Machine learning based time-series postprocessing for the interferometric SAR remote sensing data

Adrianna Kaźmierczak

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Machine learning based time-series postprocessing for the interferometric SAR remote sensing data

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

Adrianna Kaźmierczak

August 18, 2021

Faculty of Mechanical, Maritime and Materials Engineering (3mE) and Faculty of Civil Engineering and Geosciences (CITG) · Delft University of Technology

Source of the cover image: https://www.nasa.gov/image-feature/southeastern-europe-italy-and-into-the-mediterranean-sea, retrieved on 27.04.2018

Abstract

A satellite remote sensing technique, Interferometric Synthetic Aperture Radar (InSAR), is able to provide surface displacement information on a millimeter level. In this study, data from the TerraSAR-X satellite collected in the years 2009-2018 over the area of Amsterdam is used. Even though radar data is a subject to multi-step processing, there are still several problems observed that can make the interpretation of the time-series difficult for users who are not experts in the radar remote sensing field. In this study we focus on unwrapping errors, partial decorrelation, and incorrectly fitted models. The unwrapping errors are handled as outlier detection problem and the rest as a time-series segmentation task.

In order to address these issues, a data-driven approach is proposed. We show a method to detect unwrapping errors based on spatially neighboring points. A GUI is developed to collect expert knowledge in a form of assessing the time-series correctness, marking unwrapping errors, and dividing time-series into separate segments. This information is later used in the evaluation of several outlier detection and segmentation algorithms. We propose a supervised learning approach based on neural networks in order to use expert knowledge. Due to not enough labelled data available, a simulation is developed and used for training of the networks. We present two different approaches, one based on multi-label classification and one on binary classification. For each of them fully-connected neural networks and convolutional neural networks are compared.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to greatly thank Prof. Hanssen for accepting me as a thesis candidate from a different Faculty and for introducing me into the Radar Group. It was a valuable chance for me to learn and grow. Thank you for sharing your knowledge and believing in me.

I would like to express my gratitude to Prof. Hellendoorn for being my supervisor and always offering help and encouragement.

I am incredibly grateful to you both for not giving up on me.

Thank you to my parents for making this whole Dutch adventure possible. Additionally, I would like to thank Dominik Jargot for his endless and unconditional support, and last but not least, many thanks to Coen Hutters for being the best study partner and a true friend.

Delft, University of Technology
August 18, 2021

Adrianna Kaźmierczak

To Grandpa,
who taught me the beauty of science and nature.

Chapter 1

Introduction

In the last decades, remote sensing techniques have become a more and more popular source of information about the Earth. Nowadays, numerous missions have a task of observing and collecting measurements for various applications, such as atmospheric research, oceanography, ice cover or vegetation monitoring, and many others. The missions can use either satellites or aircraft as a carrier for different kinds of sensors that provide proper measurements.

Among different imaging techniques, SAR (Synthetic Aperture Radar) imagery has found numerous applications. One of the methods using SAR images is InSAR (Interferometric SAR)—a technique that can provide height and deformation measurements. It is used in research related to, e.g. volcanoes, land subsidence and uplift, glaciers or ice motion [1]. Among the currently used processing methods there is still room for development and improvements, especially regarding automation of the processing task. Moreover, various additional information sources are available that could potentially be beneficial. The InSAR technique is a focal point for the further research and thereby the main motivation for this work.

Processing of the acquired data, i.e. parameter estimation, requires dedicated algorithms and expert knowledge. The amount of data is growing rapidly, which makes its analysis impossible to be performed only by people, as the work is laborious and time consuming. Despite advanced processing methods, by inspection, one can find that there are still some obvious errors present, such as unwrapping errors, i.e. measurements shifted due to ambiguities caused by the angular nature of the signal. They are easily spotted by a human expert, however, currently used algorithms are unable to detect these problems at all times.

Another aspect is that as nowadays a great number of measurements of environmental factors is available, which can be used to enhance the analysis of the observations. There are historical records of e.g. temperature, precipitation, and the ground water table for numerous locations. By making use of these data, it may be possible to find correlations between the radar measurements and other signals, what can be a sign of a causality or at least an indication that there is some relationship between the behaviour of a scatterer point and some considered environmental factors. This kind of knowledge could be achieved using machine learning or data mining techniques.

Recently, the machine learning, and especially deep learning, approach has been applied in multiple remote sensing research areas, however, most of them are related to image processing. Among these applications there are image preprocessing, such as denoising or sharpening, classification, target recognition, or scene understanding [2, 3].

1-1 Problem formulation and research goal

Following from the above, a contribution to the existing methods could be achieved by designing a system which would be able to spot unwrapping errors in the InSAR time-series and correct them accordingly. The machine learning approach is not widely applied in the field of InSAR processing and hence, this creates a possibility to investigate how this field could benefit from this new approach.

Among many challenges in the InSAR data postprocessing, the ones presented below will be considered in this thesis. A more detailed explanation, together with background information, can be found in Section 3-2.

Problem 1 - Single unwrapping errors and unwrapping shifts

Despite pre-processing of the data to unwrap the raw measurements, often there are still multiple unwrapping errors present in the data (see Sections 2-2 and 3-2-1). One can observe either single, individual unwrapping errors occurring during one epoch or unwrapping shifts affecting a number of consecutive epochs. This is pictured in the Figure 1-1. This error causes further problems with fitting a model and, in general, in the assessment of the underlying process and situation.

Problem 2 - Partial temporal decorrelation

Some of the time-series also exhibit partial temporal decorrelation (see Section 2-2 and Section 3-2-3). This means that some part of the time-series does not actually yield any valuable information and, if the series is treated as a whole, this only distorts the general behaviour. This can be seen in Figure 1-2. Getting rid of these segments, or just ignoring them, can allow for better modelling and analysis of the behavior. On the other hand, the partial decorrelation can also be used as an indicator itself that there have been some changes in the scattering characteristics for the observed point (location), which is relevant information.

Problem 3 - Incorrectly fitted models

There is no “one size fits all” model that could optimally represent the behaviour of all scatterers in the observed area. Misfits are often due to the previously mentioned problems with unwrapping errors and partial decorrelation. Moreover, after analysis of many time-series one can notice that it is not uncommon that there are cases where in one time-series it is possible to distinguish different behaviour of the scatterers (see Section 2-1). Taking this fact into account would allow to avoid fitting one model through the whole timespan and hence, to fit better models and to look for relations with other factors only in some of the segments. An example is given in Figure 1-3.

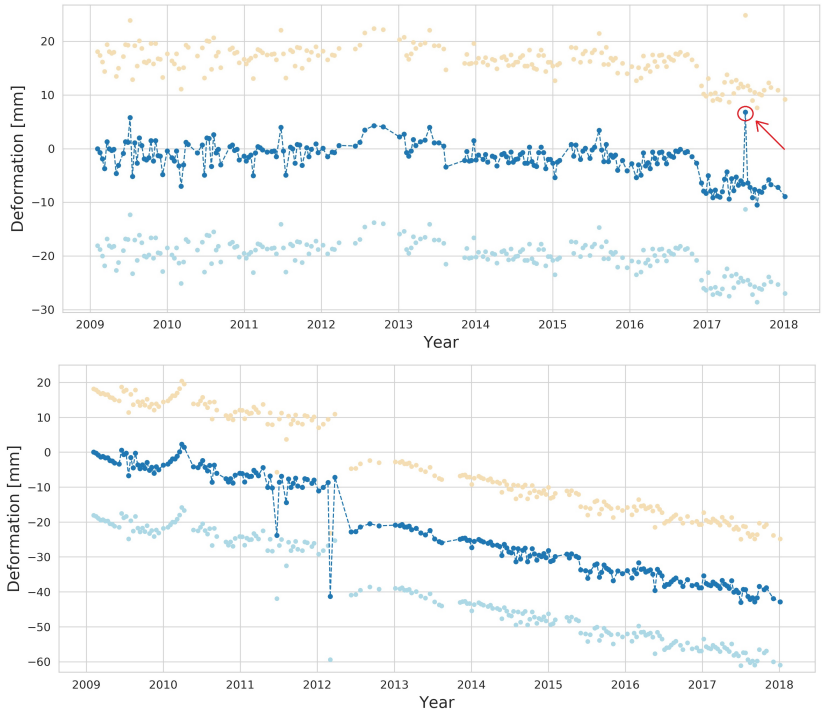


Figure 1-1: Examples of Problem 1. The plots present measurement series of the TerraSAR-X mission with $\lambda = 31$ mm (middle curve, dark blue) and two additional series, which are shifted copies of the original series corresponding to $\pm 2\pi$ shifts (see Section 2-2). It is possible to spot that in the top figure there is a high likelihood that one of the points in mid-2017 has been misplaced—a *single unwrapping error*. Similarly, in the bottom figure, apart from two potential single unwrapping errors, one could notice a shift of the whole remaining series after early 2012 (the upper series seems to be a better continuation)—an *unwrapping shift*.

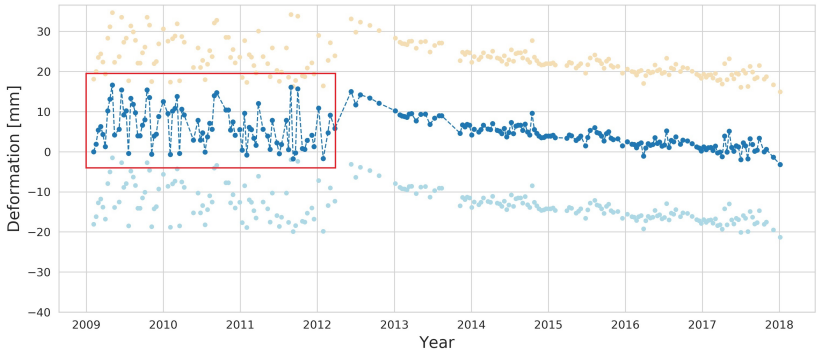


Figure 1-2: Example of Problem 2. On the left side, the measurements from 2009 till early 2012 seem to be random and follow what appears to be a uniform distribution (see Section 3-2-3), they basically fuse with the $\pm 2\pi$ copies. This is an example of a *partial temporal decorrelation* and this part should not be used in model fitting.

Research question

The main research question of this work has been formulated as follows:

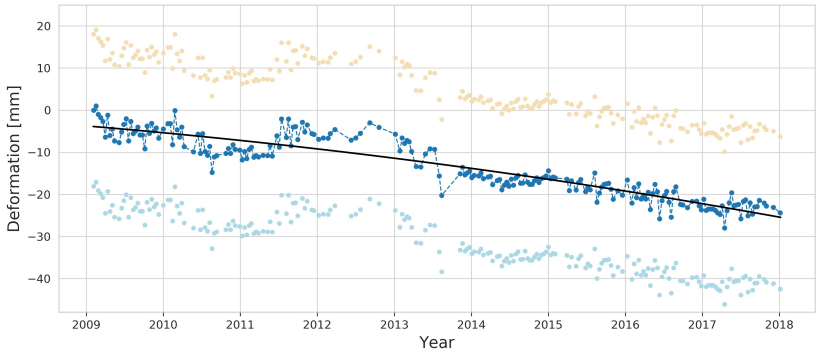


Figure 1-3: Example of Problem 3. The black curve is a model fitted by default. As one can see, the model does not reflect the behaviour of the time-series, as it has periods of different behaviour (e.g. mid-2011 till mid-2013, between 2014 and 2015, and from 2015 till 2018) and the model does not account for that, it is an *incorrectly fitted model*.

How can a machine learning approach be used to reduce the errors of types 1, 2, and 3 in InSAR postprocessing?

To investigate this in more detail, several subquestions have been posed:

1. Why are there unwrapping errors present? How to decide whether it is an unwrapping error or an outlier?
2. There are multiple works addressing time-series analysis that might serve useful in this application. Which algorithms perform well in the tasks of unwrapping error detection and time-series segmentation in the InSAR data context?
3. Raw measurement data may not be the most efficient input for a machine learning algorithm. How to preprocess the time-series to be the most useful for the machine learning algorithm? What features could improve the performance?
4. There are multiple algorithms available within the current state of the art in the machine learning field. Moreover, within one algorithm various parameters have to be adjusted. Which machine learning approach and architecture is the most suitable for the application and why?
5. Points within close proximity can be helpful in detecting specific events in the neighboring points. How to include the spatial context into the unwrapping errors detection and into segmentation task?

In order to answer this question, the following tasks have been defined:

1. Analyze the data and prepare an extensive simulation.
2. Perform a literature study to find methods that can be used for the unwrapping detection and segmentation tasks. Experiment with the found, non-machine learning approaches.
3. Find and implement a suitable machine learning approach to detect the potential unwrapping errors and shifts, and correct them accordingly.
4. Segment the time-series based on different characteristics.

1-2 Outline of the thesis

An introduction to radar remote sensing is given in Chapter 2. Basics of SAR remote sensing technique are presented, as well as basics of the SAR Interferometry (InSAR). Also, a brief description of a specific technique within InSAR, namely, Persistent Scatterer InSAR (PS-InSAR) is presented, as it is the source of the data used in this thesis.

The analysis of the considered time-series is performed in Chapter 3. At first, the used dataset is introduced shortly. Then, different characteristics and observations of the data are presented. This is followed by the suggestions about what could be done or improved. Lastly, available contextual information is discussed, namely, spatial context and the environmental factors.

Theoretical introduction to the knowledge representation and expert knowledge incorporation is provided in Chapter 4. Moreover, the chapter contains a detailed description of the technique used for the data collection (expert knowledge data).

Subsequently, several algorithms for outlier detection and segmentation are tested and discussed in Chapter 5. For the outlier detection such methods as moving average, exponential smoothing, or ARIMA are presented. Experiments in segmentation involve probabilistic approaches, as well as BFAST and piecewise linear representation techniques.

Machine learning basics and the architecture design are described in Chapter 6. Several approaches are used for building a machine learning solution and then tested, compared, and discussed.

Chapter 2

Radar remote sensing

Nowadays, numerous Earth-observing satellites orbit the planet and perform various kinds of measurements in order to explore and study miscellaneous aspects of Earth. The remote sensing techniques can be divided into active and passive, based on the way the measurements are acquired. Radar remote sensing is an active sensing technique. The radar illuminates the target with microwave radiation and collects the reflected signal. On the other hand, passive sensors can only register the signals emitted by the Earth's surface or atmosphere (e.g. a microwave sensor) or detect radiation reflected from the surface (e.g. an optical sensor). One of the advantages of radar being an active sensor is its ability to penetrate through the cloud cover and to operate at night.

The general principle of how the radar works is as follows. The radar is sending electromagnetic wave pulses in the direction where the transmitting antenna is pointing, the signal hits a target and backscatters into the receiving antenna. In case of a monostatic system there is only one antenna which is both transmitting and receiving. The frequency with which the pulses are emitted is the PRF (pulse repetition frequency) and it determines the size of a resolution cell of the obtained image. What is important, is that a radar is a distance measuring instrument and hence, all targets within the same range are treated as one (they all end up in the same resolution cell).

There are different types and forms of radars and the one that is particularly suitable for Earth observation is the synthetic-aperture radar (SAR).

In this chapter, a brief introduction to radar remote sensing is given. At first, some basic information about SAR imaging and then, about the SAR Interferometry, is presented. Later on, more details are given about the Persistent Scatterer InSAR technique, including its processing steps and the resulting data characteristics.

2-1 SAR remote sensing

In radar technology, the resolution in the azimuth direction is proportional to the size of the antenna. Therefore, the resolution is constrained by the construction capabilities of a physical

antenna. A solution to this problem is SAR imagery (Synthetic Aperture Radar). The principle there is to use a moving antenna (space- or airborne) and combine the acquisitions, thereby creating a "synthetic" antenna of a much larger size [4], see Figure 2-1.

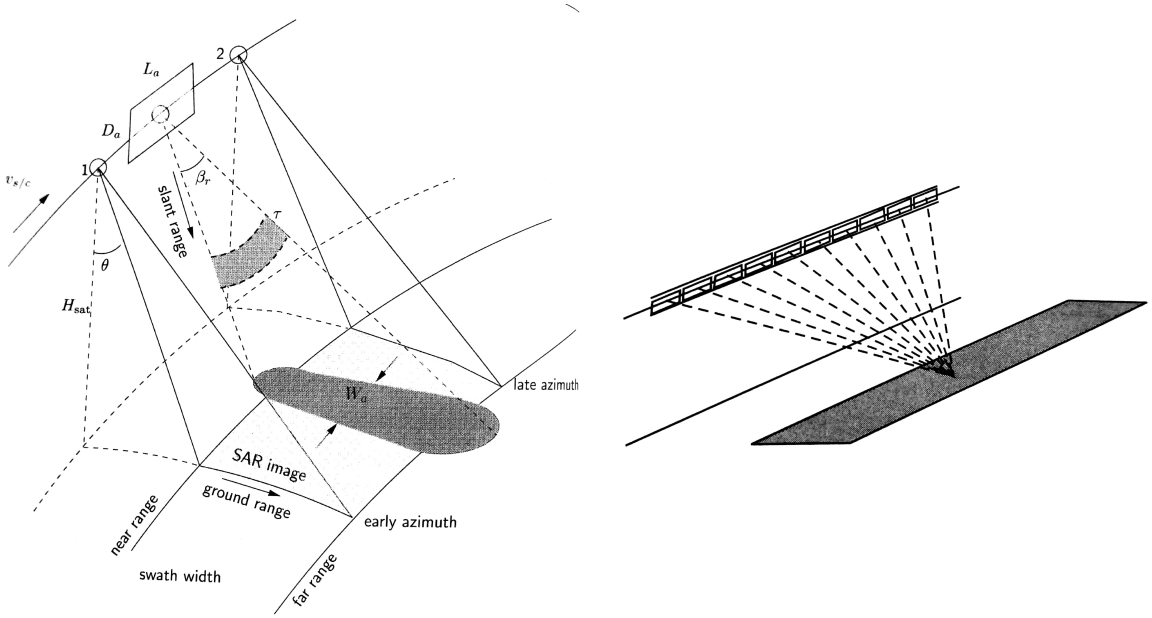


Figure 2-1: SAR observation geometry [1] (left) and the synthetic aperture principle, combining images to obtain an image as of it would be acquired using a very long antenna [5] (right).

The satellite illuminates a strip of the ground (also called swath) and moves in the azimuth direction creating a strip map. There are also other acquisition modes available, such as ScanSAR and spotlight SAR, but the Stripmap mode is the most popular one, due to its fine azimuth resolution [1]. The antenna is mounted in a side-looking configuration, i.e. it illuminates the surface with a non-zero off-nadir angle. Such geometry provides range sensitivity without ambiguous reflections. However, this causes also geometric distortions due to specific characteristics of the illuminated terrain, see Figure 2-2. A positive slope in the direction towards the radar is much shorter in the line of sight than on the ground, this is called foreshortening (A in Figure 2-2). Additionally, the foreshortened areas are imaged as brighter as the backscattering area there is much larger, so there is more power in the returning signal. Layover (B) happens when the slope is steeper than the off-nadir angle and the top is registered earlier than the bottom, the order is reversed and the contribution in the image is mixed with other areas. The shadow (C) occurs if the area is occluded, the beam is not able to illuminate it and no signal is backscattered [4, 1, 5].

The satellites bearing the SAR instrument orbit the Earth on almost polar orbits and collect images in ascending (south to north) and descending (north to south) passes. Due to the aforementioned distortions, observing an area from two different perspectives may result in different images which are complementary [4].

The radars can operate in different frequency bands. Depending on the wavelength of the emitted signal, the radar system is suitable for different applications. For instace, the X-band, with the shortest wavelength of around 3.1 cm is used in urban areas surveillance. Both X- and C-band have too short wavelength to penetrate significantly through canopies

and vegetation, this is, however, possible with the L-band with a longer wavelength of around 23 cm [6].

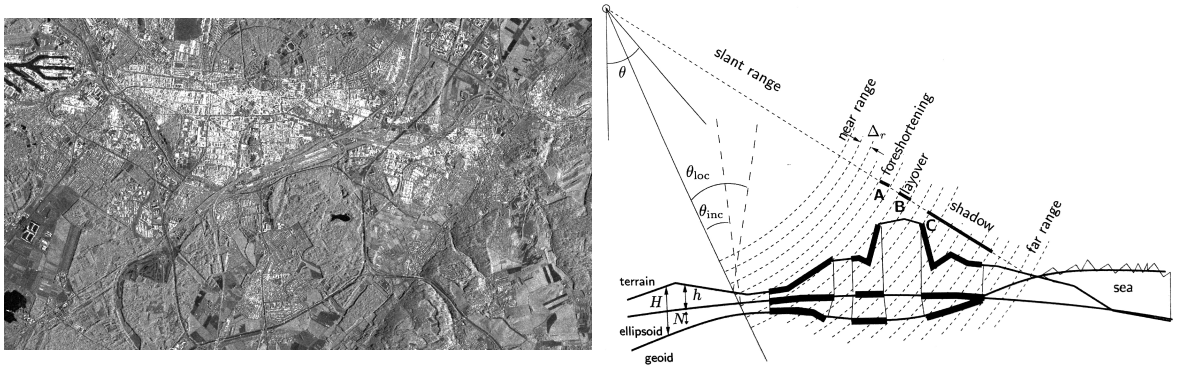


Figure 2-2: SAR image example [7] (left) and geometric distortions, i.e. foreshortening, layover and shadow, with different reference surfaces (ellipsoid or geoid). [1] (right)

The Figure 2-2 shows an example of a SAR image. One of the data types provided by the space agencies is the Single-Look Complex (SLC) [8]. Each pixel is associated with a complex value, representing the amplitude and phase information. Pixel size depends on the azimuth PRF and the range sampling frequency.

The amplitude and phase in each pixel are a superposition of signals from all the scatterers within the corresponding resolution cell. Therefore, the phase information follows a uniform distribution and does not yield any meaningful information [1]. Moreover, the fact that there are multiple scatterers in each cell and they can often behave in a random way (e.g. grass in a field) is responsible for an effect called speckle, which causes a radar image to look grainy and noisy. A way to reduce this effect is, for instance, averaging multiple images of the same area or filtering during the processing [5]. One can distinguish two types of scatterers. One type are point scatterers, where a single object dominates the reflected signal, and the other one are distributed scatterers, where multiple objects contribute to the reflection [1].

2-2 Interferometric SAR

The SAR system is measuring only distance in the line of sight direction. In case of two objects located at different heights (e.g. object at height H in Figure 2-3 and any point at the same range), the distance measurements are the same and the points are indistinguishable. However, a satellite can observe the same area from slightly different angles. Consequently, using this angular differences, with two images it is possible to retrieve information about the height. There are two configurations in which the images can be acquired. It can be a single-pass interferometry, where there is one carrier with two antennas, or a repeat-pass, where the same area is observed after days, months, or even years. The spatial separation between the orbit positions of a satellite during both acquisition is called a spatial baseline. Similarly, the temporal baseline defines the temporal separation between acquisitions.

Each SAR image consists of a grid of complex values, which can be represented by their

amplitude and phase as follows:

$$y_1 = |y_1|e^{j\psi_1}, \tag{2-1a}$$

$$y_2 = |y_2|e^{j\psi_2}, \tag{2-1b}$$

Then, the interferogram is formed by complex multiplication of corresponding values:

$$I_{12} = y_1 y_2^* = |y_1||y_2|e^{j(\psi_1 - \psi_2)} \tag{2-2}$$

This is preceded by the co-registration of the images. As two different acquisitions are made with slightly different geometries, they have to be aligned and resampled in order to be comparable and compatible.

In order to assess the accuracy of the interferometric phase, the coherence can be used as a measure. The coherence is defined as follows:

$$\gamma = \frac{E\{y_1 y_2^*\}}{\sqrt{E\{|y_1|^2\}E\{|y_2|^2\}}}, \qquad 0 \leq \gamma \leq 1 \tag{2-3}$$

where y_1 and y_2, are zero-mean circular Gaussian variables. For calculating the expectation values several interferograms obtained under the same circumstances are required. However, in each SAR acquisition every full-resolution pixel is registered only once [1]. Therefore, Eq. (2-3) is often replaced with an estimator of the coherence magnitude using a spatial average over a small window of N pixels, instead of ensemble average, provided that all the deterministic phase components are taken into account [1, 4, 9]:

$$|\hat{\gamma}| = \frac{|\sum_{n=1}^{N} y_1^{(n)} y_2^{*(n)}|}{\sqrt{\sum_{n=1}^{N} |y_1^{(n)}|^2 \sum_{n=1}^{N} |y_2^{(n)}|^2}} \tag{2-4}$$

Phase contributions

The interferometric phase is a difference in the phase between two SAR acquisitions:

$$\phi = \Delta\psi = \psi_1 - \psi_2 \tag{2-5}$$

InSAR phase measurements are a combination of multiple factors. The signal is influenced by effect such as ground deformation, atmospheric conditions, or electrical properties of the scatterer, in particular [11]:

$$\phi = \phi_{\text{flat}} + \phi_{\text{topo}} + \phi_{\text{defo}} + \phi_{\text{orbit}} + \phi_{\text{tropo}} + \phi_{\text{iono}} + \phi_{\text{scat}} + \phi_{\text{noise}} \tag{2-6}$$

The "flat" earth phase ϕ_{flat} results from the assumption of the reference shape of the Earth made during the processing of the images that can be accounted for based on satellite orbits. It is related to the parallel baseline (see Figure 2-3):

$$\phi_{\text{flat}} = -\frac{4\pi}{\lambda} B_{\parallel} \tag{2-7}$$

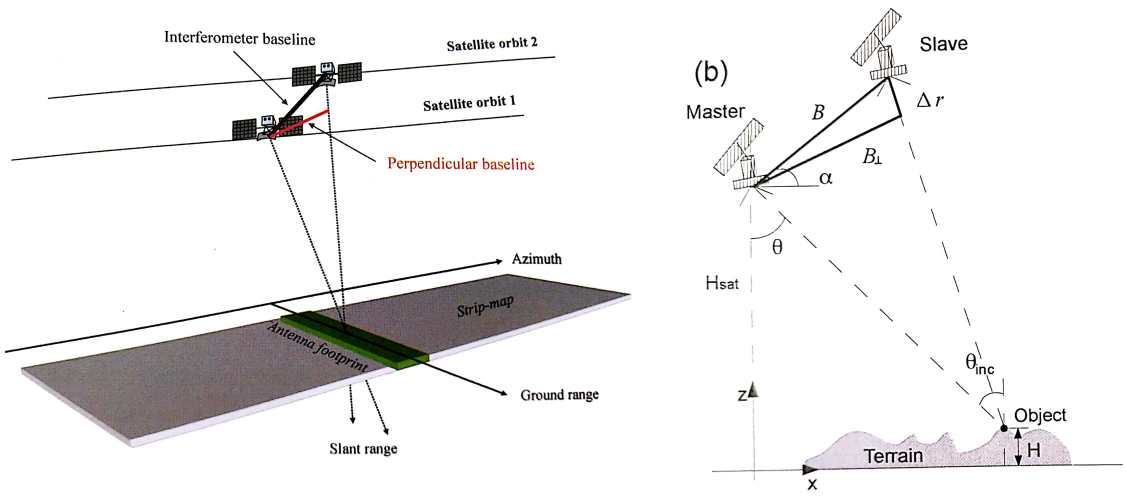


Figure 2-3: Geometry of a repeat-pass satellite interferometric SAR system [4] (left) and [10] (right).

The topographic phase ϕ_{topo} is related to the terrain shape above the reference ellipsoid (see Figure 2-2). It is related to the perpendicular baseline:

$$\phi_{topo} = -\frac{4\pi}{\lambda}\frac{B_{\perp}}{R\sin\theta}H_p \tag{2-8}$$

where R is the range to the target, θ is the look angle, and H_p is the height of the point.

The phase component due to ground deformation ϕ_{defo}, which is measured in the line-of-sight direction, is

$$\phi_{\text{defo}} = -\frac{4\pi}{\lambda}D_p \tag{2-9}$$

where D_p is the surface displacement.

The spatial baseline is based on the orbit information, hence, any errors in this information can cause phase errors ϕ_{orbit}. Furthermore, atmospheric effects also affect the phase measurement, such as the phase delay occurring in the troposphere ϕ_{tropo} due to refractivity and water content. Additionally, especially for longer wavelengths, ionospheric total electron content (TEC) can influence the interferometric phase measurement ϕ_{iono}.

Moreover, electrical properties of the scatterer also contribute to the interferometric phase. They are, however, often assumed negligible for the studies concerning ground deformation or topography [11].

The noise term ϕ_{noise} expresses the noise factors coming from different decorrelation sources.

Decorrelation

One can distinguish several decorrelation sources, such as spatial baseline decorrelation, Doppler centroid decorrelation, thermal or system noise, or temporal decorrelation [1, 11].

The spatial baseline decorrelation is related to the geometry of the acquisitions. It is caused by the difference in the incidence angles between two SAR images.

The Doppler centroid decorrelation comes from the difference in Doppler centroid frequencies between two SAR acquisitions. It can be a result of different squint angles or the convergence of the orbits.

Thermal noise is present as an inherent characteristic of a radar system.

Temporal decorrelation happens when the physical properties of scatterers within a resolution cell change over time. Examples can be human activity such as farming or contructions sites, as well as natural processes such as vegetation growth.

Phase ambiguity and unwrapping

The observed interferometric phase is an ambiguous measurement. It is a relative phase, which is wrapped in the interval $[-\pi, \pi)$ and expressed by

$$\phi^w = W\{\phi\} = \text{mod}\{\phi + \pi, 2\pi\} - \pi \tag{2-10}$$

where W is the wrapping operator and ϕ is the unknown true phase, as in Eq. (2-6). The task of retrieving the unknown absolute phase from the wrapped measurement is one of the main challenges in radar interferometry due to its non-uniqueness and non-linearity [1, 10].

There are several methods to approach this problem. To solve for the unwrapped phase value, some additional information, such as a priori knowledge of the terrain, or assumptions are required. Usually, to impose a certain degree of smoothness, an assumption is made that differences between adjacent pixels do not exceed half a cycle, i.e. phase gradient is in the interval $[-\pi, \pi)$. If the observed differences are bigger, a cycle is added or subtracted accordingly. While this procedure works for a signal with small gradients and with low noise level, any mistake results in propagating the error [1, 4]. Among more complex, conventional methods are residue-cut (or branch-cut) method, least-squares approach, or minimal cost flow methods [1, 11].

Differential interferometry

Differential interferometry provides additional information compared to a single interferogram and allows to study the deformation signals. For that purpose two interferograms are required. One is used to obtain the topography and the other one to observe the deformation. The interferograms can be obtained either in a three-pass or a four-pass setting. In the former, there is one common reference image and two others are used to create interferograms with the reference one. In the latter, two separate interferograms are created and then, the topographic pair is subtracted from the deformation pair. What is important, from the differential pairs not only displacement but also changes in the atmospheric phase component can be observed.

2-3 Persistent Scatterer InSAR

The temporal and spatial decorrelation are one of the main difficulties in differential InSAR. Therefore, a method focusing on identifying scatterers which are coherent over long time

periods. Such coherent scatterers are called Persistent (or Permanent) Scatterers (PS) and this multi-epoch InSAR technique is called Persistent Scatterer Interferometry (PSI) [4, 10]. The technique is multi-epoch, a stack of multiple SAR images is analyzed together. The PS are abundant mainly in urban environments, where scatterers are "stable" and with good reflectivity. All images are co-registered on a chosen master image and a set of interferograms is generated. There are several processing steps to obtain PS points and they are shortly presented below. More detailed information can be found in [1, 10].

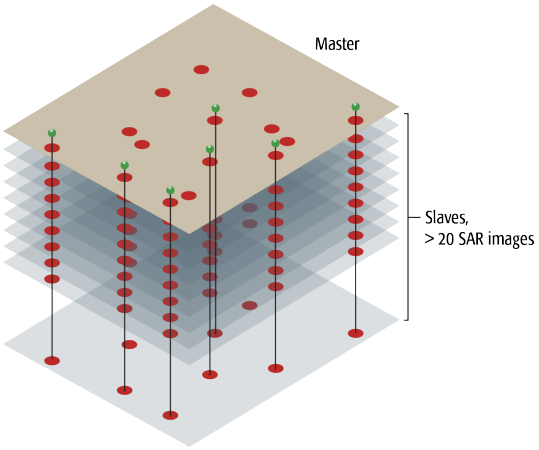


Figure 2-4: An illustration of a stack of SAR images with scatterers coherent over multiple time intervals [12].

2-3-1 Processing steps

At the very beginning a master image for the whole stack has to be selected. This is performed by maximizing the stack coherence of a batch of interferograms. The next step is to select a preliminary set of constantly coherent pixels, the so called PS candidates. At that point, the PS phase cannot be used as it still contains unknown signals. Hence, the amplitude is used to assess whether a PS is potentially coherent, under an assumption that high and almost constant amplitude implies low phase dispersion. The normalized amplitude dispersion,

$$D_a = \frac{\sigma_a}{m_a}, \tag{2-11}$$

where σ_a is the standard deviation and m_a is the mean of the amplitude values. Typically, the pixels below $D_a = 0.25$ are selected. Subsequently, a *reference network* of the selected PS candidates is constructed using, for instance, Delaunay triangulation. The phase differences between the connected points are called *arcs*. The arcs are unwrapped in time and integrated in space. Later on, further estimation is performed on the unwrapped arc phase in order to reduce the orbital inaccuracies and the atmospheric signal in each interferogram. In order to account for the atmospheric component of the signal an assumption is used, that the atmospheric phase is spatially correlated and temporally uncorrelated. After removing the estimated orbital and atmospheric influence, a second round of selecting the PS candidates is performed. New candidates increase the density of the network and new arcs are added to the network. All PS candidates are re-evaluated and points which occurred to be noise are

discarded from the network. After the final estimation of the displacement parameters, the PS points are geocoded [10]. As a result, a deformation time-series for each PS point is derived.

In this study, the deformation time-series obtained from the PSI processing technique are analyzed and post-processed. Especially, different inaccuracies in time-series related to handling the phase unwrapping are presented.

Chapter 3

Characteristics of the data

3-1 Dataset description

The raw data is acquired by multiple satellites observing the Earth, such as Sentinel-1, ERS, and TerraSAR-X. The measurements are geolocated, i.e., the location of each point scatterer is mapped onto an earth-centered, earth-fixed reference system. The available measurements are also initially pre-processed).

The primary dataset that is used in this work[1] comes from TerraSAR-X measurements over the area of Amsterdam. The satellite pass direction was ascending and the incidence angle was 31.1°. The timespan covered is from 05-02-2009 until 05-01-2018 with the measurement interval of 11 days. There are 219 samples in each time-series. The dataset is divided into 8124 'low' and 14999 'high' points, based on the initial height of the points in the DEM, hence, the whole size of the dataset consists of 23123 time-series. In this work, the time-series are denoted as AL### and AH### (*ascending low* and *ascending high*), where AL and AH indicate whether the time-series belongs to the 'low' or high' points subset and the number that follows is the number from the corresponding subset.

Other information that can be found in the dataset for each scattering point are: the point id, the position in latitude and longitude, the position in the Dutch RD-system, the height of the point in the DEM (provided by AHN), the residual height of the point measured, the estimated linear fit for all measurements, and the quality of the point, which is the degree of fit with respect to the estimated linear fit.

Additionally, the dataset contains a default model fitted to each time-series, it consists of linear, quadratic, and seasonal components. The model is represented by the following function:

$$y(t) = a + b(t - t_o) + c(t - t_o)^2 + d\sin((t - t_o) \cdot 2\pi) + e\cos((t - t_o) \cdot 2\pi) \tag{3-1}$$

where a, b, c, d, e are the parameters to be estimated and t_o is the offset value and equals $t_o = 4.5181$ for all of the time-series in this dataset.

[1] For the purpose of this work, the datasets come from the resources of the company SkyGeo.

(a) **(b)**

Figure 3-1: (a) The area covered by the considered dataset - Amsterdam, (b) a scheme for conversion between the line-of-sight and vertical projection.

3-2 Exploratory analysis

The deformation data provided in the dataset is a vertical projection of the line-of-sight (LOS) displacement computed in the InSAR processing steps. In the LOS a change of phase of 2π corresponds to the deformation of $\frac{\lambda}{2}$ (as the wave travels to the target and back). The TerraSAR-X satellite operates in the X-band and the wavelength λ is 3.1 cm. The vertical projection for a deformation corresponding to a 2π shift is

$$L_{\text{vert}(2\pi)} = \frac{\lambda}{2\cos\theta} = \frac{3.1}{2\cos(31.1°)} = 1.81 \text{ cm} \tag{3-2}$$

where θ is the incidence angle. The data in the dataset is given with a discretization of 0.1 mm. Later on in this study the deformation values are given in *mm* and the velocity values in *mm/year*.

Visual inspection of the deformation time-series from the dataset reveals that numerous examples contain some kind of errors or misfits. Below, different aspects of the data are presented and discussed.

3-2-1 Unwrapping

The unwrapping errors are caused by an incorrect resolution of the phase ambiguities in the unwrapping process. There are two main types of inaccuracies introduced by unwrapping. The first one are only single, individual points which are shifted by $k \cdot 2\pi$, where $k \in \mathbb{Z}$, from their actual position. The second type are the shifts of the whole subseries. Both types are shown in Figure 1-1. Determining whether a point is indeed an unwrapping error or just an outlier is not trivial. An assumption is made, that a point is an unwrapping error if it fits better to a series shifted by $k \cdot 2\pi$ than to an original time-series.

In Figure 3-2, there are three unwrapping errors present. They are all shifted down by the vertical equivalent of the 2π shift. It is not always straightforward to determine whether

an outlier is of the unwrapping nature or comes from an unusual state of the scatterer or some external disturbance. Time-series visualizations with the shifted copies of the series help a user/expert to recognize the unwrapping errors, however, not all the cases are easily interpretable.

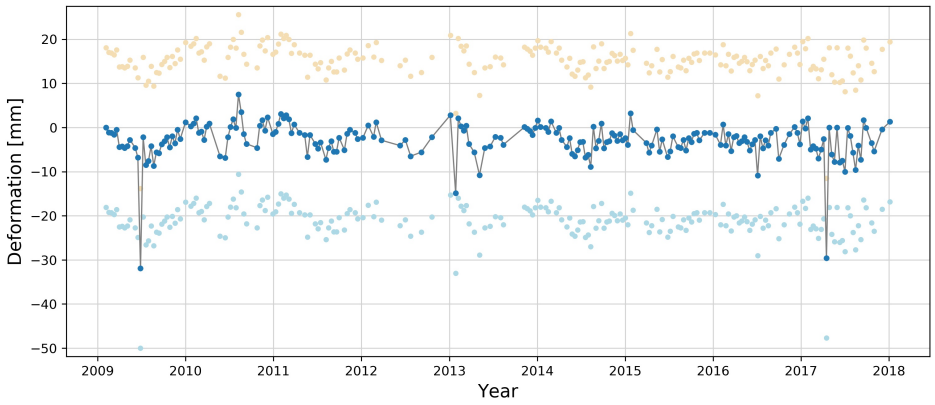


Figure 3-2: An example of a time-series with three single unwrapping errors, (AL1306).

Another example is shown in Figure 3-3. The time-series labelled in the plot as 1 (AL324) exhibits very similar behaviour to the nearby time-series 2 (AL325), with around 3m distance. The exception is that after January 2013 there is suddenly an offset between them. Figure 3-4 shows screenshots from the Time-series assessment GUI (see Section 4-2-1) with both time-series and with relative position of both scatterers. The value of the offset corresponds to a 2π shift. This allows to reason that both time-series should have the same behavior for the whole timespan but in one of them there was an unwrapping error which propagated till the end of the time-series, creating an unwrapping shift. Without any external information it is not possible to determine which of the two time-series is the correct one. One of the ways is to take into account the behavior of the neighboring scatterers, which could give an indication whether in this case the point went up or down in early 2013.

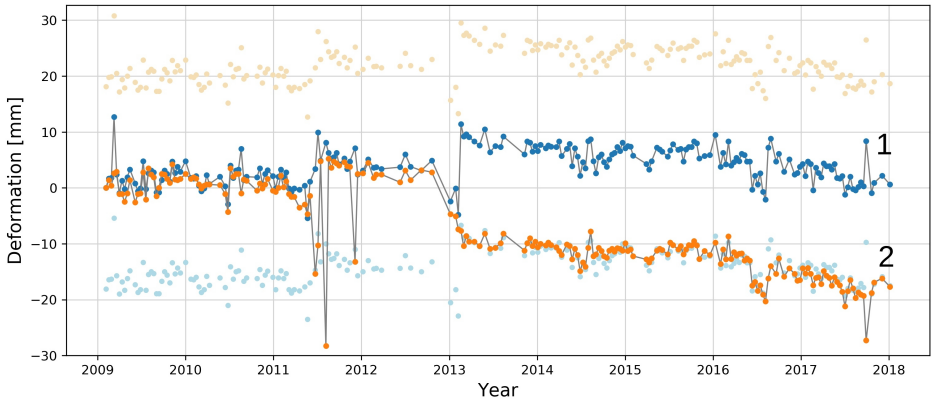


Figure 3-3: An example of two neighboring time-series where subseries are shifted by the 2π shift with respect to each other. The first series (blue, 1, AL324) is plotted together with its $\pm 2\pi$ copies. The second time-series (orange, 2, AL325) is very similar but the subseries after early 2013 overlaps with the -2π copy of the first time-series.

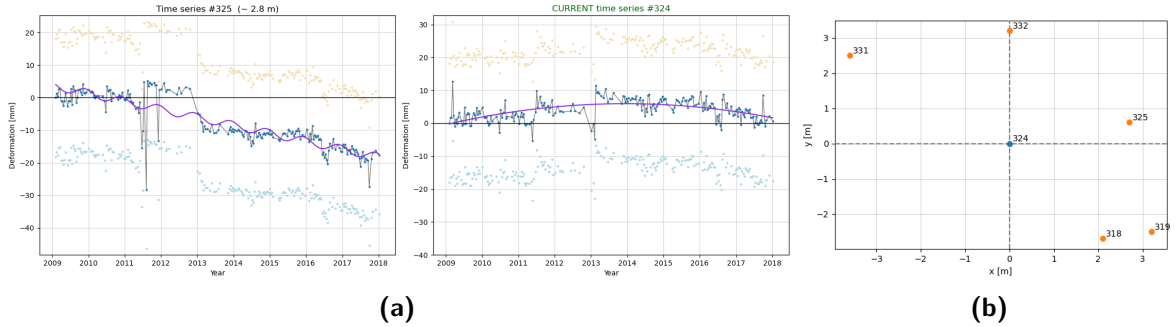


Figure 3-4: Screenshots from Time-series assessment GUI (see Section 4-2-1) view of AL324. (a) A part of the "Show neighbors" view with time-series AL324 and AL325. (b) Mini-map view showing the relative position of the neighboring time-series for AL324.

3-2-2 Missing measurements

The revisit time of the TerraSAR-X satellite is 11 days. For the timespan covered in the considered dataset that would correspond to 297 measurements collected with the 11-days intervals. However, there are only 219 measurements available, which is ∼73% of the whole timespan. The most of the missing measurements are between early 2012 and late 2013 (see Figure 3-5). One of the reasons for that could be that there were other tasks performed by the satellite at that time and it could not provide the measurements for this particular area.

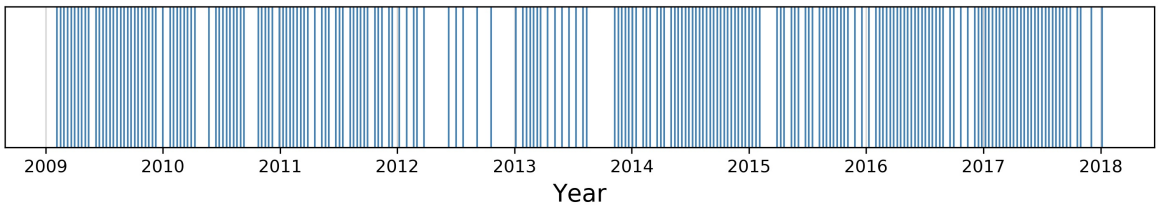


Figure 3-5: Timestamps for the available acquisitions

The gaps could be filled by interpolation, however, it is not straightforward. There are three options to be considered, namely with interpolation, without interpolation, or considering only available measurements.

Interpolation. The time-series can be interpolated in order to approximate the missing samples. There are multiple techniques available, such as linear, polynomial, or nearest-neighbour interpolation. However, in many cases the available neighbouring samples are not sufficient for a reliable prediction of the missing samples as the signal is noisy. Moreover, the behavior in the interval with gaps sometimes differs from the other parts of the signal, making it difficult to reliably interpolate the data.

No interpolation. There are 219 out of 297 samples available. The missing samples could be explicitly treated as ones without a numerical value and get a `NaN` assigned. This approach allows to keep the original structure of the time-series, however, this may also cause implementation issues while processing.

Ignore the gaps. The last alternative is to discard the empty samples and 'squeeze' the time-series, so that in only contains actual measurements. This approach simplifies processing of the data, nonetheless, it makes the time-series much less representative of the underlying process.

Further in this chapter only the original time-series are considered in order to avoid artificially distorting the real signal and making additional assumptions. However, in the following chapters the time-series are interpolated due to implementation constraints, see Section 5-2.

3-2-3 Non-homogeneity

Many among the time-series are not homogeneous. The behavior of the scattering point (or at least of the registered signal) differs in time. In some cases the transition between periods with a different behavior is smooth, in some it is abrupt. The differences that are easily discernible are, for instance, various trends within one signal, sections with nonlinear behavior, and varying dispersion. Below, a more detailed description and examples are given.

A time-series can be divided into trend, cyclic, seasonal, and a remainder component [13, 14]. Trend is a long-term, general tendency in the data to either increase, decrease, or stagnate. The cyclic component occurs when there are medium-term changes without fixed frequency which are modifying the trend, however, this pertains mainly to business or economic data. Sometimes trend and cyclic components are treated together as a trend-cycle component [13]. This convention is used also in this thesis and the trend-cycle is shortly called a trend for simplicity and as the cycles are not always observable in the data. The seasonal pattern occurs when the fluctuations in the time-series are caused by a seasonal factor and its frequency is known and fixed, such as yearly.

There are two models of the time-series composition. The first one, the additive model, assumes that the components are independent:

$$y(t) = T(t) + S(t) + R(t) \tag{3-3}$$

where $y(t)$ is a time-series data, $T(t)$ the trend, $S(t)$ the seasonal component, and $R(t)$ is the remainder (residual). On the other hand, the multiplicative model is expressed by

$$y(t) = T(t) \times S(t) \times R(t) \tag{3-4}$$

Here the components may affect each other and, for instance, the seasonal pattern appears to be proportional to the trend component. In the considered InSAR deformation time-series it is assumed that the components are independent of each other and that the model is additive.

Different trends

In Figure 3-6 a time series with two different trends is presented. A default model that was fitted to this signal is a quadratic function. This model is fitted to the whole time-series and it does not take into account that there is a step in early 2010. After dividing the time-series into two sections and fitting linear trends to each of them, one can see that they represent different behaviour. Additionally, the root-mean-square error (RMSE) between the

time-series and the default model (RMSE = 2.309) is higher than the model with two linear trends (RMSE = 1.883). Different trends can be associated with the changing behaviour of the scattering point or with the unwrapping shifts. Therefore, forcing a single continuous model such as in Eq. (3-1) to fit to such a signal depreciates the quality of the model and its ability to represent the real situation.

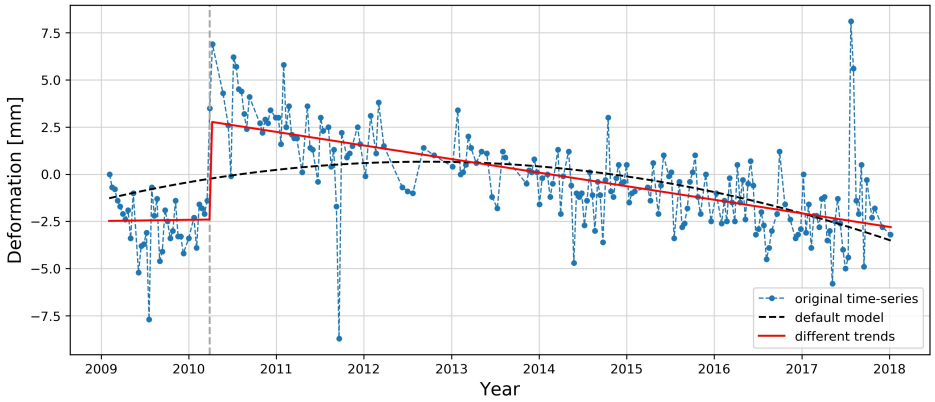


Figure 3-6: An example of a time-series with different trends, the original time-series with a default model (dashed) and two linear fits (solid) after manual segmentation (vertical dashed line) (AL109)

Nonlinearity

Not all of the distinctive segments of a time-series tend to have only linear trends. In some cases the whole time-series exhibits nonlinear behaviour, for instance, signals with strong seasonality, and in some, only a part of a time-series seems to be nonlinear. In Figure 3-7 one can distinguish three segments, first and last are mainly linear trends, the middle one, after accounting for an unwrapping error in early 2013, seem to have more nonlinear trajectory. However, it is important to take into account that in the presented case there are few measurements available and the signal is noisy.

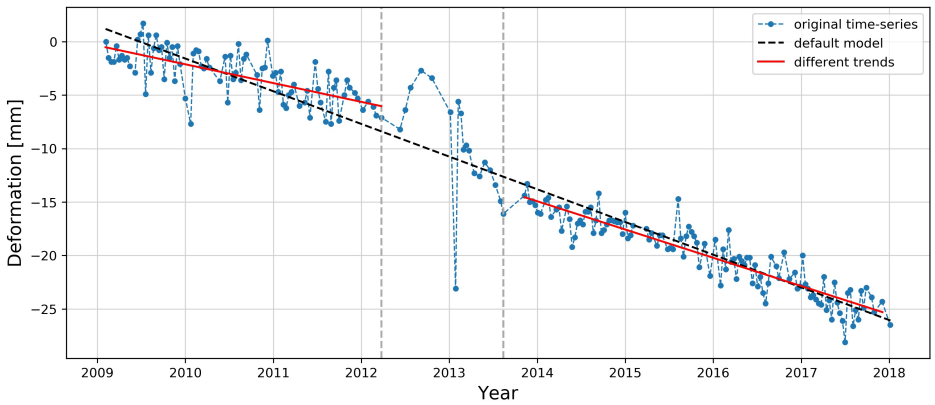


Figure 3-7: An example of a time-series with different trends and a nonlinear segment (with an unwrapping error in early 2013) (AL40)

Seasonality

From the visual inspection it can be observed that some of the deformation time-series show seasonal behavior. Usually, the period of changes is around one year. This suggests that the variation of the signal may be a result of yearly natural changes such as the weather. Other factors could be, for instance, agriculture and growth of the vegetation, however, in the considered case, the observed area is mainly an urban environment.

In the provided dataset, 43% of all the points have a default model fitted (see Eq. (3-1)) with non-zero coefficients for the periodic components, d and e in Eq. (3-1). An example of a time-series with a seasonal behaviour is given in Figure 3-8. The figure presents decomposition of the time-series into seasonal, trend, and residual components as in Eq. (3-3), using *Seasonal and Trend decomposition using Loess*, called STL decomposition [15]. The seasonal component with yearly fluctuations is clearly visible.

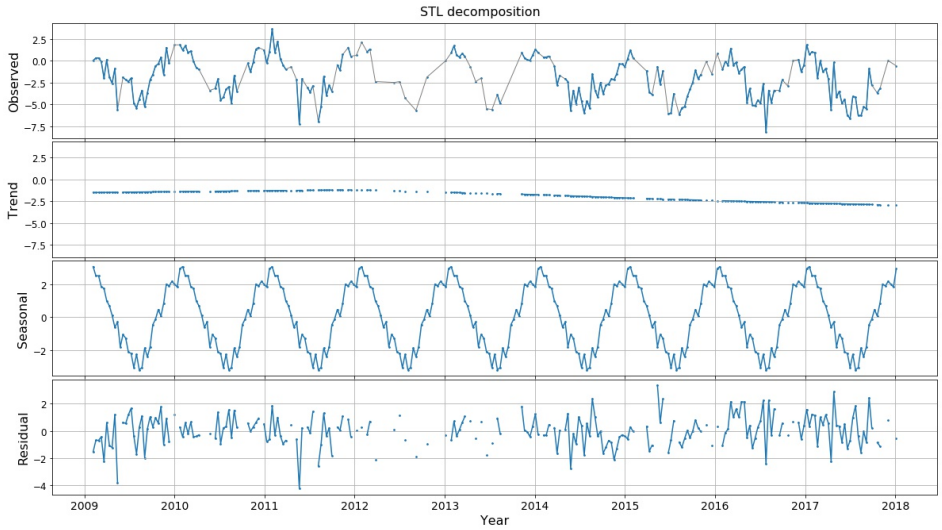


Figure 3-8: *Seasonal and Trend decomposition using Loess* (STL decomposition) of a time-series AL1223 with a yearly seasonal pattern.

Heteroscedasticity

In some of the time-series, one can observe heteroscedasticity, i.e. some subseries have different variability than the others. An example is given in Figure 3-9, where in the segment III the dispersion of the measurements is smaller than in the segment IV, and in Figure 3-10. This effect can be an indication of changes in the scattering point.

Partial decorrelation

If the measurements from a target are not coherent (see Section 2-2), the time-series from that point is usually rejected at the earlier step of the SAR processing. However, sometimes only a subseries is incoherent and the rest of the time-series is of sufficient quality to not get rejected at the PS processing step. In such case, the part that is incoherent, most probably

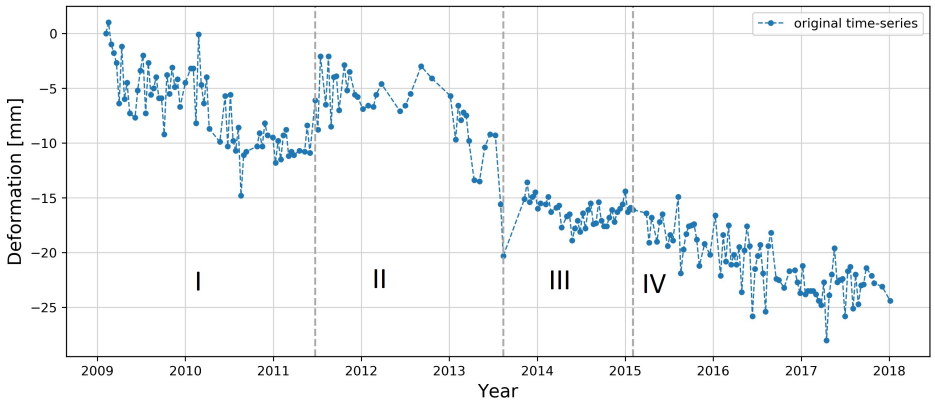


Figure 3-9: An example of a time-series with varying dispersion. The data in segment III has smaller variance than the data in segment IV. (AL50)

due to temporal decorrelation, does not yield any meaningful information for the rest of the time-series and often distorts or biases the default model. Only the information about the decorrelation itself is valuable as it suggests a certain behaviour of the target, i.e. the scatterer is changing between the satellite revisits. In [16] Rocca suggests two mechanisms. Either there is motion of the scatterers in the resolution cell, or they suddenly change their reflectivity. An example can be a construction site or water. As given in 2-2, in such case the phase measurement follows the uniform distribution, which can be seen in Figure 3-10.

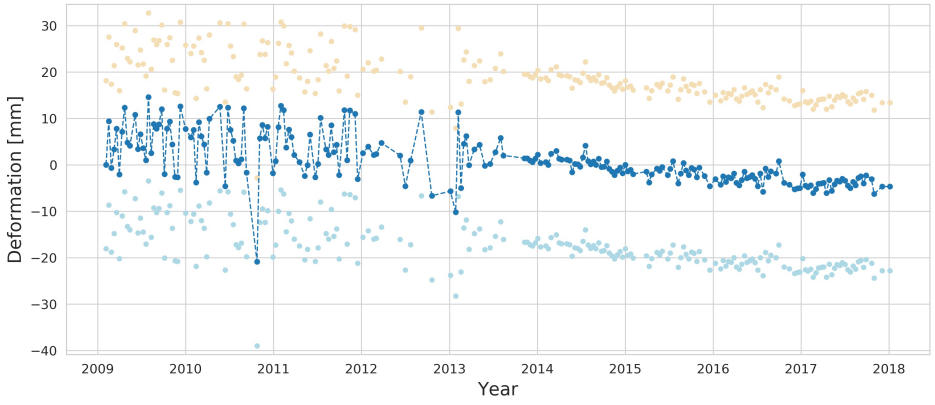


Figure 3-10: An example of partial decorrelation. In the first half of the time-series the phase measurements are incoherent and follow a uniform distribution (point AL7769).

Jumps

Another behaviour observed in the time-series for the scatterers are jumps, or steps. This means that there is a sudden change in the deformation time-series resulting in a subseries having an offset with respect to the earlier measurements. As examples can serve Figure 1-2 (bottom) and Figure 3-3. If there is no visible transition between the subseries but the change is abrupt, it is not possible to determine the true height of the jump as all the measurements

are initially wrapped and one of the unwrapping assumptions is that the difference between measurements is not bigger than π.

3-2-4 Stationarity

Many of time-series analysis techniques are based on an assumption that the time-series is stationary. Therefore a short analysis of the stationarity of the time-series in the considered dataset is presented in this section. A process is called strictly stationary if its properties, in particular the joint probability distribution associated with the observations, do not change in time [17]. However, for many practical applications, strict stationarity is too restrictive. Weak-sense (or wide-sense) stationarity is a weaker form of stationarity. The requirement is that the first moment, i.e. the mean, and the autocovariance do not depend on time, and the variance is finite.

For time-series with a trend, the trend can be either deterministic or stochastic. For the former, if the deviation from the mean, which can contain a trend, is stationary, the process is called trend-stationary. Additionally, such processes are mean-reverting, i.e. impulses have only transitory effects and the time-series converges back to the initial trend. For the latter, the processes with a unit root may exhibit a trend for which any random shocks have permanent effects. This is depicted in Figure 3-11 where the process signal is not able to return to its initial trend after a random shock. An example of a stochastic trend is a process with a linear trend and random walk. Such processes are called difference-stationary, as after first differencing they become stationary [13]. Therefore, simply detrending a time-series with a stochastic trend will not result in a stationary time-series and differencing should be applied. Unit root is an important concept related to stationarity. Its presence may cause the time-series to require additional transformations before applying various time-series analysis techniques. The unit root means that a root of a characteristic equation of the process, assuming it is a linear stochastic process, is equal to 1. This is related to a pole of a discrete system lying on a unit circle, which makes the system marginally stable.

There are statistical tests for stationarity, among the most popular ones are the Augmented Dickey-Fuller (ADF) test, where the null hypothesis is the presence of a unit root, and the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test for stationarity. More on hypothesis testing can be found in Section 3-5-1. They only test for first order stationarity, namely only the first moment (mean) has to be constant in time. They do not take the variability of the variance (second moment) into account.

The two tests are run on the considered dataset. In the python implementation it is possible to allow tests to take into account that the time-series have trends. Consequently, the ADF test suggests that in 6705 out of 8124 (83%) from low points and in 12995 out of 14999 (87%) of high points a null hypothesis of a unit root can be rejected, and hence, the time-series are stationary. In the KPSS test, for 3328 (41%) from low points and 5980 (40%) from high points the null hypothesis of stationarity is rejected, resulting in 4796 (59%) and 9019 (60%) stationary time-series, respectively. Clearly, the tests do not always give consistent results. In case when KPSS test suggests stationarity and ADF does not, the series is trend-stationary [18], there are 275 such series among low points and 342 among high points. Alternatively, if ADF indicates stationarity and KPSS non-stationarity, the series is difference-stationary. There are 2184 and 4318 such series for low and high points respectively. Another factor for

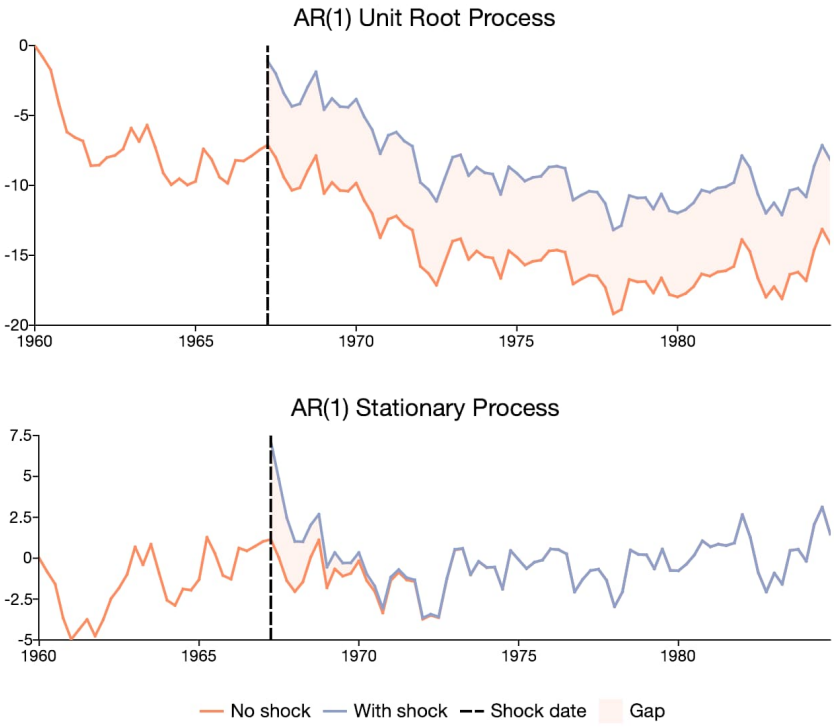


Figure 3-11: An example illustrating the concept of a unit root. A stationary process is able to return to its initial trend after a shock, i.e. an impulse caused by an exogenous factor. In case of a time-series with a unit root, the change does not have a temporary effect but rather causes a permanent shift.

non-consistency of the results can be limitations in the python implementation when it comes to the detrending and handling seasonal components.

The time-series representing four different results of the ADF and KPSS tests, namely, only ADF indicating stationarity, only KPSS indicating stationarity, both tests suggesting stationarity and both suggesting non-stationarity, are shown in Figure 3-12. It is not straightforward to assess the stationarity of a given series just by visual inspection, what justifies the use of statistical tests in order to determine the stationarity characteristics of a series.

There are different methods to impose stationarity onto a time-series, such as differencing or transformations such as logarithm or square root [13]. According to the ADF and KPSS tests results, most of the time-series are stationary or difference-stationary. Only a small number of points exhibits trend-stationary behavior and therefore differencing is chosen to be applied on all of the time-series to impose stationarity. The ADF and KPSS tests are rerun on the differenced time-series. In result, for ADF test, only two low points and eight high points are indicated as non-stationary and for KPSS test all low and high points are assessed as stationary.

However, the results of the stationarity tests should be handled with caution. Proper detrending and removing the seasonality component is difficult for all the different time-series at once. Additionally, the tests may have low power in some cases, e.g. small number of

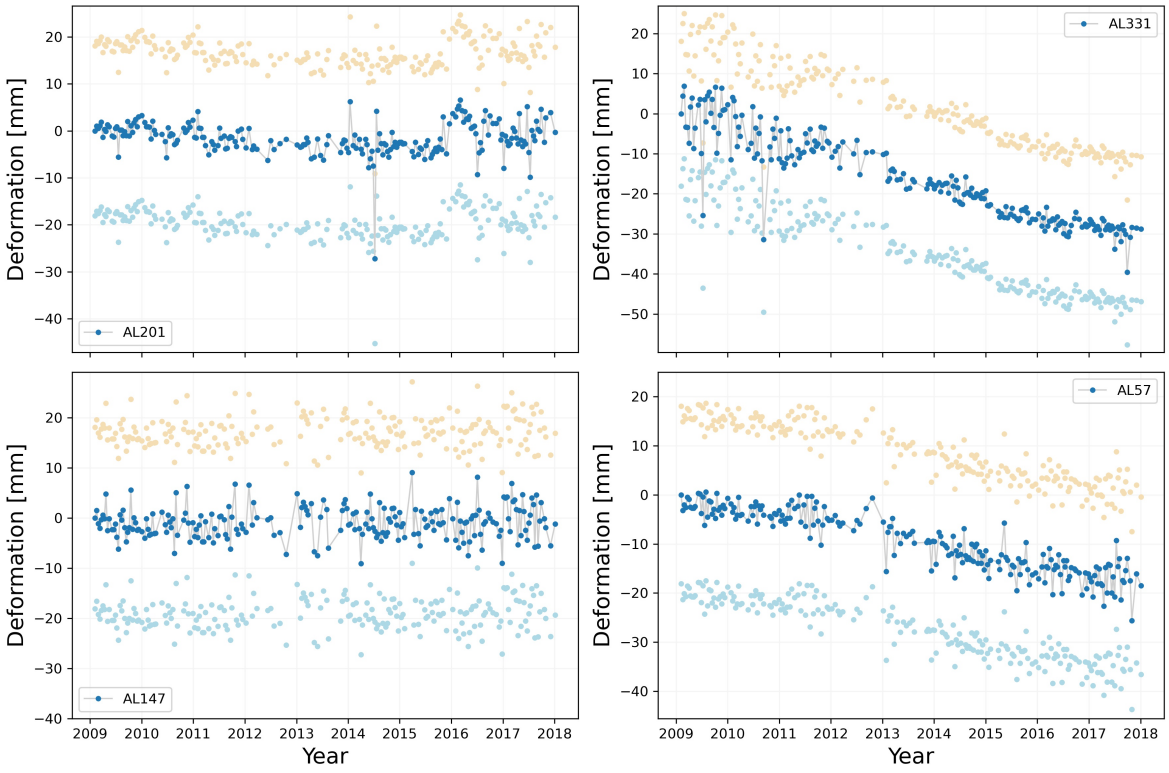


Figure 3-12: Example time-series for four different stationarity tests results. Top left: ADF - stationary, KPSS - non-stationary, top-right: KPSS - stationary, ADF - non-stationary, bottom-left: both ADF and KPSS - stationary, bottom-right: both ADF and KPSS - non-stationary.

samples, and their results may not be as trustworthy as expected. They also do not consider the changing variance of the time-series. However, the tests give some indication about the need for differencing for fitting the ARIMA models (see Section 5-2-4).

3-3 Contextual information

Analyzing an InSAR deformation time-series without any additional context may be difficult or can lead to incorrect conclusions, as due to phase ambiguities it is intrinsically impossible. Ideally, one would like to know as much as possible about the observed area in order to be able to understand the obtained InSAR measurements. E.g. , information about ongoing construction works, changes in a traffic situation, or any temporary events, may provide explanation for unusual and unexpected measurements. However, such information is rarely available and even if it is, it can only serve an operator manually working with the data.

As another type of contextual information can serve the spatial neighbourhood of a considered scattering point. Points close to each other can provide information about the behaviour of a bigger area or structure, e.g. when they all belong to one building.

Moreover, environmental factors can have an influence on the scatterer's behaviour. Such aspects as temperature or precipitation may change the physical features of the target. This

is discussed below in more detail. Additionally, some factors may influence the scattering characteristics of the target, for instance, snow cover or vegetation (represented by the normalized difference vegetation index, NDVI). Other contextual information could be soil type, land use, or the groundwater table.

Additionally, the signal itself can be a source of supplementary information that could be used in the assessment of the measurements and their interpretation. This includes, for instance, the amplitude and the coherence of the signal.

In the following, an exploration of the dataset is performed by taking into account such information as spatial neighborhood or temperature.

3-3-1 Spatial neighborhood

The time series for points which are geographically close to each other often exhibit similar behaviour, as they may, for instance, be located at the same building or street. Taking into account spatial neighbours of a point could provide more information about the observed behaviour and help with the detection of potential errors.

Two approaches for choosing the neighbors are used in this work. The dataset has information regarding the geographical location of each scatterer. In fact, the accurate position of the point is hard to determine, however, it is assumed to be sufficient for this purpose. The first approach is to choose five nearest neighbours of a point. The second one is to choose all the points that lie within a 10 m radius from the considered scatterer.

$$x^{i,k} \in N(x^i), \qquad k \in \{1, ..., \mathrm{NN}\} \tag{3-5}$$

where $N(x^i)$ is a set of points being neighbours for the considered point x^i and NN is the number of neighbours.

The nearest neighbours search is performed using the k-d Tree algorithm [19]. Based on the geographical location information, a tree of points is created. It allows to find the closest spatial neighbours, either by the number of requested neighbours or by the given radius. Other algorithms are also available, such as Ball Tree method [20]. Nonetheless, the k-d Tree algorithm is sufficient for this application as the task is only two-dimensional (latitude and longitude).

Correlation between neighboring points

It is possible that points located close to each other do not belong to the same object. Therefore, their behaviour can be substantially different and they should not be treated as 'neighbouring' points as it can lead to incorrect conclusions. Therefore, at first, the point's correlation with its the nearest neighbours is checked. For that purpose, the Pearson correlation coefficient is used, which is a measure of linear correlation between two variables, in this case two time-series:

$$r = \mathrm{corr}(x^i, x^j) = \frac{\sum_{t=0}^{T-1}(x_t^i - \overline{x}^i)(x_t^j - \overline{x}^j)}{\sqrt{\sum_{t=0}^{T-1}(x_t^i - \overline{x}^i)^2}\sqrt{\sum_{t=0}^{T-1}(x_t^j - \overline{x}^j)^2}} \tag{3-6}$$

The value of r is in the range $\langle -1, 1 \rangle$, where 1 and -1 mean perfect positive and negative correlation, respectively, while 0 means there is no correlation between the variables. The correlation of $|r| > 0.75$ is regarded as high, and the correlation of $|r| < 0.25$ is regarded as low.

For each point in the dataset, the correlation with all its neighbours is calculated. An interesting behaviour can be observed in case of comparing points which by visual inspection seem to be similar but one of them has a shift, see e.g. Figure 3-13. Then, the correlation calculated with Eq. (3-6) yields very low values of $|r|$, as the points do not exhibit linear correlation if the whole time series are considered. Instead, there are two separate parts, each showing linear correlation, but this cannot be detected using Eq. (3-6) for all samples. Figure 3-14 shows the correlation plot of an example (AL325 vs AL324). The time-series from this case are depicted in Figure 3-13.

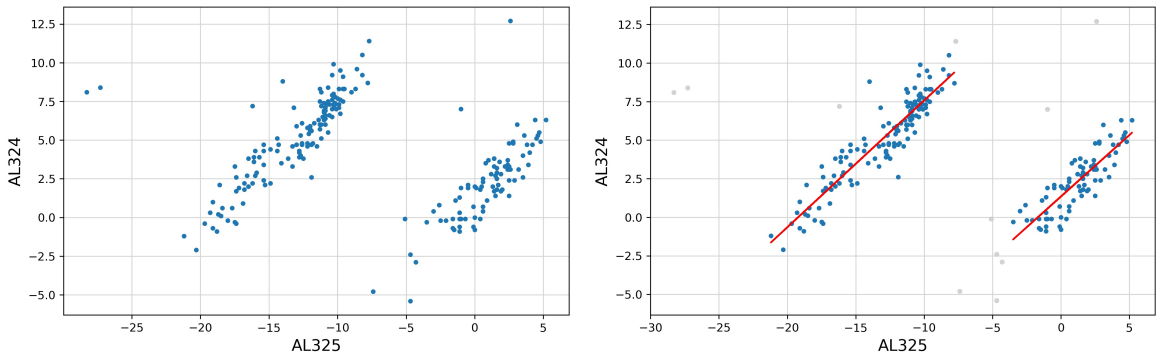
Figure 3-13: Two neighboring time-series AL324 and AL325 (top) and the correlation coefficient calculated with a sliding window of the size 30 (bottom). The window is shown in the top figure in red. The calculated coefficient within the window is centered, i.e. the result is shifted to the center of the window. The correlation coefficient median is 0.7797.

In order to find the two parts of both time-series which are correlated with each other, clustering is run in the space (x^i, x^j). The DBSCAN algorithm is used to extract the clusters [21]. The DBSCAN is chosen over, for instance, kNN algorithm because it is able to treat the points lying far from the constructed clusters as noise. The kNN algorithm takes all of the points into account, what makes it prone to distortions if any outlying points are present. Also, in contrast to DBSCAN, the kNN requires to set the number of clusters a priori. This is not desirable in this case, as one does not know beforehand how many shifts are in the

considered time-series.

Later on, the Pearson correlation coefficient is calculated separately for the created clusters. For the case of points AL325 and AL324, the correlation for whole series is -0.1598, whereas for the clustered case it is 0.8509 and 0.9248. This way of computing the correlation between the points allows to find the neighbours which, accounted for the shift, are highly correlated. A significant difference between the result of Eq. (3-6) and the one after clustering also yields an important information that may be helpful in detecting unwrapping shifts in time-series.

Another, alternative way of assessing the similarity between two time-series is to calculate the correlation coefficient using a sliding window. A sliding window of 30 samples is applied on both time series under analysis and the correlation coefficient is calculated using Eq. (3-6) for two subseries in a window. After performing the calculation for the whole time series, the median is taken from the scores obtained from the consecutive windows.



(a) **(b)**

Figure 3-14: (a) A plot of two neighbouring time-series against each other, AL324 and AL325. (b) There are two groups of highly correlated samples, lines represent linear fits to both groups and shows the linear correlation between the samples within each group.

A drawback of the DBSCAN correlation is the inability to control the clustering process for all possible cases. In some instances it may result in discarding too many samples as noise and returning a high score. However, a great advantage is that it can handle outliers and noisy subseries, as opposed to the sliding window correlation, which is sensitive to those. The sliding window correlation plot provides results that are easily interpretable and readable. Nonetheless, taking the median of the scores as the aggregated result leads to losing valuable information about the behaviour of the correlation score throughout the time-series.

3-3-2 Temperature

Variation of the temperature can influence a scatterer's behaviour and hence, a deformation time series. Due to the effect of thermal dilation, with the increase of temperature, buildings, infrastructure, and metal structures, such as lampposts or transmission towers, are subject to deformation. While most of the temperature-dependent anthropogenic targets expand in the summer, an opposite effect can be observed for some soil types. There, in the warm periods, the ground water table may get lower which results in the lowering of the ground while during the cold periods, the ground water table rises [10].

Therefore, one can observe a correlation between the temperature signal and the deformation time-series. As InSAR is a differencing technique and all measurements are taken with respect to the initial (master) one, the temperature time-series is adjusted accordingly, see Figure 3-15. The temperature data represents the average temperature over a day. The delay between the temperature change and the reaction of the target is assumed to be negligible. The source of the temperature records is the Royal Netherlands Meteorological Institute (KNMI) and the closest station to Amsterdam is at Schiphol, which is located $\sim$ 12 km from the area covered in the dataset. Hence, the temperature information is also charged with additional uncertainty.

The deformation due to the thermal dilation can be expressed by

$$\Delta L_T = \Delta T^k \cdot \eta \tag{3-7}$$

where ΔT^k is the temperature difference between the kth and the master acquisition (sample) and η is a thermal expansion coefficient, characteristic for the size and material of a specific object or target and can be estimated based on the observed deformations.

Figure 3-15: (top) Change of the temperature with respect to the first, reference measurement for the period of 02.2009 - 01.2018. The points mark the days corresponding to the acquisition days and the line shows the temperature record for the whole period with the 11-days interval. (bottom) A time-series with a very high positive correlation with the temperature correlation AH4304.

The temperature dependence is illustrated by a time-series AH4304, in Figure 3-15. The correlation coefficient between the differences in temperature and the registered deformation

Table 3-1: The percentage of points with temperature correlation coefficient ≥ 0.6 using three different methods

	full corr	DBSCAN corr	sliding window corr
low points	1.4%	4.0%	3.2%
high points	0.9%	3.2%	5.2%

is 0.8577, calculated as in Eq. (3-6). A strong seasonal behaviour is clearly visible, the target point is higher in the summer and lower in the winter.

The correlation with the temperature is calculated in three ways, similarly to the correlation with the spatial neighbours in Section 3-3-1. One is obtained using the whole time-series for both signals, temperature and deformation, the second one using clustering to find the correlation between subseries of the two, and the third one using a sliding window. The Table 3-1 shows the percentage of points showing the value of the correlation coefficient above 0.6, which is considered moderate correlation. The difference in the score using these methods suggests that there are many cases where the time-series is partly decorrelated but the coherent part exhibits temperature dependence. It is worth noting that none of the three correlation computing methods can handle the cases where a time-series shows temperature dependence but also has a significant long-term trend.

3-3-3 Precipitation

The influence of the precipitation can be observed especially on structures such as levees. There, extreme weather conditions, such as drought, heavy rainfall, storms, can affect the structure which can lead to failures and the risk of flooding. In [22] Özer *et al.* show that there is a correlation between the precipitation and temperature and the deformation of the levees. The meteorological conditions can result in swelling of the structure, due to the soil saturation and increased pore water pressure, or shrinkage, due to excessive drying. The reaction time, as well as the magnitude of the deformations depend on, among others, the soil type and the geometry of the levee.

3-4 Suggestions

Based on the above analysis, a few suggestions can be formulated. First, the potential unwrapping errors can be detected (to some extent) and corrected. Second, many of the time-seires could be divided into more homogeneous segments which are analyzed separately. Only then, they could be combined together again. Further in this thesis different approaches to the detection of the potential unwrapping errors and to the segmentation of the time-series are tested, see Section 5-2 and Section 5-3.

Regarding the contextual information, one would like to have as much additional data as possible, however, this is not achievable on a large scale and automating the incorporation of this extra context would at first require a laborious input from the human operators. On the other hand, incorporation of environmental factors can be used in an automated system to

assess the accuracy of the acquisitions and to interpret the behaviour of the point scatterers, e.g. seasonal, temperature-dependent signal.

3-5 Current approach

The time-series obtained from, e.g. Persistent Scatterer time-series technique, see Section 2-3, can be a subject to further postprocessing techniques, such as atmospheric effects or kinematic behavior (deformation modeling). Solving for the kinematic parameters is an ill posed problem because without any additional constraints it is not possible to determine whether the variability of each single point should be attributed to actual deformation or to noise or decorrelation [10, 23]. Among the assumptions that are made are smoothness of the signal in time and space.

The current approach is based on the multiple hypothesis testing to obtain an optimal kinematic model for the observed time-series. The null hypothesis is chosen as the steady-state behavior, the rest of the hypotheses is based on the *library of canonical functions*, which is presented below.

3-5-1 Hypothesis testing

A statistical hypothesis is a statement on a random variable or a process. After performing an experiment or observations, one can infer about the given statement based on the collected data. A primary hypothesis is called a null hypothesis and is denoted as H_0 [24].

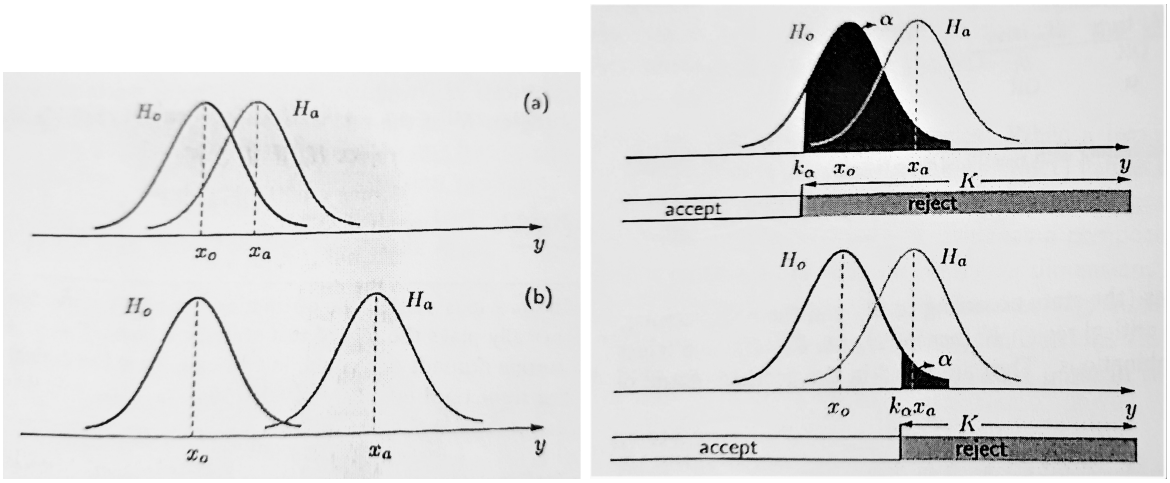
If the collected evidence is sufficient, the null hypothesis can be rejected. On the contrary, one can also fail to reject the hypothesis if the data does not support the rejection. Nonetheless, failing to reject the hypothesis is not equal to confirming it. A second, complementary hypothesis can be defined, called the alternative hypothesis H_a. The null hypothesis can be tested against the alternative hypothesis and it can be rejected in favor of the alternative hypothesis. This is called binary hypothesis testing.

The hypothesis can be formulated as follows [24]:

$$H : \underline{y} \sim f_{\underline{y}}(y|x), \tag{3-8}$$

where $f_{\underline{y}}(y|x)$ defines a probability density function (PDF) of an observable $\underline{y}$ given the unknown parameter x. x here can be a scalar, a vector, or even a matrix parameter, depending on the formulation of the problem.

An example can be a signal measurement and two hypotheses that differ in the mean values of the signal [24]. Then, the hypotheses are defined as the expectations of the observable $\underline{y}$ with two possible means, x_0 and x_a. Figure 3-16 shows two possible pairs of hypotheses. In order to decide whether the collected data speaks in favor of the null hypothesis a decision rule is required, called a test. A chosen test statistic, which is a function h of the observables, $T = h(\underline{y})$, can be used to reduce the dimensionality. The decision rule is defined in terms of the critical region K, i.e. a set of values of $\underline{y}$, or of a test statistic T, for which the null hypothesis H_0 gets rejected. This is depicted in Figure 3-16b.



(a) **(b)**

Figure 3-16: (a) Two pairs of hypotheses, H_0 and H_a, with two different pairs of means of observable y, x_0 and x_a. (b) The choice of critical region K determines for which values of y, or test statistic T, the null hypothesis H_0 gets rejected.

Therefore, if the $T \in K$ then H_0 is rejected and complementarily, if the $T \notin K$ then one fails to reject H_0. There are two decision errors to be considered, related to the chosen critical region K, namely, one can either reject a null hypothesis which is in fact true, or accept a false one. These are called type I and type II errors respectively. The whole decision matrix is given in Table 3-2 and a very similar concept is presented in Table 5-1 as well.

Table 3-2: Decision matrix of binary hypothesis testing.

		Reality	
		H_0 true	H_0 false
Decision	accept H_0	OK	β - type II error
	reject H_0	α - type I error	OK

The probability of committing a type I error is denoted by α, which is called a level of significance. This is defined as $\alpha = P(\underline{T} \in K|H_0)$, i.e. the probability of the test statistic lying within the critical region even though the null hypothesis is true.

On the contrary, the probability of accepting a false null hypothesis, i.e. committing a type II error, is denoted by β and defined as $\beta = P(\underline{T} \notin K|H_a)$. Moreover, the probability of choosing the alternative hypothesis when it is indeed true is called the power of the test and is denoted by γ, where $\gamma = 1 - \beta$.

In the considered case of InSAR observations, where the null hypothesis is a steady-state behavior, an alternative hypothesis may sound as follows: "The observed deformation time-series is caused by a temperature-dependent behavior of the persistent scatterer". Below, the approach of defining the alternative hypotheses is presented, as well as the decision flow for choosing the most optimal one.

3-5-2 Library of canonical deformation models

The canonical functions represent physically realistic models and are the building blocks for the temporal model for the time-series.

$$\begin{aligned} M_1(v(\boldsymbol{x})) &= t \cdot v(\boldsymbol{x}), \\ M_2(\eta(\boldsymbol{x})) &= \Delta T \cdot \eta(\boldsymbol{x}), \\ M_3(s(\boldsymbol{x}), c(\boldsymbol{x})) &= \sin(2\pi t)s(\boldsymbol{x}) + (\cos(2\pi t) - 1)c(\boldsymbol{x}), \\ M_4(\kappa(\boldsymbol{x}), \beta(\boldsymbol{x})) &= (1 - \exp(\frac{t}{\beta(\boldsymbol{x})}) \cdot \kappa(\boldsymbol{x}), \\ M_5(D_i(\boldsymbol{x})) &= D_i(\boldsymbol{x})\delta_{ij}, \qquad i,j \in [1,m], \\ M_6(\Delta_i(\boldsymbol{x})) &= \Delta_i(\boldsymbol{x})\mathscr{H}(t - \tau_i(\boldsymbol{x})), \qquad i \in [1, m-1], \end{aligned} \tag{3-9}$$

where t is the temporal baseline, $\eta(\boldsymbol{x})$ is the thermal expansion, $\mathscr{H}(t-\tau_i(\boldsymbol{x}))$ is the Heaviside step function, δ_{ij} is the Kronecker delta function, $\kappa(\boldsymbol{x})$ is the exponential magnitude, and $\beta(\boldsymbol{x})$ is the slope factor of the exponential magnitude.

The temperature component, as mentioned in Section 3-3-2, can be expressed using either M_2 or M_3, which are a directly temperature-related function and a seasonal periodic function, respectively [23].

This approach allows to arbitrarily combine the canonical models. Therefore a great number of deformation models can be created and used as alternative hypotheses.

3-5-3 Multiple hypothesis testing for InSAR

The idea behind the multiple hypothesis testing approach is to compare all the alternative hypotheses with the null hypothesis and assess which model among the alternative hypotheses is the optimal one. Each model, and hence each hypothesis, can be expressed in a form of a linear system of equations

$$\begin{aligned} H_0: \quad & \underset{m\times 1}{E\{\underline{y}\}} = \underset{m\times n}{A}\,\underset{n\times 1}{x}; \\ & \underset{m\times m}{D\{\underline{y}\}} = \underset{m\times m}{Q_{yy}} = \sigma^2 R_{yy} \\ H_j: \quad & \underset{m\times 1}{E\{\underline{y}\}} = \underset{m\times n}{A}\,\underset{n\times 1}{x} + \underset{m\times q}{C_j}\,\underset{q\times 1}{\nabla_j}, \quad \nabla_j \neq 0; \\ & \underset{m\times m}{D\{\underline{y}\}} = \underset{m\times m}{Q_{yy}} = \sigma^2 R_{yy}, \end{aligned} \tag{3-10}$$

where A is the design matrix, m is the number of observations $\underline{y}$, x is a vector of unknown parameters, C_j is a design matrix for the additional parameters ∇_j which are defined for each alternative model. Each hypothesis has a test statistic associated. In this case the test statistic $\underline{T}_q$ follows a noncentral Chi-squared distribution $\chi^2(q,\lambda)$, where q denotes the degrees of freedom and λ - the level of noncentrality.

For a given level of significance α and the related critical value $\chi^2_\alpha(q,0)$, if $\underline{T}^j_q > \chi^2_\alpha(q)$ then the null hypothesis H_0 is rejected.

The algorithm for the multiple hypothesis testing for InSAR is presented in Figure 3-17. At first, an Overall Model Test is performed in order to check whether any further model search is needed. If $\underline{T}_0$ is smaller than the critical value K, there is not enough evidence to reject the steady-state default model and the null hypothesis is sustained. Otherwise, each alternative hypothesis is tested and the test ratio is calculated as follows

$$\underline{\boldsymbol{T}}^j_{q_j} = \underline{T}^j_{q_j} / \chi^2_{\alpha_j}(q_j). \tag{3-11}$$

If the test ratio is smaller than 1, the hypothesis is not rejected and is considered further while searching for the most probable model, which is chosen as

$$\underline{\boldsymbol{T}}^B_{q_B} = \max_j \{\underline{T}^j_{q_j}\}. \tag{3-12}$$

Regarding the choice of the parameters, the level of significance is chosen using the rule of thumb often used for the InSAR time-series which is $\alpha_0 = 1/(2m)$, where m is the number of observations. It typically yields α_0 between 0.2% and 2%. For each alternative hypothesis and its dimensions the α has to be adjusted. The power of the test γ_0 is related to the size of an additional parameter in the alternative hypothesis that would cause the alternative hypothesis to be chosen. In this approach it is chosen as $\gamma_0 = 50\%$. More details can be found in [10].

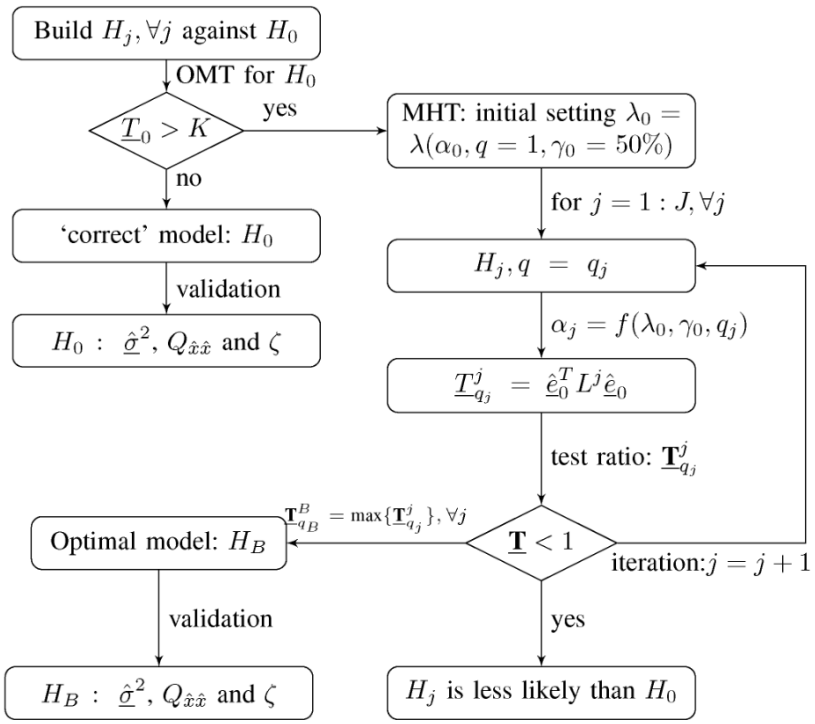


Figure 3-17: Flowchart for the multiple hypothesis testing [10, 23].

Conclusions

In this chapter an exploratory analysis of the InSAR time-series is performed. There are several aspects of the time-series data that are presented and discussed, such as unwrapping errors, both single points as well as shifts, missing measurements, and non-homogeneity of the time-series, e.g. different trends, heteroscedasticity, partial decorrelation. Also, a stationarity analysis is presented. Moreover, the relevance of the contextual information is investigated, mainly spatial neighborhood and temperature dependence.

Although the the time-series come from the same sensor and go through the same preprocessing procedure, there are many factors that influence the resulting time-series. Therefore, there is significant variability in the whole dataset which makes it difficult to find a solution that would handle all time-series using a single approach. Also the analysis is hindered by missing measurements and hence, need for interpolation of the data.

Chapter 4

Expert knowledge incorporation

In the era of rapidly growing fields of Artificial Intelligence and autonomous systems, there is a need to include knowledge of a human expert into the system design in order to benefit from the already available domain knowledge and experts' experience. Especially, when the systems are supposed to assist or even replace human operators in some tasks. Such a system can contain knowledge of procedures, typical as well as abnormal or dangerous behaviours, relations between objects, or general rules in a given domain [25].

4-1 Knowledge incorporation

A model that encodes expert and contextual knowledge can be explicit or implicit. In the first case, the information about a situation, an event, or some property of an object has to be fully described and hard coded by an expert or an operator. In the second case, the system can learn patterns or behaviours from the analysis of provided training data using different learning algorithms [26]. In this section, ways of encoding the expert knowledge as well as how can it be incorporated into a system are discussed.

4-1-1 Knowledge representation

There are numerous ways of representing knowledge. In this chapter three popular approaches are presented, namely mathematical models, rules, and first-order logic. A choice of a particular type of the knowledge representation is often determined by the form of available knowledge for a given application.

Mathematical models

Mathematical model is a very general term and different concepts can be described as such. However, a common feature of mathematical models is that the information is described using equations and formulas.

There are numerous examples in the literature where expert knowledge is included in the system. For instance, in [27, 28] additional information is incorporated directly into a kinematic or a dynamic model used for object tracking. Such extended models are used in a Kalman filter and result in a reduction of errors [28]. Another way to include auxiliary information into a system is by imposing constraints on a model [25, 27]. Moreover, a lot of information can be represented in a form of an artificial force field. Such a field does not refer to any actual force acting on the considered entity but it provides accessible models to simulate the behaviour of an object in a given environment [29].

Rules

Expressing information and knowledge in the form of rules is natural and easily understandable. Therefore, it is a popular way of incorporating the expert knowledge into the system.

Straightforward if-then-else rules are characterized by their simplicity and effectiveness. They can be employed, for instance, in finding a suspicious event or behaviour, or in system guidance for what measures should be taken.

Examples of a simple rule can be

$$\textbf{IF } rain > 2 \text{ mm/h } \textbf{THEN } qual = 0.5,$$
$$\textbf{IF } s1 \text{ and } s2 \text{ give contradictory results } \textbf{THEN } \text{use only } s1$$

where the former one encodes the following information: if the rainfall is bigger than 2mm/h, then a quality coefficient for the sensor's measurements receives value 0.5. Similar approach to incorporate quality information can be found in [30].

An alternative to this kind of rules are fuzzy rules [31, 32]. They also are an if-then-construct, however, the antecedent and consequent are fuzzy propositions. Considering a linguistic (Mamdani) fuzzy model of a following form

$$\textbf{IF } x \text{ is } A \textbf{ THEN } y \text{ is } B$$

x and y are base variables and A and B are linguistic terms, which need to be specified beforehand based on the chosen semantic rule and membership functions. An example of a fuzzy rule might be

IF the weather is bad **THEN** the reliability of the sensor is poor

In this case, the meanings of "bad" and "poor" have to be defined. This is done using membership functions which allow for a degree of truth for the statement that the weather is bad. Membership function is a generalization of a characteristic function of a conventional set, but instead of having only $\{0, 1\}$ values it can take values from the interval $[0, 1]$ [33]. "Badness" of the weather is not only 1 if it is bad or 0 if it is not bad, but values in between are possible. This kind of approach seems to be suitable for representing expert or operator knowledge as it deals with linguistic variables what is closer to the way humans express their knowledge.

First-order logic

Propositional logic is a branch of logic, it is based on propositions (facts) and is concerned with joining the propositions to form more complicated statements and with the logical relationships between the propositions [32, 34]. As example can serve a simple rule given above. "If rain bigger than 2 mm/h then quality parameter is equal to 0.5" is an implication and "rain bigger than 2 mm/h" and "quality parameter is equal to 0.5" are propositions.

An extension of propositional logic is first-order logic, which is another powerful method to represent knowledge. While in propositional logic only facts are encoded in the knowledge base and they can be either true, false, or unknown, in first-order logic, there are also the so-called objects and their relations that are taken into consideration [32]. Additionally, an important feature of first-order logic is its ability to express facts about some or all of the objects, thereby making it possible to represent general rules or laws. Consequently, this approach is better suited to represent complex environments.

The structure of first-order logic consist of the following elements: terms (which can be a constant, a variable, or a function of a term) that refer to objects, predicates that refer to relations, and atomic sentences, which are sentences that are indivisible, that state facts. In a popular example from [32], there are objects John and Richard (*Richard*, *John*), a relation of being a brother (*Brother*), and an atomic sentence stating that John and Richard are brothers (*Brother(Richard, John)*). Furthermore, there are also quantifiers which enable to represent general rules, functions, and complex sentences [31, 32]. While developing a knowledge base, it is necessary to carefully decide on a vocabulary of predicates, functions, and constants, and to thoroughly analyze the domain to encode general knowledge and more specific relations. An example related to InSAR processing could be the following: a scattering point can be an object, two points (objects) being neighbors within a certain range can be a relationship. If another object is added, e.g. a swamp, a relationship *On* can be defined and then, a statement (*On(Point, Swamp)*) would be an atomic sentence stating the relationship between the two objects. And lastly, a sentence “if *On*, then <some property>“ would be a proposition.

4-1-2 Incorporation into a system

As stated above, the implementation of the expert knowledge can be either explicit, as in all examples in Section 4-1-1, or implicit. The latter approach can also be called data-driven, as it is entirely based on the collected data. In this case, the expert knowledge can be encoded in the form of labels, or attributes, given to certain data (e.g. measurements or images). Next, a learning algorithm, such as a neural network, can use the labelled data to build a model.

Implementation of expert knowledge in the form of rules or mathematical models is often straightforward. The additional equations and constraints can be added to the model and/or the optimization cost function, depending on the problem to be solved. Rules are extensively used in, for instance, decision support systems [35]. Fuzzy logic can be incorporated into decision or control systems [36].

4-2 Data collection

For this thesis, a data-driven approach is adopted and not domain-driven approach, i.e. explicit rules, models, or logic. Therefore, there is a need to collect the data first. For that purpose, an application is built to record the decisions of a user. To provide expert knowledge multiple people were asked to contribute to this project. Among them are people who are experts in the field of InSAR and geodesy, as well as people who work with InSAR data for their applications, such as researchers and PhD students from the Geoscience and Remote Sensing (GRS) department of the TU Delft Faculty of Civil Engineering and Geosciences. They were asked to give answers based on their experience and intuition.

4-2-1 Tool description

A graphical user interface (GUI), presented in Figure 4-1, is designed to serve as a platform for the data collection. In general, a user is shown a time-series and his role is to make a decision about the following aspects. The user is asked about unwrapping errors and modelling errors. If, according to the user, there is an unwrapping error, a button *Unwrapping* is giving this time-series a label that it contains unwrapping errors. Also, a model that is fitted in the preprocessing and provided together in the dataset may not be a suitable one (as for the reasons presented in Section 3-2-3). Then, the *Model* button denotes that there are problems with a default model fitted. If both problems occur in a time-series, one should choose the button *Both*. On the other hand, if no errors are to be spotted, a user should choose *It's fine*.

Additionally, the user can manually mark individual unwrapping errors by clicking on a respective point in a plot. Moreover, it is possible to draw segmentation lines between the parts of a time-series that the user would treat separately for reasons such as jumps or changes within a time-series, see Section 3-2-3.

Both forms of contribution, namely only voting or marking the errors and drawing lines, are valuable input data and can be provided irrespective of each other.

In order to get more insight about the time-series under assessment, the user can see the plots for the five closest spatial neighbours of the considered point (see Section 3-3-1). This window is shown in Figure 4-2. Additionally, a map with the relative location of the point and its five neighbours can be displayed to get more information about the neighbourhood of the point.

The GUI can also serve as a simple viewer for the deformation time-series as it is easy to navigate between different points. Furthermore, it is possible to view the chosen time-series based on a provided subset of the dataset, and to view which points were marked as unwrapping errors and where the segmentation lines were drawn in the answers provided before or by another user.

4-2-2 Results

As mentioned above, a group of researchers and PhD candidates from GRS were asked to use the GUI and provide answers. In total seven people took part in the experiment. The answers were provided for 122 of the *low points* and 275 of the *high points*. Sometimes the

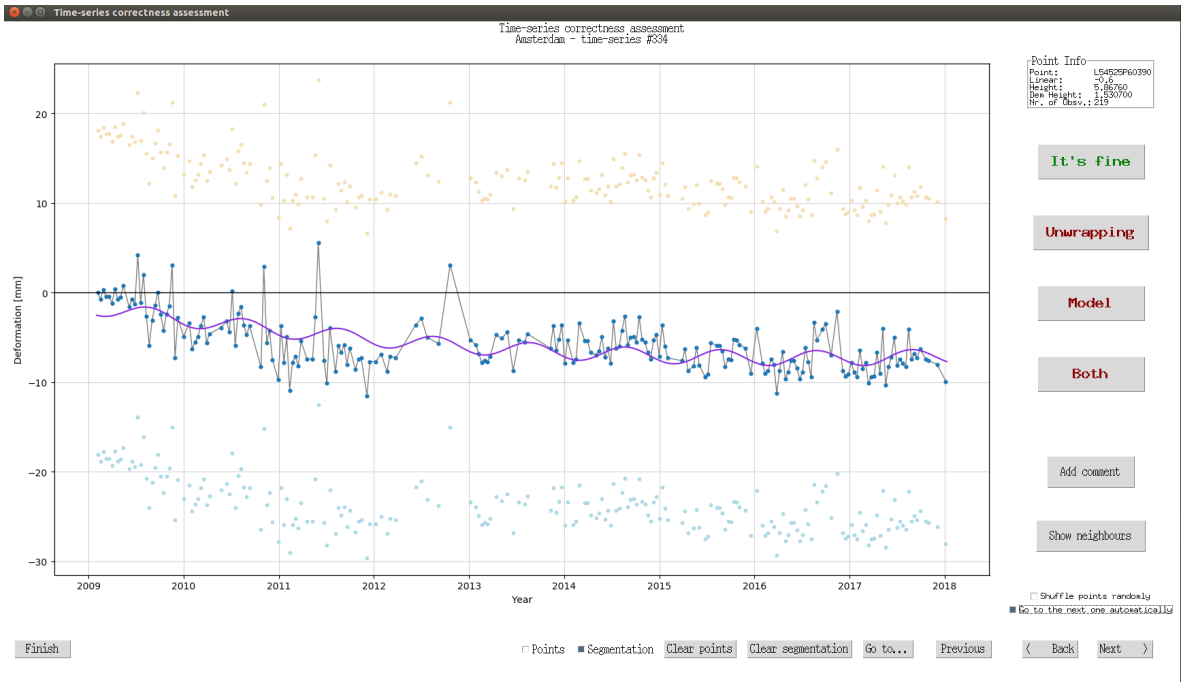


Figure 4-1: Graphical user interface for the data collection

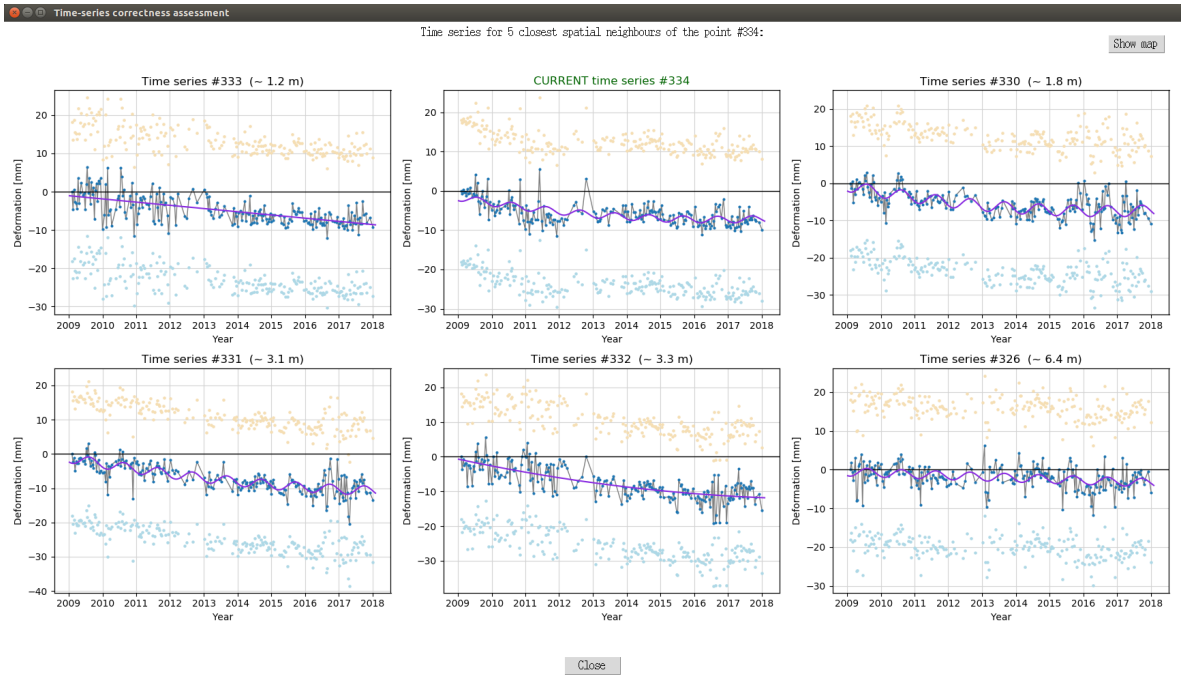


Figure 4-2: A window with the deformation time-series plots for the five nearest spatial neighbours of the considered point

time-series, for which the answers were given, were overlapping between the people. An interesting observation is that they not always agree in their judgments regarding the same time-series. In such case, a more restrictive answer was taken into account. An additional set

of more than 100 answers was provided by the author of this thesis.

4-3 Simulation

The data-driven approaches require lots of training data in order to prove their usefulness and potential, the amount differs depending on the complexity of the problem. In case when it is not easy to acquire sufficiently many expert answers, a possible solution is to use a simulation to create the training data artificially. It is a quick way to generate a lot of data for which the ground truth is known. Of course, a simulation will never be able to ideally recreate the real measurement data, as the observations are subject to random noise and unexpected and hence, unmodelled events. Additionally, multiple assumptions are made in order to generate the data and the underlying model may be incorrect or too simple.

In this thesis, two versions of the simulation are designed. The first, called *basic simulation*, generates homogeneous signals. The second one, the *extended simulation*, creates signals with changing properties. Here, the extended simulation is described in full details as the basic simulation is a simplification of this version. The differences between the versions are highlighted in the description. Examples of both simulations are shown in Figure 4-3.

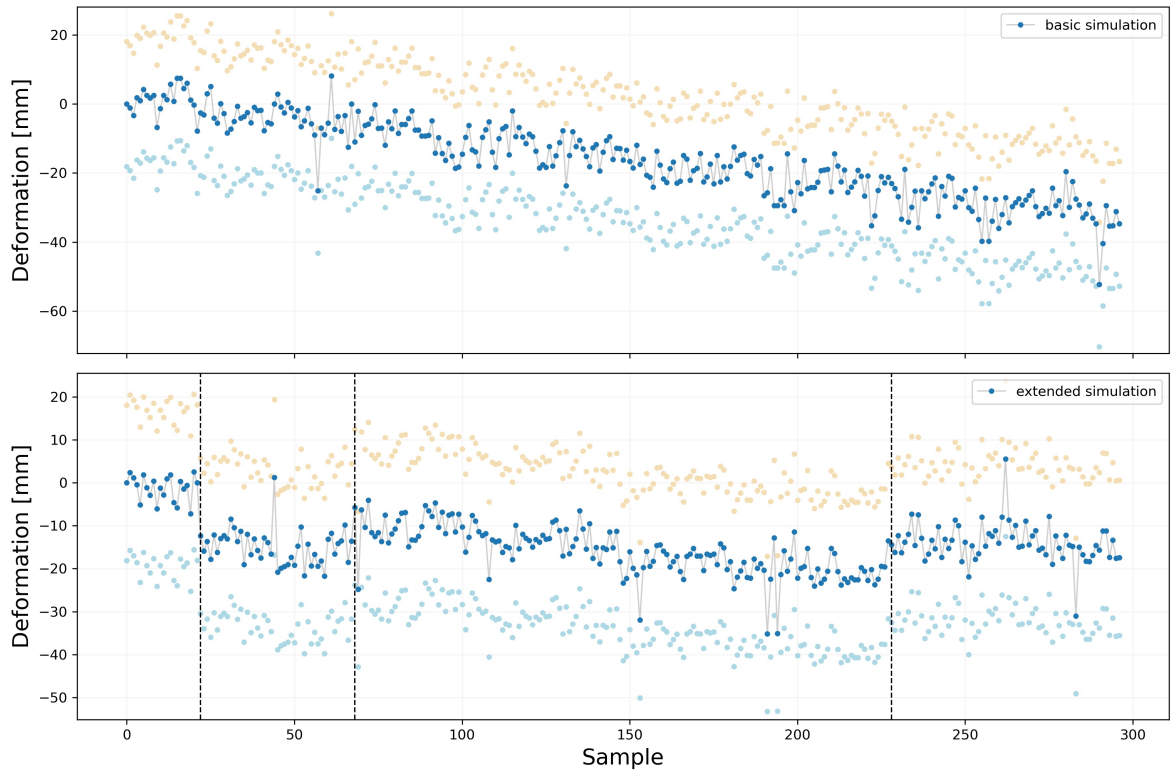


Figure 4-3: Examples of time-series generated with a basic simulation (top) and extended simulation (bottom). Black horizontal lines mark the division between the segments.

The general algorithm is presented in Appendix A. The extended simulation provides time-series with several segments with different characteristics. Therefore, the first step is to divide

the time index into segments. A number of segmentation lines to be generated is chosen randomly between 0 and 3, giving a maximum of 4 segments in a time-series. Additional constraint is that the segment cannot be shorter than 15 samples, in order to be visible at all. For the basic simulation, the number of segmentation lines is fixed to zero.

The next step is to choose a model function for the subseries to be generated in the first segment. The simulation is based on the canonical functions presented in Section 3-5-2. The available models are linear, temperature-dependent, exponential, and sinusoidal functions. The final model can be a superposition of the available functions. The number of components is chosen randomly with a 40% chance for a single component model, and 20% each for two, three, or four individual functions combined. Subsequently, for the first segment, a set of initial parameters is chosen at random, depending on the type of the model.

Later on, for each consecutive segment, a type of change is randomly drawn. There are three changes available with equal chances of being chosen, namely a change in the standard deviation of the samples, a step can be introduced, or a change in the model can be made. There is 60% chance that only one of the three will happen, 30% that two, and 10% that all three changes will be applied. If the model change is chosen, a new model is generated as described above. The parameters for the new model are chosen in such a way that the difference between the previous and the new model is significant in order to ensure that it is visible. Consequently, a subseries for the segment is generated, including possible changes in the standard deviation and level (step). This step is repeated for the rest of created segments.

The possible values for standard deviation and linear trend are taken from the analysis of the Amsterdam dataset, which is presented in Table 4-1. For the calculation of the standard deviation, the time-series are detrended using the value of the *linear* parameter, given in the dataset and estimated in the preprocessing (see Section 3-1).

Table 4-1: Statistics of the standard deviation and the linear trends for the whole Amsterdam dataset.

	min	mean	median	max	1^{st}	99^{th}
STD *low points*	1.02	2.75	2.80	8.59	1.33	4.53
STD *high points*	0.98	2.65	2.68	7.01	1.27	4.35
linear trend *low points*	-20.5	-1.5	-1.1	3.1	-8.3	0.7
linear trend *high points*	-17.4	-1.0	-0.7	3.1	-4.6	0.7

Values for the other parameters are chosen experimentally, so that the changes are visible and the time-series resemble the deformation time-series provided in the dataset. The complete list of all of the parameters and their possible values is given in Appendix A.

Conclusions

Out of several ways of incorporating expert knowledge into an automated system, a data-driven approach is selected in this study. In order to collect data required for a data-driven approach, a Python application is created. Additionally, a simulation is developed to address a problem of not enough data collected from the experts.

Chapter 5

Time-series processing

The two goals of this thesis are (i) to find potential unwrapping errors in the deformation time-series and (ii) to divide the non-homogeneous time-series into separate segments. For that purpose, several algorithms that can be found in the literature are implemented and tested.

At first, validation and performance metrics used throughout this thesis are introduced. Subsequently, different approaches to address the problem of unwrapping error detection are described in detail. Lastly, several methods for time-series segmentation are presented and applied.

5-1 Validation and performance metrics

In the tasks of unwrapping error detection and segmentation, the performance is determined based on the correctness of the obtained detections of unwrapping errors or segmentation lines. A sample is given a positive, *True*, label if it is predicted to be an unwrapping error or a segmentation line and a negative, *False*, label otherwise. In this context, a prediction is a state/label/value which is assigned as an output of a model. The

In order to assess the performance, the predicted labels are compared with the true labels. The ground truth in this study, or true labels, is collected as described in Section 4-2. There are four possible outcomes of such a comparison and they can be expressed in a form of a confusion matrix, see Table 5-1.

The first metric that can be defined is *accuracy*, ACC, which expresses the ratio between properly classified objects and all objects in the dataset:

$$\text{ACC} = \frac{TP + TN}{P + N} = \frac{TP + TN}{TP + TN + FP + FN}. \tag{5-1}$$

While accuracy is a widely used performance metric, it is not very well suited for the problems which are unbalanced, i.e. when there are much more occurrences of one label than the others.

Table 5-1: General formulation of a confusion matrix

		True condition	
		Positive (P)	Negative (N)
Predicted condition	Positive	True positive (TP)	False positive (FP)
	Negative	False negative (FN)	True negative (TN)

In fact, this is the case in the outlier detection. In a time-series, there are much more 'inliers', for which the label is negative, than actual outliers, labelled as positive. Therefore, the score for the accuracy metric will always be very high, as most of the points are indeed correctly classified as inliers. This introduces a bias which causes that even while not detecting any outliers in the series may lead to an accuracy value of 98-99%, which is not very informative. To address this other performance metrics can be introduced.

Recall, also called *sensitivity*, is a measure that focuses on how many *relevant* samples are *chosen*:

$$\text{recall} = \frac{TP}{P} = \frac{TP}{TP + FN}. \tag{5-2}$$

A complementary metric often used together with recall is *precision*, which expresses how many of the *chosen* samples are *relevant*:

$$\text{precision} = \frac{TP}{TP + FP}. \tag{5-3}$$

Furthermore, a metric that combines both recall and precision is the F_1-*score*, which is a harmonic mean of the two metrics:

$$F_1 = \left(\frac{\text{recall}^{-1} + \text{precision}^{-1}}{2} \right)^{-1} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}. \tag{5-4}$$

Recall and precision are common means of measuring performance for various machine learning applications. Other performance metrics used in outlier detection problems are *miss-rate*, MR, and *false discovery rate*, FDR. The former one expresses how many of the relevant samples are labelled incorrectly:

$$\text{MR} = \frac{FN}{P} = \frac{FN}{TP + FN}. \tag{5-5}$$

The false discovery rate shows how many of the predicted samples are false positives:

$$\text{FDR} = \frac{FP}{TP + FP}. \tag{5-6}$$

In this thesis, mainly recall and precision are used to assess the performance of the detection algorithms.

5-2 Unwrapping error detection

To detect potential unwrapping errors, these errors are treated as abnormal measurements (outliers). This approach allows the use of methods from the well established outlier (or

anomaly) detection field. There are multiple ways to perform the anomaly detection, however, for the case of time-series, a popular approach is to forecast a value and then compare it with the observed one [37, 38]. The difference between these two values is used to assess whether the considered observation is anomalous or normal:

$$d_t = |y_t - \hat{y}_t|, \qquad t \in \{0, 1, \ldots, T-1\}, \tag{5-7}$$

where y_t is an observed value and $\hat{y}_t$ is the predicted value for any time instance t. Other solutions are based on distance between samples in the data space or on density [37, 38].

The forecast-based anomaly detection has been applied, for instance, to the detection of abnormal measurements in the signals from a flight data recorder [39] or in sensor data, such as windspeed [40], hydrological data [41], or data from a chemical plant [42].

As only potential unwrapping errors, and not all outliers, are of main interest in this work, an additional constraint has been posed for the outlier detection problem:

> Constraint: The sample has to lie at least $0.75 \times (2\pi$ shift) from the forecasted value,

see Section 3-2-1 for the shift explanation and further in the current section for the rationale behind choosing this threshold value. Note that the developed algorithms are applied on 'unwrapped' data, i.e. the original phase ambiguities in the InSAR data have already been resolved.

A crucial part of the forecast-based outlier detection is the forecast step where the values are forecasted at each time instance t. Currently, there are numerous approaches to the task of forecasting time-series values [37]. The forecast can be based on the values of the neighboring samples in a window of a particular size [39, 43]. A different way is to apply moving average [41] or exponential smoothing [13] to obtain a smoothed time-series capturing the trend. Another widely used method for modeling the time-series are AR and ARIMA models [37]. A similar approach is a single-layer linear network predictor [40, 37]. Moreover, machine learning approaches such as a multilayer perceptron, support vector regression (SVR) [37, 40], k-nearest neighbours [44, 43, 45], can be used.

In this work, several of these prediction methods are described, tested, and their results are compared and assessed.

Experiment design

The experiments are run on two types of simulations, basic and extended (see Section 4-3), and on the real data with the ground truth provided from the GUI (see Section 4-2-1). As most of the methods presented in this section cannot handle missing samples, the time-series in the real data set are interpolated. The tested interpolation techniques are the *previous neighbor interpolation*, *linear interpolation*, and *quadratic interpolation*. Although the quadratic or cubic interpolation sometimes reflect the time-series behavior better, in general, the previous neighbor and the linear interpolation do not overfit as much as the former two and therefore they are chosen as more suitable for handling the whole dataset together.

Each of the two simulated datasets has 10.000 time-series. The size of this batch is chosen such that the total performance scores for the outlier detection experiments do not show significant change with the number of time-series anymore (see Appendix A).

To assess the performance of the presented methods, the recall and precision metrics are applied. There are two ways to compute the scores for both metrics in the considered case. One way is to calculate recall and precision for the outlier detection for each consecutive time-series and then take the average of both. The other way is to take into account all the samples of all time-series in a dataset collectively and then compute the recall and precision based on how many samples are labelled correctly in total. An example: if there are two unwrapping errors in each time-series in a dataset of 100 time-series, then each time-series is not considered separately but a total of 200 labelled samples is used to calculate recall and precision scores. The latter approach is slightly more rigorous. For instance, a time-series for which a forecast is very incorrect in the first case would add a score of 0.0 to the later calculated average. In the second case, it could add more than 200 false positives to the total count, leading to much lower total recall and precision values. For that reason, the second way of assessing the performance is used in the following experiments.

The choice of the threshold when a sample is classified as an outlier is not trivial and is obtained experimentally, which is depicted in Figure 5-1. The threshold cannot be equal to the respective 2π shift, as it is not possible to forecast a value with such an accuracy that the real unwrapping error would be exactly a 2π shift away from the predicted sample. Hence, a margin is needed to be able to identify outliers. Figure 5-1 presents the performance of the outlier detection (using two of the methods described in the following sections) depending on the threshold value.

For the simulated datasets the best F_1-score is obtained with the threshold of $0.8 \times 2\pi$ shift. For the real data on the other hand (100 labelled examples), the best result is for the $0.7 \times 2\pi$ shift. In conclusion, a design choice is to assume the threshold value as the value in between, namely $0.75 \times 2\pi$ shift (corresponding to 13.6 mm in this TerraSAR-X dataset), especially that in the real data it favors the precision over the recall.

Another approach is to additionally take the distribution of the data into account, so that the threshold varies based on the variance of the data. However, the local estimation of the standard deviation is not always reliable and results in incorrect threshold values, and further, incorrect detections.

In the following sections several outlier detection algorithms are tested on both simulated and real datasets. For brevity, only the scores for the extended simulation are presented, along with two interpolation methods for real data. The scores for the basic simulation can be found in the Appendix A.

5-2-1 Neighboring samples

A simple approach to predict the value of a sample y_t is to use its neighboring points (before and after) in the time-series, excluding the y_t itself [39, 43]. The forecasted value is based on the median (or mean) of the points within a chosen window, e.g. for $t-k$ and for $t+k$, where $k = 1, 2, 3$, and it's calculated by

$$\hat{y}_t = f(m_l, m_r), \tag{5-8}$$

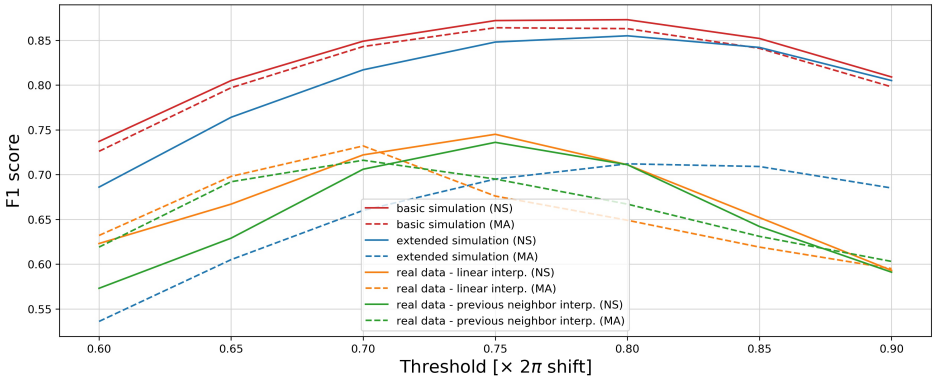


Figure 5-1: The performance of two outlier detection algorithms expressed with an F_1-score depending on the threshold value. The experiments are performed for simulated dataset as well as real data with two interpolation methods. NS refers to a neighboring samples algorithm and MA to a moving average smoothing approach, see Section 5-2-1 and Section 5-2-2 respectively.

where $m_l = f(y_{t-1}, y_{t-2}, y_{t-3})$ and $m_r = f(y_{t+1}, y_{t+2}, y_{t+3})$. The function for combining m_l and m_r and the function applied to the neighboring samples can be chosen as median or mean. The number of samples before and after x_t taken into consideration is set to three, but it is possible to introduce an offset between the sample to be predicted and the neighboring ones, i.e. use samples $t \pm k$ where, for example, $k = 2, 3, 4$. Omitting the closest samples can be helpful in a case with outliers in two consecutive samples.

Experiments

The algorithm is applied in four settings as given in Table 5-2. The windows with an offset of one sample, i.e. with $k = 2, 3, 4$, have higher recall than the ones without the offset. On the other hand, for $k = 1, 2, 3$ the precision is higher than for $k = 2, 3, 4$. This means that more outlying samples are missed but the ones that are chosen are more often correct. Overall, the method gives good results for both simulated datasets. For the interpolated data, the linear interpolation shows very little improvement over the previous neighbor interpolation. In general, for the real data, the method shows high precision while still finding two thirds of the outliers.

5-2-2 (Weighted) moving average smoothing

The moving average approach uses a sliding window of size n in which an average of samples within the window is computed, as in

$$\hat{y}_t = \frac{y_{t-1} + y_{t-2} + y_{t-3} + \cdots + y_{t-n}}{n} = \frac{1}{n}\sum_{i=1}^{n} y_{t-i}, \tag{5-9}$$

where $\hat{y}_t$ is a predicted value, y_t is a sample at the time instance i, and n is the number of the preceding samples taken into account, i.e. a window size.

Table 5-2: Results for the outlier detection with sample prediction using the neighbouring samples method. Scores for four different settings for the algorithm are presented, namely using mean or median of neighbouring samples and k samples symmetrically around the predicted sample.

	mean		median	
	$k = 2, 3, 4$	$k = 1, 2, 3$	$k = 2, 3, 4$	$k = 1, 2, 3$
Ext. simulation				
Recall	0.87	0.88	0.87	0.88
Precision	0.83	0.83	0.81	0.82
Real data - previous				
Recall	**0.67**	0.62	0.68	0.68
Precision	**0.82**	0.84	0.68	0.63
Real data - linear				
Recall	**0.68**	0.62	0.67	0.64
Precision	**0.83**	0.91	0.78	0.84

The prediction $\hat{y}_t$ is based on the past values, the number of which depends on the chosen window size. While this action introduces some lag, it smooths the signal making it possible to see more general trends in an often noisy signal.

If the whole signal is available (offline analysis), the window can be shifted so that the predicted value is located in the center of the window. This way it is possible to make use of the measurements on both sides (before and after) of the considered sample in the series. A drawback of such an approach is that one uses y_t to make a prediction $\hat{y}_t$ and if this sample is an outlier, the prediction will be biased. This can lead to difficulties with the detection of such an outlier.

In the above case, all of the samples within the window are of equal significance for the prediction of y_t. This can be changed by introducing the weights vector $\mathbf{w} = [w_i, \ldots, w_n]$, which allows to, for instance, give higher weight to more recent values, as in

$$\hat{y}_t = \frac{w_1 y_{t-1} + w_2 y_{t-2} + w_3 y_{t-3} + \cdots + w_n y_{t-n}}{n} = \frac{1}{n} \sum_{i=1}^{n} w_i y_{t-i} \ . \tag{5-10}$$

In practice, the weights should sum up to 1.

Experiments

For the moving average prediction, seven different, arbitrarily chosen window sizes are tested, namely 3, 5, 7, 10, 12, 15, and 20. Additionally, a weighted moving average for the window size of 5 is applied. The weights vector is set to $[0.1, 0.2, 0.3, 0.3, 0.1]$ in order to put more weight on the more recent samples, except for the directly preceding one, which also has lower significance. As presented in Table 5-3, the recall values increase with the size of the window until the window size of 7. For the bigger windows the values drop gradually. For the extended simulation, beyond the $n = 7$, both the precision and recall decrease with the

window size. For the real data, similarly to the neighboring samples method, more than half of the outliers are found. The precision is around 0.80 which can be considered a relatively good result. The linear interpolation gives better precision at the cost of a slight drop in the recall value, compared to the previous neighbor interpolation. Adding a weights vector did not yield better results, the recall is slightly higher than for non-weighted experiments but the precision is much lower.

Table 5-3: Results for the outlier detection with sample prediction using smoothing with the moving average (MA). Seven different window sizes n are tested, as well as one weighted moving average example (WMA).

	MA							WMA
	n = 3	n = 5	n = 7	n = 10	n = 12	n = 15	n = 20	n = 5
Ext. simulation								
Recall	0.87	0.87	0.87	0.85	0.84	0.83	0.81	0.78
Precision	0.69	0.77	0.78	0.76	0.74	0.71	0.66	0.66
Real data - previous								
Recall	0.69	0.67	0.67	0.62	**0.62**	0.53	0.46	0.68
Precision	0.57	0.70	0.74	0.78	**0.79**	0.75	0.69	0.61
Real data - linear								
Recall	0.56	0.62	0.62	0.59	**0.60**	0.54	0.44	0.65
Precision	0.61	0.76	0.73	0.79	**0.84**	0.80	0.79	0.67

5-2-3 Exponential smoothing

Exponential smoothing is yet another smoothing technique that can be used for forecasting. It is similar to the moving average but with exponentially decaying weights [13]. The exponential smoothing method takes all past samples into account and not only n past data points. There are three main approaches within this technique. In the simplest one, called the Simple Exponential Smoothing (SES), the forecasted value is computed as

$$\begin{aligned} \hat{y}_{t+1|t} &= s_t \\ s_t &= \alpha y_t + (1-\alpha) s_{t-1}, \end{aligned} \tag{5-11}$$

where s_t is the smoothed value for any time instance t and α is a smoothing factor, $0 < \alpha \leq 1$. In this case, the forecasted value $\hat{y}_t$ is equal to the last smoothed value. This is not suitable for the time-series that exhibit trends or periodic behaviour.

The second method, called Holt's method or the Double Exponential Smoothing (DES), is able to handle trends in time-series, what can be expressed by

$$\begin{aligned} \hat{y}_{t+h|t} &= s_t + hb_t \\ s_t &= \alpha y_t + (1-\alpha)(s_{t-1} + b_{t-1}) \\ b_t &= \beta(y_t - y_{t-1}) + (1-\beta) b_{t-1}, \end{aligned} \tag{5-12}$$

where h denotes the step in the $h-$step $-$ ahead forecast, b_t is an estimate of the trend, and β is a smoothing parameter for the trend ($0 \leq \beta \leq 1$).

The last approach is an extension of Holt's method, called the Holt-Winters' seasonal method or the Triple Exponential Smoothing (TES), and is able to deal with the seasonality in the signal. In the case of displacement time-series an additive nature of the seasonal component is assumed, as described in Section 3-3-2. The Holt-Winters' method is given by

$$\begin{aligned} \hat{y}_{t+h|t} &= s_t + hb_t + c_{t+h-m(k+1)} \\ s_t &= \alpha(y_t - c_{t-m}) + (1-\alpha)(s_{t-1} + b_{t-1}) \\ b_t &= \beta(y_t - y_{t-1}) + (1-\beta)b_{t-1} \\ c_t &= \gamma(y_t - s_{t-1} - b_{t-1}) + (1-\gamma)c_{t-m}, \end{aligned} \tag{5-13}$$

where c_t expresses the seasonal component of the signal and m is the frequency given in the number of periods in a year, γ is the smoothing parameter for the seasonal component and $0 \leq \gamma \leq 1$, and k is the integer part of $(h-1)/m$ [13].

Experiments

Experiments show that the smoothed time-series for both simple smoothing and Holt's method are similar and bring close results in the detection of potential unwrapping errors. The difference between them is more visible in the case of longer out-of-sample forecasting, but here only one-step-ahead prediction is performed. Holt's and Holt-Winter's methods require more computational time than the Simple Exponential Smoothing. The recall and precision scores for the Triple Exponential Smoothing (Holt-Winter's) are worse than Simple or Double Exponential Smoothing for all tested cases. Moreover, the results for tests with the Simple and Double Exponential Smoothing, SES and DES respectively, are presented in Table 5-4. Three smoothing levels α are tested, 0.5, 0.2, and 0.1. The value of $\alpha = 0.2$ gave the best results in SES so the DES method is tested keeping $\alpha = 0.2$ as the smoothing level. For the same parameters, the linear interpolation yields higher scores for precision and lower for recall compared to the previous neighbor interpolation. This means it produces more false negatives and less false positives. It can be observed that in TES the higher the value of γ, the more false positives are generated. Moreover, long gaps without the actual data which are interpolated linearly may hamper the seasonal behaviour of the time-series.

5-2-4 ARIMA

One of the most popular techniques for time-series modelling and forecasting are ARMA-type models [13, 17]. They are composed of autoregressive (AR) and moving average (MA) terms. In the autoregressive model of order p, denoted as AR(p), a variable is forecasted based on a linear combination of p its previous values

$$y_t = c + \sum_{i=1}^{p} \phi_i y_{t-i} + \epsilon_t, \tag{5-14}$$

Table 5-4: Results for the outlier detection with the prediction based on the Exponential Smoothing. Total recall and precision are obtained for Simple Exponential Smoothing (SES) and Double Exponential Smoothing (DES), for different smoothing levels α and smoothing slopes β.

	SES			DES		
	$\alpha = 0.5$	$\alpha = 0.2$	$\alpha = 0.1$	$\alpha = 0.2$ $\beta = 0.2$	$\alpha = 0.2$ $\beta = 0.5$	$\alpha = 0.3$ $\beta = 0.2$
Ext. simulation						
Recall	0.88	0.89	0.88	0.89	0.89	0.88
Precision	0.69	0.80	0.75	0.80	0.79	0.79
Real data - previous						
Recall	0.70	0.68	0.61	**0.67**	0.67	0.68
Precision	0.60	0.66	0.72	**0.79**	0.77	0.73
Real data - linear						
Recall	0.52	0.61	0.56	**0.61**	0.61	0.56
Precision	0.62	0.82	0.83	**0.83**	0.81	0.76

where c is a constant, ϕ_i are parameters of the model, and ϵ_t is white noise. The moving average model of order q, MA(q), is a regression-like model that uses current and past values of error terms or random shocks

$$y_t = \mu + \epsilon_t + \sum_{i=1}^{q} \theta_i \epsilon_{t-i}, \tag{5-15}$$

where μ is a constant, θ_i are parameters of the model, and ϵ is white noise.

One of the assumptions for the ARMA models is the stationarity of the time-series. In case of a non-stationary time-series, differencing can be applied to eliminate the non-stationarity. A combined model is an autoregressive integrated moving average (ARIMA) model of the order ARIMA(p, d, q), where d denotes the order of differencing.

The ARIMA models provide a parsimonious representation of linear processess. In order to select the best model for the considered time-series, two information criteria are popularly used, namely Akaike information criterion (AIC) and the Bayesian information criterion (BIC) [13, 17].

Regarding the considered dataset, the individual analysis of each time-series in a large dataset of time-series with varying behaviour is difficult. Usually, one model type is not able to fit all of the different occurring patterns. The majority of time series is non-stationary due to an existing trend and the differencing is used to transform the time series to a stationary signal. However, some of the signals exhibit a non-stationary behaviour other than trend, such as varying variance, see Section 3-2-4.

Experiments

Different settings of ARIMA models are tested, namely ARIMA$(1,1,1)$, ARIMA$(2,1,1)$, ARIMA$(1,0,1)$, and ARIMA$(0,1,1)$. The results for two AL time-series are given in Appendix C, the lower the scores, the better. The time-series is differenced in order to get rid of the trend and impose weak sense stationarity (see Section 3-2-4). For the most of the simulated and real data time-series the ARIMA$(1,1,1)$ and ARIMA$(2,1,1)$ have similar values of AIC, lower than the other tested models. However, the BIC is the lowest for the ARIMA$(1,1,1)$. Additionally, the difference between ARIMA$(2,1,1)$ and ARIMA$(1,1,1)$ is relatively small. Due to the fact that the less complex models should be preferred the ARIMA$(1,1,1)$ is chosen as the main model. A significant difference between the recall and precision scores for the two interpolation methods for the real data can be observed. The previous neighbor interpolation performs much better here than the linear interpolation.

Table 5-5: Results for the outlier detection using ARIMA method for prediction. Each time-series from simulations and the real data is fitted an ARIMA(1,1,1) model individually and the predictions are compared with the actual sample values.

	Ext. simulation	**Real data - previous**	**Real data - linear**
Recall	0.89	**0.63**	0.48
Precision	0.82	**0.83**	0.77

5-2-5 Local Outlier Factor

The next method approaches the outlier detection from a different angle in comparison to the already presented techniques. In detail, some points can be treated as outliers when compared globally with the rest of the set, but if one takes a more local view of the same set, it may occur that these points do not deviate from their neighbors enough. A method called Local Outlier Factor (LOF) derived from this approach and hence is able to use the local information about the point in order to find abnormal samples [46]. The authors define the notion of the local reachability density (lrd) which expresses the distance between the point being considered and its neighbors. Subsequently, the local outlier factor which is the average ratio between the lrd of the point and the lrds of its neighbors can be calculated.

This approach allows to perform outlier detection in a local manner. A time-series is treated as a set of points in a 2D space and the local outlier factor is computed for each point. If the value is much higher than 1, the point is identified as an outlier. One of the crucial parameters is the number of neighbouring points taken into consideration at all steps of the procedure, hereinafter noted with k.

The Python implementation of the LOF algorithm requires an a priori choice of the *contamination* parameter, i.e. what is the proportion of the abnormal samples in the data set (here in a single time-series). As this information is not known in advance, the algorithm tends to either miss some outlying samples or produce too many false positives depending on the chosen value of *contamination*.

Experiments

For the Local Outlier Factor method four different numbers of neighboring points and two contamination settings are tested. The combinations can be found in Table 5-6. As this is a 2D outlier detection method, scaling is applied on both axes. Both time and deformation are scaled to lie between 0 and 1. Later on, a scaling factor of 7 is applied on the time axis to maximize the performance. This value is obtained based on the results using different scaling factors. As the contamination value has to be fixed for this algorithm, there is a bound on the precision that can be obtained. The recall can get very high but it results in many false positives. Regarding the number of neighbors, higher values tend to give better results. However, these limitations cause that the method, although giving satisfactory results if the number of outliers is known, is not suitable for this kind of application. Especially in the real data, where only around 30% of the time-series has any unwrapping errors and often not more than 5.

Table 5-6: Results for the outlier detection using the Local Outlier Factor algorithm. The scores are computed for different settings with varying number of neighbouring samples taken into consideration k and the contamination value c, namely how many outliers are expected.

	$k = 3$	$k = 5$	$k = 10$		$k = 15$	
	$c = 0.033$	$c = 0.033$	$c = 0.033$	$c = 0.03$	$c = 0.033$	$c = 0.03$
Ext. simulation						
Recall	0.90	0.94	0.96	0.95	0.90	0.88
Precision	0.31	0.33	0.33	0.37	0.31	0.34
Real data - previous						
Recall	0.83	0.88	0.91	0.91	0.88	0.87
Precision	0.06	0.07	0.07	0.08	0.07	0.07
Real data - linear						
Recall	0.74	0.83	0.93	0.93	0.91	0.90
Precision	0.06	0.06	0.07	0.08	0.07	0.08

5-2-6 Summary

Several observations and remarks can be made for the performed experiments. For most of the methods, the samples at the beginning, and for some methods also at the end of the time-series, are used for the predictions and there is no way to predict values within these regions. The signals are relatively short and in some cases there is no error-free sequence that is long enough to be a good example for further detections. Moreover, the signals exhibit several types of different behaviors what makes it difficult to find a one-fits-all algorithm.

The results from the presented methods show a significant difference between the performance on the simulated datasets and the real data. There are several factors contributing to this outcome. First of all, the real data is more noisy, it has unexpected variations in the time-series which are difficult to handle for any of the algorithms. Consequently, the simulation

may not reflect the real data close enough. Another problem is that the real data is labelled by hand and there are many arguable samples where it is difficult to assess with certainty that it is an unwrapping error. Due to that fact, some of the ground truth data may not be fully consistent; a sample that is treated as an error in one time-series may be omitted in a slightly different one.

The performance of the detection is also hindered by the interpolation process, which inevitably introduces distortion to the signal and makes detrimental assumptions.

The goal was to find potential unwrapping errors in the InSAR deformation time-series and, if possible, correct them. For that purpose, for each of the methods presented in Section 5-2, a setting with the best recall and precision score (also compared using F_1 score) is chosen as the final parameter setting. Out of all methods, the best performance on the labelled set is recorded for the Neighboring Samples and the Moving Average Smoothing approaches (highlighted in the respective results tables). Additionally, among several settings with similar F_1 score, the higher precision is preferred over the higher recall. This is due to the fact that the case of a higher recall and lower precision leads to unnecessary corrections.

For the whole dataset the performance can be only assessed qualitatively as most of the data is not labelled in any way. It is possible to display the corrected time-series in the GUI (see Section 4-2-1), an example result is given in Figure 5-2. The red points and dashed connecting lines are the original samples and the corrected ones are displayed the same way as the other samples.

Figure 5-2: The GUI displays the corrected time-series #AL39. Red points are the original samples detected as the potential unwrapping errors and the new, corrected samples are plotted the same way as the rest of the time-series.

For the labelled low points in the Amsterdam dataset, the unwrapping error detection is able to recognize around two thirds of the potential unwrapping errors and a great majority of the detections is assessed correctly. For the high points, which seem to be more difficult to handle, the detection rate is lower than for the low points, however, the precision is still visibly high

and there are not many misdetections.

5-3 Segmentation

The goal of the segmentation is to divide the time-series into distinct, internally homogeneous subseries. The non-homogeneity of the time-series can potentially mean that there were significant changes, for instance, in the scattering point or measurement conditions. The segmentation problem can be also formulated as a changepoint detection problem, in which the goal is to find abrupt changes occurring in the time-series [47]. The changepoint detection has been applied in various fields, such as medical monitoring, human activity analysis, IoT systems, and environmental monitoring.

The segmentation techniques can be divided into supervised and unsupervised. The former, to which belong many of the machine learning methods, require a labelled dataset in order to learn the mapping between input and output data. For the changepoint detection such algorithms as decision trees, support vector machines, Bayesian nets, and various binary classifiers have been applied [47]. On the other hand, the unsupervised methods are able to find patterns without the need of providing labelled data. A group of methods that is well established and popular in the literature since decades are probabilistic methods, which are based on estimating the probability distributions. Examples are Offline Bayesian Changepoint Detection [48, 49], Bayesian Online Changepoint Detection [50], Adaptive sequential Bayesian changepoint detection [51], or a Gaussian Process [47]. Many papers that use the probabilistic approach share a common assumption, called *product partition model* [52], that a time-series can be divided into non-overlapping partitions and data in each partition ρ is i.i.d. from a probability distribution $P(x_t|\eta_\rho)$ [50, 52, 47].

The segmentation problem can be approached from different angles and there are numerous methods implementing various concepts. Methods based on piecewise linear representation are presented by Keogh *et al.* [53] and by Hu *et al.* [54], There are also segmentation methods drawing inspiration from control theory and system identification [47]. Further, optimization techniques and the total variation concept are used to tackle segmentation in [55] by Bleakley and Vert. A method based on least squares and time-series decomposition, called BFAST, is presented by Verbesselt *et al.* in [56]. Moreover, several kernel based and graph based methods are applied to perform changepoint detection [47]. Finally, Zhang and Huang explore adding contextual variables to the segmentation task in [57].

As in numerous fields nowadays, also in time-series segmentation there are works that make use of the most recent advances in machine learning and, especially, deep learning research. An example can be a breakpoint detection algorithm based on autoencoders [58]. A different approach can be found in [59], where the changepoint detection is performed using pyramid recurrent neural networks based on wavelets. The deep learning based techniques are discussed in more detail in Chapter 6.

In this section four different unsupervised methods are tested. The supervised approach to segmentation is discussed in Chapter 6.

5-3-1 Piecewise Linear Representation

There are multiple ways to obtain an approximate representation of a time-series, such as discrete Fourier transform (DFT), discrete wavelet transform (DWT), singular value decomposition (SVD), Symbolic Aggregate approXimation (SAX) [60] and piecewise linear representation (PLR) [54]. One of the goals of these kinds of representation is to reduce the dimensionality of the data. Moreover, it can be used in the similarity search and pattern discovery [54, 61].

In the piecewise linear representation method, a time-series is divided into non-overlapping, consecutive segments $S = (S_1, S_2, \ldots, S_k)$, where k is the number of segments. The segments are represented as linear functions [54, 61]. The objective is to find the optimal approximation, i.e. where the error between the linear representation and the actual data points is minimal.

A linear representation of a set of points can be obtained using interpolation or regression. In the case of interpolation, the resulting approximation line is connecting two data points at the boundaries of each segment. For the regression case, a line is fitted using the least squares approach. Figure 5-3 shows both methods. In the interpolation case the approximation is smoother and the computational complexity is lower. On the contrary, the regression provides a visually less intuitive result, but the fitting error is lower than in the interpolation case [54]. Additionally, using the interpolation approach, the data points at the breakpoints between the segments are treated as certain. After computing the segments, the piecewise representation relies fully on these boundary points, whereas a regression line is based on all the points in a calculated segment.

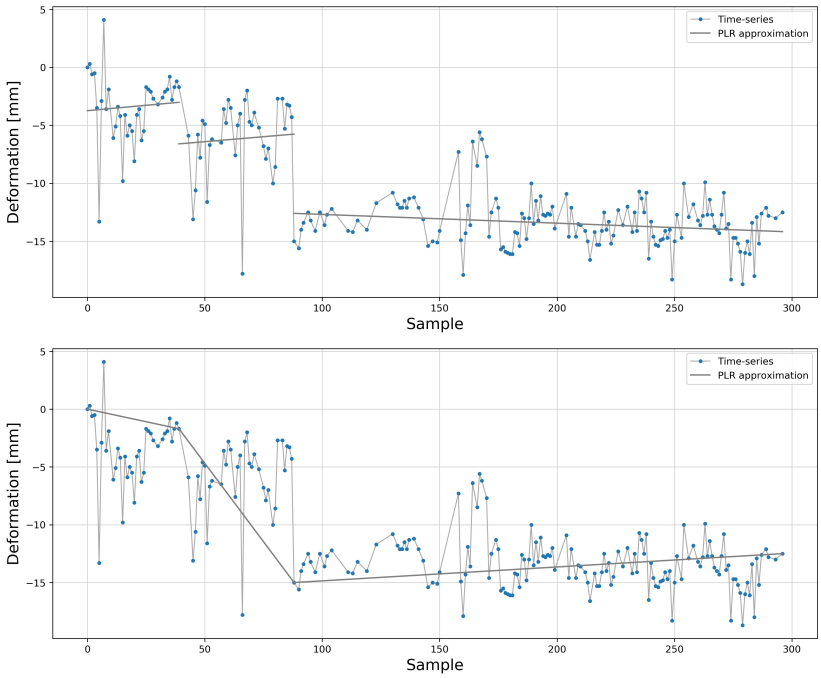


Figure 5-3: Two different types of fitting lines in PLR, the one using regression (top) and the one using interpolation (bottom), AL137.

There are three main general approaches to the segmentation task, namely Top-Down, Bottom-

Up, and the Sliding Window. While the Top-Down and the Bottom-Up require a time-series to be fully available, i.e. they are offline methods, the Sliding Window can also handle a streaming time-series.

Top-Down

In the Top-Down approach, at first, a whole series is divided into two segments, considering the most optimal location of the breakpoint, such that the difference between the two resulting segments is maximal. Subsequently, each of the segments is further divided in two the same way until the approximation error criterion is met [54, 61]. One of the main disadvantages is the method's inflexibility, as the breakpoints determined at the very beginning of the process may occur to not be the optimal ones [61].

Bottom-Up

On the contrary, the Bottom-Up approach starts from a time-series divided into $n-1$ segments, where n is the number of samples in the time-series. Then, the consecutive segments which treated together cause the smallest error increase are being merged together. This is repeated until the maximum approximation error is reached.

Sliding Window

The Sliding Window is an online method. At first a left boundary is determined and then the data points are sequentially assessed. The window size grows until the error does not exceed the maximum allowed error. A drawback of this approach is lack of a global view, due to focusing only on the subseries within the sliding window.

To improve the performance of the three aforementioned approaches several other solutions have been proposed. The most popular is the SWAB algorithm, which combines the Sliding Window and the Bottom-Up approach [53]. Other methods are PLR based on important data point (PLR-IDP), feasible space window (FSW), and stepwise FSW [54].

In order to define a stopping criterion for the PLR algorithms a maximum approximation error is set. Two approaches for choosing the maximum error are tested in this thesis. The maximum error for the whole dataset of time-series is either a fixed arbitrary value or is dependent on the standard deviation of the samples in a time-series. For each time-series two values of standard deviation are calculated, one using an original time-series and one after detrending it based on linear trend values provided in the dataset. Different coefficients are tested for the dataset with three interpolation scenarios (see Section 5-2). The average std of the detrended time-series is more than two times smaller than the average std of the original time-series. Table 5-7 presents a comparison of results for the fixed max error and the max error defined as a multiple of the std after detrending the time-series. Additionally, Figure 5-4 shows behavior of the recall and precision for a bigger range of fixed max values for the dataset of low points without interpolation. It can be observed that with increasing the value of the max error the precision rises at the cost of the recall.

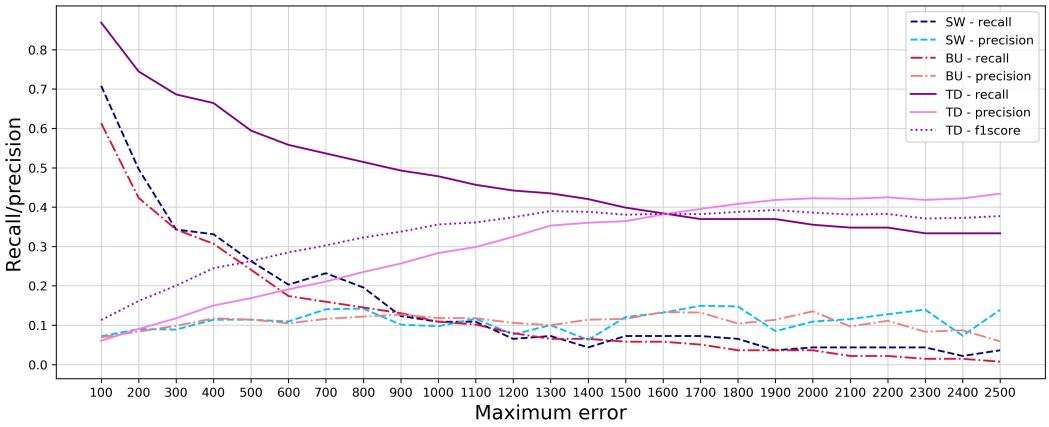


Figure 5-4: The recall and precision values for three PLR methods, i.e. Sliding Window (SW, dashed line), Bottom-Up (BU, dashdotted line), and Top-Down (TD, solid line), for different maximum approximation error values; low points dataset with linear interpolation. Additionally, the dotted line shows the F_1-score for the Top Down method.

Table 5-7: Piecewise Linear Representation - recall and precision for three PLR methods and different maximum error settings using two approches, i.e. fixed maximum error and maximum error as a multiple of a standard deviation of a time-series. Tolerance $= 5$ and std is computed for the detrended time-series. Ground truth collected as described in Section 4-2.

	fixed max error					× std				
	300	700	1000	1300	1900	200	300	400	500	600
Top-Down										
Recall	0.69	0.57	0.48	0.41	0.34	0.59	0.49	0.45	0.38	0.36
Precision	0.11	0.14	0.35	0.39	0.41	0.19	0.26	0.34	0.39	0.43
Bottom-Up										
Recall	0.33	0.15	0.10	0.06	0.05	0.22	0.13	0.07	0.06	0.04
Precision	0.11	0.14	0.16	0.17	0.25	0.13	0.12	0.10	0.11	0.10
Sliding Window										
Recall	0.30	0.10	0.06	0.03	0.04	0.23	0.11	0.06	0.10	0.04
Precision	0.11	0.10	0.10	0.07	0.14	0.12	0.09	0.06	0.14	0.08

As per Figure 5-4 and Table 5-7, the Top-Down method provides significantly better results than the other two. Table 5-8 presents more detailed results for the Top-Down method in all interpolation scenarios, using the max error based on the std of the detrended time-series.

The linear interpolation performs better than the previous neighbor method. An interesting observation is that the corrections of the potential unwrapping errors slightly impair the segmentation result in the low points dataset. An opposite situation is in the high points dataset, where the improvement is not big but it is consistent over all interpolation scenarios.

Table 5-8: Piecewise Linear Representation - recall and precision for the Top-Down method calculated for both low and high points dataset for three different interpolation approaches. Tolerance = 5 and the coefficient for the standard deviation is 600, std is computed for the detrended time-series. Ground truth collected as described in Section 4-2.

	AL		AH	
	original	corrected	original	corrected
previous interpolation				
Recall	0.31	0.29	0.35	0.36
Precision	0.34	0.34	0.26	0.28
linear interpolation				
Recall	0.36	0.33	0.33	0.35
Precision	0.43	0.41	0.29	0.31
without interpolation				
Recall	0.38	0.34	0.40	0.42
Precision	0.37	0.37	0.39	0.42

5-3-2 Bayesian changepoint detection

In multiple works, the authors take the Bayesian approach to the changepoint detection. Probabilistic methods are well-established and popular among researchers, with most of the works treating about offline detection [48, 49]. In this thesis, two algorithms are used for the time-series segmentation, namely, Bayesian Changepoint Detection as in works by Fearnhead [48] and Xuan and Murphy [49] and Bayesian Online Changepoint Detection by Adams and MacKay [50].

Fearnhead in [48] created an offline algorithm computing the posterior over the number and location of changepoints in a univariate time-series. Xuan and Murphy in [49] extended Fearnhead's work to handle multivariate time-series. The algorithm is efficiently implemented using dynamic programming [62]. The approach is related to the product partition models. There are two priors for the changepoints considered, a prior for the number of changepoints and a prior on their location. The latter is obtained from a point process for the time between two successive points, i.e. the length of a segment. The observation likelihood function is used for the modelling of $\Pr(y_{t:s}|t, s \text{ in the same segment})$, where t, s are time instances such that $s \geq t$.

The outcome of the algorithm is the posterior probability of a changepoint occurring at each time instance, see Figure 5-5. The bottom plot shows the probabilities and the upper plot shows the time-series. The segmentation lines are located at the time instances where the changepoint probability is above 0.3 threshold.

The implementation provides three prior functions: constant, geometric, and negative binomial function and three observation log-likelihood functions: Gaussian, the so called independent features presented in [49], and a full covariance model [49]. In the tests of a whole

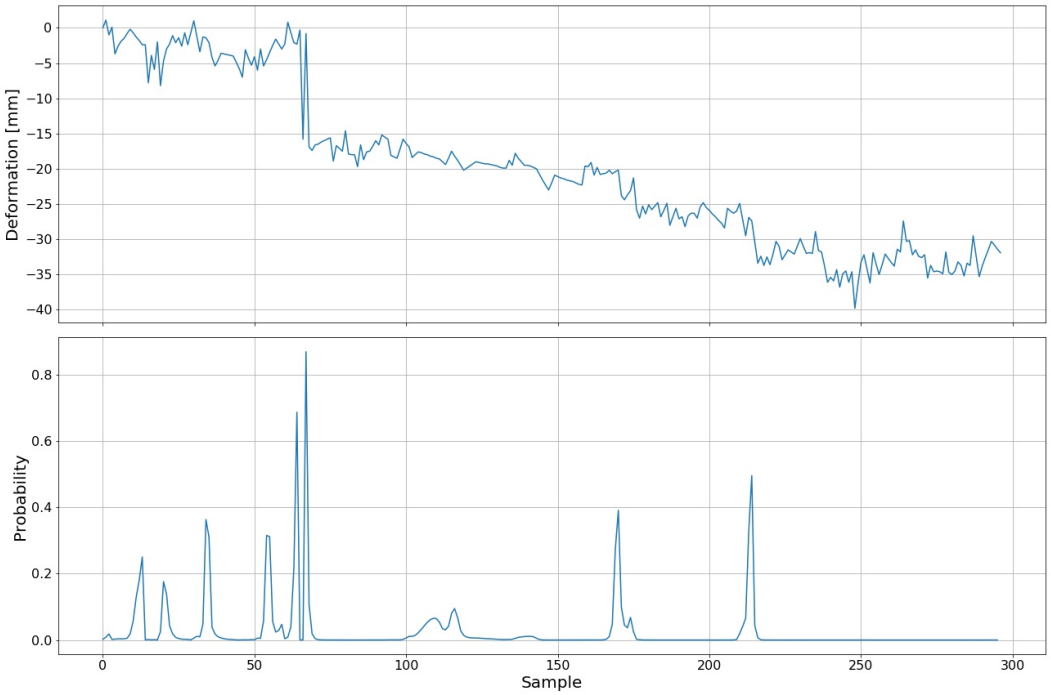


Figure 5-5: An example of the Bayesian Changepoint Detection algorithm outcome for a univariate time-series. A time-series from AL dataset with linear interpolation (top) and the probability score (bottom), AL869. The samples where the probability is higher than 0.3 are treated as changepoints.

dataset where each time-series is assessed individually a Gaussian observation log-likelihood is used. The independent features method and the full covariance model are tested for a multivariate case. Regarding the prior function, the probability of a changepoint occurring after a previous changepoint is assumed to be constant over the whole time-series and equal to $1/n$, where n is the length of a time-series.

The multivariate methods are tested on a particular group of time-series which exhibit high correlation with each other. Figure 5-6 shows an example of five similar time-series from the high points dataset. While processed individually, the changepoints in the time-series have a maximum probability score of around 0.4. However, when they are considered together, the collective score is very high. The method can be helpful in discovering hidden patterns in the time-series based on its neighbors or most similar points. Both methods, the independent features and the full covariance model provide similar results in the tested cases.

A different approach is taken by Adams and MacKay in [50]. The online method consists in modeling the run length r_t, i.e. the time since the last changepoint. The run length r_t is either zero, if the changepoint occurs at time t, or continues to grow:

$$r_t = \begin{cases} 0 & \text{if changepoint occurs at time } t, \\ r_{t-1} + 1 & \text{otherwise.} \end{cases} \tag{5-16}$$

The changepoint prior $P(r_t|r_{t-1})$ is used to model the transition probability. The probability of the next data point x_{t+1} can be computed by conditioning the predictive distribution on

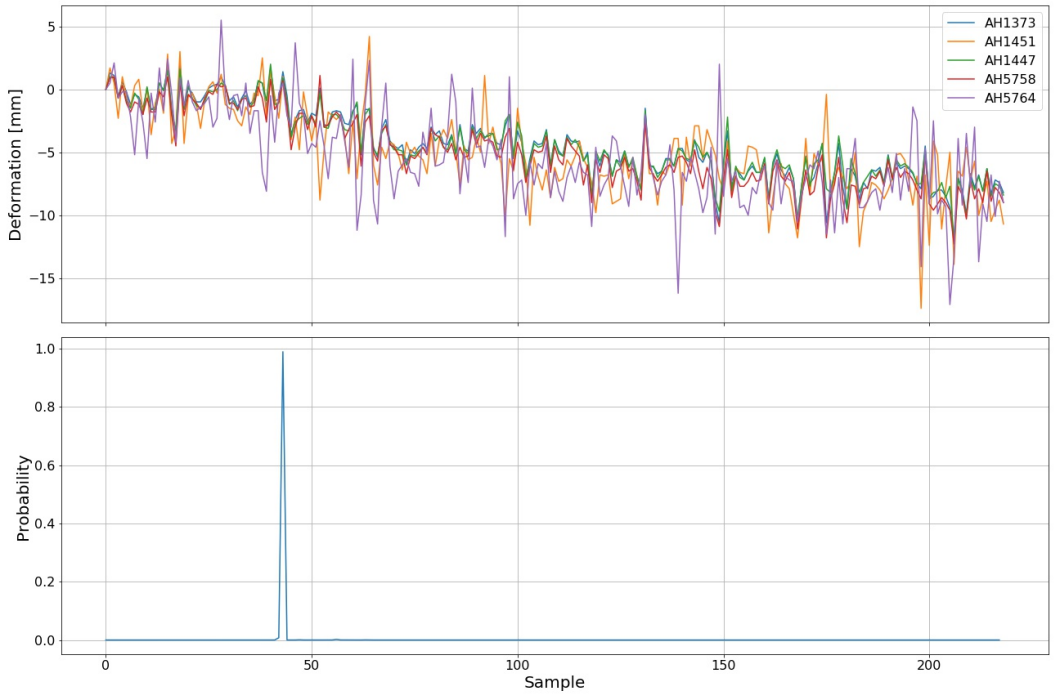


Figure 5-6: An example of the Bayesian Changepoint Detection algorithm outcome for a multivariate time-series. A set of five similar time-series from AH dataset with linear interpolation (top) and the probability score (bottom).

Table 5-9: Bayesian Changepoint Detection - recall and precision for offline and online approaches. The scores are computed for low and high points datasets, the original and corrected versions, in three interpolation scenarios. Tolerance $= 5$ and the threshold for the probability score is 0.3. Ground truth collected as described in Section 4-2.

	offline				online			
	AL		AH		AL		AH	
	orig.	correct.	orig.	correct.	orig.	correct.	orig.	correct.
previous interpolation								
Recall	0.61	0.62	0.65	0.64	0.63	0.64	0.50	0.51
Precision	0.19	0.20	0.14	0.14	0.13	0.13	0.08	0.09
linear interpolation								
Recall	0.64	0.64	0.61	0.62	0.62	0.62	0.47	0.48
Precision	0.21	0.22	0.15	0.16	0.14	0.14	0.09	0.09
without interpolation								
Recall	0.64	0.43	0.55	0.41	0.62	0.63	0.52	0.52
Precision	0.39	0.43	0.29	0.36	0.22	0.23	0.17	0.17

the run length r_t. Consequently, the marginal predictive distribution is expressed by

$$P(x_{t+1}|\mathbf{x}_{1:t}) = \sum_{r_t} P(x_{t+1}|r_t, \mathbf{x}_t^{(r)}) P(r_t|\mathbf{x}_{1:t}). \tag{5-17}$$

The run length posterior distribution is proportional to the joint distribution $P(r_t, \mathbf{x}_{1:t})$ and can be computed by a recursive message-passing algorithm. Joint distribution over the current run length and observed data is given by

$$P(r_t, \mathbf{x}_{1:t}) = \sum_{r_{t-1}} P(r_t|r_{t-1})P(x_t|r_{t-1}, \mathbf{x}_t^{(r)})P(r_{t-1}, \mathbf{x}_{1:t-1}), \tag{5-18}$$

where the first component is the changepoint prior. The changepoint prior is defined using the hazard function $H(\tau)$, which expresses the probability that the changepoint will occur, given that it has not occurred yet [63]. In [50] Adams and MacKay assume the a priori probability distribution over the interval between changepoints as a discrete exponential distribution with timescale λ. In such case the hazard function is constant $H(\tau) = 1/\lambda$, as the process is memoryless and therefore $H(\tau)$ does not depend on time.

The algorithm makes use of the properties of conjugate-exponential models. An advantage of conjugate models is that the posterior predictive has the same form as the prior predictive and only differs in hyperparameters. Furthermore, the strength of exponential family models is that they make inference possible with a finite number of sufficient statistics. Additionally, these can be calculated incrementally, what is beneficial for the computational efficiency. As the run length is modelled at each time point t, the parameters are updated one step at a time and the values become their priors for the next step. The implementation uses the Student's t-distribution as the observation likelihood, which hyperparameters are updated after each step.

As it is very challenging to assess whether a changepoint occurred after just one sample, there is a parameter N_w in the implementation which determines for how many samples the algorithm should "wait" until it decides whether a changepoint occurred at that previous time. The tests performed with different values of N_w show that the higher the value the less false positives are generated. This can be explained by the fact that in such case the algorithm has more evidence and it may be easier to assess data points in the window. Lower values of N_w raise the recall value significantly, however the precision score changes very slowly. As a result of experiments, the $N_w = 10$ is chosen for the further tests.

The tests are run for different values of a constant hazard function. Regarding the Student's t-distribution initialization, multiple settings are tested and the set of parameters (0.5, 0.005, 0.10, 0) is chosen as giving the best results in terms of the recall and precision. The results are presented in Table 5-9. The online method has worse scores than the offline one in all interpolation scenarios. Nonetheless, this is to be expected as there is a trade off between the accuracy of a method and its computational complexity.

In summary, the method is much faster than the offline Bayesian Changepoint Detection and allows to analyze even very long time-series without an excessive use of computational power. However, the need of making assumptions on the data distribution is an important drawback. Additionally, the method produces a lot of false positives. It should be noted that in this particular use case of relatively short time-series of InSAR measurements taken every 11 days, the computation time is not the most important factor.

Segmentation lines suppression

It can be observed that the Bayesian methods presented above produce segmentation lines which are located very close to each other, e.g. two consecutive points are marked as change-

points. This is not a desired behavior as such short segments could be attributed to outliers or noisy data, which not necessarily should be treated as separate segments. In order to limit the number of segmentation lines an algorithm inspired by Non-Maximum Suppression [64] is applied on the results of the time-series segmentation. It consists in selecting the segmentation lines that have the highest score and discarding the lines that are within a certain margin to the line with the highest score. This is only possible for segmentation methods that provide a score associated with the detected changepoint, such as probability. Tests are run on the results of the Bayesian Changepoint Detection with the minimum distance between segmentation lines set to 5 samples. For both AL and AH subsets, lines suppression indeed results in limiting the number of changepoints and increases the precision score from 0.39 to 0.43 for AL and from 0.30 to 0.36 for AH, for corrected datasets without interpolation. An example is shown in Figure 5-7. The green lines are the ones that are kept and the grey ones, the ones discarded.

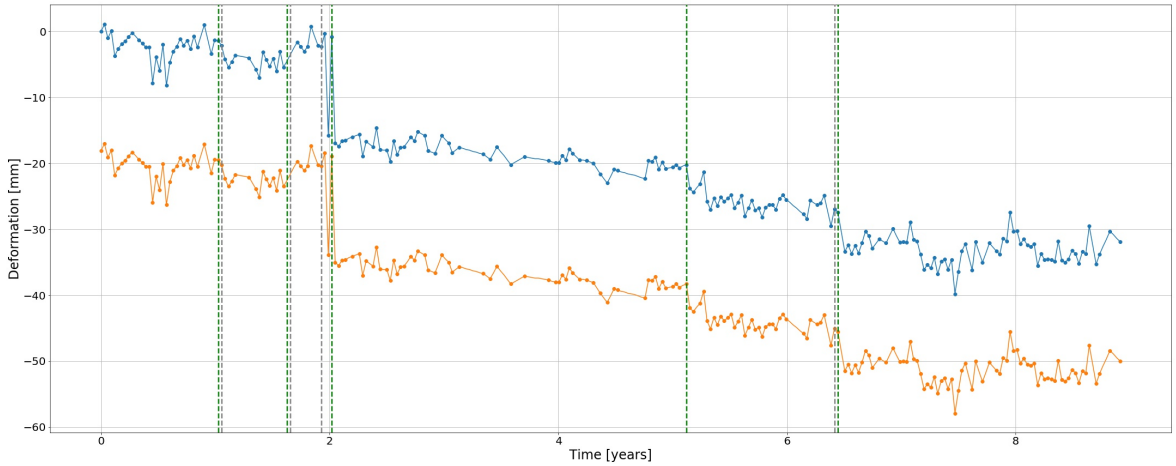


Figure 5-7: An example of the outcome of a segmentation lines suppression. Blue time-series is the original measurement and the yellow one is a -2π copy. The segmentation lines in green have higher scores than the grey ones and they are dominating the lower scored neighboring lines within the minimum distance of 5 samples. AL869.

5-3-3 BFAST

The algorithm BFAST, Breaks For Additive Seasonal And Trend, has been introduced by Verbesselt *et al.* in [56]. Verbesselt *et al.* in [56, 65] apply BFAST on the simulated and real satellite NDVI time-series data acquired by MODIS between 2000 and 2008. The method consists in iterative decomposition of the time-series into trend, seasonal, and noise components (see Eq. (3-3)) integrated with methods for change detection [56, 65]. It does not require an a priori choice of threshold or reference period. BFAST is able to find breakpoints (changepoints), in contrast to, for instance, STL procedure (see Figure 3-8 and Section 3-2-3), which is based on a locally weighted regression smoother (LOESS) [15]. The smoothing in the STL counteracts the detection of changes.

The model of the time-series in this approach is an additive model $Y_t = T_t + S_t + R_t$, where T is the trend, S is the seasonal, and R is the remainder component. The trend is assumed

to be piecewise linear with breakpoints $t_1^*, \ldots, t_m^*$. The model for trend component subseries is

$$T_t = \alpha_j + \beta_j t \tag{5-19}$$

for $t_{j-1}^* < t \leq t_j^*$ and $j = 1, \ldots, m$. The magnitude of the change is defined as a difference between linear models for subseries before and after a breakpoint. The seasonal component can have breakpoints at different times than the trend breakpoints and they are denoted as $t_1^\#, \ldots, t_p^\#$. The seasonal component can be expressed as

$$S_t = \sum_{i=1}^{s-1} \gamma_{i,j}(d_{t,i} - d_{t,0}) \tag{5-20}$$

for $t_{j-1}^\# < t \leq t_j^\#$ and where s is the seasonality period, $\gamma_{i,j}$ is the effect of season i, and $d_{t,i} = 1$ when t is in season i and 0 otherwise.

In the BFAST algorithm, ordinary least squares residuals-based moving sum (OLS-MOSUM) is used to check whether there are any breakpoints occurring in the time-series. The number of breakpoints is determined by the BIC. At first, $\hat{S}_t$ is estimated with a mean of all seasonal subseries. Then, one iteration consists of four following steps. First, OLS-MOSUM is used to detect any breakpoints in the trend component, which are estimated from a seasonally adjusted time-series $Y_t - \hat{S}_t$. Then, the trend coefficients α_j and β_j are calculated using regression and the trend estimate $\hat{T}_t$ is updated. The next two steps are analogous, but the breakpoints are computed for the seasonal component using detrended time-series $Y_t - \hat{T}_t$. The whole process is repeated until the breakpoints do not change.

In this thesis the BFAST algorithm is tested on twelve sets of time-series, i.e. low and high points are tested with original and corrected data in three interpolation scenarios. An R implementation of BFAST by Verbesselt *et al.* [66] is used in the experiments. There are three seasonal models that can be used to fit the seasonal component as well as detect seasonal breaks, namely `dummy` as given in [56], `harmonic` as in [65], and `none`, which means $S_t = 0$. All three settings are tested on all subsets. The best results are obtained with no seasonal model fitted and they are presented in the Table 5-10. This is due to the fact that in an uninterpolated scenario the BFAST is not able to handle gaps in the temporal data as the time-series is "squeezed" together causing the time-series to lose its seasonal structure. Also for the interpolated time-series, the algorithm performs better without the seasonal component, especially for the low points. Moreover, the linear interpolation is a better choice here than the previous neighbor interpolation. Nonetheless, the best scores are obtained for the time-series without any interpolation. Especially the precision is higher than in the other cases. The reason behind it may be the fact that there are discontinuities due to lacking measurements which make it more likely to put a segmentation line there. In fact, a human analyzing the time-series may have a similar impression and as the scores are calculated with respect to the human judgment, this may be a source of bias.

Usually not more than four iterations are needed to obtain the final breakpoints. A maximal number of breaks in a time-series is chosen as five. Figure 5-8 presents a seasonal decomposition of a time-series together with the breakpoint detection in the trend component. On the bottom plot, the time-series and its fitted trend are shown in more detail. There are three linear models with different slopes fitted.

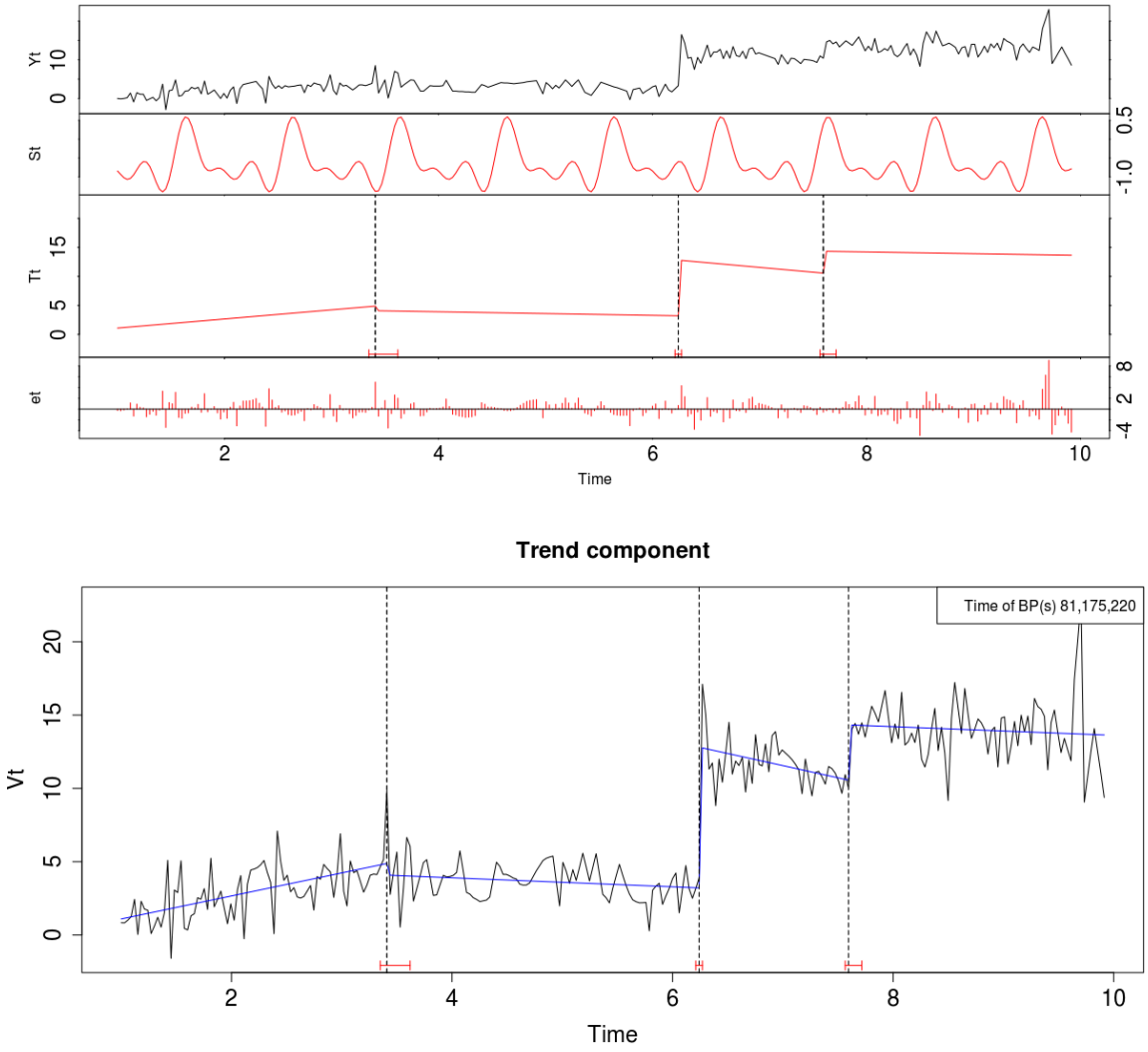


Figure 5-8: The decomposition into three components, trend, seasonal, and residual, with the breakpoints in the trend obtained with the BFAST method (top). The estimated piecewise linear trend with detected breakpoints and the time-series (bottom), AH14014.

The Table 5-10 presents results for different interpolations and for both original and corrected datasets. The tolerance level is five samples, i.e. a detection within five samples to the left or right on the time axis is still counted as a correct one. The results show that BFAST performs better on original time-series than on the corrected ones. After visual inspection it can be inferred that there are cases where uncorrected outliers cause that BFAST is less sensitive to variation of the data and is able to catch more general trends than after the corrections (e.g. AL1451).

5-3-4 Clustering methods

Additionally to the above presented methods, segmentation using several clustering techniques have been tested. The following methods were used in the experiments: K-means,

Table 5-10: BFAST - recall and precision scores computed for low and high points datasets, the original and corrected versions, in three interpolation scenarios. Tolerance = 5 and no seasonal model is fitted. Ground truth collected as described in Section 4-2.

	AL		AH	
	original	corrected	original	corrected
previous interpolation				
Recall	0.34	0.34	0.37	0.37
Precision	0.31	0.29	0.28	0.27
linear interpolation				
Recall	0.36	0.36	0.40	0.40
Precision	0.32	0.31	0.30	0.30
without interpolation				
Recall	0.31	0.33	0.37	0.37
Precision	0.39	0.39	0.44	0.41

Agglomerative Hierarchical Clustering, DBSCAN, Gaussian Mixture Model, and Gustafson-Kessel algorithm. Although for some time-series some of the methods gave good results, the performance of the clustering method was highly dependent on the shape of the time-series. A serious disadvantage in all tested methods except DBSCAN is the necessity of assuming the number of clusters beforehand. The time-series are treated as a set of points in a two-dimensional space (t, y) and the clustering is performed on this 2D space. This requires scaling of the data as the distance on the t-axis (time instance) does not correspond to the distance on the y-axis (milimeters). The scaling or normalization of the 2D set is impeded by the diversity of the time-series shapes, for instance, outliers or strong linear trends. In conclusion, whereas the clustering methods were able to handle some individual cases successfully, they failed in processing the whole diverse dataset with acceptable performance scores.

5-3-5 Conclusions

In this section four different approaches to the changepoint detection has been applied to the InSAR deformation time-series dataset. In almost all tested cases, the scenario in which no interpolation was applied and the calculations were performed on the "squeezed" time-series without any gaps occurred to produce the best results. The only exception is the PLR method for the AL dataset where it is the linear interpolation that shows the best results. Presumably this is caused by the fact that the analysis and the choice of coefficients were made mainly on the AL dataset with linear interpolation and this introduced a bias towards this setting.

The best results are provided by the offline Bayesian Changepoint detection algorithm. The online form is faster, less computationally expensive, and allows for the analysis of streaming time-series but the precision is evidently lower than in the offline case.

The BFAST algorithm's ability to isolate the seasonal component of the time-series is very promising, however, in the performed tests the best results are obtained if the seasonal com-

ponent is neglected. The reason behind it might be introducing the interpolation or, in case of "squeezed" time-series without interpolation, the loss of real temporal structure.

In general, the lacking measurements are a significant difficulty as they entail the need of interpolating the time-series or using only the available samples. Both approaches distort the time-series and hinder various transformations and the analysis of the time-series behavior. Ideally, no interpolation would be needed, but in reality the problem of missing data is sometimes unavoidable. Many computational methods cannot handle missing data and in such case linear interpolation showed to give better results than the previous interpolation.

Additionally, the PLR and BFAST are based on piecewise linear trends and are not able to take nonlinear trends into account. This may be a serious simplification, especially that 44% of low points and 45% of high points have a non-zero quadratic trend component in the default model (see Eq. (3-1)).

Providing a reliable human input is more difficult in the case of segmentation problem than in unwrapping error detection. The division of the time-series into separate segments is often not trivial and highly depends on the implicit and not clearly defined assumptions. Therefore, a human judgment can not only differ between experts but also between similar time-series assessed by the same expert.

Chapter 6

Machine learning approach

The current research trend of employing artificial intelligence, and especially machine learning, methods has not been missed in the geoscience and remote sensing fields. Among frequently used techniques are autoencoders, restricted Boltzmann machines, deep belief networks, and convolutional neural networks (CNNs) [2]. To name a few machine learning applications, one of the examples is hyperspectral image analysis. The land cover/use classification is performed with different types of CNNs, from 1-D till 3-D. Also the interpretation of SAR images or high-resolution satellite images is approached with CNNs or deep autoencoders. Another example is multimodal data fusion with different proposed solutions using CNNs, stacked autoencoders, or recurrent neural networks [2].

The researchers point out that deep learning-based unsupervised feature learning often outperforms handcrafted features [3]. Moreover, as more and more data is collected in cycles, time-series processing brings valuable insight but on the other hand, it also adds another dimension to the analyzed data, thereby complicating the computations. Deep learning methods can respond to the need of exploiting the temporal information together with spatial and spectral aspect of the data [2, 3].

However, remote sensing suffers from the same difficulty as many others, namely, only a few labeled samples is available. Another challenge, specific to remote sensing, is the multimodality of the observed data. The large number of bands in which the data is registered yields large volumes of data. Different specifics of various sources and different geometries of data to be used together make it far for trivial, especially in case of automated learning and black-box machine learning models. Moreover, rapidly growing amount of collected data stands behind the need for fast and efficient methods. Furthermore, although machine learning problems can often be reformulated as detection or classification tasks, this is not the case in the remote sensing research, where many problems cannot be reduced to widely used solution patterns. Additionally, there is a lot of well-established expert knowledge in the field and it should be included in the research using deep learning applications [2, 3].

Numerous examples of machine learning solutions applied to geoscience and remote sensing problems are listed in [67]. Lary *et al.* cite dozens of works, with many of the solutions other

than image-based, which are the most popular. An example that is specifically related to the SAR and InSAR techniques can be a problem of winter vegetation quality mapping which is approached using recurrent neural networks on multitemporal SAR images [68]. Also, in the work of Massinas *et al.* [69] convolutional neural networks are used for modeling ionospheric disturbances from InSAR data.

6-1 Preliminaries

In machine learning there are two main approaches to learning algorithms, namely supervised and unsupervised learning. In the supervised case the algorithm is exposed to example input data together with the outputs, called labels or targets. In the unsupervised approach the algorithm does not get feedback information about the correctness of its output. Its main purpose is the extraction of information from a dataset which has not been labelled. Examples of tasks performed by unsupervised learning algorithms are clustering, i.e. grouping of the data, density estimation, or denoising [70]. On the other hand, the supervised learning algorithms learn a mapping between the provided input and output data. There are also semi-supervised learning algorithms which fall between supervised and unsupervised methods. There, a small subset of labelled data is combined with unlabelled data to learn the representation for the whole dataset [70]. As acquisition of large sets of labelled data can be very expensive or can require a large amount of human labor, the semi-supervised approach allows for learning in situations where there is not enough data for supervised methods. Yet another type of machine learning algorithms is reinforcement learning, where an algorithm is not presented with example outputs as in supervised learning but it works on a trial and error principle and is only given a reward if the result is as expected [71]. In this chapter only supervised methods are considered.

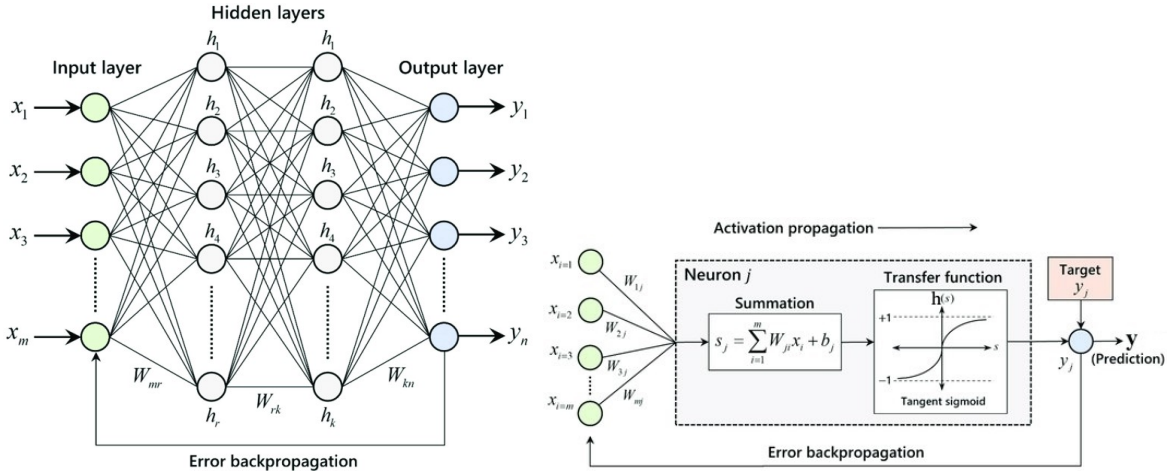
6-1-1 Supervised learning

The goal of a supervised learning algorithm is to learn a mapping between input and output data. This process requires labelling the data, namely providing the output for a given input in order for the algorithm to learn from it. This process can be laborious, expensive, or even infeasible for large amounts of data. Such a collected, labelled dataset is later divided into training, test, and validation datasets. The training of a machine learning algorithm, i.e. a process of learning, is performed only on the training set. The performance of the algorithm is then checked using a test set as it is desirable that the testing is done on the data that has not been seen during the training. Among the most popular supervised learning algorithms are probabilistic methods, such as logistic regression or naive Bayes, support vector machines, k-nearest neighbors, decision trees, or neural networks. There are two main groups of problems that can be solved with the supervised learning, namely classification and regression. In the former, the label or target is a category and in the latter, it is a real value [70, 72].

6-1-2 Neural Networks

Artificial neural networks (ANNs) owe their name to the inspiration from neuroscience and the biological neurons. As in other supervised learning techniques, the goal of a neural network is

to find an approximation of some function which maps the given inputs to provided outputs (or targets). A general representation of a neural network is shown in Figure 6-1a. The network consists of an input layer where the data enters the graph, an output layer where the data leaves the network, and the layers in between, called hidden layers. Each node in the



(a) General feedforward neural network architecture.

(b) Schematic of a single neuron with the input nodes, weights and biases, activation function (transfer function), and the output.

Figure 6-1: General representation of a feedforward neural network and its layers and the components of a single neuron.

hidden layer, also called a neuron or a unit, is a nonlinear function of a linear combination of the input nodes. The parameters in the linear combination are adaptive and their values are found in the process of training,

$$a_j = \sum_{i=1}^{m} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \tag{6-1}$$

where $x_1, \ldots, x_m$ are the input variables, $j = 1, \ldots, r$, where r is the number of neurons in the hidden layer, and the superscript indicates the layer. The adaptive parameters w_{ji} are called weights and w_{j0} are biases. a_j is called activation and it is an argument of an activation function [71]. The activation function $h(\cdot)$ introduces nonlinearity,

$$y_j = h(a_j). \tag{6-2}$$

Some of the most popular activation functions are sigmoidal function, hyperbolic tangent (tanh), or rectified linear unit (ReLU). In the output layer the neurons also have an activation function. The choice of the output activation function depends on the task, e.g. for regression this would be identity and for classification a sigmoidal function or softmax.

Overall, assuming that the network has only one hidden layer, one of the outputs of the network function can have the form [71]

$$y_k(\boldsymbol{x}, \boldsymbol{w}) = \sigma \left(\sum_{i=1}^{n} w_{kj}^{(2)} h \left(\sum_{i=1}^{r} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \right) + w_{k0}^{(2)} \right) \tag{6-3}$$

where $\boldsymbol{x}$ and $\boldsymbol{w}$ denote the input vector and weights vector, respectively, and σ is a sigmoid function. The information flows through the network from the inputs, through the hidden layers, till the outputs without any feedback connections. Therefore such network is called a feedforward network. Additionally, a network where all nodes from a preceding layer are connected to all nodes of the succeeding layer is called a fully-connected feedforward network.

The learning process in the neural network consist in feeding the training samples into the network and then comparing the output with the given target. The latter is defined by a loss function, also called an error function, which returns a loss value for each training sample. An average of losses for the whole dataset is the cost function. The loss function highly depends on the task and the objectives. For the regression it can be e.g. the mean squared error or the mean absolute error and for classification e.g. cross-entropy or Kullback-Leibler divergence [70, 71]. After calculating the loss, the error is back-propagated through the network, the algorithm is called back-propagation and it calculates the gradient of the loss function with respect to the weights. Then, the weights are updated accordingly, using a chosen optimizer, such as stochastic gradient descent or its modifications. This is done for all training samples in the dataset and one pass through the whole dataset is called an epoch. Usually, iterating over the training dataset is repeated for a chosen, fixed number of epochs. Other approaches are for instance early stopping of the learning algorithm if the loss function is not decreasing anymore for a given number of epochs [70, 71].

Convolutional neural networks

The convolutional neural networks (CNNs) are neural networks which, at least in one layer, use the convolution operation instead of general matrix multiplication. They are particularly suited for processing data with a grid-like topology, such as images (2-D) or time-series (1-D) [70]. The convolution operation is defined by

$$s(t) = (x * w)(t) = \int x(a)w(t-a)da, \tag{6-4}$$

and in the discrete case by

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a), \tag{6-5}$$

where x is the input, w is called a kernel or a filter, and the result is called a feature map [70]. In the specific case of a 2-D image I and a 2-D kernel K, the feature map can be computed by

$$S(i,j) = (I * K)(i,j) = \sum_{m}\sum_{n} I(m,n)K(i-m, j-n). \tag{6-6}$$

As the kernel is shifted over the whole input image, a CNN can work with variable input sizes, unlike the fully-connected network. Since the kernel sizes are smaller than the input images, only the neighborhood of each pixel has a direct impact on the output of the convolution step. Again, this is in contrary to the fully-connected network where an input is connected with all the nodes in the following layer. Building blocks of a convolutional network are three layers that typically follow each other and the groups can be repeated. First, the convolutional layer performs the convolution operation on the input. The next layer is the detector, also called

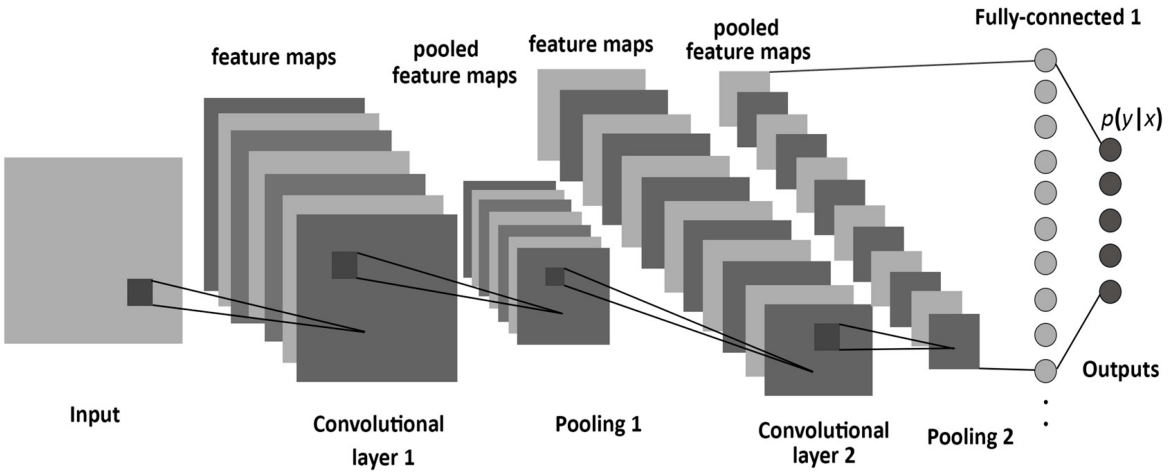


Figure 6-2: Representation of a CNN architecture. Convolutional layers are followed by pooling layers. At the end there is a fully-connected layer to transform the feature maps into the output, e.g. in classification task. Activation layer is often omitted in the architecture schemas and is treated as an intrinsic part of each convolutional layer.

activation, layer which introduces nonlinearity. Currently ReLU is the most commonly used activation function. The last, third layer is pooling. Pooling operation produces a reduced output that uses summary statistics of the neighboring outputs. For instance, max pooling takes maximum values out of the defined neighborhood, e.g. using a 2×2 window. This step makes the output invariant to local translation [70]. Typically, the convolutional layers are followed by a fully-connected layer to perform classification on the vector from feature maps. In such case, pooling helps in handling varying sizes of the input and allows for always the same size of the input to the fully-connected layer of the network. Moreover, kernel parameters are shared across the whole input, i.e. the kernel is the same at each location of the input. This makes the CNNs more efficient in terms of memory and storage compared to FCNs. Other parameters defining the CNN layers are kernel size, padding, and stride, i.e. by how many elements the kernel is shifted, typically set to one.

6-1-3 Pretraining and transfer learning

The problems to be solved by machine learning algorithms are becoming more and more complex and additionally, in case of supervised learning, it might be difficult or expensive to obtain a required amount of labeled data. A technique called transfer learning is able to use what has been learned in one setting and extend it to another setting [70]. The most popular transfer learning applications are image recognition and language models. In image recognition tasks, the low-level features learned by a base model can be repurposed in a new model to recognize a different categories of object than the original model. This can be achieved by e.g. using initial layers of the base network and then training the end layers on a dataset for a new problem. There are numerous models that have been trained on millions of data that are made publicly available and can be reused for transfer learning in similar applications.

Another way of handling an insufficient amount of labeled training data is to simulate the data

that closely resembles the original data and train a model based on the simulated data first. As a next step the model can be fine-tuned using a real dataset [73]. In this thesis, since there is not enough labeled training data in the InSAR time-series dataset, the approach of pretraining the models on a simulated dataset is taken. If models show satisfactory performance on the simulated dataset, the real time-series will be used for fine-tuning these models.

6-2 Network architectures and training

Two approaches are taken in the experiments in this thesis. The first is based on the raw data input. Whole time-series are fed into a neural network and multi-label classification is performed in order to determine whether there should be a segmentation line on a given sample or not. The second approach is to divide the time-series into smaller subseries and, by performing binary classification, to determine whether there should be a segmentation line within a given subseries. For each of these approaches different architectures are built and tested. For the experiments a Python Keras package (Tensorflow backend) is used to build neural network models. All of the experiments and trainings are initially prepared on the simulated dataset. This allows to use much more labeled time-series than only using the real dataset where only a small subset has been labeled.

6-2-1 Raw data input

A whole time-series is taken as an input to the network. The architectures that are used in experiments are fully-connected networks (FCNN) and convolutional neural networks (CNN). As the input is a whole time-series, the output is a vector of the same length as a time-series and for each sample (epoch) there is a 0/1 label associated, with 1 at the time steps where there is a division between the segments.

The whole simulated dataset is split into train, validation, and test sets with the proportion of 70%, 20%, and 10%, respectively. The training is done on the train set and the models are evaluated on the validation set. The test set is kept out and only used for getting the score from the final model in each of the approaches.

The simulated dataset contains artificially added unwrapping errors, therefore, each simulated time-series is first corrected using Neighbouring Samples method (see Section 5-2-1). Subsequently, data normalization is performed in order to limit the input values to the $[0, 1]$ range.

CNN

As a starting point, several arbitrarily chosen sets of hyperparameters are tested and the ones that show better performance are further used for tests with finer changes of hyperparameters. Ranges of values of hyperparameters are given in Table 6-1.

Both input and output layers are of length 297. The output activation layer is sigmoid and the loss function is binary cross-entropy. Sigmoid function returns a value between 0 and 1 and the threshold where the class should be treated as 1 varies depending on the

Table 6-1: Hyperparameters and their values used in experiments for CNN with raw data input architectures.

Hyperparameter	Values
number of layers (conv + fully-connected)	$1+1,\ 2+1,\ 3+1,\ 3+2,\ 4+1,\ 4+2,\ 5+2$
kernel size	$3, 5, 7, 9, 11, 13, 15, 21, 25, 45, 60$
number of filters	$64, 128, 192, 256, 512$
dropout size	$0.1, 0.3, 0.5$
optimizer	SGD, Adam
learning rate	$0.001, 0.0001, 0.0005$
fully-connected layer size	$512, 1024, 2048, 4096$

hyperparameters. Therefore, the results are presented and evaluated with respect to different threshold values. Activation function after convolutional layers as well as fully-connected layers is chosen as ReLU. After each convolutional layer and before an activation layer a batch normalization layer is added. Also, after fully-connected layers dropout is added in order to prevent overfitting. For coarse testing of different architectures the models are trained for maximum of 100 epochs, later the fine tuning is up to 200 epochs. Batch size for training is set to 32.

FCNN

Another architecture that is developed for the multi-label classification is a fully-connected network. Hyperparameters used in the experiments and their values are given in Table 6-2. Like in the CNN approach, the input and output are both of length 297. Activation function in the hidden layers is chosen as ReLU and the output activation is sigmoid. Similarly to CNN, different thresholds of the outputs are considered. The testing is run on maximum of 100 epochs and using a batch size 32. What is important, a fully-connected network on raw time-series is ignoring the structured character of the input data, which is a crucial information in the considered problem. Therefore, a lower performance compared to CNN approach is expected.

Table 6-2: Hyperparameters and their values used in FCNN experiments for raw data input.

Hyperparameter	Values
number of hidden layers	$4, 5$
number of nodes in hidden layers	$128, 256, 384, 512, 1024, 1536, 2048, 4096$
dropout size	$0, 0.3$
optimizer	SGD, Adam
learning rate	$0.001, 0.0001$

6-2-2 Subseries approach

In this approach, first the time-series is divided into subseries with a given length (window size) and offset, i.e. a shift by which the window is moved to create next subseries, see Figure 6-3. In this case, the performed machine learning task is binary classification with the objective to determine whether there is a segmentation line within the input subseries. Class (label) 0 means there is no segmentation line within a subseries and oppositely, class (label) 1 denotes a subseries with a segmentation line. An advantage of this approach is its lower complexity, binary classification is much easier than multi-label classification, and the possibility to work on extracted features. However, the downsides of this method are that it does not take global behavior of the time-series into account and it is only able to determine the segmentation point within a range (window size) and not in a specific sample.

Depending on the tested architecture, two ways of handling input data are used. In the experiments with a fully-connected neural network, features are extracted from the raw subseries and the network is trained on the features. In the experiments with a CNN however, raw subseries are used, in order to keep the structured character of the data.

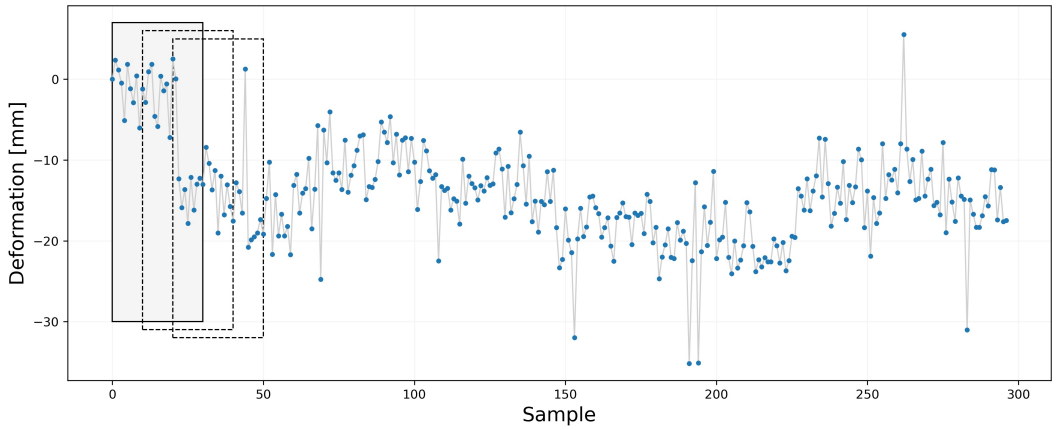


Figure 6-3: A division of the time-series into subseries. A window of defined width (number of samples) and with defined offset (by how many samples the window is shifted) is shifted through the whole time-series and generates subseries of given length. An example window of width 30 and offset 10 is shown in grey. Two next windows are marked with dashed lines.

Feature engineering and selection

For each of the subseries different features describing the subseries are extracted. The following features are computed for the whole subseries, as well as for the first and the second half of the subseries separately: mean, median, variance, third moment (skewness), and fourth moment (kurtosis). Additionally, the same set of features is extracted for a subseries which is differenced once and twice. This yields 45 features for each subseries that describe the whole subseries and its left and right halves separately.

An algorithm for feature selection called Boruta algorithm [74] is used to check the relevance of the above features. In result, all of the above 45 features are assessed as relevant and are kept. An additional set of features is obtained using a Python package *tsfresh* [75] which

computes a great number of different time-series characteristics, giving 434 features. These are subsequently also passed through the Boruta algorithm in order to only keep the features that are relevant in this particular problem of binary classification. In result, after removing features that are not relevant and the ones with infinite values, which cannot be handled in a neural network, 242 features are kept as relevant.

Dataset split and class imbalance

The whole simulated dataset of subseries has to be divided into train, validation and test sets, with the same proportion as in the raw input approach, namely 70%, 20%, and 10% respectively. The subseries are divided based on the original time-series they are a part of, in order to avoid a situation where subseries which are only a few samples apart (only one shift apart) would end up in both train and test set. That could bias the results of the learning process since testing on a very similar subseries would be close to testing on a subseries from the train set. Additionally, there is a heavy class imbalance in the dataset, since class 0, i.e. no segmentation line in a subseries, is much more than class 1. In the training set there are 141733 samples with class 0 and only 26267 with class 1. Since class 0 makes up $\sim 84\%$ of the whole training set (168000 samples in total), a model which learns to always output class 0 would have 84% accuracy. Therefore, a limited number of samples with class 0 is taken for the training to keep balance between both classes. At first all class 1 samples are taken and then the same number of samples is randomly selected from class 0 samples.

FCNN

At first, the features are normalized to have values within $[0, 1]$ range, since originally features have values of different orders of magnitude. The loss function is binary cross-entropy and the output activation function is sigmoid. Two separate networks are developed for automatically generated features and for simple, hand-crafted features. The trainings start with arbitrary chosen networks using hyperparameters from Table 6-3 and Table 6-4. The input has 242 features in the first case, andt 45 in the second. Both optimizers, SGD and Adam, are tested, with different learning rates. All activations in hidden layers are chosen as ReLU. The networks are trained for 100 epochs, the batch size is set to 32 and the validation split for training is 0.2. In some cases, the models are re-trained again for a short number of epochs using the same training set.

Table 6-3: Hyperparameters and their values used in FCNN experiments using automatically generated features.

Hyperparameter	Values
number of hidden layers	$2, 3, 4, 5$
number of nodes in hidden layers	$32, 64, 128, 192, 256, 384, 512$
dropout size	$0.1, 0.3, 0.5$
optimizer	SGD, Adam
learning rate	$0.01, 0.001, 0.0001$

Table 6-4: Hyperparameters and their values used in FCNN experiments using simple, hand-crafted features.

Hyperparameter	Values
number of hidden layers	$2, 3$
number of nodes in hidden layers	$16, 32, 64, 96$
dropout size	$0, 0.1, 0.3$
optimizer	SGD, Adam
learning rate	$0.01, 0.001, 0.0001$

CNN

In the CNN approach only raw values in subseries are used. Hence, data normalization is performed sample-wise and not feature-wise, as in the FCNN case. The same as in the FCNN case, the loss function is binary cross-entropy and the output activation function is sigmoid. Hyperparameters and their values are given in Table 6-5. The approach to experiments is the same as in the CNNs for raw inputs, at first arbitrarily chosen sets of hyperparameters are tested and then, depending on their performance, they are further fine-tuned. As the subseries are of length 30 and not 297, the kernel sizes are smaller accordingly.

Table 6-5: Hyperparameters and their values used in experiments for CNN with raw data input architectures.

Hyperparameter	Values
number of layers (conv + fully-connected)	$1+1,\ 2+1,\ 2+2,\ 3+1,\ 3+2$
kernel size	$3, 5, 7, 9, 11$
number of filters	$32, 64, 128, 192$
dropout size	$0.1, 0.3, 0.5$
optimizer	SGD, Adam
learning rate	$0.001, 0.0001, 0.0005$
fully-connected layer size	$32, 64, 128, 256, 512$

6-3 Evaluation and results

In this section the tested models are evaluated and the results of the experiments are described. For each of the approaches, the best model is chosen based on validation set and then its performance is given based on its performance on the test set. Then, these best models are applied on the labeled subset from the real dataset. Apart from the validation set, the models are also checked on a trivial example (a constant line with two steps) to make sure the model can handle a simple case that is different than the ones seen in training but also having separate segments.

6-3-1 Raw data input

CNN

Numerous experiments with different sets of hyperparameters are run in order to find the most suitable values for the hyperparameters. At first, an architecture with only one convolutional layer and one fully-connected layer is tested to check the influence of the kernel size on the output results. In general, results have shown that, up to some point, the bigger the kernel size, the better the scores. The first convolutional layer captures high-level features, therefore, a longer kernel might be able to find more long-term changes of the time-series.

Adding a second convolutional layer with a higher number of filters and a smaller kernel than in the previous layer improved the results. Moreover, an additional fully-connected layer after flattening the feature maps (step between convolutional layers' feature maps and the fully-connected layers, see Figure 6-2) has brought further improvement.

After initial experiments with changing mainly the number of layers, number of filters, and kernel sizes, four best performing models are chosen for comparison and fine-tuning to choose the final model out of them.

The performance of the models is evaluated using recall, precision, and F_1-score. The tolerance is set to 5, i.e. a predicted line can be off by 5 samples in both directions and still be counted as a correct prediction. The output of the model is first reduced using line suppression algorithm (see Section 5-3-2). Then the scores are generated for several cut-off thresholds between classes 0 and 1. If the output of the model at given time step (epoch) is greater than the threshold, then the predicted class is 1. The scores are presented for results before and after line suppression to show the improvement the line suppression provides.

The four presented models are differing in the number of layers and kernel sizes. Further they are referred to as models A, B, C, and D. Model A is the simplest with only one convolutional layer with 32 filters and a kernel 25. The best performance was found for two fully-connected layers of sizes 2048 ans 1024. After each fully-connected layer dropout of value 0.5 is added. Model B has two convolutional layers with 64 filters and kernel 60 and 128 filter with kernel 45, respectively. Model C has two more convolutional layers added on top of the layers from model B: 192 filter and kernel 25 and 256 filters and kernel 7. Finally, model D also has four convolutional layers with the same number of filters but the kernel sizes are much smaller, 15, 9, 5, and 5, respectively. In this model a smaller dropout of 0.3 proved to be better than the 0.5 one.

In all the models the Adam optimizer and a learning rate of 0.0001 gave the best results. Fully-connected layers of sizes other than 2048 and 1024 only worsened the final results, therefore this same structure is kept across all four models.

Figure 6-4 shows the recall and precision at several thresholds for each of the four models. Dashed lines represent the scores after line suppression. It can be observed that the line suppression significantly improves the precision scores. The best scores, with the highest F_1-score, are given in Table 6-6. For models B and D two pairs of recall/precision scores are presented because they have the same F_1-score but different ration between recall and precision. The decision which threshold to keep can depend on the needs, whether more detections with lower precision are preferred or less detections but more precise.

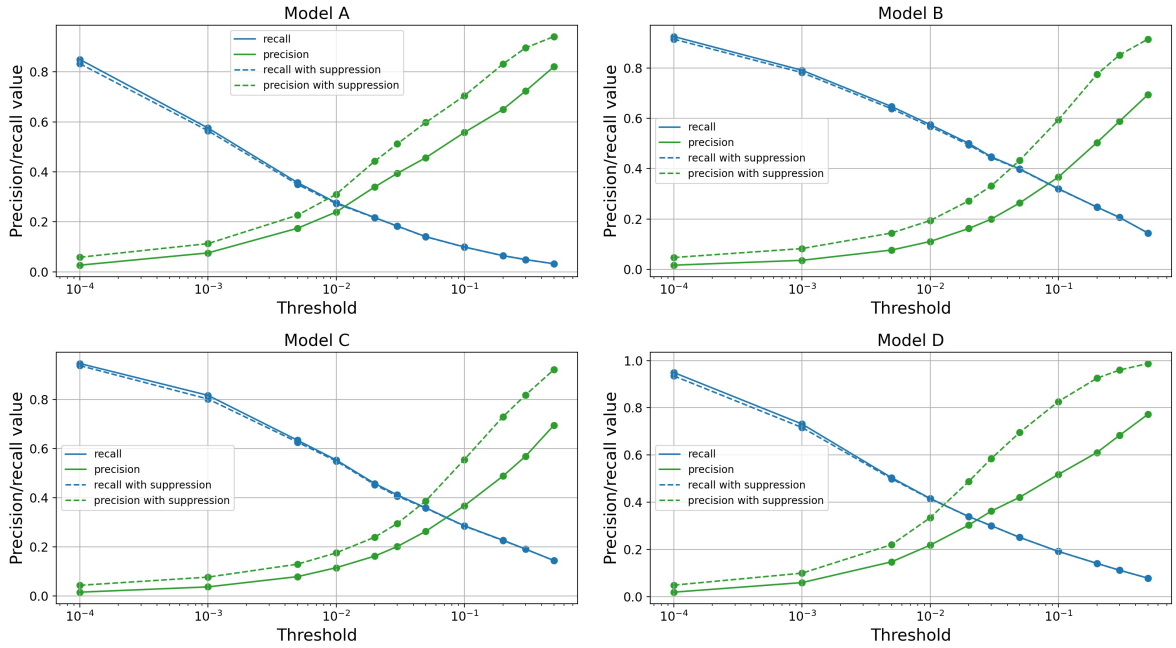


Figure 6-4: Recall and precision scores at different thresholds for four CNN models. Solid lines show original scores and dashed lines - scores after line suppression. The higher value of precision and recall, i.e. both recall and precision lines closer to the top, the better performance the model offers. Recalls close to 1 with very low precision mean that there are a lot of false positives. Additionally, using line suppression (dashed lines) significantly limits the number of false positives and improves the performance.

Table 6-6: Four CNN models scores on the validation set. All scores are given after line suppression and at the best threshold. For models B and D values in brackets show scores with a very close F_1-score but with different ratio between recall and precision.

	model A	model B	model C	model D
best threshold	0.02	0.05 (0.1)	0.1	0.02 (0.03)
recall	0.22	0.40 (0.32)	0.28	0.34 (0.30)
precision	0.44	0.43 (0.59)	0.56	0.49 (0.59)
F_1-score	0.29	**0.42** (**0.42**)	0.38	0.40 (0.40)

In result, model B shows best performance on the validation set and is chosen as the final model with the threshold 0.1. A scheme with all layers is shown in Figure 6-5 and a detailed scheme with the shapes of the layers can be found in Figure B-1.

The scores on the test set and on the labelled subsets from real InSAR dataset AL and AH are presented in Table 6-7. As expected, the scores on the real data are lower than on the test set. However, the model was only trained using the simulated dataset and the real dataset can be treated as more difficult to handle, also because of the interpolation. Figure 6-6 shows two InSAR time-series with labelled segmentation lines and the predicted line, output from the final CNN model. Top figure shows one correctly predicted line (tolerance is set to 5 samples) and one misdetection, however, the labelled segmentation line is not objectively true, it is an

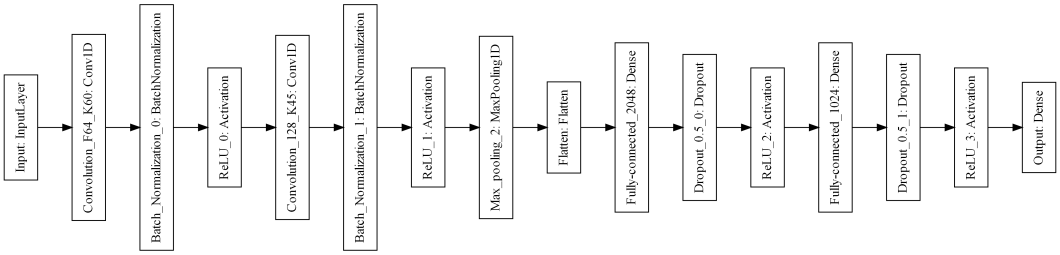


Figure 6-5: Architecture scheme for the final CNN model.

opinion of the labelling person. In the bottom figure a predicted line is close to the correct label but due to interpolation it is more than five samples away from the label and is not counted as a correct prediction.

Table 6-7: Results of the final CNN model with the raw data input on test set and two real datasets, AL and AH. Ground truth collected as described in Section 4-2.

	test set	real data AL	real data AH
recall	0.32	0.27	0.25
precision	0.58	0.33	0.36
F_1-score	0.41	0.30	0.29

FCNN

At first an arbitrary set of hyperparameters is used as a starting point. A model with four layers of sizes between 128 and 512 shows poor performance on the validation set. Therefore, networks with more units in the hidden layers are tested. Adding dropout layers improves the scores in all cases. A model with four layers of sizes 512, 1024, 2048, and 512, respectively, is chosen for comparison and is referred to as Model E. Subsequently, further increasing the number of units in the biggest layer to 4096 brings improvement. This model is referred to as Model F. In order to check whether the performance would constantly improve with bigger networks, models with more layers are tested but their scores are worse than Model F. Also tests with removing dropout show that this is not a correct way, keeping dropout with rate 0.3 results in better scores. In multiple models raising the learning rate from 0.0001 to 0.001 causes the training to be unstable and gives very poor results. Moreover, choosing SGD optimizer leads to very poor results and hence, Adam is kept as the optimizer.

Table 6-8 presents the results of models E and F on the validation set and Figure 6-7 shows the plots with values of recall and precision (after line suppression) for different thresholds. Consequently, model F is chosen as the final model in this approach as it produces slightly better results than model E. The architecture of the network is shown in Figure 6-8 and detailed scheme is presented in Figure B-2.

In general, the tested FCNN models produce more predictions than the CNNs, resulting in higher recall scores. This can be observed by comparing Figure 6-4 and Figure 6-7 where the recall line is closer to value of 1.0 for FCNN models, compared to CNN models.

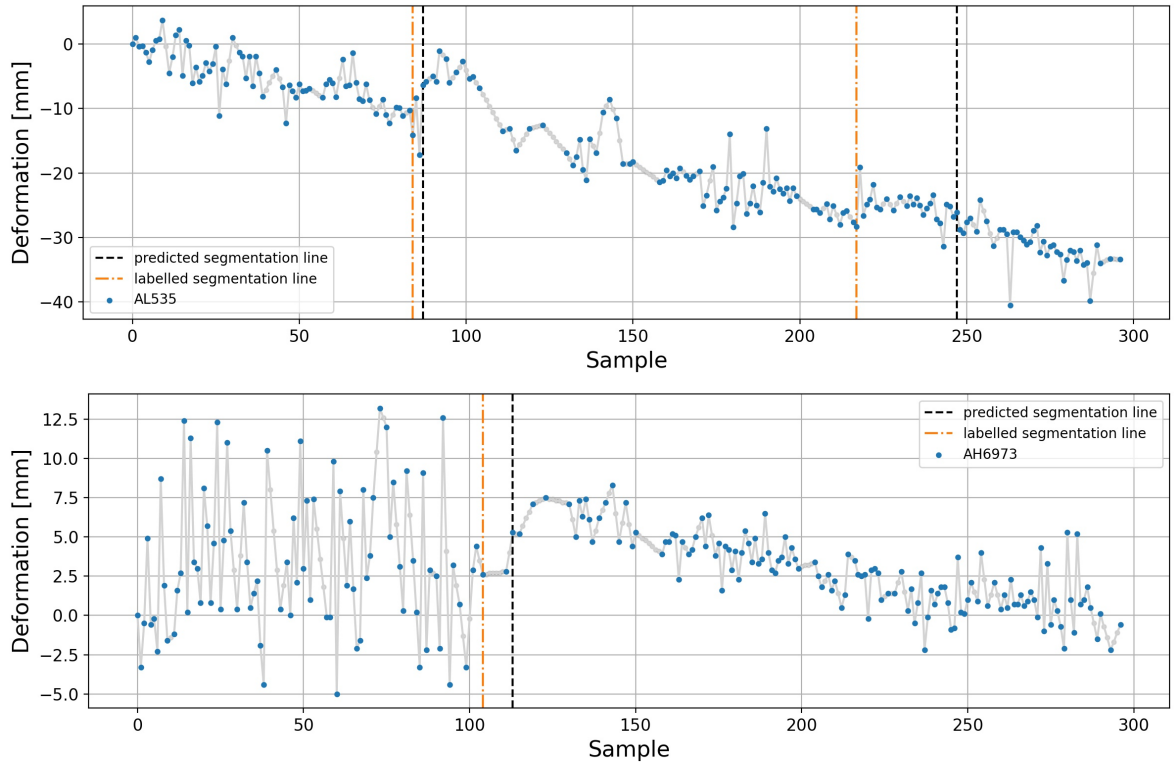


Figure 6-6: Two examples of the CNN final model outputs on AL535 (top) and AH6973 (bottom) time-series. Dashed lines represent predicted segmentation lines and dash-dotted - the labelled ones. Interpolated points are shown in gray and gray lines are connecting the points for readability. Ground truth collected as described in Section 4-2.

Table 6-8: Two FCNN models scores on the validation set. All scores are given after line suppression and at the best threshold.

	model E	model F
best threshold	0.1	0.1
recall	0.29	0.28
precision	0.28	0.33
F_1-score	0.29	**0.31**

The final scores on the test set and on the real datasets are presented in Table 6-9. As expected, the results are not as good as in the CNN model. Figure 6-9 shows the outputs from the FCNN model on the AL and AH datasets. In the top figure two out of three predicted lines are close to the labelled ones. In the bottom figure in the middle of the time-series, the network finds two segmentation lines next to each other but one can see that in fact the left one, which is overlapping with the labelled line, and the right one are only divided by a fully interpolated part, effectively making it lines in two consecutive time steps.

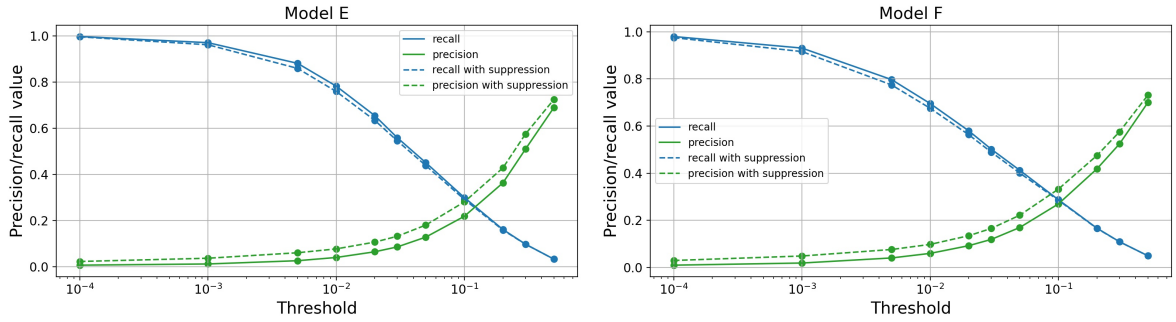


Figure 6-7: Recall and precision scores at different thresholds for two FCNN models. Solid lines show original scores and dashed lines - scores after line suppression. The precision and recall lines are closer to their boundary values which means that there are mainly false positives produces by the models. The recall and precision lines cross at a relatively small value showing a lower F_1-score.

Table 6-9: Results of the final FCNN model with the raw data input on test set and two real datasets, AL and AH.

	test set	real data AL	real data AH
recall	0.28	0.22	0.18
precision	0.32	0.25	0.24
F_1-score	0.30	0.26	0.21

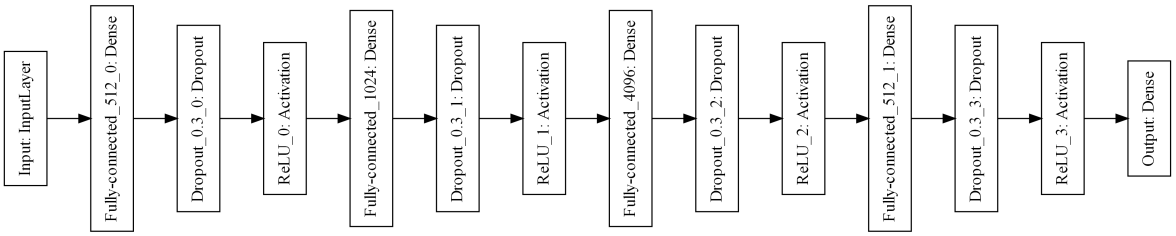


Figure 6-8: Architecture scheme for the final FCNN model for raw data input and multi-label classification.

6-3-2 Subseries approach

The threshold between classes is set to 0.5, which is a default value in the sigmoid output as the sigmoid function is symmetric around 0 with a value of 0.5. It proves to give the best results in terms of F_1-score, other values favor either recall or precision but at the cost of the overall F_1-score.

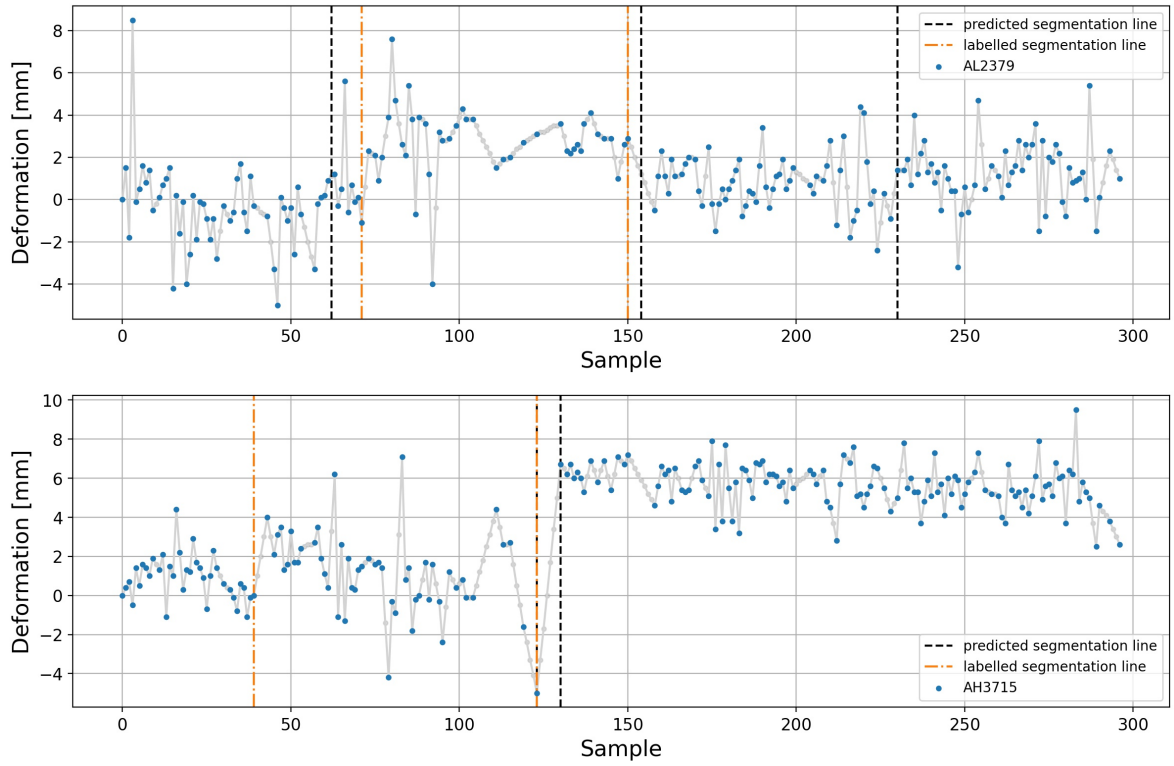


Figure 6-9: Two examples of the FCNN final model outputs on AL2379 (top) and AH3715 (bottom) time-series. Dashed lines represent predicted segmentation lines and dash-dotted - the labelled ones. Interpolated points are shown in gray and gray lines are connecting the points for readability. Ground truth collected as described in Section 4-2.

FCNN

Automatically generated features

The training starts with an arbitrary chosen network. The input has 242 features so a relatively big network, with five layers, is tested at the beginning. However, there occurred a problem with overfitting and in order to mitigate it higher dropout and smaller networks are investigated. Raising dropout rate to 0.5 does not help in this case, also a four-layer network gives similar results. For a change, a smaller, two-layer network is tested, but the results are worse than in the previous experiments. Across different settings, SGD optimizer with different learning rates performs worse than Adam. In the end, a network size in the middle, namely three-layer model but with less units per layer than before gives best results among all experiments: 128, 196, and 64 units per layer accordingly. Further lowering the number of units does give as good results. Dropout of rate 0.3 is a better choice in this case than 0.5 or 0.1. The best model scores 0.48 recall, 0.47 precision, and 0.48 F_1-score. Figure 6-10 shows the output of the model on a validation set. Most of the windows are correctly predicted to contain a segmentation line. On the right hand side there are two windows that are missed (orange, \pattern which does not overlap with blue / pattern). The results on the test and real datasets are given in Table 6-10 and the architectures are shown in Figure 6-11 and Figure B-4. Although the results on the test set are relatively good, the scores on the real

data do not show similar performance and the recall is much higher than precision. This is reflected in Figure 6-12 where there are much more predicted windows compared to ground truth.

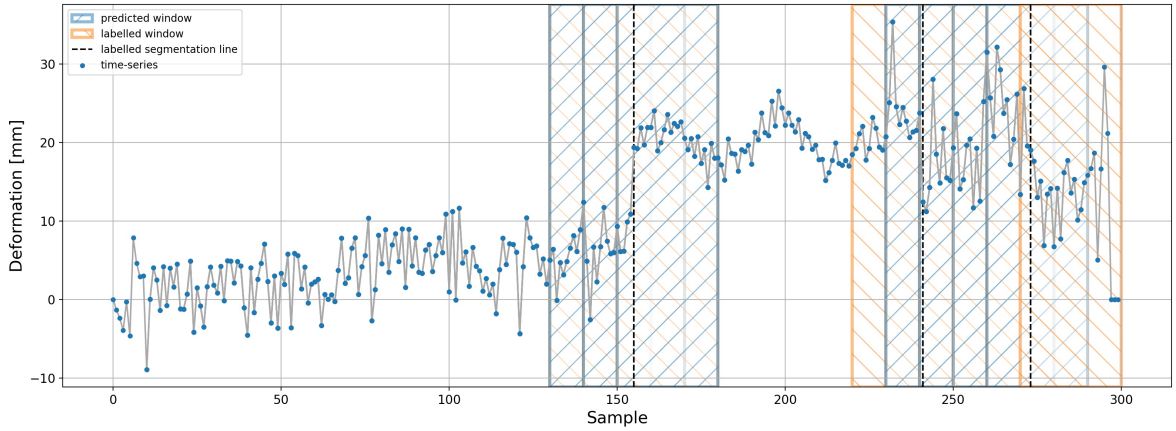


Figure 6-10: A simulated time-series with windows predicted to contain a segmentation line (blue, / pattern) and windows labelled to contain a segmentation line (orange, \ pattern). Areas where both pattern cross show where the prediction agrees with ground truth. In the middle of the time-series the final FCNN model (automatically generated features is able to correctly find a segmentation line in all three windows in which the line is included. On the right hand side, there are two lines to be detected, the model is able to partially find them but not in all windows.

Table 6-10: Results of the final FCNN model with the subseries input (automatically generated features) on test set and two real datasets, AL and AH. Ground truth collected as described in Section 4-2.

	test set	real data AL	real data AH
recall	0.50	0.69	0.56
precision	0.47	0.17	0.16
F_1-score	0.48	0.28	0.25

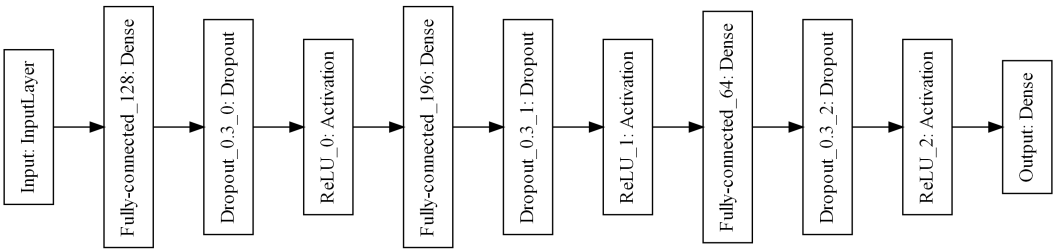


Figure 6-11: Architecture scheme for the final FCNN model for subseries with automatically generated features.

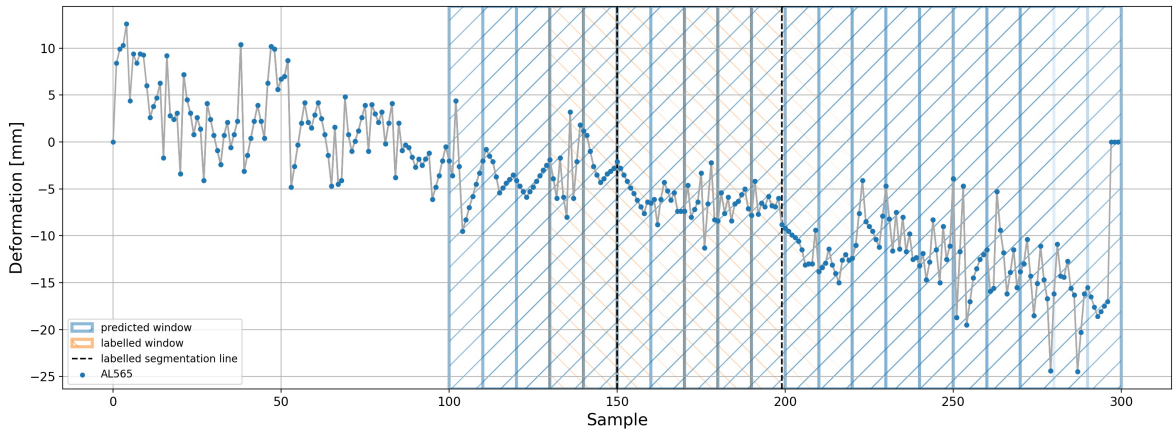


Figure 6-12: Output of the FCNN model with automatically generated features on a real time-series AL565. The model produces a lot of false positives, i.e. predicts windows to contain a segmentation line when in fact there are no ground truth windows there.

Simple hand-crafted features

Since there are only 45 features, this network is much smaller, only two or three layers. This however makes it prone to overfitting. Bigger rates of dropout do not help overcome the issue, the best results are obtained with 0.1 or no dropout. Also here SGD does not yield any valid results. In the final model, which is found to have 32, 64, and 16 units on the respective layers, recall is much higher than precision and therefore the model is retrained again for several epochs with a smaller learning rate. The score on the validation set is 0.44 recall and 0.47 precision, which give F_1-score of 0.45. Figure 6-13 shows the output of the model on a simulated time-series. This is the same time-series as in Figure 6-10 and the simple features model produces less predicted windows in this case compared to the model with automatically generated features. This is consistent with a lower recall score for the simple features model.

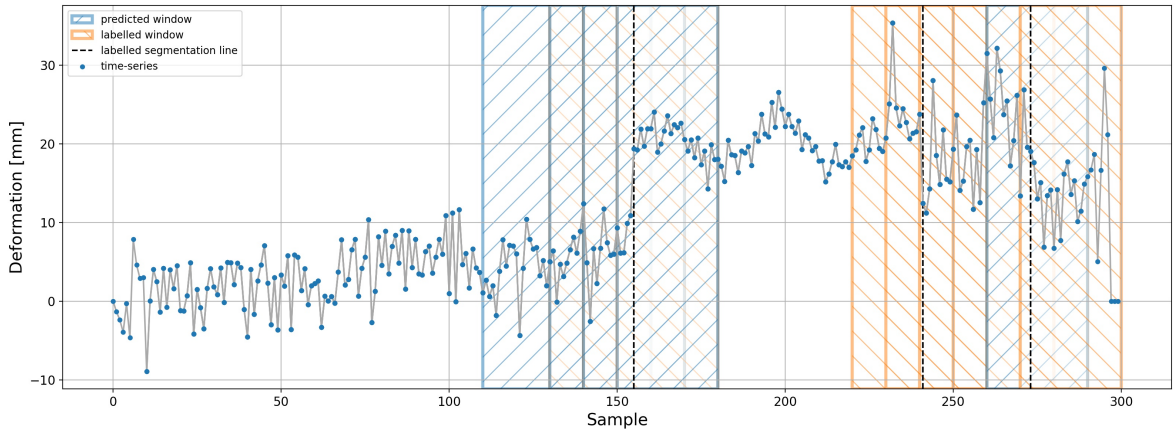


Figure 6-13: The output of the model for the same simulated time-series as in Figure 6-10. This model has lower recall than the model with automatically generated features and indeed, there are less predicted windows in this plot than in Figure 6-10.

The final results on the test and real datasets are presented in Table 6-11. The architectures

are shown in Figure 6-14 and Figure B-5. The performance of the model on an example from the real dataset is presented in Figure 6-15. One of the segmentation lines is found with all the windows covering it predicted correctly. Another line, in the middle of the time-series, is missed but there are no false positives produced.

Table 6-11: Results of the final FCNN model with the subseries input (simple features) on test set and two real datasets, AL and AH. Ground truth collected as described in Section 4-2.

	test set	real data AL	real data AH
recall	0.46	0.36	0.47
precision	0.47	0.47	0.44
F_1-score	0.47	0.41	0.45

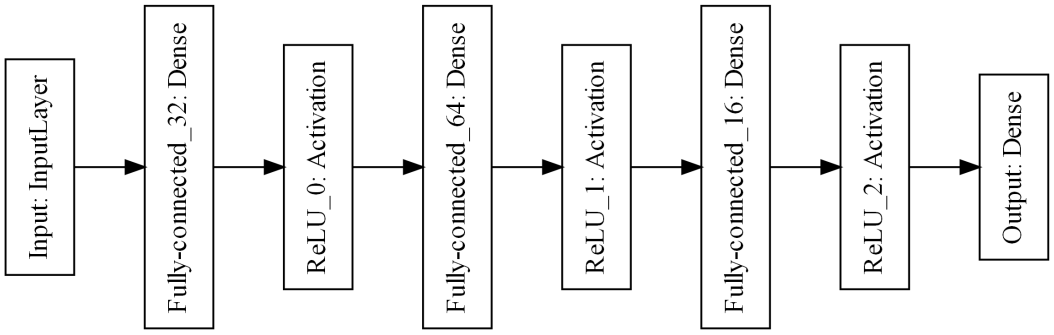


Figure 6-14: Architecture scheme for the final FCNN model for subseries with simple features.

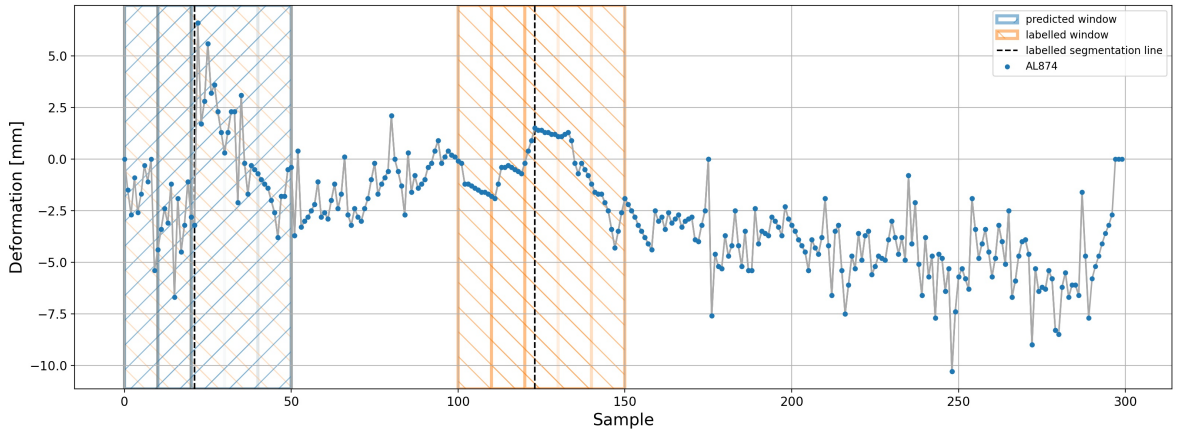


Figure 6-15: Output of the FCNN model with simple features on a real time-series AL874. The model produces a lot of false positives, i.e. predicts windows to contain a segmentation line when in fact there are no ground truth windows there. One of the segmentation lines was detected correctly (all windows containing this line are predicted) and the other one is missed completely. There are no false positives. Ground truth collected as described in Section 4-2.

Even though the results on the validation and test sets are similar for both FCNN approaches, the simple features model has shown much better performance on the real data than the

automatically generated features model. Apparently, the simple features model is able to generalize better as it does not catch that many details of a subseries as the automatically generated features. And while on the simulated dataset both models are performing fine, the real time-series are much less smooth than the simulation and most probably introduce a lot of noise into all the different automatically generated features. It is probable that if more real labelled data was available and used for training, the simple features would occur to be too simple and too few.

CNN

Compared to the multi-label classification in Section 6-2-1, the binary classification task in the subseries case is simpler and the input sizes are much smaller as well. Therefore, both the number of layers and the kernel sizes are smaller. In the experiments, smaller kernels, 7 and 5, perform better than the bigger ones, 11 and 9. Also smaller values of dropout (0.3 and 0.1) are more suitable than 0.5. A learning rate of 0.001 for Adam optimizer gives better results than e.g. 0.0001.

Out of many experiments with different settings of hyperparameters the best results are obtained for a model with two convolutional layers and two fully-connected layers. The convolutional layers have 32 filters with kernel 7 and 64 filters with kernel 5 accordingly. The fully-connected layers have 256 and 32 nodes. The learning rate is 0.005 and the optimizer is Adam. The exact architecture scheme is presented in Figure B-3 and a general scheme is shown in Figure 6-17. The rest of experiments is described in the Appendix C. Regarding the performance, for most of the experiments, the scores on the validation set all seem to be around 0.4 for both recall and precision.

The results of the final model on the test set and on the real InSAR data are given in Table 6-12. The results on the test set resemble the results on the validation set. The precision and recall are on a similar level. The real datasets however seem to cause more problems to the trained model considering high recall and low precision scores. An example of the model output on a real time-series is presented in Figure 6-18, where multiple blue regions indicate a lot of false positives.

Table 6-12: Results of the final CNN model with the subseries input on test set and two real datasets, AL and AH. Ground truth collected as described in Section 4-2.

	test set	real data AL	real data AH
recall	0.47	0.76	0.72
precision	0.46	0.25	0.23
F_1-score	0.47	0.38	0.35

In general, if a segmentation line lies close to an edge of a subseries, then the network cannot detect it correctly. Especially if the segmentation line is exactly at the edge. This is one of the disadvantages of the subseries approach. Also, sometimes windows are too short to yield meaningful information about potential changes in the time-series that would require segmentation.

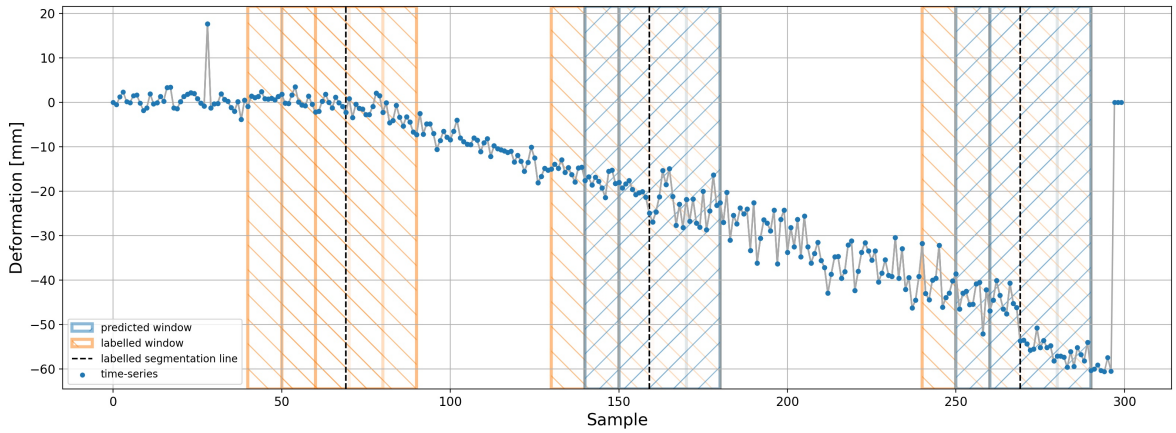


Figure 6-16: A simulated time-series with windows predicted to contain a segmentation line (blue, / pattern) and windows labelled to contain a segmentation line (orange, \ pattern). Areas where both pattern cross show where the prediction agrees with ground truth. The CNN model is able to detect two out of three segmentation lines. In the middle and right parts one more window for each line could be detected on the left side of the line, but the line is very on the edge of the 30 sample window and the model was not able to find it.

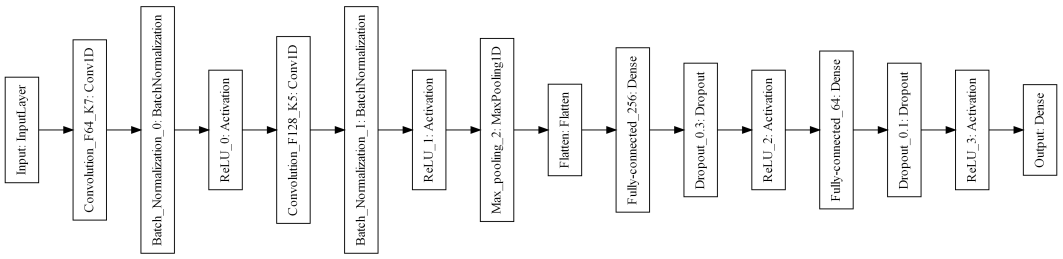


Figure 6-17: Architecture scheme for the final CNN model for subseries approach.

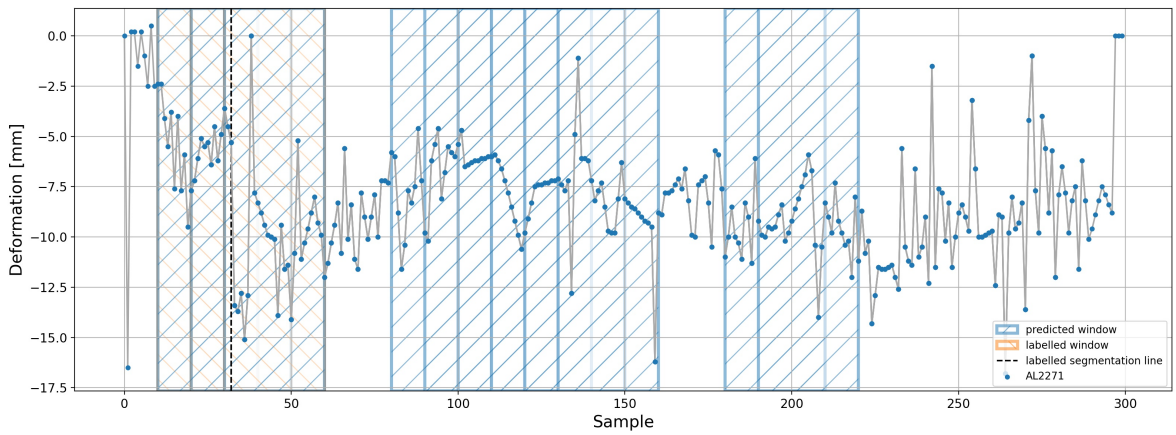


Figure 6-18: An example of the final subseries CNN model output on a real time-series (AL2271). At the beginning of the time-series, areas with predicted segmentation (blue, / pattern) overlap with the labelled areas (orange, \ pattern). In the middle and on the right hand side there are several false positive areas. Ground truth collected as described in Section 4-2.

The multi-label classification approach is more complicated than the subseries approach in terms of difficulty of a machine learning task. Even thought both CNN classifiers give similar results on the validation set, there is a significant difference in handling the real datasets. A simpler task (subseries) performs better than the multi-label classification in terms of the F_1-score, which is biased by a high recall score. However, overwhelming number of false positives causes that the results look unreliable and random. Therefore, qualitatively, the multi-label classification gives more interpretable results, due to higher precision. This may be caused by the fact that the real data is noisy and its behavior is less predictable than the simulated time-series. And therefore, cutting such time-series into smaller parts creates short series of data which are hard to interpret without broader context.

Conclusions

The best results among all the tested neural networks are obtained with the FCNN subseries model with simple features. Although, after qualitatively assessing the outputs of the networks on the real datasets, the CNN multi-label approach seems to provide the results with the best balance between the scores, readability, and interpretability. Interestingly, all of the final networks have very similar performance on the simulated test set but differ a lot when evaluated on the real datasets. However, the models are only trained using simulated data, if more labelled real time-series data was available so that the models could be additionally trained on these datasets, the models would exhibit different and most probably, improved behavior. While comparing the results of machine learning approaches with the results from Section 5-3 it is important to remember that the real time-series are all linearly interpolated as the networks cannot handle empty values in their inputs. Also the need of using the simulation and designing the network with a fixed input size forced the use of interpolation. In general, the machine learning approaches presented in this chapter do not outperform methods such as PLR, BFAST, or Bayesian Changepoint Detection. However, supervised learning requires a lot of reliable training data and considering that the current performance is obtained with simulation training only, the machine learning approach in general shows a lot of potential.

Chapter 7

Conclusions

The main objective of this thesis was to investigate *How can the machine learning approach be used to reduce the errors of types 1, 2, and 3 in InSAR postprocessing?* (see Section 1-1).

At first exploratory analysis of the considered dataset of InSAR time-series was performed. Main focus was on different aspects of non-homogeneity of the time-series, such as different trends, jumps, heteroscedasticity, or partial decorrelation. Additional contextual information was discussed, with the biggest attention given to spatial neighborhood of the points. Analysis of the correlation between neighboring time-series allowed to determine whether there is a potential unwrapping shift present between the time-series.

In this thesis a data-driven approach was taken and hence, collection of the labelled data was an important part. Therefore, a tool for time-series assessment was developed. A graphical interface was built and it allowed to record experts' knowledge that would be later on used in the training of machine learning models. Since the amount of collected data was not sufficient for machine learning purposes, a simulation was built that made it possible to generate labeled time-series resembling the original dataset of InSAR time-series.

Several algorithms based on literature were implemented and tested for unwrapping error detection as well as for segmentation tasks. The best results in the unwrapping error detection were obtained with Neighboring Samples and Moving Average Smoothing methods with $\sim$ 70% of recall and more than 80% precision. In the most of the methods the performance was better in the low points dataset, compared to high points, which is influenced by the higher variability and more noisy data in the high points. In the segmentation task, among several tested algorithms the offline Bayesian Changepoint Detection algorithm proved to give the best results. The need of interpolating the data in the time-series introduces distortion into the data and is one of the reasons behind mediocre results of segmentation in general.

Two different machine learning approaches to segmentation are developed in this thesis, namely multi-label classification and subseries binary classification. In multi-label classification, using only raw data as inputs, CNNs show better performance than FCNNs with 15% and 38% higher F_1-score compared to FCNNs, for AL and AH, respectively. However, the best results are obtained with a subseries binary classification FCNN with a set of 45 simple

features. In terms of F_1-score, it is better by 0.12 and 0.20 than FCNN with automatically generated features, for AL and AH, respectively. While it outperforms a multi-label CNNs, this method is less precise, as it only detects existence of segmentation lines within a wider window, which is larger than the tolerance margin in multi-label classification networks.

The results of the networks were not better than the non machine learning methods tested in this thesis. The main reasons behind this are: using simulated data for learning due to lack of enough annotated data, no objective ground truth in the real labelled data as the decision whether a segmentation line should be placed at a given epoch can be different among different labelling people, and the interpolation of the real dataset.

In conclusion, experiments done in this thesis are a step towards automatic assessment of the InSAR time-series based on the experts' knowledge and input. This is a data-driven approach and in such case a main advantage is no need for explicit providing of rules and logic, an expert could transform their experience and intuition into an input to an automatic system. On the other hand, the facts that there can be inconsistent decisions from different experts, the objective truth is not available in many cases, and the variability of the InSAR time-series all contribute to the difficulty of the posed task. Although the results in this thesis did not prove that the machine learning approach is able to outperform classical (non machine learning) methods, there is still potential in this approach and it might be worth exploring it further.

Contributions

Different types of InSAR errors of postprocessed time-series are simplified into three distinctive classes, namely unwrapping errors (single errors and shifts), partial temporal decorrelation, and incorrectly fitted models (see Section 1-1).

An alternative way of identifying unwrapping shifts by comparing neighboring points and analyzing correlation between them is proposed (see Figure 3-14).

Methodology for using supervised learning techniques in the InSAR postprocessed time-series data is designed and tested. A graphical tool is developed for the collection of experts' knowledge (see Section 4-2, Figure 4-1, and Figure 4-2).

A simulation for InSAR time-series is designed and developed (see Section 4-3 and Appendix A).

7-1 Future work and recommendations

Next step after successful segmentation of the time-series would be to fit models, e.g. using the library of canonical functions, to the obtained segments. The time-series would then look like the original processed InSAR time-series, which have the models fitted, but divided into separate segments if the time-series nature indicated that there is no consistent behavior throughout the whole time range. That is a final goal of this project, since that would improve the understanding of the presented data to the end-user which happen to not be experts in the field.

Regarding the performance and the results, main recommendation would be to collect more labelled data, i.e. time-series assessed by the experts. That would allow to apply transfer learning or retrain and fine-tune the model on the real data, which would improve the performance of a neural network. Also an expert could additionally give labels on the output of the model in order to correct the model's outputs. Then, the model could be further trained on this additional input and learn even better.

There are numerous other machine learning approaches that could potentially prove useful in the considered task. Apart from other methods using neural networks, such as e.g. autoencoders, other machine learning algorithms, such as ones based on decision trees are worth investigating.

Moreover, another method of expert knowledge incorporation could be investigated. In this thesis a data-driven approach was taken, however, a more domain-specific, hand-crafted method may be more suited for the considered task.

Appendix A

Data simulation

Two versions of the data simulation have been prepared in this thesis. Fist, simple simulation and second, extended simulation where the time-series with different segments are generated. In the simple simulation there are five kinds of models available and they can be combined together to get a more complicated model. The models are based on the library of canonical functions, see Section 3-5-2 and equations (3-9). In the simulation the following models are used, where $\boldsymbol{x}$ is the argument of the function, i.e. list of samples or time steps, and $\boldsymbol{y_0}$ is the initial vector:

- linear (M_1 in (3-9)):

$$\boldsymbol{y} = a\boldsymbol{x} + \boldsymbol{y_0}, \tag{A-1}$$

  where a is the slope of the linear function;

- temperature-dependent (M_2 in (3-9)):

$$\boldsymbol{y} = \eta\Delta\boldsymbol{T} + \boldsymbol{y_0}, \tag{A-2}$$

  where η is the thermal expansion coefficient and $\Delta\boldsymbol{T}$ is the temperature difference with respect to the first sample;

- sinusoidal (M_3 in (3-9)):

$$\boldsymbol{y} = s \cdot \sin\left(2\pi\frac{\boldsymbol{x}}{T}\right) + c \cdot \cos\left(2\pi\frac{\boldsymbol{x}}{T}\right) + \boldsymbol{y}_0, \tag{A-3}$$

  where s is the coefficient for the sine component, c is the coefficient for the cosine component, and T is the period;

- exponential (M_4 in (3-9)):

$$\boldsymbol{y} = y_{st}(1 - \exp(-(\boldsymbol{x}/\beta)) \cdot \kappa + \boldsymbol{y_0} \tag{A-4}$$

  where κ is the exponential magnitude, β is the slope factor of the exponential magnitude, and y_{st} is the starting value;

- step (M_6 in (3-9)):

$$\boldsymbol{y} = \boldsymbol{y_0} + \sum_i \Delta_i \mathscr{H}(\boldsymbol{x} - \tau_i), \tag{A-5}$$

where Δ_i is the value of the step, $\mathscr{H}$ is the Heaviside function, τ_i is the sample (or time step) at which the step occurs, and $i \in [0, m]$, where m is the number of steps applied in the series.

The models are generated by giving the values for all the parameters in the expected model. After that, the simulated time-series are generated using the above models and given parameters. At the end, the single unwrapping errors are added at random samples. In practice, the simple simulation can be treated as the extended simulation with just one segment and hence, only the extended simulation is described in detail. Examples of series generated with basic simulation are shown in Figure A-1.

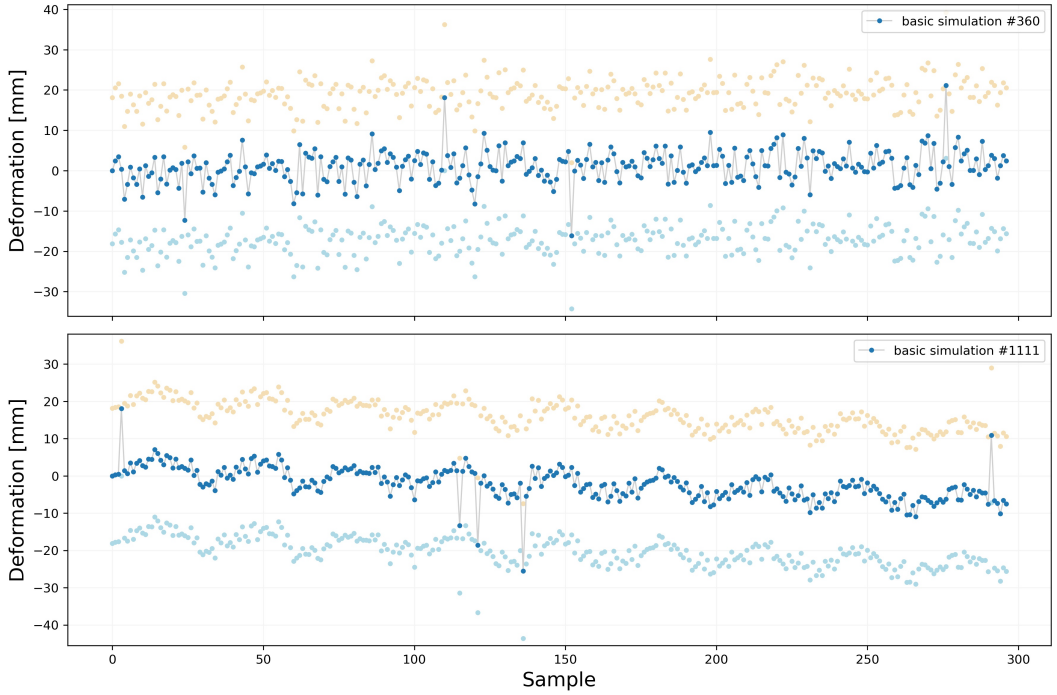


Figure A-1: Two examples of series from basic simulation.

At first, a number of segments is chosen. The number of segmentation lines is a random integer between 0 and 3, giving a maximum of 4 segments. The minimal length of a segment is set to 15 samples. A sample at which the segmentation line will be drawn is chosen randomly. Then, each next segmentation line is placed randomly, with the constraint that a segment length cannot be shorter than 15 samples. Figure A-2 shows the iterative process of choosing the sample for segmentation line. Blue horizontal lines represent available samples for segmentation line, black dot is a chosen sample. Grey crossed area shows where the next segmentation line cannot be placed.

With the determined number of segments, the first step is to choose a model for the time-series. There are three available models, namely linear ((A-1)), temperature-dependent ((A-2)), and

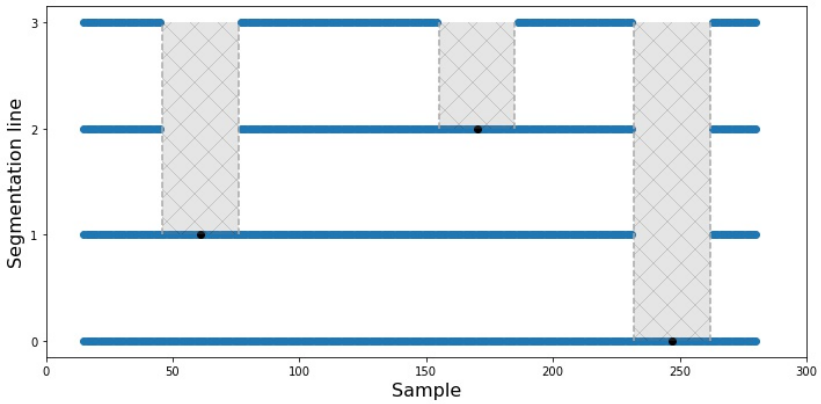


Figure A-2: Iterative process of choosing the segmentation lines. Horizontal lines represent available ranges for a respective segmentation line. Grey crossed area shows ranges that are excluded in the choice of the subsequent lines. In this case, segmentation lines will be at $[61, 170, 247]$.

exponential ((A-4)), but the final model can also be a sum of two or three of these models. The chance for choosing only one model is 40%, and for two or three, both 30%.

The next step is to choose starting parameters for the time-series model for the first segment. The starting value is always 0, just like in the original InSAR time-series dataset. The standard deviation is set to a random value in range $[1.0, 5.0)$. Then, depending on the model components, parameter values are randomly chosen from a uniform distribution in the ranges defined in Table A-1.

Table A-1: Parameter ranges and minimal changes values for the simulated time-series model components.

Model component	Parameter ranges	Minimal change
Linear	$a \in [-8.3, 0.7)$	1.0
Temperature-dependent	$c_t \in [-0.5, 0.5)$	–
Exponential	$\beta \in [1.0, 10.0)$	3.0
	$\kappa \in [-30, 30)$	1
all - standard deviation	$\sigma \in [1.0, 5.0)$	2.0
all - step	$\Delta \in [0.3, 0.7)$	–

The values are generated using the randomly created model for the length of the first segment, which might also be the only segment in the simulated time-series if the number of segmentation line is zero. At the end of the first segment, the changes that are going to be applied on the simulated time-series are randomly chosen. There are three types of changes, namely changing the standard deviation, adding a step, and changing the model completely. There is 60% chance that only one of these three changes is applied, 30% chance that two of the changes, and only 10% that all three effects come together.

Standard deviation change: If the standard deviation change is applied, the samples in the next segment are generated using a new standard deviation, the difference between

the previous and the new value of standard deviation has to be at least 2.0, in order to cause any easily visible changes.

Step change: The step change consists in adding a step, the coefficient is in range $[0.3, 0.7)$ and the sign $-1, 1$ of the step change is randomly chosen. This is multiplied by 18.1 to simulate a step change that is not an unwrapping shift.

Model change: If the model change is chosen, a new model is generated, taking the last point of the preceding segment as a boundary condition for the new model. Then, the parameters are chosen randomly but under the condition of minimal change values (see Table 4-1) to keep the changes discernible.

The process is repeated until all the segments are generated. At the very end, when the whole simulated time-series is ready, the single unwrapping errors are added at randomly chosen samples. The number of added unwrapping errors is between 0 and 10. Examples of series from extended simulation are shown in Figure A-3.

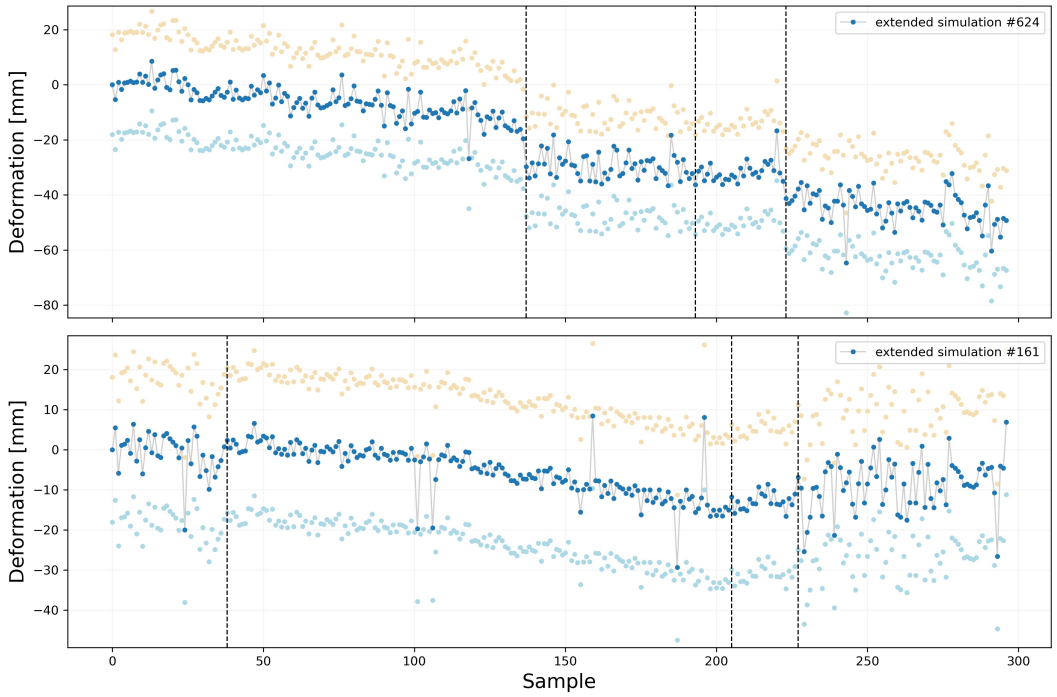


Figure A-3: Two examples of series from extended simulation.

Appendix B

Model details

B-1 CNN - raw input

Figure B-1 shows detailed architecture of the final CNN model for raw input approach.

B-2 FCNN - raw input

Figure B-2 shows detailed architecture of the final FCNN model for raw input approach.

B-3 CNN - subseries

Figure B-3 shows detailed architecture of the final CNN model for subseries approach.

B-4 FCN - subseries - automatically generated features

Figure B-4 shows detailed architecture of the final FCNN model for subseries approach with automatically generated features.

B-5 FCN - subseries - simple features

Figure B-5 shows detailed architecture of the final FCNN model for subseries approach with simple features.

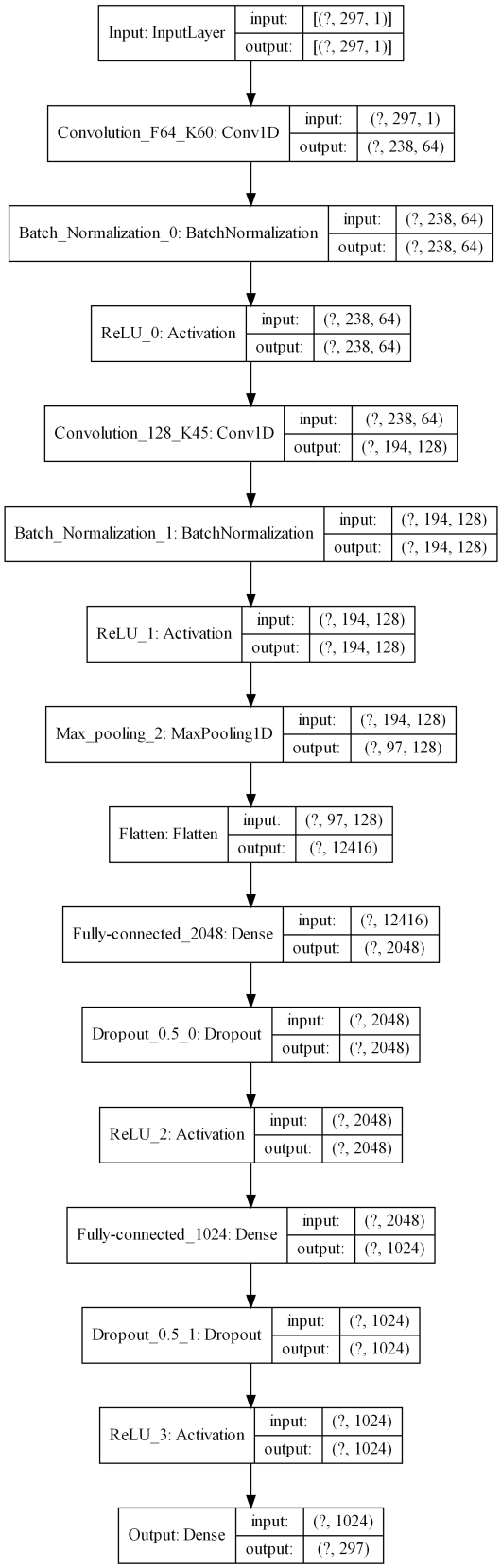


Figure B-1: Detailed architecture of the final CNN model for raw input.

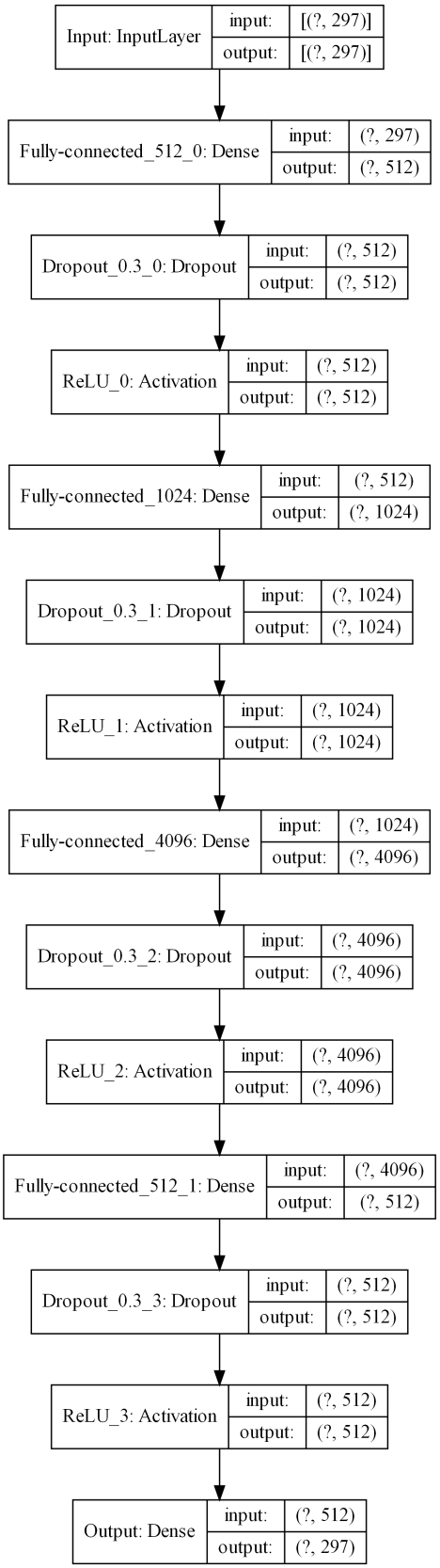


Figure B-2: Detailed architecture of the final FCNN model for raw input.

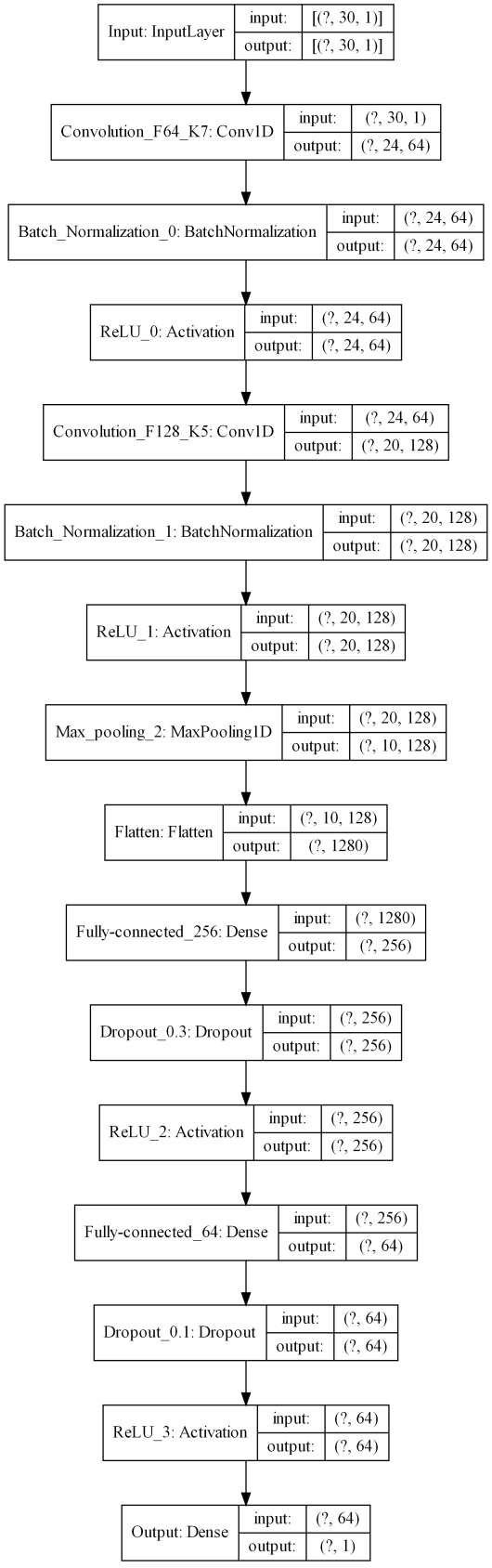


Figure B-3: Detailed architecture of the final CNN model for subseries.

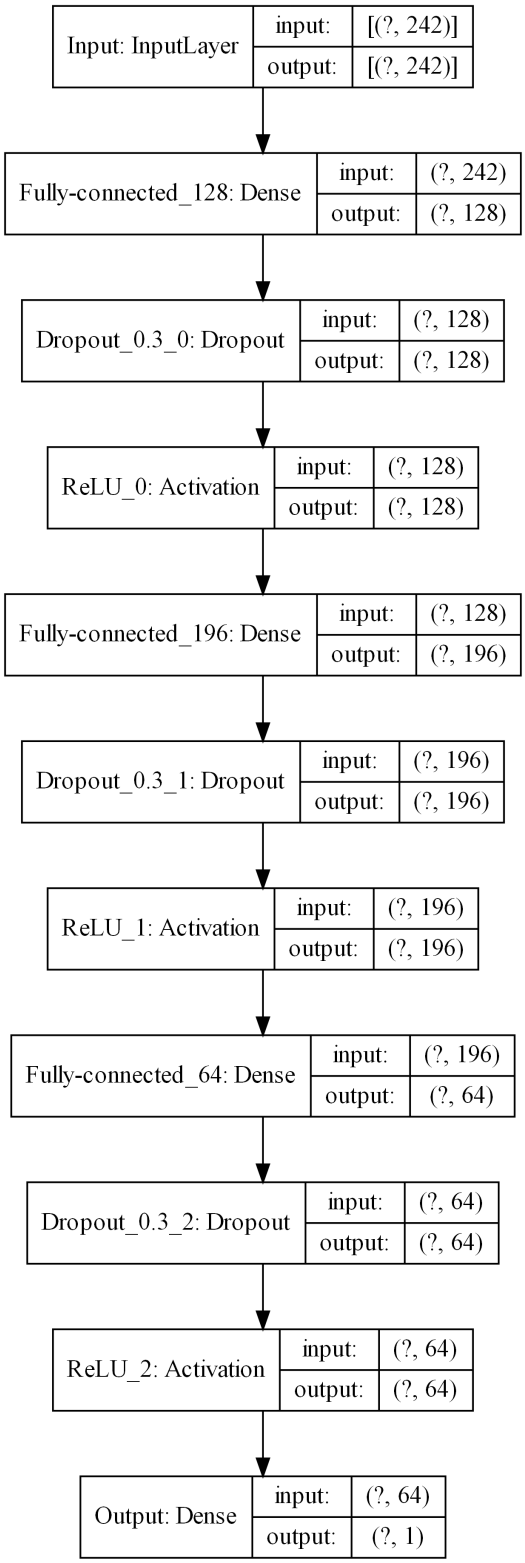


Figure B-4: Detailed architecture of the final FCNN model for subseries with automatically generated features.

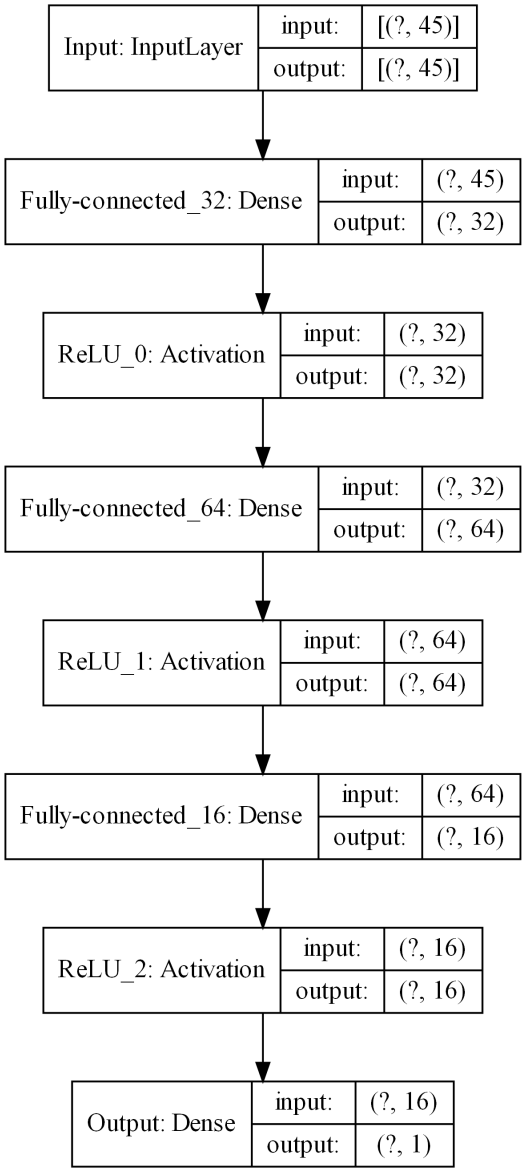


Figure B-5: Detailed architecture of the final FCNN model for subseries with simple features.

Appendix C

Experiments - settings and results

The appendix with the experiments and results is available in a separate PDF file upon request.

Bibliography

[1] R. F. Hanssen, *Radar Interferometry: Data Interpretation and Error Analysis*. Remote Sensing and Digital Image Processing, Springer Netherlands, 2001.

[2] L. Zhang, L. Zhang, and B. Du, "Deep learning for remote sensing data: A technical tutorial on the state of the art," *IEEE Geoscience and Remote Sensing Magazine*, vol. 4, pp. 22–40, June 2016.

[3] X. X. Zhu, D. Tuia, L. Mou, G. Xia, L. Zhang, F. Xu, and F. Fraundorfer, "Deep learning in remote sensing: A comprehensive review and list of resources," *IEEE Geoscience and Remote Sensing Magazine*, vol. 5, pp. 8–36, Dec 2017.

[4] A. Ferretti, A. Monti-Guarnieri, C. Prati, F. Rocca, and D. Massonet, *InSAR Principles - Guidelines for SAR Interferometry Processing and Interpretation*. ESA Publications, TM-19, 2007.

[5] H. Maitre, *Processing of Synthetic Aperture Radar (SAR) Images*. Wiley-IEEE Press, 1 ed., 2008.

[6] J. Penman, M. Baltuck, and C. Green, *Integrating remote-sensing and ground-based observations for estimation of emissions and removals of greenhouse gases in forests Methods and Guidance from the Global Forest Observations Initiative*. 01 2013.

[7] P. López-Dekker, "Slides for the course Geodesy and Natural Hazards CIE4609, TU Delft," 2018.

[8] "ESA: Sentinel Online - Technical Guide." https://sentinel.esa.int/web/sentinel/technical-guides/sentinel-1-sar/products-algorithms/level-1-algorithms/single-look-complex. Retrieved on 15.09.2018.

[9] F. Rocca, C. Prati, and A. Ferretti, "An overview of ERS-SAR interferometry," in *ERS Symposium on Space at the Service of Our Environment, 3 rd, Florence, Italy*, 1997.

[10] L. Chang, *Monitoring civil infrastructure using satellite radar interferometry*. PhD thesis, Delft University of Technology, 2015.

[11] B. Osmanoğlu, F. Sunar, S. Wdowinski, and E. Cabral-Cano, "Time series analysis of InSAR data: Methods and trends," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 115, pp. 90–102, 2016.

[12] https://mdacorporation.com/geospatial/international/products-services/insar/technology/permanent-scatterer-insar/.

[13] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*. OTexts.com, 2014.

[14] R. Adhikari and R. K. Agrawal, "An Introductory Study on Time Series Modeling and Forecasting," *CoRR*, vol. abs/1302.6613, 2013.

[15] R. B. Cleveland, W. S. Cleveland, J. E. McRae, and I. Terpenning, "Stl: A seasonal-trend decomposition procedure based on loess (with discussion)," *Journal of Official Statistics*, vol. 6, pp. 3–73, 1990.

[16] F. Rocca, "Modeling interferogram stacks for Sentinel-1," 01 2019.

[17] G. E. P. Box, G. M. Jenkins, and G. C. Reinsel, *Time series analysis: forecasting and control*. Wiley, 4 ed., 2008.

[18] "statsmodels: Stationarity and detrending (ADF/KPSS)." https://www.statsmodels.org/dev/examples/notebooks/generated/stationarity_detrending_adf_kpss.html. Retrieved on 20.06.2021.

[19] J. L. Bentley, "multidimensional binary search trees used for associative searching,"

[20] S. M. Omohundro, "Five balltree construction algorithms," tech. rep., International Computer Science Institute Berkeley, 1989.

[21] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pp. 226–231, AAAI Press, 1996.

[22] I. E. Özer, S. J. H. Rikkert, F. J. van Leijen, S. N. Jonkman, and R. F. Hanssen, "Sub-seasonal levee deformation observed using satellite radar interferometry to enhance flood protection," *Scientific Reports*, vol. 9, no. 1, p. 2646, 2019.

[23] L. Chang and R. F. Hanssen, "A probabilistic approach for insar time-series postprocessing," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 1, pp. 421–430, 2016.

[24] P. J. G. Teunissen, D. G. Simons, and C. C. J. M. Tiberius, *Probability and observation theory*. Delft Institute of Earth Observation and Space Systems (DEOS), Delft Univesity of Technology, The Netherlands, 2005.

[25] L. Snidaro, J. García, and J. Llinas, "Context-based Information Fusion: A survey and discussion," *Information Fusion*, vol. 25, pp. 16–31, Sept. 2015.

[26] L. Snidaro, I. Visentini, and K. Bryan, "Fusing uncertain knowledge and evidence for maritime situational awareness via Markov Logic Networks," *Information Fusion*, vol. 21, pp. 159–172, Jan. 2015.

[27] G. Battistello, M. Ulmke, F. Papi, M. Podt, and Y. Boers, "Assessment of vessel route information use in Bayesian non-linear filtering," in *2012 15th International Conference on Information Fusion*, pp. 447–454, July 2012.

[28] W. J-H and Y. Gao, "The aiding of MEMS INS/GPS integration using artificial intelligence for land vehicle navigation," *IAENG International Journal of Computer Science*, vol. 33, Jan. 2007.

[29] G. Battistello and M. Ulmke, "Exploitation of a priori information for tracking maritime intermittent data sources," in *14th International Conference on Information Fusion*, pp. 1–8, July 2011.

[30] L. Snidaro, R. Niu, G. L. Foresti, and P. K. Varshney, "Quality-Based Fusion of Multiple Video Sensors for Video Surveillance," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, pp. 1044–1051, Aug. 2007.

[31] S. Grimm, P. Hitzler, and A. Abecker, "Knowledge Representation and Ontologies," in *Semantic Web Services*, pp. 51–105, Springer, Berlin, Heidelberg, 2007.

[32] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2010.

[33] R. Babuška, "An overview of fuzzy modeling and model-based fuzzy control," in *Fuzzy Logic Control*, pp. 3–35, World Scientific Series in Robotics and Intelligent Systems - Vol. 23, 1999.

[34] "Internet Encyclopedia of Philosophy: Propositional logic." https://iep.utm.edu/prop-log/. Retrieved on 20.06.2021.

[35] A. Smirnov, T. Levashova, and N. Shilov, "Patterns for context-based knowledge fusion in decision support systems," *Information Fusion*, vol. 21, pp. 114–129, Jan. 2015.

[36] O. Linda, M. Manic, and T. R. McJunkin, "Anomaly detection for resilient control systems using fuzzy-neural data fusion engine," in *2011 4th International Symposium on Resilient Control Systems*, pp. 35–41, Aug. 2011.

[37] M. Gupta, J. Gao, C. C. Aggarwal, and J. Han, "Outlier detection for temporal data: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, pp. 2250–2267, Sep. 2014.

[38] H. Wu, "A survey of research on anomaly detection for time series," in *2016 13th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, pp. 426–431, 2016.

[39] S. Basu and M. Meckesheimer, "Automatic outlier detection for time series: an application to sensor data," *Knowledge and Information Systems*, vol. 11, no. 2, pp. 137–154, 2007.

[40] D. J. Hill and B. S. Minsker, "Anomaly detection in streaming environmental sensor data: A data-driven modeling approach," *Environmental Modelling Software*, vol. 25, no. 9, pp. 1014 – 1022, 2010.

[41] Y. Yu, Y. Zhu, S. Li, and D. Wan, "Time series outlier detection based on sliding window prediction," *Mathematical Problems in Engineering*, vol. 2014, 2014.

[42] O. Linda, M. Manic, and T. R. McJunkin, "Anomaly detection for resilient control systems using fuzzy-neural data fusion engine," in *2011 4th International Symposium on Resilient Control Systems*, pp. 35–41, 2011.

[43] F. Martínez, M. P. Frías, M. D. Pérez, and A. J. Rivera, "A methodology for applying k-nearest neighbor to time series forecasting," *Artificial Intelligence Review*, Nov 2017.

[44] A. Troncoso Lora, J. M. Riquelme Santos, J. C. Riquelme, A. Gómez Expósito, and J. L. Martínez Ramos, "Time-series prediction: Application to the short-term electric energy demand," in *Current Topics in Artificial Intelligence* (R. Conejo, M. Urretavizcaya, and J.-L. Pérez-de-la Cruz, eds.), Springer Berlin Heidelberg, 2004.

[45] A. G. D'yakonov, "2006/07 artificial neural network computational intelligence forecasting competition," 2007.

[46] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: Identifying density-based local outliers," *SIGMOD Rec.*, vol. 29, pp. 93–104, May 2000.

[47] S. Aminikhanghahi and D. J. Cook, "A survey of methods for time series change point detection," *Knowledge and Information Systems*, vol. 51, pp. 339–367, May 2017.

[48] P. Fearnhead, "Exact and efficient bayesian inference for multiple changepoint problems," *Statistics and Computing*, vol. 16, pp. 203–213, 2006.

[49] X. Xuan and K. P. Murphy, "Modeling changing dependency structure in multivariate time series," in *ICML '07*, 2007.

[50] R. P. Adams and D. J. C. MacKay, "Bayesian online changepoint detection," 2007.

[51] R. Turner, Y. Saatci, and C. E. Rasmussen, "Adaptive sequential Bayesian change point detection," in *Advances in Neural Information Processing Systems (NIPS): Temporal Segmentation Workshop*, 2009.

[52] D. Barry and J. A. Hartigan, "Product partition models for change point problems," *Ann. Statist.*, vol. 20, pp. 260–279, 03 1992.

[53] E. Keogh, S. Chu, D. Hart, and M. Pazzani, "Segmenting Time Series: A Survey and Novel Approach," *Data Mining in Time Series Databases*, vol. 57, 03 2003.

[54] Y. Hu, P. Guan, P. Zhan, Y. Ding, and X. Li, "A novel segmentation and representation approach for streaming time series," *IEEE Access*, 2018.

[55] K. Bleakley and J.-P. Vert, "The group fused lasso for multiple change-point detection," 2011.

[56] J. Verbesselt, R. Hyndman, G. Newnham, and D. Culvenor, "Detecting trend and seasonal changes in satellite image time series," *Remote Sensing of Environment*, vol. 114, no. 1, pp. 106 – 115, 2010.

[57] H.-J. Zhang and J. Huang, "A segmentation technology for multivariate contextual time series," *2017 IEEE 4th International Conference on Soft Computing Machine Intelligence (ISCMI)*, pp. 71–74, 2017.

[58] W.-H. Lee, J. Ortiz, B. Ko, and R. Lee, "Time series segmentation through automatic feature learning," 2018.

[59] Z. Ebrahimzadeh, M. Zheng, S. Karakas, and S. Kleinberg, "Deep learning for multi-scale changepoint detection in multivariate time series," 2019.

[60] J. Lin, L. Keogh, Eamonnband Wei, and S. Lonardi, "Experiencing sax: a novel symbolic representation of time series," *Data Mining and Knowledge Discovery*, vol. 15, pp. 107–144, Oct 2007.

[61] M. Lovric, M. Milanovic, and M. Stamenkovic, "Algorithmic methods for segmentation of time series: An overview," *Journal of Contemporary Economic and Business Issues (JCEBI)*, vol. 1, pp. 31–53, 01 2014.

[62] J. Kulick, "Bayesian changepoint detection." https://github.com/hildensia/bayesian_changepoint_detection, 2014.

[63] G. Rodríguez, "Lecture notes on generalized linear models - survival models." https://data.princeton.edu/wws509/notes/c7s1, 2007.

[64] R. Girshick, "Fast r-cnn," in *International Conference on Computer Vision (ICCV)*, 2015.

[65] J. Verbesselt, R. Hyndman, A. Zeileis, and D. Culvenor, "Phenological change detection while accounting for abrupt and gradual trends in satellite image time series," *Remote Sensing of Environment*, vol. 114, no. 12, pp. 2970 – 2980, 2010.

[66] "BFAST, Breaks For Additive Season and Trend - R package,." http://bfast.r-forge.r-project.org/. Retrieved on 15.11.2019.

[67] "Machine learning in geosciences and remote sensing," *Geoscience Frontiers*, vol. 7, no. 1, pp. 3 – 10, 2016. Special Issue: Progress of Machine Learning in Geosciences.

[68] D. HO TONG MINH, D. Ienco, R. Gaetano, N. Lalande, E. Ndikumana, F. Osman, and P. Maurel, "Deep recurrent neural networks for winter vegetation quality mapping via multitemporal sar sentinel-1," *IEEE Geoscience and Remote Sensing Letters*, vol. PP, pp. 1–5, 01 2018.

[69] B. Massinas, A. Doulamis, N. Doulamis, E. Protopapadakis, and D. Paradissis, "Deep convolutional neural networks for modeling patterns of spaceborne interferometric sar systems signals," 09 2017.

[70] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[71] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[72] S. Theodoridis and K. Koutroumbas, *Pattern Recognition, Fourth Edition*. USA: Academic Press, Inc., 4th ed., 2008.

[73] Å. Brekke, F. Vatsendvik, and F. Lindseth, "Multimodal 3d object detection from simulated pretraining," *CoRR*, vol. abs/1905.07754, 2019.

[74] M. B. Kursa and W. R. Rudnicki, "Feature selection with the Boruta package," *Journal of Statistical Software*, vol. 36, no. 11, pp. 1–13, 2010.

[75] "tsfresh: Python package documentation." https://tsfresh.readthedocs.io/en/stable/. Retrieved on 20.06.2021.