Unsupervised Satellite Image Time Series Analysis using Deep Learning Techniques

Ekaterina Kalinicheva

DOCTORAL THESIS OF
SORBONNE UNIVERSITY

Speciality

Computer Science

École doctorale Informatique, Télécommunications et Électronique (Paris)

Presented by

Ekaterina KALINICHEVA

To obtain the grade of

DOCTOR OF SORBONNE UNIVERSITY

Thesis subject :

Unsupervised Satellite Image Time Series Analysis using Deep Learning Techniques

defended on 30/09/2020

Jury :

Mme. Maria TROCAN, Full Professor, HDR	Thesis Director
M. Jérémie SUBLIME, Associate Professor	Supervisor
M. Germain FORESTIER, Full Professor	Reviewer
M. Basarab MATEI, Associate Professor, HDR	Reviewer
M. Dino IENCO, Research Scientist, HDR	Examiner
M. Cédric WEMMERT, Full Professor	Examiner
M. Christophe MARSALA, Full Professor	Examiner

THÈSE DE DOCTORAT DE
SORBONNE UNIVERSITÉ

Spécialité

Informatique

École doctorale Informatique, Télécommunications et Électronique (Paris)

Présentée par

Ekaterina KALINICHEVA

Pour obtenir le grade de

DOCTEUR DE SORBONNE UNIVERSITÉ

Sujet de la thèse :

Analyse Non-supervisée de Séries d'Images Satellites avec des Méthodes d'Apprentissage Profond

soutenue le 30 septembre 2020

devant le jury composé de :

Mme. Maria TROCAN, Professeur, HDR	Directrice de thèse
M. Jérémie SUBLIME, Enseignant-Chercheur	Encadrant
M. Germain FORESTIER, Professeur des Universités	Rapporteur
M. Basarab MATEI, Maître de Conférences, HDR	Rapporteur
M. Dino IENCO, Chargé de Recherche, HDR	Examinateur
M. Cédric WEMMERT, Professeur des Universités	Examinateur
M. Christophe MARSALA, Professeur des Universités	Examinateur

Dedication

To my bright future.

Acknowledgements

I would like to express my gratitude to all the people that helped me during these last years of doctoral studies, as well as all these without whom this work may not have been possible:

First of all, I would like to thank my thesis director Professor Maria Trocan and my supervisor Associate Professor Jérémie Sublime. This work would not have been possible without their recommendations, as well as their advises and insightful guidance. I would like to express my gratitude to my thesis director for helping me to participate to different conferences. I especially want to thank my supervisor for not stopping believing in me during these three years which helped me to make it through.

Credit must also be given to my colleague and co-author Doctor Dino Ienco who was my Master 1 internship supervisor in 2016 and introduced me to satellite imagery. I would also like to thank Dino for his priceless advices during my PhD, as well as for our scientific collaboration. Without him, none of my research achievements would be possible.

Then, I also wish to extend my gratitude to all the other people who were there for me these last three years:

- Associate Professor Patrick Wang with whom I worked when I was a teaching assistant.
- Associate Professors Rafael Angarita and Kevin Malléron with whom we shared memorable moments during work apéros.
- People with whom I shared the third floor ISEP office - Guillaume Dupont, the best intern ever, and a PhD student Arthur Vervaet with whom I spent long hours at the climbing gym and around some beers after. Also Doctor Denis Maurel who used to be in the third floor office and who forced me to move there.

– My best friend Alexandra Kazlova (aka Zhuchok, aka Greta) whom I have known for an eternity and who never stopped supporting me and motivated me to take most of the important life decisions : to move abroad, to do my Master in Montpellier and start this PhD, because I wanted to do the same things she did.

– My the most Parisian friend Alan Tran. He made my Paris PhD life a bit better by introducing me to the authentic bourgeois places, feeding me French cuisine, bringing to cultural events and just by being supportive when I needed it.

– My running buddies with whom I talked about life and science and shared wonderful moments during post-run Monday apéros and who supported me morally through the dark periods of these years.

– Other people who somehow appeared in my life and helped me to improve myself as a person and as a researcher and who made this world a better place for me.

– My family.

Finally, I want to thank the referees for dedicating their time to read my manuscript and for giving me their valuable opinions.

Abstract

This thesis presents a set of unsupervised algorithms for satellite image time series (SITS) analysis. Our methods exploit machine learning algorithms and, in particular, neural networks to detect different spatio-temporal entities and their eventual changes in the time. We distinguish three different types of temporal behavior that we aim to identify in our thesis: no change areas, seasonal changes (vegetation and other phenomena that have seasonal recurrence) and non-trivial changes (permanent changes such as constructions or demolishment, crop rotation, vegetation that do not follow the overall seasonal tendency present in SITS, etc). Therefore, we propose two frameworks: one for detection and clustering of non-trivial changes and another for clustering of "stable" areas of the series (seasonal changes and no change areas).

The change detection and analysis framework is composed of two essential steps which are bi-temporal change detection and the interpretation of detected changes in a multi-temporal context with graph-based approaches. The bi-temporal change detection algorithm is performed for each pair of consecutive images of the series and is based on feature translation with joint autoencoders. At the next step, the changes from different timestamps that belong to the same geographic area form evolution change graphs. These graphs are then clustered to identify different types of change behavior. For graph clustering, we exploit an AE model based on recurrent neural networks.

For the second framework, we propose an approach for object-based SITS clustering. First, we encode SITS with a multi-view 3D convolutional AE in a single image. Second, we perform the segmentation of the whole series that is composed of two steps (preliminary segmentation and its correction) and exploits both the encoded image and the original ones. Finally, the obtained segments are clustered using their encoded descriptors.

Table of Contents

List of Figures

List of Tables

Introduction

0.1 Motivations

Nowadays, different satellite missions provide numerous images of the Earth surface giving the knowledge about our planet that was not available before or was difficult to extract. Thanks to the remote sensing techniques, we can study almost any area of interest and obtain its full description in no time.

The increasing amount of freely available satellite image time series (SITS) has led to the creation of many projects exploiting data mining techniques. Among them, there are applications such as land use analysis, vegetation and water monitoring, analysis of the impact of disasters (e.g. floods and tsunamis), biodiversity analysis and monitoring, objects detection for military, urban development analysis, and many others.

Contrary to a single image analysis, multi-temporal SITS analysis gives us an additional information about an object's temporal behavior. This information is often indispensable for some applications, such as the detection and analysis of the evolution of a certain phenomena. In addition, the information about temporal behavior helps us distinguish more sub-classes, especially in the vegetation which can then be more easily divided into several classes based on their seasonal behavior. With such number of applications and the variety of satellite images, it is impossible to produce a unique database with labeled classes. It has motivated many researchers to develop different unsupervised data mining techniques that do not demand any labeled data. However, with the growing image quality and detail level, the classic image analysis methods have become outdated as they often cannot capture the complexity of the information contained in the satellite images.

With the increasing hardware capacity at a lower cost, deep learning techniques have gained their popularity in almost any research area. Neural network models were recently introduced in remote sensing field and have already outperformed most of the traditional approaches in therms of accuracy. Nevertheless, most of these models

are still supervised and very little algorithms are available for unsupervised analysis.

Moreover, existing satellite image analysis algorithms often aim to detect a particular event, without exploiting all the available information, though some applications demand the complete information about the study area. In this thesis, we focus on unsupervised deep learning approaches for satellite image analysis, especially on those applied to the multi-temporal data. This kind of data is the most complex one but, at the same time, is the most informative as it helps to better describe each detected phenomena. We aim to show the advantages and eventual disadvantages of unsupervised deep learning techniques applied to approaches such as satellite image time series (SITS) clustering and bi- and multi-temporal change detection. All algorithms were developed in *Python* language and do not demand any commercial software or training data. They were applied to freely available high resolution SITS to highlight the accessibility of the proposed approaches and their relevance for different applications.

0.2 Thesis Organization

This thesis is organized as follows:

In **Chapter 1**, we explain the basics of remote sensing data acquisition and interpretation and give an overview of different satellite missions. We equally present some basics of data mining applied to satellite images.

In **Chapter 2**, we present machine learning algorithms for different data analysis applications. We mostly focus on unsupervised algorithms that were used in our work, such as data clustering and anomaly detection.

Chapter 3 overviews unsupervised feature extraction methods used in satellite image analysis. A large part of this chapter is dedicated to deep learning feature extraction methods based on the autoencoder models that were used in our research.

Chapter 4 introduces our algorithm for bi-temporal change detection in SITS that is based on joint AEs feature translation and reconstruction. This algorithm is completed by an approach proposing the clustering of the detected change areas.

Chapter 5 describes our algorithm for multi-temporal change modeling and clustering. This approach presents the adaptation of the bi-temporal change detection and analysis method to the multi-temporal context by exploiting graph-based modeling techniques and Recurrent Neural Networks (RNN) for feature extraction and clustering of the obtained change graphs.

Chapter 6 features our approach for an object-based SITS clustering. In this

approach, we first exploit a two branches multi-view 3D convolutional AE to encode the time series. Second, we perform its segmentation and. Finally, we cluster the obtained spatio-temporal objects. The SITS segmentation is performed in several steps: we start by the preliminary segmentation of the two most representative images which usually does not give all the desired segments. Then we segment the encoded SITS which results in many parasite objects due to the encoding specificity. At the last step, we merge the results of both segmentation to obtain a unique segmentation map for the whole series.

Finally, **Chapter 7** concludes this thesis giving the overall review of the results obtained so far. It also presents some directions for the future research.

0.3 Publications

International Journals

- Jérémie Sublime, Ekaterina Kalinicheva, "Automatic Post-Disaster Damage Mapping Using Deep-Learning Techniques for Change Detection: Case Study of the Tohoku Tsunami," Remote. Sens. 11(9): 1123 (2019)
- Ekaterina Kalinicheva, Dino Ienco, Jérémie Sublime, Maria Trocan, "Unsupervised Change Detection Analysis in Satellite Image Time Series Using Deep Learning Combined With Graph-Based Approaches," IEEE J. Sel. Top. Appl. Earth Obs. Remote Sensing 13: 1450-1466 (2020)
- Ekaterina Kalinicheva, Jérémie Sublime, Maria Trocan, "Unsupervised Satellite Image Time Series Clustering Using Object-Based Approaches and 3D Convolutional Autoencoder," Remote Sensing 12: 1816 (2020)
- Guillaume Dupont, Ekaterina Kalinicheva, Jérémie Sublime, Florence Rossant, Michel Pâques, "Analyzing Age-Related Macular Degeneration Progression in Patients with Geographic Atrophy Using Joint Autoencoders for Unsupervised Change Detection," Journal of Imaging 6(7): 57 (2020)

International Conferences

- Ekaterina Kalinicheva, Jérémie Sublime, Maria Trocan, "Object-Based Change Detection in Satellite Images Combined with Neural Network Autoencoder Feature Extraction," IPTA 2019: 1-6

– Ekaterina Kalinicheva, Jérémie Sublime, Maria Trocan, “Change Detection in Satellite Images Using Reconstruction Errors of Joint Autoencoders,” ICANN (3) 2019: 637-648

Unranked and National Conferences

– Ekaterina Kalinicheva, Jérémie Sublime, Maria Trocan, “Neural Network Autoencoder for Change Detection in Satellite Image Time Series,” ICECS 2018: 641-642

– Ekaterina Kalinicheva, Jérémie Sublime, Maria Trocan, “Analysis of Objects Evolution in Satellite Image Time Series Transformed with Neural Network Autoencoder,” ASPAI 2019

Chapter 1

Introduction to Remote Sensing and Satellite Image Analysis

Contents

1.1 Introduction

Remote sensing is a powerful tool for Earth observation and analysis which allows to study an area of interest without direct physical interaction with it. With the development of satellite technologies, we got access to images acquired all over the World that makes it possible to study almost any phenomena from any place on the globe.

While only several decades ago, remote sensing image analysis was a difficult, time-consuming and manual task; nowadays, data mining algorithms suited for image processing make it possible to analyze the image data using computer in no time.

In this thesis, we present unsupervised algorithms for satellite image time series analysis. This chapter present some basic concepts required to understand our work and satellite image analysis in general. Section 1.2 presents the keypoints of remote sensing images. Section 1.3 describes different satellite missions. And, finally, Section 1.4 gives a general idea of data mining applied to satellite images.

1.2 Remote Sensing Images

Remote sensing is the acquisition of information about an object or a phenomenon without making a direct physical contact with it. Usually, remote sensing refers to the use of aircraft- or satellite-based technologies to acquire the properties of the Earth surface and corresponding objects. Remote sensing technologies for Earth observation involve an aircraft or an artificial satellite and a sensor. Sensors can be divided in two types: active and passive. Active instruments use their own source of energy to interact with an object, while passive instruments use the energy emitted from a natural source, in particular, from the sun.

Most often, active sensors refer to Synthetic Aperture Radar (SAR) that measures surface roughness. The main idea of this approach is to measure surface backscattering - a portion of emitted radar signal that is redirected back by the target. Passive sensors are mostly represented by optical remote sensors that measure the amount of sun energy reflected by the target.

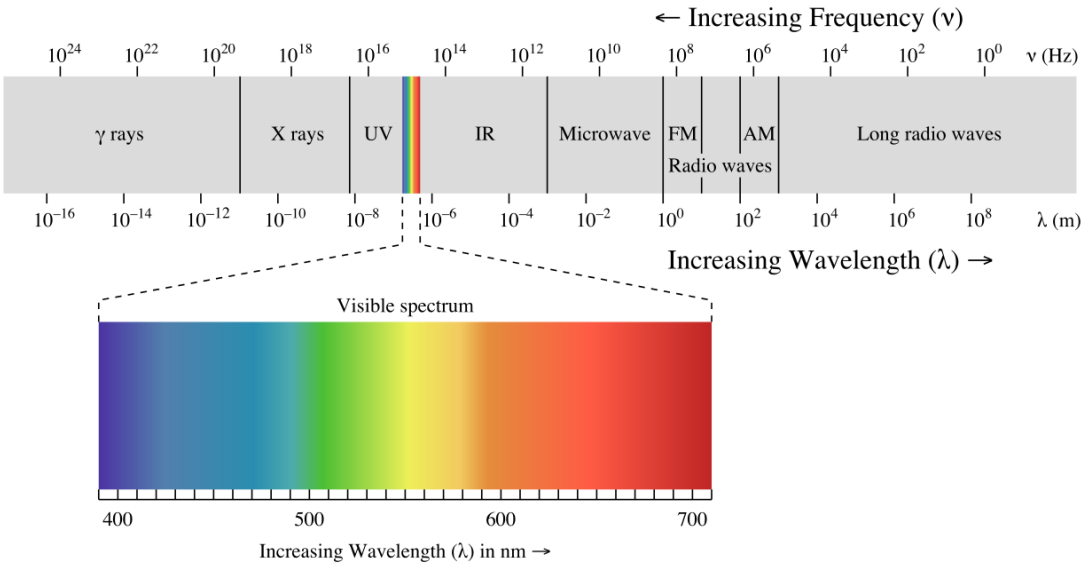

Figure 1.1: Electromagnetic spectrum.

In remote sensing, we exploit the properties of the radiation that is defined by the electromagnetic waves. These waves are characterized by their frequency (Hz) or their wavelength (meters). Figure 1.1 illustrates the electromagnetic spectrum with the corresponding wavelengths.

The spectrum is divided in different spectral ranges. We distinguish the visible spectrum (the radiation perceived by human eye), infrared, ultraviolet, etc. The Earth surface reflects different types of radiation, depending on its coverage. Figure 1.2 shows that vegetation absorbs the visible radiation and reflects near-infrared radiation, and it is the same for bare soil. At the same time, water surfaces absorbs almost all the visible waves.

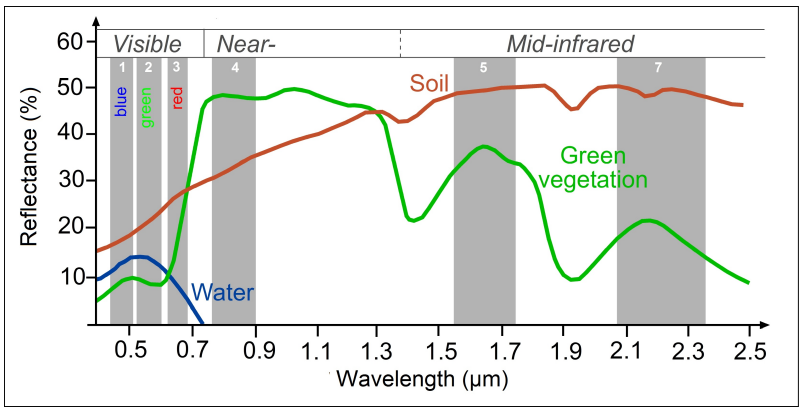

Figure 1.2: Spectral signatures of the water, green vegetation and soil within the different windows of the electromagnetic spectrum.

Every satellite image acquired by an optical sensor is characterized by its:

- **number of spectral bands**;
- **spectral resolution** - spectral width of each spectral band or the capacity of sensor to define fine wavelength intervals for each band (for example a panchromatic band contains all the visible spectrum, while a red band is able to capture the wavelength around 625-740 nm, depending on a satellite, etc);
- **spatial resolution** - pixel size of each band;
- **temporal resolution** - time gap between two consecutive images taken over the same geographic area[1];

[1]Note that a spatial mission may have several satellites that acquire the information over the same geographical area during different period of time. In this work, the term "temporal resolution" refers to the frequency of image acquisition by the ensemble of satellites of the mission.

– **radiometric resolution** - range of image bits, reflects the capacity of an instrument to distinguish differences in object reflectance.

We distinguish mono-, multi- and hyperspectral images. Monospectral images contain a unique spectral band and the image pixels are characterized by a single value. Multispectral images contain from 3 to 10 spectral bands, while for hyperspectral images their number is higher that 10. Their pixels are characterized by vectors of radiometric values from each band. Monospectral images usually contain a single panchromatic band. In the meantime, multispectral images should have at least green, red and NIR bands as they are the most informative.

1.3 Satellite Missions

The first satellite image of Earth was obtained the 14th August 1959 by U.S. orbital satellite Explorer VI [4]. However, the satellites started to acquire Earth data on regular basis more than 10 years later. The launch of the Landsat mission in 1972 gave people a unique possibility to observe different spots of Earth surface every 18 days. Landsat-1 was the first satellite specifically designed to study and monitor our planet's landmasses [5]. This satellite provided images with 60 meters spatial resolution with green, red, and two infrared bands.

This breakthrough gave the researchers the previously unavailable possibility to remotely monitor the evolution of Earth surface coverage at global level. Later, with the development of technology, many other missions with different spatial and temporal resolution were launched providing numerous products for Earth observation. The most known satellite missions of the last 20 years are listed in Table 1.1.

While some missions provide satellite image time series for the whole globe coverage and are freely available for the public use (Landsat, Sentinel, MODIS), others provide time series for commercial use only. Finally, the images of certain areas are freely available as a part of a specific program.

In our work, we use time series issued from two different spatial missions - SPOT-5 and Sentinel-2. These missions were chosen because they provide high resolution freely available images (10 m in visible and infrared spectrum). Currently, some SPOT-5 time series are available as a part of SPOT World Heritage (SWH) program. Despite the good spatial resolution for that time, the final SPOT-5 time series product has irregular temporal resolution.

Contrary to SPOT-5 satellite, nowadays, Sentinel-2 mission supplies free images of any spot of the world with high temporal resolution that makes it a primary source

Table 1.1: Satellite missions.

Mission	**Active years**	**Temporal resolution**	**Spatial resolution**	**Spectral bands**
Landsat-7 (USA, USGS/NASA)	1999 - late 2020	16 days	30 m 30 m 60 m 15 m	B, G, R, 2 NIR, MIR, thermal, PAN
Landsat-8 (USA, USGS/NASA)	2013 -	16 days	30 m 30 m 30 m 15 m 30 m	coastal aerosol, B, G, R, NIR, 2 SWIR, PAN cirrus
MODIS (USA, NASA)	1999 -	1-2 days	250 500 1000 m	36 bands
SPOT-4	1998 - 2013	26 days	20 m 10 m	G, R, NIR, SWIR, PAN
SPOT-5	2002 - 2015	26 days	10 m 20 m 5 m	G, R, NIR, SWIR PAN
RapidEye	2009 -	1 day	6.5 m	B, G, R, R edge, NIR
SPOT 6/7	2012/2014 -	days	6 m 1.5 m	G, R, NIR, PAN
Pleiades	2011 -	1 day	2 m 0.5 m	B, G, R, NIR, SWIR PAN
Sentinel-2	2015 -	1-2 days	60 m 10 m 20 m 60 m 20 m	coastal aerosol, B, G, R, NIR, 4 vegetation R edge, water vapour, cirrus 2 SWIR

B- blue, G- green, R- red, NIR- near-infrared, MIR- mid-infrared,
SWIR- shortwawe-infrared, PAN- panchromatic bands.

of images for different applications. For this reason, in our work, we also use datasets captured by the Sentinel-2 mission, so the experiments are led both on archive data and on the recent one.

SPOT (French: *Satellite Pour l'Observation de la Terre* or *Système Probatoire d'Observation de la Terre*, lit. Probationary Satellite for Earth Observation or Satellite for Earth Observation) - is a civil program for Earth observation. It is developed by the french National Center for Space Studies (CNES - French: *le Centre National d'Etudes Spatiales*). SPOT is a family of seven satellites: SPOT-1, SPOT-2, SPOT-3, SPOT-4, SPOT-5, SPOT-6, and SPOT-7. The chronology of the missions is presented in Figure 1.3.

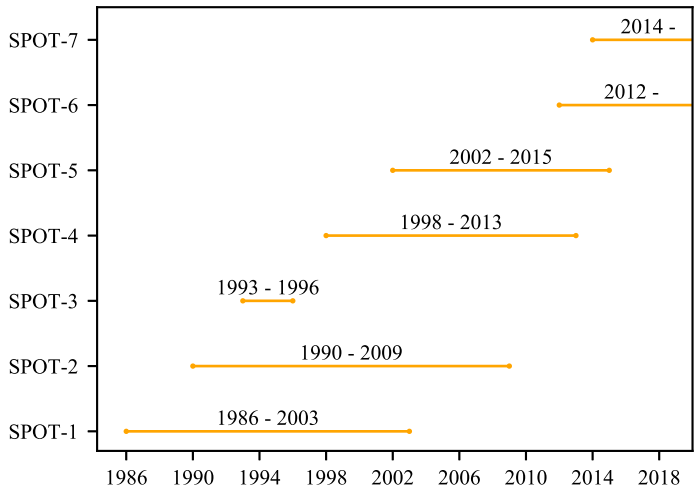

Figure 1.3: SPOT satellites.

Some SPOT 1-5 time series are available as a part of SWH program, however, only SPOT-5 mission provides 10 meters resolution bands in visible spectrum contrary to 20 m for others missions. The wavelength range for each spectral band of SPOT-5 satellite is the following ([6]):

- Green (0.50-0.59 μm),
- Red (0.61-0.68 μm),
- NIR (0.78-0.89 μm),
- SWIR (1.58-1.75 μm).

Green, Red and NIR bands have 10 m spatial resolution. SWIR band yields 20 m images, which are then resampled to 10 m in the final product.

Sentinel mission is a part of the EU Copernicus program in collaboration with the European Spatial Agency (ESA). The goal of this program is to replace the current older Earth observation missions and to ensure a supply of continuous data. Each mission focuses on a different aspect of Earth observation: atmospheric, oceanic, and land monitoring, and the data will be of use in many applications [7]. The Sentinel missions are currently composed of Sentinel-1 (SAR for land and ocean monitoring), Sentinel-2 (land monitoring), Sentinel-3 (sea surface topography, sea and land surface temperature, and ocean and land surface colour) and Sentinel-5P (air quality monitoring) with Sentinel-4 (monitoring of atmospheric composition) and Sentinel-5 (air quality monitoring) launched in the nearest future. Each of Sentinel 1, 2, 3 missions is comprised of 2 satellites to ensure the best globe coverage and image acquisition frequency within the mission. Two more satellites for Sentinel-3 mission are in order to be launched. The chronology of Sentinel missions is presented in Figure 1.4.

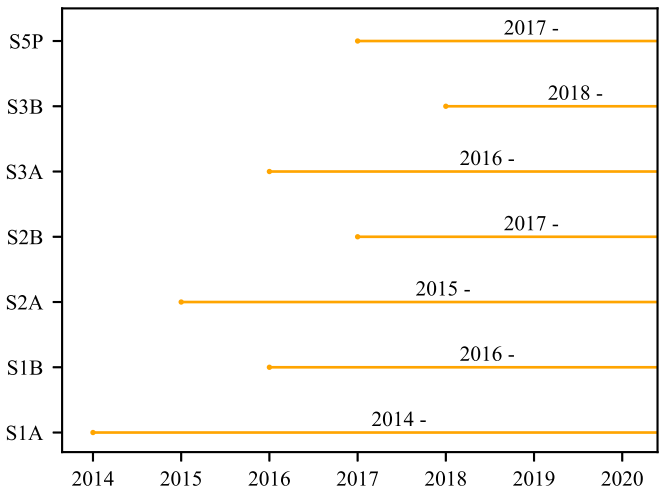

Figure 1.4: Sentinel satellites.

Optical images are supplied only by Sentinel-2 and 3 missions. In our work we use time series issued from Sentinel-2. The wavelength for each spectral band of Sentinel-2A and 2B satellites is indicated in the Table 1.2.

Table 1.2: Sentinel-2 spectral bands [3].

Spectral band		**Wavelength, μm**	
Spectrum	**Resolution, m**	**S2A**	**S2B**
Coastal aerosol	60	0.432 - 0.453	0.432 - 0.453
Blue	10	0.459 - 0.525	0.459 - 0.525
Green	10	0.541 - 0.578	0.540 - 0.577
Red	10	0.649 - 0.680	0.650 - 0.680
Red Edge 1	20	0.697 - 0.712	0.696 - 0.712
Red Edge 2	20	0.733 - 0.748	0.732 - 0.747
Red Edge 3	20	0.773 - 0.793	0.770 - 0.790
NIR	10	0.780 - 0.886	0.780 - 0.886
Red Edge 4	20	0.854 - 0.875	0.853 - 0.875
Water vapour	60	0.935 - 0.955	0.933 - 0.554
Cirrus	60	1.358 - 1.389	1.362 - 1.382
MIR 1	20	1.568 - 1.659	1.563 - 1.657
MIR 2	20	2.115 - 2.290	2.093 - 2.278

1.4 Introduction to Data Mining Applied to Images

Data mining applied to images is a trendy field since the early 1980 and is more commonly known under the name *Computer Vision*. It has numerous applications: analysis of medical images, security scans, remote-sensing, etc. The final goal of computer vision is to mimic the skills of human vision when analyzing an image: finding the different element of the image and eventually identifying them.

Automatizing computer vision is a complex task the goal of which is to reduce the need for human intervention when extracting knowledge from an image. This process can be split into several steps for any kind of image:

1. acquiring the data,
2. preparing the data (denoising, encoding, and so forth),
3. mining the data (choosing a model and a method, and then applying it to the data),
4. validating and interpreting the results,
5. integrating the mined knowledge into a data base for further use (optional).

From a computer point of view, an image is a data set made of several elementary objects called pixels. These pixels have characteristics of their own such as gray level value for each image channel and their position (x,y coordinates) on the image. Therefore, one of the main difference between an image set and a regular data set is that in the cases of images, the data to classify have coordinates -and thus geographical dependencies- in addition to their regular attributes. Another important difference is that the pixels are not necessarily the most interesting image elements to process. As we will discuss bellow, while it is possible to use an image analysis algorithm directly on the pixels, it is quite often more interesting to process "*regions*" that are made of several pixels.

1.4.1 Pixel-Based Approaches

Pixels Used as Data

Pixel-based approaches are among the most common approaches for data mining in computer vision. This type of approaches have been widely studied in the last 50 years and still remains widely used [8–10]. These approaches consider the pixels composing an image as data to be labeled (supervised learning) or clustered (unsupervised learning). Pixels are described as vectors built from physical attributes (radiometric and geographic attributes), to which can be added *neighborhood dependencies* either in the form of a set of neighboring data, or in the form of coordinates in the computer image itself. By neighborhood dependencies, we mean the possible links between clusters that are next to each other and the way they may influence each other classification or clustering.

For instance, the description of a regular red, green, blue (RGB) computer image of size $H \times W$ is the following:

$$X = \{x_1, ..., x_N\}, \quad x_n = \{r, g, b\}, \quad (r, g, b) \in [0, 255], \quad N = H \times W$$

After the process of clustering (or classification), each pixel x_n will be associated to a label, independent of the value of the neighboring pixel.

Limitations of Pixel-Based Approaches

While pixel-based approaches are still widely used, they suffer from many defects.

First, most pixel-based approaches only use the radiometric information from the pixels leaving all other characteristics unused [11]: shape, length, width, texture, etc.

However, the most important and recent issue with pixel-based approaches is that in modern imaging single pixels have no signification because the objects of interest are generally covering a large number of pixels. This is particularly true with high resolution (HR) and very high resolution (VHR) satellite images, such as those that we used in this thesis. When dealing with a high resolution satellite image, even an average quality one, the objects of interest such as roads, trees, or houses are already too big for any of them to fit in a single pixel. They are actually composed of several heterogeneous pixels. Thus a pixel-based analysis not only would make little sense, but it would also result in loosing key information -such as the real shape of these objects- that is key to their identification.

One solution to reduce this kind of issue is to label the pixels, not only based on their own attributes, but also depending on the characteristics and labels of the pixels in their neighborhood [12–14]. These approaches consider a neighborhood window around the pixel to analyze and add texture information to the pixel color attributes.

While neighborhood enhanced pixel-based methods are an interesting first step, recent studies have shown that in the case of very high resolution pictures, for some applications, it is still not enough to achieve good performances [15]. To cope with these issues, other approaches based on regions of agglomerated pixels have been developed. The principles of these methods as well as their pros and cons are introduced in the next subsection.

1.4.2 Region-Based approaches

Region-based approaches are the basis of Object Based Image Analysis (OBIA) [16]. The main idea behind these methods is that since the pixels themselves have no semantic meaning, a first step is required to regroup the pixels into regions that will represent the real objects of interest to be identified or put into clusters. Therefore, for region-based approaches the data mining process consists in two steps instead of only one:

1. segmentation of the original image to determine the border of the objects of interest,

2. clustering or classification of the newly identified regions.

These regions have new and unique characteristics that are based on both the characteristics of the pixels they are made of, but also shape, size and texture features.

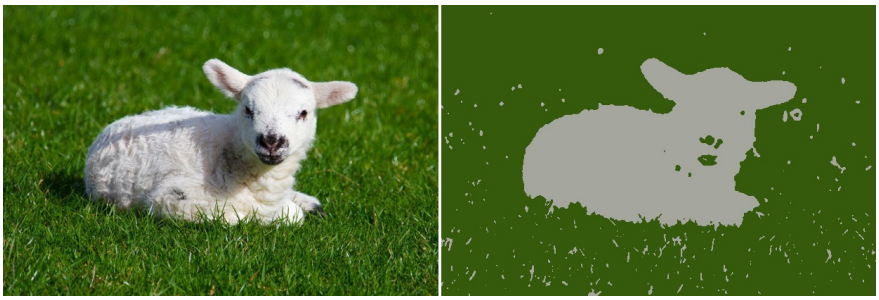

Figure 1.5: Example of a crude segmentation to detect a white lamb on a grass background

Image Segmentation

The segmentation of an image is a process that consists in grouping together neighbor pixels with the goal of finding homogeneous segments the borders of which will be a good approximation of the objects present within the image [17]. The segments created using this process are supposed to be relevant and match the real objects that can be found in the picture (Figure 1.5).

The definition of a proper image segmentation has been formalized by Pavlidis and Zucker [18, 19] in the form of the 4 following axioms:

1. Each pixel of the image must belong to one and only one segment.
2. Each segment must be continuous, i.e. made of connected neighbor pixels.
3. Each segment must be an homogeneous entity.
4. Two adjacent segments must be two distinct homogeneous entities.

Among these four conditions, axioms 3 and 4 rely on a notion of homogeneity that is rather difficult to assess due to complexity of some objects textures. Thus, image segmentation is a difficult process that can lead to results of varying quality depending on the homogeneity criterion and the algorithm that are used. Therefore, over-segmentation and under-segmentation are the two most common problems:

– *Over-Segmentation*: The image contains too many segments after the segmentation process. In this case, many of the objects to be found remain spread over several small segments that do not contain enough pixels. This problem can generally be solved by merging together segments that are too similar or do not represent anything.

– *Under-Segmentation*: The image does not contain enough segments. The resulting segments are so big that they contain several objects inside of them. Unlike with over-segmentation, this problem is more difficult to solve.

In Figure 1.6, we show an example of over-segmentation: the river and some of the buildings are clearly over-segmented. The colors in the image are representing the real object of interest that "should" be found.

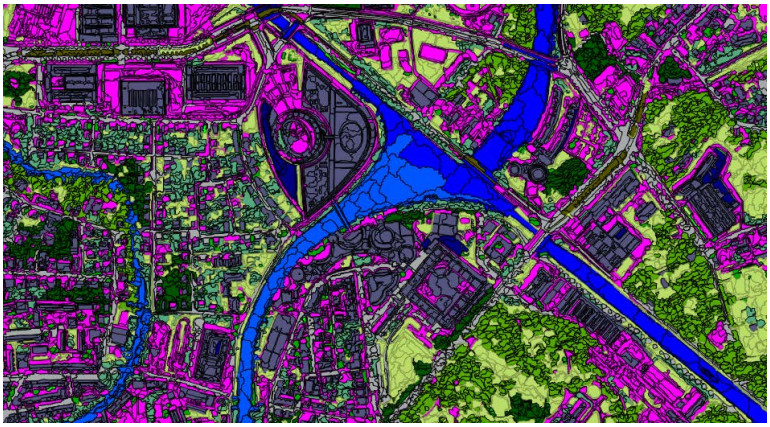

Figure 1.6: Example of an over-segmentation

While both cases should be avoided when possible, there is no generic method that solves these problems. In any case, it is always better to have an over-segmentation rather than an under-segmentation. In the case, over-segmentation the real objects may still be found during the clustering or classification process, even if they are split between several segments. However when several objects are merged in the single segment because of an under-segmentation, there is no way to fix it during the clustering/classification process, and some classes or clusters may be lost for good. Over-segmentation it therefore a much more preferable preprocessing result.

More details on the different segmentation algorithms can be found in the literature [20].

Limitations of Region-Based Approaches

While region-based approaches are more adapted than pixel-based approaches when dealing with HR and VHR images, they also have their limits and disadvantages.

The first obvious limitation is the segmentation process needed to create the regions. As we have shown previously, this process can be cumbersome and requires that the user choose carefully a potentially large number of parameters to achieve acceptable results. Because the segmentation process is a mandatory step for region-based approaches, the quality of the segmentation will have a huge impact on the subsequent clustering or classification process.

Another important aspect is that when the segments and regions are created they add a large number of new attributes that may have to be taken into consideration:

surface of the segments, perimeter and elongation, extrema, variance and average values of the attributes in a given segment, contrast with the neighboring segments, etc. The choice of the right attributes can be tricky, as some of them may or may not be relevant or redundant depending on the type of objects that one wants to identify.

Finally, another obvious limitation of region-based approaches lies in the fact that - particularly with satellite images - there may be several levels of objects of interests to be found depending of the desired level of detail during the clustering process. However, it is not yet possible for that kind of hierarchy between objects made of other objects to be displayed in a segmentation. Therefore the risk of having an under-segmentation at an acute level of detail remains high, while on the contrary the image may end up being over-segmented for a lesser and broader level of detail.

Example: An urban area is made of several different urban sectors that in turn are made of different buildings and streets.

1.5 Discussion

In this chapter, we have presented the essential concepts of remote sensing such as image acquisition and interpretation. In this PhD thesis, we use optical image time series data, so the concept of optical images and their properties were equally reflected in this chapter. Moreover, we have given an overview of the most known satellite missions that provide freely-available time series image data. We have especially focused on the SPOT-5 and Sentinel-2 missions as we exploit their images in our work. Finally, we have presented the concept of computer vision - data mining techniques applied to image processing. These image analysis approaches can be divided between pixel- and object-based. In our work, we use both, although, our pixel-based methods exploit moving window technique where each image pixel is associated to a patch - a square of certain fixed size - that contains neighboring pixels.

The two following chapters will describe some of the existing machine learning algorithms used for different data analysis purposes, including computer vision and satellite image analysis. We will mainly focus on unsupervised algorithms (the ones that do not demand any labeled data) - data clustering, anomaly detection and feature extraction.

Chapter 2

Machine Learning. Clustering and Anomaly Detection

Contents

2.1 Introduction

Machine Learning is a subfield of Computer Science defined in 1959 by Arthur Samuel as "*a field of study that gives computer the ability to learn without being explicitly*

programmed". The methods of machine learning allow analyze different type of data (text, video, images, etc) to define the correlation between entities in order to extract some knowledge.

With the omnipresence of numerous data within a large number of science fields, Machine Learning has become a common tool for Data Mining, model building and prediction in a large number of areas such as biology and medicine [19, 21–24], mathematics [25], finance and business [26–28], physics [29], chemistry [30], marketing and so many others. Thus, Machine learning makes it possible to automatically analyze huge amount of remote sensing images to perform various tasks: land cover mapping, object detection, satellite images time series analysis, change detection, etc.

Machine learning tasks are usually divided into three categories:

- **Supervised Learning:** The computer program is presented with a set of input examples provided with their desired label (training set) from which it will have to build a model or learn some rules that maps the inputs to the right outputs. Once the model has been learned, the computer can apply it to new data for which the output labels are unknown. Tasks related to supervised learning include classification, regression and time series predictions.

- **Unsupervised Learning:** With no labels given, the computer program has to find interesting structures, patterns and groups in a set of data. Potential applications include clustering (that we will formally introduce in the next section), feature learning, regression and pattern recognition.

- **Reinforcement Learning:** Given a dynamic environment, a computer program must perform a certain task and will improve its behavior based on positive or negative rewards inputs decided according to its actions. The algorithm is never directly told how to find the right answer, but has to explore different possibilities based on the rewards it gets for each of its action.

In this thesis, we present a complete time series analysis that includes change detection and clustering performed using unsupervised machine learning techniques. The unsupervised approaches were chosen as they make it possible to analyze almost any data as no training database is required.

In this chapter, we are going to present some of the concepts of machine learning used in our thesis focusing on the unsupervised methods. In Section 2.2, we explain the basics of unsupervised learning, the Section 2.3 reviews of different clustering algorithms. Section 2.4 gives the general information about the anomaly detection.

Finally, in Section 2.5 we present some indices to assess the quality of the presented algorithms.

2.2 Unsupervised Learning

Unsupervised learning is a machine learning task the aim of which is to find hidden structures and patterns in unlabeled data. There are several tasks linked to unsupervised learning, the most known of them are:

- data partitioning (or clustering),
- anomaly detection,
- latent variables model learning (data reprojection or dimensionality reduction),
- data visualization (often based on dimensionality reduction).

Unsupervised learning is said to be "unsupervised" because it finds structures and build a model from completely unlabeled data. It therefore differs from supervised learning which builds a model given already labeled data.

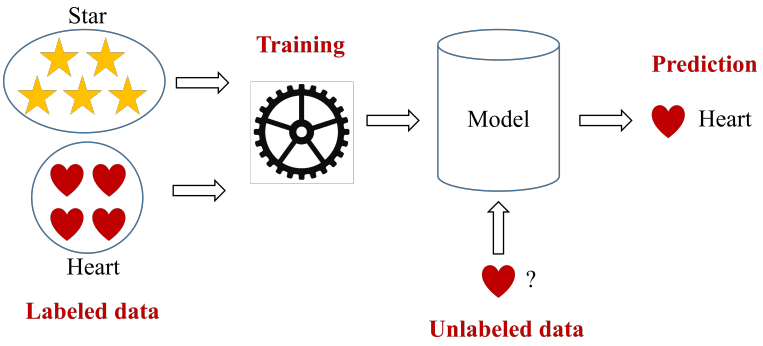

Figure 2.1: Supervised learning.

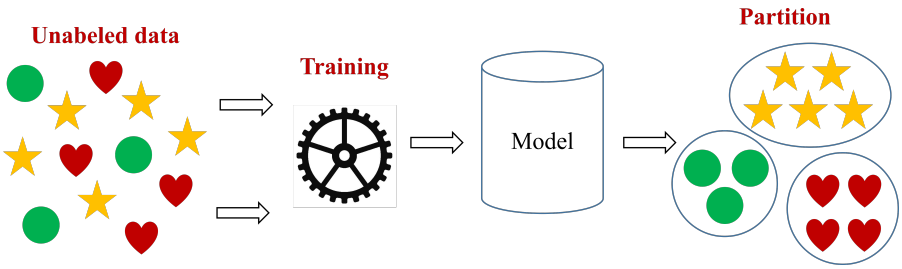

Figure 2.2: Unsupervised learning.

While in supervised learning, we use labeled data to create a model that is then used to make predictions on a new data (Figure 2.1), in unsupervised learning, the same dataset is used both for model creation and final analysis (Figure 2.2).

2.3 Clustering

Clustering is a machine learning task of exploratory data mining the aim of which is to split a data set made of several data (also called *objects*, *data objects*, or *observations*) into several subsets. Each object is described by several *attributes*, also called *features* that describe its properties. The subsets created by the process of clustering a data set are called *clusters*. Objects from a given cluster are supposed to be homogeneous and to share common characteristics. A very simple example of a data set with two attributes and three visually distinct clusters is shown in Figure 2.3.

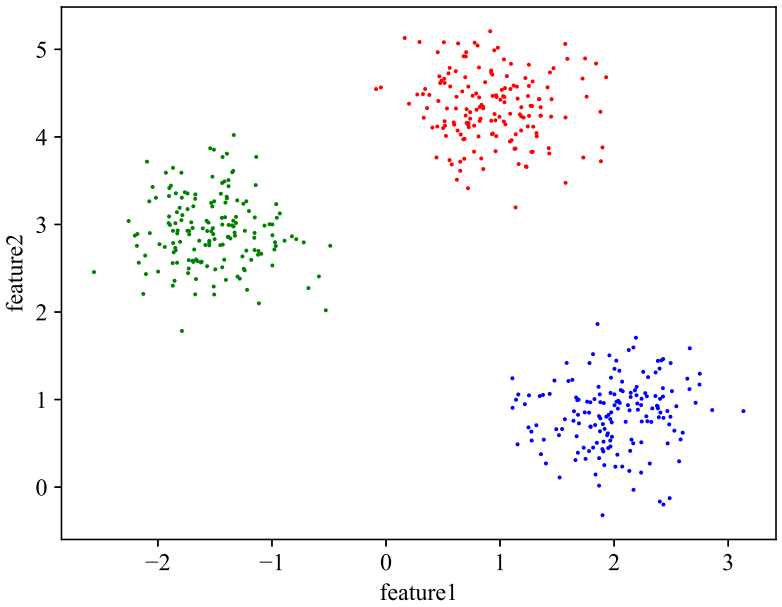

Figure 2.3: Example of a 2 dimensional data set with 3 visually distinct clusters.

There is a huge number of clustering methods that automatically create clusters, each with its own strategy and criteria. The main criterion to build clusters relies on the notion of *similarity* between two objects. Based on this concept of similarity, a clustering algorithm will have to make the decision to group several objects in the same cluster, or to separate them. In this context, the choice of the *similarity measure* is critical since it will ultimately determine the final clusters.

The vast majority of clustering algorithms are based on the notion of distance between the data as a similarity (or dissimilarity criterion). Within this context, clustering algorithms often try to optimize an objective function which favors clusters that are both compact and well separated. For these algorithms, the choice of the distance function is key. Examples of common distances are shown in Table 2.1.

Euclidian dist.	$\|\|a-b\|\|_2 = \sqrt{\sum_i (a_i - b_i)^2}$
Squared Euclidian dist.	$\|\|a-b\|\|_2^2 = \sum_i (a_i - b_i)^2$
Manhattan dist.	$\|\|a-b\|\|_1 = \sum_i \|a_i - b_i\|$
Maximum dist.	$\|\|a-b\|\|_\infty = \max_i \|a_i - b_i\|$
Mahalanobis dist.	$\sqrt{(a-b)^\top S^{-1}(a-b)}$ where S is the covariance matrix
Hamming dist.	$Hamming(a,b) = \sum_i (1 - \delta_{a_i,b_i})$

Table 2.1: Examples of common distances.

While this notion of similarity is a first step to define a clustering method, it is however not enough. Once an agreement as been found on which similarity measure will be used, the next step is to define a strategy to build the clusters using this similarity measure. Given the large number of similarity measures available, and considering that several strategies are usually available for each of them, it is no surprise that a huge variety of clustering methods is available in the specialized literature.

Clustering methods have been used in a large array of applications such as pattern recognition, web mining, business intelligence, biology for taxonomy purposes, or security applications. Clustering can also be used for *outliers detection* [31, 32], where *outliers* (objects that are "far" from any cluster) may be more interesting than common cases.

In business intelligence, clustering can be used to sort a large number of customers into groups where customers share strong similar characteristics. In pattern recognition, clustering can be used to discover clusters in handwritten character recognition systems. Clustering also has many applications in web-mining. For example, a keyword search may often return a very large number of hits (i.e., pages relevant to the search) due to the extremely large number of web pages. Clustering can be used to organize the search results into groups and present the results in a more concise and easily accessible way. Moreover, clustering techniques have been developed to cluster documents into topics, which are commonly used in information retrieval practices. Applications of outliers detection include the detection of credit card fraud and the monitoring of criminal activities in electronic commerce. For example, exceptional cases in credit card transactions, such as very expensive and infrequent purchases,

may be of interest as possible fraudulent activities [33].

Far from being an exhaustive state of the art, this section introduces some of the key concepts in clustering. We advise our readers that may want a more exhaustive state of the art on clustering methods, or to read more about the differences between existing methods to read one of the following documents [34–36].

In this section, we will present 3 main families of clustering methods:

- distance-based that can be divided in the following subfamilies:
  - hierarchical,
  - prototype-based,
  - density-based,
- spectral,
- probabilistic.

In the forthcoming subsections, we will present several clustering methods based on different measures and strategy. In particular, we will focus on distance-based and prototype-based methods that are the most used in satellite image processing.

2.3.1 Prototype-Based Algorithms

The principle of prototype-based algorithms is based on *vector quantization*, a data compression process which consists in representing the data with a few representatives called *prototypes*. Each data will then be linked to its closest prototype in the data space. The main task of these algorithms is therefore to build relevant prototypes and link the data to them.

A common example of prototype would be a centroid of a high density area. Depending on the number of prototypes, each of them may represent a cluster, or several of them may need to be regrouped to find the clusters.

The K-Means Algorithm

The K-Means algorithm is one of the most famous prototype-based clustering algorithm. It is a simple and fast, yet relatively good clustering method. Its principle is the following [37, 38]: Suppose that we would like to divide our data into K clusters, the value of K being known in advance. We allocate K cluster prototypes (also called mean-values) to our input space, and we would like to move these prototypes so that each of them will become the centroid of a cluster. Given that we have chosen a distance measure, the procedure to do so consists in alternating the following two

steps until convergence: 1) Link each data to the closest prototype, 2) Move the prototype so that is becomes the barycenter of the current data to which it is linked. This procedure is described in Algorithm 1 and an example with 2 clusters on the "*Old Faithful*" data set is shown in Figure 2.4.

It is convenient at this point to give a notation that describes the assignment of data points to clusters. For each data point x_n, let $s_{n,i} \in \{0, 1\}$ be a set of binary indicator variables with $i \in [1..K]$. The $s_{n,i}$ are used to describe to which one of the K clusters a data has been assigned. For instance, if x_n is assigned to cluster c_k, then $s_{n,k} = 1$ and $\forall i \neq k, s_{n,i} = 0$. Ultimately, what the K-Means algorithm does is to optimize a cost function $\tilde{R}(\mu)$ as given in Equation (2.1).

$$\tilde{R}(\mu) = \sum_{n=1}^{N} \sum_{i=1}^{K} s_{n,i} ||x_n - \mu_i||^2 \tag{2.1}$$

Because each phase reduces the value of the objective function $\tilde{R}(\mu)$, convergence of the algorithm is assured. However, it may converge to a local rather than global minimum of $\tilde{R}(\mu)$. The convergence properties of the K-means algorithm have been studied in [37].

Algorithm 1 K-Means Algorithm

Choose a value for K
Randomly initialize the K centroids μ_i
while *Learning* **do**
  forall $x_n \in X$ **do**
    Assign x_n to the cluster c_i with the closest centroid:
    $s_{n,i} = \begin{cases} 1 & \text{if } i = argmin_i ||x_n - \mu_i||^2 \\ 0 & \text{otherwise} \end{cases}$
  end
  Minimize Equation (2.1):
  forall μ_i **do**
    $\mu_i = \frac{\sum_n x_n \cdot s_{n,i}}{|c_i|}$
  end
end

Several algorithms based on improved or modified versions of the K-Means algorithm have been proposed in the literature [39–45]. Algorithms based on the K-Means algorithm suffer from several weaknesses: The main one is the need to provide a K. It requires to know in advance how many clusters are to be found. In practice this is rarely the case because we expect the clustering algorithm to actually discover the clusters. Therefore, the only solution when the number of

clusters is really unknown is to run the algorithm several times with different values of K and to pick the best clustering based of a given quality index (for instance the *Silhouette index* [46] or the *Davies-Bouldin* index [47]). This method is costly and may prove ineffective because of the non-deterministic nature of the K-Means algorithm. Adaptations of the K-Means algorithm have been proposed [48] to solve this issue, but they remain only partially satisfying. Second, algorithms based on the K-Means can only find hyper-spherical clusters and will also fail to detect the clusters properly if their sizes are significantly different.

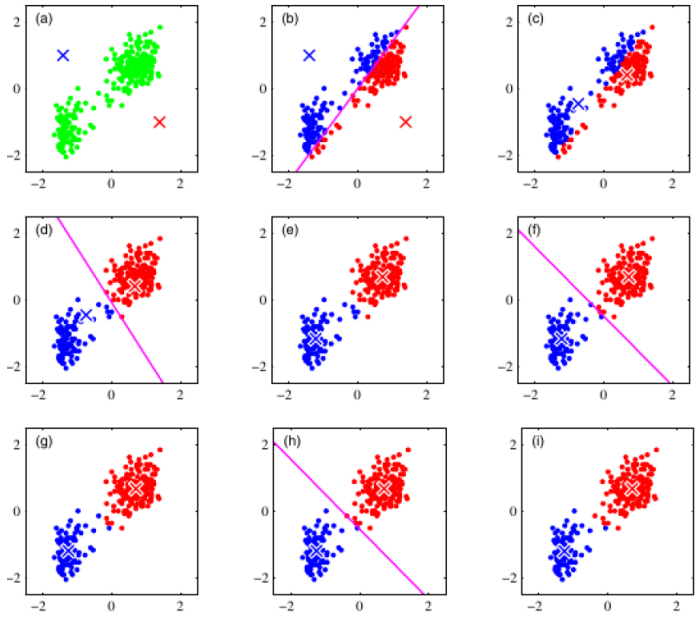

Figure 2.4: Illustration of the K-Means algorithm on the Old Faithful data set

2.3.2 Hierarchical Clustering Methods

Most of the clustering algorithms are building results that come in the form of a flat separated clusters where the clusters are all independent and no structure exists between them. However, another approach consists in trying to have results in the form of clusters between which there is a hierarchical structure. The most common structure is to build clusters as trees, very similar to phylogenetic trees in biology: at the top of the tree is a single cluster containing all the objects. This cluster will then be split into several sub-clusters that will also be split into other clusters and so on. The clusters close from the root of the tree will be crude and will contain a lot of objects that may still be relatively dissimilar, and the clusters far from the root will contain less but more similar objects [36, 49–52].

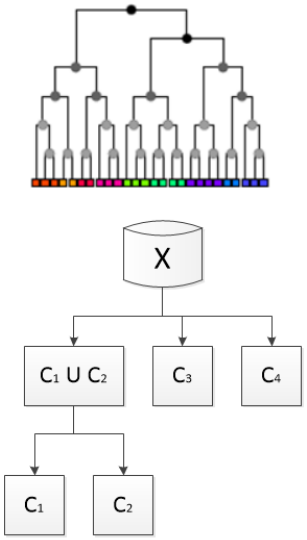

Figure 2.5: Example of a hierarchical clustering result.

This type of clustering where the solution is given in the form of a tree (or dendrogram) is called *hierarchical clustering algorithm* (HCA). In this case, each data object belongs to a single leaf cluster, but as a consequence it also belongs to all the father nodes up to the root of the tree. Hierarchical clustering is further divided into two sub-categories: agglomerative approaches (or "bottom-up" approaches) and divisive approaches (or "top-down" approaches). In agglomerative methods, the clustering algorithm starts with all objects belonging to a different leaf and then regroup them until there is a single cluster containing all the objects. Divisive approaches on the contrary start with all the data in the same cluster, and this cluster is then divided into sub-clusters in a recursive manner. In Figure 2.5, we show an example of a hierarchical result with 4 clusters.

There are many different methods to create a tree of hierarchical clusters. In Algorithm 2, we show the main framework followed by agglomerative methods.

Algorithm 2 Hierarchical clustering algorithm: general framework (agglomerative)

```
Create a cluster for each element
Initialize the dendrogram's leaves
while There is more that one cluster left do
    Compute all the pairwise similarities between the clusters
    Merge the two clusters that are the most similar
    Update the Dendrogram
end
Cut the dendrogram depending on the desired number of clusters
```

The main difference between the various algorithms proposed in the literature lies on the choice of a similarity measure to merge (or split) the clusters. The main measures for hierarchical clustering are listed bellow:

– *Single-linkage*: Assesses the minimal distance that exists between data belonging to two different clusters. This linkage is very popular because it can be used to detect clusters of any shapes and that may not be hyper-spherical. However, it is noise-sensitive and cannot detect clusters that are in direct contact.

$$D_s(c_1, c_2) = \min_{x \in c_1, y \in c_2} d(x, y) \tag{2.2}$$

– *Complete-linkage*: Assess the maximal distance that exists between data belonging to two different clusters. It is highly noise-sensitive and rarely used in practice.

$$D_c(c_1, c_2) = \max_{x \in c_1, y \in c_2} d(x, y) \tag{2.3}$$

– *Average-linkage*: It considers the average distance between the data belonging to two different clusters. It is less noise-sensitive than the two previous links. But it tends to favor hyper-spherical clusters.

$$D_a(c_1, c_2) = \frac{1}{|c_1||c_2|} \sum_{x \in c_1} \sum_{y \in c_2} d(x, y) \tag{2.4}$$

– *Centroid-linkage*: It assesses the distance between the mean values of two clusters. This linkage is not noise-sensitive but also tends to favor hyper-spherical clusters.

$$D_\mu(c_1, c_2) = ||\mu_1 - \mu_2|| \tag{2.5}$$

– *Ward-linkage*: One possible variation of *Centroid-linkage* is the Ward Criterion [49] where the mean-values are weighted depending on the number of elements in the cluster.

$$D_w(u, v) = \sqrt{\frac{|v| + |s|}{T} d(v, s)^2 + \frac{|v| + |t|}{T} d(v, t)^2 - \frac{|v|}{T} d(s, t)^2} \tag{2.6}$$

where u is the newly joined cluster consisting of D and t, v is an unused cluster in the forest, $T = |v| + |s| + |t|$, and $| * |$ is the cardinality of its argument.

The CURE algorithm [53] uses an alternative linkage that enables detecting clusters of any shape while remaining resistant to noise. To do so, a few elements are

selected in each cluster. These elements are chosen by first picking the farthest element from the cluster centroid, and then the farthest element from the previously picked one, and so on until c elements per cluster have been picked. These representative are artificially modified and moved closer to the cluster centroid. Finally the single-linkage criterion is used as a merging criterion.

Hierarchical clustering has two main limitations: First, once the clusters dendrogram is built, one still needs to decide where to cut to get the final clusters. This choice remains a difficult one despite a plethora of available methods in the literature (see [36]). Second, these methods have a high computational complexity of at least $O(N^2)$ for a data set of size N, which makes them difficult to use with large datasets.

2.3.3 Density-Based Clustering Methods

Density-based clustering methods [54, 55] consider the most basic and perhaps the most visual definition of a cluster: a cluster is an area of space with a high density of data and is separated from other clusters by low density regions. This notion of density relies on the concept of object neighborhood. By object neighborhood, we mean other objects that are located at a certain distance of the observed object. For density-based clustering methods, the higher the number of neighbors in an object's vicinity, the more chances that this object belongs to a high density region, and thus is part of a cluster formed with its neighbors. Unlike many other clustering methods, density-based algorithms do not assume that the clusters should have specific shapes and can easily detect concave clusters [56] if the parameters are well tuned.

The parameters for this type of clustering algorithms generally include a distance threshold to determine what should be considered a given object's neighborhood: let $V_\epsilon(x)$ be the neighborhood of an object x so that $V_\epsilon(x) = \{y \in X | d(x, y) \leq \epsilon\}$, with ϵ a threshold and $d(x, y)$ a distance between x and y. Examples of such density-based method include the DBSCAN algorithm (Density-Based Spatial Clustering of Applications with Noise) [39, 57, 58], or the OPTICS algorithm (Ordering points to identify the clustering structure) [59] which adds a second threshold determining the minimum number of objects that must be in a neighborhood for the said neighborhood to be considered dense.

Density-based clustering methods can be equally used for anomaly detection. While the high density areas form separate clusters, the points not included in these areas are considered as outliers and are not attached to any cluster. Figure 2.6 presents an example of DBSCAN clustering result with 6 well defined clusters and anomalous points (in violet).

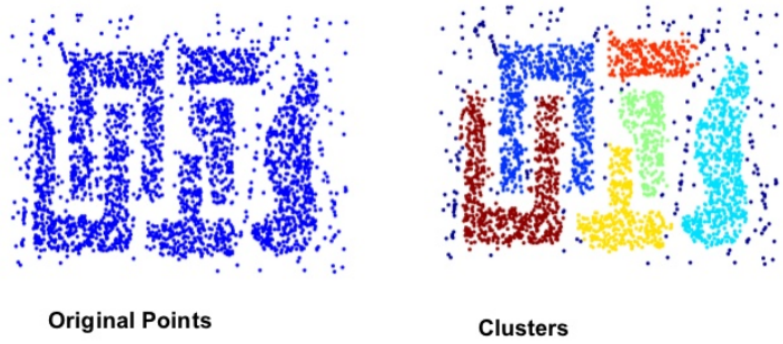

Figure 2.6: Example of a DBSCAN clustering result with 6 clusters and outliers points in violet.

2.3.4 Spectral Methods

Spectral methods are recent clustering techniques that have had a lot of success recently. The main idea of spectral clustering is to see clustering as a graph partitioning problem. Spectral clustering considers the adjacency matrix of the graph representing the data, and the eigenvalues of this matrix. This type of clustering is called "spectral" because it uses the spectrum (eigenvalues) of the data set similarity matrix. Since these methods use a similarity matrix between the different objects, without using proper kernel matrices, they are actually quickly limited when the number of objects becomes relatively big.

Example of spectral techniques include the *Normalized Cuts* algorithm (also called Shi-Malik algorithm) [60]. This algorithms split the data into two subsets (X_1, X_2) based on the eigenvectors matching the first and second smallest eigenvalue of the similarity matrix Laplacian. The algorithm then uses a recursive hierarchical approach to create clusters based on the eigenvectors values and a chosen threshold.

The Meila-Shi algorithm [61] is another example of spectral method. For k given, it considers the eigenvectors matching the k highest matrix eigenvalues. Then it uses a regular clustering algorithm (such as the K-Means algorithm) to regroup the data based on their respective components in the eigenvectors.

2.3.5 Probabilistic Clustering Methods

Probabilistic clustering methods (sometimes called probabilistic model-based methods, or generative models) are algorithms the main hypothesis of which is that the data are following a given probability density function. The goal of such algorithms is to estimate the parameters of these density functions and to define a mixture model

to represent the different clusters. Many clustering techniques can be depicted in this model, e.g. fuzzy C-Means, Gaussian mixtures models (GMM), mixtures of Bernouilli distributions, etc. These methods make the hypothesis that each cluster c_i is linked to probability density function $p(X, \theta_i)$, where θ_i contains the parameters of the function. These laws can then be used to assess the probability of a data x_n to belong to a cluster c_i, thus generating a fuzzy partition. If we note π_i the proportion of the i^{th} component in the mixture model, then the parameters of the model are: $\Theta = \{(\pi_1, \theta_1), ..., (\pi_K, \theta_K)\}$, and the global density function is the following:

$$p(X, \Theta) = \sum_{i=1}^{K} \pi_i p(X, \theta_i) \tag{2.7}$$

This type of model is called a *generative model*, because once the parameters are known, it is possible to re-create the data just from the probability density functions and the mixing coefficients.

The EM algorithm for the Gaussian Mixture Model The Expectation-Maximization (EM) algorithm [62] is an iterative method used to find the *maximum likelihood* or the *maximum a posteriori* (MAP) estimate of parameters in probabilistic and statistical models. As such it can be seen as an alternative to gradient-descent/ascent methods [63, 64] to find the optimal parameters of a given function.

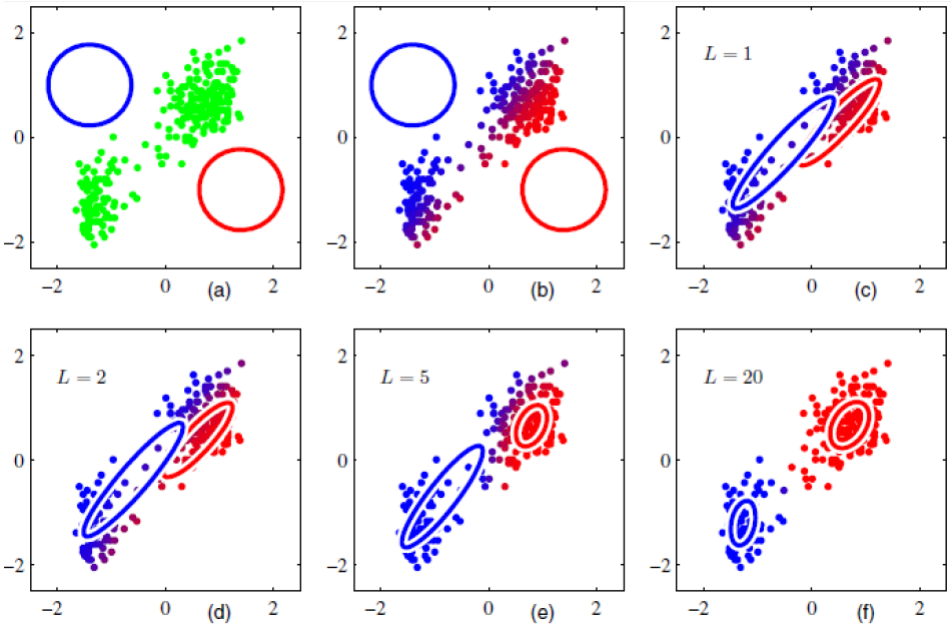

Figure 2.7: Illustration of the EM algorithm (GMM) on the Old Faithful data set

When it comes to applications of the EM algorithm for clustering, most of the time the model used will be the Gaussian Mixture Model (GMM), where each cluster c_k follows a distribution given by a mean value μ_k and covariance matrix Σ_k in addition to a mixing probability π_k. This version of the EM algorithm with the Gaussian Mixture model can been seen as a generalized version of the K-Means algorithm where clusters can have different sizes and can take an elliptical shape instead of being limited to a spherical one. In Figure 2.7 shows an example of such application on the Old Faithful data set.

2.4 Anomaly Detection

Anomaly detection is a process of identifying unexpected items or events that differ from the norm. Typically, anomalous data can be connected to a rare event or some problem, such as, for example, malfunctioning equipment, bank fraud, spam e-mails, disease detection from medical images, etc.

As any machine learning task, anomaly detection can be supervised or unsupervised. In this work, we are interested only in the unsupervised methods, and we will further refer to them as just anomaly detection. Unsupervised anomaly detection presents the problem of finding patterns in data that do not conform to the common behavior.

Anomaly detection is commonly formulated as two-class recognition problem, where the first class (usually annotated as 0 or negatives) corresponds to normal entities and the second one (usually annotated as 1 or positives) presents some anomalous behavior. Unlike in classic two-class detection problem, anomaly detection presumes, firstly, that the anomalous objects are less numerous than the normal ones and, secondly, that they do not necessary share the same properties with each other.

We formulate anomaly detection as follows: given a dataset D with S entities $x = \{x_1, ..., x_i, ..., x_S\}$, we build a model M that describes D the best. Then we choose some similarity measure that defines how much each entity x_i fits to M. This similarity measure can be a profitability $p(x_i)$, if M is a distribution, distance to other points of the model, points density within a certain radius, distance to the model mean, etc. Figure 2.8 illustrates a data model with two well defined big clusters c_1 and c_2, while all other points can be considered as anomalies as they are not located in the areas of high point density.

To this end, anomalies can be divided into several groups depending on different criteria. First of all, anomalies can be *global* and *local*. Global anomalies, such as x_1

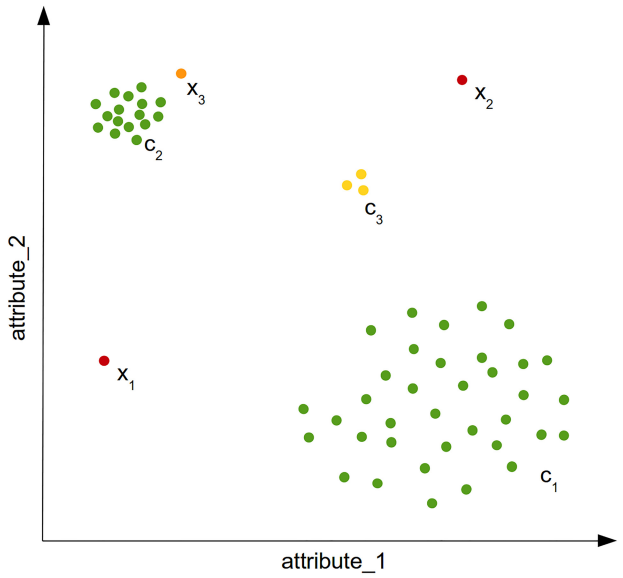

Figure 2.8: A simple two-dimensional anomalies example [1]. It illustrates global anomalies (x_1, x_2), a local anomaly x_3 and a micro-cluster c_3.

and x_2 (see Figure 2.8), can easily be distinguished by eye and are very different from the rest of the dataset. Local anomalies, such as x_3, can be seen as a normal record since it is not too far away from the cluster c_2. However, if we focus only on the cluster c_2 while ignoring any other entities, x_3 can be interpreted as anomalous. Therefore, x_3 is a local anomaly of c_2. As it can be seen, the concept of anomalies is subjective for each dataset and some similarity threshold should be set. For instance, a *micro-cluster* c_3 can be interpreted as a group of anomalies or as normal points of the dataset.

We can equally divide anomalies based on their "uniqueness" [1]:

- *point anomaly* - a single instance that does not share properties with any other dataset entity (x_1 or x_2);
- *collective anomaly* - anomalous situation is represented as a set of many instances, the size of this set is still small comparatively to the dataset size (micro-cluster c_3);
- *contextual anomaly* - describes the effect that a point can be seen as normal, but when a given context is taken into account, the point turns out to be an anomaly. For example, $+30°C$ is a normal temperature in summer, but an outlier in winter.

Anomalies can be detected in time-invariant data (as in the early mentioned example), as well as in multi-temporal data, where each data point x_i of the previously

mentioned dataset D is presented by a sequence $x_i = \{x_{i,1}, ..., x_{i,j}, ..., x_{i,S}\}$ of a certain length S. In the case of the multi-temporal change detection, the whole sequence x_i can be an anomaly, its subsequence or even a single point of this sequence $x_{i,j}$. Figure 2.9 shows different anomalies in time series data: x_1, x_2, x_3 are the normal examples (only 3 entities are given to simplify the figure), x_4 and x_5 contain a one time anomaly and an anomalous subsequence respectively, and the whole sequence x_6 is an anomaly. In multi-temporal anomaly detection, a sequence or its part is marked as anomaly when its behavior does not confirm to the previously detected temporal trend of the sequence itself and/or the dataset.

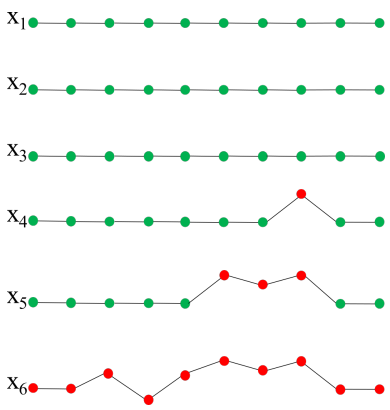

Figure 2.9: An example of anomalies in multi-temporal data. x_1, x_2, x_3 are the normal examples, x_4 contains a one time anomaly, x_5 contains an anomalous subsequence and the whole sequence x_6 is an anomaly.

While we can easily list the most used clustering algorithms, no such thing can be done for the anomaly detection. Anomalous data is very diverse, therefore, we have numerous algorithms for its detection that have to be adapted according to the domain, the type of data, its size and quality, etc. More information about general anomaly detection can be found in [1, 65–67].

2.5 Quality Indices

Evaluating the quality of clustering and anomaly detection is a difficult task that has been an active research area for years, with new methods being proposed on a regular basis.

We usually refer to anomaly detection as to a two-class partition problem, so in most cases we use validation data to assess the quality of the results. However, clustering results are more difficult to analyze for several reasons. The main difficulty with the evaluation of clustering results lies in the inherent unsupervised nature of

clustering itself and the lack of consensus about what a *"good clustering"* should be. In this context, evaluating a clustering result is always more or less subjective, with each evaluation criterion favoring one concept of a *good clustering* (shape, compactness separation, etc.) over the others. Therefore the notions of *good clustering* and *best clustering* will depend on both the evaluation criterion and the clustering algorithm, with some evaluation criterion favoring some algorithms over others.

Still, despite this assumed relative subjectivity, there are a wide range of evaluation criterion that are commonly used in machine learning to assess and compare clustering results. There are several taxonomies available in the literature for these evaluation criteria [68–70], most of them defining 3 distinct groups:

- *Unsupervised indexes*, also called internal indexes: they only use internal information from the data as well as the clusters' characteristics.
- *Supervised indexes*, also called external indexes: they assess the degree of similarity between a clustering solution and a known partition of the data set (sometimes called a ground truth).
- *Relative indexes*: they are a separate class of criteria that make it possible to compare several clustering results of the same algorithm. Relative indexes simply use both external and internal criteria to choose the best solution among several proposed partitions.

In this thesis, we dispose of the ground truth for our algorithms, therefore, we exploit only supervised indices. We detail different supervised indices used in our work below. A non-exhaustive list of unsupervised indices is available in Appendix B.

2.5.1 Supervised Indexes

Whenever the real objects classes are known, it is possible to compare the result of a clustering with the real partition. While these external criteria are not proper clustering indexes, it is the most convenient way to evaluate a new clustering algorithm by applying it to a data set for which the classes or clusters are known. Indexes that makes it possible to rate a clustering based on the comparison between a clustering partition and the real classes are called supervised indexes or external criteria, because they rely on information that comes neither from the data nor from the clustering but from an external ground truth used for comparison purposes.

Confusion Matrix is a table that presents summary of prediction results on a classification problem 2.2. More often, for unsupervised learning, the confusion matrix is built for two class anomaly detection results. The confusion matrix can be equally used for a multi-class problem, however, for an unsupervised approach, we would have to associate the obtained clusters to the ones provided by the ground data, which is not always possible or complicated when we have too many clusters in the resulting partition.

Here we give an example of a confusion matrix computed for a binary classification problem. Given a set of n elements in dataset D, ground truth (GT) labels $\boldsymbol{C}_{GT}$ and the predicted clustering partition $\boldsymbol{C}_{Pr}$, the confusion matrix is presented as follow (Table 2.2).

Table 2.2: Confusion matrix for a binary classification problem.

		Predicted	
		P	**N**
GT	**P**	TP	FN
	N	FP	TN

Two researched classes are denoted as positives (P) (or 1) and negatives (N) (or 0). Columns and rows values of the confusion matrix correspond to the number predicted and actual values respectively. The diagonal values of the matrix correspond to the correctly predicted elements - True Positives (TP) and True Negative (TN) values. The higher are these values, the better are the results. The off-diagonal values stock the incorrectly predicted results - False Positive (FP) and False Negative (FN) values. Obviously, $TP + TN + FP + FN = n$.

Knowing the values of the elements of the confusion matrix, we can compute the following metrics to estimate the quality of the classification.

Accuracy defines how many entities from the whole dataset were correctly clustered:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{TP + TN}{n} \tag{2.8}$$

However, the accuracy value is not an informative metric, especially, for the unbalances partitions. Usually, for the binary classification problem, the “true” class is much smaller. To determine the quality of the partition, it is recommended to compute at least Precision and Recall values.

Precision defines the purity or how many entities that are labeled as positive are indeed positive (the smaller is FP - the better are the results).

$$Precision = \frac{TP}{TP + FP} \tag{2.9}$$

Recall indicates if the class is correctly recognized (the smaller is FN - the better are the results).

$$Recall = \frac{TP}{TP + FN} \tag{2.10}$$

Both Precision and Recall vary between 0 and 1, with 1 being the best value

F1 score also known as the F-Mesure, or balanced F Score is the harmonic mean between the precision and the recall and is computed as follows:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{2.11}$$

The F1 score values also range in $[0, 1]$, with 1 being the best value.

In remote sensing, the most used quality coefficient based on the confusion matrix is Cohen's Kappa ($\mathcal{K}$) score as it is not biased as F1 score.

Cohen's Kappa coefficient ($\mathcal{K}$) is a coefficient mostly used to estimate the quality of supervised learning or anomaly detection. $\mathcal{K}$ is more robust than any other index as it is takes into account the possibility of the agreement occurring by chance. $\mathcal{K}$ is a statistic that is used to measure inter-rater reliability (and also intra-rater reliability) for qualitative (categorical) items and is defined as follows:

$$\mathcal{K} = 1 - \frac{1 - p_0}{1 - p_e} \tag{2.12}$$

with $p_0 = \frac{TP+TN}{n}$ and $p_e = \frac{1}{n^2}(TP+FN)\cdot(TP+FP)+(FP+TN)\cdot(FN+TN)$, where p_0 is the empirical probability of agreement on the label assigned to any sample (the observed agreement ratio), and p_e is the expected agreement when both annotators assign labels randomly.

The Kappa coefficient ($\mathcal{K}$) takes its values between -1 and 1, with 1 corresponding to the best clustering results.

When dealing with multi-class problem with C classes, confusion matrix will be a square matrix of size $C \times C$, each i, j-th element is computed as $\omega_{i,j} = C_{GT_i} \cap C_{Pr_j}$, where C_{GT_i} is the i-th GT cluster and C_{Pr_j} is j-th predicted cluster. We usually do not use Precision, Recall and F1 score for a multi-class problem. On the other hand, $\mathcal{K}$ score is the most popular metric to estimate the multi-class classification

results, but mostly for the supervised methods. As we have mentioned early, it can be difficult to associate the predicted clusters to the GT classes, as the results of the unsupervised methods are often far from the expected for the real-life datasets. For this reason, the special coefficients that to do not demand one-to-one class association were introduced. We present the most popular of them below.

Normalized Mutual Information (NMI) [71] calculates the information shared by two partitions $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ - a partition obtained from a clustering algorithm and the ground truth labels. NMI can also be used to estimate the similarity between two different clustering partitions. This index is defined as:

$$\mathrm{NMI}(C_1, C_2) = \frac{\mathrm{I}(C_1, C_2)}{\sqrt{\mathrm{H}(C_1), \mathrm{H}(C_2)}} \tag{2.13}$$

where

$$\mathrm{I}(C_1, C_2) = \sum_{c_j \in \mathcal{C}_1} \sum_{c'_j \in \mathcal{C}_2} p_{(C_1,C_2)}(c_j, c'_j) \log \left(\frac{p_{(C_1,C_2)}(c_j, c'_j)}{p_{C_1}(c_j)\, p_{C_2}(c'_j)} \right)$$

$$\mathrm{H}(C_i) = -\sum_{c_j \in \mathcal{C}_i} p_{C_i}(c_j) \log p_{C_i}(c_j)$$

In these equations $\mathrm{I}(C_1, C_2)$ denotes mutual information between two partitions and $\mathrm{H}(C_i)$ corresponds to partition entropy, where $p_X(x)$ is the probability function.

NMI considers every cluster c_j of a computed partition $\boldsymbol{C}_1$ and the reference partition $\boldsymbol{C}_2$. In other words, NMI compute the number of common entities between $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$.

The values of NMI vary between 0 and 1, with 1 corresponding to two identical partitions.

Rand Index (Rand) and Adjusted Rand Index (ARI) The Rand Index [72] takes its values between 0 and 1 and is equal to 1 if the two partitions $\boldsymbol{C}^1$ and $\boldsymbol{C}^2$ are identical. Given a set of n elements in dataset D and two partitions $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ to compare, Rand index is defined as follow:

$$Rand = \frac{a+b}{a+b+c+d} = \frac{a+b}{\binom{n}{2}} \tag{2.14}$$

where

a is the number of pairs of elements in D that are in the same subset in $\boldsymbol{C}_1$ and in the same subset in $\boldsymbol{C}_2$,

b is the number of pairs of elements in D that are in different subsets in $\boldsymbol{C}_1$ and in different subsets in $\boldsymbol{C}_2$,
c is the number of pairs of elements in D that are in the same subset in $\boldsymbol{C}_1$ and in different subsets in $\boldsymbol{C}_2$,
d is the number of pairs of elements in D that are in different subsets in $\boldsymbol{C}_1$ and in the same subset in $\boldsymbol{C}_2$.

The adjusted Rand index [73] is the corrected-for-chance version of the Rand index. The adjusted Rand Index can be formulated as follows shown in Equation (2.15) bellow:

$$ARI = \frac{Rand - Exp(Rand)}{max(Rand) - Exp(Rand)} \tag{2.15}$$

where

$$Exp(Rand) = \frac{\pi(C_1) \cdot \pi(C_2)}{n(n-1)/2}$$

$$max(Rand) = \frac{1}{2}(\pi(C_1) + \pi(C_2))$$

with $Exp(Rand)$ and $max(Rand)$ - expected and maximum Rand index, $\pi(C_1)$ and $\pi(C_2)$ are numbers of pairs of the entities regrouped in the same cluster in C_1 and C_2 respectively.

Unlike the original Rand index, the Adjusted Rand Index can have negative values. However, the adjusted Rand Index still is equal to 1 if and only if the two partitions $\boldsymbol{C}^1$ and $\boldsymbol{C}^2$ are identical.

2.6 Discussion

In this section, we have presented the concept of Machine Learning as well as different types of ML algorithms. We have mostly focused on the unsupervised ML methods such as clustering and anomaly detection and their quality indices.

In this thesis, we develop different methods for SITS change detection and clustering. We exploit hierarchical agglomerative clustering algorithm due to its ability to build a unique model to analyze cluster data at different levels. Contrary to other clustering algorithms, HCA model does not rely a researched number of clusters or a complex set of parameters that will further define the clusters number. During the algorithm execution, data points are presented as individual clusters and then, at every step, the model iteratively merges the two clusters with the highest likelihood value. At the end, the user should simply choose the clustering level that corresponds the best to the desired clustering partition.

We made this choice, because an output of a clustering algorithm often does not correspond to the desired classes as some of them might be merged or, on the contrary, divided into several new ones. For this reason, a user might build several models in order to find an optimal number of clusters for the desired output partition which in not necessary when the hierarchical clustering is used.

Considering the change detection, we develop the algorithms both for bi- and multi-temporal data. Firstly, we propose a bi-temporal change detection algorithm that is based on the feature translation between each couple of two consecutive images of the dataset. The described method detects only the non-trivial changes (changes free of seasonal trend presented between two images), ignoring all the non-change areas and the seasonal trend prevailing between two images. The detected non-trivial changes can be seen as contextual anomalies. Secondly, we analyze the detected bi-temporal changes in the multi-temporal context by introducing some logical constraints.

In the next chapter, we are going to overview different feature extraction methods used in satellite image processing and in our work in particular. The feature extraction is an important part of many remote sensing algorithms. Often, to obtain better results, we need to firstly process the image data, prior to cluster it. In the following chapters (5 and 6), we show that feature extraction can improve the results of clustering algorithms, as we exploit new less noisy entities comparatively to the original data. Moreover, feature extraction and translation lay at the core of our change detection algorithms.

Chapter 3

Feature Extraction using Deep Learning Techniques

Contents

3.1 Introduction

In the previous chapter, we have given an overview of different clustering and anomaly detection methods. However, in real-life applications, we often cannot apply these algorithms directly to unprocessed data.

First, if the data have too many features, we may experience the curse of dimensionality problem. Second, in image processing and particularly in remote sensing,

a pixel value is often not informative and some textural information about its neighborhood or its associated segment is needed.

The "*curse of dimensionality*" is a term introduced in 1961 by Bellman referring to the problem of the explosive increase in data volume associated with adding extra dimensions in a mathematical space. While computer algorithms can grasp information in higher dimensional spaces than humans, past some point, they too have trouble extracting meaningful information.

Feature extraction is one of the methodologies of dimensionality reduction or dimension reduction process which aims to find better and more generalized data representation by projecting it in lower-dimensional space. Let Let $X = \{X_1, ..., X_D\}$ be a set of D features of a dataset. There are two main methodologies for dimension reduction:

– **Feature selection:** Choosing a subset of M features from all the features.

$$\begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ \\ X_D \end{bmatrix} \xrightarrow{\text{feature selection}} \begin{bmatrix} X_{i_1} \\ X_{i_2} \\ \\ X_{i_M} \end{bmatrix}$$

– **Feature extraction:** Creating a small set of new features $Y = \{Y_1, ..., Y_M\}$ by combinations of the original ones.

$$\begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ \\ X_D \end{bmatrix} \xrightarrow{\text{feature extraction}} \begin{bmatrix} Y_1 \\ Y_2 \\ \\ Y_M \end{bmatrix} = f\left(\begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ \\ X_D \end{bmatrix}\right)$$

Feature extraction consists in building new features from the original ones with one or several of the following goals:

– having a lower number of features while keeping a maximum of information,

– having better features with which the data are easier to process,

– having feature with which the data are easier to visualize.

We can divide all feature extraction methods in linear and non-linear combination of features. The most known linear methods are Principal Component Analysis (PCA) [74], Linear Discriminant Analysis (LDA) [75], Multi-Dimensional Scaling (MDS) [76]. These methods have low computational cost and work well when the data are not complex. Non-linear methods are usually used for more complex data structures and a wide variety of the algorithms, such as isometric feature mapping (Isomap) [77], Locally Linear Embedding (LLE) [78], spectral clustering [61], autoencoders (AEs) [79] and some supervised methods (for example, pretrained neural networks [80, 81]).

Some of feature extraction methods are specific only for image processing as they extract spatio-spectral features (textures). The texture extraction methods do not aim to reduce the data dimentionality, but to mine some hidden information about the study area. The textures are extracted either for the whole image, either for windows of specific size -also called patches- associated to each image pixel. In the first case, we obtain a single value for the whole image. In the second one, we build a new texture image of the same width and height as the initial one. The most known set of image textures was proposed by Robert M. Haralick [82] and is still used in many different domains of image processing.

Finally, different spectral indices, such as Normalized Difference Vegetation Index (NDVI) [83], can be extracted from remote sensing images. They are computed pixel-wise for the whole image and combine the values of different spectral bands. These indices can give us the information whether some phenomena is presented in a pixel (vegetation, water, moisture, burn area, etc) and in what quantity.

However, the analyzed data is becoming more and more complex and often classical approaches can not correctly model them. Moreover, no conventional methods are available for feature extraction from image time series. For these reasons, deep learning techniques that are able to analyze complex patterns in the data have become popular.

In this chapter, we are going to overview deep learning feature extraction methods and detail the ones used in our research. Section 3.2 gives a brief explanation of the deep learning concept. Section 3.3 reviews unsupervised deep learning feature extraction methods used in image analysis and Section 3.4 describes different elements of neural network structures.

Some traditional feature extraction methods used in remote sensing can be consulted in Appendix C.

3.2 Deep Learning

In the last years, with growing performance and memory capacity of computers for a lower price, it has become possible to deploy more powerful and complex deep learning algorithms for different machine learning tasks. Deep learning models are composed of several layers that iteratively extract different levels of features from the raw input data. Deep learning algorithms are able to non-lineary map the dependencies in hidden data structure, therefore, the accuracy of many ML applications was drastically increased.

The most popular deep learning models are based on *neural networks* (NN). Neural networks are a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs (Dr. Robert Hecht-Nielsen). In other words, neural networks are modeled like a human brain and contain one or more layers of connected neurons (nodes) that are able to recognize patterns.

Nowadays, neural networks are widely used in almost any research area. A great number of applications exploit video and image analysis: medical image analysis for disease detection and analysis of its evaluation [84], face recognition for security [85], video analysis for autonomous driving [86], gesture recognition [87], handwriting recognition [88] etc.

Deep neural networks for image analysis have many advantages comparatively to traditional methods. First, deep NNs can deal with any type of the input data - from a single channel image to a time series of multispectral images or a video - and preserve relations between data in each dimension. Second, the features extracted with NNs are more robust: while traditional methods exploit a fixed set of rules for feature extraction, NN model is built using the optimization algorithm which allows to find the features that describe the best the data. Finally, NNs are less sensitive to different image deformations such as noise, scaling, rotation, pixel shift etc.

We can roughly divide all NN models in supervised and unsupervised (as in Chapter 2, see Figures 2.1, 2.2). The supervised models are built by mapping the input to some labeled target (image $\rightarrow$ label, image $\rightarrow$ segmentation map, time series $\rightarrow$ label, etc); the trained model is then used to make the prediction on some new unlabeled data. On the contrary, the unsupervised models are built in order to find some meaningful patterns in the unlabeled data. These models are usually made in the form of autoencoders and use the same data as the input and the target.

3.3 AutoEncoders in Image Processing

Among the different neural network models, autoencoders have found application in many domains. In image processing, autoencoders are widely used for image segmentation [2, 89], image compression [90], image reconstruction [91], for feature extraction [92, 93] and clustering [79, 94].

Despite the variety of presented models, only the ones which have the same input and output are considered to be "traditional" AEs (Figure 3.1), like the ones used for feature extraction and clustering.

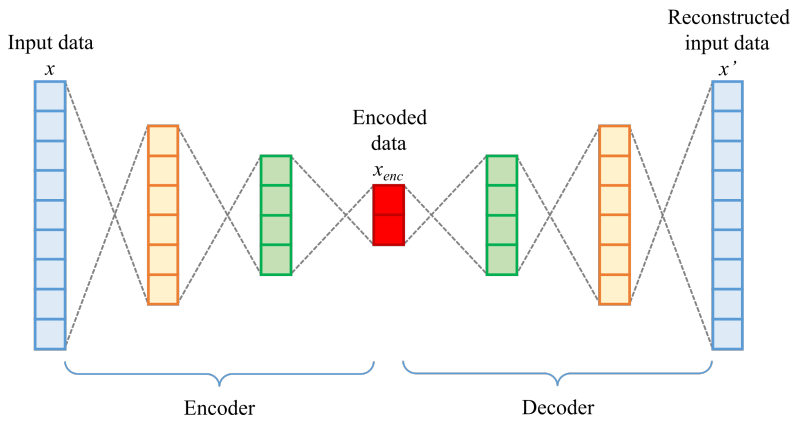

Figure 3.1: Example of a "traditional" AE structure.

While these "traditional" models are totally unsupervised, some models like ones for image segmentation require some training data. Figure 3.2 presents a SegNet model for urban image segmentation. As can be seen, during the model training, the input image should be mapped to the corresponding ground truth segmentation.

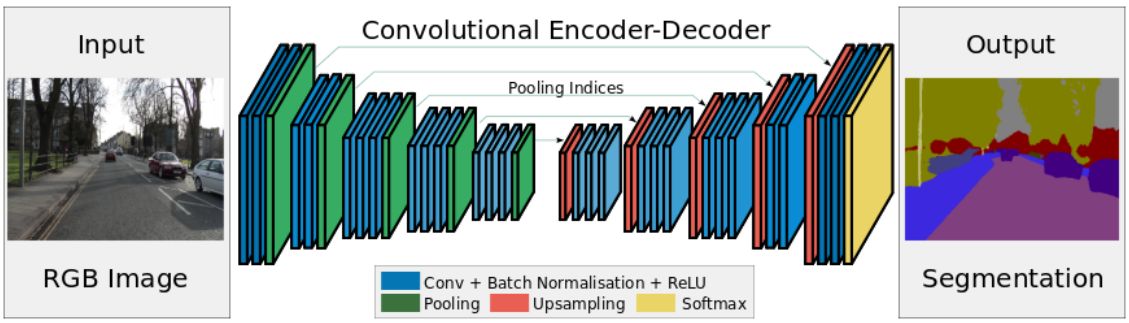

Figure 3.2: SegNet autoencoder model [2].

In this thesis, we will be focusing only on unsupervised feature extraction AE models. The deployment of an AE model ensures the extraction of robust spatial or

spatio-temporal features from the images in an unsupervised manner and is composed of several steps: model training, its eventual validation and the data encoding.

The conventional AE is composed of two parts: the encoder $f_{\theta_e}(\cdot)$ and the decoder $g_{\theta_d}(\cdot)$ (Figure 3.1), where $\theta = [\theta_e, \theta_d]$ is the ensemble of model parameters and θ_e and θ_d are the ensemble of the encoder and decoder parameters respectively. During the training, the model firstly encodes the input data x in some compressed latent representation x_{enc} and then decodes it back to its initial self x'.

$$x_{enc} = f_{\theta_e}(x)$$

$$x' = g_{\theta_d}(x_{enc})$$

The model is optimized in such manner that the reconstructed output resembles as much as possible to the original input data.

The model training performed as follows (note that the following information is not specific to just autoencoders, but can be used to train any type of neural networks [95] with an input data mapped to some target):

1. θ is initialized with some initial random values.
2. The model reconstructs x' from x with given θ.
3. We compute the loss function L (also called "cost function") of x with a chosen algorithm ℓ:
$$L = l(x, x')$$
4. The model parameters θ are optimized to minimize L.
$$\min_{\theta_e, \theta_d}(L)$$
5. We repeat steps 2-4 until the desired L is achieved ($L \to 0$).

Once the model is optimized, the encoder part is used to transform the input data to its encoded version.

Usually, the input data x are chopped in small parts of even size - *batches* - to improve the model optimization. The model parameters are being optimized for each new batch. One iteration through all the batches of the entire dataset is called an *epoch*.

To optimize the model, we use a backpropagation algorithm. The goal of back-propagation is to compute the partial derivatives $\frac{\partial L}{\partial \theta}$ of the loss function with respect

to all θ parameters in the network. These derivatives are called gradient. The model is optimized in such manner that the local minima of L is achieved. For each batch iteration, the backpropagation algorithm updates each of model parameters θ_i in the following manner:

$$\theta_i = \theta_i - \eta \frac{\partial L}{\partial \theta_i} \tag{3.1}$$

where η is the learning rate ($\eta > 0$). The *learning rate* is a hyper-parameter that defines how much we are adjusting the parameters of our model with respect to the loss gradient. Choosing learning rate might be challenging as a value too small may result in a long training time and/or the training can get stuck, whereas if a value too large, the loss function will never achieve the local minima and/or the training process will be too unstable.

In most cases, to estimate model loss of an AE, we use mean square error (MSE):

$$MSE(o,t) = \frac{\sum_{n=1}^{N} l_n}{N},\ l_n = (o_n - t_n)^2 \tag{3.2}$$

where o is the output patch of the model, t is the target patch and N is the number of patches per batch.

Each autoencoder model is composed of "layers" - structures that perform feature extraction at different levels. In image processing, the encoding pass is usually composed of convolutional and pooling layers for feature maps (FM) extraction that are followed by some fully-connected (FC) layers for feature compression. The decoding pass is often symmetrical to the encoding one and uses the inverted layer structure to reconstruct the encoded data in its initial self.

In the next section, we are going to present different types of layers that we used in our work to build different types of autoencoders.

3.4 Neural Networks Structure

Les Net be a neural network composed of R layers $L = [l_1, l_2, ..., l_i, ..., l_R]$, where l_i is an i-th layer. Let $X = [x_1, x_2, ..., x_i, ..., x_R]$ be the ensemble of the inputs of each layer and $X' = [x_2, x_3, ..., x_i, ..., x_{R+1}]$ be the ensemble of the outputs of each layer. The model has sets of learnable weights and biases $w = [w_1, w_2, ..., w_i, ..., w_R]$ and $b = [b_1, b_2, ..., b_i, ..., b_R]$, where w_i and b_i belong to l_i. For a layer l_i, we will have an input x_i and the output x_{i+1}, then x_{i+1} will be an input of l_{i+1}, etc.

In this thesis, we will focus on networks that are able to deal with image data

(single image and time series). Some of the common abbreviations we are going to use in the following subsections include:
B - size of data batches,
C, H, W - image bands (channels), height and width,
D - temporal dimension of the data (depth) (for layers that deal with multi-temporal inputs),
N - data features (for layers that deal with flattened vector data).

3.4.1 Convolutional Layers

In neural network models, convolution layers are used for feature extraction. Three types of convolutional layers exist depending on the shape and type of the input data:

- 1D convolutions - extract textures from vector data, usually used for text analysis as they do not preserve textural information;
- 2D convolutions - extract textures from 2D data preserving spatial information, used in image analysis;
- 3D convolutions - extract textures from image time series or from multi-view images, preserves spatial and depth dimensions.

In this thesis, we are going to use 2D and 3D convolutional layers as well as 3D transposed convolutions which can be considered as a variation of traditional convolutional layers.

A convolutional layer present a set of learnable weights (kernel) and biases applied to the input data. The kernel of size k can be seen as a filter that slides over an input data with a certain step - stride st. The filter is multiplied element-wise to the corresponding subregion of the input data and then the obtained values are summed up.

2D Convolutional Layers

The principle of 2D convolution operation is presented in Figure 3.3. In this example a kernel of size $k = [3, 3]$ slides an image channel with stride $st = [1, 1]$.

As it can be seen from this example, the output data has smaller H and W that the input. To deal with this problem, a padding can be added to the input data. Padding is a artificial augmentation of data size by adding rows and columns with

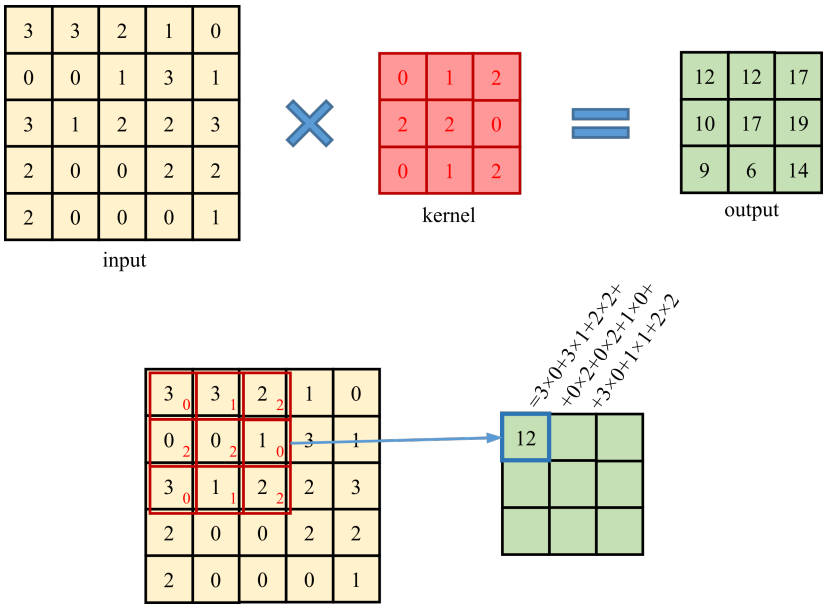

Figure 3.3: Example of a 2D Convolution layer with kernel size $k = [3, 3]$ and stride $st = [1, 1]$.

fixed values (usually zero) to the data (Figure 3.4). If we want to preserve data width and height after convolutions, the padding size *pad* for each data dimension should be computed as follow:

$$pad = \text{int}(k/2) \tag{3.3}$$

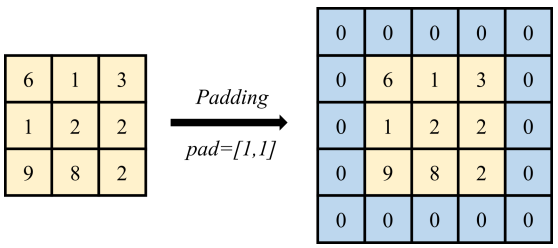

Figure 3.4: Example of zero-padding with $pad = [1, 1]$.

For some applications dilated convolutions may be used. Dilation defines a spacing between kernel elements. Figure 3.5 gives a visual example of dilated convolutions with $dil = 2$ in both dimensions. Note, that when we implement standard convolutions without dilations, $dil = 1$ in all dimensions.

The 2D convolutions can be mathematically written as:

$$x_{i+1}(B_m, C_{\text{out}_j}) = b_i(C_{\text{out}_j}) + \sum_{c=1}^{C_{\text{in}}} w_i(C_{\text{out}_j}, c) \star x_i(B_m, c) \tag{3.4}$$

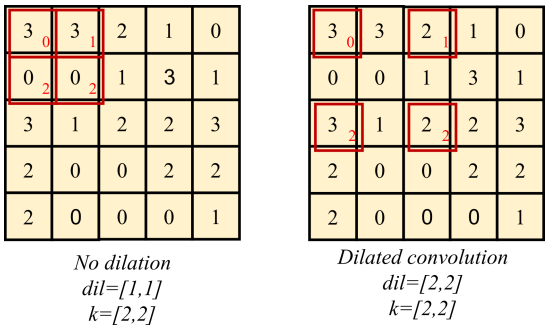

Figure 3.5: Example of "standard" and dilated convolution with $dil = [2, 2]$.

where C_{out_j} is the j-th output channel, B_m is the m-th element of the batch, $\star$ is a cross-correlation operator.

The input and output data have the same dimensions with the following shape: $(B, C_{in}, H_{in}, W_{in})$ for the input and $(B, C_{out}, H_{out}, W_{out})$ for the output, where C_{out} is defined by the user and

$$H_{out} = \left\lfloor \frac{H_{in}+2\times pad[0]-dil[0]\times(k[0]-1)-1}{st[0]} + 1 \right\rfloor,$$
$$W_{out} = \left\lfloor \frac{W_{in}+2\times pad[1]-dil[1]\times(k[1]-1)-1}{st[1]} + 1 \right\rfloor.$$

The shapes of learnable weights w_i and biases b_i are $(C_{out}, C_{in}, k[0], k[1])$ and (C_{out}).

3D Convolutional Layers

While 2D convolutions can capture only spatial textures, 3D convolutions are able to preserve temporal information and extract spatio-temporal features. Kernel, stride, padding and dilation follow the same principle as for 2D convolutions, but have 3-dimensional shape.

The input and output data have the same dimensions with the following shape: $(B, C_{in}, D_{in}, H_{in}, W_{in})$ for the input and $(B, C_{out}, D_{out}, H_{out}, W_{out})$ for the output, where C_{out} is defined by the user and

$$D_{out} = \left\lfloor \frac{D_{in}+2\times pad[0]-dil[0]\times(k[0]-1)-1}{st[0]} + 1 \right\rfloor,$$
$$H_{out} = \left\lfloor \frac{H_{in}+2\times pad[1]-dil[1]\times(k[1]-1)-1}{st[1]} + 1 \right\rfloor,$$
$$W_{out} = \left\lfloor \frac{W_{in}+2\times pad[2]-dil[2]\times(k[2]-1)-1}{st[2]} + 1 \right\rfloor.$$

Mathematically, 3D convolutions can be written exactly as 3D convolutions (see equation 3.4). The shapes of learnable weights w_i and biases b_i are $(C_{out}, C_{in}, k[0], k[1], k[2])$ and (C_{out}).

Transposed Convolutional Layers

Transposed convolution (or deconvolution) layers are used for data upsampling and can be seen as the gradient of the ordinary convolution with respect to its input. The simplest way to think about a transposed convolution on a given input is to imagine such an input as being the result of a direct convolution applied on some initial feature map. The transposed convolution can be then considered as the operation that allows to recover the shape of this initial feature map [96].

Both for 2D and 3D transposed convolutions, the shape of input and output data as well as weights and biases are the same as for the standard convolutions.

More about transposed convolutions as well as about other convolutions can be found in [96].

3.4.2 Pooling Layers

Pooling operations are made to reduce the data size - downsample it. Pooling operations are defined by the size of a pooling filter (kernel) of size k and the stride st (usually its value equal to k). After the filter is applied to the subregion of the input data, it outputs a single value that corresponds to the maximum value of this subregion (MaxPooling) or to the average one (average pooling).

When using AEs structures, the data are downsampled during the encoding pass and then upsampled to the initial size during the decoding pass. To ensure the upsampling operations, we use maximum unpooling (MaxUnPooling) layers that "invert" the MaxPooling operations.

Pooling layers may be categorized in three types - 1D, 2D and 3D - exactly like the convolutional ones and define the number of dimensions of the pooling kernel.

MaxPooling Layers

Maximum pooling (MaxPooling) layer outputs a maximum value withing a kernel subregion. Figure 3.6 presents 2D MaxPooling operation with two-dimensional kernel $k = [2, 2]$ and stride $st = [2, 2]$.

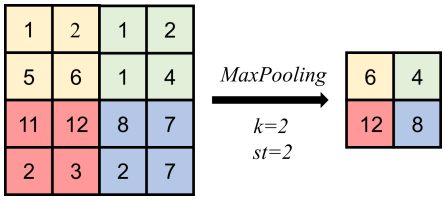

Figure 3.6: Example of a 2D MaxPooling layer with kernel size $k = 2$ and stride $st = 2$.

The input and output shapes should de the following:
For 2D MaxPooling the input and output sizes are (B, C, H_{in}, W_{in}) and (B, C, H_{out}, W_{out}), where
$H_{out} = \frac{H_{in} - k[0] + 2 \times pad[0]}{st[0]} + 1,$
$W_{out} = \frac{W_{in} - k[1] + 2 \times pad[1]}{st[1]} + 1.$
For 3D MaxPooling the input and output sizes are $(B, C, D_{in}, H_{in}, W_{in})$ and $(B, C, D_{out}, H_{out}, W_{out})$, where
$D_{out} = \frac{D_{in} - k[0] + 2 \times pad[0]}{st[0]} + 1,$
$H_{out} = \frac{H_{in} - k[1] + 2 \times pad[1]}{st[1]} + 1,$
$W_{out} = \frac{W_{in} - k[2] + 2 \times pad[2]}{st[2]} + 1.$

Average Pooling Layers

Average pooling (AvgPooling) layer outputs an average value withing a kernel subregion. Figure 3.7 presents 2D AvgPooling operation with two-dimensional kernel $k = [2, 2]$ and stride $st = [2, 2]$.

The input and output shapes are the same as for the MaxPooling layers.

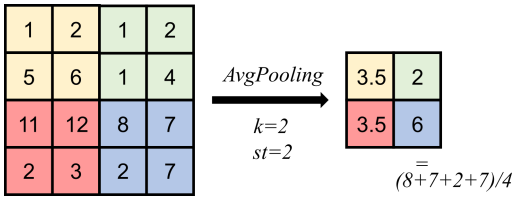

Figure 3.7: Example of an 2D AvgPooling layer with kernel size $k = 2$ and stride $st = 2$.

MaxUnPooling Layers

Maximum UnPooling (MaxUnPooling) layers are usually used in AEs models and are paired with MaxPooling layers. The shape of the input and output of MaxUnPooling layer should be equal to the shape of the output and and input of paired MaxPooling layer. In this case, during the encoding, we extract the position indices of maximum values of the MaxPooling layer. Then during the decoding pass, these indices are used to reconstruct the upsampled data, so that the input values take the position of these indices, other output values are set to zero.

Figure 3.8 presents 2D MaxUpPooling operation with two-dimensional kernel $k = [2, 2]$ and stride $st = [2, 2]$. This operation is paired with the one presented in Figure 3.6.

Note that unpooling operations can be paired only with MaxPooling layers.

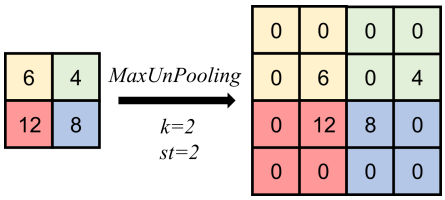

Figure 3.8: Example of a 2D MaxUnPooling layer with kernel size $k = 2$ and stride $st = 2$.

3.4.3 Fully-Connected (Linear) Layers

Fully-connected (FC) layers (also called linear or dense) connect every neuron in one layer to every neuron in another layer. FC layers can also be seen as vector-matrix multiplication:

$$x_{i+1} = x \times w_i^T + b_i \tag{3.5}$$

The shape of the input data should be (B, N_i), where B is number of training elements in a batch and N_i is the number of input vector features. The output shape will be (B, N_{i+1}), where N_{i+1} is the number of the output vector features defined by user. In this case, the sizes of learnable weights and bias w_i and b_i will be (N_{i+1}, N_i) and N_{i+1} respectively.

3.4.4 Activation Functions

Activation functions are mathematical equations that determine the output of a neural network. They are crucial as they introduce non-linear properties to the models. Activation functions are usually applied after convolutional and linear layers (sometimes except for the final output layers) to determine whether their neurons should be activated ("fired") or not when passed to the next layer. The activation function can influence the model training and its accuracy.

The input of activation functions can have any shape as the activations are applied element-wise which gives us the output of the same shape as the input.

ReLU

The Rectified Linear Unit (ReLu) is the most popular activation function in NNs. It output the input directly if is positive, otherwise, it will output zero.

$$\text{ReLU}(x) = \max(0, x) \tag{3.6}$$

Leaky ReLU

Leaky ReLU has the same principle as ReLU activation function, thought it gives a small slope for negative values, instead of altogether zero.

$$\text{LeakyReLU}(x) = \begin{cases} x, & \text{if } x \geq 0 \\ \text{negative\_slope} \times x, & \text{otherwise} \end{cases} \tag{3.7}$$

where negative slope value should be defined by user and is usually set to 0.01.

Sigmoid

A sigmoid function is a mathematical function having a characteristic "S"-shaped curve or a sigmoid curve. The output of this function is always between 0 and 1. However, sometimes when using sigmoid function, a model can get stuck during training, because if a strongly-negative input is provided, the function output values very near zero.

$$\text{Sigmoid}(x) = \frac{1}{1 + \exp(-x)} \tag{3.8}$$

Hyperbolic Tangent

Hyperbolic tangent activation function (Tanh) has also a sigmoidal shape ("S"-shaped), but instead outputs values that range between -1 and 1.

$$\text{Tanh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3.9}$$

3.4.5 Recurrent Neural Networks

Recurrent neural networks (RNN) were introduced in 1986 by David Rumelhart [97]. This type of networks was developed to process temporal sequences data. Contrary to 3D convolutional layers, RNN is able to deal with temporal data of varying length.

In general, RNN is constructed as follows: let $X = \{x_1, x_2, ..., x_n, ..., x_S\}$ be a sequence composed of S timestamps, where each element x_n is a vector of composed of fixed number of features. For each timestamp T_n, the RNN unit computes a hidden state h_n (Figure 3.9) which represent the accumulative value of the previous hidden states of the sequence. The final hidden state h_S characterizes the whole sequence and is used afterwards for classification or clustering.

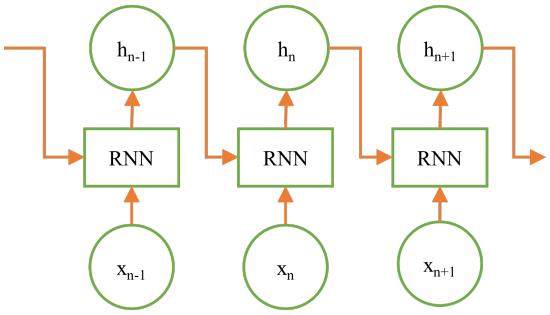

Figure 3.9: RNN model.

The main problem of RNN is that the value of each hidden state h_n depends only on the value of previous hidden state h_{n-1}. For this reason, RNN networks may suffer from a long-short memory problem caused by vanishing gradient, hence, they do non consider long term dependencies. To solve this issue, more complex Long Short-Term Memory (LSTM) networks were introduced in 1997 [98]. Contrary to RNN, LSTM contains input, output and forget gates as well as memory cell c_n at each timestamp that makes it possible to retain meaningful information from the previous steps and, as a consequence, the value of h_n depends on all the previous hidden states of the sequence and not only on h_{n-1}.

Later, to facilitate the computation and implementation of the LSTM model, GRU networks were developed [99] for Natural Language Processing (NLP) tasks. GRU contains only update and reset gates that allows model to train faster with less memory consuming. GRU were successfully adapted for remote sensing applications and proved to be more efficient than LSTM networks in this research area as they give better or similar results to LSTM using less training parameters and time [100–102].

Originally RNNs were not able to capture spatial information. However, later convolutional RNNs were introduced [103]. This networks are based on principle of combining convolutional operations with any type of presented networks (RNN, LSTM or GRU). They are able to process video or temporal image data, as each element of the sequence x_n, as well as its corresponding hidden state h_n is presented by a 3D data. The compexity of the network of the number of the operations make these networks very slow for big datasets. That makes it difficult to find the right network configuration, especially for an unsupervised implementation. For this reason, we do not use these models in our research.

As can be seen from Figure 3.9, one RNN layer is composed of S RNN units. The following subsections will present in details units of each types of RNN.

Recurrent Networks Unit

Each RNN unit takes 2 inputs - the sequence value x_n at the timestamp T_n and the hidden value obtained for the previous timestamp h_{n-1}. For the first timestamp, the initial hidden state h_0 is initialized to zeros.

Mathematically a RNN unit can be written as:

$$h_n = \tanh(w_{ih}x_n + b_{ih} + w_{hh}h_{(n-1)} + b_{hh}) \tag{3.10}$$

where w_{ih} and w_{hh} are learnable input-hidden weights (to pass from the input to the hidden state) and hidden-hidden weights respectively and b_{ih} and b_{hh} are the corresponding learnable biases.

For each RNN unit, shape of the input data and the corresponding input/output hidden state are (B, N_x) and (B, N_h), where N_x is the number of features of the input sequence and N_h is the number of features of hidden state defined by user.

The shapes of all learnable weights and biases are equal to (N_h, N_h) and (N_h) respectively.

Long Short-Term Memory Unit

Contrary to recurrent units, a LSTM unit contains several gates: input i_n, forget f_n and output o_n gate. Moreover, in addition to the hidden states h_n, LSTMs equally have cell states c_n. Each cell state c_n stores the information about previous cell states of the model.

Mathematically a LSTM unit can be written as:

$$\begin{aligned}
i_n &= \sigma(w_{ii}x_n + b_{ii} + w_{hi}h_{n-1} + b_{hi}) \\
f_n &= \sigma(w_{if}x_n + b_{if} + w_{hf}h_{n-1} + b_{hf}) \\
g_n &= \tanh(w_{ig}x_n + b_{ig} + w_{hg}h_{n-1} + b_{hg}) \\
o_n &= \sigma(w_{io}x_n + b_{io} + w_{ho}h_{n-1} + b_{ho}) \\
c_n &= f_n \odot c_{n-1} + i_n \odot g_n \\
h_n &= o_n \odot \tanh(c_n)
\end{aligned} \tag{3.11}$$

where g_n is an intermediate cell gate, σ is the sigmoid function (3.8), tanh is the hyperbolic tangent function (3.9),$\odot$ is the Hadamard product,
w_{ii}, w_{if}, w_{ig}, w_{io} is the ensemble of learnable input-hidden weights,
b_{ii}, b_{if}, b_{ig}, b_{io} are their corresponding learnable biases,

w_{hi}, w_{hf}, w_{hg}, w_{ho} is the ensemble of learnable hidden-hidden weights,
b_{hi}, b_{hf}, b_{hg}, b_{ho} are their corresponding learnable biases.

For each LSTM unit, shape of the input data and the corresponding input/output hidden state and input/output cell state are (B, N_x), (B, N_h) and (B, N_h) respectively, where N_x is the number of features of the input sequence and N_h is the number of features of hidden state defined by user.

The shapes of the ensemble of each set of learnable weights and biases are $(4 \times N_h, N_h)$ and $(4 \times N_h)$ respectively.

Gated Recurrent Unit

GRU is a computationally more efficient variant of LSTM with only reset r_n and update z_n gates that to stock the internal state of the model.

Mathematically a GRU unit can be written as:

$$\begin{aligned} r_n &= \sigma(w_{ir}x_n + b_{ir} + w_{hr}h_{(n-1)} + b_{hr}) \\ z_n &= \sigma(w_{iz}x_n + b_{iz} + w_{hz}h_{(n-1)} + b_{hz}) \\ n_n &= \tanh(w_{in}x_n + b_{in} + r_n * (w_{hn}h_{(n-1)} + b_{hn})) \\ h_n &= (1 - z_n) * n_n + z_n * h_{(n-1)} \end{aligned} \tag{3.12}$$

where n_n is an intermediate new gate parameter, σ is the sigmoid function (3.8), tanh is the hyperbolic tangent function (3.9), $\odot$ is the Hadamard product,
w_{ir}, w_{iz}, w_{in} is the ensemble of learnable input-hidden weights,
b_{ir}, b_{iz}, b_{in} are their corresponding learnable biases,
w_{hr}, w_{hz}, w_{hn} is the ensemble of learnable hidden-hidden weights,
b_{hr}, b_{hz}, b_{hn} are their corresponding learnable biases.

For each LSTM unit, shape of the input data and the corresponding input/output hidden state are (B, N_x) and (B, N_h) respectively, where N_x is the number of features of the input sequence and N_h is the number of features of hidden state defined by user.

The shapes of the ensemble of each set of learnable weights and biases are $(3 \times N_h, N_h)$ and $(3 \times N_h)$ respectively.

3.5 Discussion

In this section we have presented different deep learning feature extraction methods used in image processing and remote sensing in particular. Along with the traditional

methods, we have reviewed deep learning feature extraction techniques used in our work. All the presented deep learning methods exploit autoencoder structure, composed of layers able to process different type of data - vector, spatial and spatio-temporal.

Deep learning models have been proved to be the most efficient tool for the feature extraction and classification for the complex data. In our thesis, we exploit 2D convolutional autoencoders for feature translation applied to bi-temporal change detection (Chapter 4). We equally use GRU networks for the clustering of the detected multi-temporal change graphs (Chapter 5). Finally, we exploit a 3D convolutional model for SITS feature extraction and clustering (Chapter 6).

In our work, we equally exploit NDVI index (Chapter 6). We successfully prove that enriching data with NDVI drastically improves the accuracy and helps to distinguish more vegetation clusters comparatively to the raw data analysis.

In the following chapters, we are going to present our contributions to multi-temporal data analysis. All developed methods exploit deep learning feature extraction techniques combined with various data analysis algorithms.

Chapter 4

Bi-temporal Change Detection

Contents

4.1 Introduction

Nowadays, change detection in satellite image time series is required for many different applications. While in some applications we are interested in seasonal changes such as evolution in agricultural parcels, others require the detection of permanent changes such as buildings or roads constructions. Nevertheless, due to image resolution and preprocessing level (most of the SITS do not have a correction of the atmospheric factors), properly detecting changes remains a difficult task.

Most of the existing change detection algorithms are still supervised or semi-supervised, and therefore need some labeled data or a pre-trained model for feature

extraction. Providing the labeled data for remote sensing images and especially SITS is a costly and time-consuming task due to the variance of objects present in them and the time needed to produce the great amount of labeled images required to train large models.

Usually, to properly detect changes in a SITS, the series should be long enough and have a more or less regular temporal resolution to detect different seasonal trends which is not always possible to achieve. In this thesis, we propose to decompose a SITS change detection problem into two steps: we first perform a bi-temporal change detection for each couple of consecutive images Im_n and Im_{n+1}, then, we interpret the obtained changes in the multi-temporal context.

In this thesis, we distinguish three different types of temporal behavior in SITS: no change areas (mostly urban areas), seasonal or trivial changes (mostly presented by vegetation) and non-trivial changes that contain permanent changes such as new constructions, demolishment, permanent crop rotations and some vegetation that does not follow the overall tendency. We consider only non-trivial changes as the interesting ones, hence, these are the changes we aim to detect with our algorithm. In this chapter, we propose an unsupervised approach for bi-temporal change detection that is based on feature translation with neural network autoencoder. In the next chapter, we interpret the detected bi-temporal changes in the multi-temporal context. Finally, in Chapter 6 we propose an algorithm for the clustering of the whole SITS in order to regroup no-change areas and seasonal changes in different clustering partitions.

In the presented bi-temporal change detection approach, we use joint AEs to create models able to reconstruct Im_{n+1} from Im_n and vice versa by learning the image features. Obviously, the non-changed areas and trivial changes such as seasonal ones will be easily learned by the model, and therefore reconstructed with small errors. As the non-trivial changes are unique, they will be considered as outliers by the model, and thus will have a high reconstruction error (RE). Thresholding on the RE values allows us to create a binary change map (CM).

The proposed method has showed promising results on a dataset with high ratio of agricultural areas and outperformed the concurrent approaches. Different joint AE models were tested in order to find the most accurate one for change detection. Our change detection method has a low complexity and gives high quality results on open source high resolution images.

The remainder of this chapter is organized as follows: In Section 4.2, we present the existing algorithms for bi-temporal change detection in satellite images. Section 4.3

details our proposed approach for change detection. Section 4.5 is dedicated to experimental results and some conclusions are drawn in Section 4.6.

4.2 Related Works

Different algorithms for bi-temporal change detection are proposed in the literature. As any other machine learning methods, change detection can be supervised and unsupervised. While the supervised algorithms usually aim to identify a particular change type (i.e. changes in urban areas), the unsupervised algorithms tend to detect "everything that changes" which might complicate the interpretation of the results, especially when images suffer from illumination problems or even belong to different vegetation seasons.

The variety and specificity of change detection applications do not allow to create a unique database with training data. Creating such a database can be time-consuming and costly as every entity should be validated manually. However, some datasets accessible for public use exist.

The most known change detection dataset is the ONERA Satellite Change Detection (OSCD) dataset [104, 105]. The OSCD dataset addresses the issue of detecting changes between satellite images from different dates. It comprises 24 pairs of multispectral images taken from the Sentinel-2 satellites between 2015 and 2018 with the locations picked all over the world. For each location, registered pairs of 13-band multispectral images are provided. Images vary in spatial resolution between 10 m, 20 m and 60 m. Pixel-level change ground truth is provided for 14 of the image pairs. The annotated changes focus on urban areas, such as construction of new buildings or roads.

The main problem is that usually these datasets contain objects issued from a specific satellite mission for a specific study area or/and a specific change type. For this reason, they can be used only for a small number of projects compatible with the dataset purposes.

To overcome this problem, several change detection methods exploiting transfer learning were proposed [106, 107]. In these methods, the authors exploit pretrained neural network models to extract features from a couple of satellite images in order to compare them. In [108], the satellite images are first segmented at three different levels, second, the feature extraction is performed at each level and, finally, the extracted features fused and clustered to detect the change areas. Feature extraction is realized with the VGG-16 network [80] pretrained on the ImageNet dataset [109].

The authors claim that among all the existing pretrained models, only this one can provide weights that can used on satellite images, because they are tasks oriented and cannot be generalized to another dataset [106].

In [107], the authors pretrain a U-Net [89] model for semantic segmentation of urban areas with a publicly available Vaihingen dataset (provided by the International Society for Photogrammetry and Remote Sensing). The pretrained model is then used for feature extraction and change detection on other datasets.

However, transfer learning for change detection has several disadvantages: First, the data under the study should have the same dimensions as the data used to train the model. Second, transfer learning was applied mostly to urban areas and its behavior for natural areas is not known.

As we have mentioned before, in our work, we interest in more general-purpose change detection methods that do not presume that the input data should have any specific properties.

Most traditional unsupervised change detection methods are based on the analysis of a difference image (DI). The DI can be computed differently depending on the application, but usually it is defined as the absolute-valued difference of intensity values of two images:

$$\mathrm{DI} = |Im_2 - Im_1| \tag{4.1}$$

The early change detection methods were based on clustering of the obtained DI in order to isolate the change pixel in a separate cluster. For example, in [110], the Principle Component Analysis (PCA) transformation is used to extract a feature vector for each pixel of DI and its corresponding neighborhood pixels, then k-means with 2 clusters is applied to the resulting transformed DI in order to isolate the changes. In [111], the authors use another strategy for DI transformation into 2D space: the compression is accomplished by computing the magnitude of spectral change vectors (pixels of DI) and its angle (direction) with a reference vector. Then the EM algorithm is applied to new polar coordinates of DI image. The method is able to directly cluster different types of changes. In [112], the authors presume that change and no change pixels of DI follow the Gaussian distributions. Following this idea, they propose an adaptive threshold method based on EM algorithm for distribution modeling that separates the change pixel from the no changes ones. Finally, several Markov Random Field (MRF) models were proposed [113, 114] for change detection. While [114] applies MRF model to refine the results of the initial change map obtained with fuzzy C-means applied to the DI; in [113], the authors firstly estimate the DI pixel distribution with EM algorithm and detect the DI edges,

then the weights of MRF prior energy are adaptively adjusted by considering the gray level differences between neighboring pixels in order to obtain change labels.

The main problem of DI analysis is that the DI is often noisy, especially when two images are not perfectly aligned and/or they have high or very high spatial resolution. The main difficulty with unsupervised approaches to analyze satellite images is that they usually produce lower quality results than supervised ones.

To overcome the problem of missing labels, a set of automatic methods for selection of changed and unchanged pixels was developed [115]. The main idea of these methods lays in a change vector analysis to find thresholds that separate pixels that have the highest probability of being change or no change pixels. These pixels are then chosen as training samples for supervised classification algorithms to identify change areas [116]. However, the detected training samples are often concentrated in one single area which gives biased classification results. To solve this problem, [117] proposes a more adapted method that select training sample with different probabilities. Moreover, the authors of this article propose a multiple classifier system that fuses the results of different algorithms to obtain a better score. Following this paper, the authors of [118] propose the improved backpropagation method of a deep belief network (DBN) for change detection based on automatically selected change labels.

In [119], a regularized iteratively reweighted multivariate alteration detection (MAD) method for change detection was presented. This method is based on linear transformations between different bands of a couple of hyperspectral satellite images and canonical correlation analysis.

However, spectral transformation between multi-temporal bands is very complex. For these reasons, deep learning unsupervised algorithms which are known to be able to model non-linear transformations have proved their efficiency to solve this problem [120].

Nevertheless, classic feature comparison approaches do not separate trivial (seasonal) changes from non-trivial ones (permanent changes and changes that do not follow seasonal tendency). This weakness can drastically complicate the interpretation of change detection results for regions with high ratio of vegetation areas. In fact, when analyzing two images belonging to different seasons of the year, almost all the area will be marked as change and further analysis will be needed to identify meaningful changes (non-trivial).

Our change detection method is based on the approach proposed in [120] (detailed in the next section). In that work, the authors use a Restricted-Boltzmann Machines-

based (RBM) model to learn the transformation model for a couple of VHR co-registered images Im_1 and Im_2. RBM is a type of stochastic artificial network that learns the distribution of the binary input data. In the case of image analysis, the input data is continuous, so Gaussian-Bernoulli RBM (GBRBM) is used [121].

The principle of the proposed method to detect changes is the following: most of the trivial changes can be easily modeled from Im_1 to Im_2, at the same time, non-trivial changes will not be properly reconstructed. Therefore, the reconstruction accuracy can be used to detect the non-trivial change areas. The proposed approach consist of the following steps: feature learning, feature comparison and thresholding. During the feature learning step, the algorithm learns some meaningful features to perform transformation of patches of Im_1 to the patches of Im_2. Once the features are learned by the model, Im_1 is transformed in Im_2'. Then the difference image (DI) of Im_2 and Im_2' is calculated. The same steps are performed to create a DI of Im_1 and Im_1'. The thresholding is then applied on an average DI. Obviously, the areas with high difference values will be the change areas.

For the feature learning, the authors use an AE model composed of stacked RBMs layers GBRBM1-RBM1-RBM2-GBRBM2. The authors indicate that the algorithm is sensitive to changing luminosity and has high level of false positive changes in real change data. To our knowledge, this algorithm was tested only on urban areas.

4.3 Methodology

Our change detection method is similar with [120] introduced in the previous section. However, contrary to RBM models that are based on a stochastic approach and distribution learning, we propose to use a deterministic NN model based on feature extraction. Furthermore, we use patch reconstruction error for every pixel of the image - instead of image difference - as extracting features from every pixel neighborhood is an important step for any eventual subsequent pixel-wise classification task.

In our thesis, we test two NN AE architectures and assess their performance for change detection in order to pick the best adapted one. The tested models are joint fully-convolutional AEs and joint convolutional AEs.

Fully-convolutional AEs consist of a stack of layers that apply different convolutions (filters) to the input data in order to extract meaningful feature maps (FM). Convolutions are often used in image processing as they deal with non-flattened data (2D and 3D). Therefore, unsupervised feature extraction with fully-convolutional AEs has been proved efficient in different remote sensing applications [122]. Convolu-

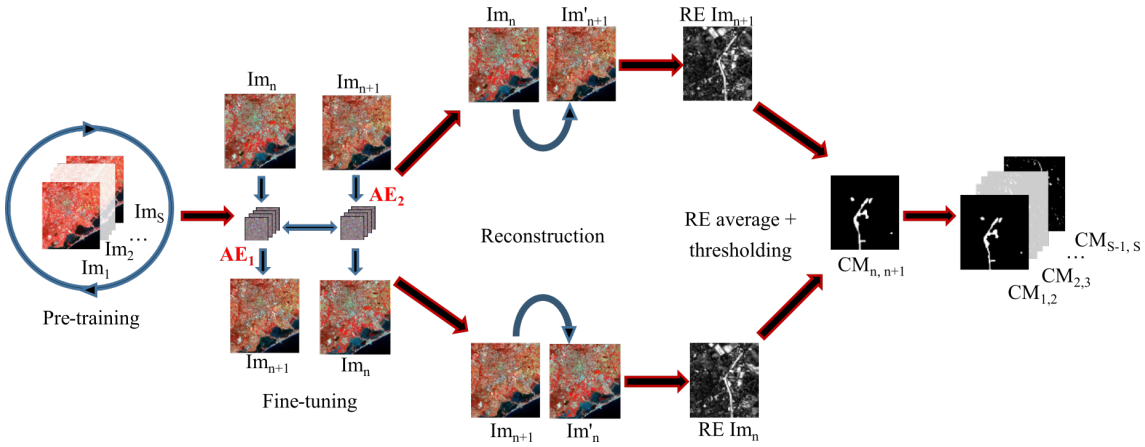

Figure 4.1: Bi-temporal change detection algorithm.

tional AEs equally contain different convolutional layers that are followed by some fully-connected layers to compress the feature maps.

4.3.1 Change Detection Algorithm

Let $Im_1, Im_2, ..., Im_{S-1}, Im_S$ be a SITS made of S co-registered images taken at dates T_1, T_2, ... , T_{S-1}, T_S. In this subsection, we present our change detection algorithm applied to a whole SITS, but, obviously, it can be applied to just a couple of images. The CD algorithm steps are the following (Figure 4.1):

- The preprocessing step consists in the radiometric normalization of the whole dataset [123].
- The first step consists in patch-wise model pre-training on the whole dataset.
- During the second step, we fine-tune the joint AE model for every couple of images (Im_n, Im_{n+1}). Once the model is trained, we calculate the reconstruction error of Im'_{n+1} from Im_n and vise versa for every patch. In other words, the reconstruction error of every patch is associated to the position of its central pixel on the image.
- In the last step, we identify areas of high reconstruction error using Otsu's thresholding method [124] in order to create a binary change map $CM_{n,n+1}$ with non-trivial change areas.

4.3.2 Model Pre-training and Fine-tuning

In our method, we use deep AEs to reconstruct Im_{n+1} from Im_n and vise versa. During the model pre-training, the feature learning is performed patch-wise for a sample extracted from the SITS. We sample $\frac{H \times W}{S}$ patches (H and W are image height and width respectively) from every image to prevent the model from overfitting. The patches for the image border pixels are generated by mirroring the existing ones in the neighborhood. During the encoding pass, the model extracts feature maps (FM) for the fully-convolutional AE (and feature vector for the convolutional AE) of i, j, m-patch of chosen samples, and then during the decoding pass, it reconstructs them back to the initial i, j, m-patch ($i \in [1, H]$, $j \in [1, W]$, $m \in [1, S]$).

The fine-tuning part is done for each couple of images independently and consists of learning two joint reconstruction models AE_1 and AE_2 for every patch of a couple of co-registered images (Im_n and Im_{n+1}). The patches are extracted for every i, j pixel of the images as the local neighborhood of size p *timesp* wherein the processed pixel is the central one (i.e., the image i, j-pixel corresponds to i, j-patch central pixel).

Our joint fully-convolutional AEs model is presented in Figure 4.1). The joint model for the convolutional AE has the same structure, though the bottleneck output is presented by a feature vector. AE_1 and AE_2 have the same configuration of layers as the pre-trained model and are initialized with the parameters it learned. In the joint model, AE1 aims to reconstruct patches of Im'_{n+1} from patches of Im_n and AE2 reconstructs Im'_n from Im_{n+1}. The whole model is trained to minimize the difference between:

- the decoded output of AE_1 and Im_{n+1} ($loss_1$),
- the decoded output of AE_2 and Im_n ($loss_2$),
- the encoded outputs of AE_1 and AE_2 (feature maps for fully-convolutional AEs and bottleneck feature vectors for convolutional AEs) ($loss_3$).

$$modelLoss = loss_1 + loss_2 + loss_3 \tag{4.2}$$

The three losses are not weighted and equally participate in the model optimization.

For our AE models we, use the architectures presented in Table 4.1, where C is the number of spectral bands and ℓ_2-norm is ℓ_2 normalization [125]. All convolution layers, except for the last layer of the decoder part, have batch normalization. The size of different parameters are the same for all convolutional layers: kernel size=3,

stride=1, padding=1. MSE (3.2) is used for model optimization and calculation of the patch reconstruction error.

Table 4.1: Models architecture for bi-temporal change detection.

	Fully-convolutional AE	**Convolutional AE**
encoder	Conv(C,32)*+ReLU Conv(32,32)+ReLU Conv(32,64)+ReLU Conv(64,64)+ℓ_2-norm	Conv(C,32)+ReLU Conv(32,32)+ReLU Conv(32,64)+ReLU Conv(64,64)+ReLU Linear($64 \times p^2, 12 \times p^2$)**+ReLU Linear($12\times p^2$,$2\times p^2$)+$\ell_2$-norm
decoder	Conv(64,64)+ReLU Conv(64,32)+ReLU Conv(32,32)+ReLU Conv(32,C)+Sigmoid	Linear($2\times p^2, 12\times p^2$)+ReLU Linear($12\times p^2, 64\times p^2$)+ReLU Conv(64,64)+ReLU Conv(64,32)+ReLU Conv(32,32)+ReLU Conv(32,C)+Sigmoid

*In Conv(a,b) a and b stand for the number of the input and output channels respectively.
**In Linear(a,b) a and b stand for the sizes of the input and output vectors respectively.

Once the model is trained and stabilized, we perform the image reconstruction of Im'_n and Im'_{n+1} for every patch, and we create two images representing their reconstruction errors. We apply Otsu's thresholding [124] to the average reconstruction error of these images in order to produce a binary change map.

4.4 Data

Our change detection algorithm was applied to couples of images extracted from the SPOT-5 SITS of Montpellier area, France (see Appendix A.1). The initial dataset has four spectral bands, however, for change detection we use only green, red and NIR bands as they contain the most essential information for change detection.

We assess the algorithms performances on two extracts from the SPOT-5 SITS. The image couples were taken between 2004-05-14 and 2005-04-27 for the first extract, and between 2006-02-18 and 2008-08-21 for the second one. Though these couples of images are not consecutive, they contain much more changes than the consecutive ones, hence, it is more interesting to use them to evaluate our algorithm.

We also present the quantitative evaluation of change detection results for the Sentinel-2 SITS of Rostov-on-Don, Russia (see Appendix A.3, later called the Rostov dataset) as both these datasets are used in the next chapter for the multi-temporal change detection. We perform change detection on two couples of the consecutive images taken on 2015-08-30 and 2015-09-19 for the first couple and on 2017-09-18 and 2018-01-11 for the second one. Sentinel-2 images dispose of multiple spectral bands with different spatial resolution, nevertheless, only green, red and NIR bands with 10 m resolution were used. Note that only the Montpellier dataset is used for the analysis of the performance of our methods and their comparison to the concurrent ones.

To evaluate the performance of the proposed methods on two datasets, we have used ground truth maps extracted for different parts of each dataset. For the Montpellier dataset, the overall ground truth area was 600000 pixels, while for the Rostov dataset this area was 780000 pixels.

For the Montpellier dataset, the ground truth maps were created for an extract of the image of size 800×600 pixels (48 km^2) for the first couple and for 320×270 pixels (8,64 km^2) for the second one. However, the change detection was performed on the full images of 1600×1700 pixels (272 km^2).

For the Rostov dataset, the ground truth maps were created for two extracts of size 900×700 pixels for the first couple of images and 500×300 pixels for the second one. While the first GT focuses in changes in crops, the second one captions the changes during the construction of a stadium for FIFA 2018 and its surrounding area. As for the Montpellier dataset, the change detection was performed on the full images of 2200×2400 pixels (528 km^2).

4.5 Experiments

4.5.1 Experimental Settings

As we mentioned previously, we propose different architectures: joint convolutional AEs and joint fully-convolutional AEs, and we compare them in order to assess their strengths and weaknesses. We further compare our approaches with the RBM-based method presented in [120] that we have discussed in Section 4.3 and with an improved RBM method. Initially, in [120] the images are clipped in $\frac{H \times W}{p \times p}$ not overlapped patches. In the improved method, we propose to extract patches with neighborhoods of every pixel of the image ($H \times W$ patches). Equally, we use the

Table 4.2: Architecture of RBM models for bi-temporal change detection.

	RBM	**Improved RBM**
enc.	GBRBM($p^2 \times C, 384$)+Sigmoid RBM($384, 150$)+Sigmoid	GBRBM($p^2 \times C, (p+1)^2 \times C$)+Sigmoid RBM($(p+1)^2 \times C, (p-2)^2 \times C$)+Sigmoid
dec.	RBM($150, 384$)+Sigmoid GBRBM($384, p^2 \times C$)+Sigmoid	RBM($(p-2)^2 \times C, (p+1)^2 \times C$)+Sigmoid GBRBM($(p+1)^2 \times C, p^2 \times C$)+Sigmoid

patch reconstruction error instead of the image difference to detect the changes. In other words, the improved RBM method uses the same steps as in our proposed algorithms, but exploits different type of network layers. The layers parameters of the RBM methods are presented in Table 4.2.

In the experiments, we use the architectures presented in Table 4.1 for our models. Note, that all the methods are evaluated only on the Montpellier dataset and the most appropriate model configuration is later applied for the change detection in the Rostov dataset.

For the Montpellier dataset, we test different patch sizes to estimate their influence on the final results and computation time (presented later in the text). After the experiments, the patch size of 5×5 is chosen for all our methods. Adam algorithm was used to optimize the models. During the pre-training phase, the learning rate was set to 0.0005 and then changed to 0.00005 for the fine-tuning phase.

The RBM model presented in [120] is developed for VHR images, but we have kept the patch size 10×10 pixels and the layer sizes suggested by the authors. In the improved RBM method we use 5×5 pixels patch size as in our methods.

In our approaches and in the improved RBM method, before applying Otsu thresholding we equally exclude $high_v$% of the highest reconstruction error values considering they correspond to some noise and extreme outliers. The variation of the $high_v$ only balances precision/recall values, without influencing *kappa* value. The higher $high_v$ is - the higher is the recall and the smaller is the precision, and vise versa. To balance these values, we set $high_v$ to 0.5% for both datasets. The analysis of the influence of $high_v$ value on different metrics is detailed later in the text.

The following quality criteria were used to evaluate the performances of the different approaches: precision (2.9), recall (2.10) and Cohen's kappa score (2.12).

The patch size of 5×5 pixels was chosen after results estimation for different parameters. The correlation between the patch size and the performance of our algorithms for the Montpellier dataset is shown in Table 4.3. We can observe that

Table 4.3: Algorithm performance based on patch size p for Montpellier images taken in 2004 and 2005 years.

Methods	p×p	Classification performance			
		Precision	***Recall***	***Kappa***	***Time*[a]*, min***
Fully-Conv.	3×3	0.69	0.73	0.70	**11+7**
	5×5	0.67	**0.78**	0.70	15+8
	7×7	**0.69**	**0.78**	**0.72**	19+13
Conv.	3×3	0.67	0.74	0.69	**12+9**
	5×5	**0.68**	0.79	**0.71**	17+12
	7×7	0.61	**0.81**	0.69	24+18

[a]Pre-training+fine-tuning.

$p = 3$ gives us poor results for both models as the patch does not contain enough information about the neighborhood, $p = 7$ gives us slightly better results for fully-convolutional AE than $p = 5$ though learning time is higher. However, we see that for $p = 5$ performance of convolutional AE is much better than for $p = 7$. It can be explained by layer flattening when passing from convolutional layers to linear.

As it was mentioned before, we set $high_v$=0.5% for our method to balance the precision and recall values. The influence of $high_v$ value on the precision, recall and kappa values is presented on Figure 4.2. It was decided that the optimal threshold $high_v$ should provide the results with recall slightly better than precision, as some FP changes will be deleted during the next steps. We observe that for the Montpellier dataset $high_v = 0.5\%$ gives us the best result. At the other hand, we notice different behavior of metrics values for the Rostov dataset ground truth and we can state less smooth changes in these values. We equally state that the best results are obtained for $high_v \geq 0.2\%$. Therefore, it was decided to set constant $high_v = 0.5\%$ for our method to ensure the most optimal results for both datasets.

4.5.2 Results

The Montpellier Dataset

Some change detection results are presented on Figures 4.3, 4.4, 4.5, 4.6, 4.7. All the images are represented in false colors, where red corresponds to vegetation and green to empty fields.

Figure 4.3 features changes in an urban area: several buildings were constructed (or started to be constructed). The images extracts have great change in luminosity

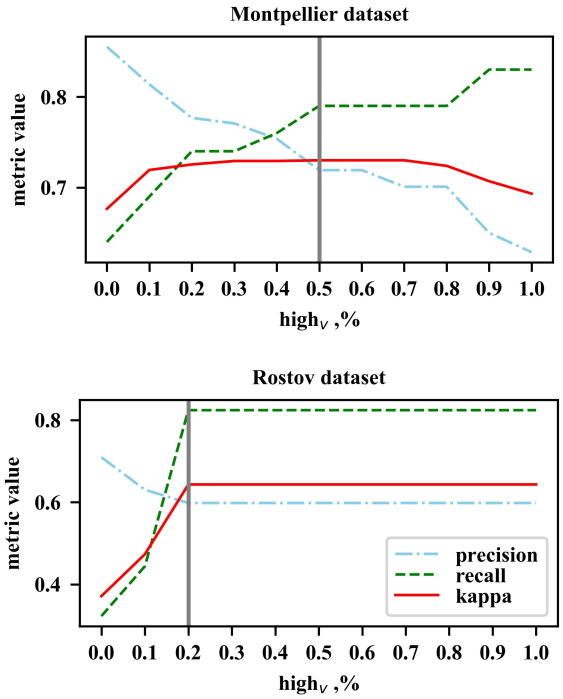

Figure 4.2: The influence of $high_v$ on the recall, precision and kappa metrics values for the Montpellier and Rostov datasets. The vertical line corresponds to the best $high_v$ for our ground truth data.

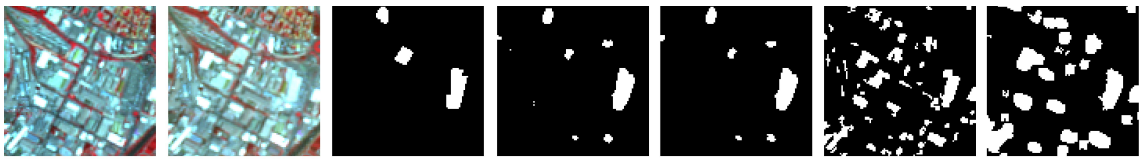

Figure 4.3: Classification results. Image extract 100×100 pixels. Example of luminosity sensitivity. (a)- image taken on May 2004, (b) - image taken on April 2005, (c)- ground truth, (d)- fully-convolutional AE, (e)- convolutional AE, (f)- RBM, (g)- improved RBM.

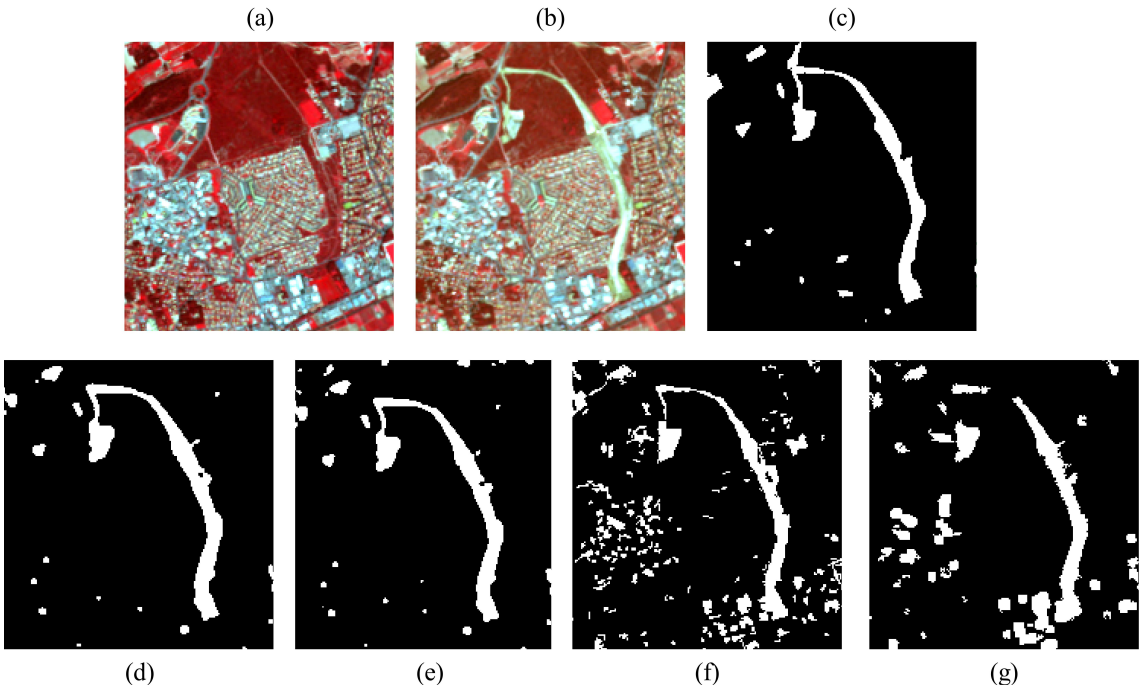

Figure 4.4: Classification results. Image extract 180×190 pixels. (a)- image taken on May 2004, (b)- image taken on April 2005, (c)- ground truth, (d)- fully-convolutional AE, (e)- convolutional AE, (f)- RBM, (g)- improved RBM.

between the two dates. We observe that both convolutional and fully-convolutional AEs have low ratio of false positive changes while the RBM sensitivity in urban area claimed by its authors is confirmed. At the same time, the improved RBM method has less false positives changes than the initial one. This can also be seen in Figures 4.4 and 4.6.

Figure 4.4 shows the construction of a new road. The road limits were correctly identified by all the models, except for the improved RBM model that did not detect the narrow part of the road.

Figure 4.5 displays changes in an agricultural area between May 2004 and April 2005. The overall seasonal change tendency is the following: the vegetation is more dense in May, the empty fields and fields with young crops have different minor changes between the two images. All the models except improved RBM showed relatively high ratio of false positive changes in vegetation. Nevertheless, the improved RBM missed more changes than other algorithms. The high ratio of false positives changes detected by first three architectures can be explained by the fact that vegetation density might be irregular and it is considered by the algorithm as changes. We observe that convolutional AE have slightly better results than other models.

Figure 4.6 represents changes in an agricultural area between February 2006

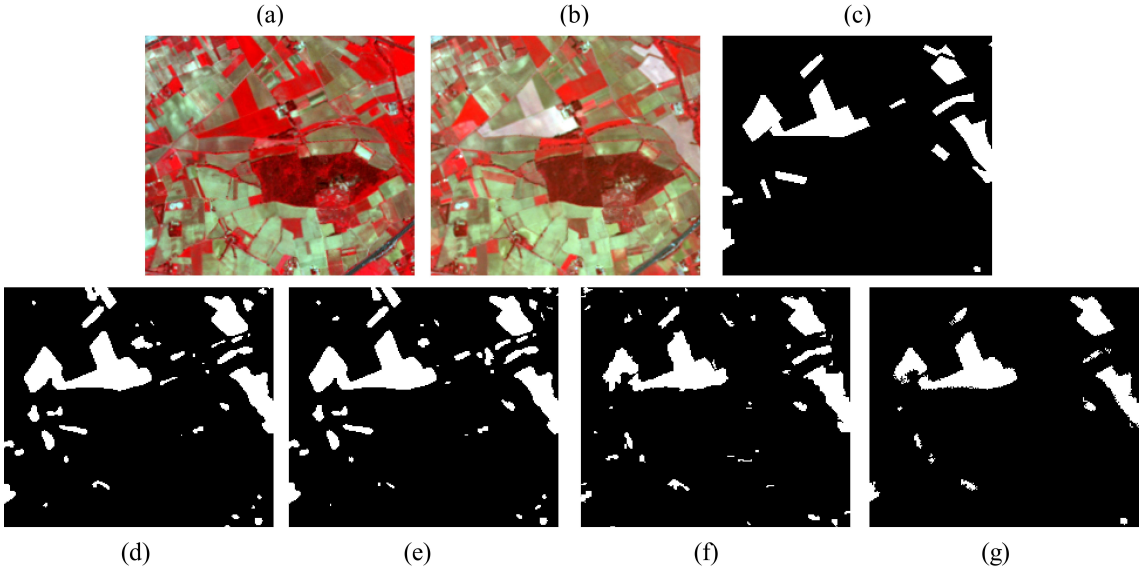

Figure 4.5: Classification results. Image extract 230×200 pixels. (a)- image taken on May 2004, (b)- image taken on April 2005, (c)- ground truth, (d)- fully-convolutional AE, (e)- convolutional AE, (f)- RBM, (g)- improved RBM.

and August 2008 as well as some new constructions. The overall seasonal change tendency is the following: the fields that are empty in February have vegetation in August, and vise versa. Forest's vegetation state (bottom right corner) has some minor changes. We can see again that the convolutional AE has slightly better results than the fully-convolutional AE. However, the ratio of false positive changes is elevated. Moreover, in most cases, only a part of a field is incorrectly labeled as change. As in the previous example, it can be explained by irregular vegetation density, and further morphological analysis might be needed to obtain better results. At the same time, the initial RBM model showed better performance for the detection of a linear object that corresponds to constructions at the roadside at the lower left part of the image, though the level of false positive changes is high both in urban and agricultural areas.

Figure 4.7 shows the limitations of the proposed approach for the detection of the construction of a tramway line. Our method have poor quality of change detection for linear objects that can be explained by the patch-wise learning. As a patch reconstruction error determines the change class of its central pixel, changes in 1-2 pixel width linear objects can not be properly detected.

The different approaches performances are presented in Table 4.4. All the algorithms were tested on 6 cores Intel(R) Core(TM) i7-6850K CPU 3.60GHz with 32 Gb of RAM computer with a NVIDIA Titan X GPU with 12 GB of RAM and

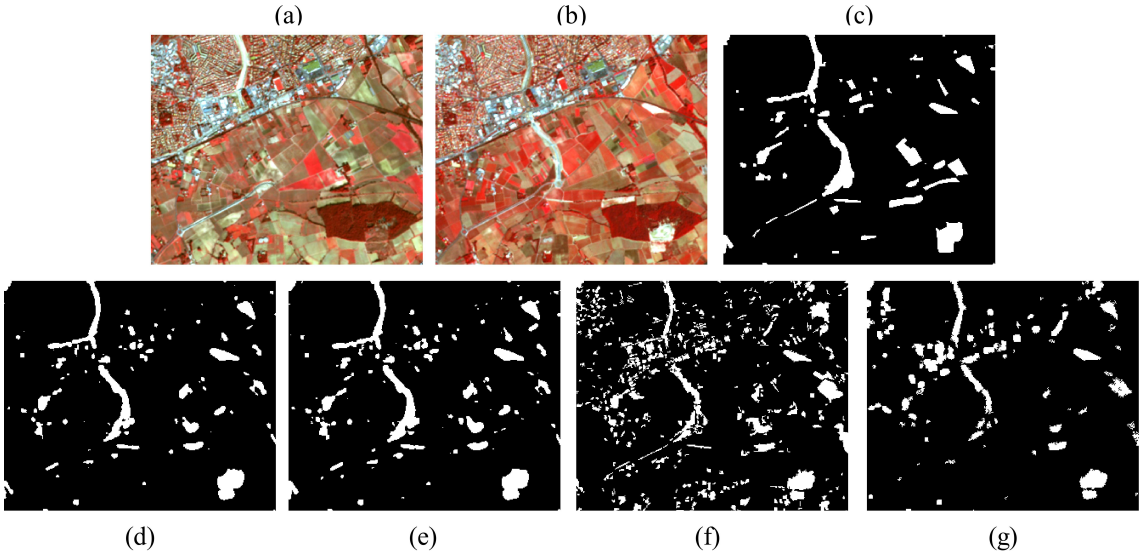

Figure 4.6: Classification results. Image extract 320×270 pixels. a- image taken on February 2006, b - image taken on August 2008, c- ground truth, d- fully-convolutional AE, e- convolutional AE, f- RBM, g- improved RBM.

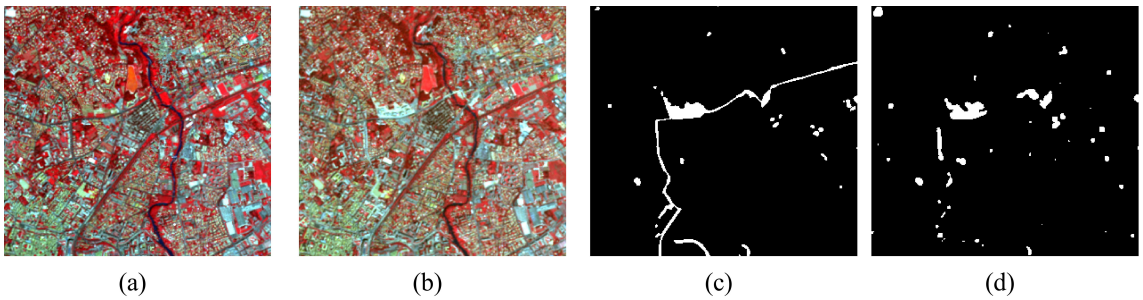

Figure 4.7: Classification results. Algorithm limitations. Image extract 300×280 pixels. a- image taken on May 2004, b- image taken on April 2005, c- ground truth, d- convolutional AE.

developed in *Python* 3 programming language using PyTorch 1.3 library on Ubuntu 16.4. Based on the presented results and on the performance estimators, we can conclude that joint convolutional AEs slightly outperformed fully-convolutional ones, though the training time stays higher as the model is more complicated. The performance of joint convolutional AEs can be explained by the higher complexity of the convolutional model. At the same time, both models of our approach showed better performances for change detection than the RBM-based models. However, it can be noted that the initial RBM method still has a high recall and the best training time despite a high level of false positive changes in urban areas compared to our approaches. We can equally conclude that methods with pixel-wise extracted patches have higher performance than initial RBM method where patches are not overlapped. Nevertheless, the improved RBM method detected less changes than the initial RBM method, though the number of false positives changes is much lower and overall classification performance characterized by kappa is higher.

Table 4.4: Performance of the change detection algorithms for the Montpellier dataset.

	Methods	**Classification performance**			
		Precision	***Recall***	***Kappa***	***Time*[a]*, min***
2004 2005	RBM AE	0.48	0.64	0.52	**8+2**
	Impr. RBM AE	0.52	0.63	0.54	20+10
	Conv. AE	**0.68**	**0.79**	**0.71**	17+12
	Fully-Conv. AE	0.67	0.78	0.70	15+8
2006 2008	RBM AE	0.40	0.61	0.43	**8+2**
	Impr. RBM AE	0.50	0.54	0.48	20+10
	Conv. AE	0.76	**0.79**	**0.75**	17+12
	Fully-Conv. AE	**0.80**	0.71	0.73	15+8

[a]Pre-training+fine-tuning.

The Rostov Dataset

Due to the shift presented in Rostov dataset, we have increased the patch size to $p = 7$ in order to minimize the shift influence on patch translation. We have equally chosen fully-convolutional AE model for feature translation as it gives better results than convolutional AE for bigger patch sizes.

The qualitative results of change detection for the Rostov dataset are given in Table 4.5.

Table 4.5: Performance of change detection algorithm based on the fully-convolutional joint AEs for the Rostov dataset.

Methods	Classification performance			
	Precision	***Recall***	***Kappa***	***Time*[a]*, min***
20150830-20150919	0.60	0.87	0.67	20+10
20170918-20180111	0.59	0.63	0.53	

[a]Pre-training+fine-tuning.

4.6 Discussion

In this chapter, we have presented an unsupervised approach for bi-temporal change detection in SITS. Change detection is performed for each pair of consecutive images of the series and is based on feature translation between two images realized with joint AEs. Contrary to excising approaches, our algorithm detects only non-trivial changes and ignores the seasonal ones.

The main experiments were performed on the Montpellier dataset (comparison to the state of the art methods, analysis of the values of different parameters, etc), then the most appropriate model configuration was chosen for change detection in the Rostov dataset.

Our experiments on the Montpelleir dataset have shown that our NN AE models perform better on a large area with various land cover occupation than the state-of-the-art RBM approaches. Among our proposed architectures, the joint convolutional AEs model showed slightly better performances for the patch size of 5×5 in spite of a longer training time.

However, as we notice pixel shift in the Rostov dataset that could not be corrected automatically. It was decided to choose the patch size of 7×7 for the Rostov dataset and fully-convolutional AEs model as it is better suited for bigger patches.

The changes detected for both datasets can be interpreted as contextual anomalies (in the context of the seasonal trends of two images). In the next chapter, these anomalies will be analyzed in the multi-temporal context to form multi-temporal change graphs. Therefore, some of the detected bi-temporal changes (including the FP ones) will be eliminated.

Chapter 5

Multi-temporal Change Detection

Contents

5.1 Introduction

Many remote sensing applications aiming to detect land cover changes in satellite image time series have been developed in the last years. Among them, there are numerous ecological applications such as the analysis and preservation of the stability

of ecosystems [126], the detection and the analysis of such phenomena as deforestation and droughts [127–130], real-time monitoring of natural disasters [131, 132], study of the evolution of urbanization [133], crop changes follow up [134, 135], etc. While for some applications we can create a training database with the objects presented in SITS [136, 137], for others it may be a challenging task as the nature of the researched spatio-temporal phenomena is often not known, unique or, on the contrary, has a lot of variations. For this reason, unsupervised and semi-supervised change detection and time series analysis have been a hot research topic for the past years. However, not so many successful studies are available for the given subject.

In this chapter, we propose an end-to-end unsupervised approach for the detection and classification of non-trivial changes in HR open source SITS. In this thesis, we imply that non-trivial changes are the changes free of seasonal trend prevailing in SITS (trivial changes).

The proposed framework firstly exploits neural network autoencoder (AE) to create bi-temporal change maps that contain only non-trivial changes (permanent changes and vegetation that do not follow the overall tendency). Second, segmentation of the extracted change areas is performed in order to detect different spatial entities. Then we use object-based techniques to model temporal behavior of the detected change phenomena in the form of evolution graphs. Finally, we obtain a summarized representation of the extracted evolution graphs - synopsis - and apply Gated Recurrent Units (GRU) [99] AE combined with hierarchical agglomerative clustering [49] to regroup them in different types of spatio-temporal change phenomena. Our algorithm framework does not demand any training data and gives us promising results on two real-life datasets of cities of Montpellier, France and Rostov-on-Don, Russia.

The rest of the chapter is organized as follow: Section 5.2 presents the existing algorithms for the analysis of different temporal phenomena in SITS. Section 5.3 describes the proposed framework for multi-temporal change modeling and clustering. Section 5.4 presents the time series we have used. In Section 5.5, we give the settings of our framework and show the obtained results. Finally, the last section concludes on the presented approach.

5.2 Related Works

Globally, we can divide the existing methods of SITS analysis in 1/ semi-supervised change detection which combines 1a/ anomaly detection [128, 129, 138] and 1b/

change detection in land-cover types in dense low/medium resolution time series [139, 140], and 2/ unsupervised clustering of the whole SITS [141, 142].

Most of semi-supervised SITS change detection algorithms aim to detect a particular type of change (mostly in the forest areas) based on previously available images. For example, in [138], the authors propose an algorithm for online prediction of forest fires in MODIS time series achieved by comparing real data with non-anomalous behavior modeled by Long Short-Term Memory (LSTM) network. In another study [128], the authors use Breaks For Additive Seasonal and Trend framework (BFAST) combined with a seasonal-trend model techniques to detect tropical forest cover loss exploiting multiple data sources (MODIS and Landsat-7). The principle of BFAST is based on iterative decomposition of time series into three components - trend, seasonal and remainder.

Similar approaches are used for change detection in land-cover types. In [139], the previously mentioned BFAST framework is deployed to detect land-cover changes and seasonal changes in MODIS NDVI series. To overcome this approach, in [140], the authors present a sub-annual change detection algorithm based on pixel behavior analysis. In this approach, firstly, temporal evolution of each pixel is presented as a signal with wavelet decomposition and then, each two points corresponding to the same acquisition day of two consecutive years are compared to determine the time of land-cover change.

To correctly model "normal" behavior and detect seasonal trends, the SITS data should be long enough and have regular and frequent temporal resolution for the study area. At the moment, this type of freely-available time series can be provided only by long existing missions such as MODIS (250 m-1 km resolution) and Landsat (30 m resolution) that are mostly used for vegetation analysis and, hence, have a low spatial resolution in favor of a high frequency of image acquisition. As the proposed techniques are adapted for low resolution images, they are not suitable to perform SITS analysis at the local level or to distinguish different land-cover sub-classes. Moreover, the aforementioned methods are adapted for univariate data (usually, NDVI index) that sometimes can not properly characterize all the data present in SITS.

With such high resolution (HR) freely-available time series as SPOT-5 and Sentinel-2, it has become possible to obtain more detailed information about the Earth surface. Unfortunately, freely-available HR SITS, especially the old ones, often contain some unexploitable images (flawed images, images with high ratio of cloud coverage, etc) or do not have sufficient temporal resolution to detect seasonal trends.

For these reasons, unsupervised HR SITS analysis demands different approaches, including ones that do not depend on temporal resolution (for example, general clustering of the whole SITS). In [143], the authors introduce a graph-based approach to detect spatio-temporal dynamics in SITS. In this method, each detected spatio-temporal phenomena is presented in a form of an evolution graph composed of spatio-temporal entities belonging to the same geographical location in multiple timestamps. Following this method, [142] proposes an adapted clustering algorithm to regroup the extracted multi-annual evolution graphs in different land-cover types. This algorithm exploits spectral and hierarchical clustering algorithms with dynamic time warping (DTW) distance measure [144] applied to the summarized representation of graph structure. It shows promising results both in intra- and inter-cite studies. In [141], the authors propose a bag of words (BoW) approach combined with Latent Dirichlet Allocation (LDA) [145] to detect different abstract clusters of land-cover. Though this algorithm is developed for synthetic aperture radar (SAR) SITS, it can be adapted for optical images after some data preprocessing. This method firstly exploits K-means clustering to determine the type of temporal evolution of each pixel with excessively large number of clusters ($100 - 150$). Second, every pixel is associated to a bag of words [146] representing its neighborhood patch. Each pixel of a patch contains an obtained discrete cluster value. Finally, all BoWs are regrouped in abstract classes with LDA method. Despite the fact that the proposed approaches can be used on HR images with irregular temporal resolution, they are not conceived to detect anomalies or land-cover changes. For example, the first method is used for clustering of the prevailing land-cover types without considering eventual land-cover changes or some anomalies. Contrary to this approach, the second method cluster the whole SITS into a large number of abstract classes without associating them to any specific spatio-temporal phenomena, so the land-cover change is not differentiated from seasonal phenological variation.

Some applications demand another kind of SITS analysis performed at a finer level such as, for example, detection and classification of different types of changes applied to the urban development analysis or some local ecological disturbances. For this reason, we propose an algorithm able to model multi-temporal changes as evolution graphs and cluster them. The proposed change detection approach can be applied to short time series and does not depend on the temporal resolution, as it does not demand the prior knowledge of the seasonal trend that can be computed only if the series is long enough and regularly distributed. For our knowledge, no such approaches have been proposed yet. Moreover, there is no other unsupervised

framework that does both change detection and the clustering of the subsequent change graphs.

Our proposed method can be used for projects studying the urban evolution of cities: changes in density of residential areas, major urban structure constructions (such as transportation systems or stadiums), trends in the evolution of green areas and parks, etc. Our algorithm is also concerned with the evolution of the landcover in crop cultures. Other applications may include ecological problematic such as the study of SITS to assess phenomena such as deforestation or the impact of droughts.

5.3 Methodology

In this section, we present the developed framework for the detection and analysis of non-trivial changes in SITS. We start by giving the overall description of the framework and then we detail each step in the corresponding subsections.

5.3.1 Proposed Framework

The proposed framework has been developed as a part of a SITS analysis algorithm. This algorithm aims to identify spatio-temporal entities presented in SITS and associate them to three different types of temporal behaviors. These temporal behaviors correspond to no change areas, seasonal changes and non-trivial changes. No change areas are mostly presented by spatio-temporal entities that have the same spectral signature over the whole SITS, such as city center, residential areas, deep water, sands, etc. Trivial (seasonal) changes correspond to cyclic changes in vegetation prevailing in the study area. Finally, non-trivial change areas are mostly represented by permanent changes such as new constructions, changes caused by some natural disasters, crop rotations and the vegetation that do not follow the overall seasonal tendency of the study area. As non-trivial changes are less numerous and sometimes unique, they are considered by most clustering algorithms as outliers and, hence, demand a special approach for their identification and analysis that is presented in this thesis. Moreover, as often we do not dispose of HR SITS with regular temporal resolution, the presented approach does not depend on temporal resolution of SITS.

The proposed approach is composed of several steps. Let R_S be a time series of S co-registered images Im_1, Im_2, ..., Im_S acquired at timestamps T_1, T_2, ..., T_S. The algorithm steps are the following (Figure 5.1):

– We start by applying a bi-temporal non-trivial change detection algorithm

(Chapter 4) to every couple of consecutive images Im_n-Im_{n+1} ($n \in [1, S-1]$) in order to get $S-1$ binary change maps $CM_{1,2}$, $CM_{2,3}$, ..., $CM_{S-1,S}$ for the whole dataset. Then the detected bi-temporal changes are analyzed in the multi-temporal context.

– We extract spatio-temporal change areas by applying the change masks to corresponding images of SITS. Then we perform image segmentation within these change areas to obtain changed objects.

– Afterwards, the change objects located in the same geographic area are grouped in temporal evolution graphs [142, 143].

– Finally, we cluster the obtained graphs using the features extracted from the change areas. We use summarized representation of graph structure - synopsis - as input sequences of clustering approach.

5.3.2 Change Detection

Bi-temporal Change Detection

For the proposed framework, we have developed a bi-temporal non-trivial change detection algorithm (Chapter 4) that is applied to the whole SITS. The obtained bi-temporal change detection results are then analyzed together in the context of the whole SITS.

Contrary to multi-temporal approaches that demand regular temporal resolution of SITS to get correct results, our two steps method does not assume the temporal regularity. Moreover, when using the bi-temporal approach, we can easily detect the beginning and the end of a change process in a SITS of a fixed size without applying any supplementary methods.

Multi-temporal Change Interpretation

Note that the detected bi-temporal non-trivial changes can also be interpreted as contextual anomalies [1] as they depend on the overall change tendency presented in the couple of images. Their interpretation might be changed when passing from bi- to multi-temporal analysis as the context will be changed. To introduce multi-temporal context when detecting changes that appear between timestamps T_n and T_{n+1}, we propose to check if the detected change polygons have equally appeared at certain other change maps, see Figure 5.2.

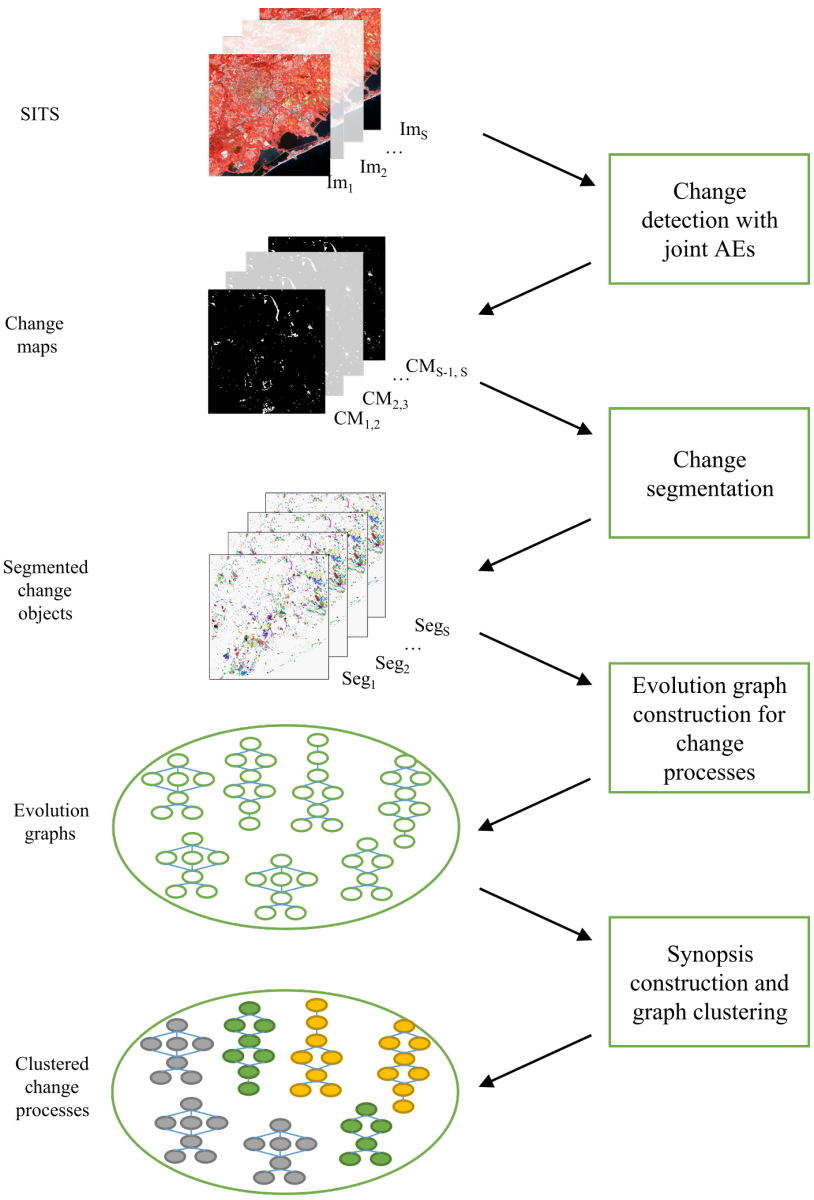

Figure 5.1: Proposed framework.

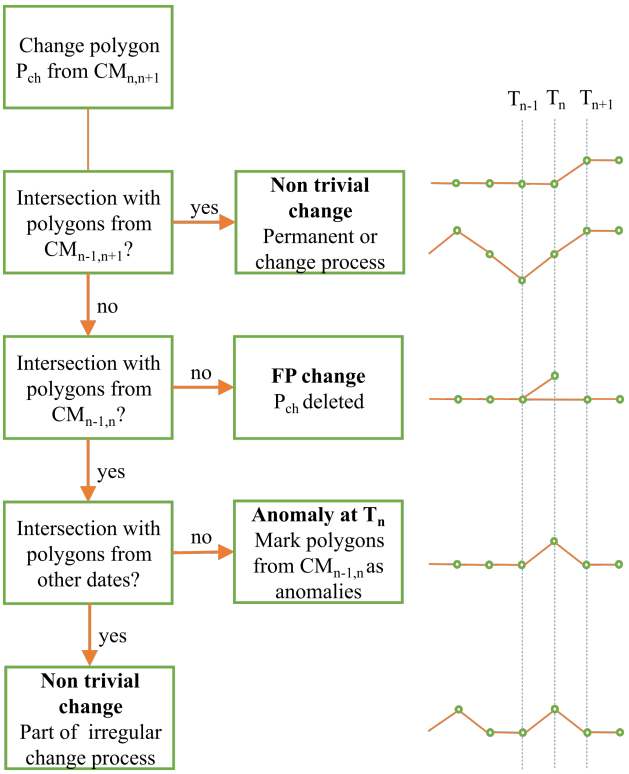

Figure 5.2: Correction of the detected bi-temporal contextual anomalies accordingly to the multi-temporal context.

Firstly, we apply the aforementioned bi-temporal change detection algorithm for every couple of Im_{n-1}-Im_{n+1} images ($n \in [2, S-1]$) to calculate $CM_{n-1,n+1}$. As an isolated change area may contain several change objects, we perform image segmentation to obtain bi-temporal change polygons (the segmentation algorithm is explained in the following subsection and applied directly to the concatenated couples of images Im_n-Im_{n+1} and Im_{n-1}-Im_{n+1}). Then, we choose $CM_{n,n+1}$ as reference and check if each change polygon P_{ch} from $CM_{n,n+1}$ is equally present in $CM_{n-1,n+1}$. If P_{ch} has a spatial intersection with any polygon(s) of $CM_{n-1,n+1}$, we mark P_{ch} as change in the SITS context (a permanent bi-temporal change or a part of a change process). If P_{ch} does not have spatial intersection with any polygon(s) of $CM_{n-1,n+1}$, it may belong to different types of temporal behavior:

- If P_{ch} does not have any spatial intersection with any polygon(s) from $CM_{n-1,n}$, it is marked as false positive (FP) change alarm caused by some image defaults and accuracy issues of unsupervised methods.

- If P_{ch} has a spatial intersection with any polygon(s) from $CM_{n-1,n}$ and with polygon(s) in at least one other change map ($CM_{1,2}$, $CM_{2,3}$, $CM_{3,4}$, etc), it is

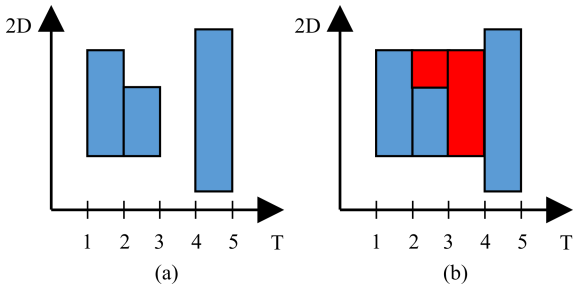

Figure 5.3: Transformation of a discontinuous change process into a continuous one. Horizontal and vertical axes represent timestamps and object pixels respectively. (a)- discontinuous change process, (b)- corrected blue polygons correspond to detected change objects, red polygons are added to transform a discontinuous change process into a continuous one.

marked as a part of an irregular change process.

– Finally, if P_{ch} has an intersection only with polygon(s) from $CM_{n-1,n}$ and does not have any intersection with polygons from other change maps, it is marked as a one time anomaly that happened at timestamp T_n. In this case, all change polygons from $CM_{n-1,n}$ that have intersection with P_{ch} are also marked as one time anomalies.

Note that here we use a threshold t_{int} that defines the minimum percentage of spatial intersection of P_{ch} with other change polygon(s), otherwise, it is considered that there is no intersection.

In this thesis, we suppose that every change process belonging to the same geographical location is continuous. For example, if a pixel i, j has been classified as change in $CM_{1,2}$, $CM_{2,3}$ and $CM_{4,5}$, it should be also marked as change in $CM_{3,4}$ (Figure 5.3).

We correct the whole SITS in order to transform change processes into the continuous ones. Finally, for every image Im_n, we apply the union of change maps $CM_{n-1,n}$ and $CM_{n,n+1}$ to extract the change areas. Obviously, we apply only one change map for the first and last images of the SITS.

5.3.3 Image Segmentation

In the previous subsection, we have proposed an approach that introduces the multi-temporal context to the analysis of bi-temporal changes. As it was mentioned, an isolated change area may potentially contain several types of changes. The identification of change segments is needed at two different steps of the proposed framework. First, we detect change objects for the aforementioned algorithm. The

segmentation is performed for every couple of concatenated images Im_n-Im_{n+1} within the associated change mask. For the second step, we perform the segmentation of change areas of every image Im_n, so the detected segments will later be used for the construction of evolution graphs.

For both segmentation steps of the framework, we leverage a graph-based tree-merging segmentation algorithm [147] due to its ability to produce relatively large segments without merging different classes together. Large segments facilitate further construction and interpretation of evolution graphs as the shapes of some change segments may have important variations from one image to another when they are over-segmented.

The principle of the segmentation algorithm is the following: an image is represented as a graph where the pixels are nodes and resemblance values between two pixels are edges. Usually, each pixel has 8-connected neighborhood. At the beginning of the algorithm, each pixel belongs to a separate segment, then, at each step of the segmentation algorithm, the adjacent segments are merged if the difference between them is not much less than the internal variability of these segments. To measure the resemblance value between two pixels, we use Mahalanobis distance:

$$d\left(\vec{x},\vec{y}\right)=\sqrt{\left(\vec{x}-\vec{y}\right)^{\top}Cov^{-1}\left(\vec{x}-\vec{y}\right)}, \tag{5.1}$$

where $\vec{x}$, $\vec{y}$ are two pixels values and Cov is the covariance matrix of the image values.

The proposed algorithm requires three parameters which are σ - the standard deviation for Gaussian kernel that is used for image smoothing during the pre-segmentation phase, k - the constant for the thresholding function used for segment merging, and min_size that defines the minimum component size for the post-processing stage.

5.3.4 Evolution Graphs

For our framework, we adapt the evolution graph construction algorithm presented in [142, 143] by passing from the analysis of the whole SITS to the analysis of change areas. This approach combines graph-based techniques and data-mining technology and is used to describe a spatio-temporal evolution of a detected phenomena. The initial approach is the following: given a SITS and its associated segmentation, we choose a set of objects that corresponds to the spatial entities we want to monitor. This set of objects is called Bounding Boxes (BBs). A BB can come from any timestamp and is connected to the objects covered by its footprint at other timestamps. A BB

and objects connected to it form an evolution graph. Each evolution graph can have only one BB and has to be continuous. Every object of a graph represents a node and overlapping values between two objects at two consecutive timestamps represent an edge. Objects at timestamp T_n can be connected only to objects from T_{n-1} or T_{n+1}, a timestamp that contains a BB can have only one object that corresponds to this BB.

Bounding Box Selection

For BB selection, we use the same strategy as in the initial articles [142, 143], except that we do not work with the whole dataset, but only with change areas. The methodology is the following:

1. We create an empty 2D grid G that covers all changed areas. If a segment is chosen as a BB, the grid is filled with its footprint.
2. We create an empty list of candidate BB $L_{cand_{BB}}$.
3. We sort all the segments presented in SITS change areas by their size in the descending order.
4. We iterate through each segment of the sorted list.
5. We calculate the novelty and weight of each element comparatively to already iterated segments. These values are calculated as follows:

$$\text{novelty} = \frac{Pix(O) - CA}{Pix(O)} \tag{5.2}$$

where $Pix(O)$ is the segment's size in pixels and CA is the size of a grid area that is covered by this segment and already filled by previous candidate BBs.

$$\text{weight} = \begin{cases} Pix(O) & \text{if novelty} = 1 \\ \text{novelty} & \text{if } \alpha \leq \text{novelty} < 1 \\ 0 & \text{if novelty} < \alpha \end{cases} \tag{5.3}$$

If $weight \geq \alpha$, we select this segments as a candidate BB and add it to $L_{cand_{BB}}$.

6. We iteratively update the weights in $L_{cand_{BB}}$ and delete previously detected candidate BBs if needed.

7. We iterate through the list until no more segments are left or until the grid is filled.

Construction of Evolution Graphs

In our method, we construct graphs in such a manner that every graph contains only coherent information. In other words, every BB is connected only to its best matching segments comparatively to neighbor BBs and each segment can belong only to one or no evolution graph.

In order to construct graphs that contain only objects belonging to the same phenomena, in [142, 143], the authors propose to use two parameters for the construction of evolution graphs that are independent of each other: at least τ_1 of the object should be inside of BB footprint, and the intersection with the object should represent at least τ_2 of BB footprint.

$$\tau_1 = \frac{Pix(O) \cap Pix(BB)}{Pix(BB)} \tag{5.4}$$

$$\tau_2 = \frac{Pix(O) \cap Pix(BB)}{Pix(O)} \tag{5.5}$$

The first parameter τ_1 is the most important and allows to select only the objects that are covered the most by BB footprint. The second parameter τ_2 is used to keep the objects filling only certain percentage of the footprint.

In our experiments, we are interested only in change areas of SITS, hence an isolated evolution processes may not cover the same area at different timestamps (e.g. a growing construction of a residential area).

However, it may cause that some objects will be covered by two or more BBs. To deal with this issue, we propose to associate each object only to its best matching BB, i.e the one with the highest τ_1 value. In addition, for $\alpha < 0.5$, an object O_i may be a bounding box BB_i of a corresponding graph and at the same time attached to an evolution graph of a neighbor bounding box BB_j. In this case, we compare BB_i's (O_i) *weight* value with O_i's τ_1 value in the evolution graph of a neighbor bounding box BB_j. If $\tau_1 > weight$, O_i is attached to BB_j evolution graph and the evolution graph of BB_i is destroyed and its objects are attached to other graphs, otherwise BB_i and its evolution graph are kept and O_i is deleted from BB_j's evolution graph.

Due to pixel shift and to some false positive changes, we may observe many parasite objects presented in evolution graphs that usually correspond to crop fields. Parasite objects are the small objects attached to the beginning or to the end of

the graph, these objects are the only objects presented at given timestamp and are much smaller than the footprint of the previous/next timestamp. If an evolution graph contains a parasite object, it can influence its further interpretation. Given the size of these objects, we can consider that they correspond to false positive changes and need to be deleted. To minimize the number of parasite objects, we introduce τ_3 parameter that represents the minimum ratio between coverage of two consecutive timestamps.

$$\tau_3 = \frac{\sum_1^q Pix(O_i^{n+1})}{\sum_1^r Pix(O_j^n)}, \tag{5.6}$$

where q and r are the number of objects at timestamps T_{n+1} and T_n respectively, $Pix(O_i^{n+1})$ is i-th object at timestamp T_{n+1} and $Pix(O_j^n)$ is j-th object at timestamp T_n.

5.3.5 Graph Synopsis and Feature Extraction

To cluster the extracted evolution change graphs, we calculate each graph synopsis as in [142]. A synopsis summarize the information about each graph and allow to compare them with each other. A synopsis Q is defined as a sequence of the same length as the corresponding evolution graph. Each timestamp T_n of the sequence Q contains the aggregated values of graphs objects at this timestamp. The influence of each object at the aggregated value at the timestamp T_n is proportional to its size and calculated as follows:

$$Q_n = \frac{\sum_1^r Pix(O_j^n) \cdot v_j}{\sum_1^r Pix(O_j^n)} \tag{5.7}$$

where Q_n is the synopsis value at timestamp T_n, $Pix(O_j^n)$ is the size of a j-th object at timestamp T_n ($j \in [1, r]$, where r is the total number of the objects presented in the evolution graph E at the timestamp T_n) and v_j is the corresponding mean of the object value.

In our algorithm, each object is characterized by the feature values extracted with deep convolutional denoising autoencoder. We add a Gaussian noise to the initial patches of the dataset during the training to extract more robust features. The noise is added only to the input of the first convolutional layer. To calculate the MSE for model optimization, we use the initial patches without noise as targets. For every pixel i, j, n its features are extracted in a patch-wise manner.

The choice of feature extraction and its efficiency is described in the experiment section, where we compare the clustering performance for our proposed framework using a feature extraction step and also using raw pixel values for graph descriptors.

All the images are normalized using dataset mean and standard deviation before texture extraction.

In our proposed framework, the feature extraction steps are the following:

1. We extract patches of size p for every pixel of every image of SITS to train a convolutional AE model.

2. We divide the extracted patch dataset in training and validation parts (67% and 33%). Validation part is used for early stopping algorithm [148] that prevents our model from over-fitting and works in an unsupervised manner. The more the model is generalized- the more robust features we get.

3. We train the AE model in such manner that every patch from the training dataset is firstly encoded in a feature vector and then is decoded back to the initial patch. We use mean square error (3.2) of the patch reconstruction to optimize the model at each iteration.

4. The early stopping algorithm is applied at every epoch and check the loss value when fitting validation dataset to the obtained model. If the validation loss does not improve during chosen number of epoch, the model is considered stable and the training is stopped.

5. We use the encoding part of the AE to encode every patch of SITS change areas in a feature vector.

The feature extraction model is presented in Table 5.1, where f is the size of the encoded feature vector and C is the number of image spectral bands (channels). Note that for all convolutional layers the kernel size is 3. We equally apply batch normalization to all convolutional layers.

5.3.6 Clustering

In the presented framework, we propose to use GRU [99] AE combined with hierarchical agglomerative clustering [49] to regroup the obtained evolution graphs. GRU is a recurrent neural networks-based (RNN) type of NN that is able to process time series in order to extract some meaningful information from it (Figure 3.9). Unlike many other approaches for time series analysis, RNNs are able to deal with varying sequence lengths. More about different RNNs and about the choice of GRU can be found in Chapter 3.4.5.

Table 5.1: Feature extraction model.

	Feature extraction
encoder	Conv(C,32)*+ReLU
	Conv(32,32)+ReLU
	Conv(32,64)+ReLU
	Conv(64,64)+ReLU
	MaxPooling(kernel=3, stride=3)
	Conv(64,128)+ReLU
	Conv(128,128)+ReLU
	MaxPooling(kernel=3)
	Linear(128,64)**+ReLU
	Linear(64,32)+ReLU
	Linear(32,f)+ℓ_2-$norm$
decoder	Linear(f,32)+ReLU
	Linear(32,64)+ReLU
	Linear(64,128)+ReLU
	UnPooling(kernel=3)
	Conv(128,128)+ReLU
	Conv(128,64)+ReLU
	UnPooling(kernel=3, stride=3)
	Conv(64,64)+ReLU
	Conv(64,32)+ReLU
	Conv(32,32)+ReLU
	Conv(32,C)+ReLU

*In Conv(a,b) a and b stand for the number of the input and output channels respectively.
**In Linear(a,b) a and b stand for the sizes of the input and output vectors respectively.

The main idea of all RNNs is to compute hidden states $h = \{h_1, h_2, ..., h_n, ..., h_S\}$ for each timestamp of a sequence $X = \{x_1, x_2, ..., x_n, ..., x_S\}$ of length S. The hidden state at each timestamp presents an accumulated value of all previous timestamps of the sequence. The final hidden state h_S characterizes the whole sequence and is used as its descriptor.

The principle of a GRU AE is the same as of the convolutional one mentioned before. During the encoding pass, GRU AE extracts the accumulated hidden state of the sequence h_S at the last timestamp. The last hidden state is then self-concatenated S times [149] and passed to the decoding part that aims to reconstruct the inversed initial sequence $X_{inv} = \{x_S, ..., x_n, ..., x_2, x_1\}$. As it is usually recommended to set hidden state size large (> 100), we add fully-connected layers before the GRU AE bottleneck to compress the size of hidden state to ameliorate the further clustering results. The overall GRU AE schema is presented on Figure 5.4. Finally, we apply hierarchical clustering to the bottleneck of GRU AE to obtain the associated change clusters.

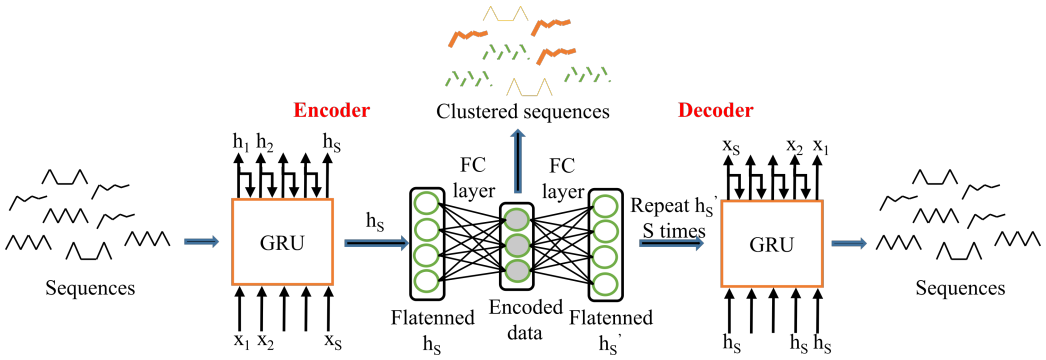

Figure 5.4: GRU AE clustering model.

As the input sequences have varying length, some data preparation is necessary, so the GRU is able to correctly process it. Data preparation is performed for every training batch individually, after the input GRU dataset has been created. For every batch B_i, we perform the following steps (see Figure 5.5):

1. We define the maximum sequence length d of B_i.
2. We zero-pad the end of all the sequences of B_i, so they have the same length d.
3. The padded sequences are passed to the encoder, for each sequence its final hidden state h_S is obtained.
4. As indicated before, we use the cloned h_S as the input of GRU layer in the decoding part, where h_S is repeated S times.

5. We apply the inverted padding mask to the cloned h_S sequence that is fed to GRU layers of the decoder.

6. The output of the decoder should resemble to the inverted padded input sequence.

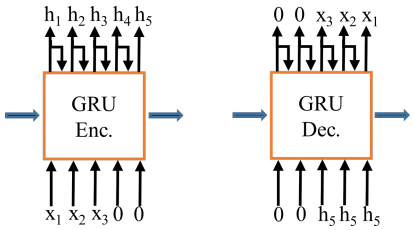

Figure 5.5: Padding of data sequences. In this example, the initial sequence $\{x_1, x_2, x_3\}$ has the length of 3 timestamps and the maximum sequence length per batch is $d = 5$. For the simplicity of representation, we do not consider the number of features of each sequence.

While the padding of the encoder input sequence allows us to process batches with varying length sequences, the padding of the decoder input improves the model quality, especially, it lowers the influence of sequence lengths on the extracted encoded features.

We do not divide the sequence data into training and validation datasets as the nature of some change sequences may be unique. For these reason, we control the training loss changes between two consecutive epochs to prevent the model over-fitting. We have used MSE as the loss function.

The model configuration is presented in Table 5.2, where f is number of features of the input sequences, *hidden_size* is the length of the hidden state vector, d is the maximum sequence length per batch, f*_hidden* is the size of the encoded hidden state vector.

5.4 Data

We test the proposed approach on two real-life publicly available time series - the SPOT-5 series of the Montpellier area, France and the Sentinel-2 series of the city of Rostov-on-Don, Russia. While the first SITS contains 12 images that are irregularly sampled over 6 years (see Appendix A.1, later called the Montpellier dataset in this chapter), the second one contains 17 images taken over 2 years with more regular temporal resolution (see Appendix A.3, later called the Rostov dataset in this chapter).

Table 5.2: GRU model.

	Sequence feature extraction
enc.	GRU(f,hidden_size, dropout=0.4)* (2 layers) Linear(hidden_size,f_hidden)**+ℓ_2-$norm$
dec.	Linear(f_hidden,hidden_size)+ReLU Repeat hidden state d times Apply inversed padding mask GRU(hidden_size,f, dropout=0.4) (2 layers)

*In GRU(a,b dropout=) a and b stand for the number of features of the input sequence and the size of the hidden state vector.
**In Linear(a,b) a and b stand for the sizes of the input and output vectors respectively.

For both datasets, we have annotated the most frequented of detected change processes to evaluate the clustering results. Note that we do not dispose of the labels for different crops, so the vegetation is annotated accordingly to its variation in the detected change sequence (for example, vegetation → weak vegetation → empty field → vegetation) and that we do not know the exact number of change process classes. For the Montpellier dataset, the annotated change processes that were regrouped in 10 classes are:

1. 5 different vegetation variations (3644, 4429, 4307, 2542 and 2206 pixels per annotated class),
2. changes in deep water (8164 pixels),
3. construction in a dense area (4122 pixels),
4. construction of a new area (13648 pixels),
5. beginning of the construction process (bare soil/construction field) (2196 pixels),
6. changes in buildings luminosity (false positive changes) (501 pixels).

For the Rostov dataset we got 11 classes:

1. 5 different vegetation variations (2864, 16930, 14805, 7521 and 6183 pixels),
2. construction of a new building (969 pixels),
3. construction of a new area (dense construction) (13669 pixels),
4. construction of a new residential area (sparse small private buildings) (241 pixels),
5. changes in bare soil (2201 pixels),
6. seasonal changes in trees/bushes growing areas (8970 pixels),
7. shadow from tall buildings (FP changes) (3708 pixels).

5.5 Experiments

In this section, we present parameters settings for the proposed framework and the corresponding results. We equally give some quantitative and qualitative analysis of the obtained results to show the advantages and eventual limits of the proposed approach. All the algorithms were tested on 6 cores Intel(R) Core(TM) i7-6850K CPU 3.60GHz with 32 Gb of RAM computer with a NVIDIA Titan X GPU with 12 GB of RAM and developed in *Python* 3 programming language using PyTorch 1.3 library on Ubuntu 16.4. *tslearn* library [150] was used to calculate DTW distance matrices for concurrent approaches.

5.5.1 Experimental settings

The chosen datasets have some differences in temporal resolution and images quality. For this reason, the proposed framework have some dissimilarities based on the particularities of each dataset.

Due to the high temporal gap between the images of the Montpellier dataset, we tend to extract the maximum of the information from it, so all extracted change sequences are kept. On the other hand, for the Rostov dataset, we keep only the change sequences that are at least 3 timestamps long. As the average temporal resolution between two images is less than two months, we consider that 2 timestamps change sequences most probably correspond to some false change alarms that could not be corrected when introducing the multi-temporal context.

For the Rostov dataset we manually apply a water mask and a mask for a part of the city with dense constructions. While the first mask is applied to eliminate FP changes caused by boats, the second one is indispensable to lower the number of FP changes caused by the shadows from tall buildings and changing luminosity that can not be corrected.

The water mask was obtained with Sen2Cor[1] classification module. Firstly, the water surfaces were detected for every image of the dataset individually. Secondly, the pixels that were marked as water for more than 50% of the images of the dataset were added to the water mask. Dense urban area mask was created manually and has almost a rectangular shape that cover a specific city area.

The last difference is related to the fact that the images of the second dataset are not perfectly aligned and some of them have one pixel size shift from the origin in

[1]Available on `https://step.esa.int/main/third-party-plugins-2/sen2cor/`

different directions. This shift could not be eliminated with automatic techniques and, for this reason, we have decided to use bigger patches comparatively to Montpellier dataset for bi-temporal change detection.

Bi-temporal change detection

All the information about bi-temporal change detection is provided in Chapter 4. We remind that for the Montpellier dataset, we have chosen the patch size $p = 5$ and the convolution AE model for feature translation as they were proved to give the best results for 10 m resolution images. Due to the shift presented in Rostov dataset, we have increased the patch size to $p = 7$ in order to minimize the shift influence on patch translation. We have equally chosen fully-convolutional AE model for feature translation as it gives better results than convolutional AE for bigger patch sizes.

Image segmentation

To segment the extracted change areas, we have used a tree-merging bottom-up segmentation algorithm [147] described earlier. The segmentation parameters were chosen empirically and allow us to produce large segments for both SITS without mixing different classes in one segment. For each dataset, we have manually chosen reference adjacent objects that have to be divided in separated segments. These objects need to have similar spectral properties, but still be easily distinguished by human eye. We set the same parameters for single image segmentation and for the segmentation of concatenated images, so the change objects extracted from concatenated images are relevant to single-image objects.

The segmentation parameters for both datasets are the same and equal $\sigma = 0.1$, $k = 7$, $min\_size = 10$.

Multi-temporal change detection

When the preliminary change maps with bi-temporal contextual anomalies are obtained, we move on to their analysis in multi-temporal context. t_{int} should have the same value as τ_1, because both parameters define the level of spatial relation between segments at different timestamps. In other words, t_{int} defines if changes detected between different timestamps belong to the same change process and τ_1 defines if a candidate object belongs to a reference BB. If we set t_{int} smaller than τ_1 and a candidate change polygon is kept after multi-temporal context analysis, it may not be attached to any BB during evolution graph construction. On the contrary,

if t_{int} is set much higher than τ_1, it will produce mostly spatially compact change processes, so small τ_1 value is unnecessary.

For both datasets, we have set t_{int}=0.4, as τ_1=0.4 with corresponding α gave the best results comparatively to other parameters combinations for both datasets (explained in the next subsection). Logically, the performance of the proposed approach fully depends on segmentation quality. The approach was qualitatively validated using manually selected reference objects.

Evolution graph construction

As it was mentioned before, in our thesis, we tend to create evolution graphs that contain only the objects that belong to the same change process. In the previous works, α value varies between 0.3 and 0.5. Smaller α values provide more overlapping graphs and, on the contrary, if α is high, we may obtain low graph coverage ratio, hence, a lot of change processes will be missed. To obtain the optimal BB coverage and graph overlapping, we choose α values 0.4 and 0.5 for the Montpellier and Rostov datasets respectively.

As every object can belong to only one evolution graph that ensures the maximum possible spatial overlay with corresponding BB, we propose to omit τ_2 value and to set τ_1 value excessively small. The smaller τ_1 is, the bigger graph footprint we might obtain. However, most evolution graphs are compact independently of τ_1 value. For both datasets, τ_1=0.4 provided the best graph coverage. τ_3 value has to be chosen after visual analysis of detected changes and parasite objects that correspond to false positive changes. Clearly, its value should not be too high as the footprint of change processes may have important variations at each timestamp. The parameters for evolution graph construction that gave us the best results are presented in Table 5.3.

Table 5.3: Parameters for evolution graphs construction.

Dataset	α	τ_1	τ_3
Montpellier	0.4	0.4	0.2
Rostov	0.5	0.4	0.3

Feature extraction autoencoder

To cluster the evolution graphs, it is necessary to extract robust descriptors of change objects. In our framework, we deploy a deep convolutional AE model for non-supervised feature extraction. We use the same model for both datasets (see Table 5.1) with patch size p=9 as it provides sufficient coverage of neighborhood pixels without high influence of objects border pixels. Contrary to the change detection algorithm, we keep all 4 spectral bands for the Montpelier dataset (Green, Red, NIR and SWIR) as SWIR band may ameliorate the clustering of some similar classes. We keep C=3 for the Rostov dataset as in change detection. We encode the patches of the Montpellier and Rostov datasets in feature vectors of size 5 and 4 respectively. Small size of feature vectors was chosen to obtain only the most representative features. Moreover, feature vectors size is equal to the number of spectral bands plus 1 ($f = C + 1$) to avoid potential linear correlation between a spectral band and a corresponding element of the feature vector.

Evolution graph clustering

After the texture extraction, we create synopses for all evolution graphs that are then passed to the GRU AE model combined with a chosen clustering method. As we do not know beforehand the exact number of change classes presented in a SITS, we choose hierarchical agglomerative clustering algorithm. This choice is justified in Chapter 2.6.

The model parameters were set as follows for both datasets: $hidden\_size$=150, f_hidden=20. Both parameters were achieved empirically, though some general recommendations were followed to set the second parameter. For example, the size of AE bottleneck should correspond to the number of researched clusters [94], although, this number is not known beforehand, we can distinguish around 10-15 the most frequented types of change processes for both datasets. To improve the clustering results, the extracted sequences are normalized by their mean and standard deviation.

After the model is stabilized, we apply the hierarchical agglomerative clustering to the encoded results. As parameters for clustering algorithm, we use Ward's linkage [49] and Euclidean distance between the encoded points. We test the results for different number of clusters in the range from 5 to 50.

Unfortunately, the right choice of the number of clusters is still an open question. No robust methods for the automatic selection of the number of clusters were find in literature and for some datasets it seems impossible to guess it. However, we can

visually estimate the number of prevailing change classes and set an optimal range of the number of clusters for the hierarchical clustering. Considering the specificity of the algorithm and that its model does not depend on the selected number of clusters, we do not need to know their exact number.

5.5.2 Results

In this subsection, we present quantitative and qualitative results obtained with the proposed framework. We start by presenting the intermediate results of the framework, then we present the final results obtained by the clustering algorithm and discuss the influence of the parameters of the intermediate algorithms on the final results.

We start by bi-temporal change detection that defines the final overall accuracy of our framework. The more false positive changes there are - the more parasite objects or even parasite graphs are detected. For this reason, during the next steps, we tend to reduce their quantity by using different techniques (for example, by introducing τ_3 value).

The performance of the algorithm for non-trivial bi-temporal change detection is presented in Chapter 4. We observe that both datasets have high recall values that indicate that most of the changes are correctly detected. At the same time, precision value is low for the Rostov dataset and depends mostly on the quality of the images itself (a lot of saturated objects and shadows are detected as changes).

Once the obtained change maps are analyzed in multi-temporal context, we move on to the construction of the evolution graphs. Table 5.4 presents the evolution graph statistics for the chosen parameters that ensure the best coverage. It provides total graph coverage, graph overlapping percentage, minimum graph compactness represented by the ratio of BB footprint to the total graph coverage footprint, dataset average graph compactness, percentage of graphs with compactness inferior to 50%

Table 5.4: Statistics about evolution graphs construction.

	Statistics, %					
Dataset	**Cove-rage**	**Over-lapping**	**Min. comp.**	**Avg. comp.**	**Comp. <50%**	**Comp. <75%**
Montpellier	96	14	30	94	0.4	7.5
Rostov	92	9	34	93	0.7	9.1

Table 5.5: Time of evolution graphs construction.

Dataset	Time, min	
	BB selection	**Graph constr.**
Montpellier	2.5	9.5
Rostov	7.0	14.5

and 75% (to justify the choice of small τ_1 value). Table 5.5 shows the computation time for different steps of graph construction.

The Figure 5.6 presents an example of an evolution graph that corresponds to the construction of a rugby stadium. The constructed graph correctly reflects the overall change process, although, we can observe that some extracted changes and corresponding segments are not "pure". For example, in timestamp 18/02/2006, some small vegetation objects are attached to larger construction segments. We can also notice that the presented change process is composed of several subprocesses. One of these subprocesses starts at 03/06/2006 and at this timestamp is presented by segment $7 - 29$ that corresponds to some vegetation fields that are later transformed into constructions.

With the given evolution graph parameters, we obtain *4388* graphs for the Montpellier dataset and *1850* graphs for the Rostov dataset (after excluding 2 timestamps length sequences) that are used in totality for the AE models training.

To evaluate the proposed clustering framework, we compare it with hierarchical agglomerative clustering with DTW [144] distance measure used in [142] that is equally able to process time series data with varying length. DTW is a time-series distance measure algorithm that finds optimal match between two time series by scaling one to another. Ward's linkage was used as clustering algorithm parameter. DTW was applied to the extracted graph synopsis calculated for the encoded features (called "DTW features") and to the raw image values (called "DTW w/o features"). We highlight the advantage of our proposed framework by comparing it with the same framework without feature extraction (i.e. we compute the synopsis for raw image values, called "Framework w/o features"). The obtained results are presented in Table 5.6 for different number of clusters in order to choose the optimal one.

For both datasets we equally calculate NMI for primary change classes such as changes in vegetation, construction processes and changes in water (only for the Montpellier dataset) to verify if the extracted clusters contain only one primary type of changes (Table 5.7).

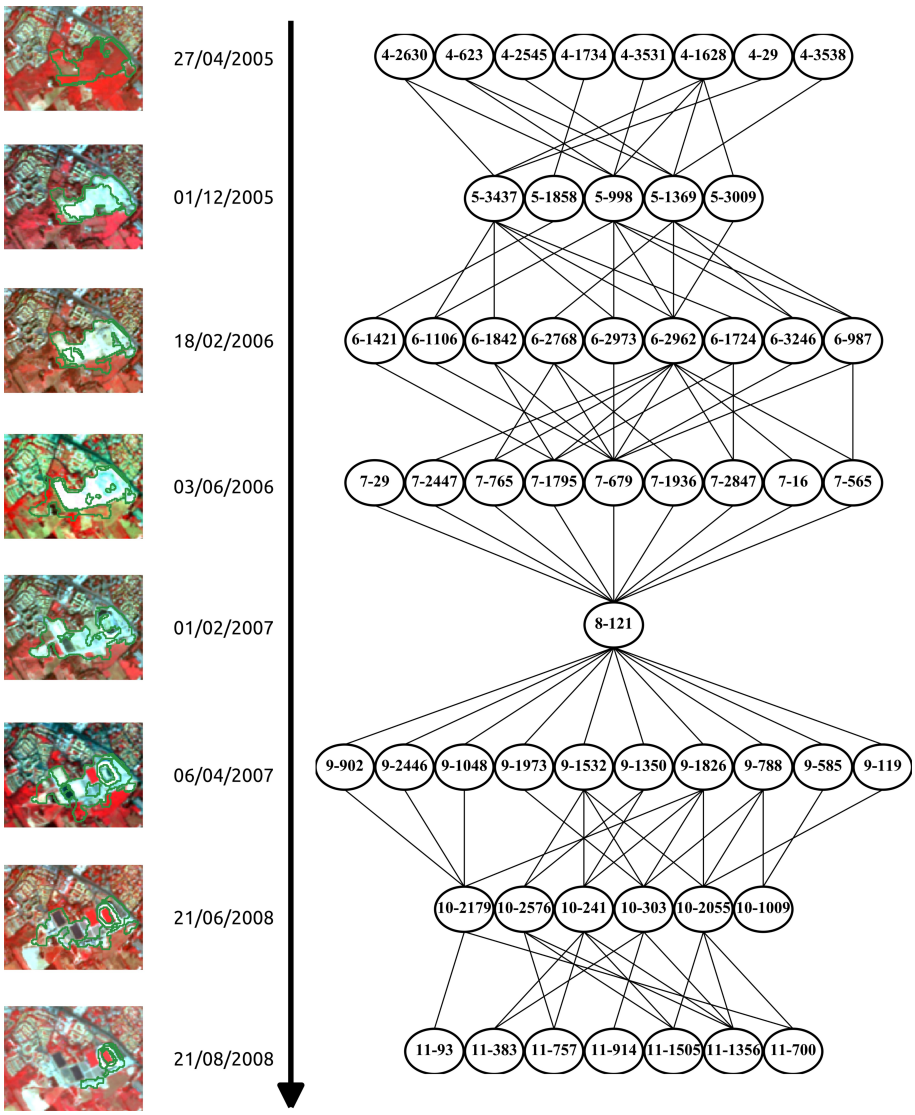

Figure 5.6: An evolution graph of rugby stadium construction.

Table 5.6: Clustering results for reference change classes.

	Method	NMI									
		Number of clusters									
		5	10	15	20	25	30	35	40	45	50
Montpellier	Our framework	0.5	**0.57**	**0.55**	0.55	0.55	0.56	0.56	0.57	0.59	0.59
	Framework w/o features	**0.53**	0.56	0.53	0.54	0.52	0.54	0.55	0.55	0.56	0.56
	DTW features	0.43	0.48	0.49	0.51	0.51	0.56	0.6	0.61	**0.61**	**0.65**
	DTW w/o features	**0.53**	0.51	**0.55**	**0.58**	**0.58**	**0.59**	**0.6**	**0.62**	0.6	0.61
Rostov	Our framework	**0.56**	**0.55**	**0.64**	**0.65**	0.64	0.63	0.7	0.71	0.71	0.71
	Framework w/o features	0.36	0.48	0.52	0.62	0.63	0.65	0.69	0.69	0.71	0.71
	DTW	0.33	0.45	0.58	0.64	**0.67**	**0.71**	**0.71**	**0.73**	**0.73**	**0.73**
	DTW w/o features	0.48	**0.55**	0.56	0.58	0.62	0.62	0.66	0.66	0.67	0.67

Table 5.7: Clustering results for primary classes.

		NMI									
		Number of clusters									
	Method	5	10	15	20	25	30	35	40	45	50
Montpellier	Our framework	**0.65**	**0.73**	**0.65**	**0.64**	**0.64**	**0.64**	**0.61**	**0.6**	**0.62**	**0.62**
	Framework w/o features	0.62	0.58	0.53	0.52	0.5	0.49	0.49	0.49	0.47	0.47
	DTW	0.54	0.51	0.5	0.48	0.48	0.48	0.51	0.51	0.51	0.5
	DTW w/o features	0.55	0.52	0.54	0.59	0.59	0.58	0.57	0.56	0.55	0.56
Rostov	Our framework	0.44	0.36	0.34	0.36	0.35	0.35	0.33	0.34	0.34	0.34
	Framework w/o features	0.27	0.27	0.34	0.31	0.32	0.31	0.3	0.3	0.34	0.34
	DTW	0.27	0.29	0.31	0.29	0.38	0.37	0.38	**0.4**	0.4	0.4
	DTW w/o features	**0.5**	**0.43**	**0.41**	**0.43**	**0.42**	**0.42**	**0.4**	**0.4**	**0.41**	**0.41**

Table 5.8: Computation time.

Method	**Time, min**	
	Montpellier	Rostov
GRU AE + hier.	**4**	**3**
Hier. with DTW	16	4

The execution time of GRU AE combined with hierarchical agglomerative clustering and hierarchical agglomerative clustering with DTW distance matrix measure are presented in Table 5.8. Note that DTW matrix was calculated with *tslearn* library. Computation of DTW distance matrix is a time consuming process, hence, with growing number of change graphs, the computation time may drastically increase. At the same time, GRU AE training takes the same time for both datasets, regardless the number of graphs.

We observe that for small number of classes (10-15) our proposed framework gives the best results for the reference ground truth (10 and 11 classes for Montpellier and Rostov datatsets respectively).

However, for the Montpellier dataset, hierarchical agglomerative clustering with DTW measure outperformed our framework for the number of clusters superior to 15. At the same time, primary classes of the Montpellier dataset are much better separated by our framework, even for small number of clusters. In general, for both datasets, NMI increases with the number of clusters, but at the same time, it may be complicated to interpret the obtained results when its number is elevated.

In our case, as the NMI is calculated at pixel and not object level, we obtain high NMI values for 11 reference classes of the Rostov dataset due to the bigger surface of vegetation changes. As DTW measure does not depend on the classes balance, it has achieved better separation of primary classes for the unbalanced dataset. Nevertheless, for large datasets, DTW computation is time-consuming, so GRU AE is the most adapted for balanced datasets of any size, and DTW approach performs better for unbalanced datasets of a small size.

After visual analysis of the obtained results and analysis of Tables 5.6 and 5.7, we can state that for small number of clusters our proposed framework separates the primary classes much better than the framework without feature extraction. It confirms that feature extraction ameliorates the class generalization for GRU, so the primary classes are better distinguished.

We observe that some construction processes that belong to the same classes for

human eye are regrouped by the algorithm in different clusters. This fact justifies the necessity to extract large objects during the segmentation as over-segmentation may produce numerous change graphs that correspond to the same change process, but are clustered in different groups. However, visual examination of the results with the chosen segmentation parameters shows that some graphs contain several different change sub-processes. For this reason, we may find different classes of objects at the same timestamps in one evolution graph (as in Figure 5.6). This class mixture may potentially influence the graph synopsis and lower the performance of our clustering algorithms.

If we lower the threshold k or increase σ values for the image segmentation, we obtain smaller segments that give us more compact evolution graphs. At the same time, the computation time for evolution graph construction will increase with the number of obtained graphs. In this case, we may obtain numerous graphs that correspond to some sub-processes that will decrease the clustering performances due to the lack of generalization of the obtained change graphs. For example, if we change only the segmentation parameters for the Montpellier dataset to $\sigma = 0.3$ and $k = 6$, we obtain 5316 evolution graphs that are clustered with lower precision. For 15 clusters, we obtain a NMI of 0.49 and 0.5 for 10 classes and primary classes respectively contrary to 4388 graphs and 0.55 and 0.65 NMI values for the initially chosen segmentation parameters. On the other hand, if we increase the threshold k or lower σ values, we might obtain under-segmented change objects that will lead to the creation of less "pure" evolution graphs.

The influence α, τ_1 and τ_3 values on the evolution graph construction was explained in Experimental settings section. Clearly, we can attend similar effect as in previous paragraph if the number of graphs is elevated and, on the contrary, miss some change processes if the graph coverage is not sufficient.

5.6 Conclusion

In this chapter, we have presented an end-to-end approach for change detection and analysis in SITS. The approach consists in the extraction of bi-temporal change maps for the whole SITS that are then analyzed in multi-temporal context in order to construct different change processes that are further regrouped in different clusters. Our framework combines graph based techniques with unsupervised feature learning with neural networks and does not depend on the temporal resolution of SITS and on its length. The proposed approach is fully unsupervised and gives us promising

results on two real-life datasets. However, for unbalanced datasets, we may observe that smaller classes are not well separated from the majority ones.

As our multi-temporal changes are built from bi-temporal contextual anomalies, some of these changes can be interpreted as separate seasonal changes classes that are in minority, comparatively to the overall SITS trend. For this reason, it is indispensable to perform a complete SITS clustering to isolate these minority classes from the real changes. As SITS clustering deals with the whole area, it will not be able to detect the outliers and only no change areas and seasonal changes will be identified. Then, the results of both methods should be analyzed together for better understanding. For example, if one change cluster from the method presented in this chapter matches a cluster from the results of SITS clustering, we can presume that it corresponds to some non-numerous seasonal changes rather that to non-trivial changes.

In the following chapter, we present the final step for SITS analysis that performs the clustering of the whole series to identify no change areas and seasonal changes.

Chapter 6

Satellite Image Time Series Clustering

Contents

6.1 Introduction

Contrary to the past decades, nowadays we dispose of a huge amount of time-spread geospatial data that provide us a full description of almost any area of interest in the world [151]. Exploiting SITS gives us better comprehension of a study area, its landscape, land cover, evolution and more comparing to a single image analysis [152].

While some applications demand SITS analysis in order to detect or monitor a specific event (constructions, droughts, deforestation, etc) [127, 130, 133], others exploit SITS to perform a land cover analysis of the whole area and/or its eventual evolution [153]. For the second type of applications, the prior knowledge about temporal behavior of some classes (usually vegetation) is indispensable to make a correct classification map [136, 137].

In the previous chapter, we have proposed an end-to-end approach for multi-temporal non-trivial change detection and clustering. This algorithm aims to analyze only specific areas of the SITS and ignores no change areas and seasonal changes. In this chapter, we propose an approach to cluster the whole series in order to identify no change areas and seasonal changes. At the same time, as non-trivial changes are considered as the outliers, they will not be identified by the algorithm proposed in this chapter. For this reason, the results of both algorithms presented in this and previous chapters should be interpreted together.

The proposed SITS 3D clustering algorithm presented in this chapter is based on SITS compression with a 3D convolutional AE model. The whole SITS is encoded into a unique image. SITS objects are then identified with two steps (preliminary and fine-tuning) segmentation approach that uses both original images and the encoded series. Finally, the segments are clustered using their encoded descriptors.

The rest of this chapter is organized as follow: in Section 6.2 we overview existing works for SITS clustering, Section 6.3 presents the proposed approach, Section 6.4 describes datasets we used, Section 6.5 gives the review of the experimental results with their qualitative and quantitative evaluation. In the last section, we resume the work done.

6.2 Related Works

Due to the variety of objects presented in the remote sensed images and in SITS in particular, few labeled data are available. For this reason, unsupervised approaches are becoming more and more popular for various projects. Most of the currently used unsupervised approaches for SITS clustering deploy pixel-wise analysis [154, 155]. In these approaches, the pixels corresponding to the same geographical position on different images form temporal sequences that are further compared to each other and associated to different classes. Numerous studies have proven Dynamic Time Warping (DTW) algorithm [144] to be an efficient tool to compute the similarity measure between temporal sequences. The main idea of this approach is to non-lineary map

one series to another by minimizing the distance between them. Thus, the DTW distance matrix is computed for every point of the series and used as a similarity measure for a chosen clustering algorithm.

In general, DTW distance matrix has a high computational cost. To this end, the analysis of large datasets at pixel level may be extremely time-consuming and, hence, unreasonable. To deal with this issue, several object-based DTW clustering approaches have been proposed [156–158] to analyze the data both at temporal and spatial dimension. In these methods, spatio-temporal segments (in a form of a 2D map) are extracted for the whole SITS, then, the temporal sequences constructed for segment descriptors are clustered. Therefore, the object-based SITS analysis has drastically reduced computational cost and ensured more homogeneous results of clustering algorithms compared with the pixel-based approaches.

Nevertheless, not so many SITS segmentation approaches are available in literature [159] and it can be tricky to create a proper segmentation map for the whole series as sometimes objects change from one image to another. If a series is short enough (does not cover more than a year), we can presume that objects shapes stay invariant and, in this case, we can project a single image segmentation to the whole SITS. However, this approach can not be used for a series that covers a large period of time, especially if it contains some permanent changes or important phenological variations. To capture some of these changes, segmentation may be performed on the concatenated product of two or three most representative images of the SITS [156] or even on the concatenated product of the whole time series [157]. In the first case, we may miss some objects. In the second one, the segmentation may have high computational cost and be difficult to parameterize if a SITS is long.

To overcome multi-temporal segmentation issues, in [143] the authors propose a graph-based approach to analyze different spatio-temporal dynamics in SITS. In this method, each image is segmented independently and all the spatio-temporal entities that belong to the same geographical location are connected to each other and form evolution graphs. Every graph is characterized by a bounding box - an object which footprint has the intersection with all graph objects at different timestamps. Following this method, the authors of [142] propose an algorithm to cluster the extracted multi-annual graphs. Each evolution graph is firstly described by a simplified representation - synopsis. Secondly, spectral and hierarchical clustering algorithms with DTW distance measure are applied to graphs synopsis. This approach showed promising results for the clustering of natural habitat areas. However, it may be complicated to construct evolution graphs for urban areas as their segmentation is more complicated

due to the non-homogeneity of the features. For this reason, the segmentation results of urban areas are usually not uniform from one image to another, contrary to the agricultural lands where a parcel is presented by one or two well-delimited segments that repeat over time if no changes happened.

To create a single segmentation map for the whole SITS, the authors of [160] propose a time series segmentation approach based on DTW distance measure. In this approach, at the beginning, each pixel is characterized by its temporal sequence, each sequence firstly represents an isolated segment, then the segments with similarity measure higher that a certain threshold are iteratively merged. However, for the aforementioned reasons, we estimate that the proposed approach can be slow, even if the distances are not computed for all pixel couples.

In this chapter, we propose a SITS object-based clustering algorithm based on SITS compression with 3D convolutional autoencoder (AE). 3D convolutional networks have been successfully used in remote sensing applications for supervised classification [161, 162] due to its ability to deal with multi-temporal image data in addition to lower computational cost comparatively to other temporal models such as, for example, convolutional LSTM network [163]. Contrary to these methods, our 3D convolutional AE model is unsupervised and does not require any labeled data and, to our knowledge, no such models have been used in time-series remote sensing yet.

In our work, we deploy an AE neural networks structure. Traditionally, autoencoders are used for unsupervised dimensionality reduction or feature learning [95]. Different AE models have been widely used in remote sensing [93, 122, 164]. In these articles, the features are extracted from a single image using AEs and then used for a land scene classification. However, the AE structure can be adapted for any type of data, therefore, we propose to use AEs for the feature extraction and compression of the image series.

In our method, we first encode the whole SITS into a new feature image with a multi-view 3D convolutional AE. Both encoder and decoder parts contain two branches that are concatenated together before the bottleneck. While the first branch allows to obtain deep features from the spectral bands of the whole SITS, the second one only extracts some general information from the corresponding NDVI [83] images. Second, we perform a preliminary segmentation of the SITS on its two most representative images. Then, we correct the preliminary segmentation by using the encoded feature image. Finally, we regroup the obtained objects with hierarchical clustering algorithm [49] using the encoded features as descriptors. The proposed

approach showed us good results in the two real-life datasets and outperformed the concurrent methods, including the ones based on the DTW measure.

6.3 Methodology

Our proposed approach is developed for segmentation and clustering of a SITS. Let R_S be a time series of S co-registered images Im_1, Im_2, ..., Im_S acquired at timestamps T_1, T_2, ..., T_S. The framework is composed of several steps which are the following:

1. We start by relative normalization of all the images of the SITS using an algorithm described in [123] and correction of saturated pixels.
2. We deploy a two-branch multi-view 3D convolutional AE model in order to extract spatio-temporal features and compress the SITS.
3. Then, we perform a preliminary SITS segmentation using two farthest images of the dataset taken in different seasons.
4. We correct the preliminary change segmentation using the compressed SITS.
5. Finally, we perform the clustering of extracted segments using their spatio-temporal features as descriptors.

6.3.1 Time Series Encoding

For the compression and encoding of the SITS, we propose to use the two-branch multi-view 3D convolutional AE. While the first branch of the AE extracts deep temporal features from the initial series, the second one extracts some primary temporal features from the associated NDVI images (Figure 6.1). The NDVI branch improves the model capacity to distinguish different vegetation types, especially the ones with weak seasonal variance. Moreover, by allocating a separate branch to NDVI images instead of just adding a supplementary NDVI channel to the initial images, we "force" the model to extract more robust and independent vegetation features.

Contrary to traditional 2D convolutional networks where convolution filters are applied in 2D plane, 3D convolutions preserve the temporal relations between data by extending the filters to the depth dimension [165]. Therefore, the 3D convolution network extracts both spatial and temporal features.

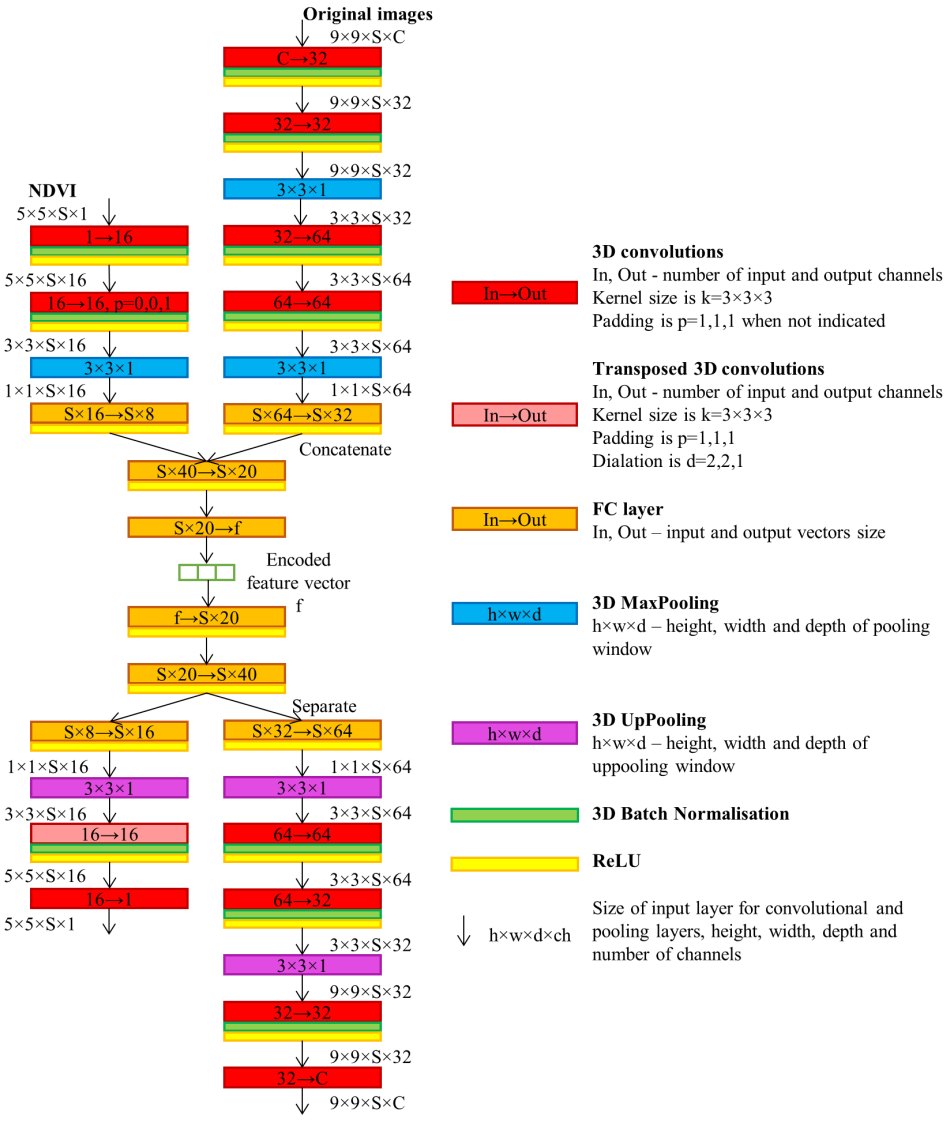

Figure 6.1: 3D convolutional AE model.

The deployment of an AE type of model ensures the extraction of robust spatio-temporal features in an unsupervised manner without using any reference data. In classic AEs, the model firstly encodes the input data in some compressed latent representation and then decodes it back to its initial self. In image processing, the encoding pass is usually composed of convolutional and pooling layers for feature maps (FM) extraction that are followed by some fully-connected (FC) layers for feature compression. The decoding pass is often symmetrical to the encoding one. Once the model is trained, the extracted compressed representation is used to describe the data under the study. The encoder-decoder model allows us to compress the whole dataset in an uniform way. Moreover, it can compress any type of data independently

of its shape and size.

In case of our multi-view AE, during the encoding step, we independently extract features from two different stack of images (original and their corresponding NDVI), the features are then merged together to obtain a combined descriptor. During the decoding pass, the features are separated and reconstructed independently into the initial stacks of the original and NDVI images.

The training and encoding processes of the whole series are performed patch-wise for the stack of SITS images. The patches of size p are extracted for every i, j-pixel of the SITS ($i \in [1, H]$, $j \in [1, W]$, where H and W are images height and width respectively) and represent stacks of size $p \times p \times S \times B$, where B is the number of image bands. Obviously, for the first branch, B corresponds to the number of spectral bands, for the second one $B = 1$ as we deal with single channel NDVI images. To extract deep features from the original images, we propose to use patches of size $p = 9$, however, as we extract only general information from the NDVI images, the patch size of $p = 5$ is sufficient. We consider that $p = 9$ is big enough to get necessary information of the neighbor pixels as it makes a 90×90 m^2 surface footprint. In addition, it ensures smooth maxpooling with 3×3 window size and does not produce important border effect for the patches that contain two (or more) different classes (see more about it in the next subsection). For the NDVI branch, we believe that $p = 5$ is the minimum sufficient patch size to get the information about the neighborhood vegetation features ($p = 3$ covers only 1 pixel radius, so this information can not be considered relevant). Moreover, we apply no padding to the second 3D convolutional layer of the NDVI branch to reduce the size of extracted feature maps before applying the maxpooling operation. Note that we tend to decrease the network complexity and its training time by choosing a smaller NDVI patch size as all the important information about land cover textures are extracted in the main branch, while the NDVI branch is used only to detect vegetation tendencies. As one may observe from the model schema, the configuration of FC layers depends on the number of images of the SITS. It guarantees that all the layers within different models have the same input/output step ratio while compressing the features. Note than if S is elevated, one might consider to add a supplementary FC layer.

For model evaluation and optimization, we use the mean-squared error (MSE) (3.2).

Once the model is stable, every temporal stack of patches is encoded in a feature vector of size f that corresponds to the i, j-pixel of a new feature image of size $H \times W \times f$ that will be further used as a compressed version of the whole dataset.

6.3.2 Segmentation

Satellite image segmentation is a task of image processing that partitions an image into non-intersecting regions (segments) so that the ensemble of pixels of each region shares similar properties. Segmentation can therefore be seen as a first step before doing a classification or a clustering of the newly created segments for any object-based method.

As it was mentioned in the previous section, SITS segmentation can be a complicated and challenging process, especially when the number of images is elevated. The main idea of our segmentation approach is the following: to get a more robust SITS clustering that is easy to visualize, we need to obtain a unique segmentation map for the whole series. To accomplish this task, we could directly perform the segmentation on the encoded SITS image. However, as the encoding is performed in a patch-wise manner for every image pixel, one may observe a border effect. This effect is produced for pixels located close to a border of two regions. The patches extracted for these pixels contain information about two (or more) different classes, their encoded spatio-temporal features will not be "pure". For this reason, these pixels may be segmented as new objects (mostly linear) or segment borders may be shifted. Moreover, the linear objects, such as roads or rivers may not be distinguished or, on the contrary, over-segmented. Figure 6.2 presents two examples of the border effect and its eventual correction with our method (explained later in the text). The first row shows the shifted borders in crop segmentation at the limits of different types of crops. The second row displays the segmentation of a road. We can observe that the road is over-segmented and its borders are shifted at the same time.

To tackle this problem, we propose to perform a two steps segmentation that includes the correction of the preliminary segmentation in respect to all objects borders of the time series. The preliminary segmentation is performed on two most representative concatenated images of the SITS. To obtain the maximum of coherent spatio-temporal objects in the preliminary segmentation Seg_{pr}, the chosen images should be as far apart as possible (e.g. the first and the last image) and correspond to different seasons.

For all image segmentations, the MeanShift algorithm [166] available in *Orfeo ToolBox* software (`www.orfeo-toolbox.org`) under *QGIS* interface was chosen. The most important parameters of the MeanShift segmentation algorithm are spatial radius R_s and range (spectral) radius R_r. The main idea of the algorithm is to firstly reproject a n-channels image into n-dimensional space and simplify its representation by replacing each pixel with the mean of the pixels in R_r neighborhood that have

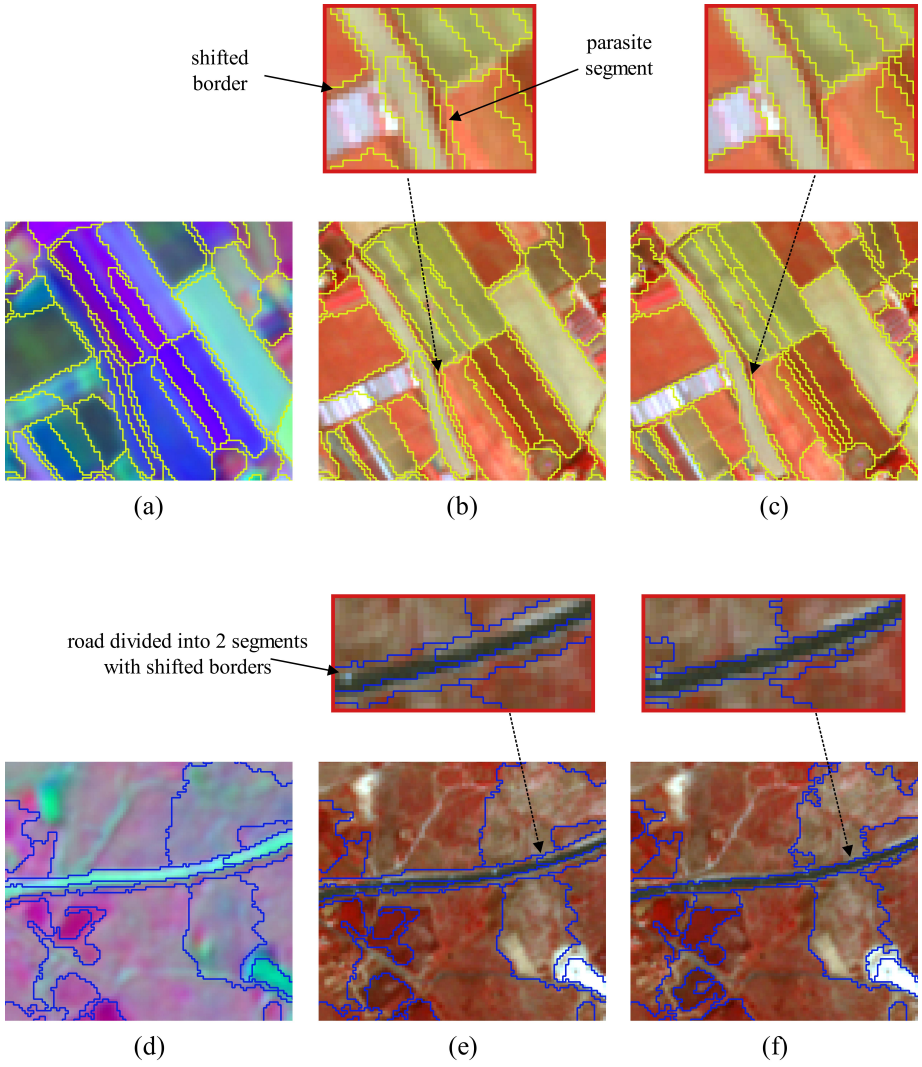

Figure 6.2: Influence of the border effect on the segmentation results and its eventual correction, example issued from SPOT-5 dataset (see dataset description in the following section). Top row - border effect in crops segmentation, bottom row - border effect in road segmentation. (a), (d) - encoded images and corresponding segmentations, (b), (e) - projection of these segmentations on the last image of the dataset, (c), (f) - images corrected for the border effect. Note that 10-feature encoded image is presented in the limits of 3 channel combination, original SPOT-5 image is presented in false colors.

values within R_s. The regions smaller than Reg_{min} are merged. Secondly, the algorithm reprojects the data back into the image plane and separates the areas with the same mean value into non-overlapping segments. At the end, the segments smaller than O_{min} are merged with their neighbors.

Despite the fact that Seg_{pr} gives us correct segment borders, it is impossible to identify all the objects presented in SITS on the base of only two images. Therefore, in the next step, we perform the segmentation Seg_{enc} of the encoded SITS that is represented as a f-channels image. As it was mentioned before, this segmentation would contain numerous irrelevant objects and shifted borders. Finally, we choose Seg_{pr} as the reference and we correct it by fitting the segments from Seg_{enc} to obtain the final segmentation map Seg_f.

The correction process is performed separately for each segment and is the following (see Figure 6.3):

- Let P_i be a segment from Seg_{pr} to correct.
- We firstly fill P_i with the segments from Seg_{enc} that have spatial intersection with it. P_i borders are preserved and used as the reference.
- Second, we check the average width of these segments in horizontal and vertical axes of the SITS coordinate system. We select objects with width smaller than min_{width} in at least one of the axes. min_{width} size should not exceed half of the encoder patch size and be set after estimating the influence of the border effect.
- At the third place, each of these objects is merged with a neighbor with the biggest common edge if the edge is at least 3 pixels long or if the object's size does not exceed O_{min} (minimum object size that we want to distinguish in our experiments). Note that in case we have several segments to merge, we sort them by ascending size and start by merging the smallest one while other segments sizes are being iteratively updated.
- Finally, we fill a new segmentation map Seg_f with new merged segments.

Our method might still produce some shifted borders for some corrected segments, but at the same time, it allows to reduce the border effect to minimum, to preserve correct shapes of linear objects and to avoid parasite segments that correspond to border pixels.

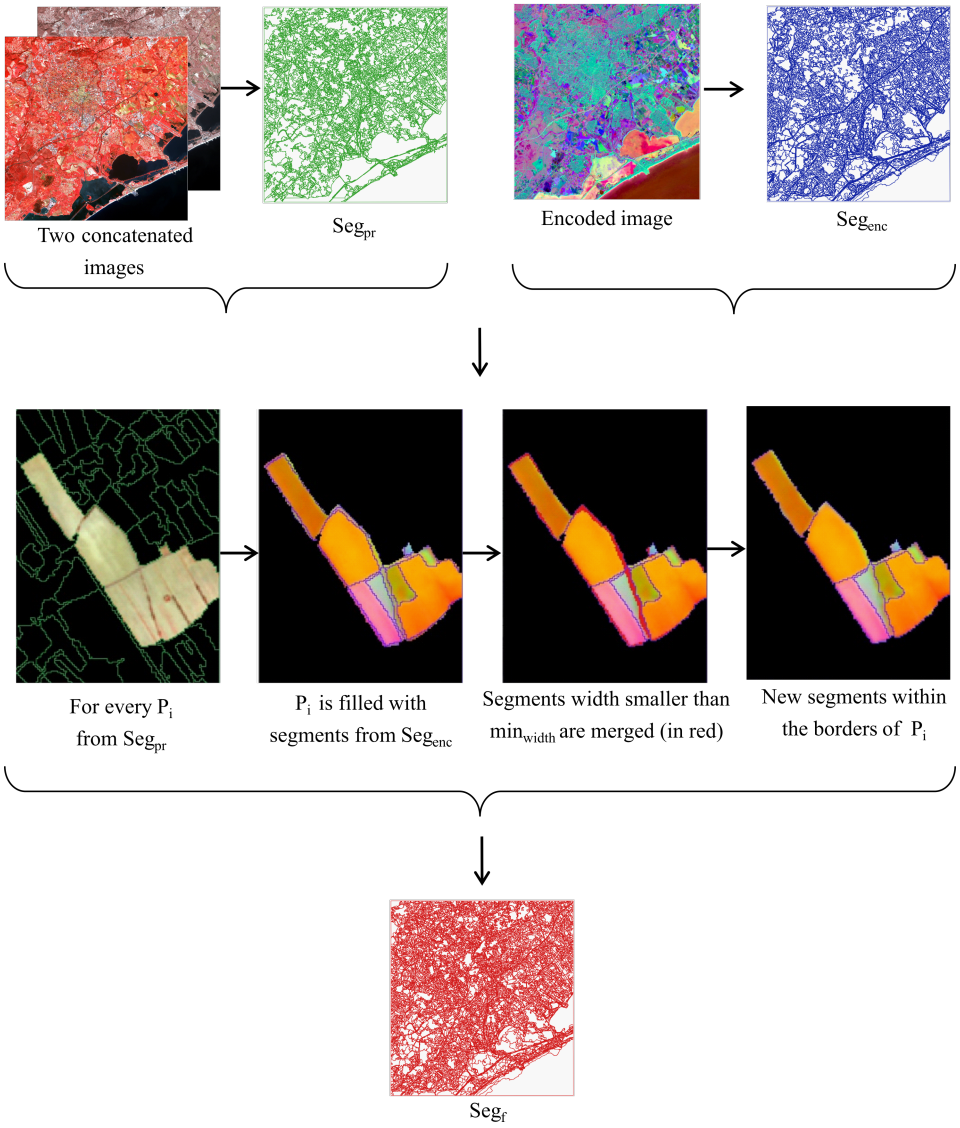

Figure 6.3: Segmentation correction.

6.3.3 Clustering

To regroup the obtained segments, we deploy the hierarchical clustering algorithm (HCA) [49] applied to segments descriptors. The choice of HCA is justified in Chapter 2.6.

For segment descriptors, we use the median values of the encoded features of pixels within these segments. We choose the median values over the mean ones so the border pixels are not taken into account. We use Ward's linkage[49] and Euclidean distance between the segments as parameters for clustering algorithm.

6.4 Data

We evaluate the proposed approach on two real-life publicly available time series issued from SPOT-5 and Sentinel-2 missions. Both SITS are taken over the same geographical location (Montpellier area, France), but, however, differ in terms of spectral and temporal resolution.

While the first SITS contains 12 images that are irregularly samples over 6 years, the second one contains 24 images taken over 2 years with more regular temporal resolution. The description of the SPOT-5 and Sentinel-2 Montpellier datasets can be found in Appendices A.1 and A.2 respectively.

All SPOT-5 images provide green, red, NIR and SWIR bands with 10-meters resolution. Sentinel-2 images provide multiple spectral bands of different spectrum and spatial resolution, however, it was decided to keep only 10-meters resolution spectral bands - blue, green, red, NIR.

The ground truth (GT) for both datasets was taken from an open data website of Montpellier agglomeration[1] and correspond to landcover maps which we have manually modified to keep only distinguishable classes and merged the look-alike classes. While for the SPOT-5 dataset we have used Corina Land Cover (CLC) map of the 2008 year, for the Sentinel-2 dataset CLC of the 2017 year was taken. We have defined 9 well-distinguished GT classes:

1. urban and artificial area,
2. wooded area (include forests, parks, family gardens etc),
3. natural area (not wooded),
4. water surface,
5. annual crops,

[1] `http://data.montpellier3m.fr/`

6. prairies,
7. vineyards,
8. orchards,
9. olive plantation.

For both datasets, the olive plantation class is very small, so we choose 8 reference classes for our clustering algorithm. The GT olive plantation class will be ignored during the evaluation.

Note that it is difficult to create a GT for a multi-annual SITS analysis as some objects may go through changes and it is impossible to detect all these changes manually. For this reason, for the SPOT-5 dataset, we use the GT that corresponds to the last year of the SITS. The SPOT-5 dataset was taken over 6 years and contains many change processes, mostly such as different constructions and permanent crop rotations. The study for change detection in the SPOT-5 dataset is presented in Chapter 5. As these changes are less numerous, they will be considered by most of clustering algorithms as outliers, hence, they will be mixed with "stable" classes. However, some of these changes are only several timestamps long, so we still perform the clustering of the whole SITS instead of only free-change areas. Thus, the change areas will be regrouped with no change areas with the most similar temporal behavior or even make their proper clusters. At the same time, we consider that the Sentinel-2 dataset does not have any or has very few change areas as it is spread over only two years.

6.5 Experiments

All the algorithms were tested on 6 cores Intel(R) Core(TM) i7-6850K CPU 3.60GHz with 32 GB of RAM computer with a NVIDIA Titan X GPU with 12 GB of RAM and developed in *Python* programming language using *PyTorch 1.3* library on Ubuntu 16.4. For segmentation, we used *Orfeo ToolBox 6.6.1 under QGIS 2.18*. *tslearn* library [150] was used to calculate DTW distance matrices for the concurrent approaches.

6.5.1 Experimental Settings

Time Series Encoding

As it was mentioned in Section 6.3, we have the same AE model for both datasets with the parameters that depend on the time series length. Thereafter, we set different

sizes of the encoded feature vector f for our datasets that is proportional to the SITS length. f values were obtained empirically and correspond to $f = 10$ and $f = 20$ for the SPOT-5 and Sentinel-2 datasets respectively (for 12 and 24 images in the datasets).

During the model training, we equally add Gaussian noise with 0.25 factor to the input patches of the original images to extract more robust and generalized features. We do not add any noise to the NDVI patches as it reduces model capacities to differentiate some minor variations in the vegetation.

After several trials to assess the best parameters values, we set learning rate to 0.0001 and batch size to 150 to ensure the most optimal model converging during the training. We use all SITS patches during the training of the model. We train the model for both datasets for 2 epochs until it is stable. The number of epochs was obtained after the analysis of loss trend and the visual analysis of the encoded images. However, if one does not dispose of a sufficient graphic memory for 2 epochs model training, it can be trained for only one epoch without significant loss in accuracy.

Preliminary Segmentation

As it was mentioned earlier, we perform the preliminary segmentation of two concatenated images of the dataset that should be as far apart as possible and belong to two different seasons. For the SPOT-5 dataset, these images were the first and the last images of the dataset (taken on 2002-10-05 and 2008-08-21), for the Sentinel-2 - the fifth and the last image (taken on 2017-06-12 and 2018-12-29). The chosen segmentation parameters for the MeanShift algorithm are presented in Table 6.1, other parameters are used by default in the *Orfeo ToolBox* software, including $Reg_{min} = 100$. For the choice of range and spatial radius, note that pixel values of SPOT-5 images do not exceed 475, while the Sentinel-2 images have 4096 maximum pixel value (after the elimination of saturated pixels). To simplify the choice of the parameters and reduce the computation time, the pixel values of the Sentinel-2 images were divided by 10 only for segmentation to bring these values closer to the ones of the SPOT-5 images.

The segmentation parameters were chosen to obtain the most relevant results for the reference objects.

Correction of Segmentation Results

As for the preliminary segmentation, we use MeanShift algorithm to segment the encoded images. Initially, the pixel values of encoded images are contained between -1 and 1 and then they have been re-scaled between 0 and 255 for the segmentation.

Table 6.1: Preliminary segmentation parameters.

	Parameters		
Dataset	R_s	R_r	O_{min}
SPOT-5	45	40	10
Sentinel-2	40	35	10

The choice of both radiuses fully depend on the number of encoded features f and is the following: $R_s = 3 \times f$, $R_r = 2 \times f$. $O_{min} = 10$ as for preliminary segmentation.

For the correction of segmentation results we set $min_{width} < 4$ as it corresponds to the radius of neighborhood for the patches extracted for the original images.

Clustering

As in any unsupervised algorithm, the obtained clusters often do not correspond to the ones defined by the human. In most cases, one real-life class is often divided into two or even more clusters, so the manual inspection of the results is needed to choose the right number of clusters. For this reason, we perform clustering for different number of classes (from 8 to 15). We evaluate the obtained results with Normalized Mutual Information (NMI) index [71].

6.5.2 Results

To evaluate the proposed clustering framework, we compare it with different time series clustering approaches: object-based DTW, graph-based DTW [142], 3D convolutional AE without NDVI branch and a variation of our pipeline without segmentation correction.The following parameters are set for the concurrent approaches:

- **Object-based (OB) DTW** - as the segmentation reference we use the preliminary segmentation results for our method, we exploit hierarchical clustering algorithm with DTW distance matrix to regroup the obtained segments.
- **Graph-based (GB) DTW** - we use MeanShift algorithm with the following parameters to segment every image of the SITS $R_s = 20$, $R_r = 20$, $O_{min} = 10$ for SPOT-5 dataset and $R_s = 20$, $R_r = 15$, $O_{min} = 10$ for Sentinel-2 dataset. For both datasets we use the following parameters for graph construction, see [142] for details: $\alpha = 0.3$, $\tau_1 = 0.5$. We omit τ_2 as it lowers the quality of the results. The hierarchical clustering method with DTW distance matrix is applied as in the original article.

– **3D convolutional AE without NDVI branch** - we use the same pipeline and segmentation parameters as in our proposed method, however, the feature extraction model is different. For feature maps extraction, we have the same 4 convolutional and 2 maxpooling layers as in the original images branch that are followed by 2 FC layers with the following input and output sizes for the encoding part: $S \times 64 \rightarrow S \times 12$, $S \times 12 \rightarrow f$. The decoder FC layers are symmetrical to the encoder.

– **3D convolutional AE without segmentation correction** - we use the same pipeline and segmentation parameters as in our proposed method, however, the clustering is performed for the preliminary segmentation made for two concatenated images.

For all methods with the DTW matrix computation we apply Sakoe-Chiba constraint [167] with bandwidth=2 to speed up the algorithm and improve its results. This constraint restricts the alignment of the time series and prevents the shifting greater that one timestamp. We use hierarchical algorithm with Ward's linkage [49] and Euclidean distance between the segments to cluster the SITS as in our methods. Mean segment descriptors are used as in the initial articles.

Moreover, for our method and for the object-based DTW, we equally perform clustering for the ground truth segmentation to analyze the robustness of the proposed object descriptors with an "ideal" segmentation map (referred as "GT seg." in the resulting table). All DTW algorithms are computed for the original images and for the ones enriched with the NDVI band.

We do not compare our method to any pixel-based approaches due to their high computation cost for the chosen datasets related to the distance matrix size and the memory allocation.

Quantitative Evaluation

The evaluation of the obtained results with the NMI index is presented in Table 6.2. For the aforementioned reasons, we present NMI for the reference number of classes (8) as well as the best NMI score within the selected range of classes "NMI best" in the table). As the results obtained with the AE methods may vary between several attempts, each AE method was launched 3 times. For these methods, we present the mean NMI value with the error margins.

As it was mentioned in Section 6.4, the SPOT-5 dataset was taken over 6 years and contains several changes. However, the GT corresponds to 2008 year (the end of

the series). For this reason, we compute NMI for the whole SITS as well as for the SITS without outliers (which are mostly made of these changes and therefore do not match the 2008 end of the series GT). We exclude the airport area when computing NMI as it is mostly represented by a grass field. Its temporal behavior is unknown to us, so we can not associate it to any of existing clusters.

Table 6.3 presents the computation times for the most essential and time consuming steps of the presented approaches. For the DTW methods, we also present the number of segments or graphs to give an idea about the size of the distance matrix as it is the most defining parameter of the computation time. Note that we present the results for the DTW matrix computation made with *tslearn* library based on CPU computations with parallelisation. For a fairer comparison, we also customized parts of *tslearn* source code from the original library to run it on GPU with *numba* library. Doing so has greatly improved the computation time (maximum computation time for the longest sequence was less than 2 minutes).

Table 6.2: Accuracy of different methods.

	SPOT-5		**SPOT-5 w/o outliers**		**Sentinel-2**	
	NMI	NMI best	NMI	NMI best	NMI	NMI best
Our method	**0.45±0.01**	**0.45±0.01**	**0.5±0.01**	**0.5±0.01**	0.44±0.01	**0.45±0.01**
OB DTW	0.4	0.4	0.43	0.46	0.36	0.38
OB DTW+NDVI	0.41	0.41	0.44	0.44	0.36	0.37
GB DTW	0.36	0.38	0.37	0.39	0.36	0.36
GB DTW+NDVI	0.41	0.42	0.42	0.42	0.37	0.37
AE w/o NDVI	**0.45±0.01**	**0.46±0**	0.48±0.02	0.48±0.02	0.43±0.01	0.43±0.01
AE w/o segm. corr.	0.42±0.01	0.42±0.01	0.47±0	0.47±0	0.43±0.01	0.43±0.01
Our method+GT seg.	0.52±0.01	**0.52±0.01**	0.58±0.01	**0.59±0**	0.46±0.05	**0.5±0.01**
OB DTW+GT seg.	0.51	**0.53**	0.52	0.54	0.48	0.48
OB DTW+GT seg.+NDVI	0.52	**0.53**	0.54	0.54	0.4	0.45

Table 6.3: Computation time.

	Algorithm step	Computation time, min	
		SPOT-5	Sentinel-2
Our method	Model training[a]+encoding	15+5	25+6
AE w/o NDVI	Model training[a]+encoding	11+3	15+4
OB DTW	DTW calc.[b]	35(4433 objects)	30(3751 objects)
OB DTW+NDVI	DTW calc.[b]	40(4433 objects)	33(3751 objects)
GB DTW	Graph constr.+DTW calc.[b]	26+33(3550 graphs)	55+51(3739 graphs)
GB DTW+NDVI	Graph constr.+DTW calc.[b]	26+36(3550 graphs)	55+56(3739 graphs)
OB DTW+GT seg.	DTW calc.[b]	202(10725 objects)	299(12236 objects)
OB DTW+GT seg.+NDVI	DTW calc.[b]	232(10725 objects)	335(12236 objects)

[a]Time given for one epoch.

[b]Time given for the full computation using *tslearn* library.

It is still worth mentionning that GPU transfer is not adapted to all types of data, for instance even with a GPU, the computation time increases greatly when the sequence lengths grows. Moreover, regardless of whether it is paralellized on a GPU or a CPU, DTW computations are limited by the number of sequences: the size of the distance matrix grows with the number of sequences, and at some point can not be stored in memory anymore.

Please, note that the computation time of our algorithm can be improved by around 30% with the parallelisation of branches computations.

Qualitative Evaluation. Segmentation

Firstly, we analyze the results obtained for the "ideal" GT segmentation for our proposed method and for the object-based DTW. We can observe from the Table 6.2 that both methods gave higher scores comparatively to the user-made segmentation. At the same time, in average, our method has slightly outperformed its concurrent approach. Moreover, when using the existing *Python* libraries, our method has less computational cost, hence, is more adapted for big datasets.

For the clustering results obtained for user-made segmentation, we observe that the results of our method are significantly better than the concurrent approaches. It can be explained by more precise segmentation results that were achieved by our proposed methodology. We remind that for our method, segmentation is realized in several steps: preliminary segmentation, segmentation of the encoded SITS image and the final corrected segmentation that combines the previous ones. At the same time, the object-based clustering method is performed on the preliminary segmentation results. The preliminary segmentation results are always under-segmented as they can not caption all the seasonal vegetation variations that are reflected in the encoded SITS image.

The obtained results highlight the necessity of the segmentation correction as one can notice the decreasing of the accuracy when none is performed. Figure 6.4 presents the advantage of our proposed approach with higher detalization level over the segmentation performed on the two most representative images of the dataset.

Moreover, our proposed approach correctly produces segments free of border effect that perfectly correspond to true object borders (see Figure 6.2) that facilitate the interpretation of the obtained objects.

The graph-based DTW approaches gave the results similar to the object-based DTW methods in addition to the slowest computation time related to the graph construction. This poor performance can be explained by the difficulty to segment

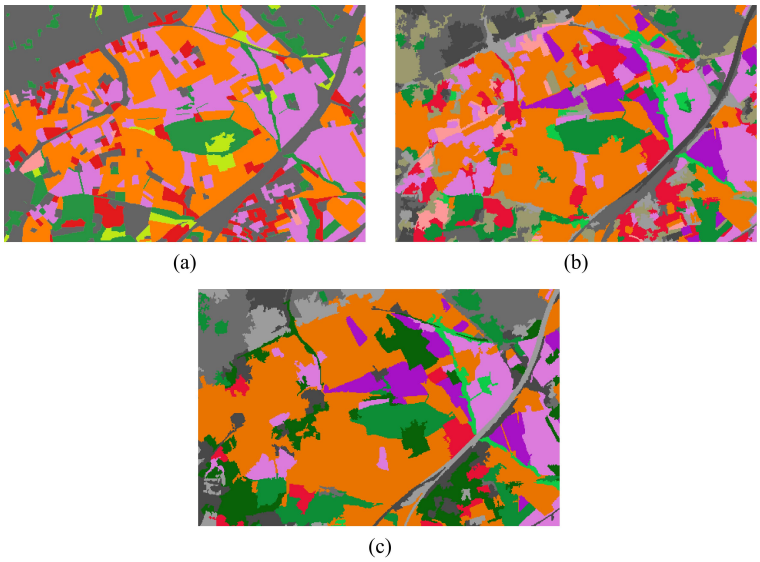

Figure 6.4: Example of clustering results highlighting the advantage of our proposed segmentation correction. (a)- GT for 2017 year, (b)- results for our method for the Sentinel-2 dataset, (c)- results for our pipeline without segmentation correction for the Sentinel-2 dataset (clustering performed on objects extracted from the segmentation of two most representative images). The clusters are colored accordingly to the GT map. The reference map legend can be consulted on Figure 6.5.

the whole SITS, especially in the urban area. However, this method might give better results for a smaller dataset with prevailing agricultural areas.

Qualitative Evaluation. Feature Extraction and Clustering

Figure 6.5 presents clustering results for our method for both datasets for the best of 3 runs. After visual analysis and the analysis of the NMI values, it was established that 14 clusters provided the best data partitioning for both datasets. The obtained clusters were associated to the ground truth clusters and colored in the same manner.

As it was mentioned in the data section, the olive plantation class was ignored during the analysis due to its small size in addition to the fact that none of the proposed algorithms has identified it in a separated cluster.

Clusters corresponding to water surfaces and urban areas were correctly identified for both datasets. Three vegetation classes corresponding to annual crops, vineyards and prairies were mostly correctly detected as well due to the specificities of their temporal behavior. We do not dispose of finer classification level for annual crops, however, the hierarchical algorithm has divided it in several clusters which we can easily distinguish by analyzing crops temporal behavior. At the other hand, the

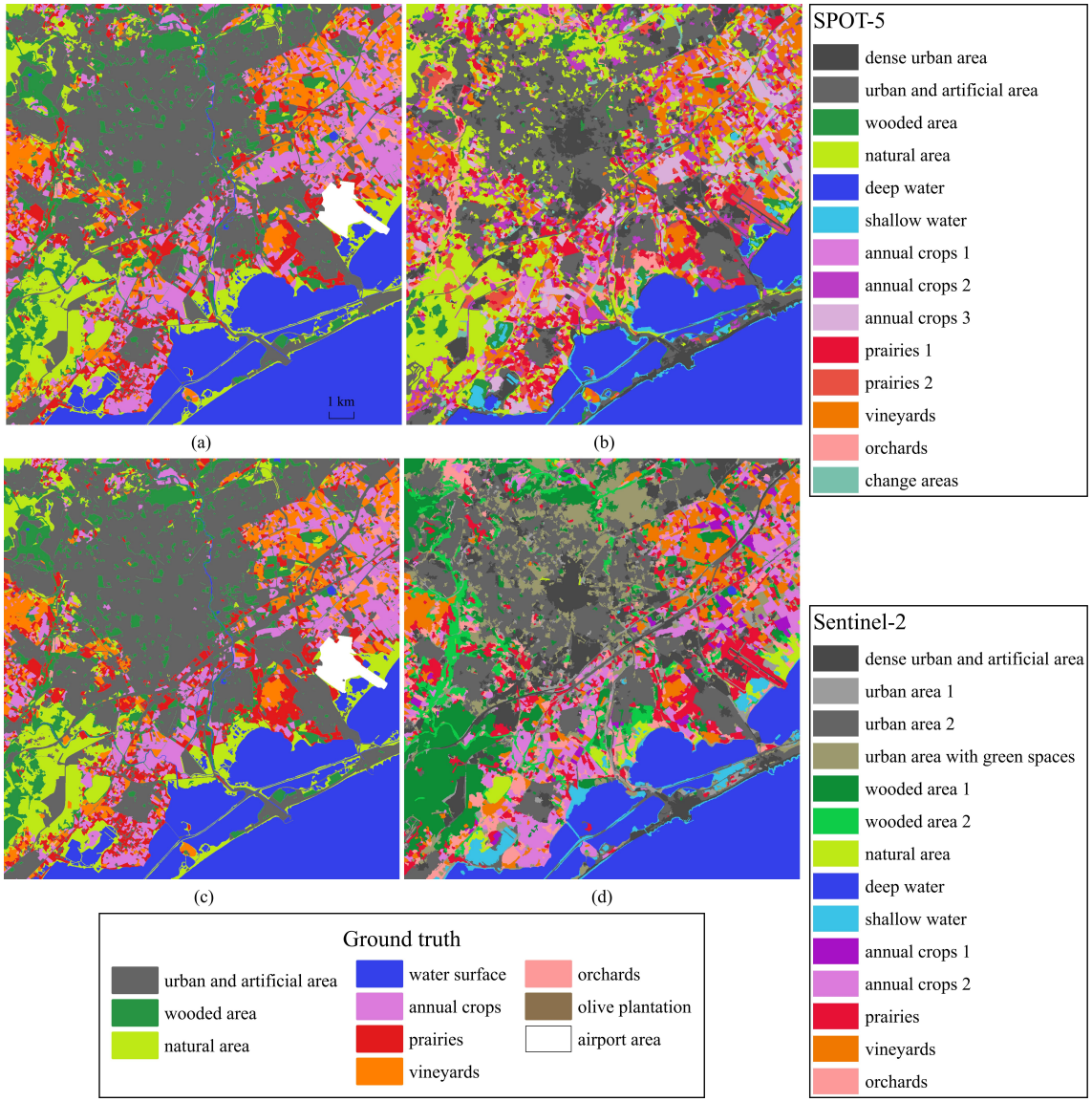

Figure 6.5: Clustering results for our proposed pipeline. (a)- GT for 2008 year, (b)- results for the SPOT-5 dataset, (c)- GT for 2017 year, (d)- results for the Sentinel-2 dataset.

orchard class is mixed with other vegetation classes for both datasets and the natural areas with small vegetation are mixed with wooded areas. It can be explained by the fact that both classes have similar growing cycles and spatial textures.

After comparing the obtained clusters with change maps obtained in Chapter 5, we associate one of the clusters obtained for SPOT-5 dataset with changes corresponding to the construction of a new area.

Unfortunately, some linear objects are not correctly clustered despite the fact that they are correctly segmented. Linear objects presented by rivers and narrow vegetation areas are mostly misclassified. We find several explanations for it: the main reason is that the segments are too narrow and contain many encoded feature pixels affected by the border effect. For the clustering, we use the median feature values of the segments as their descriptors which are biased with border pixels. On the other hand, the algorithm does not detect roads as a separate class, but it still classifies them as different variations of the urban cluster most of the time. We explain it by the fact that the detected roads are in average a little bit larger than other linear objects and their feature response is better defined, hence, their median segment values are less biased.

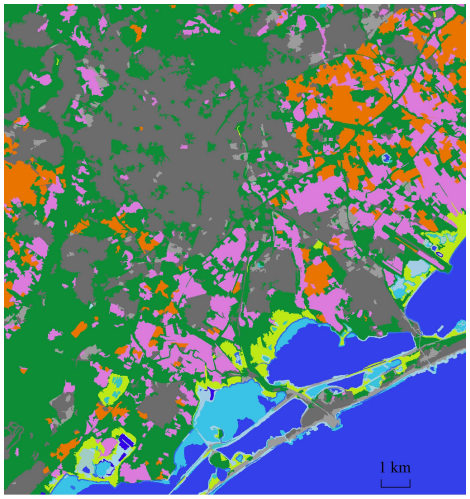

Figure 6.6: Clustering results for the proposed pipeline without NDVI branch. The clusters are colored according to the clustering map (b) in Figure 6.5. 6 clusters of water surface are colored in different shades of blue.

As it can be seen, the variation of our model without NDVI branch gives the NMI score similar to our method, but after the visual examination of the results (see Figure 6.6), we can state that our method with the NDVI branch gives better clustering results of vegetation areas. For example, for the number of clusters higher than 8, it distinguishes different types of annual crops that do not figure in GT maps,

but can be easily spotted on the SITSs. Furthermore, when increasing the number of clusters, our model with the NDVI branch gives the results that are more intuitive to interpret as the model iteratively divides each cluster into two clusters of more or less proportional sizes. At the other hand, the model without the NDVI branch produces many small clusters: for example, for 14 clusters segmentation results, we obtain 6 clusters of size less that 1000 pixels each that correspond to some minor variations in water bodies. At the same time, some vegetation types are not detected at all: classes corresponding to prairies and orchards are missing.

For the concurrent approaches, we can spot that enriching the original images with NDVI band has only slightly increased the accuracy of the DTW methods, though no significant improvements were done.

Figure 6.7 presents the classification results for the concurrent approaches for the best number of clusters with the added NDVI band. We notice that clusters that correspond to the residential area and the forest are often mixed. It can be explained by the fact that no spatial features were extracted and the mean segment value can not always discriminate these classes. Moreover, the prairies and orchards clusters are mixed with other vegetation clusters, that is probably related to the imperfection of the segmentation. It made it more difficult to associate the obtained results to the real-life classes than for our method as the "computer clustering logic" is less obvious.

We equally notice that the graph-based method does not ensure the whole coverage of the study area due to the specificity of the algorithm (non-covered areas are shown in white) that leads to some missing information. Moreover, this method gives the most uneven shapes of clusters, especially, in the city area that might complicate the interpretation of the results.

We believe that the chosen area is a relatively complex landscape because it contains many different land cover types, so potentially it should work for any similar area or areas with less classes (including areas that are predominantly agricultural, as about half of our study area corresponds to vegetation). However, for a larger number of clusters, the readers should understand that no ideal unsupervised clustering algorithm exists. When the number of ground truth classes is much higher, we usually have to deal with look-alike classes. For example, two classes have the same growing cycle, but slightly different textures. These classes will be difficult to distinguish when no supervised constraint is applied.

Furthermore, besides the weakness that we mentioned for the clustering of linear objects, and the cases of a large number of lookalike land cover classes, we did not detect any limitations specific to our algorithm. Although, we insists that our

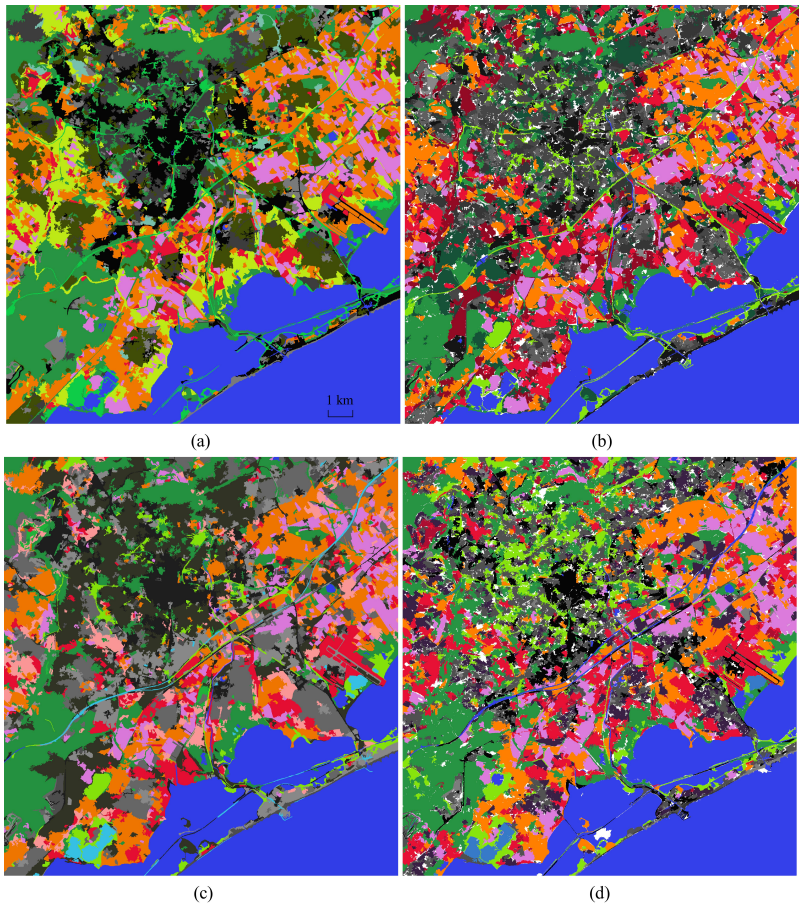

Figure 6.7: Clustering results for the concurrent approaches. (a)- Object-based DTW for the SPOT-5 dataset, (b)- Graph-based DTW for the SPOT-5 dataset, (c)- Object-based DTW for the Sentinel-2 dataset, (d)- Graph-based DTW for the Sentinel-2 dataset. For the legend, please, refer to Figure 6.5. The clusters are colored accordingly to the GT maps, if one class is presented by several clusters, these clusters are colored in different shades of the referent class color. For graph-based DTW white pixels correspond to the areas that are not covered by graphs.

algorithm still possesses all the limitations common to unsupervised neural networks and clustering. For example, for a dataset with unbalanced classes, it may be difficult to extract the features for small size classes; the final model varies from one experiment to another; setting the right parameters for the network may sometimes be complex and depends on the data. Moreover, our clustering method as well as many others is not able to detect the outliers that can be seen as non-trivial changes in the context of this thesis. If one is aware that the study area contains various changes in land cover types, we recommend to perform a non-trivial change detection analysis presented in the previous chapter and interpret it together with SITS clustering results. Finally, as for any unsupervised methods, the obtained clusters do not always correspond to the expected classes, as the computer's pattern analysis is different from the human perception.

6.6 Discussion

In this chapter, we have presented a fully unsupervised approach for SITS clustering based on a two branch multi-view 3D convolutional AE that does not demand any labeled data. The proposed approach exploits the AE model to compress a time series into an encoded image by extracting its spatio-temporal features. Then it performs the segmentation of the encoded image with the eventual correction of shifted segment borders related to the specificity of the encoding. The proposed approach was tested on two real-life datasets and showed its efficiency comparatively to the concurrent approaches.

The main advantages of the proposed algorithm include the improvement of traditional segmentation methods that are not initially adapted for the SITS that leads to higher NMI score. In addition, we have shown that we can improve clustering results by simply introducing a temporal NDVI branch in the AE model. The presented approach is a good alternative to traditional DTW-based methods as deep learning techniques are able to extract more robust and complex features compared with traditional Machine Learning methods.

However, the user should be aware that if a SITS contains multiple heterogeneous change processes, he/she should equally perform non trivial change detection analysis and interpret it together with the results of the presented 3D SITS clustering.

Chapter 7

Conclusion

Contents

In this thesis, we have proposed a set of algorithms for unsupervised SITS analysis using deep learning techniques. Below we summarize the contributions of this work and propose some directions for future research.

7.1 Thesis Contributions

In this thesis, we proposed an ensemble of approaches for fully unsupervised HR SITS analysis that included SITS clustering and bi- and multi-temporal change detection and analysis. We distinguished three types of temporal behavior: no change areas such as cities, seasonal or trivial changes (mostly presented by the vegetation) and the non-trivial changes that are presented by some permanent changes, i.e. changes that do not contain seasonal trends, for example, new constructions, permanent crop rotations, or vegetation that does not follow overall tendency, etc. The contributions of this thesis can be divided in three different parts.

In the first part (Chapter 4), we proposed an unsupervised algorithm for non-trivial bi-temporal change detection in satellite images. We exploited a joint neural network autoencoder (AE) model for feature translation to detect changes between two images. Contrarily to traditional change detection algorithms that detect “everything

that changes", our proposed algorithm detects only non-trivial changes as the most interesting.

In the second part (Chapter 5), we developed an algorithm that interprets the detected bi-temporal changes in the multi-temporal context. We proposed a method to model spatio-temporal change phenomena in form of evolution graphs. Then, to cluster these change graphs, we developed a clustering model based on feature extraction with a GRU autoencoder that is able to proceed varying length graphs. Moreover, the proposed framework for SITS change detection and clustering does not depend on the temporal resolution and does not demand any archive images to detect different seasonal trends.

In the third part (Chapter 6), we introduced an object-based SITS clustering approach that exploits a 3D convolutional AE for spatio-temporal feature extraction to regroup no change areas and seasonal changes. We proposed a two-branch multi-view AE that extracts more robust spatio-temporal features comparatively to a classic 3D convolutional AE. Moreover, we developed a segmentation approach that produces a unique segmentation map for the whole SITS. We performed a complete object-based SITS clustering exploiting the spatio-temporal properties of the detected segments.

7.2 Short Term Perspectives

The methods presented in this thesis have showed their effectiveness on experimental datasets. However, due to the absence of reference data, unsupervised methods do not always give us the results we expect. Some techniques can be integrated in our SITS analysis algorithms to improve their quality. In this section, we propose to summarize some of the work perspectives for this research.

Bi-temporal change detection: Our method exploits patch-wise change detection, hence, no analysis at object level is performed. For this reason, the algorithm might miss some changes within a closed change area or, on the contrary, detect some false positive change pixels in an unchanged object. This can be mostly seen in crops, as the growing vegetation is often non-homogeneous within the limits of an agricultural parcel. The accuracy of bi-temporal change detection algorithm can be improved by introducing a morphological analysis or by refining the detected changes with object-based approaches, e.g. if only 10% of pixels of a detected object are marked as changes, we can consider these pixels as FP changes.

Despite the fact that some of the false positive changes are eliminated during multi-temporal change modeling with graph-based methods, introducing object and shape analysis at this step, can improve the overall performance of the proposed framework, both for bi- and multi-temporal change detection algorithms.

Multi-temporal change interpretation and clustering: Our multi-temporal change detection algorithm is based on the interpretation of non-trivial bi-temporal changes in the multi-temporal context. The bi-temporal change detection method aims to detect major seasonal trends presented in each couple of images, hence, during the multi-temporal change interpretation, only the major seasonal trends are detected. For this reason, some minor season trends are interpreted as non-trivial changes in the multi-temporal context and more deep analysis is needed to isolate these changes. In the future, some methods to detect minor seasonal vegetation trends can be elaborated.

More robust change graph construction and interpretation methods can be proposed. The graphs constructed with our method make the global description of the detected change process without considering that it may potentially contain several sub-processes.

Finally, a study to estimate an optimal number of clusters to regroup the detected changes could result in facilitating clustering interpretation. However, finding the optimal number of clusters is not specific to our method and is considered as one of the most important unsolved problems of unsupervised algorithms.

SITS clustering: In the future works, our clustering algorithm can be adapted to all spectral bands of Sentinel-2 images. The framework might be improved to better distinguish linear objects, e.g. we can combine enrich feature descriptors with some raw image values to overcome the border effect problem. Other spectral indices can be also integrated in the proposed model, however, a closer study is needed to estimate the influence of each index on the clustering results. Moreover, contextual constraints may be introduced to distinguish more classes (e.g., artificial areas such as beaches can be discriminated from urban areas as they are close to the water).

7.3 Long Term Perspectives and Limitations

The most interesting long term perspectives for this thesis would be to integrate different constraints or even some labeled data in our algorithms.

As it was mentioned multiple times before, unsupervised algorithms, especially clustering, rarely give us the desired output.

The most negative factors that influence the accuracy of change detection algorithms are the changes in image luminosity and the presence of defected pixels. While some mild changes in luminosity are neglected by our change detection algorithms, surfaces with high brilliance coefficient (roofs or some types of bare soil) are often detected as changes. Unfortunately, for a general unsupervised change detection approach, it seems impossible to model all luminosity changes, even with the best algorithm.

For clustering algorithms, the biggest challenge still is to define the optimal number of clusters and associating the obtained partitions to the desired real-life classes. Most of the time, two or more clusters are associated to one real-life class or on the contrary, one cluster contains two or more real-life classes. To solve this problem, one-shot learning algorithms can be exploited [168]. This type of algorithms presumes that for each researched class we have only one labeled sample which should give its best and the most generalized description. One-shot learning techniques may be successfully applied to our 3D SITS clustering, for example, to finetune a pretrained unsupervised feature extraction model in order to obtain a supervised classification model.

At the same time, for the clustering of detected multi-temporal changes it might be a complicated task to guess all the types of changes present in a SITS and even more complicated to find the best samples for the one-shot learning algorithm. However, we believe that integration of some constraints may improve the results of unsupervised change detection and clustering. For example, we can use different spectral indices to roughly estimate types of some areas or/and even take into account the acquisition dates of the images to make some assumptions of seasonal trends.

Moreover, the results of unsupervised algorithms are rarely stable. For our algorithms, it was established that change detection results do not have important variations between several runs, but at the same time, feature extraction and corresponding clustering results may give different results at each run. To obtain more stable results, we may integrate one shot-learning algorithm in our frameworks or exploit some novel methods which could be used as a base for further researches not only for SITS analysis, but for unsupervised learning in general.

Appendices

Appendix A

Datasets

In this work, several different image datasets were used to lead the experiments and evaluate the results. All the datasets correspond to high resolution satellite time series with 10 meters spatial resolution.

A.1 The SPOT-5 Montpellier dataset

SPOT-5 dataset was taken over the Montpellier area, France between 2002 and 2008 and belongs to the archive Spot World Heritage[1]. This particular dataset was chosen due to its high ratio of agricultural lands and progressive construction of new areas. We have filtered out the cloudy and flawed images from all the images acquired by SPOT-5 mission over the considered geographic area and obtained 12 exploitable images with irregular temporal resolution (minimum temporal distance between two consecutive images is 2 months, maximum - 14 months, average - 6 months). Distribution of dataset images is presented in Table A.1.

All SPOT-5 images provide green, red, NIR and SWIR bands with 10 meters resolution. The pre-processing level of the dataset is 1C (orthorectified images, reflectance at the top of atmosphere). For this reason, the SITS was radiometrically normalized with the aim to obtain homogeneous and comparable spectral values over each dataset. For the image normalization, we have used an algorithm introduced in [123] that is based on histogram analysis of pixel distributions. The original images are clipped to rectangular shapes of 1600×1700 pixels and transformed to UTM zone 31N: EPSG Projection. The clipped image extent corresponds respectively to the following latitude and longitude in WGS-84 system:

[1]Available on `https://theia.cnes.fr/`

– bottom left corner: $43°30'6.0444''N$, $3°47'30.066''E$
– top right corner: $43°39'22.4856''N$, $3°59'31.596''E$

Table A.1: Image acquisition dates for the Montpellier dataset.

Acquisition date, yyyy-mm-dd			
1	2002-10-05	7	2006-02-18
2	2003-09-18	8	2006-06-03
3	2004-05-14	9	2007-02-01
4	2004-08-22	10	2007-04-06
5	2005-04-27	11	2008-06-21
6	2005-12-01	12	2008-08-21

A.2 The Sentinel-2 Montpellier dataset

Sentinel-2 dataset was taken between January 2017 and December 2018[2]. After deleting unexploitable images as well as the images that were less than 15 days apart from previous images, we have obtained 24 images with more regular temporal resolution (minimum temporal distance between two consecutive images is 15 days, maximum - 2.5 months, average - 1 month). Distribution of dataset images is presented in Table A.2.

Table A.2: Image acquisition dates for the Sentinel-2 dataset.

Acquisition date, yyyy-mm-dd							
1	2017-01-03	7	2017-08-20	13	2017-12-19	19	2018-07-17
2	2017-03-14	8	2017-09-20	14	2018-01-23	20	2018-08-06
3	2017-04-03	9	2017-10-10	15	2018-02-12	21	2018-08-26
4	2017-04-23	10	2017-10-30	16	2018-02-27	22	2018-09-20
5	2017-06-12	11	2017-11-14	17	2018-04-18	23	2018-10-05
6	2017-07-12	12	2017-11-29	18	2018-06-27	24	2018-12-29

Sentinel-2 images provide multiple spectral bands of different spectrum and spatial resolution, however, it was decided to keep only 10-meters resolution spectral bands - blue, green, red, NIR. The pre-processing level of the dataset is 1C (orthorectified

[2]Available on `https://earthexplorer.usgs.gov/`

images, reflectance at the top of atmosphere). For this reason, the SITS was radiometrically normalized with the aim to obtain homogeneous and comparable spectral values over each dataset. For the image normalization, we have used an algorithm introduced in [123] that is based on histogram analysis of pixel distributions.

The original images are clipped to rectangular shapes of 1600×1700 pixels and transformed to UTM zone 31N: EPSG Projection. The clipped image extent corresponds respectively to the following latitude and longitude in WGS-84 system:

- bottom left corner: $43°30'6.0444''N$, $3°47'30.066''E$
- top right corner: $43°39'22.4856''N$, $3°59'31.596''E$

A.3 The Sentinel-2 Rostov-on-Don dataset

Sentinel-2 dataset was taken over the city of Rostov-on-Don[3] (the dataset called later Rostov), Russia between 2015 and 2018. This dataset was chosen as the city of Rostov-on-Don underwent the constructions coincided with FIFA world cup 2018 and equally because it has waste agricultural areas. We have filtered out cloudy, snow-covered and flawed images from 2015 (beginning of the Sentinel-2 mission) till the end of 2018 that gave us 26 images. We deleted 6 more images from the dataset as the temporal distance between two consecutive images did not exceed 15 days which resulted in 20 imagess. Distribution of dataset images is presented in Table A.3.

However, for the multi-temporal change detection, we have deleted the first three images of the dataset because of the irregular temporal distance between them (distance between 1st and 2nd image is around one month, 2nd and 3rd - 4 months, 3rd and 4th - 5 months). Finally, we for the multi-temporal change detection, we have exploited 17 images taken from July 2016 to November 2018 with relatively regular temporal resolution (images 4-20 in Table A.3). The average gap between two consecutive images does not exceed 1.5 months, minimum - 20 days, maximum - 5 months (corresponds to winter with deleted snow-covered images).

Sentinel-2 provides multi-spectral images with different spatial resolution spectral bands. It was decided to keep only 10 meter resolution green, red and NIR bands. The blue band was not used as the information that it contains is little relevant for change detection. The pre-processing level of the dataset is 1C (orthorectified images, reflectance at the top of atmosphere). For this reason, the SITS was radiometrically normalized with the aim to obtain homogeneous and comparable spectral values over

[3]Available on www.earthexplorer.usgs.gov

Table A.3: Image acquisition dates for the Rostov dataset.

Acquisition date, yyyy-mm-dd			
1	2015-08-30	11	2017-09-18
2	2015-09-19	12	2018-01-11
3	2016-02-16	13	2018-04-11
4	2016-07-15	14	2018-05-01
5	2016-08-04	15	2018-05-31
6	2016-09-13	16	2018-07-10
7	2016-11-22	17	2018-08-19
8	2017-05-01	18	2018-09-13
9	2017-06-30	19	2018-10-13
10	2017-08-09	20	2018-11-02

each dataset. For the image normalization, we have used an algorithm introduced in [123] that is based on histogram analysis of pixel distributions.

The images were provided in WGS84/UTM zone 37N projection and clipped to 2200×2400 pixels areas. The clipped image extent corresponds respectively to the following latitude and longitude in WGS-84 system:

– bottom left corner: $47°11'3.5664''N$, $39°34'3.2628''E$
– top right corner: $47°24'7.3332''N$, $39°51'41.328''E$

Appendix B

Unsupervised Indices

Unsupervised evaluation criteria [169] are based on internal information from both the data and the clusters. For instance, several of them are based on the distance between the data and the cluster centroids. These indices have been introduced based on the simplest principles of what defines a cluster:

1. Objects from a given cluster are supposed to be as close as possible from each other.
2. Objects belonging to different clusters should be as far as possible.

To assess these intuitive criteria, most indices adopt a strategy that consists in measuring the distance between each data elements and some object representing the clusters (centroids, representative data elements, etc.). By doing so, it is quite straightforward to evaluate the compactness and separability of the clusters.

However, with no clear definition of what a "good cluster" is, each unsupervised index has its own way of computing the compactness and separability of the clusters and to use these two values to compute a final quality criterion.

Some of these criteria can be used as objective functions. The goal of a clustering algorithm would then be to find a solution that maximizes the said objective function. Some criteria however are too costly to be used in an objective function and are usually only computed when the clustering process is done.

Mean Squared Error (MSE) is one of the easiest way to evaluate the quality of a result for clustering algorithms that use centroids. Given a clustering solution S

with K clusters, it can be computed as shown:

$$MSE_{cl} = \frac{1}{\sum_{i=1}^{K} |c_i|} \sum_{k=1}^{K} \sum_{x \in c_k} d(x - \mu_k)^2 \tag{B.1}$$

where $d(\cdot)$ is a distance function, $|c_i|$ is the number of elements linked to the cluster c_i and μ_k the centroid of a cluster c_k. For a clustering result to be considered good, the Mean Squared Error must be as low as possible.

Dunn Index (DI) [170] is another internal criterion defined as in Equation (B.2) where $D(c_i, c_j)$ is a distance metric between two clusters c_i and c_j, and Δ_i is a measure of scatter for a cluster c_i. Any quality index using such a kind of ratio is called a "Dunn-like index".

A higher Dunn index indicates a better clustering.

$$DU = \frac{\min_{i \neq j} D(c_i, c_j)}{\max_{i \in [1..K]} \Delta_i} \tag{B.2}$$

One particularity of the Dunn Index is that the distances $D(c_i, c_j)$ and Δ_i can be defined in many different ways:

- $D(c_i, c_j)$ can be the smallest distance between two objects belonging to c_i and c_j respectively, see Equation (2.2). In this case Δ_i must be the largest distance between two objects belonging to a cluster c_i:

$$\Delta_i = \max_{x,y \in c_i} d(x, y) \tag{B.3}$$

- $D(c_i, c_j)$ can be the largest distance between two objects belonging to c_i and c_j respectively, see Equation (2.3). This is quite uncommon and give rather poor results. In this case Δ_i must be the smallest distance between two objects belonging to a cluster c_i:

$$\Delta_i = \min_{x,y \in c_i} d(x, y) \tag{B.4}$$

- $D(c_i, c_j)$ can also be the distance between the centroids of c_i and c_j, see Equation (2.5). Then Δ_i usually is the largest distance between an object belonging to c_i and its centroid μ_i:

$$\Delta_i = \frac{1}{|c_i|} \sum_{x \in c_i} (x - \mu_i) \tag{B.5}$$

– Finally, $D(c_i, c_j)$ can be the average distance between the data belonging to c_i and c_j respectively, see Equation (2.4). Then Δ_i usually is the mean distance between all pairs of a cluster c_i:

$$\Delta_i = \frac{1}{|c_i| \cdot (|c_i| - 1)} \sum_{\substack{x,y \in c_i \\ x \neq y}} d(x, y) \tag{B.6}$$

Davies-Bouldin Index (DB) [47] is a possible alternative to the Dunn Index. It assesses whether the clusters are compact and well separated. It is based on the possible measure of separation $D(c_i, c_j)$ and measure of scatter Δ_i (B.5).

Given a clustering solution that contains K clusters, the Davies-Bouldin Index is defined as shown in Equation (B.7).

$$DB = \frac{1}{K} \sum_{i=1}^{K} \max_{j \neq i} \frac{\Delta_i + \Delta_j}{D(c_i, c_j)} \tag{B.7}$$

The Davies-Bouldin index is not normalized and a lower value indicates a better quality.

The most commonly used measure of separation and measure of scatter for the Davies-Bouldin Index -as introduced in the original article [47]- are based on the cluster mean values using Equations (2.5) and (B.5).

Silhouette Index (SC) [32] is yet another internal criterion that assesses the compactness of the clusters and whether or not they are well separated. The main difference between the Silhouette index and the Dunn index or the Davies-Bouldin index is the following: the Silhouette Index can be computed for a given object x, a given cluster c_i, or for the whole clustering C.

For a data element, the Silhouette index is defined as shown in Equation (B.8) where a_x is the mean distance between the observed object x and all other objects that belong to the same cluster that x, and b_x is the mean distance between x and all other objects that are not in the same cluster that x.

$$SC(x) = \frac{b_x - a_x}{max(a_x, b_x)} \tag{B.8}$$

The Silhouette index takes values between -1 and 1. A positive value ($a_x < b_x$) means that x is closer from the objects belonging to its clusters than from the objects belonging to other clusters. Therefore a positive value close to 1 means that x is

probably in the right cluster, while a negative one means that x should be in another cluster.

The Silhouette index for a given cluster c_i is the mean value of the Silhouette index computed on all the objects of this cluster:

$$SC(c_i) = \frac{1}{|c_i|} \sum_{x \in c_i} SC(x) \tag{B.9}$$

A Silhouette index that is positive and close to 1 means that the observed cluster is both compact and well separated from the other clusters.

Finally, the Silhouette index can be computed on the whole partition as shown in Equation (B.10). Once again, 1 is the best value meaning that the clusters are very compact and well separated and -1 is the worst value. Usually, the Silhouette Index must be positive for a clustering result to be considered acceptable.

$$SC(C) = \frac{1}{K} \sum_{i=1}^{K} SC(c_i) \tag{B.10}$$

Before considering to use the Silhouette index, one must consider the high computational cost of this index with large data sets: since it is not based on the mean vectors, the Silhouette index requires to compute several time the pairwise distances between all the data.

Wemmert-Gançarski Index (WG) [171] is another elegant way to assess the quality of a clustering result based on the compactness and separability of the clusters. For a cluster c_i, it is computed as follows with $j = \text{argmin}_{k \neq i}\, d(x, \mu_k)$:

$$WG(c_i) = \begin{cases} 0 \text{ if } \frac{1}{|c_i|} \sum_{x \in c_i} \frac{d(x,\mu_i)}{d(x,\mu_j)} > 1 \\ 1 - \frac{1}{|c_i|} \sum_{x \in c_i} \frac{d(x,\mu_i)}{d(x,\mu_j)} \text{ otherwise} \end{cases} \tag{B.11}$$

The Wemmert-Gançarski index takes its values between 0 and 1, 1 meaning that the clusters are very compact and well separated. When applied to a complete clustering result, it is defined as follows:

$$WG(C) = \frac{1}{\sum_{k=1}^{K} |c_k|} \sum_{i=1}^{K} |c_i| WG(c_i) \tag{B.12}$$

Appendix C

Feature Analysis in Remote Sensing

In this thesis, we have presented some feature extraction methods based on unsupervised neural networks. Below, we present some traditional feature and spectral index extraction methods.

C.1 Haralick texture features

Haralick textural features are based on Gray Level Co-occurrence Matrix (GLCM). GLCM is a square matrix that represents the distribution of co-occurring pixel values (grayscale values, or colors) at a given offset. Let I be an images of size $H \times W$ with N_g distinct intensity values, then C is a GLCM square matrix of size $N_g \times N_g$ that is computed for a chosen distance d and angle θ. The GLCM matrix is usually symmetrical and it can be computed for 4 different directions of θ which are $0°, 45°, 90°, 135°$. Each element $C(i, j)$ represents the number of co-occurrences of couples of pixels with values i and j with distance d at angle θ between them:

$$C_{d,\theta}(i,j) = \sum_{x=1}^{W} \sum_{y=1}^{H} \begin{cases} 1, & \text{if } I(x,y) = i \text{ and } I(x+\Delta x, x+\Delta y) = j. \\ 0, & \text{otherwise.} \end{cases} \tag{C.1}$$

where Δx and Δy are the offsets for distance d at angle θ. Figure C.1 presents the values of Δx and Δy (in braces) for 4 different spatial directions and $d = 1$ which is the most common value.

GLCM matrices are often built for 4 available directions, then each matrix is

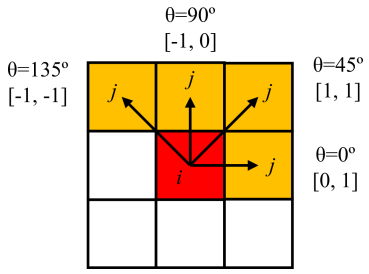

Figure C.1: 4 types of spatial relations between two pixels x (red) and j (yellow).

normalized by the total number of co-occurences in the image. Finally, the average GLCM matrix is computed and used for the extraction of different textural features. We present some of these features below. The exhaustive list is presented in the original article [82].

Angular second moment - measures the uniformity (or orderliness) of the gray level distribution of the image:

$$ASM = \sum_i \sum_j p(i,j)^2 \tag{C.2}$$

where $p(i,j)$ is a i,j entry in a normalized gray-tone spatial dependence matrix.

Contrast - represents the amount of local gray level variation in an image:

$$Contrast = \sum_{n=0}^{N_{g-1}} n^2 \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i,j),\ |i-j| = n \tag{C.3}$$

Homogeneity (inverse difference moment) - measures the smoothness of the gray level distribution of the image, it is usually inversely correlated with contrast:

$$Homogeneity = \sum_i \sum_j \frac{1}{1+(i-j)^2} p(i,j) \tag{C.4}$$

Entropy - measures the degree of disorder among pixels in the image:

$$Entropy = -\sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i,j) \log[p(i,j)] \tag{C.5}$$

C.2 Spectral indices

Different spectral indices are exploited in remote sensing applications. Spectral indices are combinations of reflectance values from two or more spectral bands that indicate

the relative abundance of the features of interest. The most popular indices are the ones for vegetation, but many other indices are also available.

Here we present a non-excessive list of some of spectral indices:

Normalized Difference Vegetation Index (NDVI) [83]:

$$NDVI = \frac{NIR - Red}{NIR + Red} \tag{C.6}$$

Normalized Difference Water Index (NDWI) [172]:

$$NDWI = \frac{Green - NIR}{Green + NIR} \tag{C.7}$$

Visible and Shortwave Infrared Drought Index (VSDI) [173]:

$$VSDI = 1 - [(SWIR - Blue) + (Red - Blue)] \tag{C.8}$$

Normalized Difference Snow Index (NDSI):

$$NDWI = \frac{Green - SWIR}{Green + SWIR} \tag{C.9}$$

Normalized Burned Ratio Index (NBRI) [174]:

$$NBRI = \frac{NIR - SWIR}{NIR + SWIR} \tag{C.10}$$

Soil Adjusted Vegetation Index (SAVI) [175]:

$$SAVI = \frac{(NIR - Red)(1 + L)}{NIR + Red + L} \tag{C.11}$$

where L is a canopy background adjustment factor. An L value of 0.5 in reflectance space was found to minimize soil brightness variations and eliminate the need for additional calibration for different soils.

These indices are successfully used for different remote sensing applications. While for some algorithms the researchers exploit only the analysis of an extracted index [135, 140, 155, 176], for others, the indices are combined with original image bands or some other data [130, 142, 177].

Appendix D

Résumé en Français

D.1 Contexte

Cette thèse portant sur l'analyse non-supervisé des séries temporelles d'images satellites (STIS) a été dirigée par Maria Trocan (ISEP) et co-encadrée par Jérémie Sublime (ISEP, Université Paris 13).

L'objectif principal de cette thèse est de développer un ensemble d'algorithmes non-supervisés pour l'analyse générique de STIS. Ces algorithmes sont adaptés aux séries d'images open source avec une haute résolution spatiale, comme Sentinel-2 et SPOT-5. Comme les images satellites sont très hétérogènes, la création d'une base de référence avec des données labellisées est impossible pour un algorithme générique comme le nôtre. C'est pour cette raison que les algorithmes proposés ne demandent aucunes données labellisées, de plus, ils n'exigent pas que la série ait une résolution temporelle régulière (images acquises avec à peu près le même intervalle temporel) ou contienne un grande nombre d'images (> 25) ce que rend nos approches encore plus génériques.

Nos algorithmes exploitent les méthodes de *machine learning* et, notamment, les réseaux de neurones afin de détecter les différentes entités spatio-temporelles et leurs changements éventuels dans le temps. Dans cette thèse, nous distinguons trois différents types de comportement temporel que nous visons à identifier :

- les zones sans changements - surtout représentées par les zones urbaines;
- les changements saisonniers (autrement dit, les changements triviaux) - la végétation et les autres phénomènes ayant la récurrence saisonnière;
- les changements non triviaux - les changement permanents comme les construc-

tions ou les démolitions, la rotation des cultures agricoles et la végétation qui ne suit pas la tendance saisonnière pressente dans la zone d'études.

Pour identifier ces comportements temporels dans STIS, nous proposons deux frameworks : un pour la détection et le clustering des changements non-triviaux et un autre pour le clustering des zones "stables" de la série (changements saisonnières et les zones sans changements). Ces frameworks exploitent des approches suivantes :

– Nous proposons une approche pour la détection et le clustering des changements multi-temporels. Comme les STIS peuvent être courtes ou avoir une résolution temporelle irrégulière, on ne peut pas établir les tendances saisonnières présentes dans la série afin de trouver les comportements temporels inhabituels. Pour résoudre ce problème, nous décomposons la tâche de détection des changements multi-temporels en deux étapes :

  1. *La détection des changements bi-temporels* pour chaque couple d'images consécutives de la série. Cette méthode est basée sur les autoencodeurs (AE) joints pour la transformation des features entre deux images. Contrairement aux approches existantes, notre algorithme détecte que les changements non-triviaux et ignore les tendances saisonnières.

  2. *L'interprétation des changements bi-temporels dans le contexte multi-temporel* à l'aide des approches basées sur les graphes. Nous proposons une nouvelle méthode qui permet de modéliser les changements qui durent plusieurs étapes. Les changements sont d'abord présentés sous forme des graphes d'évolution. Par la suite, nous faisons le clustering de ces graphes pour identifier les différents types de changements dans le temps. Nous proposons d'utiliser l'autoencodeur basé sur les réseaux de neurones récurrents (*recurrent neural networks* ou RNN) au lieu de la méthode classique basée sur *dynamic time warping* (DTW).

– Nous développons un algorithme basé objets pour le clustering de STIS pour analyser le reste de la série afin de séparer les zones sans changements et les changements saisonniers dans des clusters différents. Premièrement, nous compressons la série en une image encodée avec l'AE convolutif 3D. La première branche de cet AE extrait des *features* spatio-temporelles profondes des images originelles, alors que la deuxième branche extrait des *features* spatio-temporelles générales des images d'indice de végétation normalisé (NDVI) correspondantes. Dans un deuxième temps, nous faisons la segmentation de toute la série qui

se passe en deux étapes (la segmentation préliminaire et sa correction) et qui est réalisée en utilisant à la fois les images initiales et l'image encodée. Cette segmentation est unique pour toute la série et contient tous les objets présents sur différentes images de la série. Finalement, on fait le clustering des segments obtenus en utilisant leurs descripteurs encodés. Nous montrons que le rajout de la branche NDVI améliore significativement les résultats de clustering, surtout pour les zones végétales.

D.2 Résumé de la Thèse

D.2.1 Apprentissage Automatique

L'apprentissage automatique (ou *machine learning* en anglais) est un domaine de recherche lié à l'informatique et aux mathématiques appliquées, dont l'objectif est de permettre à une machine ou à un programme d'apprendre sans être explicitement programmé pour cela. C'est donc une science qui consiste à construire des algorithmes et des méthodes capables d'apprendre à partir de données avec divers objectifs tels que pouvoir classer les données, identifier des structures dans les données, ou encore faire des prédictions. L'apprentissage automatique est utilisé dans de nombreux domaines scientifiques tels que la biologie et la médecine, les mathématiques, la finance et le marketing, la physique, la chimie, et bien d'autres. On distingue généralement trois types d'applications pour l'apprentissage automatique :

- L'apprentissage supervisé, où la machine apprend à partir d'exemples labellisés dans le but de construire un modèle permettant de faire des prédictions ou de classer des données non-labellisées.

- L'apprentissage non-supervisé, où les données sont fournies sans étiquettes. La machine cherche alors à détecter des structures ou à faire des groupes d'éléments similaires.

- L'apprentissage par renforcement, où une machine placée dans un environnement dynamique va apprendre à effectuer une tâche donnée grâce à un système de "récompenses et pénalités" basé sur ses actions.

Cette thèse s'intéresse à l'apprentissage non-supervisé et plus particulièrement au clustering et à la détection de changements (des anomalies).

Le clustering est une tâche d'apprentissage consistant à chercher à faire des groupes d'objets *similaires* à partir de données sans étiquettes. Un cluster se définit donc comme un groupe d'objets similaires au sein d'un jeu de données.

La détection des anomalies est un processus de l'identification d'éléments, d'événements ou d'observations rares qui différent de la norme. La détection d'anomalies est formulée comme le problème de classification binaire ou la première classe correspond aux entités normales et la deuxième présente celles avec un comportement anormal. Pour la détection des anomalies, on doit trouver un modèle qui décrit le jeu de données étudié. Ensuite, on compare les entités de notre jeu de données à ce modèle pour estimer le niveau de leur déviation de la norme.

Pour mieux comprendre et représenter la structure des données complexes afin de réaliser le clustering ou la détection d'anomalies, il est indispensable de réaliser l'extraction des caractéristiques (plus souvent appelé *feature extraction*).

Extraction de caractéristiques est une branche d'apprentissage non-supervisé qui a pour le but de créer un nouvel ensemble de variables (*features*) à partir des variables initiales. En imagerie, l'extraction de caractéristiques peut être réalisée au niveau du pixel, de l'objet ou bien du voisinage de pixel (*patch*). Dans cette thèse, nous faisons l'analyse au niveau des patchs, donc pour chaque pixel de l'image ou de la séquence d'images on extrait un patch de la taille $p \times p$ qui sera par la suite compressé en *feature vector*. Nous faisons l'extraction de *features* avec les réseaux de neurones non-supervisés profonds - les autoencodeurs.

Les réseaux de neurones profonds sont composés de plusieurs couches et font l'extraction de *features* à plusieurs niveaux. De plus, les réseaux de neurones peuvent analyser différents types de données : des données vectorielles aux séquences d'images multi-spectrales ou encore de des données vidéo. Dans l'imagerie classique, les réseaux de neurones se composent habituellement de deux types de couches : couches convolutives pour l'extraction de l'ensemble de *features* spatiales (*feature maps*) et couches linéaires pour la compression de ces textures en *feature vector*. En outre, pour analyser les séquences temporelles, dans cette thèse, nous utilisons des modèles de réseaux de neurones récurrents (*recurrent neural networks* (RNN)). Ces modèles sont construites dans la manières suivante : pour une séquence $X = \{x_1, x_2, ..., x_n, ..., x_S\}$ à chaque estampilles temporelle le réseau calcule l'état caché h_n qui représente l'état accumulé du réseau à temps T_n (Figure 3.9). Le dernier état caché h_S est par la suite utilisé pour caractériser le réseau.

Parmi différents réseaux de neurones, **les autoencodeurs** ont trouvé des applications dans des nombreux domaines. En traitement d'images, les AEs sont largement

utilisés pour la segmentation [2, 89], la compression [90], la reconstruction d'images [91], l'extraction de *textures* [92, 93] et le clustering [79, 94]

Malgré la variété des modèles présentés, seulement ceux ayant les mêmes données d'entrée et de sortie sont considérés comme étant des AEs traditionnels, comme ceux pour l'extraction de *features* et le clustering dans la Figure D.1.

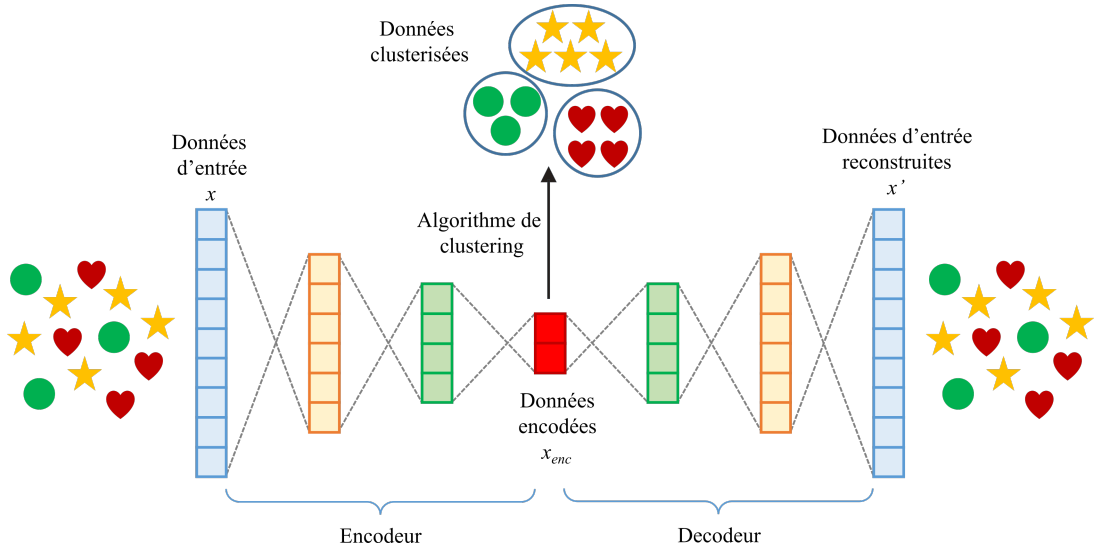

Figure D.1: Exemple d'un AE classique avec le clustering appliqué aux données encodées.

La partie encodeur vise à trouver une représentation latente des données d'entrée, alors que la partie de décodeur doit la reconstruire des données qui ressemblent le plus possible aux données d'entrée. L'optimisation du modèle est réalisé de telle manière que la différence entre la sortie de l'AE et les données d'entrée soit minimale, mais en même temps le modèle doit être le plus générique possible pour éviter le problème d'*overfitting*. Quand le modèle est stable, les données encodées sont utilisées pour le clustering (Figure D.1) ou un autre type d'analyse.

D.3 Détection de Changements Bi-temporels

Le détection de changements est un sujet de recherche d'actualité. Comme les changements peuvent être assez diversifiés il est difficile de créer une basé de données de référence, surtout si le type de changements recherché n'est pas connu. Pour cela, des nombreuses approches non-supervisées pour la détection de changements ont été proposées.

Les premiers algorithmes exploitaient l'analyse d'image de différence qui est calculée comme la différence absolue de valeurs de pixels de deux images . Cette

image est par la suite clusterisée en utilisant les différentes méthodes qui permettent d'isoler les zones de changements[110–112]. Des modèles basés sur les champs de Markov ont été proposés pour affiner les résultats de détection de changement avec l'analyse de voisinage [113, 114]. Malheureusement, ces approches donnent les résultats bruités à cause de problèmes de luminosité et de décalage de pixels. Pour cela des méthodes automatique pour sélectionner les pixels changés/pas changés ont été exploitées [115, 116] pour utiliser les pixels sélectionnés pour l'entraînement des modèles supervisés.

Cependant, la relation entre les bandes spectrales de deux images est plus complexe et les méthodes classiques ne peuvent pas la modéliser correctement. Les modèles d'apprentissage profond ont montré une meilleure performance, parce qu'elles peuvent capter les dépendances non-linéaires [120]. De plus, les approches classique ne différencient pas les changement saisonniers (triviaux) et le changements non-triviaux.

Dans cette thèse, nous proposons une approche pour la détection de changements bi-temporels non-triviaux basée sur la transformation patch par patch de *features* entre deux images avec des AEs joints. Le modèle va facilement apprendre comment transformer les zones sans changements et les changements saisonniers de la première image à la deuxième et vice versa. En même temps, comme les changements non-triviaux sont moins nombreux et uniques les *features* seront transformées avec une erreur importante. Le seuillage sur l'erreur de reconstruction nous permettra alors d'identifier ces zones de changement.

Dans notre thèse, nous avons testé deux modèles d'AEs joints pour estimer leurs points forts et faibles :

- *fully-convolutional* AEs (contiennent que des couches convolutives pour l'extraction de *feature maps* (FMs));
- *convolutional* AEs (contiennent que des couches convolutives pour l'extraction de FMs ainsi que les couches linéaires pour la compression de FMs).

Soient $Im_1, Im_2, ..., Im_S$ est une STIS de S images co-registrées acquises sur dates $T_1, T_2, ..., T_S$. Le framework est composé de plusieurs étapes (Figure 4.1).

L'étape de **preprocessing** consiste en la normalisation radiométrique de toute la série.

Le première étape consiste en un **pre-entraînement du modèle** d'AE sur toute la série. Pour cela, on extrait des patchs de la taille $p \times p$ pour chaque pixel i, j de chaque image de la série et on échantillonne $\frac{H \times W}{S}$ patchs de chaque image pour pre-entraîner le modèle.

Pendant la deuxième étape, nous faisons **le fine-tuning** du modèle pour chaque couple d'images consécutives Im_n et Im_{n+1} séparément. Le modèle est composé de deux AEs qui sont initialisés avec les poids de l'AE pre-entraîné. $AE1$ et $AE2$ visent à reconstruire Im'_{n+1} à partir de Im_n et Im'_n de Im_{n+1} respectivement. Les deux AEs sont joints au milieu. Le modèle est optimisé pour minimiser la différence entre les images reconstruites Im'_{n+1} et Im'_n et les images originelles Im_{n+1} et Im_n ainsi que la différence entre les sorties des parties encodeur de deux AEs.

Quand le modèle est stable nous faisons **la reconstruction des images**. Pour chaque patch, nous calculons son erreur de reconstruction. Nous utilisons l'erreur quadratique moyenne pour créer deux images RE_{n+1} et RE_n ou chaque pixel i, j est égal à l'erreur de reconstruction du patch i, j.

Nous calculons la moyenne de deux images RE_{n+1} et RE_n et nous appliquons **le seuillage automatique d'Otsu** pour **produire la carte de changements** $CM_{n,n+1}$ où les pixels changés sont présentés par des zones avec une erreur de reconstruction élevée.

Lors des expériences, il a été défini que nos modèles produisent des résultats plus robustes que les modèles concurrentes basés sur Restricted Boltzmann Machines (RBM) [120] qui peuvent également détecter des changements non-triviaux. Notre méthode est moins sensible aux bruits liés à la luminosité et intuitive à implémenter. Parmi deux modèles testés, nous avons estimé que le modèle convolutif donne des résultats plus robuste comparativement au modèle *fully-convolutional* pour la taille de patch $p = 5$. Parmi les points faibles de cet algorithme, nous avons déterminés qu'il ne détecte pas bien les changements dans les objets linéaires fines, parce que leur taille est inférieure à la taille du patch.

D.4 Détection de Changements Multi-Temporels

La plupart des algorithmes pour la détection de changements multi-temporels sont basés sur l'estimation des tendances saisonnières afin de trouver le comportement atypique [128, 139, 140]. Pour faire cela, la STIS doit être longue (au moins 25 images) et avoir une distribution temporelle régulière ce que peut être parfois inatteignable, surtout pour les séries avec une haute ou très haute résolution spatiale. De plus, dans la plupart de méthodes, on distingues les images archives - pour l'estimation de tendances saisonnières - et les images dans lesquelles nous cherchons les changements ce que ne nous permet pas d'exploiter toutes la série pour l'analyse des changements.

Dans cette thèse, nous proposons un framework pour la détection et modélisation des changements multi-temporels avec des approches basées graphes et leur clustering éventuel. Notre méthode ne dépend pas de nombre d'images et est invariante à la résolution temporelle ce que nous permet de détecter des changements à partir de la deuxième image de la série.

Dans la section précédente, nous avons présenté une méthode pour la détection des changements bi-temporels qui peuvent être interprétés comme les anomalies contextuelles et dépendent que du contexte saisonnier entre deux images. Dans cette section, nous présentons une approche pour l'interprétation de ces changements dans le contexte multi-temporel. Le framework contiennent les étapes suivantes pour une série de la longueur S mentionnée dans la section précédente.

Premièrement, nous faisons **la détection de changements bi-temporels** pour chaque couple d'images Im_n et Im_{n+1}, ainsi que pour chaque couple d'images Im_{n-1} et Im_{n+1}.

Ensuite, nous faisons **l'interprétation des changements dans le contexte multi-temporel**. La carte de changements $CM_{n,n+1}$ est gardée comme référence. Pour chaque objet changé P_i, nous cherchons s'il a une intersection spatiale avec d'autres objets changés sur les autres cartes de changements pour les images consécutives ou avec les objets dans $CM_{n-1,n+1}$. Suivant les règles logiques, nous déterminons si P_i est un changement non-trivial, une anomalie qui a eu lieu une seule fois ou, enfin, un changement faux positif.

Nous isolons les zones changées sur les images en appliquant les masques des changements à la STIS. Nous faisons la **segmentation** des changements non-triviaux détectés avec une méthode de segmentation basée sur les graphes [147].

Les objets changés qui correspondent à la même position géographique dans le temps forment ensuite **des graphes d'évolution** [142, 143] pour visualiser les changements multi-temporels. L'idée principale des graphes d'évolution est la suivante : parmi les segments détectés sur toutes les estampilles temporelles, nous cherchons les *bounding boxes* (BBs) - les objets qui assurent la meilleure couverture de la série sur le plan 2D (le BB doit avoir le minimum des chevauchement avec d'autres BBs). Ensuite, pour chaque BB, les objets couverts par l'empreinte de ce BB aux autres estampilles temporelles forment une graphe d'évolution en respectant les différents paramètres de chevauchement entre les objets définis pas l'utilisateur (Figure 5.6).

Un graphe d'évolution peur contenir plusieurs objets à une estampille temporelle, donc on calcule sa présentation synthétisée - **un synopsis** - qui présente une séquence de la même longueur que la durée de changement et avec le même nombre de *features*

que le nombre de bandes dans les images. Pour un synopsis Q, son élément à l'estampille T_n est calculé comme suit :

$$Q_n = \frac{\sum_1^r Pix(O_j^n) \cdot v_j}{\sum_1^r Pix(O_j^n)} \tag{D.1}$$

où $Pix(O_j^n)$ est la taille de $j-re$ objet à l'estampille T_n ($j \in [1, r]$ où r est le nombre total d'objets présents dans la graphe d'évolution correspondante à T_n) et v_j la moyenne de valeurs de pixels de cet objet. Ici, pour calculer le synopsis, nous utilisons les valeurs encodées des images au lieu des pixels bruts. L'encodage est fait avec un AE convolutif au niveau des patchs.

Finalement, nous utilisons l'AE *gated recurrent unit* (GRU) combiné avec **le clustering** hiérarchique agglomératif (HCA) **pour regrouper les graphes obtenus selon leur type de changement**. GRU [99] est une variante améliorée des modèles récurrents RNN qui permet de mieux conserver les relations des dépendances long-terme dans la séquence. L'AE GRU nous permet d'obtenir la représentation encodée de chaque graphe qui est ensuite passée au clustering hiérarchique [49].

Notre approche de clustering a été testée sur deux jeux de données (1/Montpellier, France - 12 images SPOT-5 acquises entre 2002 à 2008 et 2/Rostov-sur-Don, Russie - 17 images Sentinel-2 acquises entre 2016 et 2018) et a montré la meilleure performance comparativement aux approches concurrentes. Nous avons défini que l'utilisation des valeurs encodés pour le synopsis donne des résultats de clustering plus robustes comparativement à l'exploitation des valeurs bruts. De plus, notre approche à été comparée au clustering basé sur la matrice de distance *dynamic time warping* (DTW) qui calcule la distances entre les séquences en trouvant l'alignement optimal entre elles. Notre approche a montré de meilleurs résultats et la meilleure généralisation.

D.5 Clustering de Série Temporelle d'Images Satellitaires

Dans la section précédente, nous avons proposé un algorithme pour la détection des changement dans le STIS. Pour réaliser une analyse complète d'une zone d'étude, dans cette section, nous présentons le clustering des zones sans changements et des changements saisonniers.

Pour mieux identifier les différents types de végétation ayant des variations saisonnières, il est indispensable de connaitre leur comportement temporel. Les

premières approches de clustering de STIS ont été basées sur l'analyse basée pixel où tous les pixels de la série avec la même position géographique formaient les séquences qui ont été par la suite comparées avec la matrice DTW et clusterisées [154, 155]. Cependant, les approches basées pixel sont souvent lentes et donnent des résultats de clustering bruités. Pour résoudre ce problème, les algorithmes qui exploitent les approches basées objets ont été proposés [156–158]. Le problème principal est que la segmentation de la série est une tache compliquée, puisque les formes d'objets peuvent avoir des variations importantes d'une image à l'autre. Pour cette raison, la segmentation est souvent faite sur 2-3 images ou même sur toutes les images de la série concaténées. Cela résulte en de nombreux objets qui ne sont pas identifiés. Dans [142], les auteurs proposent de représenter les objets de STIS sous forme des graphes d'évolution (voir la Section D.4) et les clusteriser avec le HCA combiné avec la matrice DTW. Cette approche a donné des résultats prometteurs pour les zones naturelles. Malheureusement, la construction des graphes pour les zones urbaines peut être compliquée, car la zone de la ville est très hétérogène et il est impossible d'obtenir des segments qui sont invariants d'une image à l'autre.

Dans cette thèse, nous proposons une approche qui convient pour l'analyse des zones purement agricoles, ainsi que pour les zones mixtes. Notre approche est basée sur la compression de STIS avec un AE 3D convolutif. La série est compressée en une image unique que nous utilisons pour toute analyse. Les étapes de clustering de la série sont décrits ci-dessous.

Comme dans les approches précédentes, la série est d'abord radiométriquement normalisée.

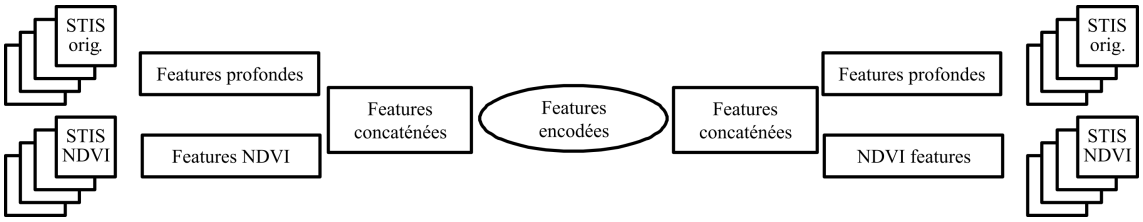

Figure D.2: AE multi-vue double branche.

Par la suite, **la STIS est encodée** en utilisant un AE 3D multi-vue double branche. La première branche est utilisée pour l'extraction des *features* spatio-temporelles profondes de la pile d'images initiales, la deuxième branche fait l'extraction des *textures* générales de la pile d'images NDVI. Notre modèle exploite des couches convolutives 3D pour l'analyse de données spatio-temporelles. Comme dans la Section D.3, nous faisons l'analyse au niveau des patchs qui sont extraits pour chaque

pixel i, j et contiennent toutes les images de la série ($p = 9$ et $p = 5$ pour les images originelles et NDVI respectivement). La taille des patchs est alors $S \times C \times p \times p$ où C est le nombre des canaux de l'image. La série encodée sera représentée sous la forme d'une image avec les mêmes largeur et hauteur que les images initiales et le nombre de canaux choisi par l'utilisateur.

Après nous passons à l'étape de la segmentation. Premièrement, nous faisons **la segmentation préliminaire** sur les deux images les plus représentatives de la série. Les images choisies doivent être les plus différentes possible pour capter le maximum d'objets. Ces images doivent être les plus éloignées l'une de l'autre dans le temps et correspondre à des saisons différentes. Nous utilisons l'algorithme de segmentation MeanShift [166] qui est l'un de les plus utilisé aujourd'hui pour la segmentation classique d'images satellites. Malheureusement, cette approche de segmentation ne nous permet pas de représenter tous les objets de la série, surtout s'il y avait des changements importants. Pour résoudre ce problème, nous faisons **la correction de la segmentation** en utilisant l'image de la série encodée. Comme l'encodage a été fait au niveau des patchs, certains pixels de l'image encodée ont l'effet de bordure. Cet effet est lié au fait que certains patchs couvrent des objets différents, donc les pixels encodés correspondants contiennent les *features* mixtes. Pour cette raison, la segmentation de l'image encodée aura des objets avec les bordures déplacées ou même les objets parasites. Nous combinons les résultats de deux segmentations en rajoutant dans la segmentation préliminaire les objets manquants de la segmentation de l'image encodée en conservant les bordures correctes. Cela nous permettra de représenter un maximum d'objets de la série.

Finalement, pour chaque segment, on extrait sa valeur médiane encodée afin de **clusteriser la série**. Nous utilisons le clustering hiérarchique comme dans la méthode d'analyse des changements multi-temporels.

Notre méthode a été testée sur deux séries temporelles de l'agglomération de Montpellier (1/ 12 images SPOT-5 acquises entre 2002 à 2008 et 2/ 24 images Sentinel-2 acquises entre 2017 et 2018) et atteint de meilleurs résultats par rapport aux methodes concurrentes basées objets et basées graphes mentionnées plus haut. Nous avons également prouvé que la branche NDVI peut améliorer résolument les résultats du clustering. De plus, nous avons montré que la correction de la segmentation préliminaire donne une carte de segmentation plus précise. Malheureusement, certain objets linéaires ne sont pas correctement clusterisés ce que peut être expliqué par le fait qu'ils sont moins larges que les patchs extraits.

D.6 Résumé des Contributions Scientifiques et Perspectives

Dans la cadre de cette thèse, nous avons développé un ensemble d'algorithmes pour l'analyse de STIS non-supervisé qui exploite des techniques d'apprentissage profonde. Ci-dessous, nous synthétisons les contributions de notre travail et proposons des pistes pour la recherche future.

D.6.1 Contributions de la Thèse

Dans cette thèse, nous avons proposé un ensemble de méthodes qui vise à identifier trois types de comportements temporels présents dans la STIS : les zones sans changement, changements saisonniers et les changements non-triviaux. Les contributions de cette thèse peuvent être divisées en trois parties différentes.

Dans la première partie (Chapitre 4), nous avons proposé un algorithme non-supervisé pour la détection des changements bi-temporels non-triviaux dans les images satellites. Nous avons exploité un modèle des réseaux de neurones basé sur les AEs joints pour la transformation des *features* pour détecter des changements entre deux images consécutives d'une série. Contrairement aux algorithmes traditionnels pour la détection des changements qui détectent "tout qui change", notre algorithme détecte seulement des changements non-triviaux et ignore les tendances saisonnières.

Dans la deuxième partie (Chapitre 5), nous avons développé un algorithme qui interprète les changements bi-temporels détectés dans le contexte multi-temporel. Nous avons proposé une méthode pour modéliser le phénomène des changements spatio-temporels en forme de graphes d'évolution. Ensuite, pour clustériser ces graphes de changements, nous avons élaboré un modèle basé sur l'extraction des *features* avec l'autoencodeur GRU. De plus, l'approche proposée ne dépend pas de la résolution temporelle de la série et ne demande pas qu'elle soit longue pour détecter les différentes tendances saisonnières.

Dans la troisième partie (Chapitre 6), nous avons introduit une approche basée objets pour le clustering de STIS qui exploite l'AE conventionnel 3D pour l'extraction des *features* spatio-temporels pour identifier les différents clusters avec les zones sans changements et les changements saisonniers. Nous avons proposé un AE 3D multi-vue double branche qui extrait des *features* plus robustes comparativement à l'AE convolutionnel 3D classique. En outre, nous avons développé une méthode pour créer la segmentation unique pour toute la série temporelle. Finalement, nous

avons réalisé le clustering basé objets complet de STIS en exploitant les propriétés spatio-temporels de segments détectés.

D.6.2 Perspectives

Perspectives à Court Termes

Les méthodes présentées dans cette thèse ont montré leur efficacité sur les jeux de données expérimentales. Cependant, en raison de l'absence de données de référence, les méthodes non-supervisées ne produisent pas toujours les résultats attendus. Certaines techniques peuvent être intégrées dans notre analyse de STIS afin d'améliorer sa qualité. Dans cette partie, nous proposons de résumer certain perspectives pour notre recherche.

Détection de changements bi-temporels : Notre méthode exploite la détection de changements patch par patch, donc aucune analyse au niveau des objets n'a pas été réalisée. Pour cette raison, l'algorithme peut rater certains pixels changés dans les limites d'un objet ou, au contraire, détecter quelques pixels de changements faux positifs dans un objet qui n'a pas changé. On peut améliorer les résultats de l'algorithme en introduisant l'analyse morphologique ou en affinant les changements détectés avec les approches basées objet.

Détection de changements multi-temporels : Notre algorithme pour la détection de changements multi-temporels est basé sur l'interprétation des changements bi-temporels dans le contexte multi-temporel. Pour cette raison, il détecte seulement les tendances saisonnières majeures des STIS et marque les changements saisonniers mineurs comme des changements non-triviaux. Pour résoudre ce problème, nous proposons d'analyser les changements non-triviaux détectés ensemble avec les résultats de clustering de STIS (voir la Section D.5). Dans les recherches futures, des méthodes pour détecter des tendances mineures peuvent être proposées.

De plus, l'interprétation des graphes pourrait être améliorée : notre algorithme fait la description globale des changements détectés sans identifier les sous-processus présents ce que pourrait nous donner de l'information supplémentaire.

Clustering de STIS : Notre algorithme pourrait être adapté à toutes les bandes spectrales des images Sentinel-2. D'autres indices spectrales pourraient être intégrées après une estimation de leur influence sur les résultats. De plus, le framework

proposé pourrait être amélioré pour la détection et le clustering des objets linéaires. Par exemple, on pourrait combiner les descripteurs de *features* encodées avec les valeurs brutes pour surmonter le problème d'effet de bordure. Finalement, l'analyse contextuelle pourrait être intégrée pour distinguer plus de clusters (par exemple, si une zone artificielle est à côte de surface en eau, on présume que c'est une plage, donc le cluster est différent de la zone urbaine).

Perspectives à Long Termes et Limitations

Les perspectives à long termes les plus intéressantes seront d'intégrer les différentes contraintes ou même quelques données labellisées dans nos algorithmes. Comme nous avons mentionné ci-dessus, les algorithmes non-supervisés et surtout le clustering produisent rarement le résultat attendu (les clusters obtenus ne peuvent pas être directement associés aux classes réelles, par exemple, une classe peut être représentée par plusieurs clusters). Pour améliorer le résultat, l'algorithme du *one-shot learning* [168] pourrait être ajouté dans notre framework. L'idée de cet algorithme est la suivante : pour chaque classe nous avons une seule entité étiquetée qui doit faire sa meilleure représentation. Ces entités sont ensuite utilisées pour entraîner un modèle de classification (ou faire son *fine-tuning*). Cela pourrait significativement améliorer les résultats de clustering de STIS. Toutefois, cet algorithme ne pourrait pas être utilisé pour le clustering des graphes de changements multi-temporels, car les changements détectés sont souvent très variés et il est impossible de connaitre en avance tous les types de changements.

Dans cette thèse, nous faisons le clustering pour le différent nombre de clusters afin de choisir le meilleure résultat. Le nombre optimal de clusters est une question ouverte et peu étudiée. Une étude pour la validation automatique du nombre optimal de clusters pourrait être ajouté à la méthode de clustering des changements multi-temporels pour minimiser l'interaction avec l'utilisateur.

Finalement, nous croyons que les différentes contraintes pourraient améliorer le clustering de changements. Par exemple, on pourrait prendre en compte les dates d'acquisition des images ou les différentes indices spectrales pour estimer approximativement le type d'objet changé.

Bibliography

[1] Markus Goldstein and Seiichi Uchida, "A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data," *PloS one*, vol. 11, 04 2016. (document), 2.8, 2.4, 2.4, 5.3.2

[2] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, Dec 2017. (document), 3.3, 3.2, D.2.1

[3] "MSI Instrument - Sentinel-2 MSI Technical Guide - Sentinel Online," `https://earth.esa.int/web/sentinel/technical-guides/sentinel-2-msi/msi-instrument`, (Accessed on 03/13/2020). (document), 1.2

[4] United States. National Aeronautics, Space Administration. Scientific, and Technical Information Division, *Scientific Findings from Explorer VI*, NASA SP. U.S. Government Printing Office, 1965. 1.3

[5] "Landsat 1 - Landsat Science," `https://landsat.gsfc.nasa.gov/landsat-1/`, (Accessed on 03/09/2020). 1.3

[6] "SPOT-5 - eoPortal Directory - Satellite Missions," `https://earth.esa.int/web/eoportal/satellite-missions/s/spot-5`, (Accessed on 03/13/2020). 1.3

[7] "Missions - Sentinel," `https://sentinel.esa.int/web/sentinel/missions`, (Accessed on 03/12/2020). 1.3

[8] Bernd Jähne, *Digital Image Processing 6th Edition*, Springer, Berlin [u.a.], 2005. 1.4.1

[9] Bhupendra Jasani, Martino Pesaresi, Stefan Schneiderbauer, and Gunter Zeug, *Remote Sensing from Space: Supporting International Peace and Security*, Springer Publishing Company, Incorporated, 1st edition, 2009.

[10] John A. Richards and Xiuping Jia, *Remote Sensing Digital Image Analysis: An Introduction*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 3rd edition, 1999. 1.4.1

[11] A. Bechtel, W. Puttmann, T. N. Carlson, and D. A. Ripley, "On the Relation between NDVI, Fractional Vegetation Cover, and Leaf Area Index," *Remote Sensing of Environment*, vol. 62, no. 3, pp. 241–252, Dec. 1997. 1.4.1

[12] Josef Kittler and J. Föglein, "Contextual classification of multispectral pixel data," *Image Vision Comput.*, vol. 2, no. 1, pp. 13–29, 1984. 1.4.1

[13] J. Besag, "On the statistical analysis of dirty pictures," *Journal of the Royal Statistical Society. Series B*, vol. 48, no. 3, pp. 259–302, 1986.

[14] Shashi Shekhar, Senior Member, Paul R. Schrater, Ranga R. Vatsavai, Weili Wu, and Sanjay Chawla, "Spatial Contextual Classification and Prediction Models for Mining Geospatial Data," *IEEE Transactions on Multimedia*, vol. 4, pp. 174–188, 2002. 1.4.1

[15] Qihao Weng, "Remote sensing of impervious surfaces in the urban areas: Requirements, methods, and trends," *Remote Sensing of Environment*, vol. 117, no. 0, pp. 34–49, 2012. 1.4.1

[16] T. Blaschke, "Object based image analysis for remote sensing," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 65, no. 1, pp. 2–16, Jan. 2010. 1.4.2

[17] Rafael C. Gonzalez and Richard E. Woods, *Digital Image Processing (3rd Edition)*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006. 1.4.2

[18] T. Pavlidis, *Structural Pattern Recognition*, Springer, 1977. 1.4.2

[19] S. W. Zucker, "Region growing: Childhood and Adolescence," *Computer Graphics and Image Processing*, vol. 5, no. 3, pp. 382–399, Sept. 1976. 1.4.2, 2.1

[20] Nikhil R. Pal and Sankar K. Pal, "A review on image segmentation techniques," *Pattern Recognition*, vol. 26, no. 9, pp. 1277–1294, Sept. 1993. 1.4.2

[21] Adi L. Tarca, Vincent J. Carey, Xue-Wen Chen, Roberto Romero, and Sorin Drăghici, "Machine Learning and Its Applications to Biology," *PLoS Comput Biol*, vol. 3, no. 6, pp. e116+, June 2007. 2.1

[22] Igor Kononenko, "Machine Learning for Medical Diagnosis: History, State of the Art and Perspective," *Artif. Intell. Med.*, vol. 23, no. 1, pp. 89–109, Aug. 2001.

[23] Élisa Fromont, René Quiniou, and Marie-Odile Cordier, "Learning Rules from Multisource Data for Cardiac Monitoring," pp. 484–493, 2005.

[24] Daoqiang Zhang, Yaping Wang, Luping Zhou, Hong Yuan, and Dinggang Shen, "Multimodal classification of Alzheimer's disease and mild cognitive impairment," *NeuroImage*, vol. 55, no. 3, pp. 856–867, 2011. 2.1

[25] Weizhong Zhang, Lijun Zhang, Yao Hu, Rong Jin, Deng Cai, and Xiaofei He, "Sparse Learning for Stochastic Composite Optimization," in *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI)*, 2014, pp. 893–899. 2.1

[26] Indranil Bose and Radha K. Mahapatra, "Business data mining - a machine learning perspective," *Information & Management*, vol. 39, no. 3, pp. 211–225, 2001. 2.1

[27] Q. Yu, A. Sorjamaa, Y. Miche, and E. Severin, "A methodology for time series prediction in Finance," in *ESTSP, European Symposium on Time Series Prediction*, Amaury Lendasse, Ed., Porvoo, Finland, September 17-19 2008, pp. 285–293, Multiprint Oy / Otamedia , Espoo, Finland.

[28] Victor Boyarshinov, *Machine Learning in Computational Finance*, Ph.D. thesis, Rensselaer Polytechnic Institute, Troy, NY, USA, 2005, AAI3173253. 2.1

[29] NICHOLAS M. BALL and ROBERT J. BRUNNER, "Data Mining and Machine Learning in Astronomy," *International Journal of Modern Physics D*, vol. 19, no. 07, pp. 1049–1106, 2010. 2.1

[30] Kazutoshi Tanabe, Bono Lučić, Dragan Amić, Takio Kurita, Mikio Kaihara, Natsuo Onodera, and Takahiro Suzuki, "Prediction of carcinogenicity for diverse chemicals based on substructure grouping and SVM modeling," *Molecular Diversity*, vol. 14, no. 4, pp. 789–802, 2010. 2.1

[31] Victoria Hodge and Jim Austin, "A Survey of Outlier Detection Methodologies," *Artif. Intell. Rev.*, vol. 22, no. 2, pp. 85–126, Oct. 2004. 2.3

[32] P. J. Rousseeuw and A. M. Leroy, *Robust Regression and Outlier Detection*, John Wiley & Sons, Inc., New York, NY, USA, 1987. 2.3, B

[33] Fatma Hamdi and Younès Bennani, "Learning random subspace novelty detection filters," in *The 2011 International Joint Conference on Neural Networks, IJCNN 2011, San Jose, California, USA, July 31 - August 5, 2011*, 2011, pp. 2273–2280. 2.3

[34] Pavel Berkhin, "Survey Of Clustering Data Mining Techniques," Tech. Rep., Accrue Software, 2002. 2.3

[35] Rui Xu and D. Wunsch, II, "Survey of Clustering Algorithms," *Trans. Neur. Netw.*, vol. 16, no. 3, pp. 645–678, May 2005.

[36] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data Clustering: A Review," *ACM Computing Surveys*, vol. 31, no. 3, pp. 264–323, Sept. 1999. 2.3, 2.3.2, 2.3.2

[37] J. B. MacQueen, "Some Methods for Classification and Analysis of Multivariate Observations," in *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, 1967, pp. 281–297. 2.3.1, 2.3.1

[38] Hugo Steinhaus, "Sur la division des corps matériels en parties," *Bull. Acad. Polon. Sci. Cl. III. 4*, pp. 801–804, 1956. 2.3.1

[39] M. Ester, H.-P. Kriegel, J. Sander, and X Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *In International Conference on Knowledge Discovery and Information Retrieval*, 1996, pp. 226–231. 2.3.1, 2.3.3

[40] Gereon Frahling and Christian Sohler, "A fast k-means implementation using coresets," in *SCG '06: Proceedings of the twenty-second annual symposium on Computational geometry*, New York, NY, USA, 2006, pp. 135–143, ACM.

[41] Chieh-Yuan Tsai and Chuang-Cheng Chiu, "Developing a feature weight self-adjustment mechanism for a K-means clustering algorithm," *Comput. Stat. Data Anal.*, vol. 52, no. 10, pp. 4658–4672, 2008.

[42] JA Hartigan and MA Wong, "Algorithm AS 136: A K-means clustering algorithm," *Applied Statistics*, pp. 100–108, 1979.

[43] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu, "An Efficient k-Means Clustering Algorithm: Analysis and Implementation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 881–892, July 2002.

[44] Juan Antonio Cuesta-Albertos and Ricardo Fraiman, "Impartial trimmed k-means for functional data," *Comput. Stat. Data Anal.*, vol. 51, no. 10, pp. 4864–4877, 2007.

[45] Shu-Chuan Chu, John Roddick, and Jeng-Shyang Pan, "Improved search strategies and extensions to k-medoids based clustering algorithms," *Int. J. Bus. Intell. Data Min.*, vol. 3, no. 2, pp. 212–231, 2008. 2.3.1

[46] R.J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics.*, vol. 20, pp. 53–65, 1987. 2.3.1

[47] David L. Davies and Donald W. Bouldin, "A Cluster Separation Measure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 1, no. 2, pp. 224–227, Feb. 1979. 2.3.1, B, B

[48] Dan Pelleg and Andrew Moore, "X-means: Extending K-means with efficient estimation of the number of clusters," in *In Proceedings of the 17th International Conf. on Machine Learning*, 2000, pp. 727–734. 2.3.1

[49] Joe H. Ward Jr., "Hierarchical Grouping to Optimize an Objective Function," *Journal of the American Statistical Association*, vol. 58, no. 301, pp. 236–244, 1963. 2.3.2, 2.3.2, 5.1, 5.3.6, 5.5.1, 6.2, 6.3.3, 6.5.2, D.4

[50] Youbao Tang, Xiangqian Wu, and Wei Bu, "Saliency Detection Based on Graph-Structural Agglomerative Clustering," in *Proceedings of the 23rd ACM International Conference on Multimedia*, New York, NY, USA, 2015, MM '15, pp. 1083–1086, ACM.

[51] Wei Zhang, Deli Zhao, and Xiaogang Wang, "Agglomerative clustering via maximum incremental path integral," *Pattern Recognition*, vol. 46, no. 11, pp. 3056–3065, 2013.

[52] William H. Day and Herbert Edelsbrunner, "Efficient algorithms for agglomerative hierarchical clustering methods," *Journal of Classification*, vol. 1, no. 1, pp. 7–24, December 1984. 2.3.2

[53] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim, "CURE: An Efficient Clustering Algorithm for Large Databases," *SIGMOD Rec.*, vol. 27, no. 2, pp. 73–84, jun 1998. 2.3.2

[54] Jörg Sander, Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu, "Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications," *Data Min. Knowl. Discov.*, vol. 2, no. 2, pp. 169–194, 1998. 2.3.3

[55] Daoying Ma and Aidong Zhang, "An Adaptive Density-Based Clustering Algorithm for Spatial Database with Noise," *Data Mining, IEEE International Conference on*, vol. 0, pp. 467–470, 2004. 2.3.3

[56] Jiawei Han, Micheline Kamber, and Jian Pei, *Data Mining: Concepts and Techniques*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2011. 2.3.3

[57] P. Viswanath and Rajwala Pinkesh, "l-DBSCAN: A Fast Hybrid Density Based Clustering Method," in *ICPR '06: Proceedings of the 18th International Conference on Pattern Recognition*, Washington, DC, USA, 2006, pp. 912–915, IEEE Computer Society. 2.3.3

[58] Jörg Sander, Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu, "Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications," *Data Min. Knowl. Discov.*, vol. 2, no. 2, pp. 169–194, 1998. 2.3.3

[59] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander, "OPTICS: Ordering Points To Identify the Clustering Structure," in *In ACM SIGMOD international conference on Management of data*. 1999, pp. 49–60, ACM Press. 2.3.3

[60] Jianbo Shi and Jitendra Malik, "Normalized Cuts and Image Segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, Aug. 2000. 2.3.4

[61] Marina Meila and Jianbo Shi, "Learning Segmentation by Random Walks," in *In Advances in Neural Information Processing Systems*. 2001, pp. 873–879, MIT Press. 2.3.4, 3.1

[62] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society. Series B*, vol. 39, no. 1, pp. 1–38, 1977. 2.3.5

[63] Sreerupa Das and Michael Mozer, "A Unified Gradient-Descent/Clustering Architecture for Finite State Machine Induction," in *NIPS*, Jack D. Cowan, Gerald Tesauro, and Joshua Alspector, Eds. 1993, pp. 19–26, Morgan Kaufmann. 2.3.5

[64] Robert Jenssen, Deniz Erdogmus, Kenneth E. Hild II, Jose C. Principe, and Torbjørn Eltoft, "Information cut for clustering using a gradient descent approach," *Pattern Recognition*, vol. 40, no. 3, pp. 796 – 806, 2007. 2.3.5

[65] Varun Chandola, Arindam Banerjee, and Vipin Kumar, *Anomaly Detection*, pp. 1–15, Springer US, Boston, MA, 2016. 2.4

[66] Varun Chandola, Arindam Banerjee, and Vipin Kumar, "Anomaly Detection: A Survey," *ACM Comput. Surv.*, vol. 41, 07 2009.

[67] M. Gupta, J. Gao, C. C. Aggarwal, and J. Han, "Outlier Detection for Temporal Data: A Survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 9, pp. 2250–2267, 2014. 2.4

[68] Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis, "Clustering Validity Checking Methods: Part II," *ACM SIGMOD Record*, vol. 31, 09 2002. 2.5

[69] Anil K. Jain and Richard C. Dubes, *Algorithms for clustering data*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[70] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar, *Introduction to Data Mining, (First Edition)*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005. 2.5

[71] Thomas M. Cover and Joy A. Thomas, *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*, Wiley-Interscience, USA, 2006. 2.5.1, 6.5.1

[72] W.M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical Association.*, pp. 846–850, 1971. 2.5.1

[73] L. Hubert and P. Arabie, "Comparing Partitions," *Journal of the Classification*, vol. 2, pp. 193–218, 1985. 2.5.1

[74] J. S. Deng, K. Wang, Y. H. Deng, and G. J. Qi, "PCA-based land-use change detection and analysis using multitemporal and multisensor satellite data," *International Journal of Remote Sensing*, vol. 29, no. 16, pp. 4823–4838, 2008. 3.1

[75] R. A. FISHER, "The Use of Multiple Measurements in Taxonomic Problems," *Annals of Eugenics*, vol. 7, no. 2, pp. 179–188, 1936. 3.1

[76] Joseph Kruskal and Myron Wish, *Multidimensional Scaling*, SAGE Publications, Inc., Thousand Oaks, California, 1978. 3.1

[77] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford, "A Global Geometric Framework for Nonlinear Dimensionality Reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000. 3.1

[78] Sam T. Roweis and Lawrence K. Saul, "Nonlinear Dimensionality Reduction by Locally Linear Embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000. 3.1

[79] Xifeng Guo, Xinwang Liu, En Zhu, and Jianping Yin, "Deep Clustering with Convolutional Autoencoders," in *Neural Information Processing*, Derong Liu, Shengli Xie, Yuanqing Li, Dongbin Zhao, and El-Sayed M. El-Alfy, Eds. 2017, pp. 373–382, Springer International Publishing. 3.1, 3.3, D.2.1

[80] Karen Simonyan and Andrew Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *International Conference on Learning Representations*, 2015. 3.1, 4.2

[81] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, "Imagenet Classification with Deep Convolutional Neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, may 2017. 3.1

[82] Robert Haralick, K. Shanmugam, and Ih Dinstein, "Textural Features for Image Classification," *IEEE Trans Syst Man Cybern*, vol. SMC-3, pp. 610–621, 01 1973. 3.1, C.1

[83] Jr. Rouse, J. W., R. H. Haas, J. A. Schell, and D. W. Deering, *Monitoring Vegetation Systems in the Great Plains with Erts*, vol. 351, p. 309, 1974. 3.1, 6.2, C.2

[84] Syed Anwar, Muhammad Majid, Adnan Qayyum, Muhammad Awais, Majdi Alnowami, and Khurram Khan, "Medical Image Analysis using Convolutional Neural Networks: A Review," *Journal of Medical Systems*, vol. 42, pp. 226, 10 2018. 3.2

[85] Mei Wang and Weihong Deng, "Deep Face Recognition: A Survey," *CoRR*, vol. abs/1804.06655, 2018. 3.2

[86] M. Teichmann, M. Weber, M. Zöllner, R. Cipolla, and R. Urtasun, "MultiNet: Real-time Joint Semantic Reasoning for Autonomous Driving," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, 2018, pp. 1013–1020. 3.2

[87] V. John, A. Boyali, S. Mita, M. Imanishi, and N. Sanma, "Deep Learning-Based Fast Hand Gesture Recognition Using Representative Frames," in *2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, 2016, pp. 1–8. 3.2

[88] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "EMNIST: Extending MNIST to handwritten letters," in *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 2921–2926. 3.2

[89] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," 10 2015, vol. 9351, pp. 234–241. 3.3, 4.2, D.2.1

[90] A. Sento, "Image compression with auto-encoder algorithm using deep neural network (DNN)," pp. MIT–99–MIT–103, Oct 2016. 3.3, D.2.1

[91] Xiao-Jiao Mao, Chunhua Shen, and Yu-Bin Yang, "Image Restoration Using Very Deep Convolutional Encoder-Decoder Networks with Symmetric Skip Connections," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2016, NIPS'16, pp. 2810–2818, Curran Associates Inc. 3.3, D.2.1

[92] Jonathan Masci, Ueli Meier, Dan Ciresan, and JÃŒrgen Schmidhuber, "Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction," 06 2011, pp. 52–59. 3.3, D.2.1

[93] Chen Xing, Li Ma, and Xiaoquan Yang, "Stacked Denoise Autoencoder Based Feature Extraction and Classification for Hyperspectral Images," *Journal of Sensors*, vol. 2016, pp. 1–10, 01 2016. 3.3, 6.2, D.2.1

[94] Xifeng Guo, Long Gao, Xinwang Liu, and Jianping Yin, "Improved Deep Embedded Clustering with Local Structure Preservation," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 2017, pp. 1753–1759. 3.3, 5.5.1, D.2.1

[95] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, Adaptive Computation and Machine Learning series. MIT Press, 2016. 3.3, 6.2

[96] Vincent Dumoulin and Francesco Visin, "A guide to convolution arithmetic for deep learning," *ArXiv*, vol. abs/1603.07285, 2016. 3.4.1

[97] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams, *Learning Representations by Back-Propagating Errors*, pp. 696–699, MIT Press, Cambridge, MA, USA, 1988. 3.4.5

[98] Sepp Hochreiter and Jürgen Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. 3.4.5

[99] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio, "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, Oct. 2014, pp. 1724–1734, Association for Computational Linguistics. 3.4.5, 5.1, 5.3.6, D.4

[100] Emile Ndikumana, Dinh Ho Tong Minh, Nicolas Baghdadi, Dominique Courault, and Laure Hossard, "Deep Recurrent Neural Network for Agricultural Classification using multitemporal SAR Sentinel-1 for Camargue, France," *Remote Sensing*, vol. 10, no. 8, 2018. 3.4.5

[101] Haowen Luo, "Shorten spatial-spectral RNN with parallel-gru for hyperspectral image classification," *CoRR*, vol. abs/1810.12563, 2018.

[102] L. Mou, P. Ghamisi, and X. X. Zhu, "Deep Recurrent Neural Networks for Hyperspectral Image Classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 7, pp. 3639–3655, July 2017. 3.4.5

[103] Jeff Donahue, Lisa Anne Hendricks, Marcus Rohrbach, Subhashini Venugopalan, Sergio Guadarrama, Kate Saenko, and Trevor Darrell, "Long-term Recurrent Convolutional Networks for Visual Recognition and Description," 2014. 3.4.5

[104] Rodrigo Caye Daudt; Bertrand Le Saux; Alexandre Boulch; Yann Gousseau, "OSCD - Onera Satellite Change Detection," 2019. 4.2

[105] Rodrigo Caye Daudt, Bertrand Le Saux, Alexandre Boulch, and Yann Gousseau, "Urban Change Detection for Multispectral Earth Observation Using Convolutional Neural Networks," in *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*. jul 2018, IEEE. 4.2

[106] Jie Wang, Chang Luo, Hanqiao Huang, Huizhen Zhao, and Shiqiang Wang, "Transferring Pre-Trained Deep CNNs for Remote Scene Classification with General Features Learned from Linear PCA Network," *Remote Sensing*, vol. 9, no. 3, pp. 225, Mar 2017. 4.2

[107] J. Liu, K. Chen, G. Xu, X. Sun, M. Yan, W. Diao, and H. Han, "Convolutional Neural Network-Based Transfer Learning for Optical Aerial Images Change Detection," *IEEE Geoscience and Remote Sensing Letters*, vol. 17, no. 1, pp. 127–131, 2020. 4.2

[108] A. M. El Amin, Q. Liu, and Y. Wang, "Zoom out CNNs features for optical remote sensing change detection," in *2017 2nd International Conference on Image, Vision and Computing (ICIVC)*, 2017, pp. 812–817. 4.2

[109] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015. 4.2

[110] T. Celik, "Unsupervised Change Detection in Satellite Images Using Principal Component Analysis and k-Means Clustering," *IEEE Geoscience and Remote Sensing Letters*, vol. 6, no. 4, pp. 772–776, 2009. 4.2, D.3

[111] F. Bovolo, S. Marchesi, and L. Bruzzone, "A Framework for Automatic and Unsupervised Detection of Multiple Changes in Multitemporal Images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 50, no. 6, pp. 2196–2212, 2012. 4.2

[112] He Pengfei, Wenzhong Shi, Hua Zhang, and Ming Hao, "A novel dynamic threshold method for unsupervised change detection from remotely sensed images," *Remote Sensing Letters*, vol. 5, pp. 396–403, 03 2014. 4.2, D.3

[113] Yin Chen and Zhiguo Cao, "An improved MRF-based change detection approach for multitemporal remote sensing imagery," *Signal Processing*, vol. 93, no. 1, pp. 163 – 175, 2013. 4.2, D.3

[114] Ming Hao, Hua Zhang, Wenzhong Shi, and Kazhong Deng, "Unsupervised change detection using fuzzy c-means and MRF from remotely sensed images," *Remote Sensing Letters*, vol. 4, no. 12, pp. 1185–1194, 2013. 4.2, D.3

[115] C. Huo, Z. Zhou, H. Lu, C. Pan, and K. Chen, "Fast Object-Level Change Detection for VHR Images," *IEEE Geoscience and Remote Sensing Letters*, vol. 7, no. 1, pp. 118–122, 2010. 4.2, D.3

[116] Guo Cao, Yupeng Li, Yazhou Liu, and Yanfeng Shang, "Automatic change detection in high-resolution remote-sensing images by means of level set evolution and support vector machine classification," *International Journal of Remote Sensing*, vol. 35, pp. 6255–6270, 07 2014. 4.2, D.3

[117] K. Tan, X. Jin, A. Plaza, X. Wang, L. Xiao, and P. Du, "Automatic Change Detection in High-Resolution Remote Sensing Images by Using a Multiple Classifier System and Spectral-Spatial Features," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 9, no. 8, pp. 3439–3451, Aug 2016. 4.2

[118] Guo Cao, Bisheng Wang, Haro-Carrión Xavier, Di Yang, and Jane Southworth, "A new difference image creation method based on deep neural networks for change detection in remote-sensing images," *International Journal of Remote Sensing*, vol. 38, no. 23, pp. 7161–7175, 2017. 4.2

[119] A. A. Nielsen, "The Regularized Iteratively Reweighted MAD Method for Change Detection in Multi- and Hyperspectral Data," *IEEE Transactions on Image Processing*, vol. 16, no. 2, pp. 463–478, Feb 2007. 4.2

[120] Yuan Xu, Shiming Xiang, Chunlei Huo, and Chunhong Pan, "Change Detection Based on Auto-encoder Model for VHR Images," *Proc SPIE*, vol. 8919, pp. 02–, 10 2013. 4.2, 4.3, 4.5.1, 4.5.1, D.3

[121] G. E. Hinton and R. R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006. 4.2

[122] Wei Cui and Qi Zhou, "Application of a Hybrid Model Based on a Convolutional Auto-Encoder and Convolutional Neural Network in Object-Oriented Remote Sensing Classification," *Algorithms*, vol. 11, pp. 9, 01 2018. 4.3, 6.2

[123] Mahmoud El Hajj, Agnès Bégué, Bruno Lafrance, Olivier Hagolle, Gérard Dedieu, and Matthieu Rumeau, "Relative Radiometric Normalization and Atmospheric Correction of a SPOT 5 Time Series," *Sensors*, vol. 8, no. 4, pp. 2774–2791, 2008. 4.3.1, 1, A.1, A.2, A.3

[124] N. Otsu, "A Threshold Selection Method from Gray-Level Histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, Jan 1979. 4.3.1, 4.3.2

[125] C. Aytekin, X. Ni, F. Cricri, and E. Aksu, "Clustering and Unsupervised Anomaly Detection with l2 Normalized Deep Auto-Encoder Representations," in *2018 International Joint Conference on Neural Networks (IJCNN)*, 2018, pp. 1–6. 4.3.2

[126] Mailys Lopes, Mathieu Fauvel, Annie Ouin, and Stéphane Girard, "Spectro-Temporal Heterogeneity Measures from Dense High Spatial Resolution Satellite Image Time Series: Application to Grassland Species Diversity Estimation," *Remote Sensing*, vol. 9, no. 10, 2017. 5.1

[127] Xiao-Peng Song, Chengquan Huang, Joseph O. Sexton, Saurabh Channan, and John R. Townshend, "Annual Detection of Forest Cover Loss Using Time Series Satellite Measurements of Percent Tree Cover," *Remote Sensing*, vol. 6, no. 9, pp. 8878–8903, 2014. 5.1, 6.1

[128] Loïc Paul Dutrieux, Jan Verbesselt, Lammert Kooistra, and Martin Herold, "Monitoring forest cover loss using multiple data streams, a case study of a tropical dry forest in Bolivia," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 107, pp. 112 – 125, 2015, Multitemporal remote sensing data analysis. 5.2, D.4

[129] Zhe Zhu, Curtis E. Woodcock, and Pontus Olofsson, "Continuous monitoring of forest disturbance using all available Landsat imagery," *Remote Sensing of Environment*, vol. 122, pp. 75 – 91, 2012, Landsat Legacy Special Issue. 5.2

[130] Mattijn Van Hoek, Li Jia, Jie Zhou, Chaolei Zheng, and Massimo Menenti, "Early Drought Detection by Spectral Analysis of Satellite Time Series of

Precipitation and Normalized Difference Vegetation Index (NDVI)," *Remote Sensing*, vol. 8, no. 5, 2016. 5.1, 6.1, C.2

[131] Stephanie Olen and Bodo Bookhagen, "Mapping Damage-Affected Areas after Natural Hazard Events Using Sentinel-1 Coherence Time Series," *Remote Sensing*, vol. 10, no. 8, 2018. 5.1

[132] T. Lei, Q. Zhang, D. Xue, T. Chen, H. Meng, and A. K. Nandi, "End-to-end Change Detection Using a Symmetric Fully Convolutional Network for Landslide Mapping," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2019, pp. 3027–3031. 5.1

[133] Abdullah Alqurashi, Lalit Kumar, and Priyakant Sinha, "Urban Land Cover Change Modelling Using Time-Series Satellite Images: A Case Study of Urban Growth in Five Cities of Saudi Arabia," *Remote Sensing*, vol. 8, 11 2016. 5.1, 6.1

[134] Sajid Pareeth, Poolad Karimi, Mojtaba Shafiei, and Charlotte De Fraiture, "Mapping Agricultural Landuse Patterns from Time Series of Landsat 8 Using Random Forest Based Hierarchial Approach," *Remote Sensing*, vol. 11, no. 5, 2019. 5.1

[135] Urška Kanjir, Nataša Durić, and Tatjana Veljanovski, "Sentinel-2 Based Temporal Detection of Agricultural Land Use Anomalies in Support of Common Agricultural Policy Monitoring," *ISPRS International Journal of Geo-Information*, vol. 7, no. 10, 2018. 5.1, C.2

[136] Roberto Interdonato, Dino Ienco, Raffaele Gaetano, and Kenji Ose, "DuPLO: A DUal view Point deep Learning architecture for time series classificatiOn," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 149, pp. 91–104, Mar. 2019. 5.1, 6.1

[137] Charlotte Pelletier, Geoffrey I. Webb, and François Petitjean, "Temporal Convolutional Neural Network for the Classification of Satellite Image Time Series," *Remote Sensing*, vol. 11, no. 5, 2019. 5.1, 6.1

[138] Yun-Long Kong, Qingqing Huang, Chengyi Wang, Jingbo Chen, Jiansheng Chen, and Dongxu He, "Long Short-Term Memory Neural Networks for Online Disturbance Detection in Satellite Image Time Series," *Remote Sensing*, vol. 10, no. 3, 2018. 5.2

[139] Jan Verbesselt, Rob Hyndman, Glenn Newnham, and Darius Culvenor, "Detecting trend and seasonal changes in satellite image time series," *Remote Sensing of Environment*, vol. 114, no. 1, pp. 106 – 115, 2010. 5.2, D.4

[140] Shanshan Cai and Desheng Liu, "Detecting Change Dates from Dense Satellite Time Series Using a Sub-Annual Change Detection Algorithm," *Remote Sensing*, vol. 7, no. 7, pp. 8705–8727, 2015. 5.2, C.2, D.4

[141] Corina Vaduva, Cosmin Danisor, and Mihai Datcu, "Joint SAR image time series and PSInSAR data analytics: An LDA based approach," *Remote Sensing*, vol. 10, pp. 1436, 09 2018. 5.2

[142] Lynda Khiali, Mamoudou Ndiath, Samuel Alleaume, Dino Ienco, Kenji Ose, and Maguelonne Teisseire, "Detection of spatio-temporal evolutions on multi-annual satellite image time series: A clustering based approach," *International Journal of Applied Earth Observation and Geoinformation*, vol. 74, pp. 103 – 119, 2019. 5.2, 5.3.1, 5.3.4, 5.3.4, 5.3.4, 5.3.5, 5.5.2, 6.2, 6.5.2, C.2, D.4, D.5

[143] Fabio Guttler, Dino Ienco, Jordi Nin, Maguelonne Teisseire, and Pascal Poncelet, "A graph-based approach to detect spatiotemporal dynamics in satellite image time series," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 130, pp. 92 – 107, 2017. 5.2, 5.3.1, 5.3.4, 5.3.4, 5.3.4, 6.2, D.4

[144] Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, M Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh, "Addressing Big Data Time Series: Mining Trillions of Time Series Subsequences Under Dynamic Time Warping," 09 2013, vol. 7. 5.2, 5.5.2, 6.2

[145] David Blei, Andrew Ng, and Michael Jordan, "Latent Dirichlet Allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 05 2003. 5.2

[146] Yin Zhang, Rong Jin, and Zhi-Hua Zhou, "Understanding bag-of-words model: A statistical framework," *International Journal of Machine Learning and Cybernetics*, vol. 1, pp. 43–52, 12 2010. 5.2

[147] Pedro F. Felzenszwalb and Daniel P. Huttenlocher, "Efficient Graph-Based Image Segmentation," *International Journal of Computer Vision*, vol. 59, no. 2, pp. 167–181, Sep 2004. 5.3.3, 5.5.1, D.4

[148] Lutz Prechelt, *Early Stopping — But When?*, pp. 53–67, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. 2

[149] Ilya Sutskever, Oriol Vinyals, and Quoc Le, "Sequence to Sequence Learning with Neural Networks," *Advances in Neural Information Processing Systems*, vol. 4, 09 2014. 5.3.6

[150] Romain Tavenard, Johann Faouzi, Gilles Vandewiele, Felix Divo, Guillaume Androz, Chester Holtz, Marie Payne, Roman Yurchak, Marc Rußwurm, Kushal Kolar, and Eli Woods, "tslearn: A machine learning toolkit dedicated to time-series data," 2017, `https://github.com/rtavenar/tslearn`. 5.5, 6.5

[151] Stefano Nativi, Paolo Mazzetti, Mattia Santoro, Fabrizio Papeschi, Max Craglia, and Osamu Ochiai, "Big Data challenges in building the Global Earth Observation System of Systems," *Environmental Modelling & Software*, vol. 68, pp. 1 – 26, 2015. 6.1

[152] Claudia Kuenzer, Stefan Dech, and Wolfgang Wagner, vol. 22, 06 2015. 6.1

[153] Cristina G'ømez, Joanne C. White, and Michael A. Wulder, "Optical remotely sensed time series data for land cover classification: A review," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 116, pp. 55 – 72, 2016. 6.1

[154] François Petitjean, Jordi Inglada, and Pierre Gancarski, "Clustering of satellite image time series under Time Warping," 08 2011, pp. 69 – 72. 6.2, D.5

[155] Zheng Zhang, Ping Tang, Lianzhi Huo, and Zengguang Zhou, "MODIS NDVI time series clustering under dynamic time warping," *International Journal of Wavelets, Multiresolution and Information Processing*, vol. 12, no. 05, pp. 1461011, 2014. 6.2, C.2, D.5

[156] Mariana Belgiu and Ovidiu Csillik, "Sentinel-2 cropland mapping using pixel-based and object-based time-weighted dynamic time warping analysis," *Remote Sensing of Environment*, vol. 204, 11 2017. 6.2, D.5

[157] Ovidiu Csillik, Mariana Belgiu, Gregory Asner, and Maggi Kelly, "Object-Based Time-Constrained Dynamic Time Warping Classification of Crops Using Sentinel-2," *Remote Sensing*, vol. 11, pp. 1257, 05 2019. 6.2

[158] François Petitjean and Jonathan Weber, "Efficient Satellite Image Time Series Analysis Under Time Warping," *Geoscience and Remote Sensing Letters, IEEE*, vol. 11, pp. 1143–1147, 06 2014. 6.2, D.5

[159] François Petitjean, Camille Kurtz, Nicolas Passat, and Pierre Gancarski, "Spatio-Temporal Reasoning for the Classification of Satellite Image Time Series," *Pattern Recognition Letters*, vol. 33, pp. 1805–, 03 2013. 6.2

[160] Wanderson Costa, Leila Fonseca, Thales Körting, Margareth Simoes, and Patrick Kuchler, "A Case Study for a Multitemporal Segmentation Approach in Optical Remote Sensing Images," in *10th International Conference on Advanced Geographic Information Systems, Applications, and Services, GEOProcessing 2018*, 04 2018. 6.2

[161] Shunping Ji, Zhang Chi, Anjian Xu, and Yulin Duan, "3D Convolutional Neural Networks for Crop Classification with Multi-Temporal Remote Sensing Images," *Remote Sensing*, vol. 10, pp. 75, 01 2018. 6.2

[162] Ying Li, Haokui Zhang, and Qiang Shen, "Spectral-Spatial Classification of Hyperspectral Imagery with 3D Convolutional Neural Network," *Remote Sensing*, vol. 9, no. 1, pp. 67, Jan 2017. 6.2

[163] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun Woo, "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, Cambridge, MA, USA, 2015, NIPS'15, pp. 802–810, MIT Press. 6.2

[164] Peng Liang, Wenzhong Shi, and Zhang Xiaokang, "Remote Sensing Image Classification Based on Stacked Denoising Autoencoder," *Remote Sensing*, vol. 10, pp. 16, 12 2017. 6.2

[165] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning Spatiotemporal Features with 3D Convolutional Networks," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 4489–4497. 6.3.1

[166] K. Fukunaga and L. Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition," *IEEE Transactions on Information Theory*, vol. 21, no. 1, pp. 32–40, January 1975. 6.3.2, D.5

[167] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, no. 1, pp. 43–49, 1978. 6.5.2

[168] Li Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 594–611, 2006. 7.3, D.6.2

[169] Malay Kumar Pakhira, Sanghamitra Bandyopadhyay, and Ujjwal Maulik, "Validity index for crisp and fuzzy clusters," *Pattern Recognition*, vol. 37, no. 3, pp. 487–501, 2004. B

[170] J. C. Dunn, "A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters," *Journal of Cybernetics*, vol. 3, no. 3, pp. 32–57, Jan. 1973. B

[171] Cédric Wemmert, *Classification hybride distribuée par collaboration de méthodes non supervisées*, Ph.D. thesis, The University of Strasbourg, December 2000. B

[172] Bo-Cai Gao, "NDWI : A normalized difference water index for remote sensing of vegetation liquid water from space," 1996. C.2

[173] Ning Zhang, Yang Hong, Qimin Qin, and Lu Liu, "VSDI: A visible and shortwave infrared drought index for monitoring soil and vegetation moisture based on optical remote sensing," *International Journal of Remote Sensing*, vol. 34, pp. 4585–4609, 07 2013. C.2

[174] María José López García and V. Caselles, "Mapping burns and natural reforestation using Thematic Mapper data," *Geocarto International*, vol. 6, pp. 31–37, 03 1991. C.2

[175] A.R Huete, "A soil-adjusted vegetation index (SAVI)," *Remote Sensing of Environment*, vol. 25, no. 3, pp. 295 – 309, 1988. C.2

[176] Sarah Harris, Sander Veraverbeke, and Simon Hook, "Evaluating Spectral Indices for Assessing Fire Severity in Chaparral Ecosystems (Southern California) Using MODIS/ASTER (MASTER) Airborne Simulator Data," *Remote Sensing*, vol. 3, no. 11, pp. 2403–2419, Nov 2011. C.2

[177] Stuart McFeeters, "Using the Normalized Difference Water Index (NDWI) within a Geographic Information System to Detect Swimming Pools for Mosquito Abatement: A Practical Approach," *Remote Sensing, vol. 5, issue 7, pp. 3544-3561*, vol. 5, pp. 3544–3561, 07 2013. C.2