



# Data Platform Using Open-Source Tools to Support Geospatial Research

**Case: SAR Satellite Based Change Detection**

Janne Alatalo

Master's thesis

March 2023

Master's Degree Programme in Information Technology,  
Full Stack Software Development

**Alatalo, Janne**

**Data Platform Using Open-Source Tools to Support Geospatial Research, Case: SAR Satellite Based Change Detection**

Jyväskylä: Jamk University of Applied Sciences, September 2020, 58 pages.

Master's Degree Programme in Information Technology. Master's thesis.

Permission for open access publication: Yes

Language of publication: English

**Abstract**

Many national and global institutions share open data for anyone to use. One such institution is the European Space Agency (ESA) that shares the images from their satellites as open data. The ESA satellite data was utilized in Jamk University of Applied Sciences' Data for Utilization project where the research investigated the use of Sentinel-1 synthetic aperture radar satellite images for storm damage detection. The research required an easy access to the satellite images over the whole research life cycle. However, no existing solutions were found for this task. Therefore, a spatial data storage platform that would support the research needed be implemented. This thesis used the design science research process to implement the solution and communicate the implemented solution and gained knowledge to other engineers and researchers that are working with similar problems. The data storage platform was implemented using open-source projects such as PostGIS, GDAL, QGIS, and SNAP. The supported research was a success thanks to the data storage platform, and it produced a scientific research article, thus demonstrating and validating the success of the solution. A preprint version of the research article is attached to this thesis report.

**Keywords/tags (subjects)**

Data Platforms, Open Source, Satellite Images, Geospatial Data

**Miscellaneous (Confidential information)**

## Contents

<b>1</b>	<b>Introduction .....</b>	<b>2</b>
<b>2</b>	<b>Research Paradigm .....</b>	<b>4</b>
<b>3</b>	<b>Research Ethics Review.....</b>	<b>6</b>
<b>4</b>	<b>Result Reliability Considerations .....</b>	<b>6</b>
<b>5</b>	<b>Theoretical Basis.....</b>	<b>7</b>
5.1	Previous Research .....	7
5.2	Vector and Raster Data .....	8
5.3	Distributing Spatial Data .....	9
5.4	Spatial Database.....	10
5.5	PostgreSQL and PostGIS.....	11
5.6	GDAL Project .....	12
5.7	Geospatial Data Visualization with QGIS .....	13
5.8	Extract Transform Load .....	13
5.9	ESA Sentinel-1 Data .....	14
<b>6</b>	<b>Results.....</b>	<b>15</b>
6.1	Solution Implementation .....	15
6.1.1	Considerations .....	15
6.1.2	Simplified System Diagram .....	16
6.1.3	Open Data Sources .....	17
6.1.4	Hardware .....	18
6.1.5	Extract Transform Load.....	19
6.1.6	Data Visualization .....	21
6.2	Solution Demonstration and Validation.....	23
6.3	Gained Design Knowledge.....	24
6.3.1	Compressed vs. Uncompressed Rasters.....	25
6.3.2	Changing the Compression of PostGIS out-db Raster .....	27
6.3.3	Workaround for Poor raster2pgsql Performance .....	28
6.3.4	Utilizing the GDAL Command-Line Tools.....	29
<b>7</b>	<b>Conclusions .....</b>	<b>31</b>
	<b>References .....</b>	<b>32</b>
	<b>Appendices .....</b>	<b>38</b>
	Appendix 1. Submitted Research Article .....	38

**Figures**

Figure 1 Simplified System Diagram .....	17
Figure 2 Screenshot of the QGIS Plugin. ....	22
Figure 3 Results from the Query Benchmark.....	27

**Tables**

Table 1 Open data sources.....	17
Table 2 Compression Benchmark .....	25

**Listings**

Listing 1 Performance Test Query.....	26
Listing 2 Script for Compressing the Raster .....	28

# 1 Introduction

Geospatial data, or often just spatial data, is commonly occurring datatype that has a geospatial location attached to it (Zhang & Yi, 2010). This type of data is often collected when observing real world events that happen in nature. A good example of geospatial data is weather data. Weather is a very localized phenomenon; therefore, the weather data always includes the location of the weather station that collected the data. The station location is the main feature that is used when querying the data later from the data storage, thus the data storage system and other tooling that handles the data must be optimized to support this feature.

This thesis report introduces open-source projects that can be used to handle geospatial data. More specifically, the thesis report focuses on a use case where the open-source geospatial tooling was used to handle data from synthetic aperture radar (SAR) satellites. The data was used in a study where an improved methodology of generating difference images for SAR satellite-based change detection classifiers was developed. The change detection classifiers are used to find changes between two satellite images that are captured at different times. The information about the change locations can be used in different ways. For example, it can be used to conduct rescue operations after a natural disaster by directing help to most damaged areas, or it can be used to monitor glacier melting (Sood et al., 2021; Sublime & Kalinicheva, 2019).

The improved methodology of generating difference images for change detection classifiers is documented in a preprint of a research article that is attached as the Appendix 1 to this thesis report. The article was submitted to IEEE Transactions on Geoscience and Remote Sensing journal that covers satellite based remote sensing topics. The article was part of a larger project where the use of SAR satellite images for forest damage detection was studied. This thesis report extends and provides more context to the article by describing the data storage system, processes that were used to handle the data, and the original assignment for the project in more detail than what can be possible in the constraints of a peer reviewed journal article.

This research was conducted in the “Data for utilization” project (Jamk University of Applied Sciences, 2021).

## 2 Research Paradigm

Design science research paradigm was used to conduct this research. Design science is a widely used research paradigm in the field of information systems and other engineering disciplines where it is used to create design knowledge from solving practical problems (Engström et al., 2020). The paradigm is a good fit for this research, since the project had a well-defined practical problem that needed to be solved with a real-world implementation. The project created knowledge during the planning, development, and validation phases that other developers and researchers can use to solve other similar practical problems, thus it is important to report the gained knowledge. This thesis uses the design science research process that is introduced by Peffers et al. (2006) where the research process consists of six distinct steps which are: problem identification and motivation, objectives of a solution, design and development, demonstration, evaluation, and communication. The following paragraphs include more information about each of the steps.

### **Problem Identification and Motivation**

The data for utilization project in Jamk University of Applied Sciences had a subproject where we studied the potential use case of applying European Space Agency Sentinel-1 SAR images to storm damage detection in forested areas. An automatic damage detection system could alarm forest owners that a storm has damaged their property, thus speeding up the cleanup and insurance claims. The satellite images are open data and available for download from Copernicus Open Access Hub (European commission, n.d.). The success of the project relied on easy access to the satellite images. However, the project did not have unlimited resources of developing a data storage system from scratch and no existing solution for the problem was found, therefore we needed to use existing open-source projects to implement a data storage, handling, and access system for the project.

### **Objectives of a Solution**

The solution must support research tasks in the different parts of its life cycle. At the beginning of the research project, the solution must support explorative data analysis where the tooling is used to store and query only few satellite images. However, the solution must also be scalable to the final task of creating a large dataset that was used to train a neural network. The dataset creation

phase required that a large number of images are easily accessible using the image properties, such as the acquisition date, as the premise for the access. Moreover, the developed solution had to support storing other sources of spatial data in addition of the SAR satellite images. Both, the exploratory phase and the dataset creation phase also required that the different sources of data are easily spatially joinable among each other.

### **Design and Development**

The theoretical basis for the solution was created by researching existing open-source projects that can be used for spatial data handling and storing tasks. In addition, existing publications in the topic were also studied. The suitable open-source tools were then used to implement the data storage and handling platform that was used in the project.

### **Demonstration**

The success of the developed storage and handling platform was demonstrated by supporting a research project that used the spatial data that was stored in the platform. The research in question was successful and produced a scientific article that is attached to this thesis report as an appendix.

### **Evaluation**

The ease of data access and performance were important for the research project. Therefore, these properties were evaluated during the development of the data storage and handling platform, and during the implementation of the supported research project. Furthermore, the overall success of the data storage and handling platform was validated by the success of the supported research project.

### **Communication**

The created design knowledge is communicated in this thesis report.

### 3 Research Ethics Review

This research follows the ethical principles of Jamk University of Applied Sciences (Jamk University of Applied Sciences, 2018). All the data that was used to implement the research is open data that is available for anybody to download free from the internet. Links to the data download portals are available in this report. The data does not include any sensitive information that would need special handling or ethical considerations. Although, the satellite images image the whole planet continuously, the resolution of the Sentinel-1 SAR images is not high enough to provide any identifiable information about small objects such as houses or other private properties. Only large terrain features such as lakes, swamps, fields, forests, and in some cases roads, are identifiable from the SAR images. The research is carried out with the best of author's ability to follow good scientific practices. The reproducibility of the research is considered. The results that are described in the article are documented in detail and the source code that includes the developed neural network model is shared openly on internet with a permissive MIT license <https://github.com/janne-alatalo/sar-change-detection>. Furthermore, the dataset is also shared openly in the Fairdata.fi service <https://doi.org/10.23729/7b22c271-5e25-40fe-aa6a-f5d0330b1872>. The openly shared code and dataset allows the repeatability of the results that are documented in the article. None of the authors of the article have conflict of interest with the work. The funding of the project is declared in the article, and the funding entity did not have influence over the research results that are reported in the article or this report.

### 4 Result Reliability Considerations

This thesis consists of two parts, this report and the article. This report communicates the acquired design knowledge that was created from the data platform development. The results are produced with the best of the author's ability to be reliable. The overall reliability of the reported solution was validated in a real-world use case where it was used to support scientific research that produced a scientific article, thus confirming that the solution works. However, some aspects of the solution, such as developer experience of accessing the data, were validated by using qualitative method that can be subjective. The SQL query language data access method was considered to offer an excellent developer experience, although many engineers might disagree with the opinion. Furthermore, the solution that is introduced in this thesis is not the only possible way of solving the described problem, or it might not be the most efficient solution. Nevertheless, this



does not mean that the result reliability is not good. The research result of the used research methodology is the design knowledge that is gained from the research. The results are communicated in this thesis report as they are. It is the readers responsibility of evaluating if the knowledge is usable in their own similar research problems.

The reliability of the results in the research article are also produced with the best of the author's ability to be reliable. The implementation of the experiment was a complex task where errors can always be present. However, special care was put in to verifying the code and data that was used in producing the results. Moreover, since the code and data is openly shared, anybody that questions the reliability of the results can themselves use the code and data to repeat the experiment.

## **5 Theoretical Basis**

### **5.1 Previous Research**

Steiniger and Hunter (2012) did related research where they listed existing open-source projects that can be used to build a spatial data infrastructure. The paper defines spatial data infrastructure to include seven categories, which are: desktop GIS, spatial database management systems, web map server, server GIS, web GIS clients, mobile GIS and GIS libraries and extensions. The paper concluded that open-source projects exist in all categories and many open-source solutions even compete with proprietary software. The same authors have published a second paper expanding the review to include research, development, and teaching use cases (Steiniger & Hunter, 2013). More recent look to the different open-source projects is given in a paper by Brovelli et al., (2017), where they list the tools by different use cases. Furthermore, in similar topic is the paper by Swain et al. (2015) where the authors surveyed published articles about water resource and earth science web applications and report what open-source projects are used to implement the applications. The paper verifies that open-source tooling is commonly used to support spatial research, however there is a clear lack of case studies that focus on reporting how the open-source projects can be used fulfill this task. The only paper in this category, that was found during the research phase, was authored by She et al. (2019) where they completed a case study of using open-source projects to extend the functionality of a geoportal.

This review of the existing research is not claimed to be complete. However, it shows that there is a clear lack of case studies that focus on how open-source project can be used to support spatial research or the studies are not titled as such to be easily discoverable. This thesis attempts to correct the lack of research in this category.

## 5.2 Vector and Raster Data

Spatial data comes in two flavors. Raster data is easy to understand as an image where the data is stored in rows and columns of a table where each cell of the table corresponds to a pixel of the image analogy. The georeferencing of the raster is done by knowing the geolocation of one corner of the raster (which is often the left upper corner), and the geographical size of one pixel (McInerney & Kempeneers, 2015). A raster can have multiple bands that are similar to color channels of a color (RGB) image, therefore every pixel in a raster is associated with one or more values of some data type. The data type in a raster can be continuous, binary, complex, or categorical to name only a few, however the physical bit representation and support for the different data types is dependent on the raster file format (McInerney & Kempeneers, 2015). Although rasters are always rectangular, many raster formats have support for NODATA values, or some valid value is reserved for this task and documented in the file format, therefore a raster can also represent a non-rectangular area (Farkas, 2017).

An example of a spatial raster data is satellite image. Satellite images are georeferenced so that the data users know what location the satellite has captured in the image. A second example of a raster data is interpolated weather information. Weather is measured at weather stations that are scattered around the country. However, many applications require to know what the weather is at some exact location that is not close to any weather stations. The scattered weather data can be transformed to continuous coverage of the whole area using interpolation algorithms (Aalto et al., 2016). Using a raster format to store this type data is a good choice.

The other flavor of spatial data is vector data. Vector data consists of geographical objects that are represented with a set of coordinates. A vector feature consists of a geometry – that can be a point, line, polygon, or some more complex geometry that is composed from multiple entities of these basic geometry types – and any number of attributes that are associated with the geometry (Farkas, 2017). The attributes can be used to store additional information about the geometry. For

example, the maximum depth of the lake and the lake name could be directly associated with the geometry that defines the lake banks. Another example of vector data could be GPS based vehicle tracking, where the tracking device produces continuous stream of location information with some time interval. This data can be represented as vector data using the point geometry type where the position of the point is the GPS location, and the measurement time is stored as an associated attribute.

The two spatial data types are inherently different. Technically it is possible to convert one data type to another, however there are multiple problems with the conversion, thus it is not advised (Congalton, 1997). Therefore, if the project needs support for both types of spatial data, it is important to verify that the tooling supports it. The SAR satellite images that were used in this study are raster data. However, the project also used vector data such as the topographical database of Finland. Furthermore, the vector and raster data needed to be spatially joined with each other, thus the tooling needed to be carefully selected to support both data types.

### **5.3 Distributing Spatial Data**

Spatial data always includes a location. The location information can be used to join spatial data to other spatial data, thus enriching it. Many government agencies create useful open spatial data that can be used for enriching purposes. Examples of such agencies in Finland are Finnish Meteorological Institute, National Land Survey of Finland, and Finnish Forest Centre, to name only a few. Joining data from that many places would be difficult without well-established standards for sharing such data.

National Land Survey of Finland provides a download portal for users to download the open data. They use data formats such as ESRI Shapefile, GeoTIFF, JPEG 2000, GML, and GeoPackage (National Land Survey of Finland, n.d.-a). From these formats, the GeoPackage format is a newer spatial data format that defines a relational database-based standard for storing both vector and raster spatial data in a SQLite database with SQL-based access interface for the data (Pons & Masó, 2016). The GML format is a XML based format that is specially suitable for sharing vector data since it is modeled after the OpenGIS Simple data model, therefore generalizing well for transformations between different formats (Zhu et al., 2011). From the raster data formats, both

GeoTIFF and JPEG 2000 are based on image formats that are extended to store geospatial reference metadata and they both support both lossless and lossy compression algorithms (GDAL/OGR contributors, 2023a; Gerlek & Fleagle, 2007). The ESRI shapefile format is possibly the most widely supported spatial data format, however it is old standard and includes multiple limitations that make the usage of the format unadvisable in modern systems (ESRI, 1998; Jachym Cepicky, 2017). Moreover, some spatial data is also distributed using the netCDF format, which is a generic data format for storing multi-dimensional array data, therefore it can work as a raster storage format (Rew & Davis, 1990). However, the netCDF does not have a universal standard for storing the georeferenced information, thus working with spatial data that is stored in netCDF format can be difficult (GDAL/OGR contributors, 2023b).

In addition to different data formats, many spatial web service standards exist, such as WMS, WMTS, WCS, and WFS (Varol & Şanlıoğlu, 2017). These standards allow the client to directly connect to the server and query the data over the network without requiring the user to download the data as a file. The client can use properties, such as spatial extent to query only part of the data in the server, thus requiring less bandwidth and processing resources. The WMS, WMTS, and WCS standards are for raster data, and the WFS standard is for vector data.

## 5.4 Spatial Database

Spatial databases are a specific type of database that are built to store and query spatial data. All of the popular relational databases, including MySQL, MS SQL, IBM Db2, Oracle Database, SQLite and PostgreSQL, have some sort of support of storing and querying spatial data (Piórkowski, 2011). There are two standards for spatial data support in relational databases. The older of the standards is OpenGIS Simple Features Specification that was first introduced in 1999, and although the newer standard, SQL/MM Spatial, was introduced in the same year it is heavily influenced by the older standard and many of the features are backwards compatible between them (Stolze, 2003).

The standards define a class model with new spatial data geometry types and their associated functions that can be used to handle spatial features in the database. The geometry types that the SQL/MM Spatial standard defines can be used to add points, polygons, lines, and collections of those types to the database tables. The functions can be used with the geometry types to convert

them to other formats, extract properties from them, compare them among each other, or create new geometries from them (Stolze, 2003).

The comparison functions enable spatial joins between the tables where the spatial positions of the features are used as the join relation. A practical example of this could be a database where there are two tables. The first table includes the borders of all municipalities in Finland as polygons, and the second table includes the polygons of all lakes in Finland. The two tables could be spatially joined in a query using `ST_Contains` function as the join relation clause. The `ST_Contains` function returns checks if one geometry is completely inside other geometry. This would result in a joined table where one could analyze the distribution of lakes based on the municipality borders, therefore a query such as “What municipality has the largest number of lakes” would be trivial to implement. Furthermore, one could use the property extraction functions of the standard, such as the `ST_Area` function, and write queries like “What is the average size of the lake in each municipality”.

The SQL/MM Spatial standard is missing types and functions for handling raster data, however storing raster data to a relational database is not unheard of. From the more popular relational databases, the Oracle database (Oracle Spatial GeoRaster) and PostgreSQL/PostGIS include raster support with functionality for in-database raster analysis and processing (Obe & Hsu, 2021; Xie et al., 2013). However, only PostgreSQL/PostGIS is open-source project whereas Oracle database is proprietary product. Since we needed to support both raster and vector data in this use-case, and PostgreSQL/PostGIS was only open-source database providing support for both spatial data types, the selection of database technology was easy. The following chapter introduces the PostgreSQL/PostGIS database in more depth.

## 5.5 PostgreSQL and PostGIS

The open-source PostgreSQL database with the open-source PostGIS plugin is one of the most capable spatial database available even when compared to proprietary spatial database products; it implements the SQL/MM Spatial standard better than many other products, and it also provides support for spatial raster data (Obe & Hsu, 2021). PostgreSQL is a normal relational database that can be extended with the PostGIS plugin to support the spatial data types and functions that are

defined by the SQL/MM Spatial standard with addition of similar data types and functions for raster data. Both projects are under permissive licenses, with PostgreSQL being licensed under a custom PostgreSQL license and PostGIS being under the more common GPL v2 license (*PostGIS: License, 2012/2023; PostgreSQL: License, 2023*). Therefore, both projects can be used even in commercial setting, with restriction that the GPL v2 license can cause additional requirements if the PostGIS source code is modified and distributed. PostgreSQL/PostGIS combination is available for all three major operating systems (Linux, Windows, MacOS).

In addition of implementing the SQL/MM Spatial compliant functions and data types, PostGIS also defines many other additional functions that work with the spatial data (*Chapter 15. PostGIS Special Functions Index, n.d.*). This includes the additional support for raster data that is not defined in the SQL/MM Spatial standard. Some functions even allow mixing the data types, such as the `ST_Clip` function that can be used to clip a raster with a geometry.

PostGIS installation comes with the `raster2pgsql` tool for uploading the raster data to the database. There are two strategies when storing rasters to the database: in-db and out-db. The in-db storage strategy uploads the raster pixel data to the database table like one would expect when thinking about storing something to a database. However, when using the out-db strategy only the metadata of the raster is uploaded to the table and the raster is referenced with the raster file path. Therefore, when using the out-db strategy, the large raster file can be stored on a cheaper storage system, or even on cloud, while the database itself is stored on a fast SSD drive. With both raster storage strategies, it is recommended that the original raster is divided to smaller tiles that are stored as rows in the database table. The `raster2pgsql` command-line tool includes arguments to automate the raster tiling process.

## 5.6 GDAL Project

The GDAL project is an important open-source project for spatial data handling as many of the other spatial projects are built on top of it. It provides a library for handling raster and vector data that can be used from multiple different programming languages (including Python, C/C++, C#, Ruby), and a set of very useful command-line utility tools that are included in the standard installation for spatial data handling (Garrard, 2016). The library defines a unified data model for raster and vector data with abstraction layers for loading and saving them in different file formats using

drivers (Warmerdam, 2008). The driver abstraction for saving and loading data enables to easily extend the GDAL project to support any new spatial data formats in the future. Both, the GDAL library and the command-line utility tools, were extensively used during the project. The Results section provides many examples of GDAL usage.

## 5.7 Geospatial Data Visualization with QGIS

An obvious way of visualizing geospatial data is to draw the data on a map layer. The open-source QGIS project allows to do that. It is an application that provides a graphical user interface to visualize, edit, and analyze spatial data. The project is built on top of the Qt project – which is a programming framework that is used to create cross platform graphical user interfaces, thus enabling QGIS to support all major operating systems (Linux, Windows, and macOS); and GDAL project – that enables support for the large variety of different spatial data formats that the GDAL project supports (Baghdadi et al., 2018).

Thanks to GDAL, QGIS supports over 75 vector formats; over 130 raster formats; and many web service formats, such as WMS, WMTS, WFS, WCS, WPS, and CSW (GDAL/OGR contributors, 2023e, 2023d; Khan & Mohiuddin, 2018). QGIS integrates well with PostGIS by offering an excellent support for visualizing vector data tables, including an option of constructing custom SQL queries for visualization. Raster tables can be visualized too, however from the author's experiences during the project the raster data visualization support is not as good. User cannot create custom queries that return raster data, instead QGIS supports only visualization of full raster tables. Fortunately, such omissions in functionality can be corrected by extending QGIS using the plugin API that QGIS offers using Python or C++ programming languages (QGIS project, 2023). The project also maintains a public repository for existing plugins that users have created and shared <https://plugins.qgis.org>.

## 5.8 Extract Transform Load

Extract transform load (ETL) is an industry standard term used to describe a system that downloads data from somewhere (extract) and uploads it to a centralized storage (load). However, often just downloading and uploading the data is not enough, instead the data needs to be cleaned,

validated, processed, or otherwise changed to different format, thus a step is needed between extracting and loading (transform). There is no single correct way of implementing an ETL system, the system design depends on many aspects, such as business needs, compliance and security requirements to name only a few (Kimball & Caserta, 2004).

Although there is no one solution to fit all, there is often some common design aspects between the ETL systems. The temporary storage location where the data is often stored when the data is extracted from the location, but not yet transformed is often called the staging area. It is also recommended to use the staging area between the transform and load steps. The staging area can be a directory in a file system, or it can be a temporary table in a relational database. Technically the staging area can be in a program memory, however it is recommended to use persistent storage for the staging area since it allows better recoverability, backups, and easier auditing (Kimball & Caserta, 2004).

The use of a staging area between the ETL steps works as a decoupler making the steps independent from each other. This way the whole ETL process does not have to be rerun when one step fails. Instead, the failed step can be restarted using the data from the staging area. Other advantage from the decoupling is that the different ETL steps can be implemented using different tools and programming languages. The data is often distributed in a well-known industry standard format as was discussed in the section 5.3. This allows the usage of existing open-source tools that often exist for handling and processing the data in these formats. Furthermore, the step-based ETL architecture that is decoupled with a staging area can work well with existing workflow management platforms such as Apache Airflow (Suleykin & Panfilov, 2019, 2020). The workflow management platforms provide a graphical user interface for handling, scheduling, and monitoring the ETL process.

## **5.9 ESA Sentinel-1 Data**

The European Space Agency (ESA) Sentinel-1 SAR satellite data was in central role in the project. As described in the section 2, the objective in the project where this solution was developed, was to research if radar satellite images are usable in the detection of storm damages to forested areas. ESA Sentinel-1 mission flies two identical satellites that are equipped with synthetic aperture radars that operate on C-band and provide fast revisit time that is measured in few days (six days



at the equator) (Fletcher & European Space Agency, 2012). The data generated by the satellites is open data and it is available for download to anyone from Copernicus Open Access Hub <https://scihub.copernicus.eu> (European commission, n.d.).

The satellites image the planet continuously thus creating massive amounts of data. The data is available for download in three different processing levels, where the level-0 data is the raw data captured by the satellite; level-1 data is generated from the raw level-0 data by applying different data processing algorithms to it; and the level-2 data is further processed from level-1 products, however the level-2 data is available only for ocean areas and include data about wind, waves, and currents (Fletcher & European Space Agency, 2012). The level-1 ground range detected (GRD) product is best suited for the storm damage detection from forested areas user case. The product is projected to earth ellipsoid model and georeferenced. In the process the radar signal phase information is lost, however for this application the backscatter intensity should suffice for the use case. The GRD products are dual polarization acquisitions, which means that the signal is transmitted from the satellite in one polarization and the backscatter signal is listened in two different polarizations. The GRD products are distributed using the SENTINEL-SAFE format, that stores the acquisition metadata in XML file and the SAR backscatter intensity data is stored as two different GeoTIFF files (one GeoTIFF for each polarization) (European Space Agency, n.d.).

Although the Sentinel-1 level-1 products already include some processing, it is advisable to further process these products before use. There are many processing steps that are advised to be applied for the GRD data (Filipponi, 2019). These preprocessing steps reduce the noise in the images, and correct imaging artifacts that are caused by the radar imaging technique. Many SAR image preprocessing algorithms are implemented in the Sentinel-1 Toolbox that is available in the Sentinel Application Platform (SNAP) software package (European Space Agency, 2022)

## **6 Results**

### **6.1 Solution Implementation**

#### **6.1.1 Considerations**

Very quickly at the beginning of the project it was decided to select PostgreSQL/PostGIS (now on referred only as PostGIS) as the base technology for the solution based on the knowledge that was

gained during the initial research for the solution. Other possible solutions were considered, such as not using a database at all, instead the solution would have been engineered from ground up using Python programming language and a spatial data handling library such as GDAL. However, that solution was quickly abandoned after the realization how large the processed satellite images are. It was known from the beginning of the project, that the solution would need to store a large number of satellite images, and the access to the images should be easy for the researchers. Therefore, if the solution would be built from ground up, a custom indexing system would have been needed for efficient access to the images based on image properties such as image acquisition time and location. Furthermore, the GDAL Python bindings are not very well documented, thus getting the project to usable level using this methodology would have required unavailable amount of programming resources to succeed.

Fortunately, during the initial research phase, PostGIS was considered as the base for the solution. As was discussed in section 5.5 of this thesis report, PostGIS provides an excellent support for both vector and raster spatial data. Using a relational database as the base for the solution, we gained the ability of using SQL query language for accessing the data. Furthermore, we had the ability of storing related metadata to the database from the satellite images, such as the acquisition time, the satellite that captured the image (Sentinel-1A or Sentinel-1B), the satellite orbit direction (North to South or South to North) etc. Since PostGIS is built on top of PostgreSQL, the normal relational database features are accessible, meaning that the metadata can be used as the premise for accessing different images in the SQL queries. The indexing of the data can be easily handled with PostgreSQL, thus providing efficient data access performance.

### **6.1.2 Simplified System Diagram**

Figure 1 Illustrates a simplified view of the implemented system. The data is extracted from the open data sources using ETL scripts that upload the transformed data to PostGIS database. The data can be visualized using QGIS. The following chapters introduce the system in more detail.

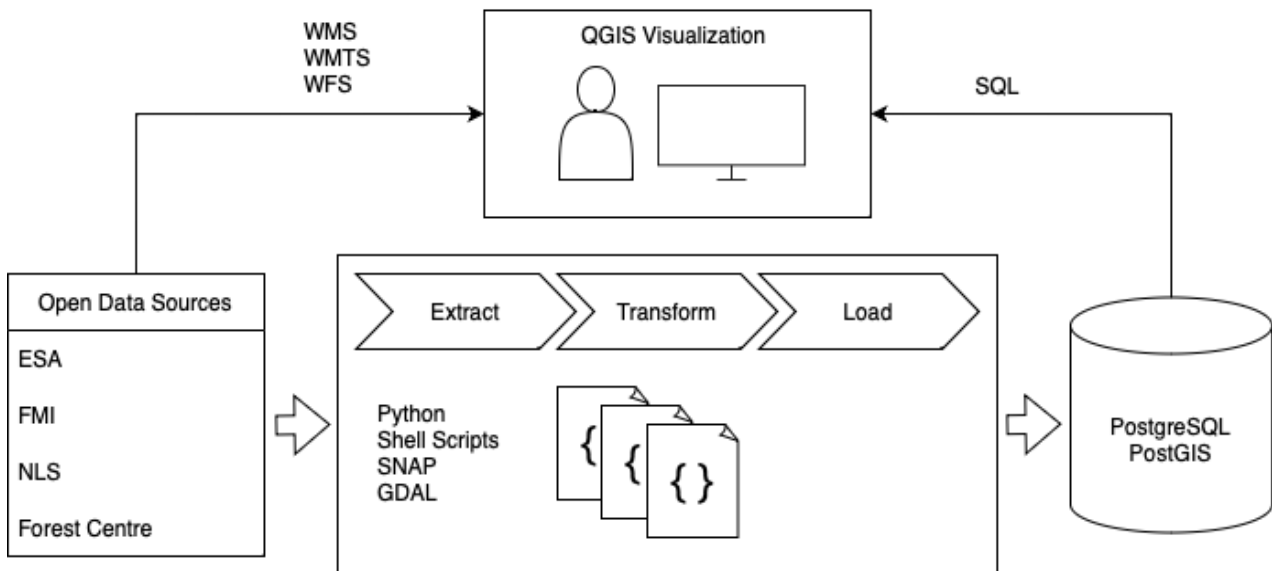


Figure 1 Simplified System Diagram

### 6.1.3 Open Data Sources

An excellent feature that was gained by using PostGIS is the ability of making spatial joins between the database tables. As was discussed in the section 5.3, there are many sources of open spatial data that can be used to enrich other spatial data using spatial joins. All the data from the different sources can be stored in one PostGIS database, thus allowing to perform the spatial joins in SQL queries. Table 1 lists the different open data sources that were saved in the PostGIS database.

Table 1 Open data sources

Provider	Description	License	Link
<b>European Space Agency</b>	Sentinel-1 radar satellite images	<a href="#">Other (European commission, n.d.)</a>	<a href="https://scihub.copernicus.eu/">https://scihub.copernicus.eu/</a>
<b>Finnish Meteorological Institute</b>	Daily weather data (precipitation, temperature, and snow)	<a href="#">CC BY 4.0</a>	<a href="https://en.ilmatieteenlaitos.fi/gridded-observations-on-aws-s3">https://en.ilmatieteenlaitos.fi/gridded-observations-on-aws-s3</a>

<b>National Land Survey of Finland</b>	Digital elevation model	<a href="#">CC BY 4.0</a>	<a href="https://asiointi.maanmittauslaitos.fi/karttapaikka/tiedostopalvelu?lang=en">https://asiointi.maanmittauslaitos.fi/karttapaikka/tiedostopalvelu?lang=en</a>
<b>National Land Survey of Finland</b>	Topographic Database	<a href="#">CC BY 4.0</a>	<a href="https://asiointi.maanmittauslaitos.fi/karttapaikka/tiedostopalvelu?lang=en">https://asiointi.maanmittauslaitos.fi/karttapaikka/tiedostopalvelu?lang=en</a>
<b>National Land Survey of Finland</b>	Division into administrative areas (defines the municipality borders etc.)	<a href="#">CC BY 4.0</a>	<a href="https://asiointi.maanmittauslaitos.fi/karttapaikka/tiedostopalvelu?lang=en">https://asiointi.maanmittauslaitos.fi/karttapaikka/tiedostopalvelu?lang=en</a>
<b>Finnish Forest Centre</b>	Forest mask	<a href="#">CC BY 4.0</a>	<a href="https://aineistot.metsaan.fi/avoimetsatieto/Metsamaski/Maakunta/">https://aineistot.metsaan.fi/avoimetsatieto/Metsamaski/Maakunta/</a>
<b>Finnish Forest Centre</b>	Forest usage data	<a href="#">CC BY 4.0</a>	<a href="https://aineistot.metsaan.fi/avoimetsatieto/Metsankayttoilmoitukset/Maakunta/">https://aineistot.metsaan.fi/avoimetsatieto/Metsankayttoilmoitukset/Maakunta/</a>

#### 6.1.4 Hardware

One important feature that enabled the PostGIS usage as the base technology for the solution was the support for out-db rasters. As was discussed in the section 5.5, PostGIS enables saving the raster data outside of the database. This means that the database files, that store the data that exists in the PostgreSQL tables, and the satellite images can be located on different storage devices. The satellite images are large. Thus, it is not likely cost-effective to store the images on fast and expensive devices when slower and cheaper storage media suffices. However, it is usually advisable to store the database files in as fast as possible storage media, since some queries can trigger a table scan which makes the database engine to go through the whole table searching matches for the query. In these cases, fast storage media can have major impact for the query performance. PostGIS allows the use of a solution that takes the best of both worlds. By using cheap disks that are based on the hard drive technology for the image storage and fast SSD disks for storing the database files, we get a cost-efficient solution that has a good performance. PostGIS saves the raster tile metadata to the PostgreSQL table such as the extent of the tile. The tile extents can be used to construct a spatial index that allows efficient execution of queries that use the tile location as a

condition. Based on the spatial index, the pixel data need to be retrieved from the slower storage media only for the tiles that match the location condition.

The first iteration of the developed solution was implemented on a consumer grade computer hardware. The computer had Intel Core i7-8700 CPU with 64GB of RAM. The database was stored on a fast Samsung 960 EVO 500GB NVME SSD, while the images were stored on two Western Digital WD Black 10TB 7200 rpm disks that were configured as a striped pool (raid-0) with ZFS. For a short time, the ZFS pool had a L2ARC cache that was implemented using Samsung 980 PRO 250GB NVME disk. The L2ARC should cache files that are accessed repeatably from the pool thus providing faster read performance when the files are served from the fast SSD disk instead of the hard drive. However, the performance improvement of the L2ARC cache was not verified because the SSD drive malfunctioned not long after the L2ARC enabled.

At the end of the project, we received proper server grade hardware where the developed solution was migrated to. The specs for the server were: Xeon Gold 6342 CPU, 512GB RAM, with 12x2.4TB Toshiba 12G SAS 10K, and 2xSeagate 12G SAS Enterprise Performance SSD disks. The other SSD disk was configured as a cache, same as in our consumer grade setup, and the other SSD was reserved for the database files.

### **6.1.5 Extract Transform Load**

The extract transform load (ETL) system was implemented using existing open-source tools or by developing scripts using Python programming language or shell scripting language. The satellite images were downloaded from the Copernicus Open Access hub using the open-source `sentinelstool` command-line tool and they were preprocessed using the SNAP toolkit (European Space Agency, 2022; Will, 2015/2023). The research that was conducted using the satellite images did not require continuous access to new images. Instead, the research was focused on a large storm event that happened on 20<sup>th</sup> of June 2021. Therefore, it was enough that the satellite images were downloaded, processed, and uploaded to PostGIS once. There was no need for scheduled uploads of fresh imagery. The same thing applied to other data sources. They were also uploaded to PostGIS only once from the same time period as the satellite images. For this reason, it was decided not to invest development resources on workflow management software such as Apache Airflow. The addition of such component would have complicated the implementation without gaining much

benefit from it. Instead, the ETL system was implemented as a set of manually executed scripts that run the ETL process. However, the addition of a workflow management software was not ruled out completely. The developed ETL scripts should work well with a workflow management software since the scripts are distinct decoupled steps that use the file system as a staging area, therefore the orchestration of the ETL process would be simple.

The satellite image preprocessing step requires considerable amount of computational resources. The first iteration of the implementation performed the satellite image preprocessing on the same server where the PostGIS database was running. As the number of satellite images grew, and we needed more processing power, the image preprocessing was moved to high-performance computation (HPC) server that we have available at Jamk University of Applied Sciences. The HPC server specifications were: 4 x Xeon Gold 6130 (Skylake) processors with hyperthreading turned off (64 CPU cores); 768GB DDR4 2666 MHz RAM; 4 x Tesla V100 32GB GPUs. The preprocessing steps were not GPU optimized, thus having the GPUs on the server did not help, however the large amount of RAM and the large number of cores allowed parallel processing of the images. The GNU parallel program was used to implement the parallel processing of the images (Tange, 2011).

The preprocessing step resulted in two GeoTIFF images per one acquisition (one GeoTIFF for both bands). It was decided to not use any compression for the images to maximize the query performance. Section 6.3.1 compares the compression vs. query performance tradeoff more closely. The average size of a single GeoTIFF image was 5.5GB with resolution around 72500 x 22000 pixels. The radar images were uploaded to PostGIS using the `raster2pgsql` command-line tool using 100x100 tile size which resulted to around 100,000 tiles per one GeoTIFF image to be inserted into the database. Tiles that included only NODATA values were not added to the database. Section 6.3.3 talks about the performance problems of the NODATA check and a proposed workaround for the problem. The database included about 1,500 unique satellite acquisitions, thus with two GeoTIFF files per acquisition the database included about 3,000 GeoTIFF files. The uncompressed GeoTIFF files required around 17TB of storage space and the uploaded images resulted to 310,810,967 tile rows in the database table.

### 6.1.6 Data Visualization

One requirement of the data platform was an easy access to the data. Since the satellite images are very much visual data, it is important that there is an easy way to visualize the images. As described in section 5.7, the open-source QGIS project is well suited for spatial data visualization. It can use the WFS, WMS, and WMTS services that many Finnish agencies provide for open data. The topographical maps that are provided by National Land Survey of Finland using the spatial web service standards were used extensively during the project (National Land Survey of Finland, n.d.-b). The satellite images are not very accurate. Therefore, it was a good practice to visualize the images on top of an accurate topographical map that displayed what geographical features were at the imaged location. The layer functionality in QGIS allowed easy comparison of the satellite image and the topographical map, where the layer containing the satellite image could be made transparent, or quickly turned on and off, thus revealing the underlining map.

Although QGIS includes an excellent support for PostGIS vector data, the support for raster data is not as good. The vector data can be visualized from custom SQL queries, however the raster data cannot. QGIS allows only visualization of full raster tables. Fortunately, QGIS has an excellent plugin API that can be used to extend QGIS functionality. Since the satellite image raster data was in such central position in this research, it was important that the developed data platform provided good visibility to the satellite data that was stored in the database. An existing solution for this problem was not found, thus a custom solution was developed during the project using the QGIS plugin API.

The plugin was developed using Python programming language. The implementation of the plugin was kept simple. The plugin was developed for this project's needs with minimal time resources; thus, the resulting implementation is not usable in other projects without further development and modifications to the code. Figure 2 illustrates the plugin functionality with a screenshot of QGIS where the plugin is used to visualize a satellite image. When the plugin cursor tool is selected the user can click anywhere on the map and the plugin queries the database for all satellite images that are acquired from that location. All acquisitions from the location are listed in the panel that is shown on the right side in the screenshot. The user can click through the different acquisitions and the layer that visualizes the radar image is updated to show the acquisition from the selected

time. Furthermore, the plugin has features to enhance the contrast of the satellite image using histogram equalization, and to cluster the different image acquisition angles to groups where the images from same angle are easy to browse. The user can also change the geographical extent of the satellite image to fetch smaller or larger images.

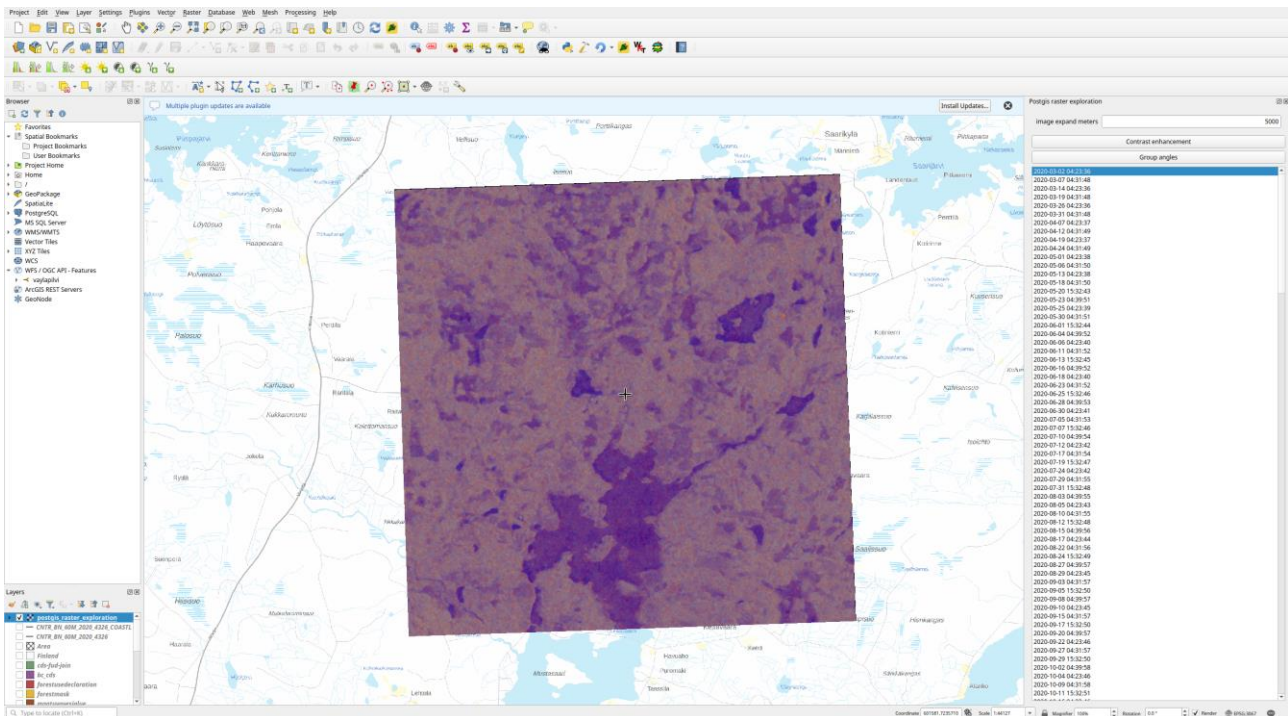


Figure 2 Screenshot of the QGIS Plugin.

The plugin was built around PostGIS database `ST_AsGDALRaster` function which is used in the main image query to make the PostGIS server to return the images from the location in GeoTIFF format. The query response is parsed to store the raster in GDAL /vsimem virtual file where it is added to QGIS layer using `QgisInterface.addRasterLayer` method. Although the plugin is not usable without modifications in other projects, it was decided to open source the implementation in case somebody finds it useful. The project was licensed under a permissive MIT license and it is available on the school's Gitlab server <https://gitlab.labranet.jamk.fi/tieto-tuottamaan/qgis-postgis-raster-exploration>.



## 6.2 Solution Demonstration and Validation

As was discussed in the section 4, the motivation to implement the system was to support research that used the SAR satellite images. The solution validity was confirmed by demonstrating that the developed system can fulfill the requirements that the research posed. The development of the solution extended the full timeline of the research. The development was an iterative process where new challenges were discovered in the research project and the data storage solution was improved to solve the problems.

The solution needed to support the research at all its phases. At the beginning the system was used for explorative data analysis. This included answering to questions, such as “how much the radar image backscatter intensity has changed inside a geometry X, between two images that are acquired at times T1 and T2”. Answering these types of explorative data analysis questions was trivial using the SQL query language and the raster and vector functions that the PostGIS database offers.

The satellite images were first visualized using QGIS, by adding the full image file as a layer to QGIS project. However, when the number of images grew it was impossible to add all images to QGIS for visualization since one full satellite image is over 5GB in size. A better solution was needed and for a while the images were visualized using custom Python scripts and matplotlib in a Jupyter notebook project where the images were queried from the PostGIS database using SQL. This solution did not support interactive exploration of different locations in the image, thus an even better solution was required. The problem was finally solved by developing the QGIS plugin that was described in the section 6.1.6. Since the solution was developed using Python programming language and it had full access to the data from the query, it was trivial to implement extra features to the plugin in addition of just image visualization, such as the histogram equalization and imaging angle clustering features. The histogram equalization feature answered to the research problem where we wanted to better utilize the full dynamic range of the full color space to better see the small changes in backscatter intensities. Likewise, the imaging angle clustering feature was in central role in our research, since it allowed us to confirm that the imaging angle influences to resulting radar image even when inspecting the images by eye.

The final challenge for the data storage system was the generation of a neural network training dataset. The dataset was used to train a deep learning model that was used to create improved difference images for change detection classifiers. The research article that is attached to this document as Appendix 1 details the developed methodology. It was an attempt of improving the accuracy of change detection classifiers enough, so that they are accurate enough to find the small changes in radar image backscatter that are caused by the storm damages. Because the methodology was based on neural network technology, it required a large training dataset to work. Fortunately, the training dataset was unsupervised, meaning that it did not need human labeling, thus the dataset could be made directly from the data that was stored in the PostGIS database.

The dataset creation script was implemented using Python. However, most of the work was done in PostGIS database with SQL. The database was queried for images from random locations in an area where the area was known to have storm damages in forests. Each execution of the query returned five images from successive overflies over the location. Furthermore, additional data was queried from the location, such as the weather conditions at the dates of the overflies, digital elevation model from the location, and imaging angle information (radar signal incidence angle and satellite orbit direction). The final sample was created from the data by adjusting the pixel alignment of the five images so that all pixels across all images were geospatially aligned. The same operation was done for the digital elevation model from the location, where the resolution of the model was also reduced to match that of the SAR images. The resulting dataset sample was encoded and stored in a TFRecord file, which is a file format specially made for neural network training data pipelines.

Overall, the research was successful as is evident from the research article that is attached to this report as Appendix 1. The research would not have been possible without the developed data storage system, thus validating the effectiveness of the solution. The data platform supported the research over the full life cycle by providing an easy access to the data with SQL query language, and by providing simple data visualization interface using the developed QGIS plugin.

### **6.3 Gained Design Knowledge**

As was discussed in section 2, the used design research methodology is focused in creating and sharing the accumulated design knowledge that was gained during the development of an artifact

that solves some real-world problem. Section 5 introduced the theoretical basis on which this research was built on and the sections 6.1 and 6.2 introduced the implementation and validation of the solution thus documenting the gained knowledge from these aspects of the research process. However, some of the gained knowledge was not fitting well to any of these sections, therefore they are introduced in this additional section.

### 6.3.1 Compressed vs. Uncompressed Rasters

GeoTIFF raster images support many different compression algorithms (GDAL/OGR contributors, 2023a). However, the format also supports storing the image without any compression. When PostGIS reads the pixel data from the GeoTIFF file, it needs to uncompress the image if it is compressed, thus reducing the query performance. The effect on storage requirements and query performance were experimented using a test SAR image. Three compression algorithms were tested: DEFLATE, LZW, and ZSTD. All three algorithms support the PREDICTOR parameter that can have one of three values: 1. No predictor, 2. Horizontal differencing, or 3. Floating point prediction (GDAL/OGR contributors, 2023a). All algorithms were tested with all three PREDICTOR values. The original file size was 6.12GB. Table 2 lists the results from the experiment.

Table 2 Compression Benchmark

<b>PREDICTOR</b>	<b>1</b>	<b>2</b>	<b>3</b>
<b>DEFLATE</b>	3.61GB	3.43GB	2.78GB
<b>LZW</b>	4.54GB	4.28GB	3.63GB
<b>ZSTD</b>	3.60GB	3.42GB	2.80GB

The best compression was achieved using ZSTD or DEFLATE algorithm with PREDICTOR parameter set to value 3. The resulting file size was under half the size of the original file ( $2.78 / 6.12 \approx 0.45$ ).

The compression results from both algorithms were so similar that they both were selected to the query benchmark. The LZW algorithm did not come even near the performance of the two other algorithms, therefore it was not selected for the query performance benchmark.

The query benchmark was conducted using the query that is shown in Listing 1 where the \$nbr\_points was doubled with each execution so that the benchmark was executed with following values: 10, 20, 40, 80, 160. The explain analyze print includes “Execution Time” information that was used as the execution time for the query.

```
explain analyze with points as (  
  select  
    (st_dumppoints(st_generatepoints(bounding_box, $nbr_points))).geom as point,  
    tiff_filename as filename  
  from radarimage_infos  
  inner join radarimage_metadata_filemaps on filename = xml_filename  
  where tiff_filename = 'radarimage.tif'  
)  
select  
  st_value(rast, point)  
from points  
inner join radarimages  
on points.filename = radarimages.filename and st_contains(radarimages.rast::geometry, point);
```

Listing 1 Performance Test Query

The query works by finding the geospatial extent of the radarimage.tif file from a metadata table in the common table expression clause. The image extent is used to generate \$nbr\_point of random points that are inside the image extent using the ST\_GeneratePoints PostGIS function. The main query finds the image tiles that contain the point locations using the ST\_Contains PostGIS function. Finally, the ST\_Value function is used to fetch the pixel value from the tile at the position of the point. By increasing the \$nbr\_point value, we can force the PostGIS access more tiles from the disk, thus making the query more demanding. All \$nbr\_point steps were executed 10 times for each compression method. Figure 3 illustrates the results from the benchmark. The plot includes error bars, however the timings between the 10 different executions were so similar that the error bars are not clearly visible in the figure. All results are linear when increasing the \$nbr\_point value. Uncompressed image access is clearly more performant when compared to compressed file data access. From the two compression algorithms ZSTD is clearly better performing.

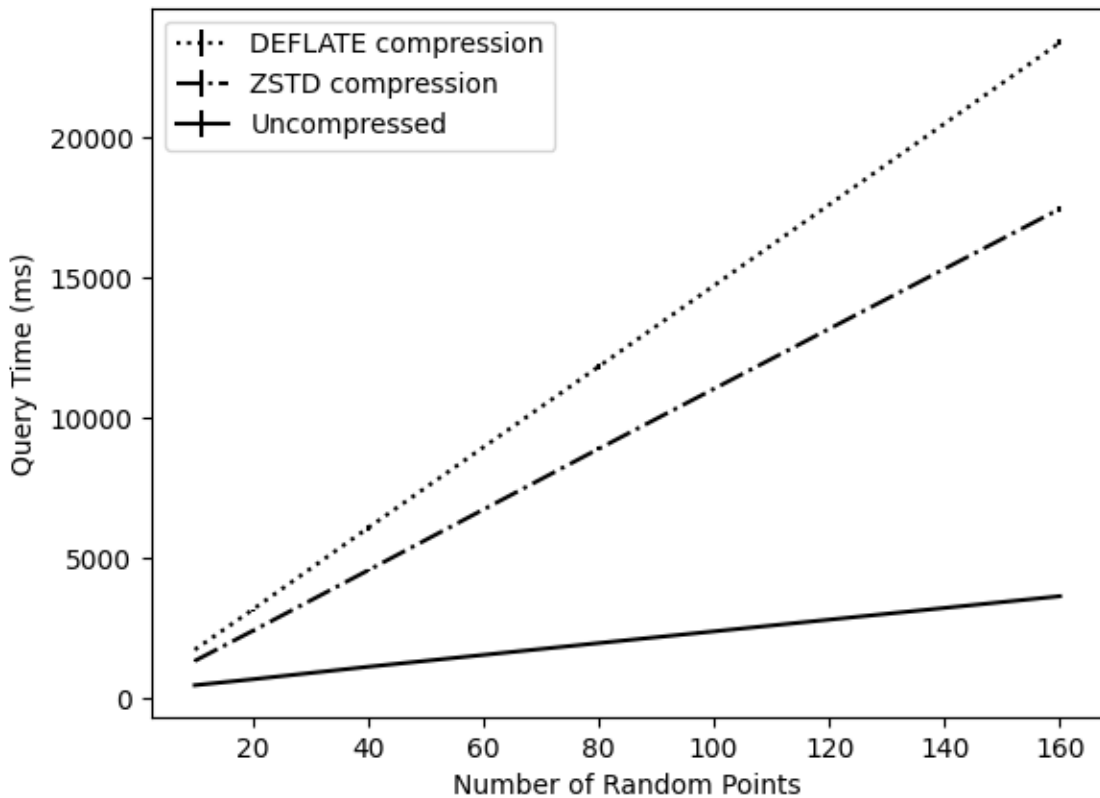


Figure 3 Results from the Query Benchmark

### 6.3.2 Changing the Compression of PostGIS out-db Raster

The compression vs. query performance experiment that was described in the section 6.3.1 was conducted because at the end of the project the data storage requirements needed to be released. However, we did not want to delete the data yet. Often the data is needed some time after the project ends, therefore we researched the option of compressing the files and still have access to the data if the need arises. The tradeoff of lower query performance is acceptable at the end of the project. The uncompressed GeoTIFF files required 17TB of storage. As the benchmark shows, after the compression the file sizes are reduced to under half of the original size. By using the ZSTD compression algorithm, the compression of all files resulted to only 7.5TB storage requirement, thus releasing over half of the required storage requirement.

The process of changing raster compression of out-db raster is simpler than one might expect. PostGIS uses the GDAL library as an abstraction layer for raster data access, thus decoupling the

file format from the database (*Postgis/Postgis at 3.1.8*, 2022). This results in the ability of changing the compression without updating the raster tile rows that are already in the PostGIS database. It seems that this feature is not documented anywhere, therefore even if the feature works in the 3.1 version of PostGIS where it was tested, it might be unwise to rely on this feature in future versions of PostGIS. Abstraction layers almost always cause overhead that reduces the performance. Therefore, it is easy to imagine a situation where the PostGIS developers decide to remove the abstraction layer to improve performance.

Since the compression does not need to touch the database rows, the process of compressing the rasters is trivially implemented using GDAL command-line tool. Listing 2 shows a script that first renames the old file; then compress the file with ZSTD algorithm with PREDICTOR setting 3; then it removes the old file. Because ZSTD is lossless compression algorithm, the original uncompressed file can be restored simply using the same method with COMPRESS=NONE argument (GDAL/OGR contributors, 2023a).

```
mv "${filename}" "${filename}.old"  
gdal_translate "${filename}.old" "${filename}" \  
-co COMPRESS=ZSTD -co PREDICTOR=3 -co BIGTIFF=YES  
rm "${filename}.old"
```

Listing 2 Script for Compressing the Raster

### 6.3.3 Workaround for Poor raster2pgsql Performance

During the development of the data platform, we noticed an extremely poor performance when using the raster2pgsql command-line tool to upload the rasters to PostGIS database. Later investigation revealed that the raster2pgsql NODATA check feature was responsible for the poor performance. The feature can be turned off using the raster2pgsql -k argument. However, the feature is very useful, since it does not upload tiles where all values are NODATA to the database. This is a very common situation at the edges of the satellite images, where there is a high number of NODATA values because the images are not perfectly square. Therefore, some other solution had to be found for the problem.

The raster2pgsql code was investigated for a possible fix for the poor performance, however the pixel value check was too tightly coupled with the GDAL file access abstraction for an easy fix. Fortunately, during the investigation the GDAL GeoTIFF driver the `GTIFF_DIRECT_IO` and `GTIFF_VIRTUAL_MEM_IO` options were discovered that improved the raster upload performance 9 to 10 times faster. The options are not very well documented, except that they turn on a special RasterIO implementation (GDAL/OGR contributors, 2023a). The options work only for uncompressed rasters, however that was not a problem in our case where we used uncompressed files anyhow. With the method that was described in section 6.3.2, the options can be utilized with compressed raster uploads too by first decompressing the file, then uploading it, and then changing the uncompressed file back to the compressed file. The issue and workaround was communicated back to the community by implementing a simple benchmark that demonstrates the problem and the solution (Alatalo, 2023).

#### 6.3.4 Utilizing the GDAL Command-Line Tools

“The best code is no code at all” (Jeff Atwood, 2007) is a commonly used saying in the field of software engineering. Code always contains bugs, and one strategy of reducing the number of bugs is to write less code. Therefore, it is sensible to use existing tools to solve the problem at hand. As was discussed in the section 5.6, the GDAL project includes a set of extremely useful command-line tools that were extensively used in this project. This section introduces some use cases for the command-line tools; about how they were used in the data platform implementation. This is not a comprehensive list of the functionality of these tools. However, they might serve as inspiration for other practitioners to read the GDAL documentation for these tools in case they have related problems where the command-line tools can be used to implement a simple reliable solution (GDAL/OGR contributors, 2023c).

The two simplest commands are `gdalinfo` and `ogrinfo`. They return information about the spatial data files. The `gdalinfo` command is for raster data formats and `ogrinfo` is for vector data formats. The commands can show information such as is the NODATA value set for GeoTIFF file, what coordinate system is in use, or what is the size of the raster in pixels and what are the spatial extent coordinates for the raster. With `-stats` argument, the `gdalinfo` command can also show simple statistics of the raster such as the minimum, maximum, and mean values.

Section 6.3.2 already showed how the `gdal_translate` command can be used to compress a GeoTIFF file. However, it can be also used to translate the raster between two formats. The following command transforms NetCDF file to GeoTIFF:

```
gdal_translate -of GTiff input.nc output.tif
```

Sometimes one needs to reproject a raster to different coordinate system. This can be beneficial when the source data is using different coordinate systems than some other spatial data that is already in PostGIS database. By using the same coordinate system there is no need to continuously use `ST_Transform` PostGIS function to transform the geospatial data between different coordinate systems, thus improving query performance. The reprojection of raster data can be achieved with the `gdalwarp` command. For example, the following command reprojects the `input.tif` to EPSG:3067 coordinate system.

```
gdalwarp -t_srs 'EPSG:3067' input.tif output.tif
```

Similar useful commands are available for vector data using the `ogr2ogr` command. The following example extracts Central-Finland municipality borders from a GeoPackage file that includes the information about Finland division into administrative areas (National Land Survey of Finland, 2023). The command also reprojects the data to EPSG:4326 coordinate system and simplifies the complex geometry.

```
ogr2ogr -t_srs "EPSG:4326" -where "namefin = 'Keski-Suomi'" \  
-simplify 10000 -f geojson \  
keskisuomi.geojson SuomenHallinnollisetKuntajakopohjaisetAluejaot_2023_10k.gpkg \  
Maakunta
```

Second example of `ogr2ogr` was used in other project where the same data platform was used to study speed limits in Finland. The command extracts the speed limits directly from Finnish Transport Infrastructure Agency's open WFS service and uploads the data directly to PostGIS (Finnish Transport Infrastructure Agency, 2023)



```
ogr2ogr -f "PostgreSQL" "PG:dbname=vaylapilvi" \  
"https://avoinapi.vaylapilvi.fi/vaylatiedot/ows?service=wfs&request=getCapabilities" \  
tierekisteri:tl168
```

## 7 Conclusion

The research demonstrates that open-source projects can be used to implement an efficient data storage platform that can be used to support geospatial research. The platform supported the simple exploratory data analysis which is crucial part of the research at the beginning where the data is first familiarized with. Likewise, the platform scaled without any issues to the final use case where it was used to build the training dataset for the neural network model that was developed in the research project. The SQL query language support that was provided by the PostGIS database was found to be an excellent interface for accessing and analyzing the data from the data storage system. Visualization of the satellite images from the data storage platform was at first a challenge, however when investigating the QGIS application and discovering its great extendibility using the plugin API, the problem was quickly solved by implementing a custom extension for the application. The open-source tooling that is available for handling and processing spatial data is excellent. The ETL system was implemented mostly by using pre-existing open-source command-line tools that were used from shell scripts, thus requiring only small amount of custom code. The overall success of the data platform was demonstrated by supporting the research that used the data from the data storage platform to conduct research. The change detection research produced a scientific research article that is attached to this document as an Appendix 1. The success of the research validated that the implemented data platform solves the problem it was designed to solve. The data storage platform could be further developed to continuously fetch the newest satellite images to the platform. The ETL system in the current implementation is executed manually only once with a predefined time range that is used to fetch and process the satellite images and other data. The implementation was good enough for this use case. However, many applications might need access to the most recent data. For example, if the model that was developed in the change detection research was used in production setting, it would need access to the most recent data to make predictions. Implementing this type of feature might be trivial with a workflow management software such as Apache Airflow as was discussed in the section 6.1.5. The software could be used to schedule the execution of the ETL pipeline with a time interval.

## References

- Aalto, J., Pirinen, P., & Jylhä, K. (2016). New gridded daily climatology of Finland: Permutation-based uncertainty estimates and temporal trends in climate. *Journal of Geophysical Research: Atmospheres*, *121*(8), 3807–3823. <https://doi.org/10.1002/2015JD024651>
- Alatalo, J. (2023). *Workaround for slow raster2pgsql upload speed when using out-db rasters and GeoTIFF files*. <https://github.com/janne-alatalo/slow-raster2pgsql-workaround>
- Baghdadi, N., Mallet, C., & Zribi, M. (Eds.). (2018). *QGIS and generic tools*. iSTE : Wiley.
- Brovelli, M. A., Minghini, M., Moreno-Sanchez, R., & Oliveira, R. (2017). Free and open source software for geospatial applications (FOSS4G) to support Future Earth. *International Journal of Digital Earth*, *10*(4), 386–404. <https://doi.org/10.1080/17538947.2016.1196505>
- Chapter 15. *PostGIS Special Functions Index*. (n.d.). Retrieved April 3, 2023, from [https://postgis.net/docs/PostGIS\\_Special\\_Functions\\_Index.html](https://postgis.net/docs/PostGIS_Special_Functions_Index.html)
- Congalton, R. G. (1997). Exploring and evaluating the consequences of vector-to-raster and raster-to-vector conversion. *Photogrammetric Engineering and Remote Sensing*, *63*(4), 425–434.
- Engström, E., Storey, M.-A., Runeson, P., Höst, M., & Baldassarre, M. T. (2020). How software engineering research aligns with design science: A review. *Empirical Software Engineering*, *25*(4), 2630–2660. <https://doi.org/10.1007/s10664-020-09818-7>
- ESRI. (1998). *ESRI Shapefile Technical Description*. <https://www.esri.com/content/dam/esrisites/sitecore-archive/Files/Pdfs/library/whitepapers/pdfs/shapefile.pdf>
- European commission. (n.d.). *Legal notice on the use of Copernicus Sentinel Data and Service Information*. Retrieved March 31, 2023, from [https://sentinels.copernicus.eu/documents/247904/690755/Sentinel\\_Data\\_Legal\\_Notice/](https://sentinels.copernicus.eu/documents/247904/690755/Sentinel_Data_Legal_Notice/)

- European Space Agency. (n.d.). *User Guides—Sentinel-1 SAR - SAR Formats—Sentinel Online*. Sentinel Online. Retrieved April 7, 2023, from <https://copernicus.eu/user-guides/sentinel-1-sar/data-formats/sar-formats>
- European Space Agency. (2022). *SNAP - ESA sentinel Application Platform v8.0.0*.  
<https://step.esa.int/main/toolboxes/snap/>
- Farkas, G. (2017). *Practical GIS: Use tools such as QGIS, PostGIS, and GeoServer to build powerful GIS solutions* (1st edition). Packt.
- Filipponi, F. (2019). Sentinel-1 GRD Preprocessing Workflow. *Proceedings*, 18(1), Article 1.  
<https://doi.org/10.3390/ECRS-3-06201>
- Finnish Transport Infrastructure Agency. (2023, March 9). *Open API*. Finnish Transport Infrastructure Agency's Open API's. <https://vayla.fi/en/transport-network/data/open-data/api>
- Fletcher, K. & European Space Agency (Eds.). (2012). *Sentinel-1: ESA's radar observatory mission for GMES operational services*. ESA Communications.
- Garrard, C. (2016). *Geoprocessing with Python* (1st edition). Manning Publications.
- GDAL/OGR contributors. (2023a). *GTiff – GeoTIFF File Format—GDAL documentation*.  
<https://gdal.org/drivers/raster/gtiff.html>
- GDAL/OGR contributors. (2023b). *NetCDF: Network Common Data Form—GDAL documentation*.  
<https://gdal.org/drivers/raster/netcdf.html>
- GDAL/OGR contributors. (2023c). *Programs—GDAL documentation*. <https://gdal.org/programs/index.html>
- GDAL/OGR contributors. (2023d). *Raster drivers—GDAL documentation*. <https://gdal.org/drivers/raster/index.html>
- GDAL/OGR contributors. (2023e). *Vector drivers—GDAL documentation*. <https://gdal.org/drivers/vector/index.html>

- Gerlek, M. P., & Fleagle, M. (2007). Imaging on the Geospatial Web Using JPEG 2000. In A. Scharl & K. Tochtermann (Eds.), *The Geospatial Web: How Geobrowsers, Social Software and the Web 2.0 are Shaping the Network Society* (pp. 27–38). Springer.  
[https://doi.org/10.1007/978-1-84628-827-2\\_3](https://doi.org/10.1007/978-1-84628-827-2_3)
- Jachym Cepicky. (2017, October 8). *Shapefile must die!* <http://switchfromshapefile.org/>
- Jamk University of Applied Sciences. (2018). *Ethical Principles for JAMK University of Applied Sciences*. <https://www.jamk.fi/en/media/34826>
- Jamk University of Applied Sciences. (2021). *Data for Utilisation—Leveraging digitalisation through modern artificial intelligence solutions and cybersecurity*. JAMK University of Applied Sciences - JAMK. <https://www.jamk.fi/en/research-and-development/rdi-projects/data-for-utilisation-leveraging-digitalisation-through-modern-artificial-intelligence-solutions-and>
- Jeff Atwood. (2007, May 30). *The Best Code is No Code At All*. Coding Horror.  
<https://blog.codinghorror.com/the-best-code-is-no-code-at-all/>
- Khan, S., & Mohiuddin, K. (2018). Evaluating the parameters of ArcGIS and QGIS for GIS Applications. *International Journal of Advance Research in Science and Engineering*, 7(3), 582–594.
- Kimball, R., & Caserta, J. (2004). *The data warehouse ETL toolkit: Practical techniques for extracting, cleaning, conforming, and delivering data* (1st edition) [Electronic resource]. Wiley.
- McInerney, D., & Kempeneers, P. (2015). Raster Data Explained. In D. McInerney & P. Kempeneers (Eds.), *Open Source Geospatial Tools: Applications in Earth Observation* (pp. 51–60). Springer International Publishing. [https://doi.org/10.1007/978-3-319-01824-9\\_3](https://doi.org/10.1007/978-3-319-01824-9_3)
- National Land Survey of Finland. (n.d.-a). *Formats of the National Land Survey of Finland's geospatial datasets*. National Land Survey of Finland. Retrieved April 7, 2023, from <http://www.maanmittauslaitos.fi/en/maps-and-spatial-data/professionals/topographic-data-and-how-acquire-it/formats-national-land>

- National Land Survey of Finland. (n.d.-b). *Karttakuvapalvelu (WMS, WMTS, Vektoritiilet)*. Retrieved April 8, 2023, from <https://www.maanmittauslaitos.fi/karttakuvapalvelu>
- National Land Survey of Finland. (2023). *Division into administrative areas (vector)*. National Land Survey of Finland. <http://www.maanmittauslaitos.fi/en/maps-and-spatial-data/professionals/product-descriptions/division-administrative-areas-vector>
- Obe, R. O., & Hsu, L. S. (2021). *PostGIS in action* (Third edition). Manning Publications.
- Peffer, K., Tuunanen, T., Gengler, C. E., Rossi, M., Hui, W., Virtanen, V., & Bragge, J. (2006). Design Science Research Process: A Model for Producing and Presenting Information Systems Research. *Proceedings of the First International Conference on Design Science Research in Information Systems and Technology*, 83–106. <https://doi.org/10.48550/arXiv.2006.02763>
- Piórkowski, A. (2011). MYSQL SPATIAL AND POSTGIS – IMPLEMENTATIONS OF SPATIAL DATA STANDARDS. *Electronic Journal of Polish Agricultural Universities*, 14(1). <http://www.ejpau.media.pl/articles/volume14/issue1/art-03.pdf>
- Pons, X., & Masó, J. (2016). A comprehensive open package format for preservation and distribution of geospatial data and metadata. *Computers & Geosciences*, 97, 89–97. <https://doi.org/10.1016/j.cageo.2016.09.001>
- PostGIS: License*. (2023). [PLpgSQL]. postgis. <https://github.com/postgis/postgis/blob/956a0bfda7ee982942c04a64a676209c7145caaf/COPYING> (Original work published 2012)
- Postgis/postgis at 3.1.8*. (2022). <https://github.com/postgis/postgis/tree/3.1.8>
- PostgreSQL: License*. (2023). <https://www.postgresql.org/about/licence/>
- QGIS project. (2023, April 4). *PyQGIS Cookbook—16. Developing Python Plugins*. [https://docs.qgis.org/3.28/en/docs/pyqgis\\_developer\\_cookbook/plugins/index.html](https://docs.qgis.org/3.28/en/docs/pyqgis_developer_cookbook/plugins/index.html)
- Rew, R., & Davis, G. (1990). NetCDF: An interface for scientific data access. *IEEE Computer Graphics and Applications*, 10(4), 76–82. <https://doi.org/10.1109/38.56302>

- She, B., Hu, T., Zhu, X., & Bao, S. (2019). Bridging open source tools and Geoportals for interactive spatial data analytics. *Geo-Spatial Information Science*, 22(3), 185–192.  
<https://doi.org/10.1080/10095020.2019.1645497>
- Sood, V., Gusain, H. S., Gupta, S., Taloor, A. K., & Singh, S. (2021). Detection of snow/ice cover changes using subpixel-based change detection approach over Chhota-Shigri glacier, Western Himalaya, India. *Quaternary International*, 575–576, 204–212.  
<https://doi.org/10.1016/j.quaint.2020.05.016>
- Steiniger, S., & Hunter, A. J. S. (2012). Free and Open Source GIS Software for Building a Spatial Data Infrastructure. In E. Bocher & M. Neteler (Eds.), *Geospatial Free and Open Source Software in the 21st Century: Proceedings of the first Open Source Geospatial Research Symposium, OGRS 2009* (pp. 247–261). Springer Berlin Heidelberg. [https://doi.org/10.1007/978-3-642-10595-1\\_15](https://doi.org/10.1007/978-3-642-10595-1_15)
- Steiniger, S., & Hunter, A. J. S. (2013). The 2012 free and open source GIS software map – A guide to facilitate research, development, and adoption. *Computers, Environment and Urban Systems*, 39, 136–150. <https://doi.org/10.1016/j.compenvurbsys.2012.10.003>
- Stolze, K. (2003). *SQL/MM spatial: The standard to manage spatial data in a relational database system*. Gesellschaft für Informatik e.V. <http://dl.gi.de/handle/20.500.12116/30056>
- Sublime, J., & Kalinicheva, E. (2019). Automatic Post-Disaster Damage Mapping Using Deep-Learning Techniques for Change Detection: Case Study of the Tohoku Tsunami. *Remote Sensing*, 11(9), Article 9. <https://doi.org/10.3390/rs11091123>
- Suleykin, A., & Panfilov, P. (2019). Implementing big data processing workflows using open source technologies. *Proceedings of the 30th DAAAM International Symposium*, 0394–0404.  
<https://doi.org/10.2507/30th.daaam.proceedings.054>

- Suleykin, A., & Panfilov, P. (2020). Metadata-Driven Industrial-Grade ETL System. *2020 IEEE International Conference on Big Data (Big Data)*, 2433–2442. <https://doi.org/10.1109/Big-Data50022.2020.9378367>
- Swain, N. R., Latu, K., Christensen, S. D., Jones, N. L., Nelson, E. J., Ames, D. P., & Williams, G. P. (2015). A review of open source software solutions for developing water resources web applications. *Environmental Modelling & Software*, *67*, 108–117. <https://doi.org/10.1016/j.envsoft.2015.01.014>
- Tange, O. (2011). GNU Parallel—The Command-Line Power Tool. *;;Login: The USENIX Magazine*, *36*(1), 42–47.
- Varol, M. B., & Şanlıoğlu, İ. (2017). Open geospatial consortium web map and feature services and free/open source server/client softwares. *International Journal of Engineering and Geosciences*, *2*(1), 17–26. <https://doi.org/10.26833/ijeg.286691>
- Warmerdam, F. (2008). The Geospatial Data Abstraction Library. In G. B. Hall & M. G. Leahy (Eds.), *Open Source Approaches in Spatial Data Handling* (pp. 87–104). Springer. [https://doi.org/10.1007/978-3-540-74831-1\\_5](https://doi.org/10.1007/978-3-540-74831-1_5)
- Will, M. (2023). *SentinelSat/sentinelSat* [Python]. sentinelSat. <https://github.com/sentinelSat/sentinelSat> (Original work published 2015)
- Xie, Q., Chen, F., Zhang, Z., & Lucena, I. (2013). In-database image processing in Oracle Spatial GeoRaster. *ASPRS 2013 Annual Conference Baltimore, Maryland*.
- Zhang, L., & Yi, J. (2010). Management methods of spatial data based on PostGIS. *2010 Second Pacific-Asia Conference on Circuits, Communications and System*, *1*, 410–413. <https://doi.org/10.1109/PACCS.2010.5626962>
- Zhu, F., Yang, J., & Guo, Q. (2011). Emergency GIS system based on GML and multi-source spatial data. *2011 IEEE 3rd International Conference on Communication Software and Networks*, 86–90. <https://doi.org/10.1109/ICCSN.2011.6014681>

## Appendices

### Appendix 1. Submitted Research Article

The article was submitted to IEEE Transactions on Geoscience and Remote Sensing journal on 31<sup>st</sup> of March in 2023. A preprint version of the article is published in arXiv preprint repository <https://arxiv.org/abs/2303.17835>.



---


# IMPROVED DIFFERENCE IMAGES FOR CHANGE DETECTION CLASSIFIERS IN SAR IMAGERY USING DEEP LEARNING

---

A PREPRINT

 **Janne Alatalo**

Jamk University of Applied Sciences  
Institute of Information Technology  
Piippukatu 2, 40100 Jyväskylä, Finland  
janne.alatalo@jamk.fi

 **Tuomo Sipola**

Jamk University of Applied Sciences  
Institute of Information Technology  
Piippukatu 2, 40100 Jyväskylä, Finland  
tuomo.sipola@jamk.fi

 **Mika Rantonen**

Jamk University of Applied Sciences  
Institute of Information Technology  
Piippukatu 2, 40100 Jyväskylä, Finland  
mika.rantonen@jamk.fi

May 4, 2023

## ABSTRACT

Satellite-based Synthetic Aperture Radar (SAR) images can be used as a source of remote sensed imagery regardless of cloud cover and day-night cycle. However, the speckle noise and varying image acquisition conditions pose a challenge for change detection classifiers. This paper proposes a new method of improving SAR image processing to produce higher quality difference images for the classification algorithms. The method is built on a neural network-based mapping transformation function that produces artificial SAR images from a location in the requested acquisition conditions. The inputs for the model are: previous SAR images from the location, imaging angle information from the SAR images, digital elevation model, and weather conditions. The method was tested with data from a location in North-East Finland by using Sentinel-1 SAR images from European Space Agency, weather data from Finnish Meteorological Institute, and a digital elevation model from National Land Survey of Finland. In order to verify the method, changes to the SAR images were simulated, and the performance of the proposed method was measured using experimentation where it gave substantial improvements to performance when compared to a more conventional method of creating difference images.

**Keywords** change detection · Sentinel-1 · SAR · U-Net · mapping transformation function · remote sensing

## 1 Introduction

Remote sensing change detection can be used for many purposes, such as damage assessment after a natural disaster [1–3], detection of forest damages after a storm [4, 5], and monitoring deforestation and glacier melting [6, 7], to name only a few. Change detection works by comparing two images that have been captured at different dates in the same geographical location and finding the areas that have changed during the time between the acquisitions [8]. Different platforms can be used to image the terrain, such as airplanes and satellites, however only satellites provide the advantage of continuously monitoring the whole planet [9]. The revisit time of some satellite systems can be as short as only a few days, and the images are available from anywhere in the planet. This makes the satellite images a useful source of remote sensing data for change detection applications. Some space agencies, such as European Space Agency (ESA), provide some of the satellite images for anybody to download and use [10]. The ease of acquiring the data further facilitates the development of change detection systems that are based on the satellite remote sensing techniques. The

images from the satellites are captured using either optical or radar sensors, with radar having the advantage of piercing the cloud layer, thus enabling it to work in various weather conditions [9]. However, the radar satellites have their disadvantages as well. The resolution of the images is not as good as what the optical instruments can produce. The resolution of the radar images is defined by the antenna length and the frequency band of the radar signal. To enable higher resolution images, the satellites use the synthetic aperture radar (SAR) technique, where the satellite movement over the ground is utilized to synthesize virtual aperture that is longer than the physical antenna on the satellite [11]. However, even with the SAR technique the radar images are lower resolution when compared to the optical images. ESA has the Sentinel-1 mission with two SAR satellites that operate on the C-band and have the spatial resolution of around  $5 \times 20$  meters [12]. Likewise, speckle noise reduces the quality of the SAR imagery. SAR images always have a grainy look from the speckle, which is random noise that is always present in the images. Despite the shortcomings of the SAR imagery, they are commonly used in remote sensing change detection [13–16].

One approach to implement a change detection system, that is generally used in unsupervised change detection, is to proceed in steps [17]. Figure 1 illustrates this method. The images are first preprocessed to make them comparable among each other. Then, two images from the same location, that are captured at different times, are used to produce a difference image (DI) using an algebraic operation like subtraction, ratio, or log ratio. Finally, the DI is analysed by a classifier algorithm to produce a change map that indicates the changed regions. The preprocessing step is crucial for this method to work well. The issue with the speckle noise is commonly recognized problem with change detection on SAR imagery [13, 15, 16], and to mitigate the issue, noise suppression algorithms are used in the image preprocessing step. However, it is impossible to remove the noise completely, thus the DI also includes noise that causes misclassifications in the classification step. Likewise, other image properties that influence the image comparability have an effect to the quality of the DI. This includes properties such as the satellite orbit direction, incidence angle, and ground moisture content. The satellite does not capture the image from the same angle during every revisit. In case of the ESA Sentinel-1 satellites, the satellite can be flying from North to South, or from South to North, during the image acquisition, and the satellite orbit can be higher or lower with respect to the horizon from the ground perspective between the overflies. The satellite imaging angle influences how the radar signal backscatters from the ground features [18], which results in that images taken from different imaging angles likely produce lower quality DI than images taken from the same imaging angle. Likewise, ground weather conditions can influence the DI quality. Soil moisture content changes the dielectric constant of the soil, thus changing the backscatter intensity of the radar signal [19]. Images that are taken in similar weather conditions are likely to produce better quality DI when compared to images that are taken in different weather conditions. One solution to improve the DI quality is to favour images with similar acquisition conditions when selecting the images that are used to produce the DI. However, this is not always possible.

In this paper we introduce a new method of producing better quality difference images by using neural network-based mapping transformation function preprocessing step that factors in the image acquisition conditions of the SAR images, thus making the SAR images more comparable. Existing research about SAR image preprocessing has focused on removing speckle noise from the images [20, 21], or correcting the incidence angle variation [22, 23]. However, to the best of knowledge of the authors, this is the first time when the comparability of the SAR images is improved by taking in to account the overall image acquisition conditions using neural network-based preprocessing step. Project code is available on GitHub <sup>1</sup>.

## 2 Materials and Methods

### 2.1 Proposed Method

Figure 2 illustrates the overall architecture of the proposed method. It replaces the conventional method that is illustrated in Figure 1 image differencing step. The idea of the proposed method is to improve the SAR image comparability by considering the acquisition conditions of the SAR images. The proposed method utilizes a mapping transformation function that creates artificial SAR images in the requested acquisition conditions. The mapping transformation function  $\mathcal{F}$  is a neural network model that is trained to predict the SAR image at the time  $t$  ( $I_t$ ). The neural network output  $\hat{I}_t$  is the artificial SAR image that is created in the acquisition conditions of  $I_t$ , therefore it should be more comparable to the  $I_t$  than previous SAR images from the location that might have been captured in different acquisition conditions. The model input consists of three distinct features, which are: The previous SAR images from the location; the acquisition conditions of the SAR images (including at time  $t$ ); and the digital elevation map from the location. The objective of the neural network model is to learn to replicate the SAR image at the time  $t$ . The only information from the time  $t$  in the model input are the image acquisition conditions of the  $I_t$ . This means that for the model to be able to replicate the  $I_t$ , it needs to learn to map the information contained in the previous SAR images and the digital elevation map to the image acquisition conditions of the  $I_t$ . With an ideal model that could perfectly replicate the  $I_t$ , the  $\hat{I}_t$  and  $I_t$

<sup>1</sup><https://github.com/janne-alatalo/sar-change-detection>

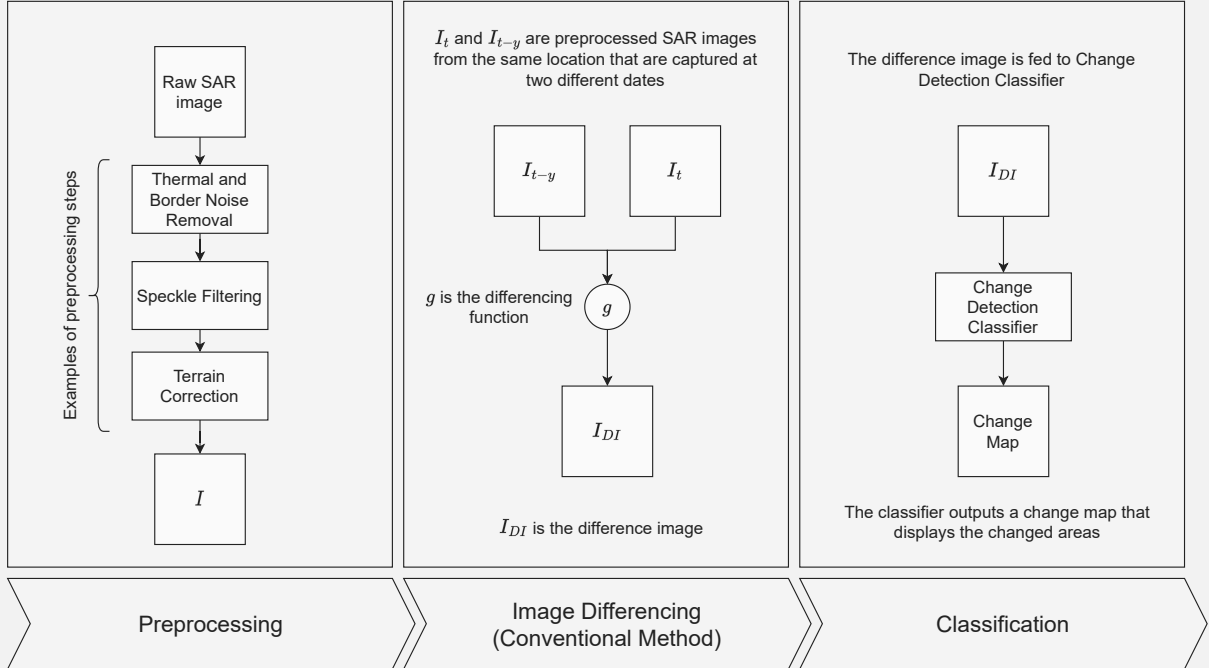


Figure 1: Change detection is often implemented in three distinct steps. The first step is to make the images more comparable to each other using a preprocessing pipeline. The preprocessed images are then used to create difference images ( $I_{DI}$ ) using a function  $g$  that is often an algebraic operation, such as subtraction, ratio, or log ratio. The  $I_{DI}$  is then used as an input to a change detection classifier that produces the change map that displays the changed areas. The figure illustrates the conventional method of producing the difference images by using two SAR images that are captured from the location in two different dates.

would be identical if nothing has changed between the image acquisition of the  $I_{t-1}$  and  $I_t$ , however the  $\hat{I}_t$  would be missing the change if something had changed after the previous image acquisition since the information of the change is not included in the model input data. In practice the SAR images include random noise that is impossible to replicate accurately, and the acquisition conditions are not accurate enough for perfect replication of the  $I_t$ , therefore the  $\hat{I}_t$  only approximates the  $I_t$ .

The intuitive description of the  $\hat{I}_t$  is that the neural network-based mapping transformation function produces a prediction how the  $I_t$  should look like based on previous information about the location and the actual imaging conditions of the  $I_t$ . The produced image  $\hat{I}_t$  can be used with the actual image  $I_t$  to create the difference image  $\hat{I}_{DI}$  by using a simple algebraic operation like subtraction, ratio, or log ratio. Generating the difference image is the standard method of conducting change detection, especially when using unsupervised methods [17].

Conventional methods of producing the difference image often use only one of the previously captured images with the most recent image to generate the image e.g.  $I_{DI} = g(I_t, I_{t-y})$  [24]. This method has the previously discussed drawbacks of noise and imaging conditions affecting the final difference image quality. By using the proposed mapping transformation function, the predicted image  $\hat{I}_t$  is used in the place of the previously captured image to generate the difference image e.g.  $\hat{I}_{DI} = g(I_t, \hat{I}_t)$ . The predicted image  $\hat{I}_t$  does not contain noise and the mapping transformation function can correct the acquisition condition mismatch between the images, therefore the proposed method should produce better quality difference images when comparing it to the conventional method.

SAR imaging is sensitive to the soil moisture content of the imaged area [19]. Change in the soil moisture level changes the dielectric constant of the soil, and that way changes the SAR backscatter intensity. Often the soil moisture content changes should be ignored by the change detection system. Otherwise, the system would notify changes after every rainy day. This is one of the advantages of the proposed method. By adding weather to the model input acquisition condition parameters, the mapping transformation function can learn to construct the  $\hat{I}_t$  in the actual weather conditions

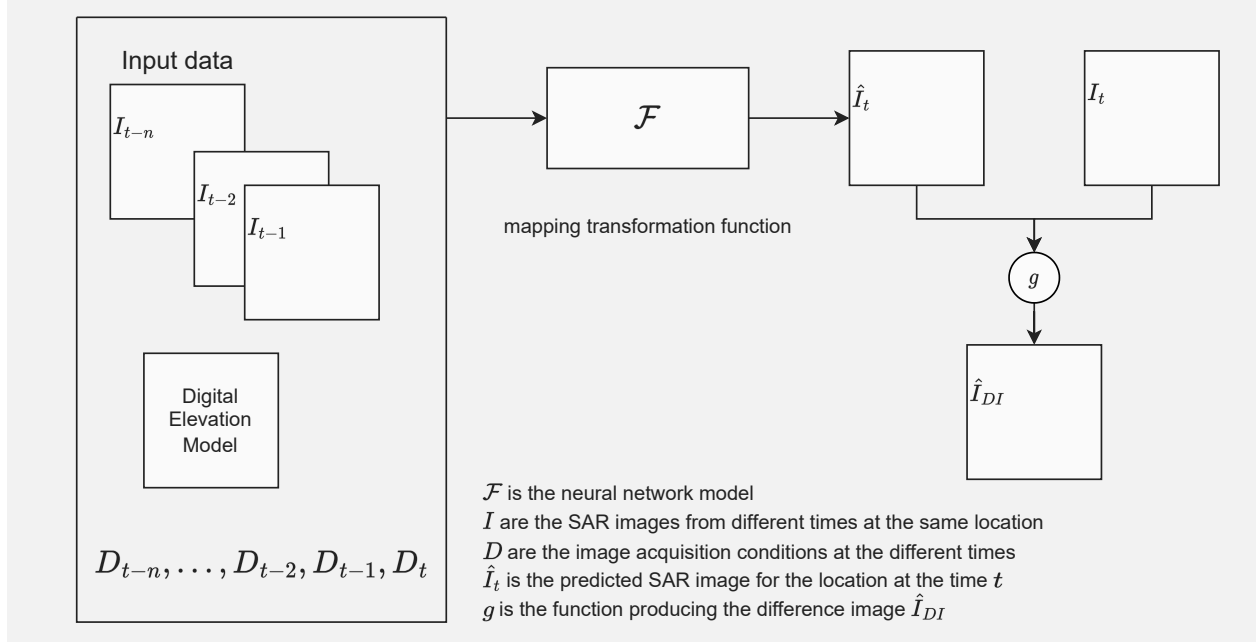


Figure 2: Architectural overview of the proposed method. The neural network-based mapping transformation function fuses the information from previous image acquisitions and predicts what the scene should look like at the imaging conditions of  $I_t$ . The model output image  $\hat{I}_t$  and the actual image  $I_t$  is used to produce the difference image  $\hat{I}_{DI}$ .

of  $I_t$  and should correctly model the changes in the soil moisture changing the backscatter intensity. Therefore, the false positive changes, that are caused by soil moisture changes, are reduced.

In addition of weather, the acquisition condition parameters also include the imaging angle and identify the satellite that captured the image. A location is imaged by one of the sentinel satellites with an interval ranging from a few days to about a week. The satellite does not capture the image from the same angle every time. The satellite can be in ascending or descending orbit during the image acquisition and the incidence angle can vary between the overpasses. The ascending or descending orbit changes the look direction of the satellite, and that way has a considerable affect to the resulting image. The Sentinel-1 satellites are right-looking. When the satellite is descending from North to South it is imaging to the direction of West, and for ascending passes it is imaging to the direction of East [25]. Various 3D features, like forest edges, lake banks and hills are sensitive to the look direction, therefore the imaging angle is an important parameter when computing the difference image. When using an image differencing method where only one previous image is used for difference image computation, the imaging angle of the most recent image can restrict what previous images can be used to produce the difference image. Seasonal changes, like foliage growth or change in snow cover, means that the most optimal image for the differencing would be the most recent previous image, however different imaging angles can limit the usage of the most recent images. This problem is not present with the proposed method. The model input includes  $n$  previous images and their imaging angle information. The model output image  $\hat{I}_t$  is produced using the actual acquisition conditions of  $I_t$ . The model can use all the information from all  $n$  input images, despite the input including images from different look directions, and the produced image  $\hat{I}_t$  represents an image that is acquired from the same angle as  $I_t$ .

## 2.2 Neural Network Architecture

Figure 2 illustrates the architecture of the neural network-based mapping transformation function. The architecture is based on the well-known U-Net neural network architecture [26]. The previous  $n$  SAR images, and the digital elevation map (DEM) are stacked to construct the input. The previous images and the DEM are all from the same location. The images are projected to the same resolution and the pixels across the different images are aligned to match the same geographical position. The U-Net architecture is constructed from encoder and decoder units. The encoder takes the input and compresses the input image stack to the latent space by using a set of downsampler blocks that half the input resolution using convolution layers with stride  $2 \times 2$ . The encoder stacks enough downsampler blocks so that the input image stack is compressed to  $1 \times 1$  resolution in image height and width dimensions. The image acquisition

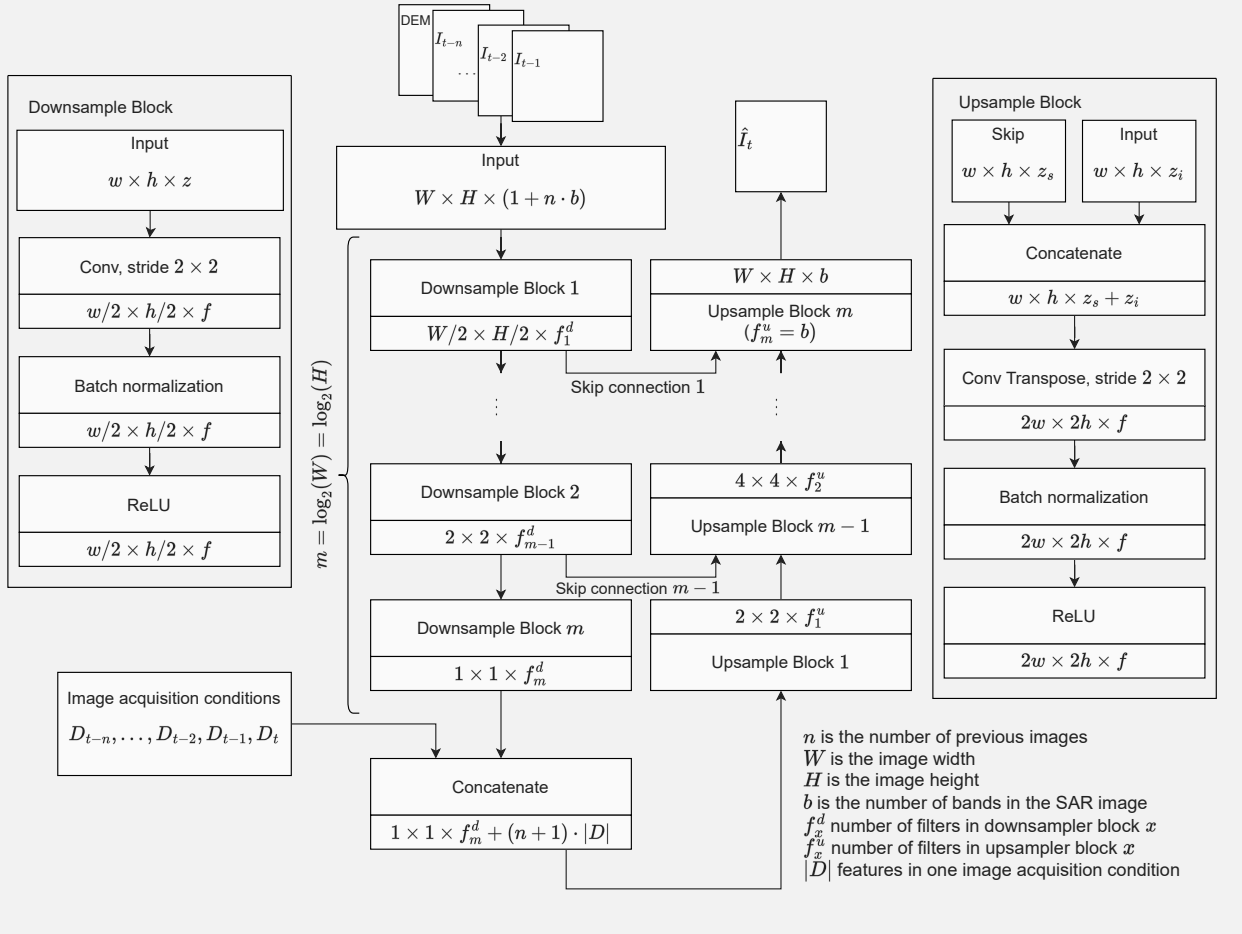


Figure 3: The neural network architecture for the mapping transformation function. The architecture is based on the well-known U-Net neural network architecture. The image acquisition conditions are injected to the latent vector between the encoder and decoder.

conditions vector, that contains the information of the acquisitions conditions for the  $n$  input images and the target image, is concatenated to the latent vector. The resulting vector is then fed to the decoder that decodes the vector back to the dimensions of a normal SAR image outputting the  $\hat{I}_t$ . The decoder is constructed from upsampler blocks that double the width and height dimensions using transposed convolution layers with stride  $2 \times 2$ . The decoder has same amount of upsampler blocks as the encoder has downsampler blocks. The number of filters, that are used in the upsampler and downsampler blocks, can be configured for every block individually, except for the final upsampler block that has the same number of filters as the SAR image has bands. The encoder and decoder layers are connected with skip connections that help the model in producing the output by not forcing the model to pack all the information to the latent vector. Instead, the information can flow from the input to the output by skipping most of the layers in the architecture. This is a standard method in U-Net style architectures.

### 2.3 Dataset

A dataset is needed for the training of the neural network-based mapping transformation function. As discussed previously, the mapping transformation function input is composed from the previously taken SAR images; the acquisition conditions of the previous and the most recent SAR image; and the digital elevation map from the location. The objective of the model is to learn to predict the most recent SAR image based on the input, therefore the most recent SAR image is the target in the training dataset. This means that the training dataset does not require any labelled data making the learning process of the proposed method unsupervised and economical to implement. The dataset can be generated directly from available data sources without needing human labelling for the data.

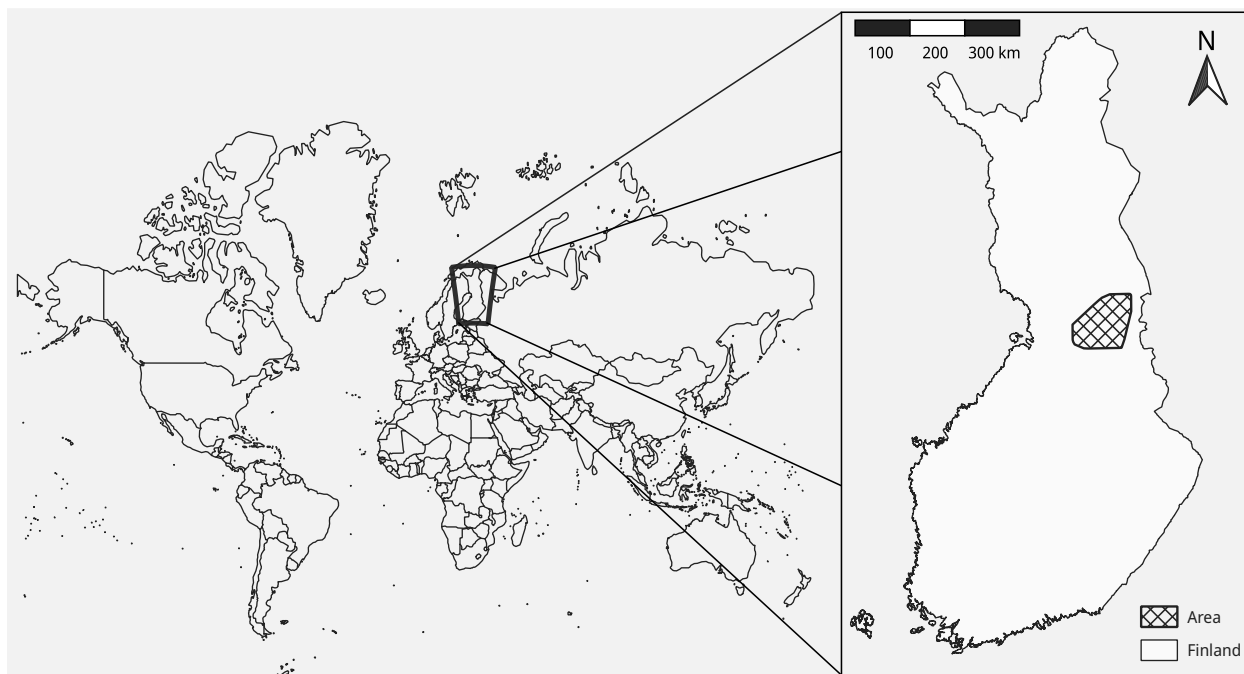


Figure 4: The dataset was generated from images acquired from the marked area. The figure contains data from the National Land Survey of Finland Topographic Database [32] and data from @EuroGeographics distributed by Eurostat [33].

The SAR images for the dataset were acquired from the ESA Copernicus Open Access Hub [27]. The Ground Range Detected products were used in this study [28]. The images were captured between March 2020 and August 2021 from the area illustrated in the Figure 4. All images from the time frame that included the area were downloaded from the Copernicus Open Access Hub. The images were preprocessed using the Sentinel-1 Toolbox from the Sentinel Application Platform (SNAP) [29], by applying the data preprocessing workflow described by Filipponi in [30]. The optional noise filtering step was applied to the dataset using the Refined Lee filter from the SNAP toolkit. The more accurate AUX\_POEORB precise orbit files were used in the Apply Orbit File step. The AUX\_POEORB files are available 20 days after the image acquisition [31], and since the processing was done in spring 2022, the more accurate orbit files were available for all images. The proposed workflow in [30] uses the SRTM Digital Elevation Database in the Range Doppler Terrain correction step, however the database does not cover the area from where the dataset was created, therefore the Copernicus 30m Global DEM was used that does cover the area. The SNAP toolkit can automatically download the required DEM files during preprocessing and the Terrain Correction step supports multiple different DEM sources, including the Copernicus 20m Global DEM, thus the change was trivial to implement. The preprocessed images were saved as GeoTIFF files and uploaded to PostgreSQL<sup>2</sup> database that was using the PostGIS<sup>3</sup> extension. Using a relational database as the storage backend simplified the dataset generation process since all the data was available in one place and queryable with SQL.

Although the Copernicus 30m Global DEM was used in the SAR image terrain correction preprocessing step, the product was not used for the mapping transformation function input. Instead, we used more accurate DEM from National Land Survey of Finland (NLS). NLS provides the DEM in multiple different resolutions of which the most accurate 2m grid DEM was used [34]. The data is open access and distributed under Attribution 4.0 International (CC BY 4.0) license<sup>4</sup>. The DEM was downloaded in GeoTIFF format and uploaded to the same PostgreSQL database with the SAR images.

As discussed before, the image acquisition condition data included information about the weather when the images were captured. This data was acquired from Finnish Meteorological Institute (FMI) that provides daily weather observations that are interpolated to  $1 \times 1$  km grid [35]. The interpolation method is described by Aalto et al. in [36]. The data is distributed in NetCDF format and uploaded once a month. Daily mean temperature, daily precipitation sum, and

<sup>2</sup><https://www.postgresql.org/>

<sup>3</sup><https://postgis.net/>

<sup>4</sup><https://creativecommons.org/licenses/by/4.0/>

snow depth data was downloaded from the time range. The daily observations were extracted from the NetCDF files, converted to daily GeoTIFF rasters, and uploaded to the same PostgreSQL database with the SAR images and DEM.

The final data samples were created by sampling random locations from the area and random dates from the time range. For training dataset, the time range was limited to the time before 20th of June in 2021, and for the test dataset the time was limited after the date. The image size was set to  $512 \times 512$  pixels, and number of previous images was set to 4. To keep the spatial resolution of the SAR images essentially unchanged, the geographical dimensions of the images was set to  $3 \times 3$  km. For each random location and date, the target SAR image  $I_t$  was the next SAR image from the location that was available after the date. The input SAR images  $I_{t-4}, I_{t-3}, I_{t-2}, I_{t-1}$  were the SAR images from the four previous acquisitions from the location that were captured before the  $I_t$ . The SAR images and the DEM was queried from the PostgreSQL database and the rasters were projected to the same projection window with the same  $512 \times 512$  resolution and  $3 \times 3$  km spatial dimensions using GDAL library [37]. The `gdal.Translate` function was used for the projection with nearest neighbor resampling algorithm. After the projection, all pixels were geographically aligned across all images and the images could be stacked to construct the input image stack. The Sentinel-1 satellites use Interferometric Wide swath mode with dual polarization over the land areas thus one SAR image has two bands [12]. That makes the input image stack to have  $1 + 4 \cdot 2 = 9$  channels (DEM has one channel and every SAR image has two bands/channels).

The acquisition conditions were composed from the following features:

1. Mean temperature of the acquisition date
2. Snow depth in the acquisition date
3. Satellite orbit direction during the acquisition (Ascending/Descending)
4. Incidence angle
5. Satellite id (Sentinel-1A or Sentinel-1B)
6. Precipitations amount in the acquisition date and three previous dates

All other features were scalar values from the acquisition date except for precipitation that is a vector with values for four different days. Since the moisture content of the soil has known effect to the signal, and moisture can linger long times in the soil, it was decided to include the precipitation amounts from multiple days to the acquisition conditions. Taking the precipitation amounts from the previous 4 days was a somewhat arbitrary decision with a reasoning that the neural network can learn to ignore the precipitation amounts from previous days if they have no use. The features were flattened to the final vector with dimensionality of  $|D| = 9$ .

## 2.4 Experiment Setup

The performance of the proposed method was measured using experimentation. The main contribution of this paper is to offer a new strategy for computing the difference image. Existing methods generally use a strategy where the difference image is computed using  $I_{DI} = g(I_{t-y}, I_t)$ , where the  $g$  is the differencing function,  $I_{t-y}$  is one of the previous images from the location captured at some previous date, and  $I_t$  is the most recent image from the location. The proposed method uses the neural network output  $\hat{I}_t$  in place of the  $I_{t-y}$  to compute the difference image  $\hat{I}_{DI} = g(\hat{I}_t, I_t)$ . The mapping transformation function factors in the imaging conditions of  $I_t$  when generating the  $\hat{I}_t$ , therefore the  $\hat{I}_{DI}$  should be higher quality when compared to  $I_{DI}$ . The difference image is generally further used in the change detection system to detect the changes by applying a classifier to the difference image. The classifier outputs a change map indicating the pixels that contain the detected changes. By using identical classifier to classify the difference images generated by the two different methods and comparing the classifying accuracy of the resulting change maps, the quality of the two difference images can be measured.

### 2.4.1 Change Simulation

The experiment needs a dataset with known changes so that the accuracy of the change detection classifier can be determined. This is a challenge since only a small number of datasets exists for remote sensing change detection even for optical satellite images [38]. For SAR images there are only few datasets such as the ones used in the following publications [39, 40], however they consist of only few SAR image pairs with a hand labelled change map. Currently there are no large enough SAR datasets for deep learning applications available online [41].

To avoid the problem with the lack of change detection datasets for SAR images, the decision was made to use simulation to add changes to real SAR images. This technique was used by Inglada and Mercier in [42] where they measured the performance of their statistical similarity measure change detection algorithm using simulated changes. The authors used three different methods for change simulation. The techniques were: *offset change*, where the original

value was shifted by a value; *Gaussian change*, where the original value was changed by adding zero mean Gaussian noise to the value; and *deterministic change*, where a value was copied from some other location in the image. Likewise, Cui et al. used change simulation for SAR images when they introduced an evaluation benchmark for SAR change detection algorithms [43]. The change simulation methods in the paper try to replicate changes that are commonly seen in the real world using techniques that correctly resemble the statistical properties of the real world changes. Based on these papers two change simulation methods were devised for this study.

1. *Offset change*: A value is added to the original pixel value. The simulation does not try to replicate any real world change, however it is trivial to implement, and the offset value can be changed to test different offsets.
2. *First-order statistical change*: The statistical distribution of the change area is converted to the statistical distribution of some other nearby geographical feature. This replicates the real world changes more accurately.

Figure 5 illustrates the simulated change methods when applied to an example SAR image. The changes were added to the SAR images by creating a random shape mask and positioning the mask to a random location in the SAR image. The pixel values inside the mask were changed using the selected method. The location of the mask was restricted to forested geographical areas in the SAR image. If the mask location was at forest edge, the mask part that landed outside of the forested area was not changed. The information about different geographical features was acquired from the NLS Topographic Database [44]. The database was also utilized in first-order statistical change implementation where the forest area pixel values were changed to follow the statistical distribution of some other geographical feature. The nearest areas of the desired geographical feature type were queried from the database, and the statistical distribution of the pixel values was estimated using a univariate kernel density estimator (KDE) from the statsmodels Python library [45]. A second univariate KDE model was fitted to the pixel values of all forested area pixels in the SAR image. The mapping of the pixel values was implemented using the method of modifying first-order statistical distribution described in [43]. The change area pixel values were first mapped to uniform distribution in the interval  $[0, 1]$  by using the cumulative distribution function (cdf) of the forest area KDE. After that, the inverse cdf of the second KDE model is applied to the uniformly distributed values, thus mapping them to the distribution of the desired geographical feature.

## 2.4.2 Change Classifier

The quality of the difference images was measured using two different classifiers. The first method is a simple threshold method. A thresholding value is chosen, and the pixels are classified to changed or unchanged based on if the value is smaller or greater than the threshold. This requires that the pixels have scalar values. The scalar valued difference images were produced using the following equations:

$$\hat{I}_{DI}(x, y) = \sqrt{\sum_b (I_t(x, y, b) - \hat{I}_t(x, y, b))^2} \quad (1)$$

$$I_{DI}(x, y) = \sqrt{\sum_b (I_t(x, y, b) - I_{t-y}(x, y, b))^2} \quad (2)$$

In the equations,  $\hat{I}_{DI}$  is the difference image that is computed using the proposed method,  $I_{DI}$  is the difference image that is computed using the conventional method,  $b$  is the band, and the  $x$  and  $y$  define the pixel location. The different bands are considered as vector dimensions. Pythagorean theorem is used to compute the vector length that is used as the value for the difference image pixel. The threshold method was used as an example of an unsupervised classifier algorithm [41]. The performance of the threshold classifiers was measured using the well known area under curve (AUC) metric that is computed from the receiver operating characteristic (ROC) curve. The metrics were computed to the test partition of the neural network mapping function dataset. The  $\hat{I}_{DI}$  and  $I_{DI}$  difference images were computed for every sample in the test dataset, and the pixels from all samples were used to generate the two datasets that were used to compute the ROC curves and AUC metrics.

The second classifier was the linear support vector classifier (SVC). The support vector classifier was used as an example of supervised machine learning algorithm. The support vector models work with multidimensional data, therefore the difference images were produced using simple subtraction:

$$\hat{I}_{DI}(x, y, b) = I_t(x, y, b) - \hat{I}_t(x, y, b) \quad (3)$$

$$I_{DI}(x, y, b) = I_t(x, y, b) - I_{t-y}(x, y, b) \quad (4)$$



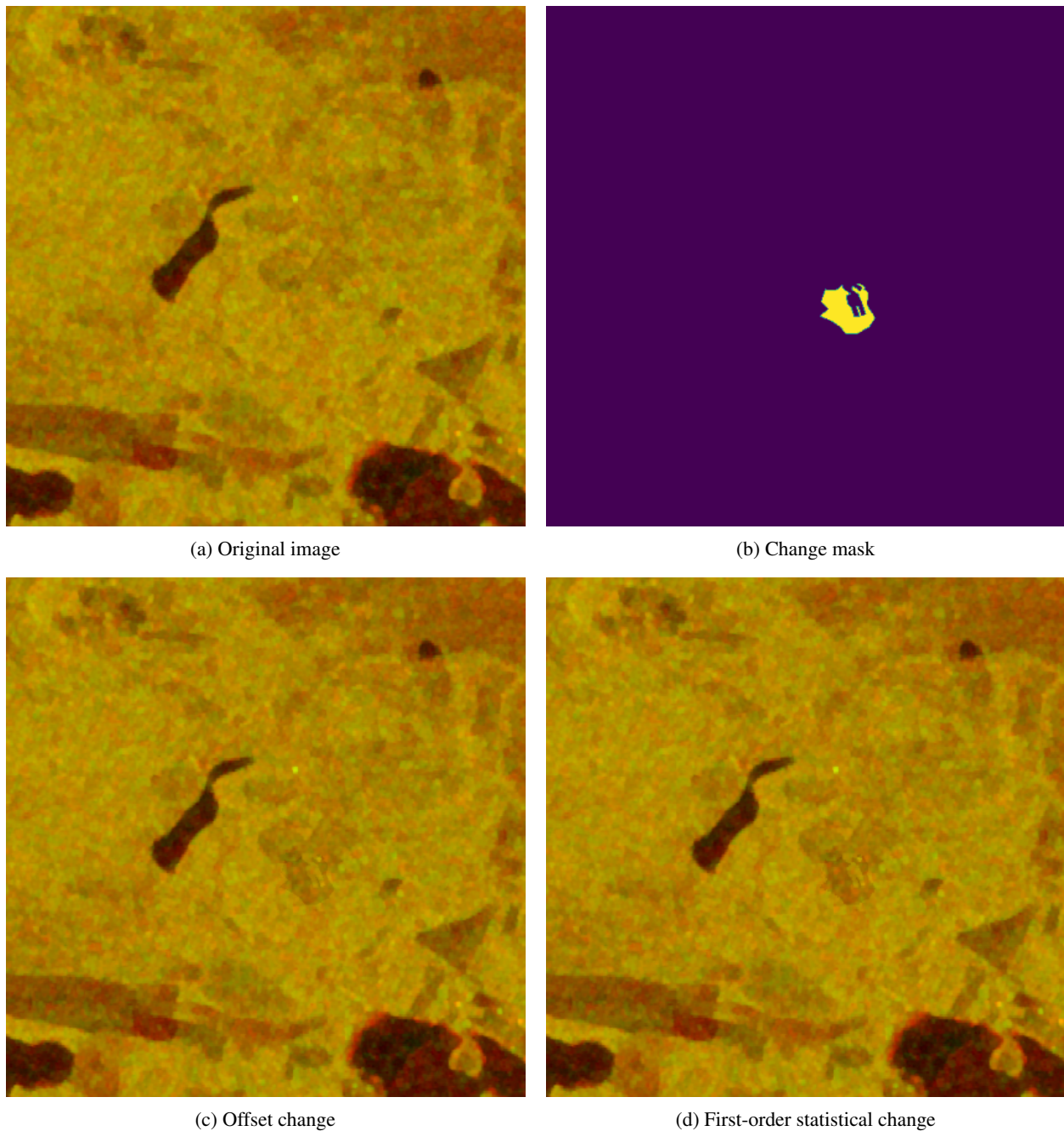


Figure 5: Example of the two simulated change methods. The SAR images are visualized as RGB image by using red and green channels for the two bands. The blue channel is set to zero. The offset change is  $-2.5$  dB in the image c, that is close to the mean change introduced by the first-order statistical change method in the image d.

Table 1: Parameters for the neural network architecture. The parameters configure the convolutional layers in the upsampler and downsampler blocks seen in the architectural diagram Figure 3.

Block number	Downsampler (filter size, kernel size)	Upsampler (filter size, kernel size)
1	64, 4	512, 4
2	128, 4	512, 4
3	256, 4	512, 4
4	512, 4	512, 4
5	512, 4	512, 4
6	512, 4	512, 4
7	512, 4	256, 4
8	512, 4	128, 4
9	512, 2	2, 4

The test dataset from the mapping transformation function training was used to train the classifiers. For each sample, the two difference images were computed, and the pixels from all difference image samples were used to create the two datasets. The first dataset was generated using the pixels from the  $\hat{I}_{DI}$  samples, and the second dataset was generated using the pixels from the  $I_{DI}$  samples. The two datasets were further divided to train and test datasets with a rule that all pixels originating from one image sample end up in the same side of the split. The train test split was also identical for both datasets. The datasets were used to train two instances of the classifier and measure their accuracy.

### 3 Results

#### 3.1 Training the Neural Network-Based Mapping Transformation Function

Different neural network parameters were experimented with, and the best results were achieved with the parameters shown in the Table 1. Mean squared error was used as the loss function, and AdamW [46] was used as the optimizer. The final training dataset had around 230,000 samples, and the training was monitored with a test dataset of around 9,000 samples. The neural network architecture was implemented using TensorFlow deep learning framework [47]. The training was conducted on one NVIDIA V100 GPU with batch size of 200, and training time of around 30 hours.

Figure 6c demonstrates the model performance for one of the test samples. Figure 6a shows the real SAR image that the model tries to predict. Figure 6b illustrates the difference between the real SAR image and the model output with a heat map where lighter color indicates a greater error. The predicted image is very close to the real SAR image except for lack of noise that is purely random and impossible for the model to predict. Likewise, the lower right corner of the image has an area that has greater error in the prediction. The error is located in a lake, therefore the error can be a result of waves that are likewise impossible to predict.

The proposed method depends on that the mapping transformation function adapts the predicted image  $\hat{I}_t$  based on the imaging conditions of  $I_t$ . To verify that the model genuinely uses the image acquisition conditions to produce the  $\hat{I}_t$ , the model was experimented to produce outputs with manually modified imaging condition vector  $D_t$ . Figure 6d and Figure 6f image pair illustrates model outputs where the  $D_t$  is modified to have opposite orbit directions. Figure 6e illustrates the difference between the images. The lake banks and the upper left corner of the image, where there is a small hill, have large differences between the two generated images. All locations, where there are greater differences between the images, are 3D features. The Sentinel-1 satellites have different look directions on ascending and descending orbit directions. Therefore, the scattering of the radar signal is different and the difference is most noticeable on 3D features. Since the differences are so clearly located on the 3D features in the image the model is clearly factored in the orbit direction when generating the output. This verifies that the imaging conditions are used by the model to produce the  $\hat{I}_t$  in the imaging conditions of  $I_t$ .

The same experiment was conducted by modifying the precipitation amounts in Figure 6g and Figure 6i. The difference between the generated images is shown in the Figure 6h. This time the difference between the generated images is focused on swamp, meadow, and agricultural land areas in the image. The forest areas have only small differences between the images. In forest areas, the radar signal is scattered back by the forest foliage where the moisture does not affect the scattering properties as much as the open areas. In open areas, the radar signal hits the ground where the soil moisture content is altered more by the rain, thus changing the backscatter intensity. This experiment suggests that the model uses the precipitation information correctly when generating the output image.

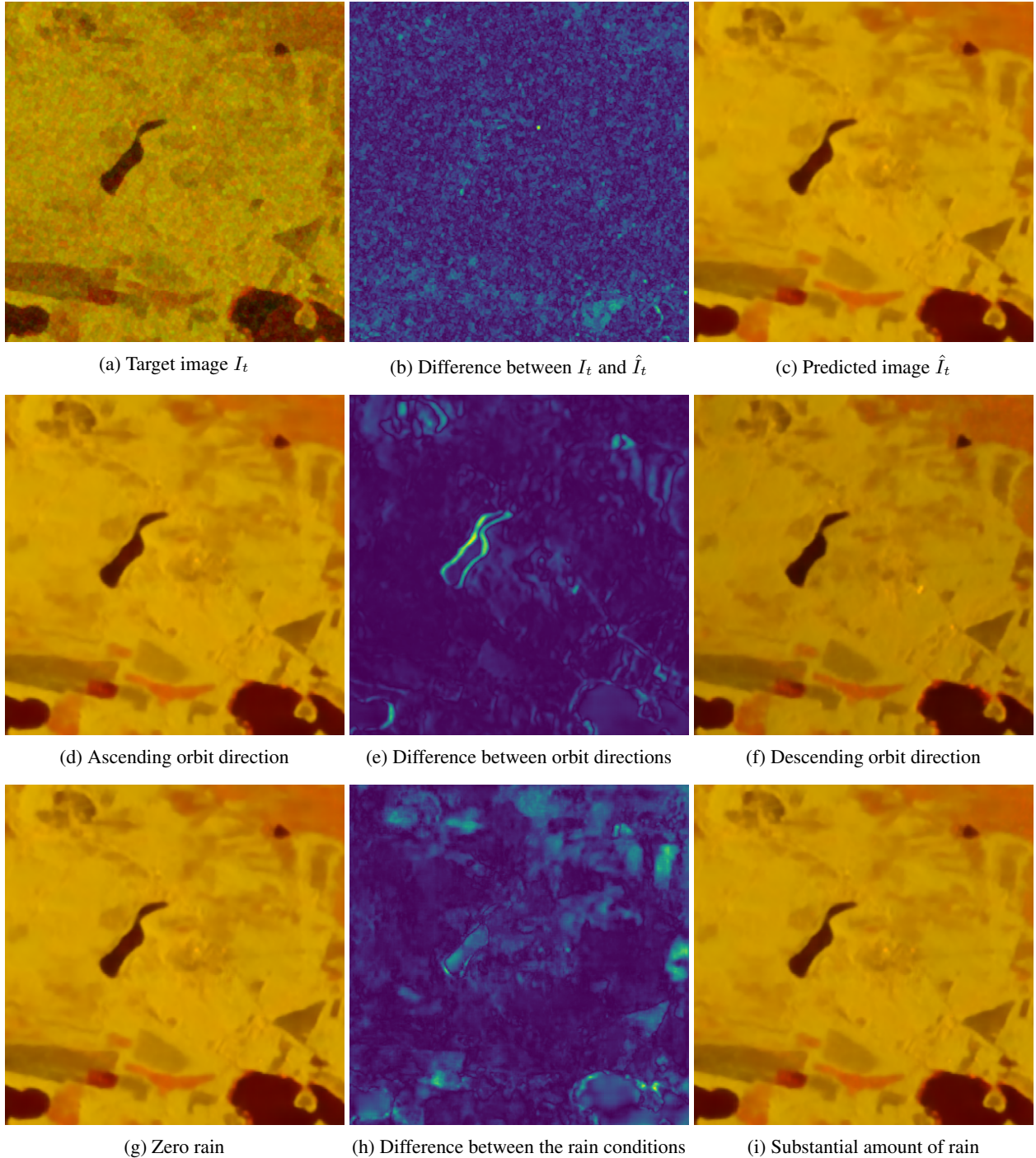


Figure 6: Mapping transformation function outputs with different imaging conditions. The image a is the original SAR image that is captured from coordinates 64.919 lat, 28.124 lon in 7th of July 2021. The image c shows the model output  $\hat{I}_t$  when it is trying to predict the  $I_t$ . Image b shows the difference between the true image  $I_t$  and the predicted image  $\hat{I}_t$ . The Images d, f, g and i are generated by manually modifying the imaging condition vector  $D_t$ . Image d has ascending and e has descending orbit direction. Image e shows the difference between the different orbit direction images. Identical experiment was conducted by varying the precipitation amount in images g and i. Image h shows the difference between the images with the different precipitation amounts.

### 3.2 Identifying the Best Conventional DI Strategy

The conventional method of computing the difference image is to use one of the previous SAR images that is captured at some preceding date with the most recent image to produce the difference image  $I_{DI} = g(I_{t-y}, I_t)$ . There are multiple different strategies when selecting the previous image. The simplest strategy is to select the previous image that is preceding the image that was captured most recently. This strategy has the advantage that the least amount of time has elapsed between the images, therefore the number of natural changes, like foliage growth or soil moisture changes, are minimized. However, the problem is that the previous image has very likely different incidence angle and it might have been captured from different orbit direction (ascending/descending). To make sure that we compare the proposed method to the best conventional method, three different previous image selection strategies were compared to identify the best strategy. The threshold classifier was used to compare the quality of the difference images that were produced using the different strategies. The strategies have different trade offs between the elapsed time and imaging angle:

Method 1: Closest incidence angle and the same orbit direction.

Method 2: Most recent previous image with the same orbit direction.

Method 3: Most recent previous image preceding the target image ( $I_{t-1}$ ).

Figure 7 illustrates the comparison of the three different methods using ROC curve plots. The strategy where the previous image is captured from the same orbit direction and has the closest incidence angle with the  $I_t$  is the best  $I_{t-y}$  selection strategy. From this on forward, the Method 1 is always used when referring to the conventional method of computing difference image.

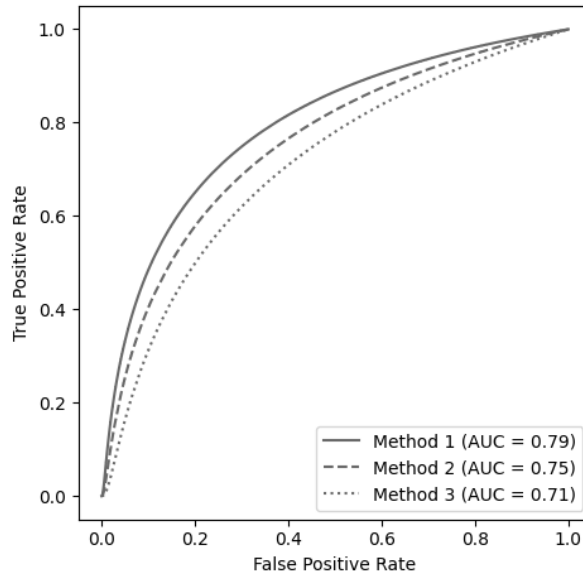


Figure 7: Comparison of different previous image selection strategies when using the traditional method of computing the difference image.

### 3.3 Proposed Method vs. Conventional Method

#### 3.3.1 Threshold Classifier

Figure 8 illustrates the ROC curve plots for the two threshold classifiers when measuring the quality of the difference images generated with the two methods. In this experiment, the changes are simulated to the dataset using the offset change method. The simulated shift is  $-2.5$  dB in the change area, which represents a considerable change. In the real world, this could be a change where the forest is clear cut, making it smoother, and that way reducing the backscatter intensity. The threshold classifier that is using the difference images that are produced using the proposed method is clearly better. This indicates that the proposed method generates better quality difference images.

Figure 9 illustrates the results of the same experiment when it is repeated to the simulated change dataset using the statistical change method. The change areas are simulated to emit the backscatter intensity of nearby forest areas

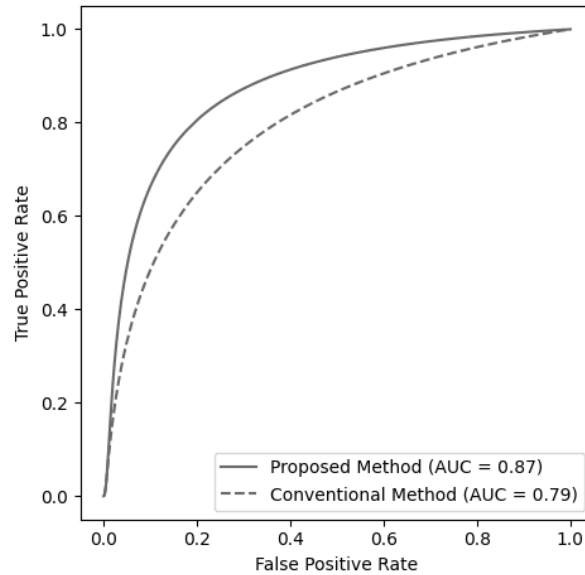


Figure 8: ROC curve for the two threshold classifiers when applied to the dataset with simulated changes using the offset change method.

that are not as densely wooded making this more realistic representation of real changes in the forest. The mean backscatter intensity change varied from around  $-0.5$  dB to  $-2.5$  dB in the change areas depending on the sample. Both classifiers have considerably worse performance, however the proposed method is still better performing. The overall poor performance is to be expected with the threshold classifiers. It is the simplest possible classifier working in single pixel level without having any kind of visibility to the neighbouring pixels. Furthermore, the changes can be small in the simulated change dataset that is created using the statistical change method.

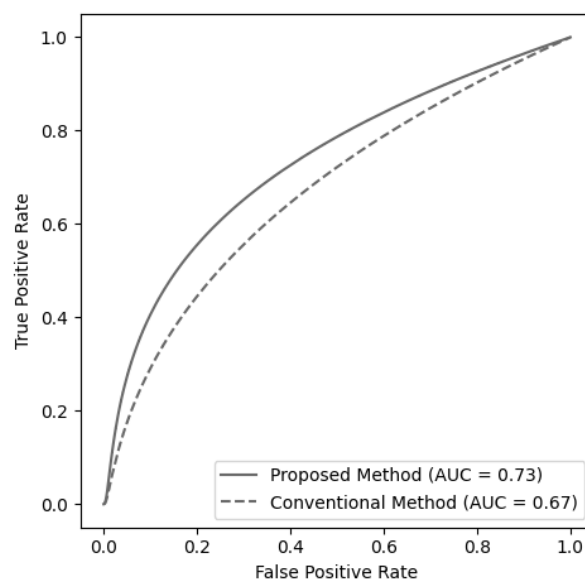


Figure 9: ROC curve for the two threshold classifiers when applied to the dataset with simulated statistical changes.

Table 2: Experiment results for the SVC models.

Dataset	Proposed method accuracy	Conventional method accuracy
Shift change	0.89	0.81
Statistical change	0.75	0.70

### 3.3.2 Support Vector Classifier

The experiments were repeated with the SVC model to the same two datasets. The linear kernel SVC implementation `LinearSVC` from Scikit-learn library [48] was used to conduct the experiment. Linear kernel SVC was chosen due to large dataset size. Other kernel types were tested, however they did not scale to the large number of samples. The samples were normalized using the Scikit-learn `StandardScaler` to ease the model convergence. Table 2 displays the results from the experiments. The proposed method is clearly superior to the conventional method in both experiments. The performance in the statistical change dataset is considerably worse when compared to the shift change dataset. However, this is to be expected with the similar loss of accuracy in the threshold classifier experiments. This experiment uses supervised learning with labeled dataset which should improve the results when comparing to the threshold classifier. However the SVC is still very simple classifier that performs the classification at pixel level without any visibility to the neighbouring pixels, thus the accuracy scores are mediocre at best. Still, achieving high accuracy score was not the goal of the experiment. Instead, the experiment is comparing the accuracies of the two classifiers and the results from this experiment support the findings from the threshold classifier experiments. The proposed method clearly produces higher quality difference images.

### 3.3.3 Model Without the Weather Data

The dataset creation for this project was a major undertaking which complicates the adaption of the proposed methodology since the model needs to be trained to every location where it is used. Finnish Meteorological Institute provides the interpolated weather data for the features we used in this study that are available in locations inside the borders of Finland. However, equivalent data sources are not necessary available in other countries. Therefore, we experimented how the neural network based mapping transformation function works without the weather data. The model training pipeline was modified to drop the weather data during training and inference, thus the acquisition conditions consisted only from incidence angle, satellite orbit direction, and satellite id. Figure 10 illustrates the results from the experiment. The experiment used simulated changes with  $-2.5$  dB shift and exact same model hyper parameters with the results that are illustrated in Figure 8, thus the result is directly comparable. The resulting AUC metric is higher at 0.83 when comparing to the conventional method at 0.79, however the result is worse when comparing to the model that has visibility to the weather data with AUC metric of 0.87. Therefore, we can conclude that the proposed methodology can be used also without weather data, and it achieves measurable improvement over conventional method. However, to achieve the best performance, the model requires the weather data in addition of the other imaging condition features.

## 4 Discussion

The experiment results show that the proposed method produces higher quality difference images than the conventional method. Since the output from the proposed method is a difference image, many of the existing change classification techniques may benefit from the method without any modifications. The techniques generally use the conventional method for producing the difference image, however it is completely separate step from the classification, and thus could be replaced with the proposed method without changes to the classification step. Some methods do not use the difference image computation step, instead they accept the two images directly to the model to carry out the classification. Even with these techniques the usage of the proposed method could be beneficial. In these cases, the earlier image ( $I_{t-y}$ ) is replaced with the  $\hat{I}_t$ , thus giving the classification model better understanding about what the scene should look like in the correct image acquisitions conditions.

This study did not experiment with the more advanced change detection classifiers since the simple classifiers were enough to prove that the proposed method is better than the conventional method. However, the clear improvement in classification accuracy with the simple methods could indicate that similar improvement can be achieved with the more advanced methods.

The use of simulated changes to measure the performance of the method was a necessary compromise caused by the lack of existing change detection datasets suitable for training the neural network. The simulated changes are not realistic enough to draw a final conclusion about how much the proposed method would improve the change detection

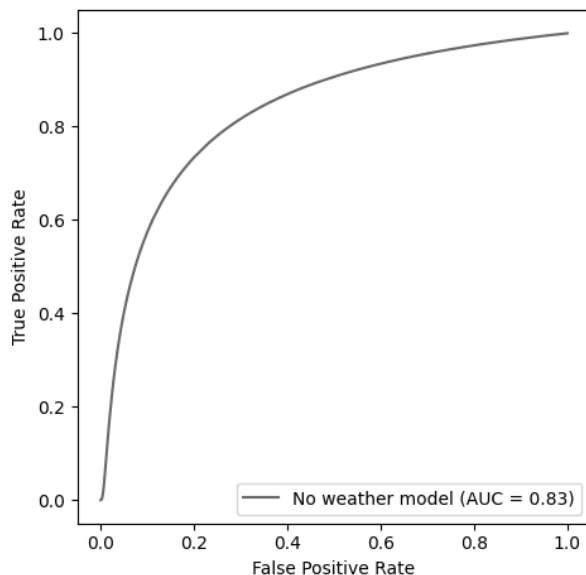


Figure 10: Threshold classifier ROC when used with mapping transformation function that is trained without the weather data.

performance in real world application. However, the experiments with the simulated changes indicate a substantial performance improvement potential.

The downside of the proposed method is that the mapping transformation function is a neural network model that requires a training dataset and considerable amount of processing power for training. The dataset creation is a complex operation that combines data from multiple data sources. Some of the sources that were used in this study are available only for geographical locations inside Finland, such as the interpolated weather data from Finnish Meteorological Institute. The model requires training data from the locations it is used at inference time which complicates the adaption of the method outside of Finland. However, many of the data sources very likely have equivalents available in other geographical locations, therefore the adoption is not impossible. Even a global training dataset could potentially be constructed, which could make the training of a universal model possible. The recent advances in neural network architectures with natural language processing and image generation have shown that the models can learn from impressive amounts of data. The model training is unsupervised, meaning it does not require labelled data, thus the creation of such a dataset could be possible. Our experiment with a model that did not see the weather data in the input shows that the method achieves measurable improvement over the conventional method even when the model has information only about the imaging angle and the satellite. That data is available in the SAR images when they are downloaded from the ESA open access portal, thus simplifying the dataset creation considerably. However, without the weather data the mapping transformation function cannot generate accurate enough SAR images to achieve same accuracy metrics that the model with the weather information achieves.

## Funding

This research was conducted in the Jamk University of Applied Sciences with funding from two projects. *Data for Utilisation – Leveraging digitalisation through modern artificial intelligence solutions and cybersecurity* project which is funded by the Regional Council of Central Finland/Council of Tampere Region and European Regional Development Fund. And *coADDVA - ADDing Value by Computing in Manufacturing* project which is funded by REACT-EU Instrument as part of the European Union’s response to the COVID-19 pandemic.

## Data Availability

The Sentinel-1 SAR imagery is available to download free of charge from Copernicus Open Access Hub [27]. The weather data is available to download free of charge from Finnish Meteorological Institute [35]. The digital elevation map and topographic database are available to download free of charge from the National Land Survey of

Finland open data file download service [34, 44]. Links to the download sites are available in the references. The derived dataset that was used to train the neural network and supports the findings of this study can be downloaded from the Fairdata.fi service [49]. The computer code that was used to produce the results is available at <https://github.com/janne-alatalo/sar-change-detection>.

## Acknowledgments

This work uses modified Copernicus Sentinel data 2020-2021. The authors would like to thank Mr. Eppu Heilimo for feedback on the draft manuscript. The authors wish to acknowledge CSC - IT Center for Science, Finland, for hosting the dataset.

## References

- [1] Jérémie Sublime and Ekaterina Kalinicheva. “Automatic Post-Disaster Damage Mapping Using Deep-Learning Techniques for Change Detection: Case Study of the Tohoku Tsunami”. In: *Remote Sensing* 11.9 (2019). ISSN: 2072-4292. DOI: 10.3390/rs11091123. URL: <https://www.mdpi.com/2072-4292/11/9/1123>.
- [2] Laigen Dong and Jie Shan. “A comprehensive review of earthquake-induced building damage detection with remote sensing techniques”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 84 (2013), pp. 85–99. ISSN: 0924-2716. DOI: 10.1016/j.isprsjprs.2013.06.011. URL: <https://www.sciencedirect.com/science/article/pii/S0924271613001627>.
- [3] Stephanie Long, Temilola E Fatoyinbo, and Frederick Policelli. “Flood extent mapping for Namibia using change detection and thresholding with SAR”. In: *Environmental Research Letters* 9.3 (Mar. 2014), p. 035002. DOI: 10.1088/1748-9326/9/3/035002.
- [4] Erkki Tomppo et al. “Detection of Forest Windstorm Damages with Multitemporal SAR Data—A Case Study: Finland”. In: *Remote Sensing* 13.3 (2021). ISSN: 2072-4292. DOI: 10.3390/rs13030383. URL: <https://www.mdpi.com/2072-4292/13/3/383>.
- [5] Marius Rüetschi, David Small, and Lars T. Waser. “Rapid Detection of Windthrows Using Sentinel-1 C-Band SAR Data”. In: *Remote Sensing* 11.2 (2019). ISSN: 2072-4292. DOI: 10.3390/rs11020115. URL: <https://www.mdpi.com/2072-4292/11/2/115>.
- [6] Pablo Pozzobon de Bem et al. “Change Detection of Deforestation in the Brazilian Amazon Using Landsat Data and Convolutional Neural Networks”. In: *Remote Sensing* 12.6 (2020). ISSN: 2072-4292. DOI: 10.3390/rs12060901. URL: <https://www.mdpi.com/2072-4292/12/6/901>.
- [7] Vishakha Sood et al. “Detection of snow/ice cover changes using subpixel-based change detection approach over Chhota-Shigri glacier, Western Himalaya, India”. In: *Quaternary International* 575-576 (2021). SI: Remote Sensing and GIS Applications in Quaternary Sciences, pp. 204–212. ISSN: 1040-6182. DOI: 10.1016/j.quaint.2020.05.016. URL: <https://www.sciencedirect.com/science/article/pii/S1040618220302494>.
- [8] D. Lu et al. “Change detection techniques”. In: *International Journal of Remote Sensing* 25.12 (2004), pp. 2365–2401. DOI: 10.1080/0143116031000139863.
- [9] Wenxue Fu et al. “Remote Sensing Satellites for Digital Earth”. In: *Manual of Digital Earth*. Ed. by Huadong Guo, Michael F. Goodchild, and Alessandro Annoni. Singapore: Springer Singapore, 2020, pp. 55–123. ISBN: 978-981-32-9915-3. DOI: 10.1007/978-981-32-9915-3\_3.
- [10] Adriana Grazia Castriotta. *Copernicus Sentinel Data Access Annual Report Y2021*. Annual Report. Available online: [https://scihub.copernicus.eu/twiki/pub/SciHubWebPortal/AnnualReport2021/COPE-SERCO-RP-22-1312\\_-\\_Sentinel\\_Data\\_Access\\_Annual\\_Report\\_Y2021\\_merged\\_v1.1.pdf](https://scihub.copernicus.eu/twiki/pub/SciHubWebPortal/AnnualReport2021/COPE-SERCO-RP-22-1312_-_Sentinel_Data_Access_Annual_Report_Y2021_merged_v1.1.pdf). 2022.
- [11] Alberto Moreira et al. “A tutorial on synthetic aperture radar”. In: *IEEE Geoscience and Remote Sensing Magazine* 1.1 (2013), pp. 6–43. DOI: 10.1109/MGRS.2013.2248301.
- [12] European Space Agency. *Sentinel-1 SAR Technical Guide - Interferometric Wide Swath*. Available online: <https://sentinels.copernicus.eu/web/sentinel/user-guides/sentinel-1-sar/acquisition-modes/interferometric-wide-swath>. (accessed on 7 February 2023).
- [13] Maoguo Gong et al. “SAR change detection based on intensity and texture changes”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 93 (2014), pp. 123–135. ISSN: 0924-2716. DOI: 10.1016/j.isprsjprs.2014.04.010. URL: <https://www.sciencedirect.com/science/article/pii/S0924271614001051>.
- [14] Huming Zhu et al. “Parallel unsupervised Synthetic Aperture Radar image change detection on a graphics processing unit”. In: *The International Journal of High Performance Computing Applications* 27.2 (2013), pp. 109–122. DOI: 10.1177/1094342013476120.



- [15] Wenhua Zhang et al. “Sparse Feature Clustering Network for Unsupervised SAR Image Change Detection”. In: *IEEE Transactions on Geoscience and Remote Sensing* 60 (2022), pp. 1–13. DOI: 10.1109/TGRS.2022.3167745.
- [16] Yangyang Li et al. “A Deep Learning Method for Change Detection in Synthetic Aperture Radar Images”. In: *IEEE Transactions on Geoscience and Remote Sensing* 57.8 (2019), pp. 5751–5763. DOI: 10.1109/TGRS.2019.2901945.
- [17] L. Bruzzone and D.F. Prieto. “An adaptive semiparametric and context-based approach to unsupervised change detection in multitemporal remote-sensing images”. In: *IEEE Transactions on Image Processing* 11.4 (2002), pp. 452–466. DOI: 10.1109/TIP.2002.999678.
- [18] Y. Gauthier, M. Bernier, and J.-P. Fortin. “Aspect and incidence angle sensitivity in ERS-1 SAR data”. In: *International Journal of Remote Sensing* 19.10 (1998), pp. 2001–2006. DOI: 10.1080/014311698215117.
- [19] Hari Shanker Srivastava et al. “Large-Area Soil Moisture Estimation Using Multi-Incidence-Angle RADARSAT-1 SAR Data”. In: *IEEE Transactions on Geoscience and Remote Sensing* 47.8 (2009), pp. 2528–2535. DOI: 10.1109/TGRS.2009.2018448.
- [20] R. Touzi. “A review of speckle filtering in the context of estimation theory”. In: *IEEE Transactions on Geoscience and Remote Sensing* 40.11 (2002), pp. 2392–2404. DOI: 10.1109/TGRS.2002.803727.
- [21] Fabrizio Argenti et al. “A Tutorial on Speckle Reduction in Synthetic Aperture Radar Images”. In: *IEEE Geoscience and Remote Sensing Magazine* 1.3 (2013), pp. 6–35. DOI: 10.1109/MGRS.2013.2277512.
- [22] Xiao Chen et al. “Incidence Angle Normalization of Dual-Polarized Sentinel-1 Backscatter Data on Greenland Ice Sheet”. In: *Remote Sensing* 14.21 (2022). ISSN: 2072-4292. DOI: 10.3390/rs14215534. URL: <https://www.mdpi.com/2072-4292/14/21/5534>.
- [23] Gregoriy Kaplan et al. “Normalizing the Local Incidence Angle in Sentinel-1 Imagery to Improve Leaf Area Index, Vegetation Height, and Crop Coefficient Estimations”. In: *Land* 10.7 (2021). ISSN: 2073-445X. DOI: 10.3390/land10070680. URL: <https://www.mdpi.com/2073-445X/10/7/680>.
- [24] Huifu Zhuang et al. “It is a misunderstanding that log ratio outperforms ratio in change detection of SAR images”. In: *European Journal of Remote Sensing* 52.1 (2019), pp. 484–492. DOI: 10.1080/22797254.2019.1653226.
- [25] Oscar Mora et al. “Earthquake Rapid Mapping Using Ascending and Descending Sentinel-1 TOPSAR Interferograms”. In: *Procedia Computer Science* 100 (2016). International Conference on ENTERprise Information Systems/International Conference on Project MANagement/International Conference on Health and Social Care Information Systems and Technologies, CENTERIS/ProjMAN / HCist 2016, pp. 1135–1140. ISSN: 1877-0509. DOI: 10.1016/j.procs.2016.09.266. URL: <https://www.sciencedirect.com/science/article/pii/S1877050916324358>.
- [26] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Ed. by Nassir Navab et al. Cham: Springer International Publishing, 2015, pp. 234–241. ISBN: 978-3-319-24574-4. DOI: 10.1007/978-3-319-24574-4\_28.
- [27] European Space Agency. *Copernicus Sentinel data 2020-2021*. Available for download: <https://scihub.copernicus.eu/>.
- [28] European Space Agency. *Sentinel-1 SAR Technical Guide - Level-1 GRD Products*. Available online: <https://sentinels.copernicus.eu/web/sentinel/technical-guides/sentinel-1-sar/products-algorithms/level-1-algorithms/ground-range-detected>. (accessed on 26 January 2023).
- [29] European Space Agency. *SNAP - ESA sentinel Application Platform v8.0.0*. <https://step.esa.int/main/toolboxes/snap/>.
- [30] Federico Filipponi. “Sentinel-1 GRD Preprocessing Workflow”. In: *Proceedings* 18.1 (2019). ISSN: 2504-3900. DOI: 10.3390/ECRS-3-06201. URL: <https://www.mdpi.com/2504-3900/18/1/11>.
- [31] *Copernicus POD Service File Format Specification*. Version 2.0. Available online: [https://sentinel.esa.int/documents/247904/351187/Copernicus\\_Sentinels\\_POD\\_Service\\_File\\_Format\\_Specification](https://sentinel.esa.int/documents/247904/351187/Copernicus_Sentinels_POD_Service_File_Format_Specification). GMV, 2022.
- [32] National Land Survey of Finland. *Division into administrative areas (vector)*. Available online: <https://www.maanmittauslaitos.fi/en/maps-and-spatial-data/professionals/product-descriptions/division-administrative-areas-vector>. (accessed on 31 January 2023). Data available for download: <https://asiointi.maanmittauslaitos.fi/karttapaikka/tiedostopalvelu?lang=en>.
- [33] European Commission, Eurostat (ESTAT), GISCO. *Countries 2020, 2020 - Administrative Units - Dataset*. Available online: <https://ec.europa.eu/eurostat/web/gisco/geodata/reference-data/administrative-units-statistical-units/countries>. (accessed on 31 January 2023).

- [34] National Land Survey of Finland. *Elevation model 2m*. Available online: <https://www.maanmittauslaitos.fi/en/maps-and-spatial-data/expert-users/product-descriptions/elevation-model-2-m>. (accessed on 24 January 2023). Data available for download: <https://asiointi.maanmittauslaitos.fi/karttapaikka/tiedostopalvelu?lang=en>.
- [35] Finnish Meteorological Institute. *Daily observations in 1km\*1km grid*. Available online: <https://en.ilmatieteenlaitos.fi/gridded-observations-on-aws-s3>. (accessed on 24 January 2023). Data available for download: <http://fmi-gridded-obs-daily-1km.s3-website-eu-west-1.amazonaws.com/>.
- [36] Juha Aalto, Pentti Pirinen, and Kirsti Jylhä. “New gridded daily climatology of Finland: Permutation-based uncertainty estimates and temporal trends in climate”. In: *Journal of Geophysical Research: Atmospheres* 121.8 (2016), pp. 3807–3823. DOI: 10.1002/2015JD024651. URL: <https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2015JD024651>.
- [37] GDAL/OGR contributors. *GDAL/OGR Geospatial Data Abstraction software Library*. Open Source Geospatial Foundation. 2022. DOI: 10.5281/zenodo.5884351. URL: <https://gdal.org>.
- [38] Wenzhong Shi et al. “Change Detection Based on Artificial Intelligence: State-of-the-Art and Challenges”. In: *Remote Sensing* 12.10 (2020). ISSN: 2072-4292. DOI: 10.3390/rs12101688. URL: <https://www.mdpi.com/2072-4292/12/10/1688>.
- [39] Luigi Tommaso Luppino et al. “Unsupervised Image Regression for Heterogeneous Change Detection”. In: *IEEE Transactions on Geoscience and Remote Sensing* 57.12 (2019), pp. 9960–9975. DOI: 10.1109/TGRS.2019.2930348.
- [40] Junjie Wang et al. “Change Detection From Synthetic Aperture Radar Images via Graph-Based Knowledge Supplement Network”. In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 15 (2022), pp. 1823–1836. DOI: 10.1109/JSTARS.2022.3146167.
- [41] Yu Du et al. “TransUNet++SAR: Change Detection with Deep Learning about Architectural Ensemble in SAR Images”. In: *Remote Sensing* 15.1 (2023). ISSN: 2072-4292. DOI: 10.3390/rs15010006. URL: <https://www.mdpi.com/2072-4292/15/1/6>.
- [42] Jordi Inglada and Grgoire Mercier. “A New Statistical Similarity Measure for Change Detection in Multitemporal SAR Images and Its Extension to Multiscale Change Analysis”. In: *IEEE Transactions on Geoscience and Remote Sensing* 45.5 (2007), pp. 1432–1445. DOI: 10.1109/TGRS.2007.893568.
- [43] Shiyong Cui, Gottfried Schwarz, and Mihai Datcu. “A Benchmark Evaluation of Similarity Measures for Multitemporal SAR Image Change Detection”. In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 9.3 (2016), pp. 1101–1118. DOI: 10.1109/JSTARS.2015.2486038.
- [44] National Land Survey of Finland. *Topographic Database*. Available online: <https://www.maanmittauslaitos.fi/en/maps-and-spatial-data/expert-users/product-descriptions/topographic-database>. (accessed on 27 January 2023). Data available for download: <https://asiointi.maanmittauslaitos.fi/karttapaikka/tiedostopalvelu?lang=en>.
- [45] Skipper Seabold and Josef Perktold. “statsmodels: Econometric and statistical modeling with python”. In: *9th Python in Science Conference*. 2010.
- [46] Ilya Loshchilov and Frank Hutter. *Decoupled Weight Decay Regularization*. 2017. DOI: 10.48550/ARXIV.1711.05101. URL: <https://arxiv.org/abs/1711.05101>.
- [47] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: <https://www.tensorflow.org/>.
- [48] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.
- [49] JAMK University of Applied Sciences. *Simulated Change Detection Dataset for Neural Networks*. <https://doi.org/10.23729/7b22c271-5e25-40fe-aa6a-f5d0330b1872>. JAMK University of Applied Sciences, School of Technology. 2023.