, whereas the validation data has more outliers. However, compared to the previous analysis the errors are at a lower percentage, which is also an improvement. Now, one can take a look at the time series, which is created by using the current neural network. As it can be seen in Figure 4.11 (a), there is indeed an improvement. The big outliers are now eliminated, and only the seasonal peak in 2016 is not captured well. Also there are now several matches between the retracked time series and the in situ time series visible. The better results can also be seen well if one regards the residual plot in Figure 4.11 (b).

As a comparison, one can have a look at the statistical data of the previous test and the actual test in the following table:

Table 4.4: *Mean and RMSE of the first test of the neural network*

	NN 2nd test [m]	NN 1st test [m]
Mean	0.95	1.49
RMSE	1.15	3.56

In Table 4.1.3, one can also see the improvement very well. Regarding the RMSE, an improvement of more than 2 meters can be seen, which is caused by the elimination of the significant outliers which could be seen in the first test in Figure 4.6 (a). Also, the mean value of the residuals shows an improvement of about 50 cm. It is also possible to visualise the residuals of this time series which can be seen in Figure 4.11 (b). It shows a very mixed result. In several cases, the neural network could retrack the leading edge well, and the errors are below 1 m. However, in other cases the leading edge is missed clearly, resulting in high errors.

But regarding the fact that this is the smallest number of training samples and it already matches suitably with the in situ data, it is a promising result. In the next step, the number of training data sets will be increased which lead to more possibilities for the neural network to learn and adjust the weights between the layer. Next, the datasets with 720 and 900 waveforms for training are used. Thereby, only the results are shown after the parameters are chosen. The decision, which parameters are used is shown in detail in A.1.

Comparison of waveforms after increasing the number training data

In the Appendix A.1 one can see, how the parameters are chosen. However, the used parameters can be seen in the following table:

Table 4.5: *Set of parameter which is used with 600, 720 and 900 training waveforms*

Parameters	600 waveforms	720 waveforms	900 waveforms
Hidden neurons	250	250	250
Learning rate	0.0125	0.0075	0.0075
Epochs	400	425	425

In Table 4.1.3 it can be seen that no big changes appear. The most significant change happens regarding the learning rate which changes from 0.0125 to 0.0075, which means that during the gradient descent the neural network uses smaller steps. Next, one can see the different time series which are created by the neural networks with the sets of parameters mentioned above. During the retracking, the neural networks reached an accuracy of 83.5% in case of the 720 training waveforms and 84.1% regarding the 900 waveforms, which both are only a bit higher than the previous test. With this we receive in Figure 4.12. As it can be seen in Figure 4.12 (a) and (b) the results are similar to the previous one. There are minor differences between the time series, but it can be said that the main differences, in which the retracked time series do not match with the in situ data are nearly the same. However, the differences become more evident by using the residuals in Figure 4.12 (c) and (d). There, some more significant errors in Figure (d) can be seen, but the number of significant outliers is smaller compared to the previous one in Figure 4.11 (b). Thus it can be said, that there is a small improvement, but still many outliers.

To have a better comparison between the three approaches, some statistics will be shown in the following table:

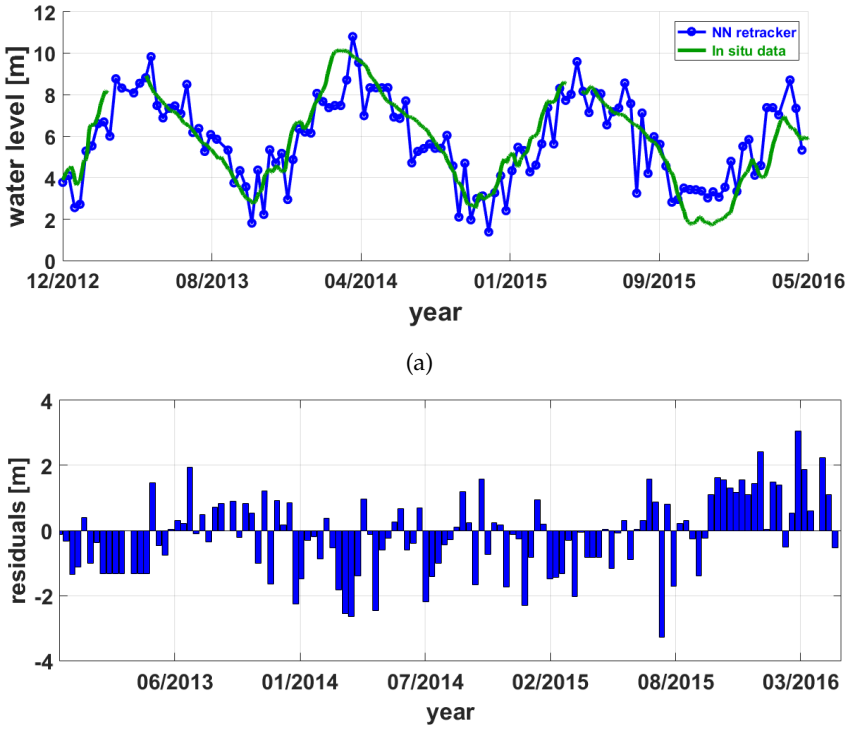


Figure 4.11: *Time series of the second test with 600 waveforms (a) for training and the residuals of it (b)*

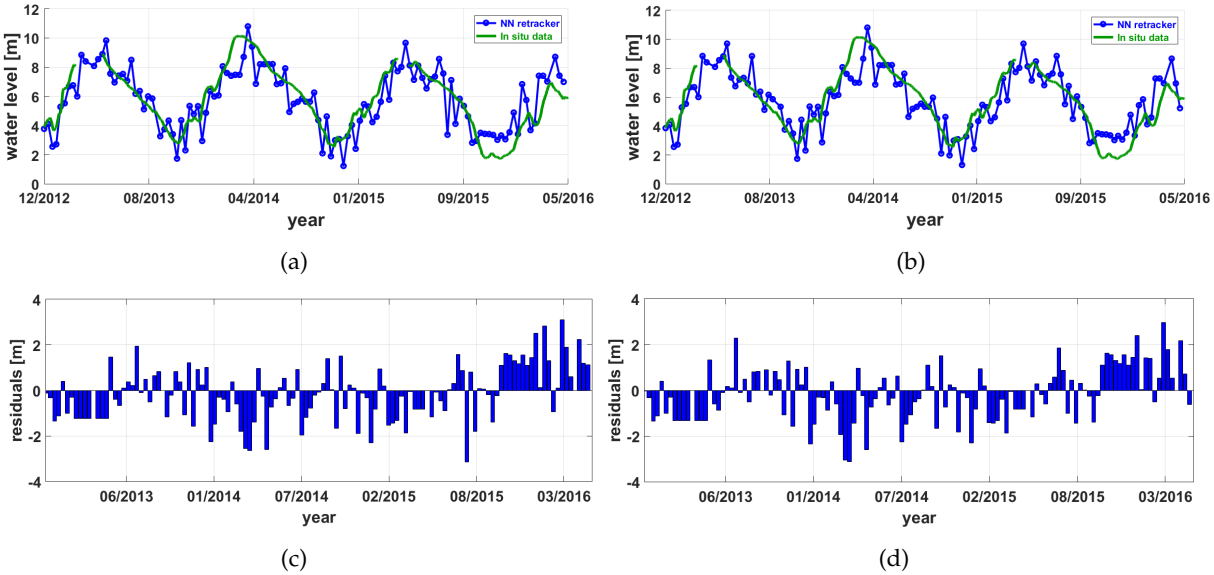


Figure 4.12: *Time series and residuals after the retracking which was done by a neural network with 720 and 900 training waveforms respectively*

Table 4.6: *Standard deviation, median and RMSE for tests with 600, 720 and 900 training waveforms*

	600 waveforms [m]	720 waveforms [m]	900 waveforms [m]
Mean	0.95	0.96	0.94
RMSE	1.15	1.17	1.14

In Table 4.1.3 only limited improvements can be seen. Interestingly the values get a bit worse between the first and second test and only improves in the third one. However, in all three cases only small variations can be seen which confirms the previous assumption. As the last comparison, the scatter plot is shown in which the retracked water levels are plotted against the in situ data.

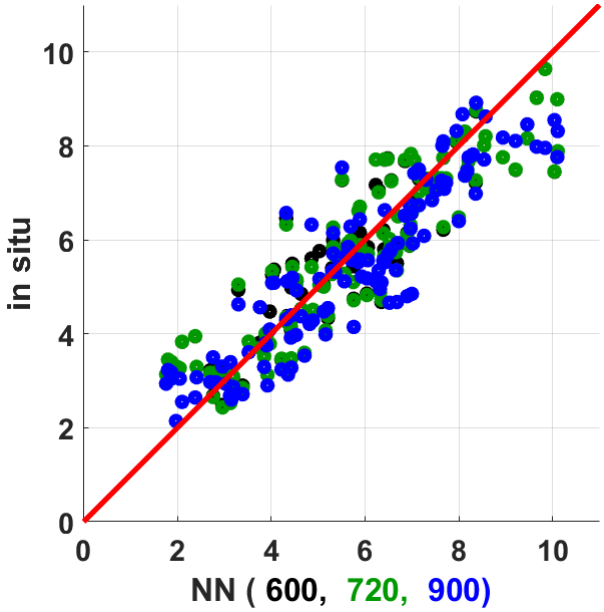


Figure 4.13: *Scatter plot of the retracked water hight with 600, 720 and 900 waveforms against the in situ data*

Interestingly in Figure 4.13 the plot of the 150 training sets looks more scattered and not as compact as the other two plots. This can be explained that it only covers the other plots and thus, only the third approach is visible, which is caused by the high similarity of the three data sets. However, it can be said that all three sets are localised close around the line and thus all show a good result.

As a conclusion, it can be said that the increase in training data does not improve the results noteworthy. Regarding the statistical values in Table 4.1.3 an improvement can be seen, but no significant one. On the other hand, the time series which were created matched to a high degree with the seasonal variations of the river. Thus it is possible to create useful data only by using a basic neural network. In the next step, a more advanced approach is applied by using the results of the neural network as input data for an algorithm which then detect the best peak in the waveforms.

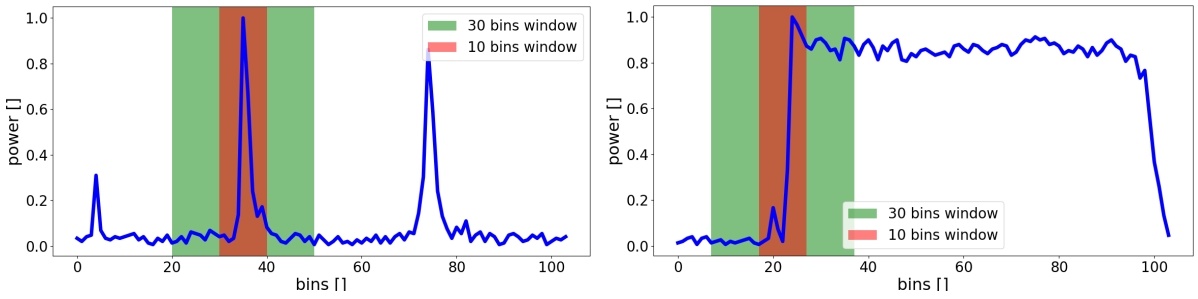
4.2 Subwaveform classification by using neural networks

Until now, it could be seen that there are only limited improvements if one uses the basic neural network and do retracking with it. Thus, a different approach will be tested for this task. Thereby, an algorithm is applied which uses the input data from a neural network to define the best peak of the waveform. Thus, not the whole waveform is regarded by the algorithm but only a smaller part of it, which is defined by a window. Thus, the algorithm is only analysing the pattern which is in the current field of view. In this algorithm, the output of the neural network is used, which gives the probability of the label, which is tested. This information is then used for the computations of the new retracking algorithm. In the first step, the algorithm is explained in detail and shown, how it can be used to find the best peak position for the leading edge. The second step concern the neural network which is used in this algorithm and how the data sets have to be prepared for it.

4.2.1 A new retracking algorithm

In this approach, one uses the fact that neural networks provide the probability of each label. The probability is calculated in this case now for a predefined window, which means that the neural network estimates which area has the highest probability to contain the leading edge. The needed steps are explained in the following:

1. Window size: In the first step, the size of the window has to be defined, which is then used to detect the best peak inside the waveform. Thereby, the window has to be big enough to include a characteristic pattern, but at the same time, it should be small enough, that the preferred peak is distinguished. With a total length of 104 bins, two different window sizes can be tested which are 10 and 30 bins. In the following figures both windows compared to each other are shown:



(a) (b)

Figure 4.14: *Examples of waveforms with the search window of 30 bins*

As it can be seen in Figure 4.14, the search window of 10 bins only covers the leading edge itself and a small area around it, whereas the 30 bin window covers a more significant area so that the neural network can also learn characteristic pattern around the peak. However, it can be tested if it is already enough to use a small area around the peak and, which would help to distinguish the best area. At the same time, it can be useful that a wider area is used to

improve the possibilities to learn the characteristic pattern. Thus, further studies will show the best option.

2. Step size: After the size of the window, also the size of the steps have to be defined. This step size is used to move the window along the waveform. In this case, the step size 1 is chosen to make sure, that the algorithm misses no part of the waveform.

3. Neural network step: After the previous parameter is determined, the first area inside the window at its first position is used as input data for the network. Thus, one will receive the label and the probability of the area which is analysed. These two values will be stored for further calculations. However, besides these two values, the maximum peak n_k inside the window is counted. Hence, also the number, how often the same peak is inside a window with the labelled peak is stored. By this, additional information can be provided, because it is assumed that the peak with the leading edge is labelled more often as peak then a smaller one.

4. Calculate the mean Probability $\bar{P}_k$: After going through the whole waveform with the window, the same peaks are tracked multiple times. Thus, it is now possible to calculate a mean probability $\bar{P}_k$ of each peak k:

$$\bar{P}_k = \frac{1}{n_k} \sum_{i}^{n_k} P_{k,i} \tag{4.10}$$

where $P_{k,i}$ is the probability of the peak k at the ith time it is tracked. After this step, the mean probability of each peak inside the waveform is known and as additional information, it is recorded, how often it is labelled as a peak.

5. Estimating the best peak: A small peak may be labelled with a high probability as a peak, and the peak with the leading edge is labelled with a lower probability as a peak. Thus, the smaller peak would be chosen for the retracking approach instead of the other one. This problem is tackled by the number, how often the peak is inside an area labelled as a peak. That means the mean probability $\bar{P}_k$ is multiplied with the number n_k, how often it is labelled, which can be seen as a weight factor. Therefore, not only the probability value is regarded, but also how often the network regard this area as a peak is included:

$$p_k = max(\bar{P}_k \cdot n_k) \tag{4.11}$$

where p_k is the position of the best peak according to the algorithm. After this step, the Formulas (4.7) until (4.9) are used to estimate the position of the leading edge. In the diagram in Figure 4.15, the discussed steps are shown, and it provides a better overview of the whole algorithm. In Section 4.1.1 it was discussed, how the data has to be prepared for the use in neural networks. Also in Section 4.1.1 it was shown, how the parameters of the neural networks can be chosen. However, in the case of this algorithm, some differences have to be regarded which are explained in the next section.

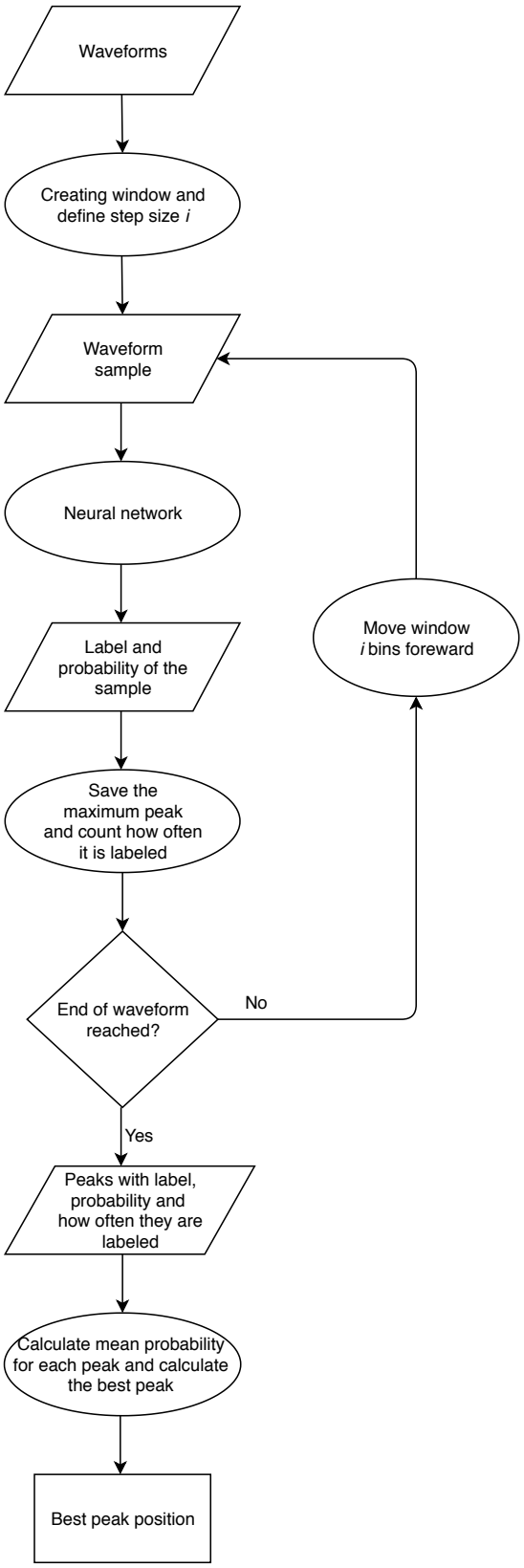


Figure 4.15: *Diagramm of the developed algorithm*

Preprocessing steps

Proprocessing the data: The main work is the same as explained for the network approach, which means that first the data has to be normalised and in a second step, the waveforms have to be labelled. The only difference now is, that not the whole waveform is used. Thus, only the area which is covered by the window is taken into account, which leads to the fact, that the whole area will get the same label. Therefore, two cases have to be distinguished:

1. Wanted peak is inside the window $\rightarrow$ label is peak
2. Wanted peak outside inside the window $\rightarrow$ label is noise

With this, one can now train and test the neural network. However, regarding the data, the previous analysis has to be regarded. There, three sets are used with 600, 720 and 900 waveforms for training. However, the improvement is limited, and the best results are reached with 900 waveforms. However, because now smaller areas for analyses are used, it could be possible to reach also good results also with a smaller number of training data. Hence, in the next analysis 600 and 900 waveform samples are used for the analysis with 10 and 30 bin input windows. With this, it will be possible to test all variations. Additionally, it can be seen, if the neural network can reach good results also with a smaller number of training data if the input area is smaller.

Network parameters: In a first step, the basic structure of the neural network can be defined. Thus, it can already be said, that the input layer contains 10 or 30 neurons respectively. The output layer only contains now two neurons, one representing the peak and the other no peak. This already leads to a much smaller neural network, compared to the previous ones. However, the other network parameter will be analysed in chapter A.2 as done before. In this chapter, the learning rate, the number of neurons of the hidden layer and the number of epochs will be defined. For the sake of convenience, these analyses are no further explained in this chapter, and only the resulting parameter is used.

4.2.2 Comparing the results with in situ data

Using 10 input neurons

The parameters, which are used in this chapter are estimated in Appendix A.2. Thus, in the following table are the parameters for the neural networks which use the 10 bin input window:

Table 4.7: *Neural network parameters for a network with 10 input neurons*

Parameter	600 waveforms	900 waveforms
Hidden neurons	5	10
Learning rate	0.005	0.0075
Epochs	55	10

After applying these parameters from Table 4.2.2 to the neural network, the time series in Figure 4.16 are gained. There it can be seen clearly, that the usage of 10 input neurons is not sufficient enough to capture the leading edges. In both cases either with 600 waveforms (a), (c) or with 900 waveforms (b) and (d), are significant differences between in situ water height and the calculated ones visible. However, in the second approach, an improvement can be seen, and the two graphs match better to each other.

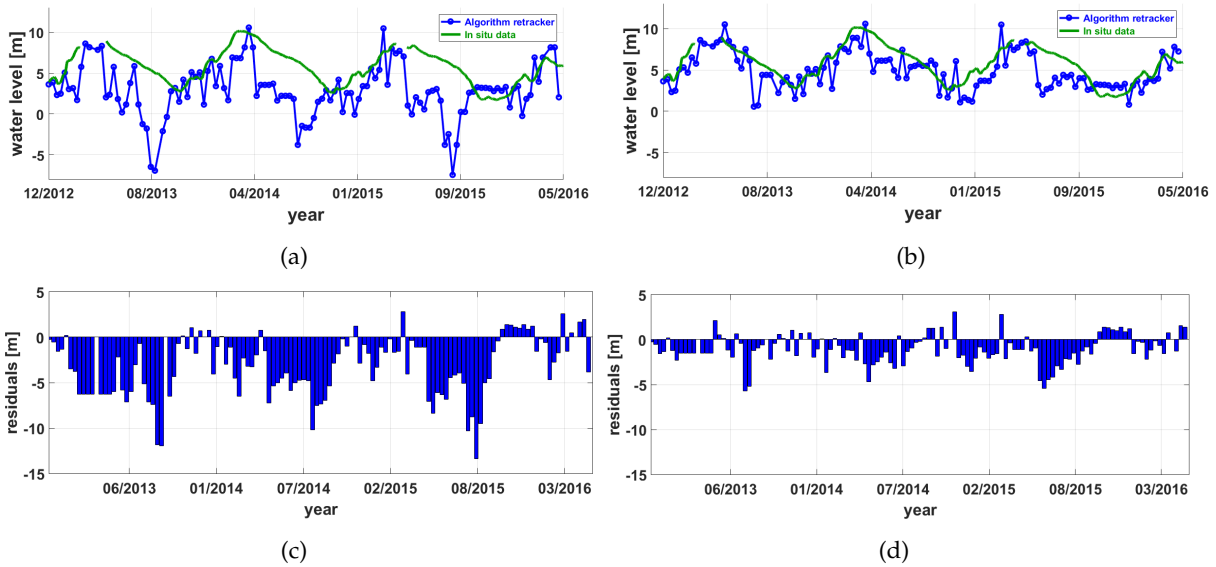


Figure 4.16: *Time series and residuals after using 10 neurons in the input layer and using 600 (a), (c) and 900 (b), (d) waveforms for training respectively*

A look at the statistics fosters this assumption. In Table 4.2.2 it can be seen that there is a big improvement by increasing the number of training data sets. However, even the improved results are worse compared to the results in Table 4.1.3. At least it could be shown here, how significant the influence of the increase of training data is. Thus, it can be concluded that the usage of 10 input neurons is not able to detect the wanted peaks. In the next step, one can have a closer look at the 30 input neurons approach.

Table 4.8: *The accuracy, mean and RMSE for tests with 600 and 900 training waveforms with 10 input neurons*

	600 waveforms	900 waveforms
Accuracy NN	72.4 %	87.8%
Mean	3.91 m	1.49 m
RMSE	4.79 m	1.84 m

Using 30 input neurons

Regarding the results during the analysis of the network parameters in Appendix A.2, an improvement in the maximum accuracy can be seen by using the 30 bin window. Thus, it can

now be tested if these improvements can also be seen in the created time series. The following sets of parameters are used:

Table 4.9: *Neural network parameters for a network with 30 input neurons*

Parameter	600 waveforms	900 waveforms
Hidden neurons	30	70
Learning rate	0.0575	0.0625
Epochs	70	95

In Table 4.2.2 one can see, that the parameters are changing more, compared to the previous approaches, which is also discussed in Appendix A.2. In Figure 4.17, one can see the resulting time series by using these sets of parameters. A first glance at Figure 4.17 shows, that the usage of 30 input neurons improved the results. It is interesting to see that the usage of 600 waveforms created better results than the usage of 900. Thereby, the created time series match to a very high percentage with the in situ time series. Only at the last seasonal peak, a mismatch can be seen, which also happens in the previous analyses. However, also the second approach by using 900 waveforms shows good results and the mean seasonal variations can also be captured there. Interestingly, the last seasonal peak is captured better in this approach than the previous one, but still contain many outliers. Thus it can be said that the promising results from the analysis of the parameters can also be seen in the final results.

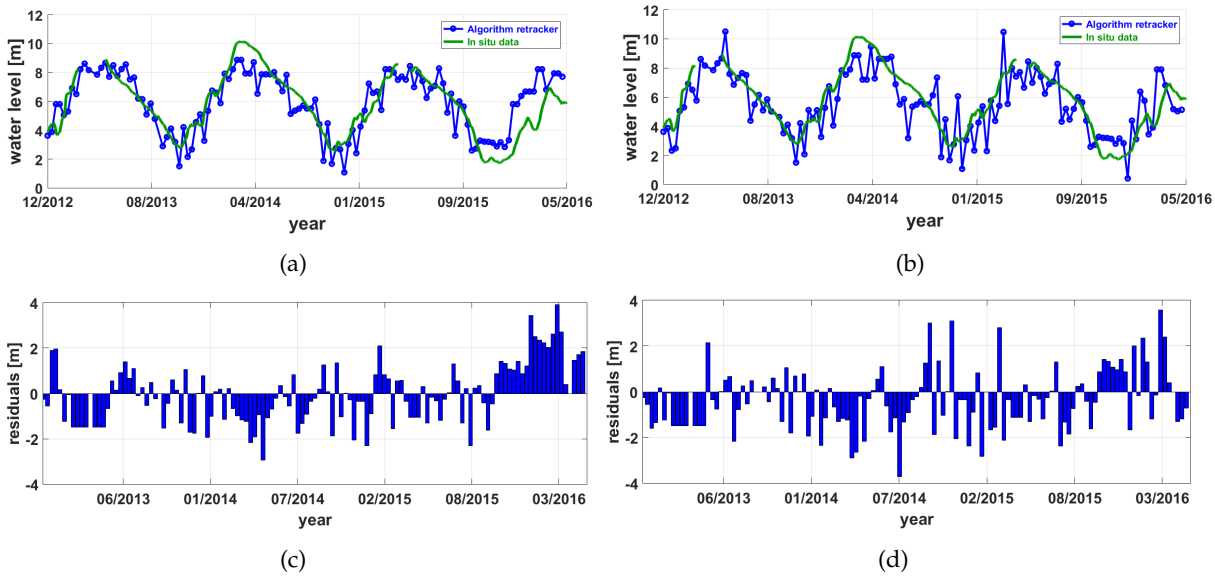


Figure 4.17: *Time series and residuals after using 30 neurons in the input layer and using 600 (a), (c) and 900 (b), (d) waveforms for training respectively*

Now, the statistics of the time series can be seen, which will finalise the analysis part of this chapter. The statistic values in Table 4.2.2 support the first assumption that the usage of 600 waveforms for the training shows a better result than the second one, but in both cases one can see only a slight improvement of a few centimetres. However, both show a good result which is a bit worse than the first approaches by using only the neural network.

Table 4.10: *The accuracy, mean and RMSE for tests with 600 and 900 training waveforms with 30 input neurons*

	600 waveforms	900 waveforms
Accuracy NN	97.1 %	97.2%
Mean	1.03 m	1.06 m
RMSE	1.26 m	1.3176 m

4.3 Conclusions

As a conclusion, it can be said, that two promising approaches are developed which can properly retrack time series. The first approach is a basic neural network which can retrack the waveforms to create usable time series. Therefore, it is easier to implement but needs more run time because of the bigger network structure. The algorithm is more sophisticated and thus, more difficult to implement. The main advantage is that it is faster because of the smaller network and it already provides time series which match well with the in situ data. However, the first attempt with the neural network shows better results in the comparison of the retracked time series and the in situ data. In the next chapter, the two approaches are applied to three study areas which were explained in the first chapter, to see, how they perform in different situations. Like the last test, the networks will be trained in one study area and be applied to another one, which is located inside the same basin. With this, one can see if the training data is transferable to other areas.

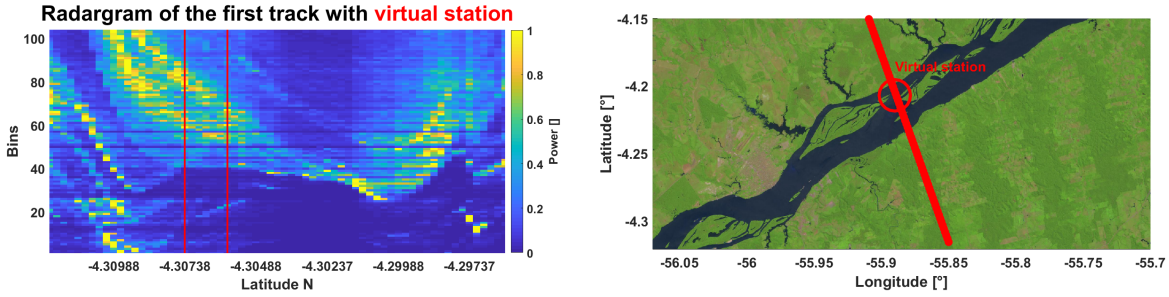
Chapter 5

Results

5.1 Analysing the results

5.1.1 The Cupari river

In this chapter, only the results regarding the 900 training waveforms will be shown. The reason is that with this number of data the best results can be reached. This network will now be applied to the Cupari river, which is also used in the previous chapter as an example. After the network, the algorithm mentioned in Section 4.2 will be applied to that river, with a detailed analysis of the results. However, before to start with the analysis, one can have a look at the so-called radargram in Figure 5.1 (a). As an orientation, an overview of the area is given in Figure 5.1 (b). By using the radargram, it is possible to get an idea, how the local situation affects the measured waveforms on which the estimated heights depend. There, one can see inside the virtual station in the horizontally, that the leading edge is well visible. This happens because the bins in the lower part of the plot contain mainly only low power, whereas the bins within the upper part of the plot more power contain. The light blue areas in the chosen area indicate that these waveforms contain noise at the beginning, as they have a higher power level. However, it can be assumed that the noisy area at the beginning, which can be seen until measurement number 12, is caused by the measurements over the land surface. Hence, in this area no clear leading edge is visible. In contrast to this, the area on the other side of the river shows still areas with clear leading edges but also often string noise in the lower bins. Also, areas with weaker signals than in the river area can be seen, which indicate reduced backscattered signals. This can be caused by the heterogeneous surface which does not reflect the transmitted signals adequately. As a conclusion, it can be said, that the chosen area shows good properties that both approaches can estimate the leading edge.



(a) (b)

Figure 5.1: *Radargram of the Cupari river (a) where the waveforms of one track are plotted along the y-axis, whereas the longitude of the satellite track is along the x-axis. The power level of the waveforms is indicated by the colour bar. An overview of the study area is shown in (b)*

Analysing the neural network approach

In the previous chapter, the estimated time series by the neural network is only compared to the in situ data. Now also the MLE3 onboard retracker and the threshold retracker will be included in the analysis, which are both explained in Chapter 2.2. Here it has to be said that the chosen threshold is 50% of the maximum peak, as for the neural network also the half of the leading edge is chosen. In Figure 5.2, the time series which are based on the three retracker are shown in comparison with the in situ data. In both plots in Figure 5.2 one can see that all three time series show very similar pattern. Regarding the comparison of the MLE3 retracker and the neural network in Figure 5.2 (a), more differences are visible. However, the the time series which is based on the neural network shows more matches with the in situ data as the time series, which is based on the MLE3 retracker. But it can be said, that all three time series show mismatches compared to the in situ data. In case of the comparison of the threshold retracker and the neural network, the network creates very similar results and they match in several cases, as it can be seen in Figure 5.2 (b). On the other side, the time series of the threshold retracker shows more significant outliers than the network approach.

The similarities of the three retracker can be shown even better if one regards the histogram of the residuals in Figure 5.3. In Figure 5.3 (a) the neural network dominates the residuals in case of the negative residuals whereas the MLE3 retracker dominates the residuals in case of the positive residuals. But the MLE3 retracker shows the highest amount of residuals overall and also more outliers. In the case of the comparison of the histogram of the neural network and the threshold retracker in Figure 5.3 (b), the outliers of the threshold retracker can be seen clearly. But the differences between the two histograms is smaller compared to the other comparison in Figure 5.3 (a). Thus it can be concluded that regarding the residuals, the neural network shows a better result in both cases.

This can be now verified by taking a look at the statistical detail in Table 5.1. Thus, all three time series show very similar results with only small variations. However, the MLE3 retracker shows a higher mean residual compared to the other two retrackers. But also in these cases, the variation is only within 1 decimeter. Regarding the threshold retracker the results are only slightly worse compared to the neural network. Thus, it can be concluded that the three approaches, in this case, are very comparable and the neural network shows only slightly better

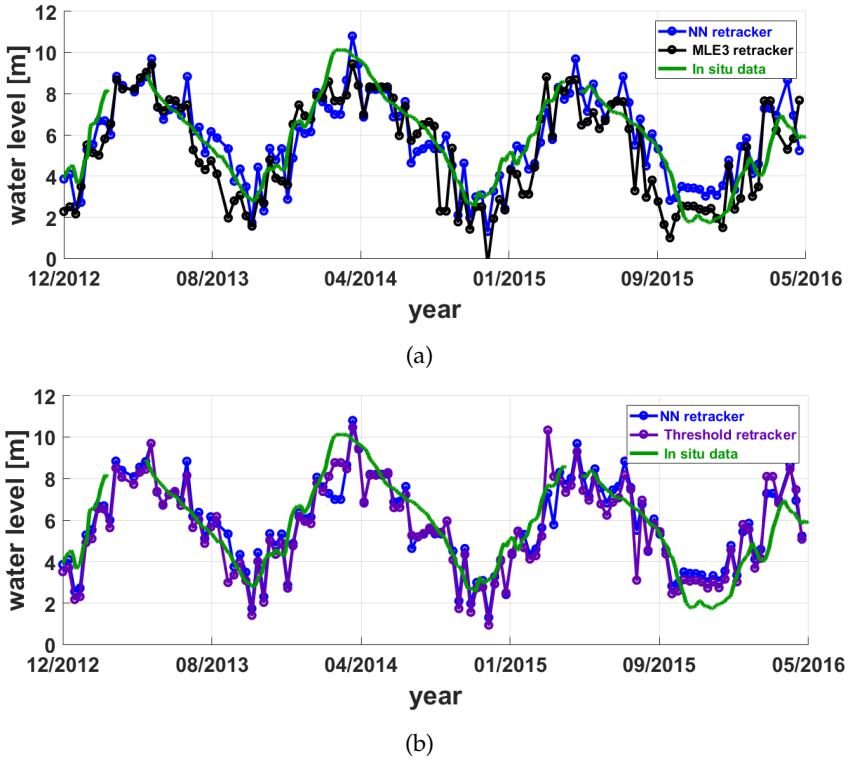


Figure 5.2: *Time series retracked by the neural network and the MLE3 retracker (a) and the threshold retracker (b) compared to the in situ data respectively by using 900 waveforms for training at the Cupari river*

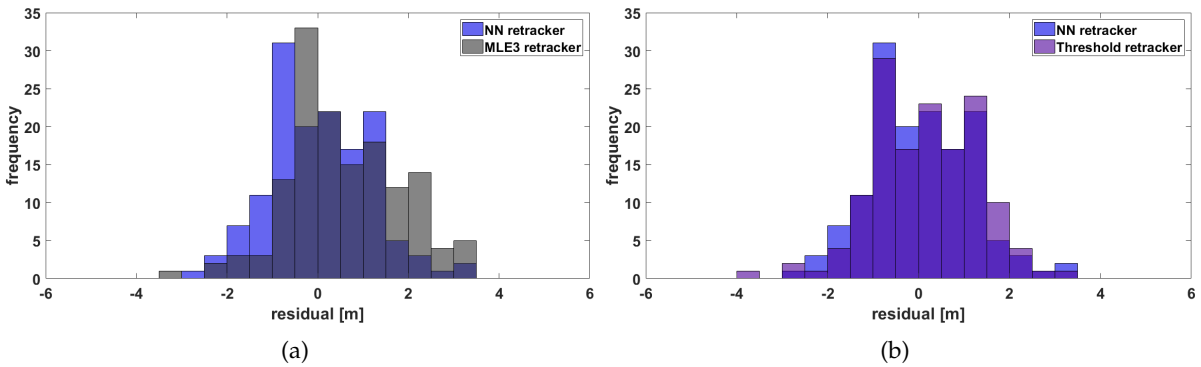


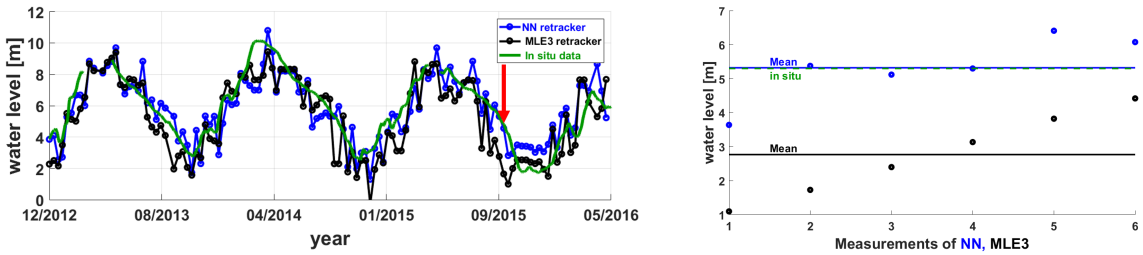
Figure 5.3: *Histogram of the residuals of the time series created by the neural network and the MLE3 retracker (a) and the threshold retracker (b) regarding the in situ data at the Cupari river*

performance. Next, the data sets of the neural network will be analysed more closely. The aim is, to understand why the neural network performs in some cases better than the MLE3 retracker and other cases not. Hence, two cases will be analysed and explained in detail.

Table 5.1: *Comparison of the MLE3 and the neural network approach with 900 trainings waveforms*

	NN [m]	MLE3 [m]	Threshold [m]
Mean	0.94	1.06	0.95
RMSE	1.14	1.14	1.17

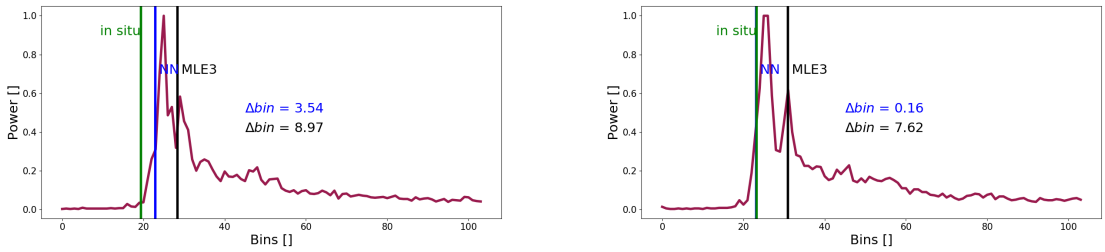
First analysis: For the beginning, one will take a look at a situation, in which the neural network matches well with the in situ data, and the MLE3 retracker shows a worse performance. For a better understanding, it is necessary to know, that each point in the time series is the mean of 6 measurements. To get an idea, how this mean value is created, all six retracked water levels from the neural network and the MLE3 retracker will be plotted. Additionally as an orientation, also the mean values are shown as a horizontal line together with the in situ value. Thus, one gets an idea how good the height could be estimated:



(a) (b)

Figure 5.4: *Chosen area in the time series and also the detailed plot of the results*

In Figure 5.4 (a) it can be seen, that the neural network in the descending part of the seasonal peak has a much better performance than the MLE3 retracker. A look at the detailed plot in 5.4 (b) shows that the retracked water height of the network is in two cases nearly exact on the correct position and one time very close. However, three times it misses the correct water level, but because the mean value is used, the final water level shows a good result. A look at the MLE3 retracker shows that the estimated water height is increasing and it is getting closer to the in situ water height. The most significant outlier of the MLE3 retracker is more than 2 meters, whereas the biggest one of the network approach is 1.66 m, which is still a high value. However, regarding the three significant outliers, a mean residual of the network regarding the in situ data with 0.6336 is still a good value. Though, the question remains, which effect created the significant outliers. To answer this, some waveforms will be analysed to see at which position the neural network and the MLE3 retracker estimated the leading edges. This will be shown in the first two waveforms, which are interesting examples. In the first one, both algorithms not found the correct position, and in the second one, the network found it:

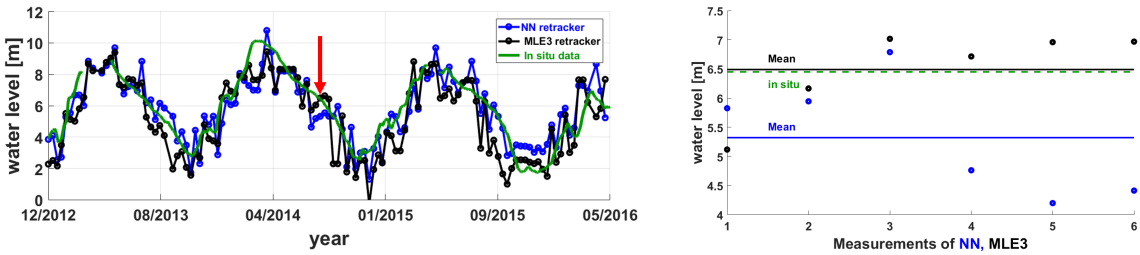


(a) (b)

Figure 5.5: *The first two waveforms and the retracked leading edges of the data set*

In Figure 5.5 (a) it can be seen, that multiple peaks in the waveform cause the differences in the estimated water levels. In this situation, both retracking algorithms estimated the leading edge at another position. The network approach estimated the leading edge at the first significant peak and missed the right leading edge for some bins. Because one bin is equivalent to 0.4684 m, even a small distance in bins results in a significant distance in meters. However, the MLE3 retracker estimated it at the second peak which is far more away from the assumed leading edge position, which results in the significant outlier. This example shows well, how difficult it to estimate the correct position, even more in case of a multi-peak situation. In the second waveform, which can be seen in Figure 5.5 (b), one can see a multi-peak situation again. However, this time, the network estimated the leading edge at the correct position, but the MLE3 retracker is still estimating the leading edge at the second peak. Here it has to be mention, that there are indeed situations in which the second peak contains the right leading edge, but in this case, it is very likely.

Second analysis: An interesting situation will be analysed now, in which the neural network performs not very well, but the MLE3 retracker shows good results. As done in the previous case, an overview of the estimated water heights will be shown first. Again, also the mean values with the in situ value will be shown:



(a) (b)

Figure 5.6: *Chosen area in the time series and also the detailed plot of the results with good performance of MLE3*

A look at Figure 5.6 (a) shows that the MLE3 retracker has a better performance whereas the neural network approach estimates a lower water height. However, a look at the plot 5.6 (b), reveal that both retracker have difficulties in estimating the correct position of the leading edge. Though, the MLE3 retracker is in most cases closer to the in situ water height than the

neural network. Interestingly, in this case, one can see more significant distances between the estimated water height and the in situ height, which is in contrast to the previous case. However, the MLE3 retracker is in most cases closer to the correct height. It is also to mention, that the estimated water heights of the neural network vary more, compared to the heights estimated by the MLE3 retracker. This indicates that the approach has difficulties in estimating the correct leading edge position. To see the reason for this behaviour, one can now have a look at the waveforms itself, which can be seen in Figure 5.7.

As it can be seen in 5.7 (a), both retracker estimated the leading edge at the first prominent peak of the waveform, but as before, the leading edge can be assumed more at the beginning of the peak. Because of the fact, that the true position can be found at the small increase at the beginning, both approaches fail to estimate the correct water height. This can be caused by an error of the onboard retracker, which placed the waveform at a wrong position. As explained in chapter 2, the onboard retracker is adjusting the waveform and keep it inside the window. However, it may fail to place the leading edge at the correct position. Thus, it is not possible for both retracker to estimate the right position. That is also the reason why both retracker have a significant distance to the correct position. In Figure 5.7 (b), the MLE3 retracker is a bit closer to the small edge than the result of the network. However, both get closer to the right position which is then reached in the third waveform in Figure 5.7 (c). There, both approaches are at this first small edge, so both have good results with only small variations. However, the MLE3 retracker went a bit too far, so the network is now closer to the leading edge, which could also be seen in 5.6 (b). In the fourth waveform in Figure 5.7 (d), there is again a multi-peak situation, which is also the reason why the neural network estimates the leading edge at the wrong position. In the previous analysis, the second peak was lower, and the MLE3 retracker estimated the leading edge there. In this current case, the second peak is higher than the first one, and this time, the neural network estimates the leading edge at this position. However, it can be seen clearly that the correct position of the leading edge is still at the first peak of the waveform. Thus it could be shown, that these four examples are challenging for both retracker and now it can be explained, why there are more variations inside the estimated water heights.

As a first conclusion, it can be said that the neural network approach shows already good results compared to the MLE3 retracker. Next, the developed algorithm will be applied and compare it with the onboard retracker.

Analysing the algorithm approach

As described in the previous chapter, two different window sizes are tested for the developed algorithm. However, the first test with ten input neurons had bad results, and it was not possible to estimate the water level accurately. The approach with 30 input neurons showed better results and the created time series matched well with the in situ data. Hence, only the second approach will be used in further study. Also during the test, two different results could be seen. In most cases the 900 waveform training data showed better results than the 600 waveform data set. However, in the test with the 30 input neuron window, the 600 data set performed better. For the sake of clarity, only the 900 waveform training set will be analysed in this chapter, because of the better results in all other cases.

Thus in Figure 5.8, the time series which is based on the algorithm is compared with the time series of the MLE3 and threshold retrackers. In Figure 5.8 (a), it can be seen, that the results of the algorithm approach show a worse result compared to the MLE3 retracker as there are much more outliers. Interestingly, the comparison with the threshold retracker in Figure 5.8 (b) shows more similarities as some of the outlier match now which each other. However, in this case, the time series which is retracked by the algorithm shows several significant outliers, which can not be seen at the time series, which is based on the threshold retracker.

To get a better idea about the residuals with respect to the in situ data, the histograms are shown in Figure 5.9. With this, the previous assumption can be verified, that the time series created by the algorithm, has more outliers. With this, it can be shown visually, that the algorithm performs worse in both cases, as it has more often a higher number of the residuals, compared to the previous test. Thereby, the algorithm has now the higher residuals which are mostly located in the positive direction. In case of the threshold retracker in Figure 5.9 (b), the two approaches show a mixed result as they both have a high amount of residuals, which shows that both of them do not perform well.

At least, some statistical data is shown, to get a better comparison between the three approaches. In this table are also the results of the neural network approach displayed for a better comparison:

Table 5.2: *Comparison of the algorithm approach with 900 trainings waveforms and 30 bin input data, the MLE3 and threshold retracker as well as the previous test with a neural network and 900 training waveforms*

	Algorithm [m]	MLE3 [m]	Threshold [m]	NN [m]
Mean	1.03	1.06	0.95	0.94
RMSE	1.31	1.14	1.15	1.14

Table 5.2 shows a mixed result as it is compared to the MLE3 retracker roughly three centimeters better in the mean of the residuals. Regarding the RMSE, the algorithm is nearly 15 cm worse. In opposite, the threshold retracke shows in all cases better results, as they are very comparable to the previous approach. At the same time, a better result can be seen for the neural network also in all cases. As done before, one can have a closer look at some cases, to see the strength and the weakness of the developed algorithm. Thus, in the first case, a situation will be analysed, in which the algorithm performs a bit better than the previous attempt. Therefore, the autumn of 2013 will be shown, where the algorithm is a bit below the in situ data, but still very close. In the previous test, the water hight of the neural network was clear above the correct value. Hence, it will be interesting to see, what changes there during the retracking. To see these differences better, besides the estimated water heights, also the mean value of the neural network, will be shown in Figure 5.10.

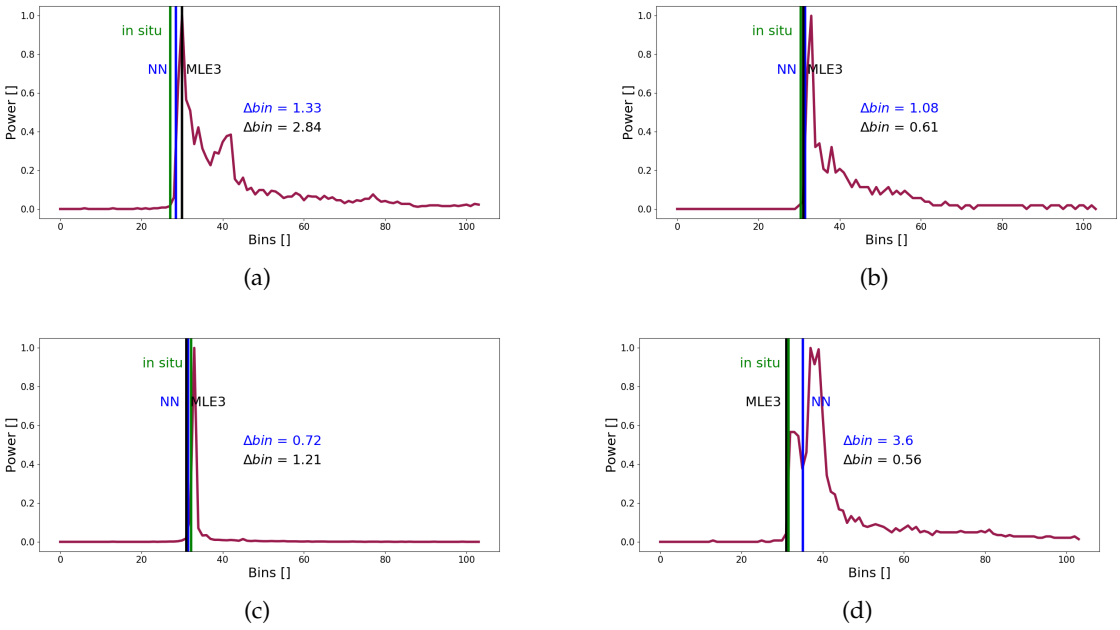


Figure 5.7: *The waveforms and the retracked leading edges of the data set*

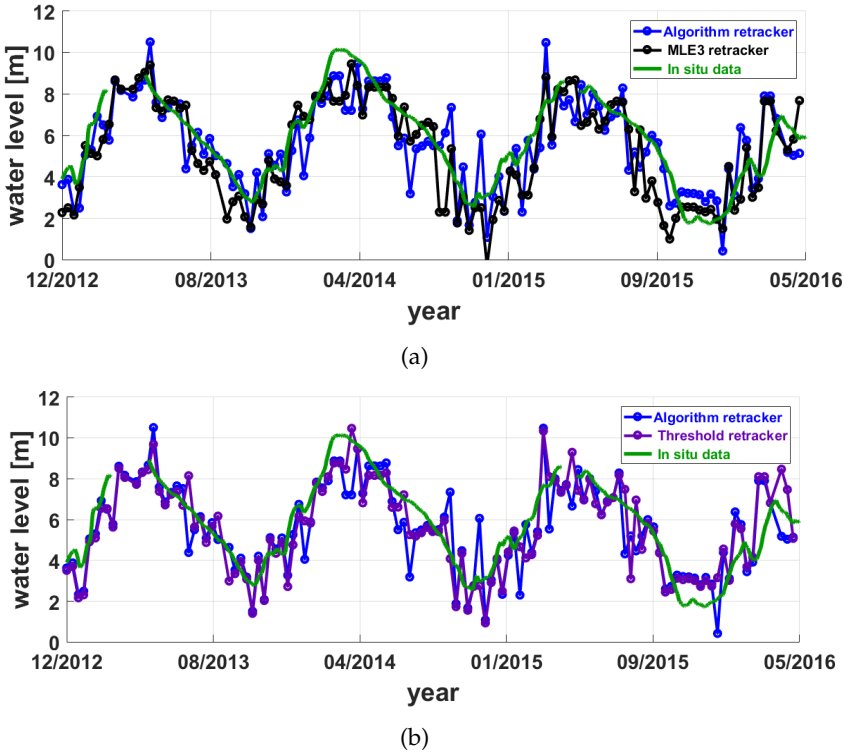
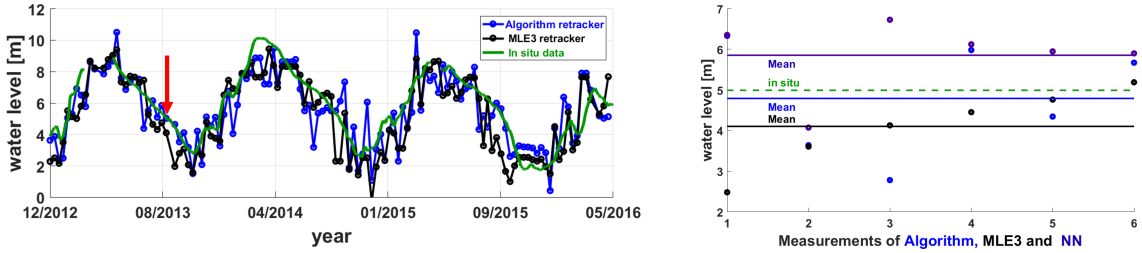


Figure 5.8: *Time series, retracked by the algorithm and by the MLE3 retracker (a) and by the threshold retracker (b) compared with the in situ data respectively at the Cupari river*



(a) (b)

Figure 5.10: *Chosen area in the time series and the detailed plot of the results of algorithm, MLE3 and the network in comparison*

As it can be seen in Figure 5.10 (b), at the first waveform, both developed approaches estimated the same water level. However, after that only at the fourth waveform both retracker show nearly the same result. Interestingly, in the second case, the algorithm and the MLE3 retracker estimated the same water level whereas the network is some centimeters above them. However, the two most striking differences appear in waveform three and five, which cause the difference between the mean values of the two attempts. However, it is interesting, that only the MLE3 retracker is two times close to the in situ data. Thus, it will be interesting to see, what caused the different retracking positions and in which situations are they similar.

Thus, the first four cases are analysed, which can be seen in 5.11. In this case, on the left side, the algorithm approach can be seen and on the right side the neural network as a comparison. In the first retracked leading edge in Figure 5.11 (a) and (b), both developed retracking algorithm chose a peak which is smaller than the correct one and not that distinct, thus it is interesting that both estimated the leading edge at this peak. The MLE3 retracker performs worse in this situation since it chose a tiny peak, which is in a more significant distance to the correct position. Looking at the second leading edge in Figure 5.11 (c) and (d), the network approach performs a bit better, whereas the algorithm and the MLE3 retracker estimate the leading edge at the same position, which is a bit more away from the leading edge. This case is also a difficult situation for the retracker, as there is a multi-peak situation and both are close to each other. However, it can be seen, that the network approach shows, in this case, a slightly better performance. A look at the next waveform in 5.11 (e) and (f) shows that there could be probably again an error of the onboard retracker because the correct leading edge position is located at a small peak behind a high and distinct peak. However, the MLE3 retracker estimates the leading edge at a position on the descending side of the true peak. In contrast, the neural network estimates the leading edge at the distinct peak whereas the algorithm estimates it at a small peak behind the correct one. One possibility for the performance of the algorithm is that this peak is more often labelled as peak as another one, thus that it has the highest probability for the algorithm to be the searched position. In the last example in Figure 5.11 (g) and (h) a similar situation can be seen in which the correct leading edge position is located at a smaller peak behind the distinct peak. This could also be caused by an error of the onboard retracker which could not fix the leading edge. This time, both developed algorithms estimate the position of the leading edge at the first distinct peak. The MLE3 retracker estimate again the leading edge at the descending part of the smaller correct peak. Thus also, in this case, the MLE3 retracker has the best performance. The tiny better result of the algorithm can be explained since it put

the leading edge a bit higher on the ascending part of the peak, which brings it closer to the real solution.

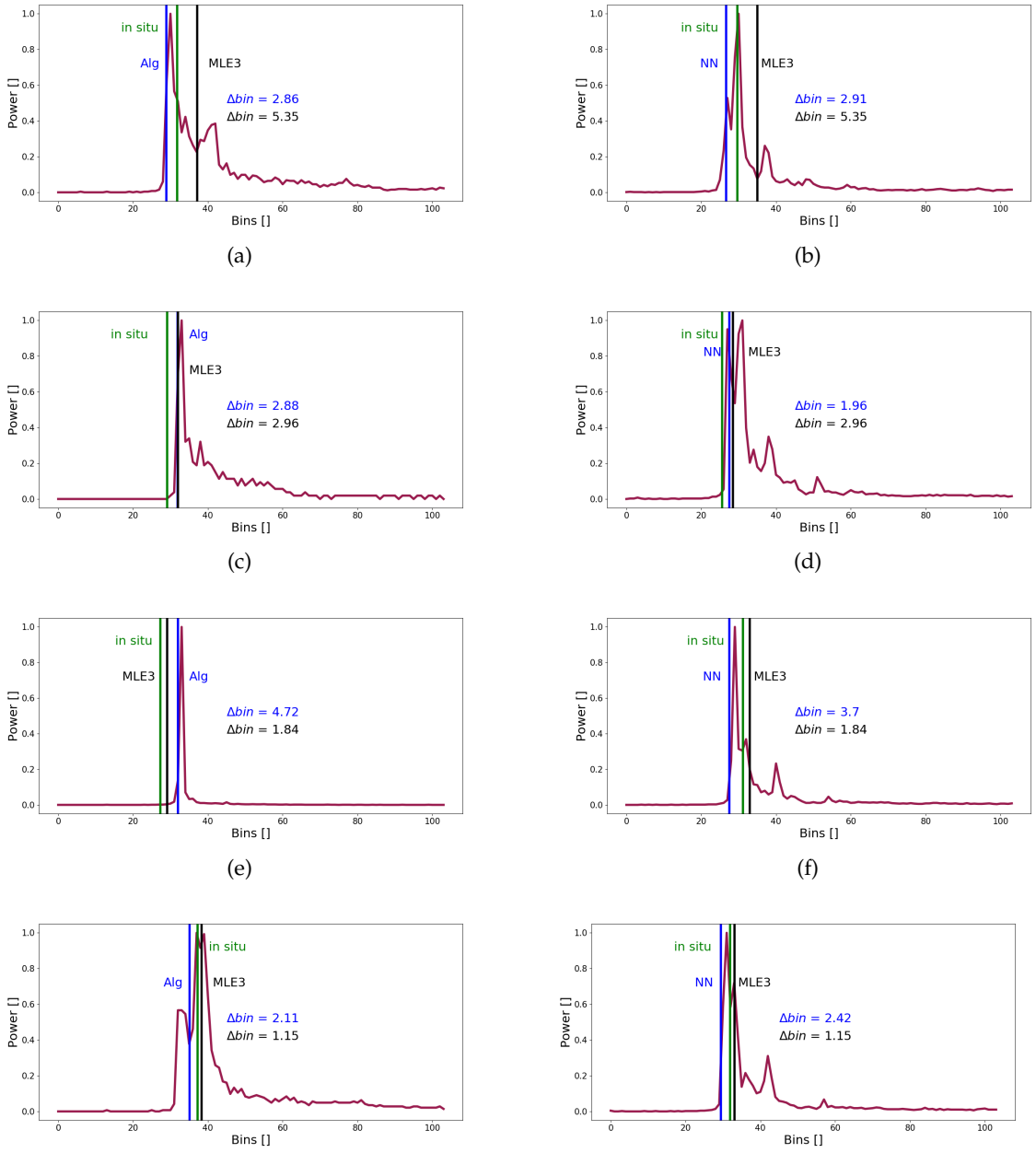


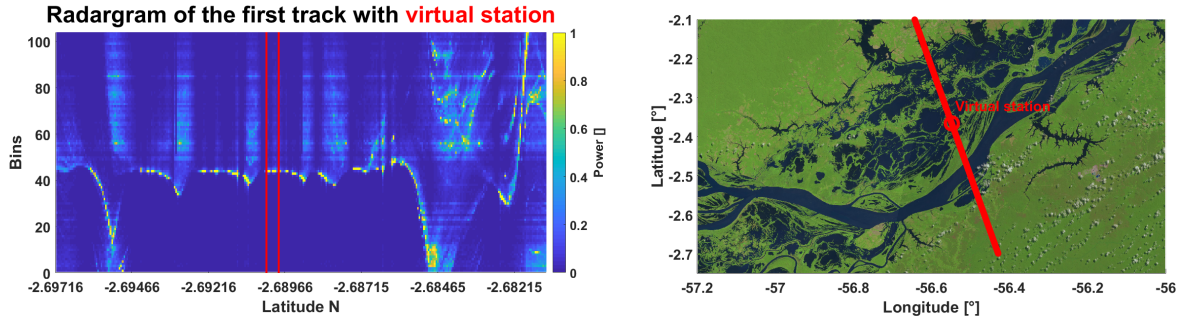
Figure 5.11: *The waveforms and the retracked leading edges of the algorithm (left) compared with the network result (right)*

Previously this case study intended to show the strength of the algorithm, but it also showed the weakness of it. Regarding 5.10 (b) it seemed that the algorithm outperforms the neural network in this case. However, by regarding the analysis, it could be shown, that the outlier in the third waveform moved the mean closer to the correct in situ value. In other cases, both

showed similar results. Thus it can already be said that the algorithm itself does not outperform the network in this example. Next, the results of the second case study, the floodplain Lago do Madabá Grande, will be analysed. As already mentioned, it is also located inside the Amazon river basin. This study area is later also used to train the network which is then applied to the test data set of the Cupari river.

5.1.2 Lago do Madabá Grande

As mentioned in chapter 1, this floodplain is located alongside the Amazon river, which is now tested by the developed approaches. To do so, first, the neural network approach with 900 training data sets will be applied. After that, the algorithm with the 30 input neuron will be tested. As done before, also the radargram can be used at the beginning to get an idea about the backscatter of this water area. Also an overview of the analysed area is given:



(a) (b)

Figure 5.12: *Radargram of the track with the chosen area (a) and an overview of the area (b) at Lago do Madabá Grande*

It is interesting to see in graph 5.12 (a) that there is less noise than expected. However, it can also be seen that there are curved lines with high power in the middle of the radargram, which could be caused by the land areas between the water bodies. However, it can be seen, that there is a distinct line between the bins with a lower number and the high number of bins. This leads to the conclusion that the waveforms can be tracked very well since a clear leading edge is visible. In opposite to areas in the middle of the radargram, land areas can be seen well, mostly at the right side of it which shows a very mixed area without a clear leading edge. Thus, it can be said, that the small river islands have no significant influence on the waveform structure. Since there is a high variation in the floodplain, the results can change a lot during the year. Thus it can happen that in case of less water during the summertime, the noise increase and thus no clear leading edge occurs.

The network approach

In this case, the same network parameters are chosen which are already mentioned in Table 4.1.3. With this, a performance of the neural network of 93.6% is reached, which is much higher compared to the previous one with 84.1%. Thus, it can be assumed, that nearly all waveforms could be retracked well. So it will be interesting to see, how the resulting time series will look

like. Unfortunately, there is no in situ data available for this region, but the DAHITI database (Schwatke et al., 2015) provides time series for this region. Because DAHITI is used to validate the results, it is not possible to go into detail, because the correct position of the leading edge is unknown. However, one can compare the results of the estimated time series with the one from DAHITI to have a comparison. Beside the neural network, also the MLE3 and threshold retracker will be applied to see the differences between these approaches. Like before, also in this comparison a 50% threshold is chosen for the threshold retracker. The resulting time series are shown in Figure 5.13.

The results in Figure 5.13 (a) are surprising, because even at first glance it can be seen, that the MLE3 retracker shows a high bias of nearly 10 m. In contrast, the neural network shows a better result and fit in several cases to the DAHITI time series. However, also one can see in this case several outliers which are up to several meters. This can also be seen regarding Figure 5.13 (b). However, all three time series show a very similar pattern, besides the MLE3 retracker shows the bias. But by comparing the neural network with the threshold retracker, one can see that the results are very mixed and both show several significant outliers. Regarding the time series as a whole, it can be seen that the most positive outliers occur during the descending of the seasonal peak during the autumn and winter. Figures 1.5, show, that during this time period the amount of water is decreasing. Thus, the radar signals are reflected more by land which results in noisy data sets which are difficult to be retracked. It is interesting that the time series of the MLE3 retracker shows indeed often the same variations than the network, but the offset can be seen during the whole time series.

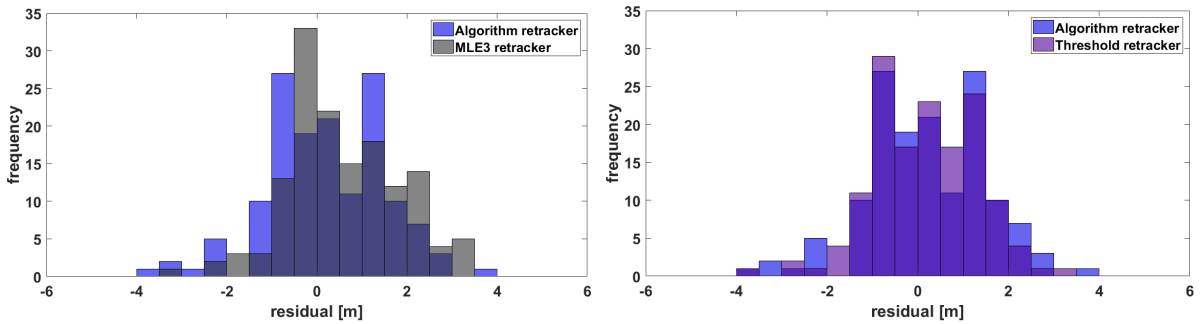
Next, one can see the histograms of the three approaches which show more the amount of residuals, compared to the DAHITI data. Regarding the histogram in Figure 5.14 (a), one can see clearly the bias between the two retracker, but regarding the amount of residuals, it seems that the neural network shows a slightly better result than the MLE3 retracker. In case of the comparison of the neural network and the threshold retracker in Figure 5.14 (b), one can see that the threshold retracker shows the highest amount of residuals, which was not clearly seen in Figure 5.13 (b). Thus, it is interesting to see that the neural network shows a better performance regarding the amount of residuals.

However, one can have a look at the statistical data to get a better impression about the variations of the three time series, compared to the reference time series of DAHITI:

Table 5.3: *Comparison of the neural network, MLE3 and threshold retracker at the Lago do Madabá Grande*

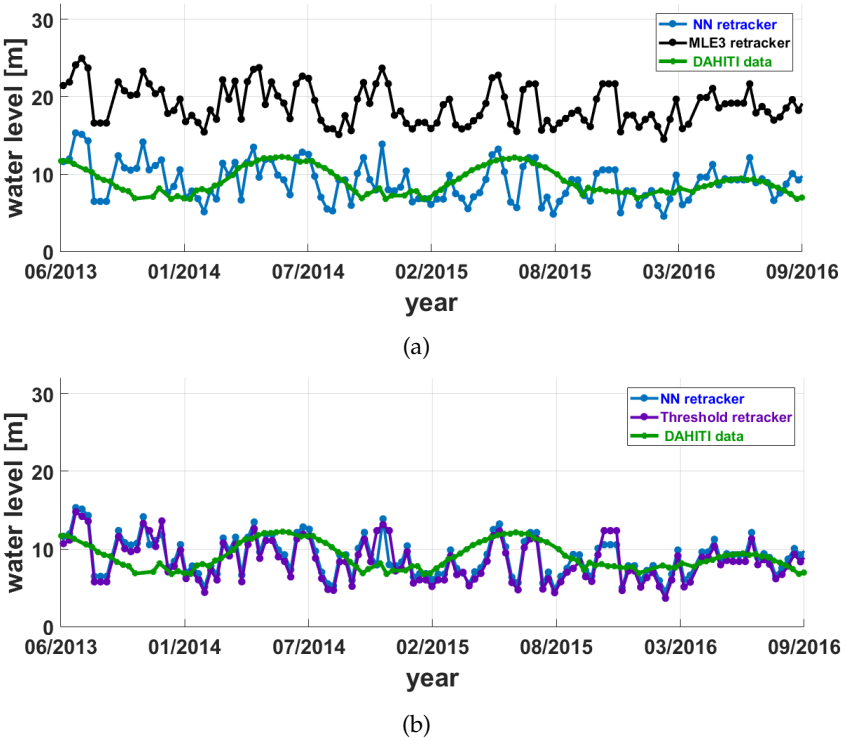
	NN [m]	MLE3 [m]	Threshold [m]
Mean	2.08	10.12	2.33
RMSE	2.63	10.46	2.91

Table 5.3 shows, that the results of the neural network and the threshold retracker are very comparable and only show slight differences of maximal 20 cm. In the case of the MLE3 retracker, one can see the influence of the bias in case of the mean value and the RMSE. However, again it can be said that all three retracker perform not well, as the values in Table 5.3 are very high. The reason for the worse results compared to the previous one could be found in the more difficult situation of the study area, as mentioned earlier. However, during the spring



(a) (b)

Figure 5.9: *Histogram of the residuals of the time series created by the algorithm and the MLE3 retracker (a) and the threshold retracker (b) regarding the in situ time series respectively at the Cupari river*



(a)

(b)

Figure 5.13: *Comparison of the time series, based on the neural network and the MLE3 retracker (a) and the threshold retracker (b) with the DAHITI time series respectively at Lago do Madabá Grande*

and summer time, in which the floodplain is covered well with water, the results improve compared to the DAHITI time series. In the next step, the developed algorithm will be applied with 900 waveform training data and the 30 neuron input layer.

The algorithm approach

The parameters which are mentioned for the 30 window approach in Table 4.2.2 are also used in this case. By applying the algorithm to the data set, an accuracy of 90.8% is reached by the neural network, which is worse compared to the previous tests in which an accuracy of 97.2% could be reached. However, one can see the influence of the lower network accuracy in the quality of the resulting time series, which can be seen compared with the MLE3 retracker in Figure 5.15 (a) and with the threshold retracker in Figure 5.15 (b). It can be seen well, that the algorithm performs not that good, compared to the previous test with the network. Thus, now the resulting time series has several more outliers which are higher compared to the previous test and also compared to the threshold retracker. This bolsters the assumption that the algorithm shows a worse performance compared to the neural network. It is interesting that significant outlier now also occur during the wet season in which it can be assumed, that water covers the floodplain. This leads to the assumption that the algorithm is not able to retrack the resulting waveforms reasonably. However, there are still several parts of the time series, in which the estimated time series match well with the DAHITI time series.

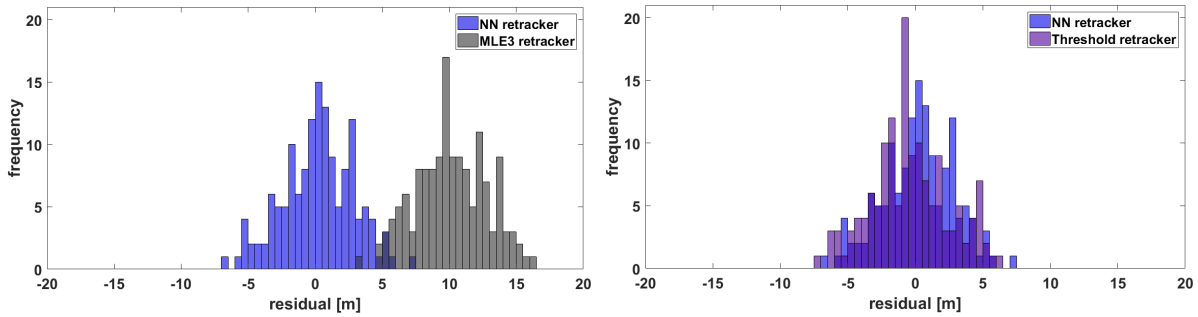
If one takes a look at the histograms of the residuals in Figure 5.16, one can see the differences more clear. In the histogram in Figure 5.16 (a) the histogram of the algorithm shows compared to the one of the MLE3 retracker a less amount of residuals. In case of the threshold retracker in Figure 5.16 (b), the result is mixed, but it seems that the algorithm has a higher amount of residuals when it comes to the higher residual values.

This can finally be verified by taking a look at the statistical data, in which again the previous neural network test will also be shown as a comparison:

Table 5.4: *Comparison of the algorithm, MLE3 and the neural network approach at the Lago do Madabá Grande*

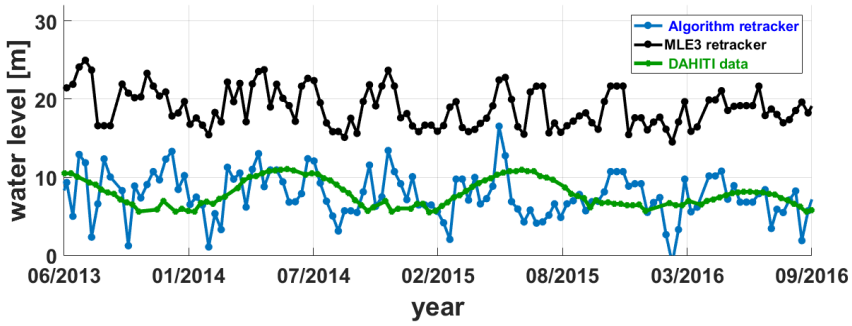
	Algorithm [m]	MLE3 [m]	Threshold [m]	NN [m]
Mean	2.71	10.12	2.33	2.08
RMSE	3.27	10.46	2.91	2.63

Table 5.4 shows again that the results of the algorithm approach are worse, compared to the previous test of the neural network. As is can be seen, the result of the threshold retracker and the neural network approach are in all cases better. It is also significant that the improvements in both cases are of several decimeters. Only the MLE3 retracker shows worse results, also in case of the standard deviation. With this, the previous assumption that the algorithm performs worse compared to the other retracking approaches could be shown again. Next, a lake will be analysed which is located on another continent with the different surrounding landscape. Thus it can be interesting to see the performance of the developed algorithms there.

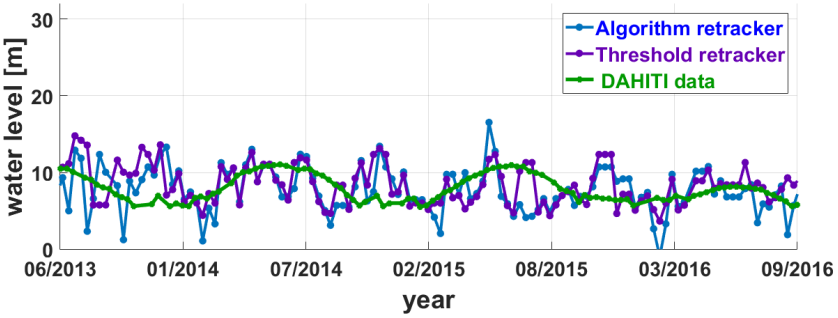


(a) (b)

Figure 5.14: *Histograms of the residuals of the time series based on the neural network, compared to the one of the MLE3 retracker (a) and the threshold retracker (b) at Lago do Madabá Grande*



(a)

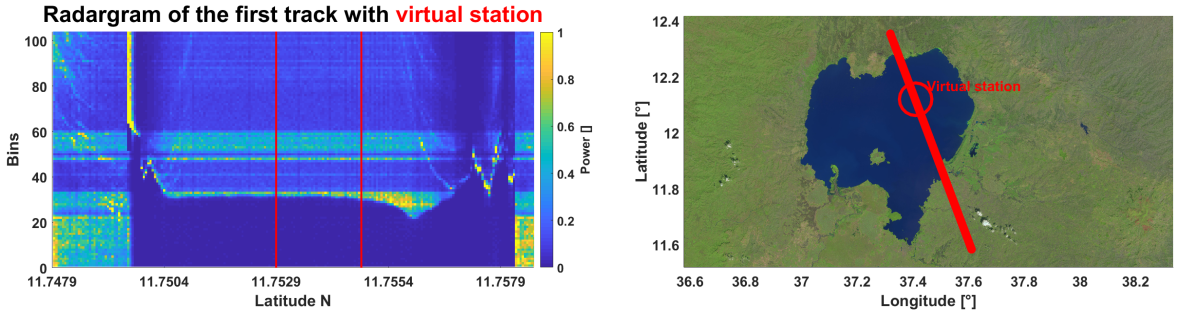


(b)

Figure 5.15: *Comparison of the time series, based on the algorithm and the MLE3 retracker (a) and the threshold retracker (b) with the DAHITI time series respectively at Lago do Madabá Grande*

5.1.3 Lake Tana

In the last test, one can see the estimated time series of the neural network and the algorithm on Lake Tana. Again there is no in situ time series available so that also the data from DAHITI database is used for the validation of the results. As mentioned in chapter 1, this study area is more different than the previous ones caused by the change from tropical rain forest to forests located in the desert. Therefore, to get an impression about the difficulties which the estimation of the water levels can face, one can use the radargram again to visualise one track, as done in the previous cases, together with an overview of the study area:



(a) (b)

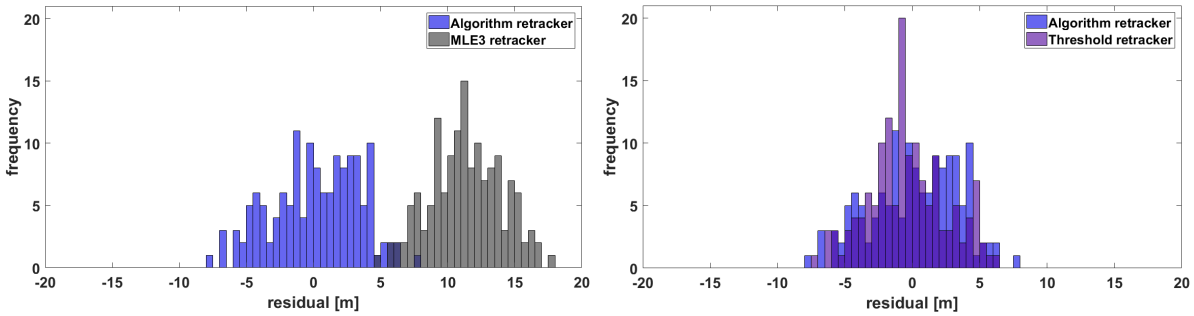
Figure 5.17: *Radargram of the track with the chosen area (a) and an overview of the area (b) at Lake Tana*

In opposite to the previous radargram one can see in Figure 5.17 two vertical lines which indicate the coastal areas which are crossed by the satellite. Thus, the separation in the water area and land area can be seen clearly. On both sides, the signals are mixed without a clear leading edge. A closer look at the area of the virtual station shows that there is a clear line between the signal and the thermal noise. In the lower bins, a homogeneous dark blue pattern is prominent which indicate low power. However, in the area of bins with high numbers, additional high power areas can be seen, which indicate noisy signals behind the leading edge. An additional problem, which can be seen in the radargram, is the appearance of additional horizontal lines inside the area of higher bins, which can also be mistakenly seen as leading edges. Thus, it can be said that in this case, the situation is more difficult as in the previous cases. However, it will be interesting to see the performance of the network in this area.

The neural network approach

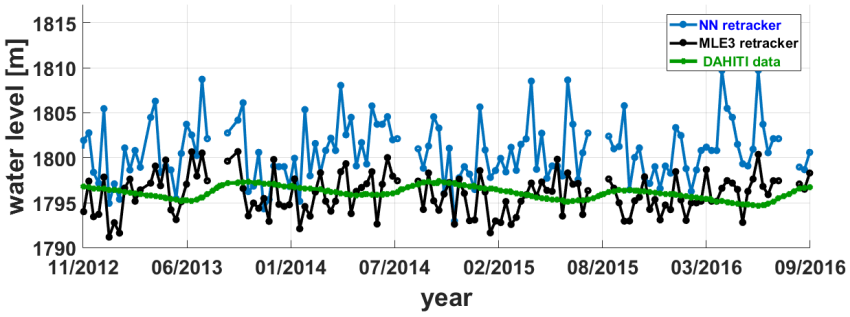
As done in the previous cases, the estimated time series of the neural network will be compared with the MLE3 and threshold retracker with again a threshold of 50%. The DAHITI data set is used as a reference to compare the different time series. By using the neural network, a performance of only 32.7% is reached, which already indicate, that there are significant problems by estimating the correct position of the leading edge. The resulting time series can be seen now below:

In Figure 5.18 (a) and (b) it can be seen clearly, that both retracker show significant variations compared to the DAHITI time series. However, while the estimated time series of the MLE3 retracker is in the same magnitude as the DAHITI time series, the network shows significant

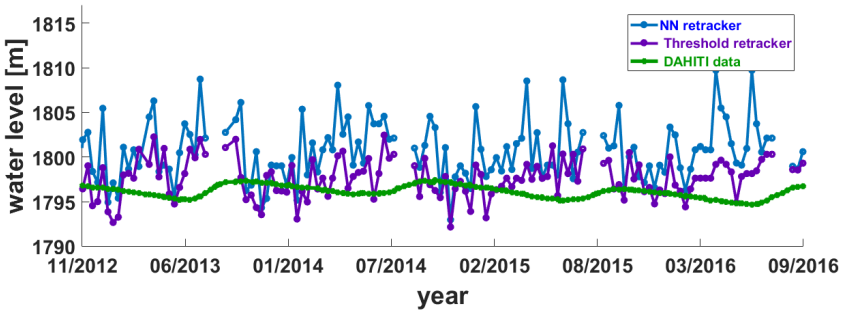


(a) (b)

Figure 5.16: *Histograms of the residuals of the time series based on the algorithm, compared to the one of the MLE3 retracker (a) and the threshold retracker (b) at Lago do Madabá Grande*



(a)



(b)

Figure 5.18: *Comparison of the time series, based on the neural network and the MLE3 retracker (a) and the threshold retracker (b) with the DAHITI time series respectively at Lake Tana*

variations and outlier. The threshold retracker also shows several outliers but is closer to the magnitude of the DAHITI time series. To understand this result better, one can take a look at some waveforms which are measured over Lake Tana, which can be seen in Figure 5.19. In these figures one can see, that there are several peaks behind the assumed leading edge and the neural network has big problems with this kind of waveforms, as it is trained to find the maximum peak. However, a closer look at the waveforms shows, that they have one characteristic pattern which is always similar. There is always a significant leading edge, and only after that, the peaks occur. This can also be seen very well at the radargram in Figure 5.17, in which the bins in front of the leading edge show a homogeneous blue, which means that there are values close to zero. Thus, one can train the network how the first ascending part looks like. Now the question is, how this can be done? The best way to do that is with the threshold retracker, which is already used as a comparison. As this retracker marks the first bin which exceeds the defined threshold. Because the threshold retracker shows at least better results compared to the network approach, it is interesting to see how the network performs, after it is trained with data sets, which are labelled with the same retracker.

Neural network based on a threshold training

As already explained, the position of the leading edge is now defined by the threshold retracker. Thus, one can choose the threshold of 35% of the maximum value. The reason is, that now the maximum peak can be located in the trailing edge of the waveforms and one wants to be sure, to find the leading edge. Also Tourian (2012) mention that at the Balaton lake 20% and at lake Urmia 40% creates reasonable results. Because this case has more similarities with Lake Urmia, one can choose a value close to this one. Thus now this trained network can be applied to the test data to see, how the results improved if all other network parameters remain the same. With this configuration, the neural network reaches an accuracy of 88.3%, which is compared to the previous 32.7% a significant improvement. With this motivative results, one can have a look at the estimated time series. Thereby it is to mention, that the threshold retracker is now also using the 35% threshold.

The results can be seen in Figure 5.20. Besides there are still several outliers, one can see in Figure 5.20 (a) an improvement as the estimated time series by the neural network is matching now better with the time series of the MLE3 retracker and the same pattern can be seen. However, at the same time, it seems that there is an offset of some meter at the time series, retracked by the neural network. At the same time, it matches well with the time series, which is based on the threshold retracker in Figure 5.20 (b). Interestingly, also an offset can be seen but it is more reduced compared to the previous comparison. However, it has to be mentioned, that all three time series show significant variations compared to the time series of the DAHITI database.

To get a better idea about the residuals, one can take a look at the histograms of the residuals. In the histogram in Figure 5.21 (a), one can see interestingly, the bias between the two time series can be seen well as the residuals of the time series. However, it can be seen that there is a higher amount of residuals of the neural network time series, compared to the one of the MLE3 retracker. In case of the comparison of the neural network with the threshold retracker in Figure 5.21 (b), one can see that the neural network shows a higher amount of residuals if the values of the residuals increase but also the threshold retracker shows a significant peak for

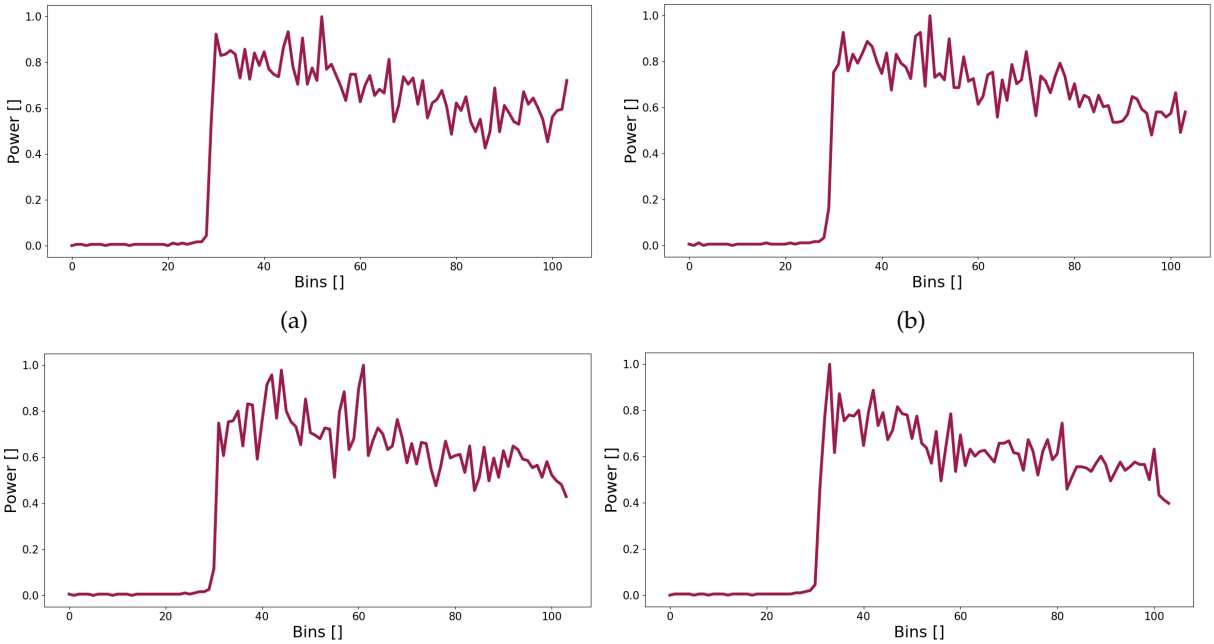


Figure 5.19: *Waveforms measured over lake Tana*

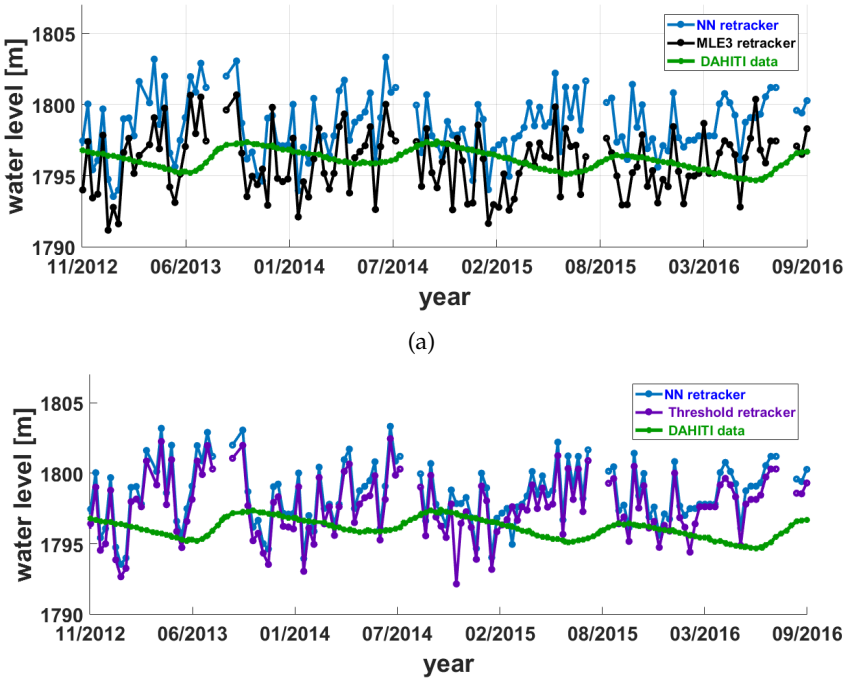


Figure 5.20: *Comparison of the time series, based on the neural network and the MLE3 retracker (a) and the threshold retracker (b) with the DAHITI time series respectively at Lake Tana by using a threshold retracker for classification*

residuals of 2 m. Also, in this case, the smaller bias can be seen well, but as seen in Figure 5.20 (b), it is not that big.

At least one can take a look at the statistical data. In that case, also the previous neural network approach is shown (NN old), which used training data which is labelled by estimating the maximum peak. As one can see in table 5.5, the results could be well improved by using the new approach. However, both neural network approaches are worse compared to the results of the MLE3 and threshold retracker. The differences between the new approach and the MLE3 retracker is in case of the mean value around 50 cm better, and in case of the RMSE, an improvement of roughly 1 meter can be seen. The threshold retracker shows in all cases an improvement of around 50 dm. However, regarding the previous test, the MLE3 and threshold retracker are several meters better. This underlines the fact that the preparation of the input data is essential and has a significant influence on the outcome of the neural network. In the last step, the developed algorithm will be applied to this case study, which is now also trained with the new method.

Table 5.5: *Comparison of the statistical values of two neural network approaches and the MLE3 and threshold retracker*

	NN [m]	MLE3 [m]	Threshold [m]	NN old [m]
Mean value	2.88	2.00	2.41	4.82
RMSE	3.44	2.39	2.91	5.79

Using the algorithm with a threshold training

In the last test, the algorithm approach is applied to Lake Tana. In this test, only the improved method by using the threshold will be used. The reason is that it can be assumed that the algorithm faces the same difficulties as the neural network with the former kind of training data set. Again a threshold of 0.35% is used for the retracker. After applying the algorithm, a performance of 65% is reached, which is low compared to the previous one. However, it can be mentioned that the algorithm approach also showed in the previous tests lower accuracies compared to the neural network approach. The resulting time series can be seen in Figure 5.22 (a) and (b). In Figure 5.22 (a) the time series can be seen compare with the one which is based on the MLE3 retracker, and it shows significant outliers. However, at the same time, there are also several matches between the two time series and also the bias, which was seen before, is now reduced. In the comparison between the time series, which is based on the algorithm and the one, which is based on the threshold retracker in Figure 5.22 (b), one can see that there are now much more similarities. However, also, in this case, the time series, based on the algorithm, shows much more significant outliers compared to the one of the threshold retracker. Thus it can be said that this approach shows better results compared to the other two retrackers, besides there are several outliers.

It is interesting to take a look at the histograms of the residuals which illustrates the amount of residuals of the different time series. This can be seen in Figure 5.23. Taking a look at the comparison with the MLE3 retracker in Figure 5.23 (a) one can see the several strong outliers in the negative direction of the algorithm approach. However, there are also some outliers

in the positive direction which are also dominated by the algorithm. Regarding the smaller residuals, the MLE3 retracker shows in several cases a higher amount of residuals compared to the algorithm. The opposite can be seen in comparison with the threshold retracker in Figure 5.23 (b). Where also the algorithm shows a high amount of residuals with smaller values, compared to the threshold retracker. Thus, by using the histogram it could be shown well, that the algorithm has now bigger residuals than before, but the bias seems to be reduced. However, one can take a look at the statistical values to see, how significant the errors are. As before, also the results of the neural network is shown to make the variations more visible:

Table 5.6: *Comparison of the standard deviation and the mean values of the algorithm, MLE3 retracker and neural network approach regarding the DAHITI database*

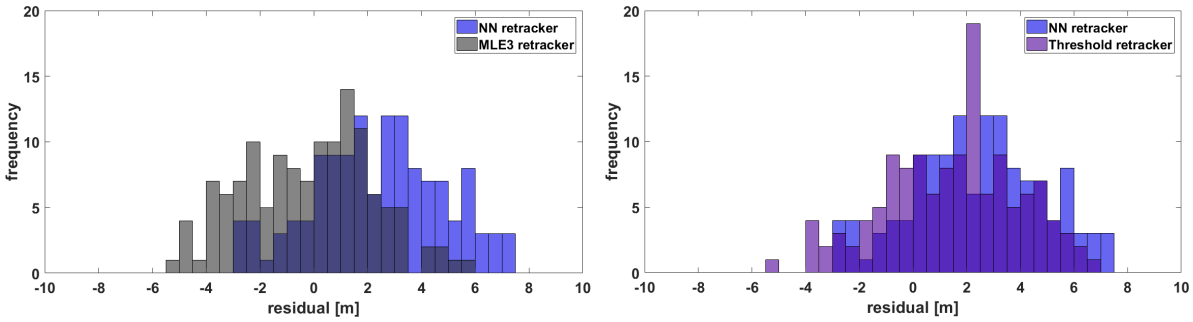
	Algorithm [m]	MLE3 [m]	Threshold [m]	NN [m]
Mean value	2.84	2.00	2.41	2.88
RMSE	3.49	2.39	2.91	3.44

Table 5.6 shows that the difference compared to the network approach is very low, and regarding the mean value, the algorithm shows a slightly better result. This can be partly explained by the reduced bias between the estimated water height and the MLE3 time series. However, the MLE3 retracker results are several decimeters better, and in case of the RMSE, an improvement of more than 1 meter is visible. In the case of the threshold retracker, the differences towards the algorithm approach are smaller but still are several decimeters. Thus, it can be concluded that both developed approaches show big problems in this study area but also the two other retrackers have more outliers in this study area. However, now, the final test can be conducted, in which the transferability of the neural network is tested.

5.1.4 The final test

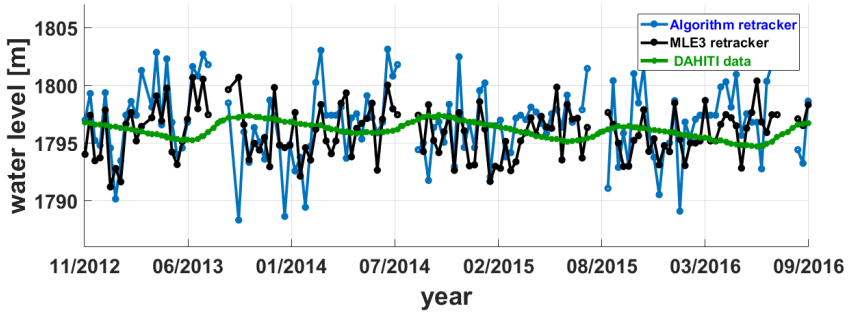
Now, a more realistic situation will be tested which means, that training data from another area is used for the neural network. The reason for this test is that it can be assumed, that one has not always time or the possibility to create training data of a water body which should be analysed. Thus, it should be possible to test the water body with a network, which is trained with data from another area. Because in situ data is only available from the Cupari river, this area will be used to test the results. However, by choosing the training data, it is only meaningful to use the data of Lago do Madabá Grande because it is the only other test area, in which reliable results could be reached and which is also located in the same river basin. In the case of Lake Tana, there were several problems to create usable training data because the shape of the waveforms differs a lot from the waveforms which are gained over the Amazon river basin. Thus, it would be of no avail to use this data as training data for the Cupari river.

Hence, now a situation is tested, in which training data inside a basin is available, and another water area in this region has to be analysed. This is even more challenging since data of a floodplain is now applied to a river. In this case, only the neural network approach will be tested, because it showed in the previous tests always the best performance. However, it has to be mention that the network parameters have to be changed to get more reliable results. Because in the previous cases the network had to detect waveforms very similar to the ones

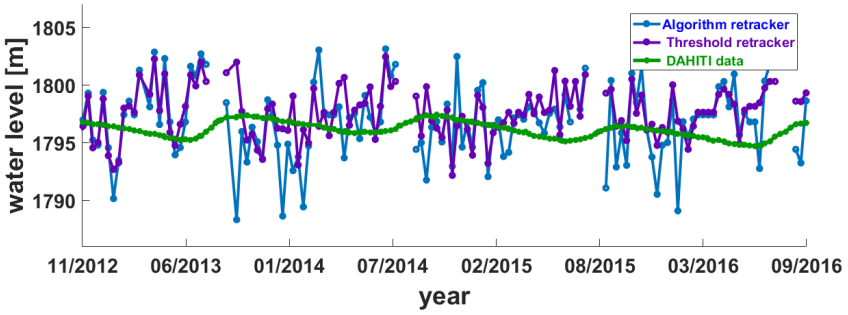


(a) (b)

Figure 5.21: *Histograms of the residuals of the time series based on the neural network, compared to the one of the MLE3 retracker (a) and the threshold retracker (b) at Lake Tana*



(a)



(b)

Figure 5.22: *Comparison of the time series, based on the algorithm and the MLE3 retracker (a) and the threshold retracker (b) with the DAHITI time series respectively at Lake Tana by using a threshold retracker for classification*

it was trained with. However, now the network has to estimate the leading edge in different waveforms. Thus it is helpful to enable it, to learn more pattern. As already mentioned, the number of hidden neurons determines, how many weights are available to learn the different pattern. By increasing this number, the neural network has more possibilities to learn. Thus, the following set of parameters will be used:

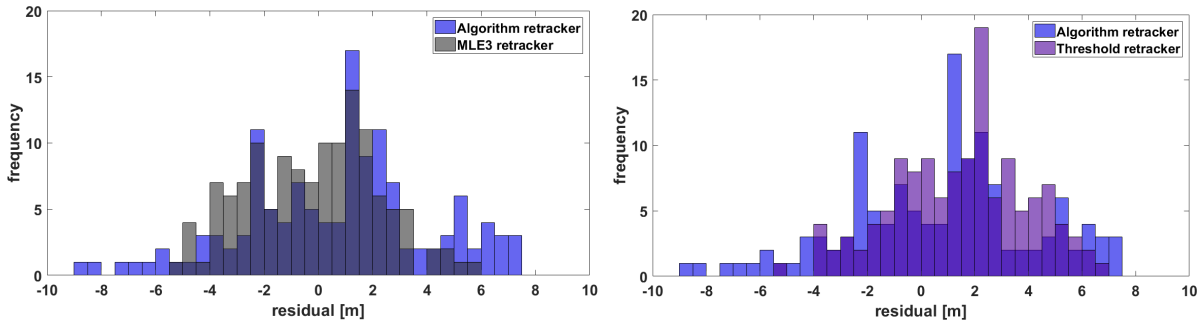
Table 5.7: *Parameter of the neural network for transferable results*

Parameter	Value
Hidden neurons	1000
Learning rate	0.00075
Epochs	700

In this case, also the whole time series of the Cupari river can be used because no training or validation data is used from it. By applying the neural network, only an accuracy of 53.4% is be reached, which is a lower result than the previous ones. In Figure 5.24, the resulting time series can be seen, which is again compared with the MLE3 and threshold retracker and the in situ data. Surprisingly, the resulting time series in Figure 5.24 (a) shows all seasonal variations and the resulting outliers are only limited in magnitude. However, it has to be mentioned that the result is much more scattered than before. But at the same time, several similar water heights compared to the MLE3 retracker can be seen. Hence, it is interesting that even with this low network accuracy this time series could be created. The most significant outlier can be seen in autumn 2012 and 2015 and winter 2013. Also, the comparison with the threshold retracker in Figure 5.24 (b) shows several similar water levels. However, it can be seen well that the time series, which is based on the neural network, even match well in 2011 with the in situ data.

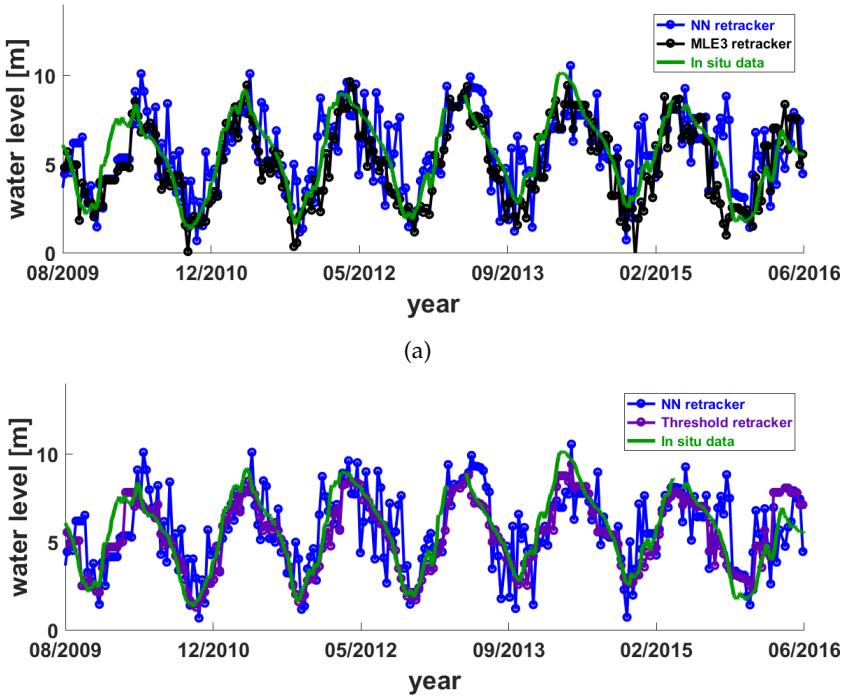
To get an overview of the performance, one can also take a look at the histogram of the residuals. The comparison between the histogram of the neural network and the MLE3 retracker in Figure 5.25 (a) shows, that the neural network shows more significant residuals than the MLE3 retracker. These residuals are mostly located in the negative direction, which could be also seen in Figure 5.24 (a). The higher amount of the residuals can be also explained as the time series, which is analysed, is longer compared to the previous ones. This can be seen very clear if one takes a look at the comparison between the neural network and the threshold retracker in Figure 5.24 (b). In that case, the number of residuals of the threshold retracker reaches very high values, which is caused by the higher number of analysed residuals. Besides that, it can be seen clearly that the neural network approach shows a higher number of residuals.

To get a better idea about the results, one can also have a look at the statistical data compared with the MLE3 and threshold retracker in Table 5.8. One can see, that the results are indeed comparable, as the results of the neural network are in decimeter difference regarding the other data. However, it is very interesting to see, that the threshold retracker performs best in this comparison. This could already be seen in Figure 5.24 (b), as there is a very high peak for the small residuals. For all other residuals, the number is much lower. However, by regarding these values, it can be concluded, that it is indeed possible to transfer training data from one area of a basin to another one to train a neural network. Next, an overview of the gained results will be given with some conclusions. At the end also some outlook for further studies will be given.



(a) (b)

Figure 5.23: *Histograms of the residuals of the time series based on the algorithm, compared to the one of the MLE3 retracker (a) and the threshold retracker (b) at Lake Tana*



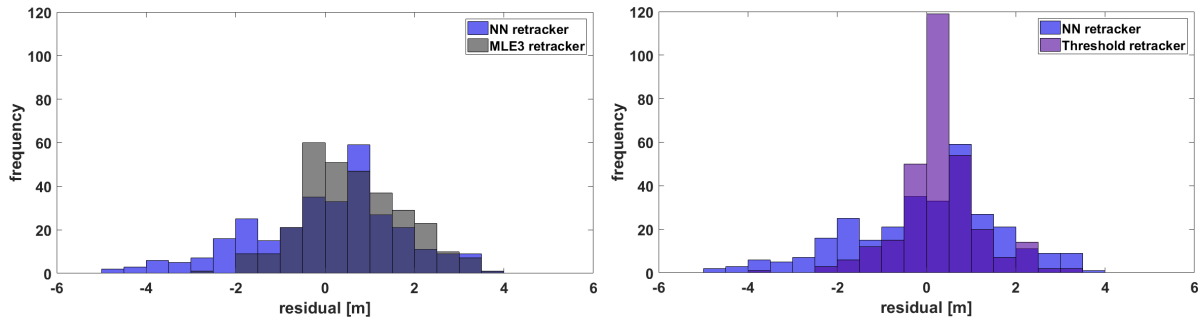
(a)

(b)

Figure 5.24: *Comparison of the time series, based on the neural network, which is trained with data of another study area, and the MLE3 retracker (a) and the threshold retracker (b) with the in situ time series respectively at the Cupari river*

Table 5.8: *Statistical data of the neural network, the MLE3 and threshold retracker of the Cupari river, after training the network with data of another area*

	NN [m]	MLE3 [m]	Threshold [m]
Mean value	1.34	1.00	0.65
RMSE	1.66	1.11	0.93



(a) (b)

Figure 5.25: *Histograms of the residuals of the time series based on the neural network, which is trained with data of another study area, compared to the one of the MLE3 retracker (a) and the threshold retracker (b) at the Cupari river*

Chapter 6

Conclusion and outlook

6.1 Overview of the work and conclusions

Now we come to an end of a long way. It began with the overview of the study areas and the data which were used in the thesis. Next, the basics of satellite altimetry are described, starting with the theory behind the measurements. Later the retracking methods are described which estimate the position of the leading edge inside the waveform. In the next chapter, we learned a lot about neural networks. This starts with the biological neurons and went deeper until it was described, how to implement an artificial neural network. After these introductions, we had all the basic knowledge to start with our two new developed retracking approaches. The first one was based solely on a neural network, which was used to analyse the waveforms and estimate the best peak position. This was done with three different training sets to see if it is even possible to create a reasonable time series with less training data sets. We also learned more about the network parameter and their influence on the results. Also, a second approach was developed, which uses the results of a neural network and estimates, based on these information, which position is the best for retracking. Thereby, we also tested two versions with different sizes of input data, to see, which one is the best. In the final chapter, we then applied the two approaches to our test areas. With this, we were able to see the performance of the two algorithms in different situations. It was also possible to compare it with the MLE3 on board retracker and the threshold retracker and also with the in situ data, or in two other cases, with data of the DAHITI database. The final test was also the most challenging one, in which we trained the neural network with data of one area and then applied it to another.

As it can be seen, we learned a lot about very different subjects, and at the same time, we could directly apply the gained knowledge in our approaches. By doing so, four key points can be identified:

- It is possible to create useful time series with very less training data
- Only the input layer with 30 input neurons created a usable time series
- The neural network approach outperforms the algorithm approach in all cases
- It is possible to train the network with one water body and apply this trained network to other water bodies inside the same basin

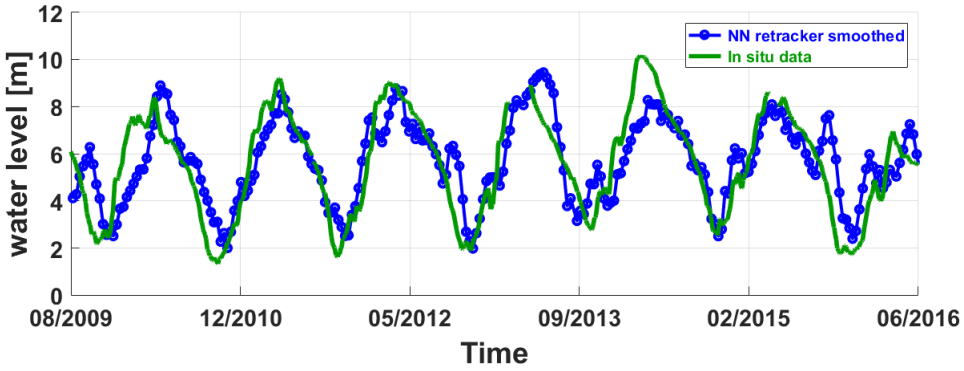


Figure 6.1: *Time series of the smoothed neural network after it was trained with data of another area compared with in situ data*

In chapter 4 we applied three different training data sets to the neural network which had 600, 720 and 900 waveforms respectively. This can be seen well in Figure 4.11, in which a time series was estimated which fit well the in situ time series. A look at Table 4.1.3 shows, that with 600 waveforms a high accuracy can be reached. However, the higher accuracy was reached with the 900 waveform training set which was used for further studies. In the case of the algorithm approach, it could be shown, that at least 30 input neurons are needed to create a reasonable time series. The tests showed that a window of 10 bins is too small to gain enough data to label the waveform segment correctly which can be seen well in Figure 4.16. This leads us to the next point, even with 30 input neurons the algorithm had in the most cases a lower performance compared to the neural network. However, it has to be mentioned that in the case of Lake Tana, the algorithm had worse performance, but at the same time the bias, which could be seen in case of the neural network, was reduced. In the case of Lake Tana it could also be shown, how important the creation of the training data is. Thus it has to be stressed, that the definition of the leading edge in the training data has an essential influence on the later results of the neural network. In the final test, it could be shown, that the training data is transferable. Besides it is noisier, a reasonable time series could be created by using a network which was trained over Lago do Madabá grande and applied to the Cupari river. The resulting time series can be seen in Figure 5.24. Both water bodies show very different behaviour, and it has to be mentioned, that the floodplain contains during the autumn very less water.

If one uses a moving mean with window size in the time domain of 40 days to eliminate the outliers, the result is even more reliable, as it can be seen in Figure 6.1. The resulting time series has now a mean value of 1.04 m and an RMSE of 1.31 m. In Figure 6.1 it can be seen that most of the seasonal variations could be captured well. Only in the year 2014 the seasonal peak of the estimated time series missed the peak and showed too low values. Whereas the seasonal peaks in 2010 and 2013 show a delay. Regarding outliers, only around the 2015 seasonal peak additional peaks can be seen. However, it can be said that by smoothing the result, the time series matches very well, and also the statistical data show a significant improvement. Thus, as a conclusion, it can be said, that the neural network shows a high potential for further studies. This gets even more visible if one regards the fact, that only a basic neural network is used in this study.

6.2 Outlook

As mentioned before, we just used a single layer feed-forward neural network. As the name implies, we use one single hidden layer, and we move from the input side to the output side. Both things could be helpful to be changed. Thus, one can use more layer which would help to detect even more difficult pattern. However, the additional layer would be even more helpful, if we start using convolutional neural networks (CNN). With them, it is possible to use images of our waveforms, as we used them in our radargrams. They would provide even more information because we would see more waveforms of a track which gives the network an idea of the whole situations, instead of just seeing the actual waveform. The other point is that it would be helpful to go also backwards. By using so-called recurrent neural networks (RNN), it is possible to give the information of the assumed actual leading edge position to the next following waveform analysis. Then it is possible to use this additional informations and, based on the previous leading edge position, it can estimate the best position in the current waveform. Because we are working with time series, it is beneficial to know the previous assumed leading edge position for the current case. However, of course, it is also possible to combine the CNN and RNN which would then maybe create the best results.

It could be shown, that there are many possibilities of further study areas. This thesis had the aim to provide an idea about the possibilities of the neural network by using the basic version, which already created good results. Thus, this is just the first step of a wild range of possible further studies, which can be conducted. If we regard the rapid development of neural networks which provide new possibilities, we soon have maybe even more chances to create reliable time series.

Bibliography

Douglas E Alsdorf, Ernesto Rodríguez, and Dennis P Lettenmaier. Measuring surface water from space. *Reviews of Geophysics*, 45(2), 2007. doi: https://doi.org/10.1029/2006RG000197.

Donald E Barrick and Belinda J Lipa. Analysis and interpretation of altimeter sea echo. In *Advances in geophysics*, volume 27, pages 61–100. Elsevier, 1985. doi: https://doi.org/10.1016/S0065-2687(08)60403-3.

Donald E Barrick and Calvin T Swift. The seasat microwave instruments in historical perspective. *IEEE Journal of Oceanic Engineering*, 5(2):74–79, 1980. doi: https://doi.org/10.1109/JOE.1980.1145457.

Henry Walter Bates. *The naturalist on the River Amazons, A record of adventures, habits of animals, sketches of Brazilian and Indian life and aspects of nature under the Equator during eleven years of travel.* London,J. Murray,, 1863. doi: https://doi.org/10.5962/bhl.title.21335. URL `https://www.biodiversitylibrary.org/item/61547`. https://www.biodiversitylibrary.org/bibliography/21335.

Charon Birkett. The global remote sensing of lakes, wetlands and rivers for hydrological and climate research. In *Geoscience and Remote Sensing Symposium, 1995. IGARSS'95.'Quantitative Remote Sensing for Science and Applications', International*, volume 3, pages 1979–1981. IEEE, 1995. doi: https://doi.org/10.1109/IGARSS.1995.524084.

Charon M Birkett. Contribution of the topex nasa radar altimeter to the global monitoring of large rivers and wetlands. *Water Resources Research*, 34(5):1223–1239, 1998. doi: https://doi.org/10.1029/98WR00124.

Charon M Birkett and B Beckley. Investigating the performance of the jason-2/ostm radar altimeter over lakes and reservoirs. *Marine Geodesy*, 33(S1):204–238, 2010. doi: https://doi.org/10.1080/01490419.2010.488983.

G Brown. The average impulse response of a rough surface and its applications. *IEEE transactions on antennas and propagation*, 25(1):67–74, 1977. doi: doi:10.1109/TAP.1977.1141536.

Stéphane Calmant, Frédérique Seyler, and Jean François Cretaux. Monitoring continental surface waters by satellite altimetry. *Surveys in geophysics*, 29(4-5):247–269, 2008. doi: https://doi.org/10.1007/s10712-008-9051-1.

DE Cartwright and Anne C Edden. Corrected tables of tidal harmonics. *Geophysical journal international*, 33(3):253–264, 1973. doi: https://doi.org/10.1111/j.1365-246X.1973.tb03420.x.

DE Cartwright and RJ Tayler. New computations of the tide-generating potential. *Geophysical Journal International*, 23(1):45–73, 1971. doi: https://doi.org/10.1111/j.1365-246X.1971.tb01803.x.

PG Challenor and MA Srokosz. The extraction of geophysical parameters from radar altimeter return from a non-linear sea surface. *Mathematics in remote sensing*, pages 257–268, 1989.

AVISO CNES. Aviso. URL https://www.aviso.altimetry.fr/en/home.html.

Richard D. Ray. A global ocean tide model from topex/poseidon altimetry: Got99. 2. *NASA Tech. Memo. 209478*, 209478, 10 1999. URL https://denali.gsfc.nasa.gov/personal_pages/ray/MiscPubs/19990089548_1999150788.pdf.

Joecila Santos Da Silva, Stephane Calmant, Frederique Seyler, Otto Corrêa Rotunno Filho, Gerard Cochonneau, and Webe Joao Mansur. Water levels in the amazon basin derived from the ers 2 and envisat radar altimetry missions. *Remote sensing of environment*, 114(10): 2160–2181, 2010. doi: https://doi.org/10.1016/j.rse.2010.04.020.

Curt H Davis. Growth of the greenland ice sheet: A performance assessment of altimeter retracking algorithms. *IEEE Transactions on Geoscience and Remote Sensing*, 33(5):1108–1116, 1995. doi: https://doi.org/10.1109/36.469474.

Curt H Davis. A robust threshold retracking algorithm for measuring ice-sheet surface elevation change from satellite radar altimeters. *IEEE Transactions on Geoscience and Remote Sensing*, 35(4):974–979, 1997. doi: https://doi.org/10.1109/36.602540.

JL Davis, TA Herring, II Shapiro, AEE Rogers, and Gunnar Elgered. Geodesy by radio interferometry: Effects of atmospheric modeling errors on estimates of baseline length. *Radio science*, 20(6):1593–1607, 1985. doi: https://doi.org/10.1029/RS020i006p01593.

Xiaoli Deng. *Improvement of geodetic parameter estimation in coastal regions from satellite radar altimetry*. Curtin University of Technology., 2003.

Xiaoli Deng and WE Featherstone. A coastal retracking system for satellite radar altimeter waveforms: Application to ers-2 around australia. *Journal of Geophysical Research: Oceans*, 111(C6), 2006. doi: https://doi.org/10.1029/2005JC003039.

Shailen Desai, John Wahr, and Brian Beckley. Revisiting the pole tide for and from satellite altimetry. *Journal of Geodesy*, 89(12):1233–1243, 2015. doi: https://doi.org/10.1007/s00190-015-0848-7.

Shailen D Desai. Observing the pole tide with satellite altimetry. *Journal of Geophysical Research: Oceans*, 107(C11):7–1, 2002. doi: https://doi.org/10.1029/2001JC001224.

Charles Desportes, Estelle Obligis, and Laurence Eymard. On the wet tropospheric correction for altimetry in coastal regions. *IEEE transactions on geoscience and remote sensing*, 45(7): 2139–2149, 2007. doi: https://doi.org/10.1109/TGRS.2006.888967.

Balázs M Fekete and Charles J Vörösmarty. The current status of global river discharge monitoring and potential new technologies complementing traditional discharge measurements. *IAHS publ*, 309:129–136, 2007. URL http://hydrologie.org/redbooks/a309/309015.pdf.

M Joana Fernandes, Nelson Pires, Clara Lázaro, and Alexandra L Nunes. Tropospheric delays from gnss for application in coastal altimetry. *Advances in Space Research*, 51(8):1352–1368, 2013. doi: https://doi.org/10.1016/j.asr.2012.04.025.

M Joana Fernandes, Clara Lázaro, Michaël Ablain, and Nelson Pires. Improved wet path delays for all esa and reference altimetric missions. *Remote Sensing of Environment*, 169:50–74,

2015. doi: https://doi.org/10.1016/j.rse.2015.07.023.

Lee-Lueng Fu and Anny Cazenave. *Satellite altimetry and earth sciences: a handbook of techniques and applications*, volume 69. Elsevier, 2000.

Matt W Gardner and SR Dorling. Artificial neural networks (the multilayer perceptron)-a review of applications in the atmospheric sciences. *Atmospheric environment*, 32(14-15): 2627–2636, 1998. doi: https://doi.org/10.1016/S1352-2310(97)00447-0.

Christine Gommenginger, Pierre Thibaut, Luciana Fenoglio-Marc, G Quartly, Xiaoli Deng, Jesús Gómez-Enri, P Challenor, and Yonggang Gao. Retracking altimeter waveforms near the coasts. In *Coastal altimetry*, pages 61–101. Springer, 2011. doi: https://doi.org/10.1007/978-3-642-12796-0_4.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

Chad Greene. borders, 2019. URL `https://www.mathworks.com/matlabcentral/fileexchange/50390-borders`.

Ad Hoc Group, C Vörösmarty, A Askew, W Grabs, RG Barry, C Birkett, P Döll, B Goodison, A Hall, R Jenne, et al. Global water data: A newly endangered species. *Eos, Transactions American Geophysical Union*, 82(5):54–58, 2001. doi: https://doi.org/10.1029/01EO00031.

Jinyun Guo, Xiaotao Chang, Yonggang Gao, Jialong Sun, and Cheinway Hwang. Lake level variations monitored with satellite altimetry waveform retracking. *IEEE journal of selected topics in applied earth observations and remote sensing*, 2(2):80–86, 2009. doi: doi:10.1109/JSTARS.2009.2021673.

S Hagemann and L Dümenil. A parametrization of the lateral waterflow for the global scale. *Climate dynamics*, 14(1):17–31, 1997. doi: https://doi.org/10.1007/s003820050205.

George S Hayne. Radar altimeter mean return wave forms from near-normal-incidence ocean surface scattering. 1980. doi: doi:10.1109/TAP.1980.1142398.

Jeff Heaton. The number of hidden layers, 2017. URL `https://www.heatonresearch.com/2017/06/01/hidden-layers.html`.

Norden E Huang and Steven R Long. An experimental study of the surface elevation probability distribution and statistics of wind-generated waves. *Journal of Fluid Mechanics*, 101(1): 179–200, 1980. doi: https://doi.org/10.1017/S0022112080001590.

WM Kaula. The terrestrial environment: Solid earth and ocean physics, application of space and astronomic techniques. *NASA Contract. Rep*, 1579:150, 1969. URL `https://ilrs.cddis.eosdis.nasa.gov/docs/williamstown_1968.pdf`.

Maurice G Kendall. *Advanced Theory Of Statistics Vol-I*. Charles Griffin: London, 1943.

Jan Kouba. Implementation and testing of the gridded vienna mapping function 1 (vmf1). *Journal of Geodesy*, 82(4-5):193–205, 2008. doi: https://doi.org/10.1007/s00190-007-0170-0.

Chung-Yen Kuo and Huan-Chin Kao. Retracked jason-2 altimetry over small water bodies: Case study of bajhang river, taiwan. *Marine Geodesy*, 34(3-4):382–392, 2011. doi: https://doi.org/10.1080/01490419.2011.584830.

Juliette Lambin, Rosemary Morrow, Lee-Lueng Fu, Josh K Willis, Hans Bonekamp, John Lillibridge, Jacqueline Perbos, Gérard Zaouche, Parag Vaze, Walid Bannoura, et al. The ostm/jason-2 mission. *Marine Geodesy*, 33(S1):4–25, 2010. doi: https://doi.org/10.1080/01490419.2010.491030.

Benoît Legrésy and Frederique Remy. Altimetric observations of surface characteristics of the antarctic ice sheet. *Journal of Glaciology*, 43(144):265–275, 1997. doi: https://doi.org/10.3189/S002214300000321X.

SB Luthcke, NP Zelensky, DD Rowlands, FG Lemoine, and TA Williams. The 1-centimeter orbit: Jason-1 precision orbit determination using gps, slr, doris, and altimeter data special issue: Jason-1 calibration/validation. *Marine Geodesy*, 26(3-4):399–421, 2003. doi: https://doi.org/10.1080/714044529.

Miloš Marković. Determination of total electron content in the ionosphere using gps technology. *Geonauka*, 2(4):1–9, 2014. doi: https://doi.org/10.14438/gn.2014.22.

Paul C Marth, J Robert Jensen, Charles C Kilgus, James A Perschy, John L MacArthur, David W Hancock, George S Hayne, Craig L Purdy, Laurence C Rossi, and Chester J Koblinsky. Prelaunch performance of the nasa altimeter for the topex/poseidon project. *IEEE Transactions on Geoscience and remote sensing*, 31(2):315–332, 1993. doi: https://doi.org/10.1109/36.214909.

Thomas V Martin, H Jay Zwally, Anita C Brenner, and Robert A Bindschadler. Analysis and retracking of continental ice sheet radar altimeter waveforms. *Journal of Geophysical Research: Oceans*, 88(C3):1608–1616, 1983. doi: https://doi.org/10.1029/JC088iC03p01608.

JOSEPH T McGOOGAN. Satellite altimetry applications. *IEEE Transactions on Microwave Theory and Techniques*, 23(12):970–978, 1975. doi: https://doi.org/10.1109/TMTT.1975.1128729.

W.H. Munk and G. J. F. MacDonald. *The Rotation of the Earth. A geophysical discussion.*, volume 134. American Association for the Advancement of Science, 1961. doi: https://doi.org/10.1126/science.134.3491.1683-a.

CNES NASA. *OSTM/Jason-2 Product Handbook*, 2017. URL `https://www.ospo.noaa.gov/Products/documents/J2_handbook_v1-8_no_rev.pdf`.

Alexey Natekin and Alois Knoll. Gradient boosting machines, a tutorial. *Frontiers in neurorobotics*, 7:21, 2013. doi: https://doi.org/10.3389/fnbot.2013.00021.

GD Quartly, MA Srokosz, and AC McMillan. Analyzing altimeter artifacts: Statistical properties of ocean waveforms. *Journal of Atmospheric and Oceanic Technology*, 18(12):2074–2091, 2001. doi: https://doi.org/10.1175/1520-0426(2001)018<2074:AAASPO>2.0.CO;2.

Tariq Rashid. *Make your own neural network*. CreateSpace Independent Publishing Platform, 2016. `http://makeyourownneuralnetwork.blogspot.com/`.

Frédérique Rémy, Philippe Shaeffer, and Benoît Legrésy. Ice flow physical processes derived from the ers-1 high-resolution map of the antarctica and greenland ice sheets. *Geophysical Journal International*, 139(3):645–656, 1999. doi: https://doi.org/10.1046/j.1365-246x.1999.00964.x.

Ernesto Rodríguez, Yunjin Kim, and Jan M Martin. The effect of small-wave modulation on the electromagnetic bias. *Journal of Geophysical Research: Oceans*, 97(C2):2379–2389, 1992. doi:

https://doi.org/10.1029/91JC02511.

Shirzad Roohi. Capability of pulse-limited satellite radar altimetry to monitor inland water bodies. Master's thesis, 2015.

C Rush. Ionospheric radio propagation models and predictions–a mini-review. *IEEE transactions on antennas and propagation*, 34(9):1163–1170, 1986. doi: https://doi.org/10.1109/TAP.1986.1143951.

Remko Scharroo and John Lillibridge. Non-parametric sea-state bias models and their relevance to sea level change studies. In *Envisat & ERS Symposium*, volume 572, 2005. URL `https://pdfs.semanticscholar.org/1328/9c4e902fa1a3102ea2d281050dd220dab965.pdf`.

William S Schreiner, Robert E Markin, and George H Born. Correction of single frequency altimeter measurements for ionosphere delay. *IEEE transactions on geoscience and remote sensing*, 35(2):271–277, 1997. doi: https://doi.org/10.1109/36.563266.

Christian Schwatke, Denise Dettmering, Wolfgang Bosch, and Florian Seitz. Dahiti–an innovative approach for estimating water level time series over inland waters using multi-mission satellite altimetry. *Hydrology and Earth System Sciences*, 19(10):4345–4364, 2015. doi: https://doi.org/10.5194/hess-19-4345-2015.

Frédérique Seyler, Stéphane Calmant, Joecila Santos Da Silva, Daniel Medeiros Moreira, Franck Mercier, and CK Shum. From topex/poseidon to jason-2/ostm in the amazon basin. *Advances in Space Research*, 51(8):1542–1550, 2013. doi: https://doi.org/10.1016/j.asr.2012.11.002.

Laurence C Smith. Satellite remote sensing of river inundation area, stage, and discharge: A review. *Hydrological processes*, 11(10):1427–1439, 1997. doi: https://doi.org/10.1002/(SICI)1099-1085(199708)11:10<1427::AID-HYP473>3.0.CO;2-S.

P Thibaut, JC Poisson, E Bronner, and N Picot. Relative performance of the mle3 and mle4 retracking algorithms on jason-2 altimeter waveforms. *Marine Geodesy*, 33(S1):317–335, 2010. doi: https://doi.org/10.1080/01490419.2010.491033.

RT Tokmakian, PG Challenor, TH Guymer, and MA Srokosz. The uk eodc ers-1 altimeter oceans processing scheme. *International Journal of Remote Sensing*, 15(4):939–962, 1994. doi: https://doi.org/10.1080/01431169408954126.

Mohammad Javad Tourian. Controls on satellite altimetry over inland water surfaces for hydrological purposes. Master's thesis, 2012.

N Tran, Doug Vandemark, S Labroue, Hui Feng, Bertrand Chapron, HL Tolman, J Lambin, and N Picot. Sea state bias in altimeter sea level estimates determined by combining wave model and satellite data. *Journal of Geophysical Research: Oceans*, 115(C3), 2010. doi: https://doi.org/10.1029/2009JC005534.

Yu I Troitskaya, GV Rybushkina, IA Soustova, GN Balandina, SA Lebedev, AG Kostyanoi, AA Panyutin, and LV Filina. Satellite altimetry of inland water bodies. *Water Resources*, 39(2):184–199, 2012. doi: https://doi.org/10.1134/S009780781202008X.

John M Wahr. Deformation induced by polar motion. *Journal of Geophysical Research: Solid Earth*, 90(B11):9363–9368, 1985. doi: https://doi.org/10.1029/JB090iB11p09363.

DJ Wingham, CG Rapley, and H Griffiths. New techniques in satellite altimeter tracking systems. In *Proceedings of IGARSS*, volume 86, pages 1339–1344, 1986. URL https://www.researchgate.net/profile/Chris_Rapley/publication/269518510_New_Techniques_in_Satellite_Altimeter_Tracking_Systems/links/55b5e51a08aec0e5f436c3f7/New-Techniques-in-Satellite-Altimeter-Tracking-Systems.pdf.

Daixin Zhao. Generating water level time series from satellite altimetry measurements for inland applications. Master's thesis, 2018.

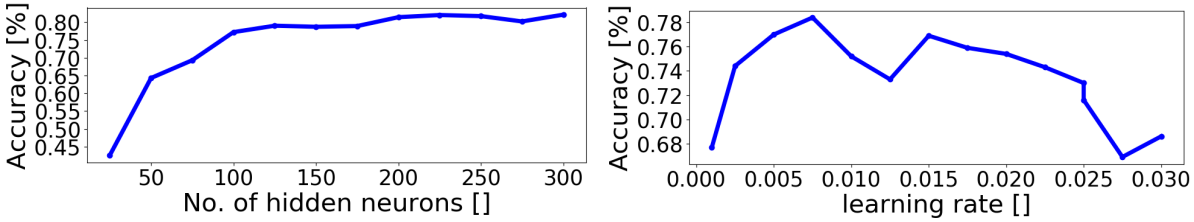
H.J. Zwally. Gsfc retracking algorithms, 2017. URL https://icesat4.gsfc.nasa.gov/radar_data/data_processing/gsfcretrackdoc.960725.php.

Appendix A

A.1 Parameter selection for the standard neural network

Neural network with 720 training waveforms

As done before, one will also at first focusing on the changes in the accuracy during the change of the hidden neurons and the learning rate:



(a) (b)

Figure A.1: *Accuracy of the neural network which is trained with 120 training data sets with different numbers of hidden neurons (a) and learning rate (b)*

A look at the development of the accuracy during the increase of the hidden neurons in Figure A.1 (a) shows, that it is very similar to the changes with 100 training data sets in Figure A.1 (a). Thus, the accuracy is increased until a certain number which is in this case around 200, which is very close to the number is the previous case which was 250. Therefore, it can be concluded, that because of the stagnation in the increase of the accuracy one can choose 200 hidden neurons. A look at the changes in the accuracy by changing the learning rate shows that this graph is not that smooth compared to the previous one in graph A.1 (b). However, the maximum peak is still distinct so that a learning rate of 0.0075 can be chosen, which is interestingly smaller than the previous one with 0.0125. In the last step, one can analyse which number of epochs should be chosen:

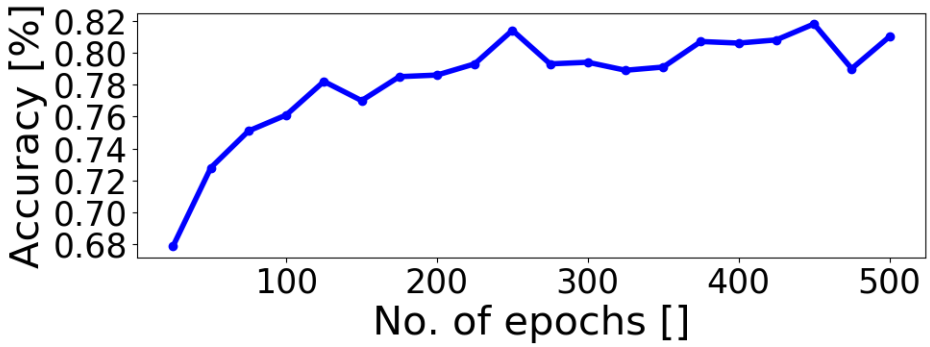
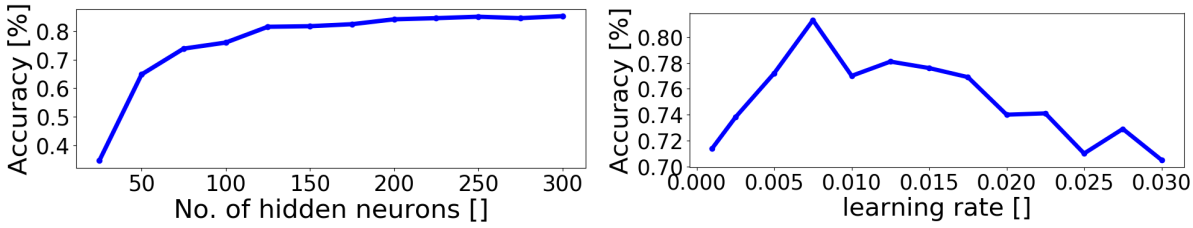


Figure A.2: *Changes in the accuracy by increasing the number of epochs with 720 training waveforms*

As it can be seen in Figure A.2, the result is, as before, not that smooth as in case of the hidden neurons. Interestingly starting at epoch 375 there is again a jump to a higher accuracy level which is over 0.8. However, comparing this result with Figure 4.9 it can be said that it shows more variations and a lower maximal value. Thus it can be concluded that there is no real improvement in this case, compared to the previous one with 600 training waveforms.

Neural network with 900 training waveforms

Now one can have a look at the last of the three training data sets which use 150 data sets which contain all in all 900 waveforms. Thereby, the schema of the previous analysis will be continued, and at first, the changes in the accuracy regarding the hidden neurons and learning rate will be analysed:



(a) (b)

Figure A.3: *Accuracy of the neural network which is trained with 150 training data sets with different numbers of hidden neurons (a) and learning rate (b)*

As it can be seen in figure A.3 (a), there is a smooth increase of the accuracy if the number of hidden neurons is increased. Interestingly, there is no real improvement compared to the previous tests. Thus the graph shows a very similar pattern as the previous graphs in which the accuracy stagnate around 0.8. Hence, a lower number of hidden neurons can be chosen to reduce the run time of the training, thus 250 hidden neurons can be chosen. The graph A.3 (b) shows a more distinct pattern compared to the previous one. In this case, a clear peak at a learning rate of 0.0075 can be seen, which is the same as in the previous test. However, the accuracy is even higher than in the previous tests which shows improvement. After this

number, the accuracy is steadily decreasing which is a comparable pattern as it was seen in (b). In the last step it can be tested if there is an improvement regarding the number of epochs.

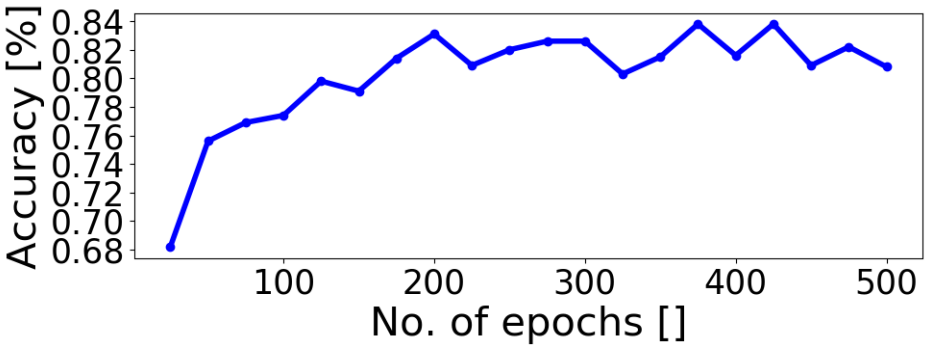


Figure A.4: *Changes in the accuracy by increasing the numbre of epochs with 900 training waveforms*

Also in Figure A.4 a small improvement can be seen compared to the previous tests. However, at the same time the accuracy is more fluctuating, but as done before, 425 epochs can be chosen. As a conclusion it can be said, that the increase of the number of training waveforms to 900 lead to an improvement in two of the three parameters, whereas the increase from 600 to 720 waveforms lead to no increase at all. However, for all three graphs of parameter similar pattern can be seen which shows, that the basic idea of the behaviour of the neural network towards the change of the parameter is correct. The comparison of the resulting waveforms can be seen in section 4.1.3.

A.2 Parameter selection for the neural network in the algorithm

Using a 10 bin window

At first, a window with ten input neurons is used to detect the right peak. Therefore, the following set of parameters is used for the analysing:

Table A.1: *Starting parameters for the neural network with 10 bin window size*

Parameter	Value
Hidden neurons	5
Learning rate	0.01
Epochs	10

As it can be seen in table A.2, the number of hidden neurons is just the half of the number of input numbers to see if it is already enough, whereas the number of epochs is also lower than the previous ones. The reason is that now the neural network has to learn fewer parameters so that it can be assumed, that it also needs less number of epochs. To see better the differences, the results with 600 and 900 training waveforms are shown at the same time. With this one can see if the 600 waveforms are already enough for this small neural network. As done before,

one will first have a look at the changes in the accuracy by changing the number of hidden neurons and the learning rate. In the Figures A.5 it is interesting to see, that the graphs for the hidden neurons are not smooth any more. Instead, they show more variations, mostly at the lower numbers of hidden neurons, where one also can see the maxima. Thus it is an interesting effect that the small neural network shows now a different behaviour compared to the previous bigger one. Also for the test with 900 waveforms for training a high accuracy for a high number of hidden neurons can be seen. There in Figure A.5 (b) a maximum of 93.1% is reached in the case in which 900 waveforms are used, which could be reached by using ten hidden neurons. Whereas in the other in Figure A.5 (a) case a maximum with five hidden neurons reaches an accuracy of 85.2%. Hence, one can already see an improvement. Next, the learning rate will be analysed which looks more similar to the previous ones. Here one can see in both cases one clear maximum peak which is for the first example in Figure A.5 (d) with 600 training waveforms with an accuracy of 95% with a learning rate of 0.0075. In the second case in Figure A.5 (e) a maximum accuracy of 95.6% is reached with a learning rate of 0.005. Again it can be seen that the accuracy is also improving by using this approach. However, the comparison shows that there is only a slight improvement between the two training data sets. Now one can have a look at the changes of the accuracy by increasing the number of epochs. It is interesting to see in Figures A.6, that now also a very different pattern can be seen. It can be assumed that in Figure A.6 (a) the effect of overfitting can be seen well, by a drop in the accuracy after around 90 epochs. Because now a smaller number of weights are used, the effect appears faster. In the other case with 900 waveforms in Figure A.6 (b), the same effect can be seen. This can also be assumed to be an overfitting effect. In the first case with 600 training waveforms, the maximum occurs with 55 epochs and reaches an accuracy of 85.2%, whereas in the second case a maximum after ten epochs can be seen which results in an accuracy of 94.9%. Next, the results for the case with 30 input neurons will be analysed.

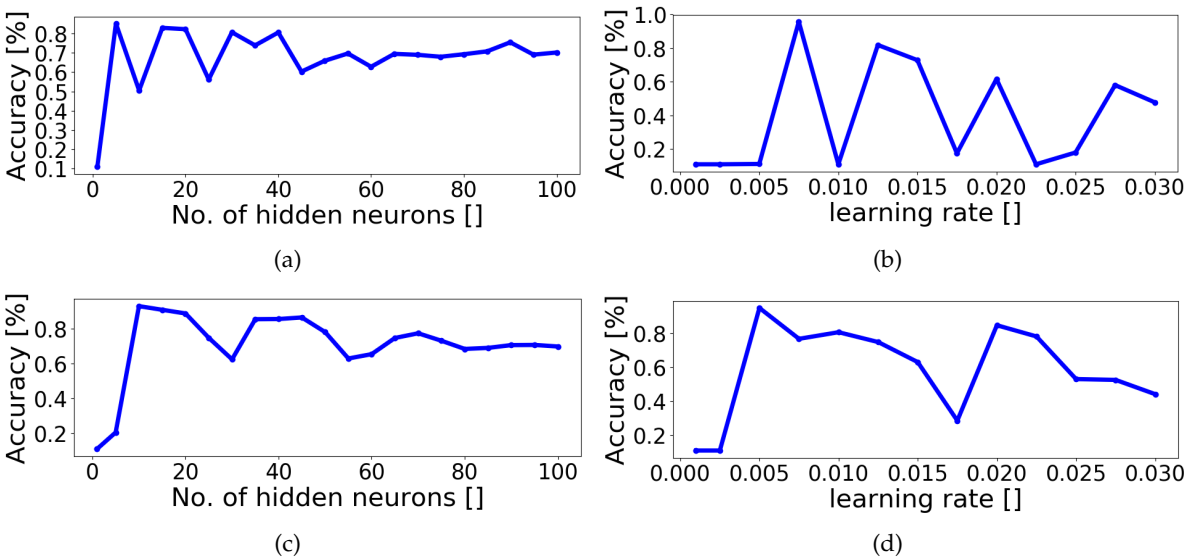
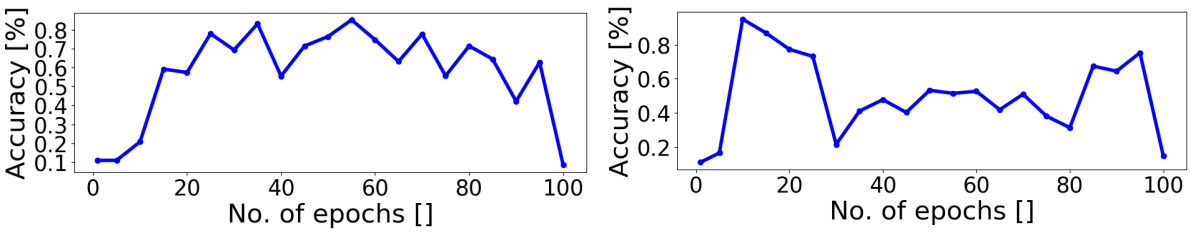


Figure A.5: *Accuracy of the neural network which uses ten input neurons with different numbers of hidden neurons with 600 (a) and 900 (c) waveforms respectively and also the learning rate with 600 (b) and 900 (d) waveforms respectively*



(a) (b)

Figure A.6: *Accuracy of the neural network which uses ten input neurons by increasing the number of epochs for 600 (a) and 900 (b) waveforms respectively*

Using a 30 bin window

In this case, the same standard parameters than in the previous section are used which can be seen in table A.2. With this, the results are more comparable. This can be justified because the number of input neurons is just increasing with 20 neurons, compared to the 104 input neurons, it is a small change. So in a first step the changes of the accuracy by increasing the hidden neurons and the learning rate are shown. As it can be seen in the Figures A.7 the accuracy increased again and reached in all four cases values close to 95%. Thus it can already be said, that using 30 bins improves the results of the tests. Also interesting is the fact, that the graphs of the hidden neurons are now more smooth than the previous ones. However, they show a different pattern than the ones of the big neural networks. In the case of the 600 waveforms for testing in Figure A.7 (a) it can be seen that the accuracy is decreasing after a while, but in the end it is increasing again. In that case, a maximum of 30 hidden neurons with 94.9% is visible. In the other case with 70 hidden neurons in Figure A.7 (b), an accuracy of 96.3% is reached which shows improvement. Striking is the fact, that the accuracy is now steadily increasing if the learning rate is also increasing, which is the opposite to the previous observations. In the previous cases, a clear peak could be seen which indicated the best learning rate for the neural network. In the first case in which 600 training waveforms are used and that can be seen in A.7 (c), the accuracy of 97.5% is reached with a learning rate of 0.0575. In the second case in A.7 (d), an accuracy of 96.2% is reached which a learning rate of 0.0625. Now the changes in the accuracy by increasing the number of epochs are shown as the last test. Figure A.8 shows, that as in the previous cases the accuracy is increasing at the same time as the number of epochs is increasing. However, it can be seen that the two approaches are close together. By using the smaller training set in Figure A.8 (a), an accuracy of 94.7% is reached by using either 70 or 85 epochs. This is a significant increase in the accuracy compared to the smaller network with just ten input neurons. In the case of the bigger training set in Figure A.8 (b), nearly the same accuracy with 94.6% is reached by using 95 epochs.

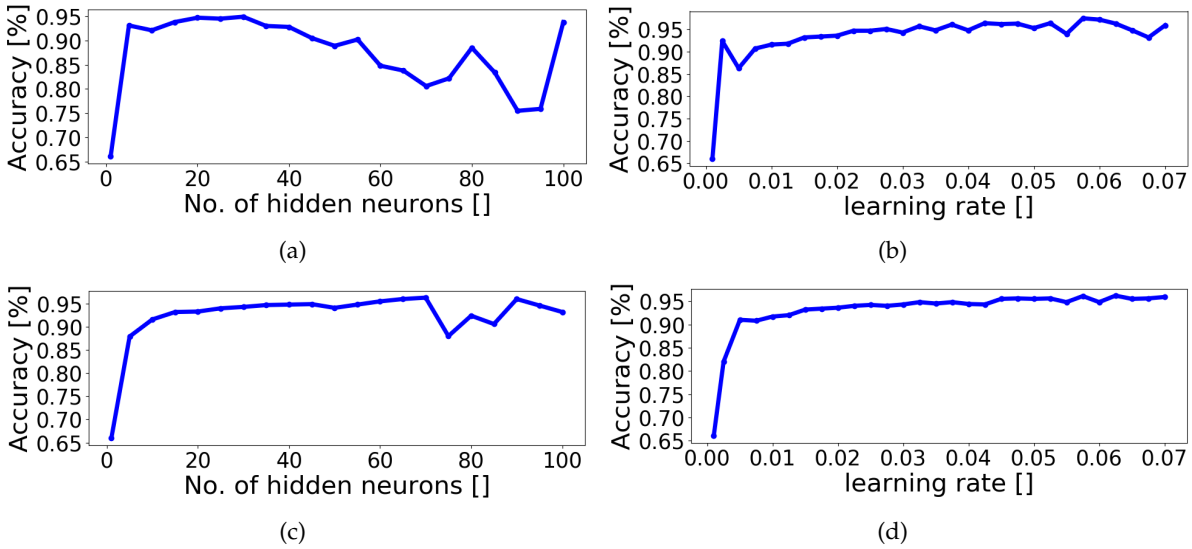


Figure A.7: *Accuracy of the neural network which uses 30 input neurons with different numbers of hidden neurons with 600 (a) and 900 (c) waveforms respectively and also the learning rate with 600 (b) and 900 (d) waveforms respectively*

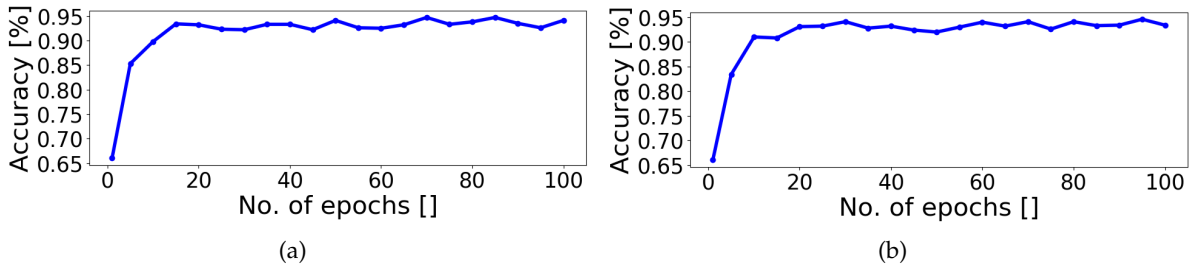


Figure A.8: *Accuracy of the neural network which uses 30 input neurons by increasing the number of epochs for 600 (a) and 900 (b) waveforms respectively*