UNIVERSITY OF LUXEMBOURG
MASTER THESIS

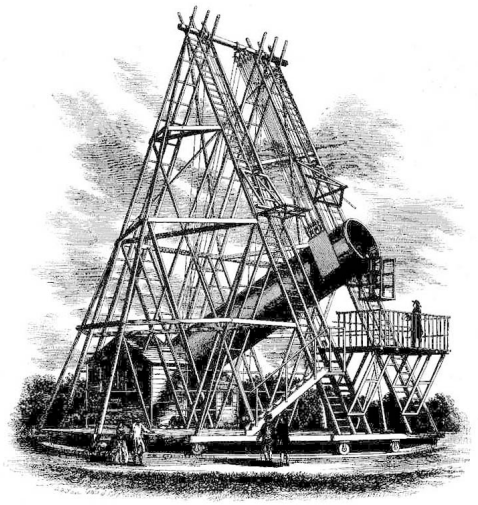
Modeling and Predicting the Resilience of Ecosystems

Author:
Tiago Alexandre DE JESUS SOUSA

Supervisor:
Prof. Dr. Nicolas GUELFI

Co-Supervisor:
Dr. Benoît RIES

Examiner:
Prof. Dr. Miguel OLIVARES-MENDEZ



A thesis submitted in partial fulfillment of the requirements for the Interdisciplinary Space Master degree

Laboratory for Advanced Software Systems
Department of Computer Science

August 22, 2022

Declaration of Authorship

I, Tiago Alexandre DE JESUS SOUSA, declare that this thesis titled, "Modeling and Predicting the
Resilience of Ecosystems" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"It is important to doubt and that the doubt is not a fearful thing, but a thing of great value."

Richard P. Feynman

UNIVERSITY OF LUXEMBOURG

Abstract

Faculty of Science, Technology and Medicine (FSTM)

Department of Computer Science

Interdisciplinary Space Master

Modeling and Predicting the Resilience of Ecosystems

by Tiago Alexandre DE JESUS SOUSA

The study and monitoring of natural ecosystems are among the most critical challenges of the current century in the context of climate change. The advancements in Earth Observation programs, either remote sensing or in-situ, have allowed us to assess, understand and monitor the natural ecosystems and man-made environments, in addition to improve our understanding related to the impact of human activities on the resilience of Earth ecosystems. In turn, those Earth Observation programs generate a significant amount of homogeneous, multi-format and multiple source data, fundamental for the analysis of ecosystem resilience and other Earth Observation applications. The intent of this thesis was to create an approach to develop systems analysing the resilience of ecosystems using Model-Driven Engineering (MDE) and Artificial Intelligence (AI). Foremost, in an MDE context, our proposed approach allows to model the requirements for ecosystem resilience analysis, supported with a dedicated metamodel. This requirements model is used consequently to generate skeleton applications for the collection and optimisation of Earth Observation data as well as artificial neural networks for the information extraction for the specified properties of interest in a given ecosystem. Then, a final step is the analysis of the computed values to conclude on the resilience of the ecosystem with respect to the properties under study. Finally, in this thesis, we illustrate our proposed approach with a case study focusing on the field of AI for social good with a case study targeting a system capable of analysing the resilience of the Luxembourg territory with respect to the Paris agreements on the reduction of greenhouse gases emissions.

Acknowledgements

First and foremost I would like to express the deepest appreciation to my supervisors, Professor Nicolas Guelfi and Dr Benoît Ries not only for making this thesis possible, but also for their guidance, wisdom and for believing and trusting in me throughout all those years. Their immense knowledge and plentiful experience have encouraged me in all the time of my studies and daily life.

I would also like to thank Professor Miguel A. Olivares-Mendez for accepting to be a reviewer of this thesis and for supporting me during the Interdisciplinary Space Master.

Additionally, I would like to thank my friend Gilles for the cherished time spent talking about computer science and research but also for all the help throughout the last years.

Finally, I would like to express my gratitude to my parents, brothers, sisters, Melissa and friends. Without their tremendous understanding and encouragement in the past few months, it would be extremely hard for me to complete my thesis. Thank you for the words that I needed to hear during the hard times those last months.

Contents

List of Figures

List of Tables

Acronyms

AI Artificial Intelligence

ANN Artificial Neural Network

CNN Convolutional Neural Network

DL Deep Learning

DNN Deep Neural Network

DREF Formal Framework for Dependability and Resilience from a Software Engineering Perspective

EO Earth observation

ERA *Ecosystem Resilience Analysis*

GAN Generative Adversarial Network

MDE Model-Driven Engineering

ML Machine Learning

NIR Near Infrared

OMG Object Management Group

PCA Principal Component Analysis

SDLC Software Development Life Cycle

SE Software Engineering

SWIR Shortwave Infrared

UML Unified Modeling Language

Para os meus pais que lutaram ao longo de todos estes anos para eu poder ter a melhor educação possível e uma vida melhor.

Para os meus irmãos Jorge, Daniel, André e Filipe assim como as minhas irmãs, Lara e Sofia, que sempre acreditaram em mim e fizeram de mim alguém melhor.

Para a Melissa, pelas palavras luminosas nos momentos mais sombrios e pelo amor e dedicação ao longo de todos estes anos.

Obrigado a todos vocês pelo apoio, dedicação e amor infinito.

♡

Chapter 1

Introduction

Fifty years ago, the United Nations Conference on the Human Environment in Stockholm raised awareness on the impact of human activities on Earth's ecosystems. Since then, the environment has become a pressing global issue for the first time, and multiple efforts have been made to ensure the sustainability of human life on Earth.

Among the most important efforts, the advancements in Earth observation (EO) remote sensing programs such as Landsat or Sentinel have allowed us to assess, understand and monitor the natural ecosystems and man-made environments, in addition to improve our understanding related to the physical characteristics of Earth.

Moreover, in the last decades, satellites have proven to be effective to monitor the impact of human activities and assess the status and evolution of resilience in natural ecosystems. Human activities are among the most critical threats to the functioning of natural ecosystems. Anthropogenic climate change, biodiversity reduction and exploitation of natural resources, are among others, the most significant threats to critical changes in the resilience of ecosystems. Rising greenhouse gas emissions have triggered climate change and put the resilience of natural ecosystems and the human societies that depend upon them at risk at an unprecedented rate.

Hence, the continuous research and development in Earth observation technologies, such as remote sensing or in-situ, to monitor the changes on Earth as well as the impact of human activities on the resilience of ecosystems have lead to the generation of significant volumes of data concerning our planet.

Due to the advancements in Earth observation technologies and instruments along with the exponential demand for Earth observation data, the complexity of the generated data increased. Thus, such increase in Earth observation data complexity has created a need for tools and methods to efficiently collect and analyse the complex data. Therefore, the development of software systems for the collection and processing of large volumes of complex data and the usage of artificial intelligence methods to analyse and extract information have become indispensable in the domain of Earth observation.

1.1 Problem Statement

Despite significant advancements in Earth observation technologies and instruments, either remote sensing satellites or in-situ, it remains complex to analyse the resilience of ecosystems or to undertake Earth observation in general.

Typically, in order to analyse and study the resilience of ecosystems with Earth observation data, multiple activities are required. The activities involve different development processes such as the specification of the dataset requirements (remote sensing or in-situ), the definition of the ecosystem to study, the entity to evaluate on the ecosystem, as well as the property of interest to observe to infer the resilience of the ecosystem. Moreover, to extract resilience information from Earth observation multi-sources, which may have multi-dimensional spatial and temporal structures evolving through time, the application of artificial intelligence algorithms becomes essential, creating an additional constraint for the *ecosystem resilience analysis*.

Accordingly, the complexity of analysing the resilience of natural ecosystems with Earth observation technologies and instruments is mainly attributable to the high number of Earth observation data sources and high volumes of generated data for different regions of the globe.

Therefore, in this context, the application of software engineering development lifecycle principles, model-driven engineering techniques, and artificial intelligence algorithms can help reduce the complexity associated with Earth Observation for the ecosystem resilience analysis.

Thus, to decrease the complexity, the usage of modeling approaches in model-driven engineering comes in as an adequate solution for the specification, design and development of a structured software system to undertake ecosystem resilience analysis using Earth observation data. Additionally, with the recent advances in artificial intelligence techniques, more specifically in deep learning, multiple algorithms can be applied to extract information from high volumes of data among different Earth observation data sources.

In this context, our research question is: *what could be a model-driven software engineering method for the development of systems analysing the resilience of ecosystems using artificial intelligence and earth observation data?*

1.2 Objectives

The principal objectives related to the subject of this thesis were the following:

- Study the literature on the modeling of ecosystems in a software engineering and model-driven engineering context. Additionally, study the literature on remote sensing and artificial intelligence for Earth observation.
- Proposition of an approach to develop software systems analysing the resilience of ecosystems using model-driven engineering and artificial intelligence.
- Definition of a case study on a system analysing the resilience of the Luxembourg territory with respect to the recent Paris agreements on the reduction of greenhouse gas emissions.

1.3 Outline

We finish this introduction with an outline of the following chapters of this thesis.

In the second chapter, we present the relevant background and related work with respect to our contribution. We present the concepts related to ecosystems, remote sensing, modeling and artificial intelligence. Due to the interdisciplinary context of this thesis, some concepts are introduced at a greater extent compared to others. Thus, concepts related to remote sensing and artificial intelligence are more detailed.

We introduce in the third chapter our proposed approach for *ecosystem resilience analysis*. Thus, we present in detail the different activities of our approach, namely the requirements specification, the intelligent observers generation and the final analysis activity.

Once we finish the presentation of our approach, we present in the fourth chapter the case study on a system analysing the resilience of the Luxembourg territory with respect to the recent Paris agreements on the reduction of greenhouse gas emissions. The three different activities of our proposed approach are presented in the approach. Hence, we present the requirements specification for the case study as well as an overview of the collected and optimised Earth observation datasets (remote sensing and in-situ). In addition, we present in detail the artificial intelligence architecture designed to extract information for the ecosystem resilience analysis with a remote sensing dataset. Finally, we present the *ecosystem resilience analysis* results of the case study.

We end the thesis with a conclusion in the fifth chapter and a presentation of the future work in the sixth chapter.

Chapter 2

Background & Related Work

In this chapter, we present the background and the related work of our contribution for the *Ecosystem Resilience Analysis* approach. We start by introducing and defining the different concepts used to produce our contribution, as well as how the concepts are used throughout this thesis. Additionally, at the end of each concept section, we present the associated related work.

2.1 Ecosystem

Based on [1], an *ecosystem* can be defined as a biological system composed of all the organisms (plants, animals, fungi, et cetera) found in a particular physical environment (air, water, soil) within an Earth-System context, interacting with it and each other. Since ecosystems are defined within or between a network of interacting organisms, an ecosystem can be of any size and composed of sub-ecosystems.

Ecosystems, as dynamic entities, are controlled and impacted by external and internal factors. External factors, as for example climate, influence the general structure of an ecosystem and how organisms behave. Internal factors such as disturbance, succession or decomposition influence the availability of resources within an ecosystem. Thus, internal factors are responsible for the resources required for organisms' growth, reproduction and maintenance.

With repeated variations in ecosystems caused by negative disturbances from both the external and internal factors, the ability of ecosystems to regenerate and create essential ecosystem processes to maintain an equilibrium can be impacted.

To measure variations in ecosystems, two parameters are fundamental:

- **Resistance**: Ability of an ecosystem to hold the general structure and processes in equilibrium when exposed to disturbances.
- ***Resilience***: Ability to recover to equilibrium as a response to disturbances and perturbations.

2.1.1 Resilience

First introduced in [2] to describe the recover of natural systems as a response to natural or human impacts, *resilience* has been defined in ecological literature as:

1. The required time for an ecosystem to return to an equilibrium or steady-state following a perturbation [3].
2. "The capacity of a system to absorb disturbance and reorganise while undergoing change so as to still retain essentially the same function, structure, identity, and feedbacks" [4]

Among the different sources of disturbances, human factors are one of the most important [5]. Human factors affect ecosystems with pollution, deforestation or with the burning of fossil fuels, releasing carbon dioxide (CO_2) into the atmosphere and changing the balance of the climate system. With climate change, new disturbances are created, progressively deteriorating ecosystems.

In case of important disturbances in a given ecosystem, the equilibrium can be impacted to such a degree that the ecosystem may completely lose its *resilience*.

Recently, with the advancements of studies regarding human-induced climate change, *resilience* has been defined in the "Sixth Assessment Report" [6] of the United Nations Intergovernmental Panel on Climate Change (IPCC) as "*not just the ability to maintain essential function, identity and structure, but also the capacity for transformation*".

Therefore, it is fundamental to create approaches to study and monitor ecosystem *resilience*. Furthermore, with rapid environmental changes, such approaches shall allow the analysis of how disturbances affect the essential ecosystem processes.

2.1.2 DREF Framework

In this thesis, we propose an approach to develop systems for *Ecosystem Resilience Analysis* (*ERA*) using model-driven engineering coupled with artificial Intelligence. We focus on the concept of *ecosystem resilience* that we define as "a property of an ecosystem that is considered to improve capabilities over time with respect to some given properties of interest", based on the Formal Framework for Dependability and Resilience from a Software Engineering Perspective (DREF) [7]. Despite the fact that it is a framework associated with the software engineering domain, its abstract and generic terminology defined mathematically allow us to use it in this thesis.

Therefore, in this section, we describe the DREF concepts that are used throughout this thesis. In DREF, the most basic concepts are the *entities* and the *properties*. *Entities* are defined as "anything that is of interest to be considered" while *properties* are defined as the basic concepts that are used to characterise entities such as an informal requirement or a mathematical property. In the context of our thesis, we can

restrict our focus on any entities that are ecosystems while the concept of properties is restricted to characterise ecosystem components (biotic or abiotic).

Moreover, DREF proposes the concept of *satisfiability*, which is used to evaluate if a given *property* is satisfied by an entity, and the concept of *observer* is used to represent subjectivity for the evaluation of satisfiability. For each observer, there is an observation function which will give the satisfiability. Furthermore, the concept of *nominal satisfiability* is defined in DREF as a satisfiability function "used to represent the expected satisfiability". Since tolerance margins may be necessary to evaluate satisfiability, in DREF, an additional concept is introduced called *tolerance threshold*. Thus, *tolerance threshold* is defined as another satisfiability function, used to represent the lower bound specifying the "tolerance margin for the satisfiability functions w.r.t. a nominal satisfiability function". Finally, DREF introduces the concept of *evolution axis*, defined as a "set of values that are used to index a set of entities or a set of properties".

In the context of our thesis, the concepts of *satisfiability*, *nominal satisfiability*, *tolerance threshold* and *evolution axis* are used as proposed in DREF. However, the concept of *observer* is extended and it is used as an abstraction of a data source providing Earth observation data to evaluate if a given *property* is satisfied by an *entity* in a given period of time, as presented in Chapter 3.

2.2 Remote Sensing

The term *"remote sensing"*, first used in 1960, refers to the science of acquiring information about objects or areas from a distance. As defined in [8], remote sensing is a "field of study associated with extracting information about an object without coming into physical contact with it".

Remote sensing started with the invention of the camera in 1816 by Joseph Nicéphore Niépce, which lead to the development of photography in the following years. With photography, the process to acquire a permanent image was invented.

Soon after, the first aerial photograph of a French village was taken in 1858, inside a hot air balloon by Gaspard-Félix Tournachon, commonly known by his pseudonym "Nadar". With this picture, the era of remote sensing had started. In the following years, remote sensing images were captured mostly using cameras mounted in pigeons, kites and hot air baloons.

With the beginning of the First World War, remote sensing emerged as a key military asset, used for aerial surveillance and reconnaissance purposes with cameras mounted on airplanes. During the Second World War, further developments were made, producing new camera sensors and photo interpretation equipment. With such advancements, the field of remote sensing began to be used for other purposes,

such as geology, agriculture, forestry and cartography. Later on, with the launch of the Landsat-1 [9] satellite in 1972, the first satellite designed to observe the Earth, remote sensing has been integrated in satellites.

Since then, remote sensing has allowed to observe and detect changes in the most remote regions of the Earth. Nowadays, the term remote sensing refers to the usage of satellite and their sensors to observe the Earth.

In the context of *ecosystem resilience analysis*, remote sensing provides useful spatial and temporal Earth observation data to identify, study and monitor ecosystems (atmosphere, soil and oceans) and the evolution of their resilience.

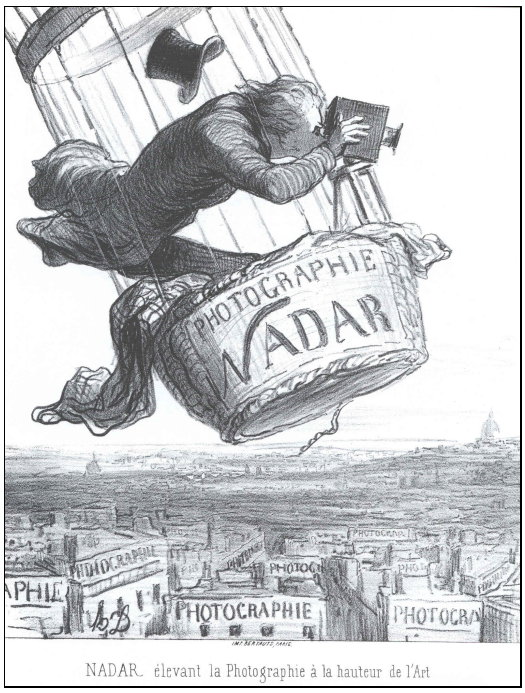


FIGURE 2.1: Caricature of aerial photography "*Nadar raising photography to the height of Art*", signed Honoré Daumier.

2.2.1 Principles

In order to observe the surface of the Earth, satellite sensors shall acquire information about the different objects on the surface without any physical contact with it. However, energy sources and radiation principles influence the ability of sensors to acquire informations and produce satellite imagery. In the following sections, we introduce the basic physics principles of remote sensing.

Electromagnetic Radiation

The basic principle of remote sensing is through electromagnetic radiation, where the Sun is the primary source. Electromagnetic radiation is one form of energy

propagation and consists of waves of the electromagnetic field propagating through space at the speed of light and being reflected by all objects.

An electromagnetic radiation is composed of a perpendicular electrical field amplitude (E) and a magnetic field amplitude (B), oscillating perpendicular to each other and traveling at a constant speed of 3×10^8m/s known as the speed of light (c), see Figure 2.2.

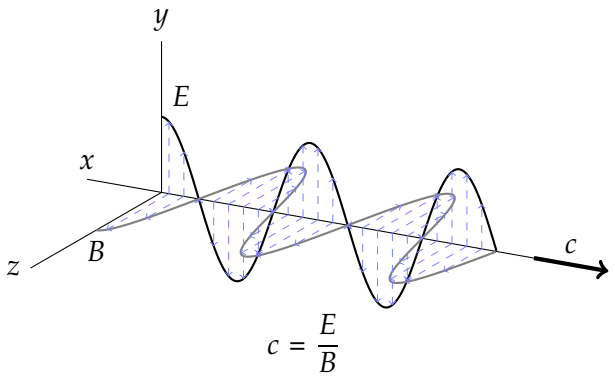


FIGURE 2.2: Illustration of an Electromagnetic Radiation.

Electromagnetic radiation has two important parameters, namely wavelength and frequency.

- **Wavelength**: Denoted as λ, and measured in a factor of meters, it refers to the length of a wave cycle, measuring the distance between two sequential crests.
- **Frequency**: Denoted as v and normally measured in hertz (Hz), it refers to the number of cycles of a wave passing a fixed point per unit time.

As shown in Equation 2.1, both parameters are related to each other, as the frequency of a wave is conversely proportional to its wavelength. As thus, the shorter the wavelength, the higher the frequency and vice-versa.

$$\lambda = \frac{c}{v} \tag{2.1}$$

The different intensities of the electromagnetic radiation wavelength and frequency, are separated into multiple ranges or regions in the electromagnetic spectrum, depicted in Figure 2.3.

The electromagnetic spectrum is divided into several ranges of electromagnetic radiation waves covering many orders of magnitude, from extremely small wavelengths known as *cosmic rays* to longer wavelengths such as *microwaves*.

Depending on a given satellite sensor, radiation energy is collected within different ranges of the electromagnetic spectrum. In remote sensing each collected range in the electromagnetic spectrum is referred to as a *"spectral band"*.

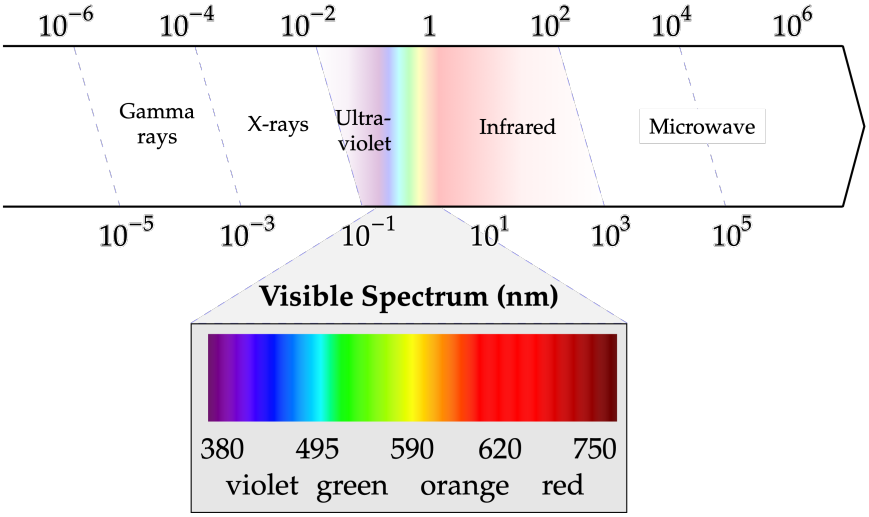


FIGURE 2.3: The Electromagnetic Spectrum.
The divisions between the ranges and the six colors are not perfectly distinct.

- **Gamma rays** have the smallest wavelengths and the most energy of any other wave in the electromagnetic spectrum.
- **X-rays** range in wavelength from 0.01 nm to 10 nm and are able to pass through many different types of materials.
- **Ultraviolet** range in wavelength from 1 nm up to 380 nm, which remains invisible to the human eye.
- **Visible spectrum** refers to the electromagnetic waves visible with the human eye. The visible spectrum region is divided into six wavelength intervals, representative of a particular color:
  - **Violet**: From 380 nm up to 450 nm.
  - **Blue**: From 450 nm up to 495 nm.
  - **Green**: From 495 nm up to 570 nm.
  - **Yellow**: From 570 nm up to 590 nm.
  - **Orange**: From 590 nm up to 620 nm.
  - **Red**: From 620 nm up to 750 nm.
- **Infrared** ranges from approximately 750 nm up to 0.01 mm in wavelength, divided into three main regions.
  - **Near Infrared (NIR)**: From 750 nm up to 140 nm.
  - **Shortwave Infrared (SWIR)**: From 140 nm to 3000 nm.
  - **Thermal Infrared**: From 3000 nm up to 0.01 mm.

- **Microwaves** ranges from approximately 1 cm up to 1 m and are mostly used for satellite communication due to their high frequency, in the context of remote sensing.

Atmospheric Influences

As mentioned before, the Sun is the primary source of electromagnetic radiation. However, before the electromagnetic radiation from the Sun reaches the surface of the Earth, some regions of the electromagnetic spectrum can be absorbed or scattered [10], and therefore only a subset of the presented electromagnetic radiation ranges are used for remote sensing.

Absorption refers to the loss of energy at various wavelengths due to molecules in the atmosphere. The electromagnetic radiation is absorbed in the Earth's atmosphere by gases, such as:

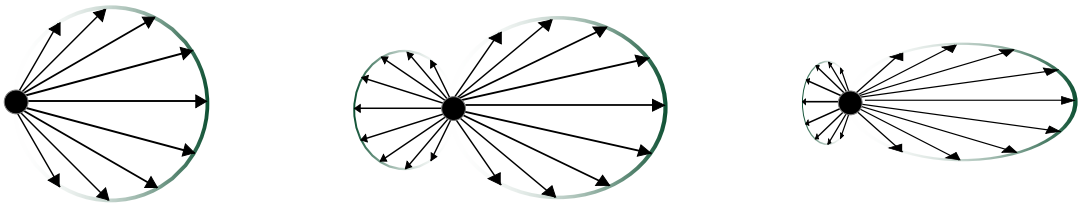
- **Carbon Dioxide (CO_2)** absorbs radiation in the thermal infrared region of the electromagnetic spectrum.
- **Ozone (O_3)** absorbs most of the radiation in the ultraviolet region of the electromagnetic spectrum.
- **Water Vapor (H_2O)** absorbs radiation from the thermal infrared and microwave ranges, depending on the location and time of the year.

In contrast to absorption, electromagnetic radiation can be scattered. Scattering refers to the redirection of the electromagnetic energy by particles such as aerosols in the atmosphere. There are three different types of scattering that impact electromagnetic radiation (see Figure 2.4).

- **Rayleigh Scatter** is a scattering caused when the particles and molecules in the atmosphere are smaller in diameter than the radiation wavelengths.
- **Mie Scatter** is a scattering caused when the particles in the atmosphere are similar in diameter compared with the radiation wavelengths. Examples of the major causes of mie scatter include water vapors, dust, pollen, among others.
- **Nonselective Scatter** is a scattering caused when the particles in the atmosphere are larger in diameter compared with the radiation wavelengths. For instance, water droplets in clouds are an example of nonselective scatter.

Surface Reflectance

When the electromagnetic radiation is not absorbed or scattered due to atmospheric influences, three possible forms of energy interactions can occur when the radiation reach and interact with the Earth's surface.



Rayleigh Scatter Mie Scatter Nonselective Scatter

FIGURE 2.4: Types of Electromagnetic Scattering.

Therefore, depending on the wavelength, material or the characteristics of the Earth's surface, the following interactions can take form:

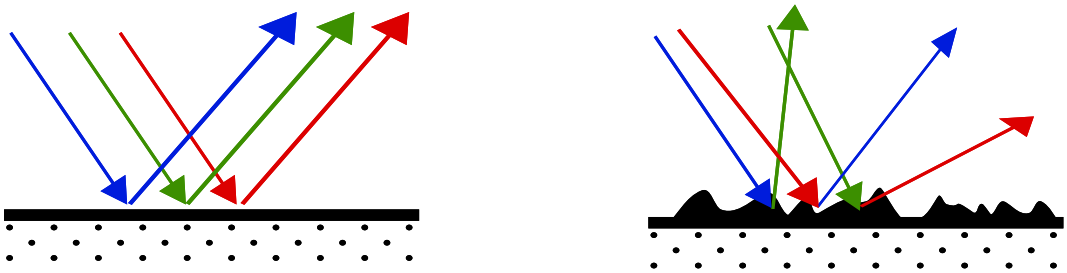
- **Absorption** occurs when the radiation energy is absorbed into the target.
- **Reflection** occurs when the radiation energy rebounces off the target, redirecting it.
- **Transmission** occurs when the radiation energy goes through the target.

In remote sensing, the most interesting energy interaction is reflection as it allows to obtain property informations related to the reflected surface target and therefore, distinguish between the different surfaces.

Additionally, reflection can be categorised in two different types (see Figure 2.5), each one representing how the energy is reflected from a target:

(A) **Specular Reflection** refers to the "mirror-like" reflection when the target surface is smooth, directing the wavelength energy in a single direction. Specular reflection occurs for example on lake surfaces with calm bodies of water.

(B) **Diffuse Reflection** refers to the scattering of wavelength energy when reflected on rough surfaces. Thus, wavelength energy is reflected almost uniformly in all directions. Diffuse reflection occurs for example on trails.



(A) Specular Reflection. (B) Diffusive Reflection.

FIGURE 2.5: Types of Surface Reflection Interactions.

Therefore, with reflection, electromagnetic radiation energy can rebounce off the target in two different ways. Depending on the type of reflection, either specular or diffusive, *reflectance* will change. Reflectance is defined as the percentage of incoming wavelength energy, reflected from the surface. As thus, depending on the reflectance

characteristics of the target, a satellite sensor can distinguish the surface and therefore, obtain surface property informations and collect EO data.

Figure 2.6 shows the typical reflectance signature curves for different types of target surfaces, such as grass, snow, dry soil and water.

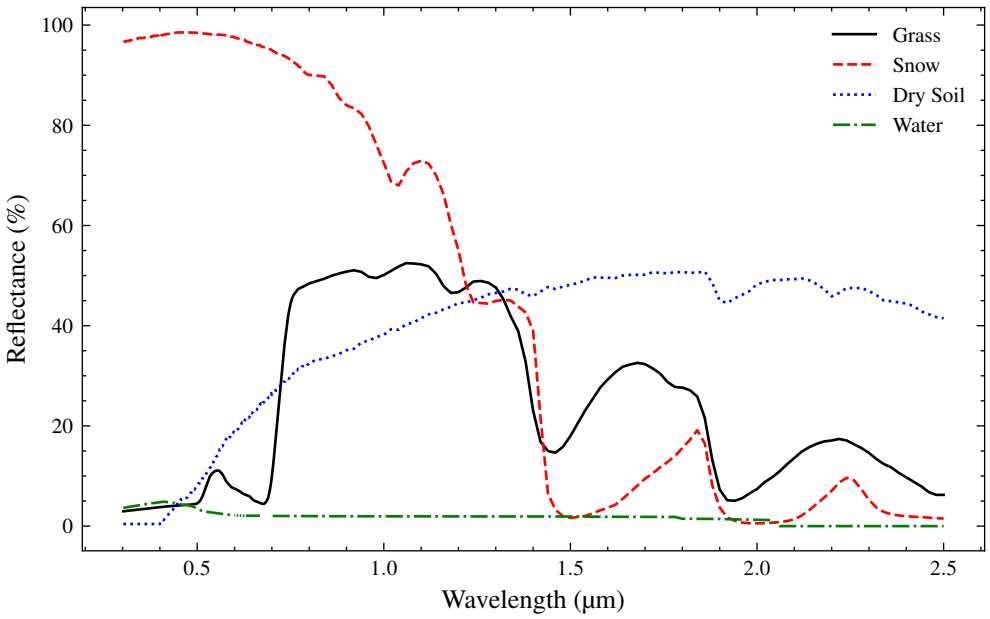


FIGURE 2.6: Spectral reflectance signature curves for grass, snow, dry soil and water surfaces. Data from the USGS Spectral Library [11].

2.2.2 Sensors

In order to collect data from Earth processes and create satellite imagery, satellite sensors require electromagnetic radiation to be reflected. Therefore, satellite sensors are differentiated in two types, namely active and passive (depicted in Figure 2.7).

Active sensors emit their own pulse of energy towards the target of interest. In turn, the sensor measures and detects the changes in the reflected radiation signal. The advantage of active sensors is that remote sensing data can be collected during the day or night, since active sensors are not dependent on electromagnetic radiation sources such as the Sun. The disadvantage concerns the amount of energy to generate on orbit to be able to "illuminate" targets.

Passive sensors do not emit their own pulse of energy. Instead, passive sensors rely on both sunlight radiation that has been reflected from the Earth and thermal radiation in the visible and infrared regions of the electromagnetic spectrum (see Figure 2.3). Therefore, passive sensors can only measure and detect reflected radiation signals when external energy is available. In other terms, passive sensors are dependent of the Sun, illuminating the Earth.

(A) Active Sensors. (B) Passive Sensors.

FIGURE 2.7: Types of Remote Sensing Sensors.

Resolution

To distinguish between the different satellite sensors, either active or passive, resolutions are important characteristics as they influence the representation of the reflected or emitted energy into imagery data. Therefore, imagery data captured with satellite sensors are characterised with four types of resolution.

Spatial resolution is defined as the size of pixels in an image representing the size of the Earth's reflected surface. For instance, if a satellite sensor has a spatial resolution of 200 meters, one pixel represents a surface area of 200 meters × 200 meters on Earth. Thus, the higher the resolution, a larger level of details can be observed on the Earth's reflected surface.

Nowadays, the spatial resolutions of satellite sensors vary from:

- High resolution: spatial resolution smaller than 2 meters.
- Medium resolution: spatial resolution between 2 meters and up to 30 meters.
- Low resolution: spatial resolution larger than 30 meters.

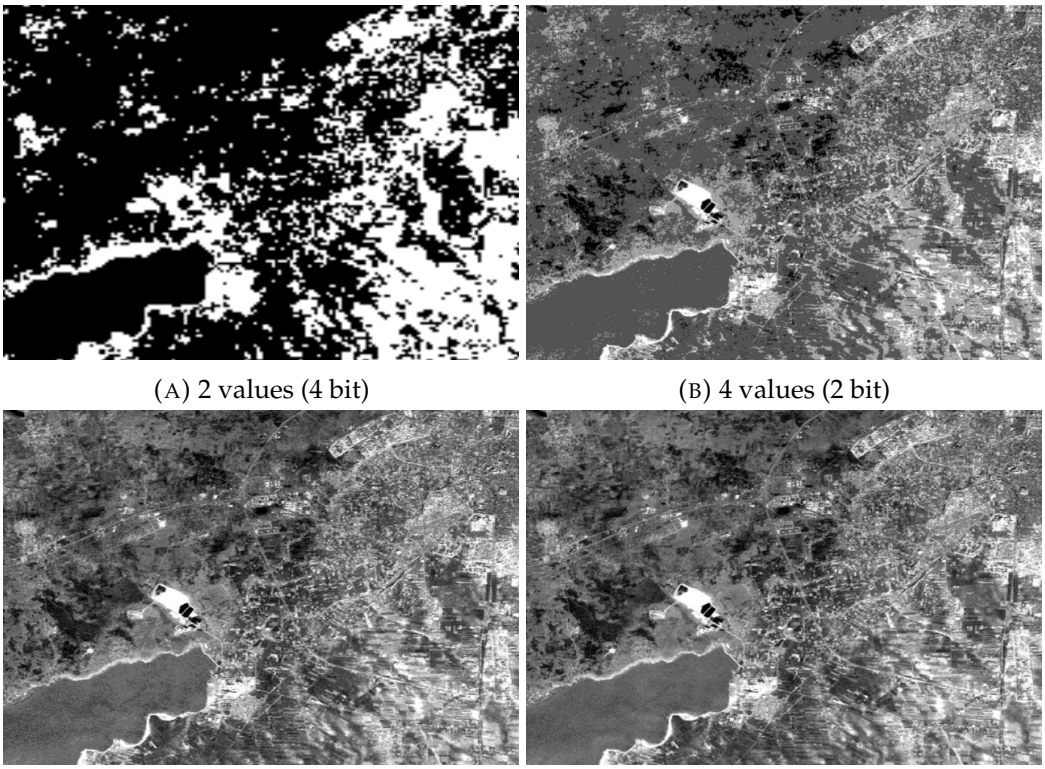
Spectral resolution is defined as the ability of the satellite sensor to measure and record the reflectance of a target on different wavelengths. For instance, if the the spectral resolution is low, the sensor covers more regions of the spectrum, and therefore, more details are obtained.

Temporal resolution is defined as the interval of time (often reported in days) between the collection of data by the satellite sensor over the same Earth region. Therefore, the higher the temporal resolution, the higher is the amount of satellite imagery in a small interval of time. Temporal resolution is one of the most important characteristics of satellite sensors as it allows to monitor, analyse and detect subtle changes on the same area of the Earth's surface.

Radiometric resolution is defined as the ability of the sensor to discriminate differences in reflected or emitted energy. Therefore, the higher the radiometric resolution value, the more variation in reflection or emitted energy is captured, producing

more details in satellite imagery. The details are expressed in levels of brightness with respect to the bit-depth of the satellite sensor.

An example of different levels of radiometric resolution is shown in Figure 2.8.



(A) 2 values (4 bit) (B) 4 values (2 bit)

(C) 8 values (3 bit) (D) 16 values (2 bit)

FIGURE 2.8: Differences in Radiometric Resolution.

2.2.3 Remote Sensing for Ecosystem Observation

With hundreds of Earth observation satellites currently in orbit, remote sensing has provided significant data sources for the observation and understanding of ecosystems and their constant evolution. Since the launch of Landsat in 1972, the Era of remote sensing has driven forward our understanding of Earth-system processes.

The advancements in the last 50 years of Earth observation satellite sensors in terms of spatial, temporal, spectral and radiometric resolutions has enabled the observation, monitoring and classification in space and time dimensions of terrestrial, aquatic and climatic environments. Examples include the study and assessment of species diversity and their conservation [12], the observation and estimation of forest biomass [13] and carbon monitoring [14], among others.

Additionally, in the context of understanding climate change, the scientific, societal and economic interests on remote sensing for ecosystem observation has transformed the investigation of climate processes [15] as well as the study and assessment of human activities and their increasing impact on Earth [16].

However, despite the advancements in remote sensing, there are remaining challenges for ecosystem observation such as development of software for ecosystem observation and monitoring, gathering of remote sensing data, data analysis and scale issues, as presented in [17]. In Chapter 3, we present a software engineering approach which tackles some of the current challenges with remote sensing for ecosystem observation.

2.2.4 Related Work

In this section, we present a few of the related works done in the domain of remote sensing for ecosystem observation.

Cohen et al. present in [18] the role of remote sensing, more specifically with the Landsat satellites, to acquire Earth observation data for ecological investigations and applications. Their summary on the role of Landsat for ecosystem observation show the application of the 30 year old record of spatial and temporal remote sensing data to identify and monitor land cover, to study the biophysical attributes of vegetation as well as to analyse the fragmentation and structure of forests with respect to biodiversity. Moreover, the authors describe the different case studies of Landsat data to study both the state and the dynamics of ecosystems. Examples of the presented case studies are the study of temporal ecosystem dynamics, evaluation of vegetation spatial patterns and mapping of large areas for ecology monitoring.

Goetz et al. present in [19] the advances in remote sensing and their implications for the monitoring and measuring of carbon stocks. In particular, the authors focus on the application of remote sensing, specifically with light detection and ranging sensors (LiDAR) for the monitoring of biomass in order to create a baseline estimation for carbon stocks, which could be used to improve the estimation of other terrestrial sensors measuring carbon changes. Likewise, the authors described the complexity of using LiDAR sensors for accurate carbon measurements in areas covered by clouds. Since LiDAR transmits energy in the near-infrared region of the electromagnetic spectrum, atmospheric influences as presented in Section 2.2.1 can absorb or attenuate the energy transmitted.

Lausch et al. introduce in [20] the application of remote sensing for the understanding of forest health. To overcome the complexity associated with the terrestrial monitoring of forest health, the authors provided an overview on how remote sensing data can help with the repetitive and objective monitoring of forest health indicators on different spatial and temporal scales and with a rapid and cost-effective method. In an additional paper [21], Lausch et al. presented a the requirements for the creation of multi-source forest health monitoring, where the usage of the different remote sensing sensor resolution characteristics (presented in Section 2.2.2) is indicated as an important solution to reduce the gaps in data, which occur with terrestrial sensors.

2.3 Modeling

Throughout the history, modeling has been crucial for the development of society, as it allowed to create *simplified versions of something that is real*.

From early civilisations who used some forms of modeling to create small-sizes of architecture plans, to ancient Greek astronomers and cosmologists such as *Ptolemy*, who used modeling and mathematics to demonstrate their hypothesis regarding the motion of the celestial bodies [22].

Additionally, modeling has proven to be fundamental in engineering and sciences, as a way of coping with the reality through abstractions and simplifications. For instance, modeling is crucial in the field of computer sciences, where it is an essential part of the specification, design and development of a software system. According to the Object Management Group (OMG), "*modeling is the designing of software applications before coding*" [23].

Hence, in this section, we present the concepts related to modeling in a software engineering and model-driven engineering context.

2.3.1 Software Engineering

> *"The major cause of the software crisis is that the machines have become several orders of magnitude more powerful!" [24]*
>
> — Edsger W. Dijkstra, *Turing Award (1972)*

In the early days of computer science, the process of writing software was cumbersome and was not seen as a dedicated type of engineering. In the late 1960s, many software projects failed due to the difficulties to produce large, correct, efficient, predictable and verifiable computer programs. The difficulties, mainly due to the increased complexity of hardware and project requirements were a source of budget overruns and delays. Therefore, the increase of complexity in software development lead to the *"software crisis"* in 1968.

To address the *"software crisis"*, discussions took place at the NATO Software Engineering Conferences which lead to the creation of the Software Engineering field in 1969 to establish methodologies and best practices for the development of software [25].

As defined by Sommerville [26], SE is an engineering discipline covering all aspects of software production, from requirements analysis and definition to operation and maintenance. With the evolution of SE in the last 50 years, new techniques, processes and methodologies have emerged, creating the basis of today's SE.

Nowadays, SE is ubiquitous and has impacted human life, with software systems being the backbone of our technology. The type of software systems are diverse, and as thus, not every software process can be applied to develop each type of software system. Nevertheless, as presented in [26], despite the different software processes, the following four central SE activities must be present in every software process:

- **Specification**: A set of software functionalities and software constraints must be specified.
- **Development**: A software complying with the specification must be produced.
- **Validation**: A software must be validated to ensure the compliance with the specification.
- **Evolution** A software must mature to comply with specification updates.

These activities are just a subset of the possible activities that may be included in software processes but constitute the major activities in every Software Development Life Cycle (SDLC) approaches. A SDLC is a streamlined abstract representation of a software processes.

An example of a SDLC approach is the *Waterfall model* [27], depicted in Figure 2.9. In this approach, the software development process is sequential and the progress flows downwards through the different phases as a waterfall. The outcome of a phase represents the input for the succeeding phase sequentially.

The sequential phases of the *Waterfall model*, reflecting the central SE activities presented previously, are:

- **Requirements definition**: A complete and comprehensive description of the software functionalities and software constraints must be defined in a requirements document.
- **System and software design**: In this phase, the requirements specification from the preceding phase is studied and a solution to define the elements of a software system such as the architecture must be prepared to develop the actual software system.
- **Development and unit test**: The different components of the software system are produced and validated with respect to both the requirements specification and the solution design from the preceding phases.
- **Integration and system testing**: The different elements constituting the software system are integrated and evaluated to determine if the software production complies with the requirements.
- **Operation and maintenance**: Once the software system is delivered, update solutions must be distributed to correct failures or errors. Additionally, with

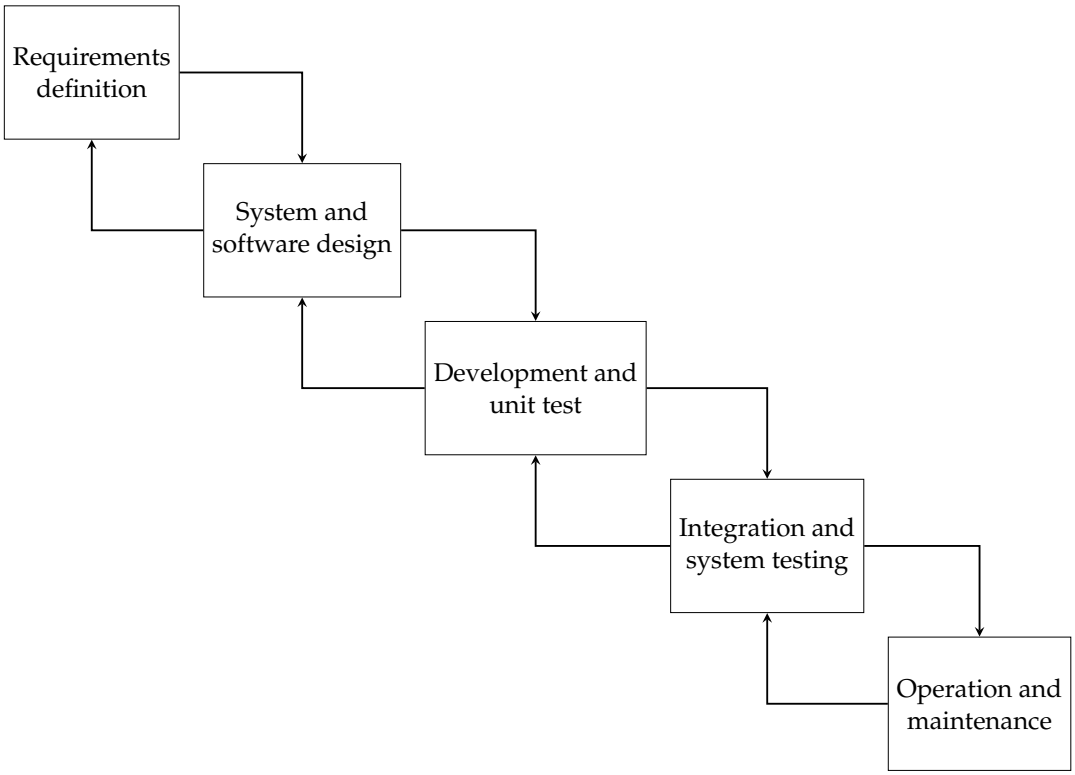


FIGURE 2.9: The Waterfall model.

the discovery of new system requirements, new adaptations must be introduced to enhance the software system.

In this thesis, a requirements definition approach for the *Ecosystem Resilience Analysis* is proposed in the context of the Software Development Life Cycle. Additionally, we propose an automation towards the SE development activity.

2.3.2 Model-Driven Engineering

With the purpose of reducing the complexity of SE development and maintenance, the Model-Driven Engineering (MDE) methodology [28] has emerged as a promising solution to leverage automation and abstraction on the SE development lifecycle. MDE aims to use engineering techniques where *models* are the first-class citizens to enable the development at higher levels of abstraction by employing concepts closer to the domain of problems, instead of using the concepts offered by programming languages.

The concept of a *model* in the context of SE has been defined as "an abstraction of a (real or language-based) system allowing predictions or inferences to be made" [29]. In other terms, a *model* is a representation of a system under study with the purpose of providing information about the represented system.

To leverage automation in software development, MDE promotes the concept of *model transformations* to transform *models* from one form to another. A *model transformation*, is a mapping function taking a source *model* as input and generating a target *model*, with respect to a set of transformation rules. Different model transformation languages have been proposed for the definition of transformation rules, such as `F-Alloy` [30] or the `QVT` family [31], standardised by the OMG.

However, to make automation a reality, *models* passed through *model transformations* must conform to a *metamodel*. The concept of *metamodel* is used to define a *model* of a *model*. A *metamodel* is an abstraction *model* defining the syntax and the semantics of a language with the purpose of specifying a modeling language to be expressed with the *models* at different levels of abstraction.

In general, *model transformations* can be classified in four different categories: a

- **Endogenous transformation**: the *metamodel* of the source and the target *model* are equal.
- **Exogenous transformations**: the *metamodel* of the source and target *model* are different.
- **In-place transformation**: the source *model* is copied to the target *model*.
- **Out-place transformation**: the target *model* is created from scratch.

In this thesis, we present a MDE approach for the modeling of requirements for *Ecosystem Resilience Analysis*, with a dedicated *metamodel* defining the syntax and the semantics to be followed with compliant models. Additionally, we propose the automated generation of software skeletons through *model transformations* for the collection and optimisation of datasets, as well as the computation of resilience values, compliant with *model* requirements.

2.3.3 Unified Modeling Language

With the the evolution of the SE field, multiple modeling methodologies and object-oriented programming languages began to appear. However, software engineers had difficulty finding a clear, visual and universal modeling language among many similar modelling methodologies.

As a result of an effort to unify distinct and leading modeling methodologies, to create a universal SE modeling language, the Unified Modeling Language (UML) [23] is created. Due to the important industry support, it became *de facto* an Object Management Group standard in 1997.

Based on three different modeling notations [32, 33, 34] for object-oriented software development, UML is created with four different goals:

1. Representation of complex software systems through modeling concepts.

2. Establish extensibility and specialisation to extend the core concepts.
3. Be independent from programming languages and software processes.
4. Provide a modeling language usable by humans as well as machines.

With the introduction of new extensions brought to the specification standard, the Unified Modeling Language became a semi-formal language allowing the specification, modeling, visualization and documentation of software system artifacts [23], with different types of diagram models at different levels of abstraction. Currently, UML supports fourteen diagram types divided into two different categories, namely the structural diagrams and the behavioral diagrams.

Structural diagrams depict the structure or the static view of a system. Thus, diagrams in this category are used to represent a general outline of a system, with respect to the structure of objects such as their attributes, relationships and operations, among others.

Behavioral diagrams depict the internal dynamic behaviour of a system, describing how the system interacts with itself and others.

- **Structural diagrams**:
  - Class diagram
  - Profile diagram
  - Component diagram
  - Deployement diagram
  - Object diagram
  - Package diagram
  - Composite structure diagram
- **Behavioral diagrams**:
  - Use case diagram
  - Activity diagram
  - State machine diagram
  - Interaction diagram:
    - Sequence diagram
    - Communication diagram
    - Interaction overview diagram
    - Timing diagram

Due to the scope of this thesis, we only present the *class diagram*. For an extended presentation of the different diagrams, please refer to the specification [23].

As the backbone of object-oriented software systems, the *class diagram* is one of the most used structural diagrams to represent the general outline of a system. A *class diagram* can be composed with the following (non-exhaustive) concepts:

Class: A class is an abstract representation of the type of an object, reflecting the structure and behaviour of the main elements within the system. It contains `attributes` and `operations` with different levels of `visibility` (`public (+)`, `protected (#)`, `private (−)` and `package (~)`).

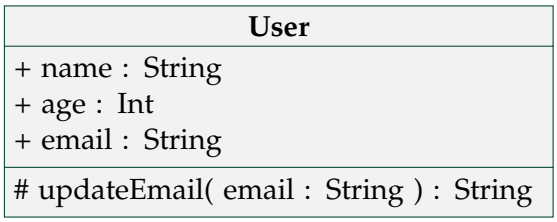


FIGURE 2.10: Example of a UML class.

Multiplicity: To indicate the number of class instances participating in the different relationships, the concept of multiplicity is used. Multiplicity is denoted together with relationships and the values can be expressed as:

- Zero or one, with the `[0..1]` notation.
- Exactly one, with the `[1]` notation.
- One or more, with the `[1..*]` notation.
- Many, with the `[0..*]` notation.
- Exact number, expressed for example with `[2]`, denoting that exactly two class objects participate in the relationship.

Relationship: Relationships are used to represent the relationship between the different classes in a class diagram. There exists five different relationship types:

- **Inheritance / Generalisation**: Refers to a *"is-a"* relationship, where a *"child"* class inherits its type from a *"parent"* class.

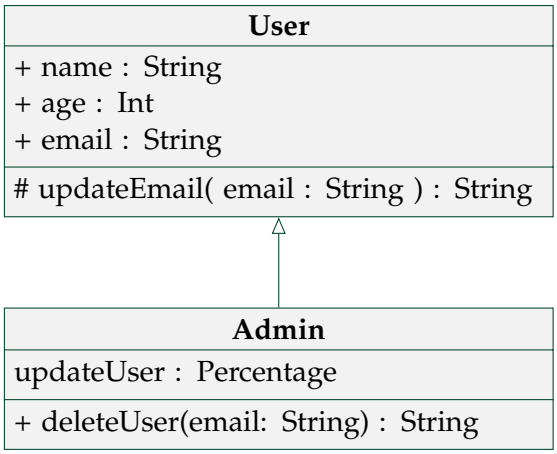


FIGURE 2.11: Example of UML class inheritance / generalisation.

- **Association**: Refers to a logical connection between two classes.
- **Aggregation**: Refers to a *"part of"* relationship, where a class is the result of being aggregated with another class, creating therefore a more complex object. It is a special type of association.

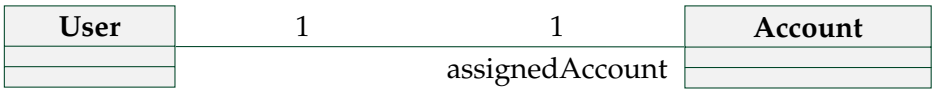


FIGURE 2.12: Example of UML class association.

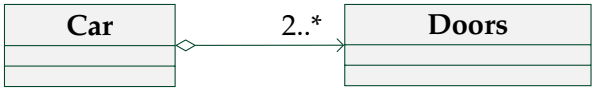


FIGURE 2.13: Example of UML class aggregation.

- **Composition**: Refers to a type of aggregation relationship with the difference being that the aggregated class cannot exist without the aggregator class.

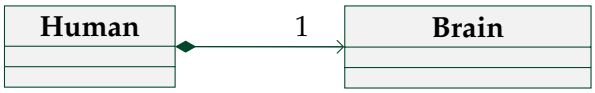


FIGURE 2.14: Example of UML class composition.

- **Dependency**: Refers to the dependency of a class with respect to another class.

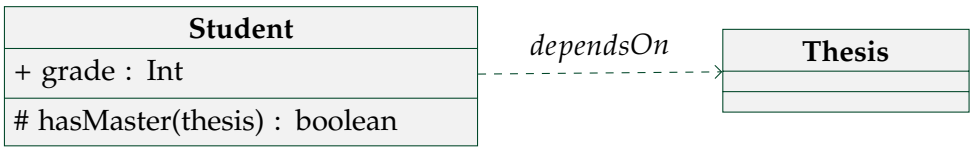


FIGURE 2.15: Example of UML class dependency.

Enumeration: An enumeration is a data type with instances representing user-defined data types.

DataType: A datatype is similar to a class data type, with the difference being that it cannot own sub DataType instances. It is used to define and represent value types in a respective domain.

In this thesis, we focus specifically on UML class diagrams for the abstract modeling of requirements for *Ecosystem Resilience Analysis* with a dedicated *metamodel* defining the domain model for the proposed software system approach. Moreover, UML class diagrams are once again used for the representation of the *metamodel* with a *model* instance.

2.3.4 Ecosystem Modeling

In the context of ecosystems, modeling has shown to be an important tool in the study of ecological systems with the construction and analysis of mathematical models for ecological processes. Modeling approaches allows the creation of analytical

and simulation systems with different levels of abstractions to study and understand ecological processes with the intent of predicting the dynamics of real ecosystems. Additionally, modeling creates opportunities to study and understand ecological processes that would be costly or impractical to perform on a real ecosystem due to "logistical, political, or financial reasons" [35].

Another advantage of modeling resides in the fact that a simulation of an ecological process that would take centuries in reality can be done almost instantaneous with the help of computers.

Due to the "ability" of models to explain phenomena and make predictions, multiple applications of ecosystem modeling have been found in fields such as agriculture [36], natural resource management [37] or environmental health risk assessment [38].

2.3.5 Related Work

In this section, we present the related work in the domain of ecosystem modeling in the context of software engineering. Based on our research, only a few works are available and relevant for this thesis.

Dufour-Kowalski et al. proposes in [39], an open-source software framework for forest growth modeling. The authors presented a state-of-the-art regarding the modeling software and frameworks for the application domains of forestry, ecology, agriculture and plants, together with their limitations. Based on the current modeling solutions, the authors found limitations regarding the creation, reusability and simplification of models in a software engineering context.

Therefore, the proposed software framework focused on the development of a software system for the reusability and simplification of model creation. With the objective of reducing the challenges faced by forest growth modellers, the open-source software framework has been produced with the following functionalities:

- A graphical user interface environment for model development, modification and integration.
- Reusable libraries of generic model components and data structures.
- A documented simulation environment for the execution, analysis and testing of models.

In [40], a presentation is given regarding the crucial role of modeling for both science and ecology. As the authors noted, with the advancements in computation, the development of models enabled to "mimic nature, generate questions, complement field experiments and observations", which would not be possible to undertake in

nature itself. Either because it would not be practical, or because it would take centuries. Additionally, in [41], Seidl argues that to study the ecosystem, ecologists must embrace "modeling and empirical research as two powerful and often complementary approaches in the toolbox of 21st century ecologists". Moreover, the author noted the importance of software for ecosystem modeling as a convenient tool to "access and apply models in recent years".

Finally, Scheller et al. present in [42] how ecology related software tools and ecological models are usually produced in an incremental manner, for a single purpose. As thus, such software tools and models cannot address new hypotheses.

Therefore, a new approach is proposed for the development of a software systems for the modeling of ecosystems. The focus of the paper is to illustrate the advantages of following software engineering modeling techniques such as UML and the software development life cycle process. The re-engineering of *LANDIS*, *an object-oriented model of forest landscape disturbance and succession* [43] is used as a case-study to illustrate the improvements related to the application of software engineering modeling techniques for ecosystem modeling.

2.4 Artificial Intelligence

> *"Every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it." [44]*
>
> — John McCarthy, *Turing Award (1971)*

Artificial Intelligence (AI) refers to the simulation of human intelligence processes demonstrated by machines programmed to mimic human cognitive behaviour and actions. In [45], AI is referred as "any system that perceives its environment and takes actions that maximise its chance of achieving its goals".

Despite being an important field of study within computer science since 1956 [44], with an important number of academic research and usage in the industry, a universal definition of the term *"Artificial Intelligence"* has not been defined. Since literature has many different and related definitions of AI, there exists four possible goals to pursue in AI, as observed in [45]:

- **Systems that think like humans.**
- **Systems that think rationally.**
- **Systems that act like humans.**
- **Systems that act rationally.**

As thus, a system with characteristics respecting such goals, can be classified as *"artificially intelligent"*. Therefore, to compare the ability of *"artificially intelligent"* systems to *think* in relation to humans, AI-based systems can be grouped into four distinct types:

- **Reactive**: Has no memory and only responds to different *stimuli*. This is the most basic type of AI, where an AI-based system was produced with a predictable output based on the received input.
- **Limited memory**: Uses memory to learn and improve its responses over time. This type of AI is the most used today and is similar to how neurons work in a human brain.
- **Theory of mind**: Understands the needs and emotions of other *intelligent* entities and interacts with them, adjusting behaviour. This type is known as the *next frontier* of AI as it replicates the human ability to attribute mental states to itself and others.
- **Self-aware**: Has memory and self-awareness. This type of AI equals human intelligence with the same needs and emotions.

2.4.1 Machine Learning

With the evolution of computing and the availability of large amounts of data, a subset of AI emerged, called Machine Learning (ML). As defined in [46], "Machine Learning is the study of computer algorithms that allow computer programs to automatically improve through experience".

With ML, the focus became the usage of artificial intelligence for the solving of practical problems, where sets of data are used as input to extract patterns in order to create predictions. With predictions, a *function approximation* allows the generalisation from the passed data and therefore, learn from it [47].

To learn from data, there exists multiple categories of ML approaches. Due to the scope of this thesis we present only the two most widely used categories:

- **Supervised Learning**: Refers to the learning of a function mapping both the inputs and the desired outputs [45] from a subset of labeled data, known as *training data*. With the production of an *inferred* function, using the *training data*, predictions can be made to associate outputs with *new* inputs.
- **Unsupervised Learning**: Refers to the ability to learn without labeled *training data* [45]. In this approach, the *training data* is used as input to identify hidden features and to learn patterns from the data.

2.4.2 Artificial Neural Networks

At the core of the different ML approaches, we have ANNs. First proposed in 1943 [48], Artificial Neural Networks are a set of algorithms modeled after the human brain and used to perceive complex patterns in sets of data in order to make decisions.

The structure of an ANN, depicted in Figure 2.16, is composed of a set of connected nodes with structures called artificial neurons (Figure 2.17) in three different types of layers, namely the input layer, the hidden layer and the output layer.

A neuron, represents the basic unit of a neural network composed of a set of inputs, a set of weights and an activation function, producing an output from its inputs.

Formally, each neuron receives a set of x vector feature values as its input and computes a predicted y output. Additionally, each unit has a set of w and b parameters, with w defined as a column vector of weights and b defined as a bias (b).

Both parameters are updated during the learning process. At each stage of the training interaction, the neuron computes a weighted sum of the values of x with respect to the current w vector of weights and finally, the bias is added to the summation of the multiplied values, giving y.

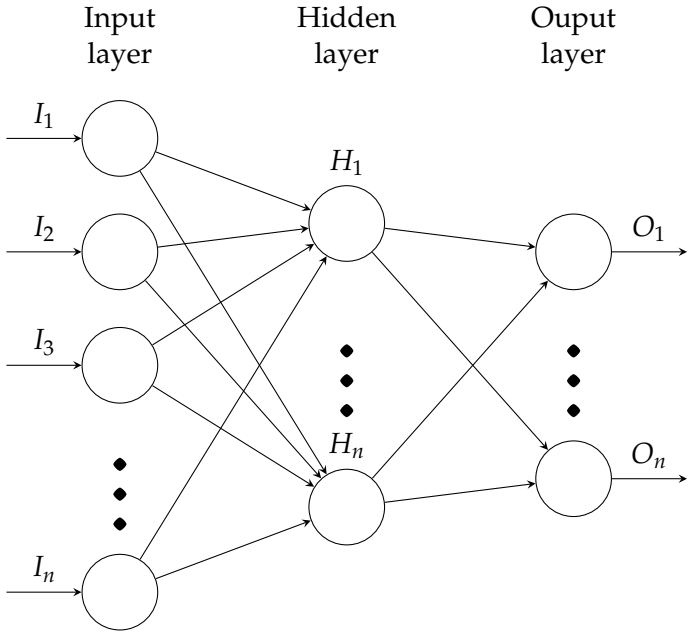


FIGURE 2.16: Example of an Artificial Neural Network architecture.

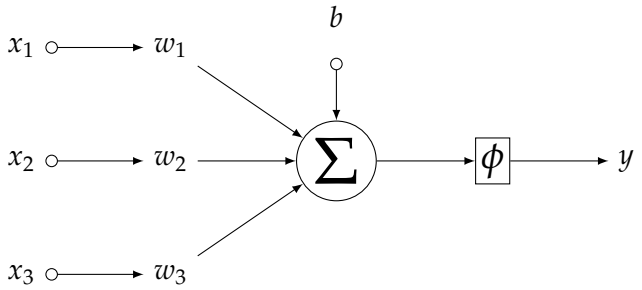


FIGURE 2.17: Structure of an Artificial Neuron.

Therefore, the simple computation process behind a trivial neuron can be modeled mathematically as follows:

$$y = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b \tag{2.2}$$

When those computations are concluded, the result can be passed to an activation function to help with the normalisation of the output.

Activation functions can be divided into two different types, namely the linear and non-linear activation functions. One the one hand, with a linear activation function, a neuron is a simple regression model and therefore, very limited and merely suitable for ordinary problems such as prediction or regression. On the other hand, the non-linear activation functions are the most used since they allow the creation of complex mappings between both the inputs and output.

Hence, the neuron model can be modelled mathematically in the following way:

$$y = \phi(w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b) \tag{2.3}$$

Additionally, the use of non-linear activation functions in a neural network model helps to relay the information backwards in a process called *backpropagation* [49], depicted in Figure 2.18.

Since *backpropagation* relays the information backward to compute the gradient, it helps consequently with the *gradient descent* [50, 51], which is an iterative optimisation algorithm used to find a local minimum of a differentiable function, (see Figure 2.20).

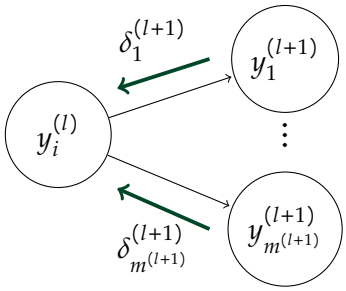


FIGURE 2.18: Backpropagation of the errors $\delta_i^{(L+1)}$ to control the weights through the neural network.

Different non-linear activation functions are used in neural network models such as the sigmoid, defined as $\phi(x) = \frac{1}{1+e^{-x}}$ which has a very simple derivative and a great impact on the speed of the backpropagation learning [52]. Other examples of non-linear activation functions are the hyperbolic tangent function which has a similar structure compared to the sigmoid function and is defined as $\phi(x) = \frac{\left(e^x - e^{-x}\right)}{\left(e^x + e^{-x}\right)}$ and the rectified linear unit (ReLU), defined as $\phi(x) = max(0, x)$, which is extremely computationally efficient by only outputting zero for negative inputs, and consequently, it is extensively used in neural networks. Nevertheless, because of the zero-hard rectification, it "does not capture negative information" [53].

For the learning of a neural network, there should be in a place a function to evaluate the best suited set of weights to be used, namely the *objective* or *loss function* [51]. Due to the fact that at the heart of a neural network exists an optimisation problem to be solved, a method to evaluate the modelling performance of the algorithm on a given set of data is required.

Therefore, in the context of optimising the learning performance of a neural network, loss functions are applied to evaluate the best set of weights during the training of neural networks. To evaluate, loss functions determine the *loss* (distance error) between the current output and the expected output. In simple words, when the loss function outputs a very high number, it implies that a high number of false predictions is occurring in the model and therefore, the tuning of parameters such

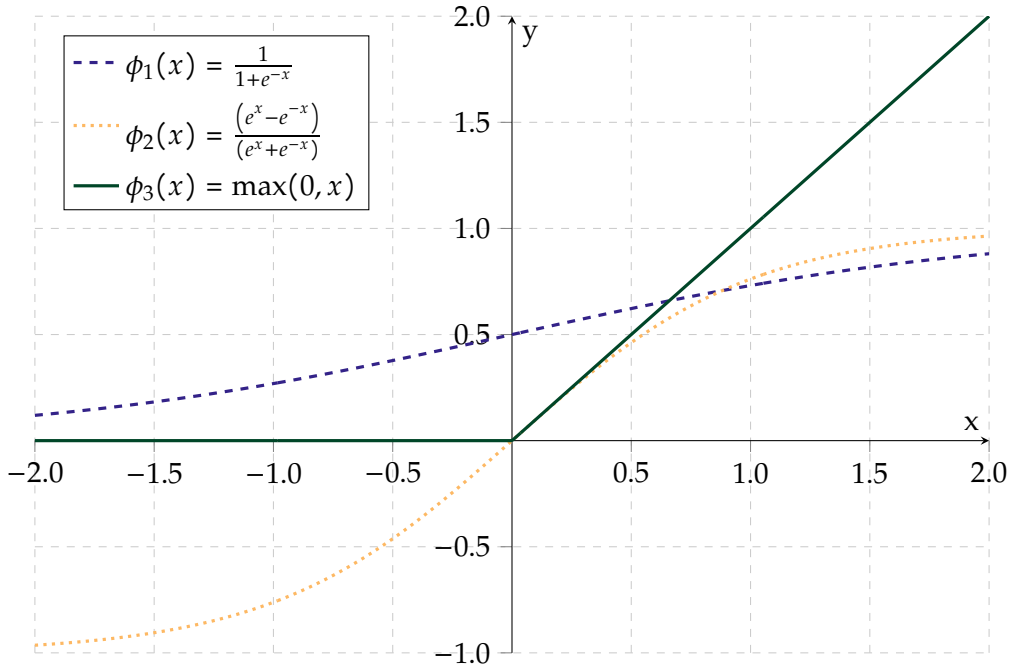


FIGURE 2.19: Sigmoid, hyperbolic tangent and rectified linear unit activation functions plotted in a $[-2, 2]$ domain.

as the number of layers, neurons, activation function et cetera, should occur. As thus, loss functions define how well the neural network is predicting with respect to a given set of data.

Multiple loss functions are used in neural networks and much like the activation functions, loss functions can be divided into two different types, in particular the binary classification losses and regression losses. Binary classification loss functions are mainly used in predictive modelling problems with two different labels, where the target values are in the set $0, 1$.

One of the most used loss functions in binary classification problems is the cross-entropy, defined as:

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log\left(p\left(y_i\right)\right) + \left(1 - y_i\right) \cdot \log\left(1 - p\left(y_i\right)\right) \tag{2.4}$$

where y is the label value in the corresponding target, $p(y)$ is the prediction probability and N is the number of points considered. Thereby, the binary cross-entropy computes an average score of the differences between the target and predicted probability distributions.

Regression loss functions are used instead to predict a finite quantity and consequently different loss functions are taken into account for that purpose. One of the most commonly used and simple regression loss functions are the mean absolute error (MAE) and the mean squared error (MSE), which computes the sum of squared distances between the target variable and the predicted values.

Gradient Descent

During the training of a neural network, we are trying to find the best solution for an optimisation problem by determining the parameters with the best output in terms of the objective function. During this process, gradient descent occurs by computing the gradient of the loss function by taking the first-order partial derivatives with respect to the parameters and subsequently, updating the parameters with respect to the gradient, giving the direction of the steepest ascent. Therefore, in every iteration of the process, the parameters are updated in the opposite direction of the gradient computed for the cost function.

Finally, to reach the descent point, a learning rate parameter should be specified to define the step size to be updated for the purpose of attaining the (local) minimum. This cycle process is repeated until a minimum is reached, which however is not guaranteed to occur due to convergence.

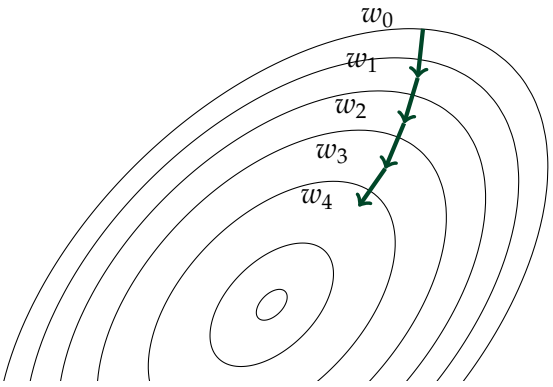


FIGURE 2.20: Example of a gradient descent to solve an optimisation problem.

With the evolution of the Machine Learning field, multiple ANNs architectures have been created. Each new ANN architecture has been designed following the same principles as depicted in Figure 2.16, but with the purpose of being more efficient for different problems and tasks.

Due to the scope of this thesis, we present only a subset of the most used ANN architectures in the remote sensing field.

Autoencoders

First introduced in 1986 [54], an *autoencoder* is an unsupervised learning architecture with the aim of reducing the dimensionality of inputs by recreating them.

The architecture of an autoencoder is composed of two components, namely the *encoder* and the *decoder* and has the unique characteristic of having the same dimensions on both the input and output layers.

On one side, the *encoder* maps the input into the code, encoding the input data and storing it in the hidden layer. On the other side, the *decoder* decodes the information stored in the hidden layer allowing the autoencoder to approximately reconstruct the output and thereby, preserving the most intrinsic aspects of the data. An architecture example is depicted in Figure 2.21.

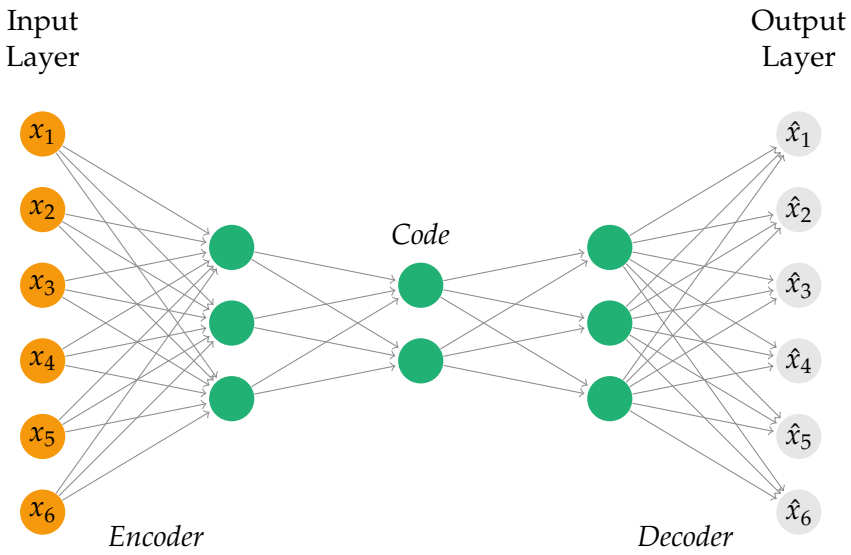


FIGURE 2.21: Example of an autoencoder architecture.

Generative Adversarial Networks

Another popular application of deep learning on remote sensing data is through generative models, namely *Generative Adversarial Networks (GANs)* [55].

GANs are an example of an unsupervised learning architecture, using two different neural networks competing with each other in a *zero-sum game* situation, where one agent's gain is another agent's loss, with the purpose of modeling and generating new data instances that resemble the feed training data.

Therefore, given a set of data instances X and a set of labels Y, one neural network works as the *Generator*, capturing the joint probability $p(X, Y)$ to generate new data instances in the domain while the other neural network, the *Discriminator* captures the conditional probability $p(Y|X)$ to discriminates between different kinds of data instances.

To put it simply, a GAN architecture works in the following manner:

1. The generator takes a noise sample from the latent space, which is a simpler representation of a data point.
2. With the latent space sample, the generator generates a *fake sample* of data.

3. The generated sample is fed into the discriminator alongside a *real sample* of data, taken from the training dataset.

4. The discriminator returns a prediction with a probability of values between 0 and 1, where 0 means that data is definitely *fake* and 1 meaning that the received sample of data is definitely authentic.

Hence, with the adversarial training of GANs, the *Generator* progressively generates data closer to the training data distribution, making it an effective architecture for data augmentation [56]. However, if one neural network is better than the other, the GAN will not converge [55], making the training process harder.

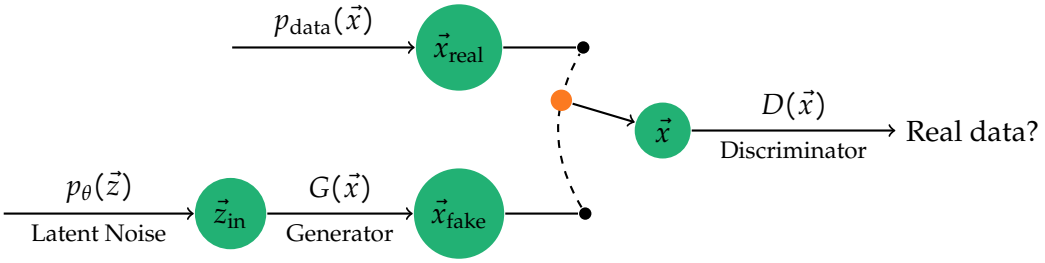


FIGURE 2.22: Generative Adversarial Network Architecture.

Convolutional Neural Networks

First introduced in 1989 to recognize handwritten zip code digits [57], Convolutional Neural Networks (CNNs) are an example of supervised learning methods, with an architecture inspired by the organisation of the *visual cortex* and similar to the connectivity pattern of neurons in the *Human brain*. CNNs rely on annotated data to compute the loss cost and consequently, backpropagate the errors to update the gradients of each parameter of the network.

In general, the architecture of a CNN (shown in Figure 2.25) is comprised of one or multiple convolutional layers, one or multiple pooling layers which are followed by one or more fully connected layers, used to create a prediction. Since it contains multiple layers between the input and output layers, CNNs are in fact a special type of artificial neural networks, more formally it is a Deep Neural Network (DNN), most commonly applied to analyse and classify visual imagery in different fields.

Among the different architectures, *CNNs* [57] are the most dominant application of deep learning, a subfield of machine learning presented in Section 2.4.3.

The building blocks of a CNN are described below:

- **Convolutional Layer**: Mathematical operation applied to the input data to filter and reduce the information to produce a feature map, a summarised set of the most descriptive information. The filter in the convolutional layer is called

a kernel and depending on the application, can have diverse dimensions such as 2×2 or 3×3. Figure 2.23 shows an example of a convolutional operation.

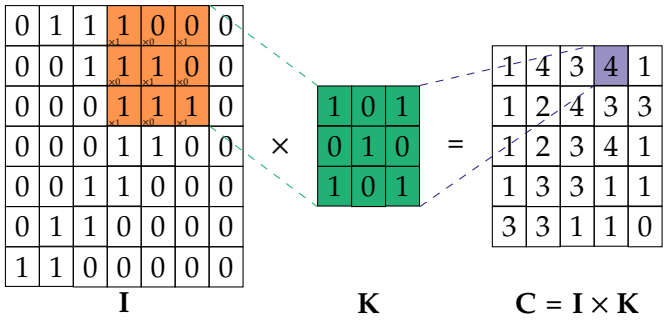
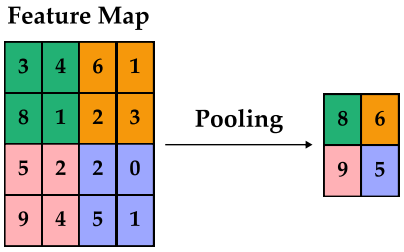
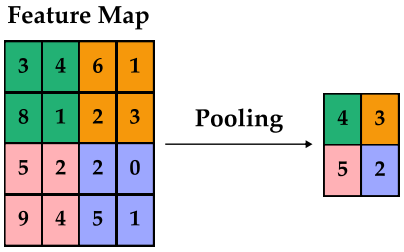


FIGURE 2.23: Two-dimensional convolution operation, where a kernel matrix (K) "slides" (from top-left to bottom-right) over the the target input matrix (I) to produce an elementwise multiplication. As a result, it will be summing up the results into a single output pixel, in a matrix (C).

- **Pooling Layer**: Operation applied on a feature map to reduce its dimensions. Pooling layers summarise the information contained in a feature map with the intent of reducing the number of parameters to learn during the training as well as the amount of computations to perform. Figure 2.24 shows an example of common pooling operations.



(a) Max-Pooling



(b) Average-Pooling

FIGURE 2.24: Example of two common pooling operations in CNNs, used to downsampling the feature maps obtained from the convolution layer.

- **Fully-Connected Layer**: With the 3-dimensional matrix outputs from the previous pooling or convolutional layers, the fully-connected layer first flattens in steps the received information and creates a single vector with a size corresponding to the trained target classes. In the last step, the vector is used to give the final probability for each class.

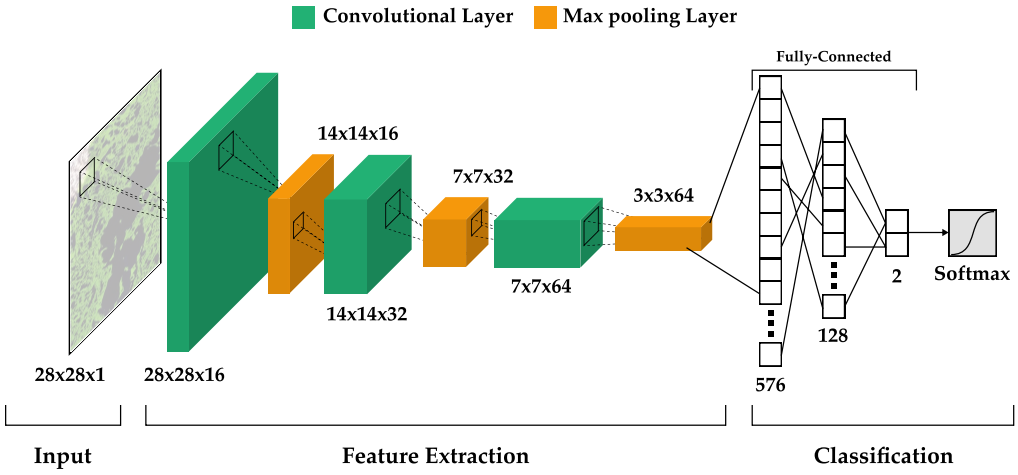


FIGURE 2.25: Convolutional Neural Network Architecture.

Since CNNs are very effective architectures in reducing the number of parameters without losing on the quality of trained models, research on convolutional neural networks has progressed rapidly and created many different CNN architectures being used nowadays, for different types of tasks and problems.

Due to the intrinsic convolutional characteristics of reducing the input image dimensionality while retaining information capable to learn abstract and effective features, CNNs have been adopted as a solution for a wide range of Earth observation problems [58, 59, 60].

2.4.3 Deep Learning

First introduced in 1986 [61], the term *"Deep Learning"* referred to the usage of multiple layers to build more complex functions in order to compose shallow artificial neural networks. Nevertheless, the training of such deep neural network architectures was impractical due to limitations in hardware.

With the advances in hardware allowing more processing power and the improvements in data collection since early 2000s, the modern era of Deep Learning (DL) begun. DL is a subfield of machine learning and it is currently the most applied type of artificial intelligence due to the application of *feature learning*. *Feature learning* refers to the set of techniques allowing a system to automatically discover and learn the abstract representation of data. Such techniques allows a system to learn

complex functions mapping the input to the output directly from data, without requiring and being dependent on human decisions impacting the data. As thus, DL algorithms do not require human instructions to perform a given task.

To automatically discover and learn the abstract representation in data, deep learning algorithms use *"deep"* layers in artificial neural network architectures. In other terms, artificial neural networks are stacked, so the output of the first becomes the input of the second and so on. Consequently, as the number of layers in the network is increased, the descriptive power of the network is likewise improved.

Therefore, as opposed to traditional neural networks, composed of a few hidden layers, the architectures of modern Deep Learning networks can be composed with hundreds of layers, each one with thousands of outputs (hidden units) traversing the network. The number of outputs in each layer of the network is referred as the *width* and the number of layers is referred as the *depth*. Both are an example of *hyperparameter* variables in DL, determining the capacity to learn and the structure of the network.

The advancements in artificial neural network architectures allowed the description of arbitrarily complex functions in high dimensions, making it efficient on different fields such as climate science, medical image analysis, computer vision, natural language processing, among many others.

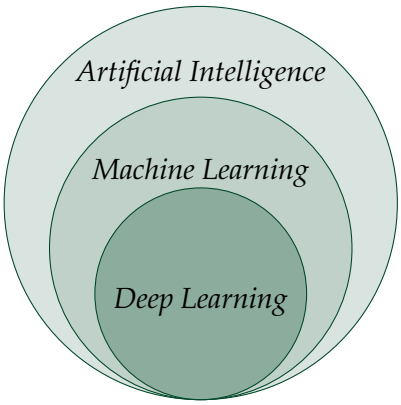


FIGURE 2.26: Relationship between AI, ML & DL.
ML is a subset of AI whereas DL is a subset of ML algorithms.

2.4.4 Related Work

In this section, we present the related work in the context of artificial intelligence for ecosystem observation.

Camps-Valls et al. present in [62] a comprehensive approach to apply artificial intelligence, more specifically deep learning, for Earth sciences. The authors present how the advancements in deep learning, congruent to the success in the domain of computer vision, have proven to be effective to extract information from complex

and homogeneous data in the remote sensing field. Additionally, a presentation is given about the different families of neural networks such as convolutional neural networks, autoencoders, generative adversarial networks or recurrent neural networks together with different use cases and applications for ecosystem observation. Examples of the applications include remote sensing image classification, semantic segmentation, object detection or ecosystem simulation.

Ma et al. [63] present a detailed review regarding the rise of deep learning in remote sensing applications. The authors present how deep learning algorithms have been applied for remote sensing image analysis tasks including image registration, scene classification, object detection, land use and land cover (LULC) classification and segmentation, among many others.

Lary et al. [64] present how the different modeling capabilities of machine-learning methods have resulted in an extensive application for solving problems in geosciences and remote sensing for the study of as land, ocean and atmosphere ecosystems. Moreover, it is shown that new applications of genetic algorithms in the geoscience and remote sensing domain has demonstrated to be an efficient approach to generate practical prediction equations, in cases where artificial neural networks are considered as black-box models since their behaviour may be complicated to understand in certain situations.

Rolnick et al. [65] present how the application of machine learning methods can help combat climate change. In their paper, the authors identify the important problems where the application of machine learning approaches can have an high impact and therefore, tackle climate change problems together with other fields in processes involving not only mitigation to reduce greenhouse gas emissions but also in the adaptation to unavoidable consequences.

Stewart et al. [66] demonstrate the complexity to associate machine learning algorithms to remote sensing data for Earth observation. In their paper, the authors show that the main complexity is due to the variance in the data collection methods and the processing of geospatial metadata. Since Earth observation data may contain multiple spectral bands, coordinate systems and resolutions, the processing is complex and creates issues to apply machine learning methods for remote sensing applications. Hence, the authors present a new framework to integrate Earth observation data into the PyTorch [67] deep learning ecosystem, simplifying the preprocessing of geospatial imagery.

Chapter 3

Ecosystem Resilience Analysis

The analysis of ecosystem resilience requires a large quantity of data, and based on our literature review, we have found that main source of observations for *Ecosystem Resilience Analysis* is supported by Earth observation data [18, 21]. Despite the great number of Earth observation sources of data, either remote sensing or in-situ, we found three primary problems with the exploitation of the Earth observation data for *Ecosystem Resilience Analysis*.

1. Earth observation data may contain multiple multiple resolution bands, coordinate systems, resolutions, formats, et cetera. These different and complex characteristics of multi-source and homogeneous data create issues to create a specification of requirements for the *Ecosystem Resilience Analysis* process.
2. Since natural ecosystems are evolving and dynamic entities, a precise specification of data requirements is necessary not only to collect the most adapted source of data, but also to select the most adequate method to extract information regarding the resilience of ecosystems.
3. To undertake *Ecosystem Resilience Analysis*, complex and repetitive tasks are required to be done regarding the collection and pre-processing of Earth observation data.

We found that with the help of modeling approaches, complexity could be reduced in order to create not only a precise specification of requirements for the *Ecosystem Resilience Analysis* process but also to automate the processes regarding the data collection and optimisation. Additionally, we found that the usage of artificial intelligence algorithms, more specifically deep learning, can be an effective method to extract information from complex and homogeneous data in the context of Earth observation. Examples of the applications include image classification, land use and land cover classification, semantic segmentation, object detection, image fusion or ecosystem simulation, among many others [62, 64, 65, 66, 68].

Hence, in this chapter, we present our software engineering approach [1], created with

[1] This approach has been submitted to the *MDE Intelligence* Workshop on Artificial Intelligence and Model-driven Engineering.

the objective of allowing the development of software systems for the ecosystem resilience analysis. The proposed *Ecosystem Resilience Analysis* approach follows a model-driven engineering perspective coupled with artificial intelligence (MDE4AI). The approach is formalised using the BPMN 2.0 standard [69] as shown in Figure 3.1

In the following sections, we present the three tasks of the approach and the five generated artifacts.

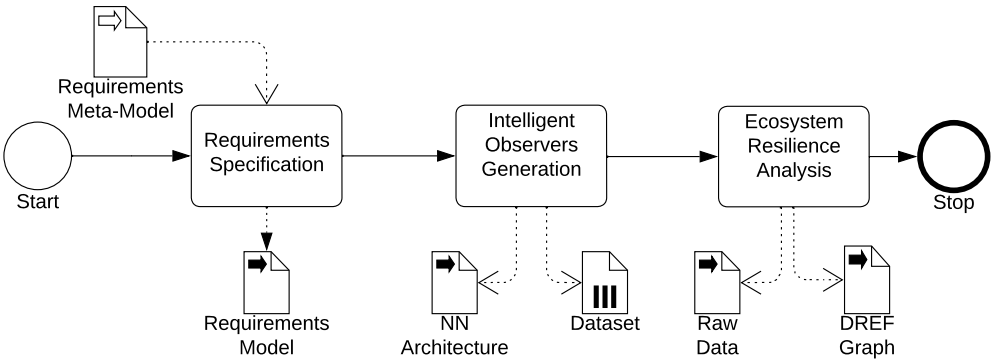


FIGURE 3.1: Our proposed Business Process for the Ecosystem Resilience Analysis.

3.1 Requirements Specification

In this section, we focus on the first activity of software engineering, namely the requirements engineering phase in a MDE context.

With the technology advancements in remote sensing satellite sensors over the last decades [70] and the progress of machine learning approaches applied to remote sensing data [62], the requirements specification for Earth observation have become quite complex.

On one side, the complexity is attributable to the high-dimensional and diverse remote sensing data generated from the satellite sensors. As presented in Section 2.2.2, satellite sensors can have, among other characteristics, different resolutions in the spatial, spectral, radiometric and temporal dimensions, creating a wide range of possible requirements for a stakeholder.

On the other side, the characteristics of remote sensing generated data such as the number of spectral bands impacts the reliability of traditional artificial intelligence architectures, which have been developed with the conventional three channel RGB imagery data [66].

Therefore, to benefit from the potential of artificial intelligence methods on remote sensing datasets as well as in-situ generated datasets, a clear and precise requirements specification is necessary to select the most adapted data from the correct

satellite sensor or in-situ source, for a given objective in the stakeholder domain. The requirements specification phase becomes even more complex when a stakeholder, such as a data scientist, requires the analysis of an *ecosystem resilience*. Not only due to the dynamic nature of ecosystems but also because not every satellite sensor nor in-situ sources can observe with the same level of detail every ecosystem property, either because of atmospheric influences or because of surface reflectance, as explained in Section 2.2.1.

Nonetheless, with modeling approaches, complexity can be reduced through abstractions and simplifications as presented in Section 2.3. Since modeling has proven to be crucial in the specification, design and development of a software systems, in this section we present a compliant UML (see Section 2.3.3) metamodel for the modeling of requirements for software systems analysing the resilience of ecosystems.

With the metamodel shown in Figure 3.2, we aim to allow the modeling of the different requirements for the *entities* under study, the *properties* of interest and finally, the *observers* collecting the data.

In the following subsections, we describe and illustrate the different concepts of the metamodel. The definition of the three fundamental concepts of entities, properties and observers are made within the *DREF framework* [7], as presented in Section 2.1.2.

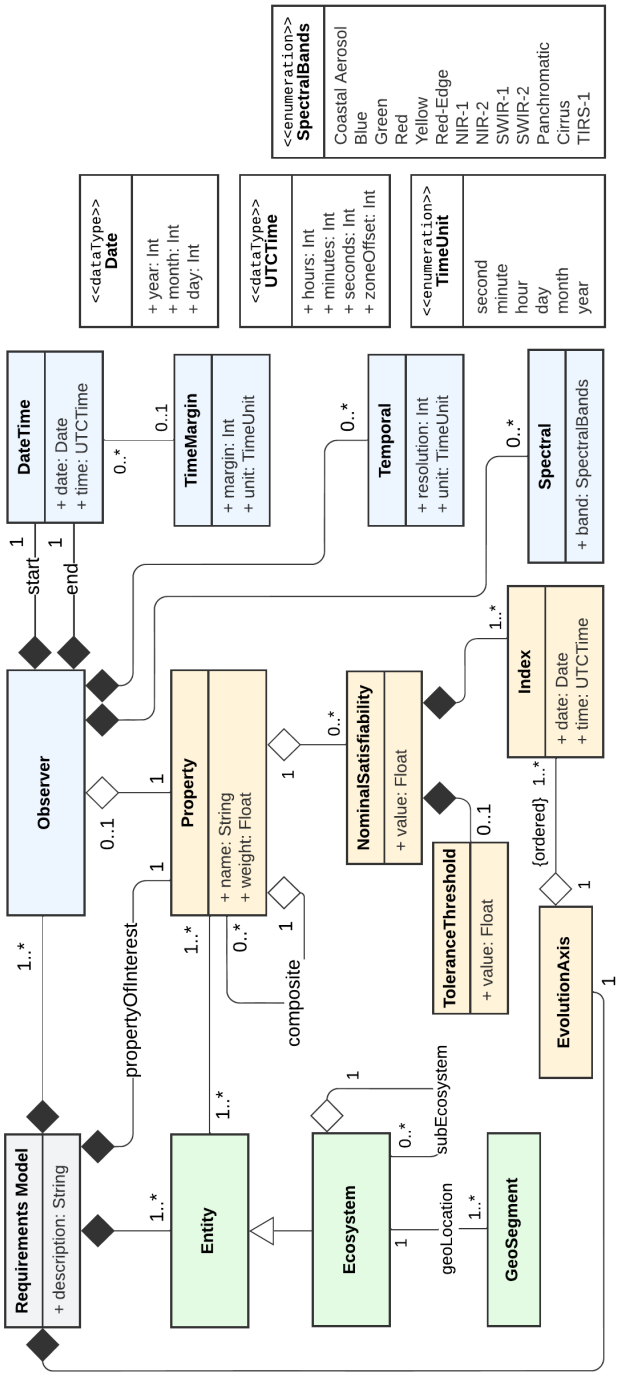


FIGURE 3.2: Requirements Specification Metamodel.

3.1.1 Entities

An *Entity* is defined as "anything that is of interest to be considered" [7]. In this approach, we restrict our focus on any entities that are ecosystems, namely terrestrial, aquatic or climatic environments.

Therefore, the concept of *Ecosystem* in the metamodel is a specialisation of an *Entity*, with the goal of defining the ecosystem to study.

Additionally, the concept of *GeoSegment* is proposed to allow the specification of a geographic delimitation of the ecosystem to analyse. Hence, with the *GeoSegment*, data scientists can define a set of coordinates points on the *Entity* of interest where the ecosystem resilience analysis should be performed.

Moreover, since ecosystems may be spatially fragmented, the concept of sub-ecosystem is introduced, to allow the study of properties in specific sub-ecosystems, as part of a larger ecosystem *Entity*.

Running Example

As an example of the different concepts, we can say that the territory of Luxembourg is an *Ecosystem*, a specialisation of an *Entity* of interest, with the delimitation of the territory borders as a set of *GeoSegments* such as `5.67405195478, 49.4426671413, 197 6.24275109216, 50.1280516628`. To illustrate the concept of sub-ecosystem, we can define the *"Grünewald"* forest in central Luxembourg as the sub-ecosystem, with the delimitation defined with a GeoSegment with the following values: `6.194487, 49.658628, 6.218090, 49.669683`.

A UML model instance of the example described above is shown in Figure 3.3. The model instance is compliant with the dedicated metamodel shown in Figure 3.2.

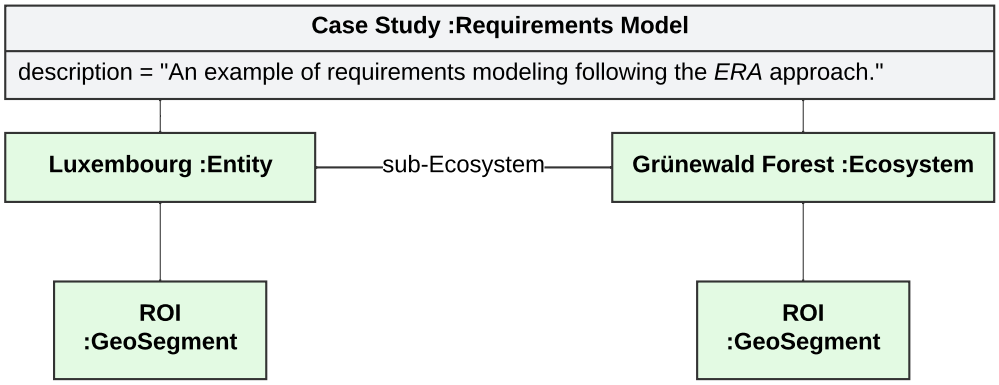


FIGURE 3.3: Example of modeling the Entity concepts (in green).

3.1.2 Properties

A *Property* is defined as "the concept characterising an entity" [7]. In the context of ecosystems, a *Property* can be biotic components such as plants, animals, and other organisms, as well as abiotic components, the non-living chemical and physical parts of the ecosystem environment. Abiotic components affect the ability of organisms to survive and their functioning which are characterised by physical and chemical factors such as water, radiation, light, humidity, temperature, atmosphere, or soil, among others.

In the metamodel, there is one *Property* named *propertyOfInterest* being the main property against which we analyse the resilience of the ecosystem under study (see Section 3.3).

Due to the complex nature of ecosystems, an *Entity* may be related to multiple properties of interest. Additionally, a *Property* may be decomposed in multiple properties. Consequently, we provide the concept of *weight*, allowing to give more importance to a *Property* over other ones. The ability to specify a *weight* to a *Property* is important since multiple *Properties* may be relevant with respect to each other for the *Ecosystem Resilience Analysis* but they might have different impacts in the ecosystem. For instance, a *Property* of interest might be for example the greenhouse gas in the atmosphere, decomposed in multiple properties such as carbon dioxide (CO_2), methane (CH_4) and nitrous oxide (N_2O), gases that accumulate in the atmosphere. Since methane (CH_4) has a "global warming potential 86 times stronger per unit mass than CO2 on a 20-year timescale" [71], its *weight* shall be more important in the study of the *Ecosystem Resilience Analysis*, compared to other gases.

Moreover, a *Property* may have many *NominalSatisfiability* values associated with it, allowing the specification of a set of expected nominal satisfiability values. Additionally, a *NominalSatisfiability* may have an associated *ToleranceThreshold* value, allowing the specification of an upper bound margin value for the comparison with the computed satisfiability of a *Property*.

Using the concepts of *NominalSatisfiability* and *ToleranceThreshold* on a *Property*, the proposed system can compare the specified requirements and the computed satisfiability values of the *Property* of interest, as we will present in Section 3.2.

Additionally, to allow the specification of *NominalSatisfiability* requirements in different intervals of date and time for a given *Property*, we introduce the concept of *Index*. With the *Index* concept, the *NominalSatisfiability* requirements are indexed in a determined *date* and *time*.

During the ecosystem resilience analysis, the satisfiability of a *Property* is evaluated at the specified *date* and *time* of an *Index*, with respect to the *NominalSatisfiability* and *ToleranceThreshold*. To represent the set of *Index* requirements in a time axis for the ecosystem resilience analysis, we propose the concept of *EvolutionAxis*.

Running Example

In the previous example (see Figure 3.3), we have illustrated the different requirements for the concepts of *Entities*. We now extend the previous example with the specification of requirements related to the *Property* concept and the associated concepts of *NominalSatisfiability*, *ToleranceThreshold* and *Index*.

Hence, we specify the total CO_2 absorption in the "Grünewald" forest *Ecosystem* as the *Property* of interest, with a weight of 1.0. Moreover, we specify that the collected satisfiability values for CO_2 absorption shall be compared with a *NominalSatisfiability* value of 651.945, representing as estimation of the potential of the forest area to absorb more carbon from the atmosphere than it releases.

To estimate the *NominalSatisfiability* value of 651.945, we have computed the area of the "Grünewald" forest (186.27 hectare) multiplied by the average typical absorption rates in temperate regions of 3.5 tonnes, as defined in [72]. Since we use an average for the computation of the *NominalSatisfiability* value, we specify a *ToleranceThreshold* of 20%, to represent an error margin. Finally, since we are specifying a requirement for a determined time, we use the concept of *Index*. As thus, the specified *NominalSatisfibiality* is associated with an *Index*, having the date of 08/08/2022 and a time of 00:00 in a UTC zone offset of +1, which is later used to evaluate the satisfiability of the *Property* at the *Index date* and *time* in the *Ecosystem Resilience Analysis*.

Once again, a UML model instance of the example described above is shown in Figure 3.4.

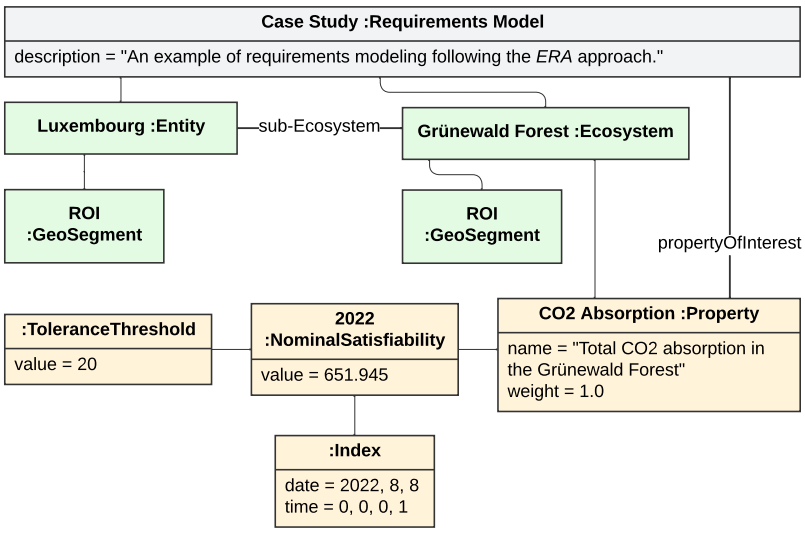


FIGURE 3.4: Example of modeling the Property concepts (in orange) with the Entity concepts (in green).

3.1.3 Observers

An *Observer* is an abstraction of a data source providing data for the evaluation of a related *Property* in a given period of time. Therefore, we propose the concept of *DateTime*, with the intent of allowing the specification of an interval with a *start* and an *end* date during which data shall be collected.

Furthermore, we propose the concept of *TimeMargin*, allowing the specification of a time margin for the collection of the data by the system in both the *start* date and *end* date of the observation interval. Thus, with the concept of *TimeMargin*, data scientists may specify a tolerance of time and thus, increase the probability of finding interesting data sources with the system under development. For instance, remote sensing data may not be updated daily over a given region of interest.

Since Earth observation data may come from multiple sources, in the dedicated metamodel, we introduce two different concepts allowing the specification of data characteristics to be collected with in-situ or remote sensing data. With the *Temporal* concept, data scientists can specify the data acquisition interval, convenient for both the remote sensing and in-situ data sources. With the *Spectral* concept, which is directed towards the specification of remote sensing data, the specification of the required spectral bands to be collected with the data source can be easily defined.

Running Example

We illustrate now the concepts of *Observers*. In our running example, the *Property* of interest that we have specified is the total CO_2 absorption in the Grünewald forest. Therefore, with the *DateTime* concept, we specify that we shall collect data (in-situ or remote sensing) starting from the 1st January 2022 until the 1st August 2022. Moreover, we specify a *TimeMargin* with a value of 7, and a unit of "day", to allow a tolerance margin of 7 days , with the intent of being less restrictive for the collection of data. Finally, we specify with the *Spectral* concept that the data to collect shall have the red, green and blue spectral bands, making it a specification oriented towards a remote sensing *Observer*, since in-situ data does not contain a spectral resolution.

The model instance respecting the dedicated metamodel (see Figure 3.2) is shown in Figure 3.5, updated with the concept of *Observers* described above.

3.2 Intelligent Observers Generation

In the previous section, we have presented the requirements engineering phase resulting in a dedicated metamodel for the modeling of requirements. With the proposed metamodel, stakeholders are able to model the *Ecosystem Resilience Analysis*

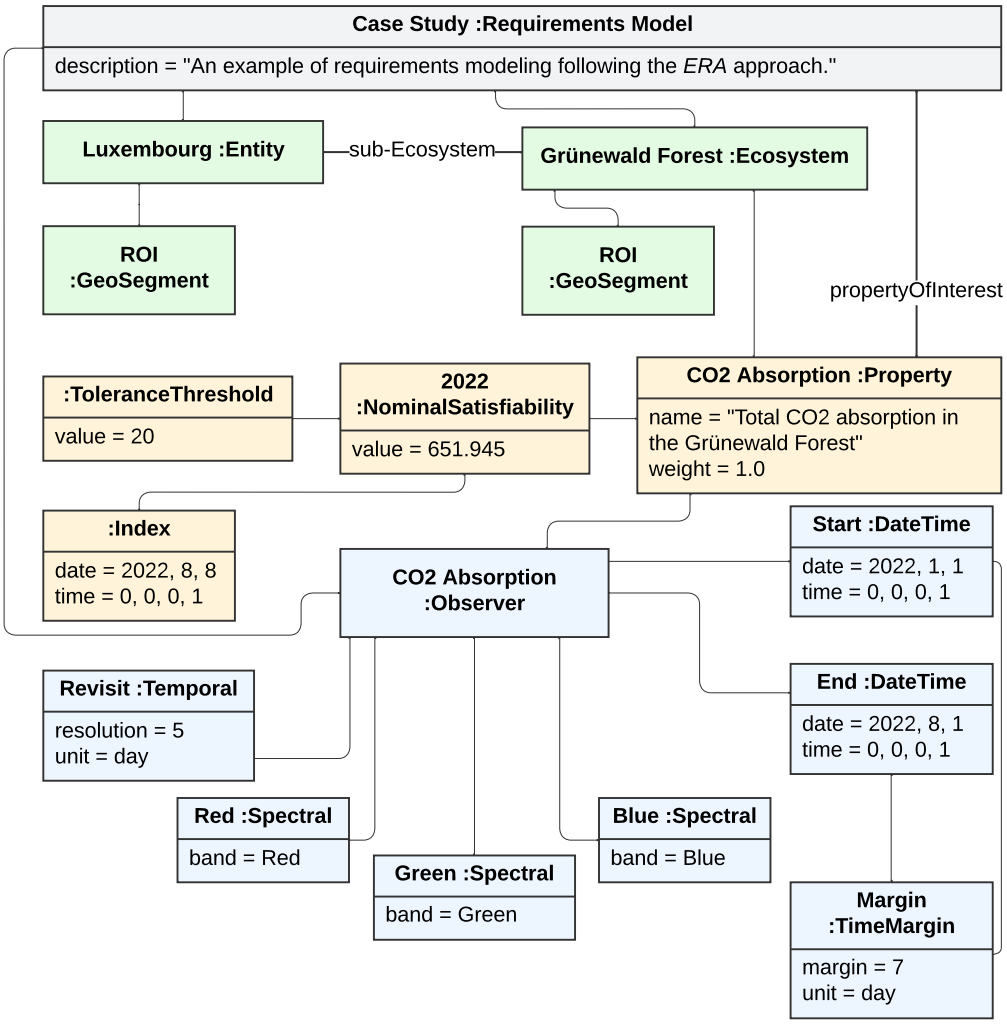


FIGURE 3.5: Example of modeling the Observers concepts (in blue), the Property concepts (in orange) and the Entity concepts (in green).

requirements in the system under development using the presented concepts such as *entities*, *properties* and *observers*.

However, as presented in the Section 2.3.1, once the requirements are specified, a software system complying with the requirements specification must be produced. In our proposed approach, data is fundamental to comply with the requirements and undertake *Ecosystem Resilience Analysis*. Therefore, the system shall collect automatically the data to enable the resilience analysis of the specified *properties* of interest.

Additionally, since artificial intelligence has demonstrated a great potential to extract valuable information from remote sensing data [68], the software system shall generate intelligent skeletons of neural network architectures to make use of the collected data with artificial intelligence architectures, to evaluate the resilience of the *properties* of interest on the specified *entities*.

Therefore, in this section, we present how the adequate information is extracted from the specified requirements with informal model transformations (see Figure 3.7). As presented in Section 2.3.2, MDE promotes the concept of model transformations to transform models from one form to another to leverage automation in software development. Hence, we leverage automation with model transformations to use the specified requirements with the functionalities of the system, namely the data collection and data optimisation methods.

Finally, we present how the proposed system extracts information for the *Ecosystem Resilience Analysis* from the specified *properties* with the collected data.

3.2.1 Collection

Earth observation data can be acquired with multiple sources, mainly categorised in remote sensing data and in-situ data. With the proposed system approach, the data collection functionality retrieves multi-source datasets available to the general public such as open data platforms, government websites or from open-source repositories, to name a few.

Accordingly, using model transformations, the system extracts the specified data requirements among the *entities*, *properties* and *observers* concepts. With the *GeoSegment* requirement, the system identifies the area of the *entity* of study. Hence, with the specified area, the system retrieves sources of data where data collection is available for the specified *entity* area.

Moreover, with the specified set of *properties*, the system searches specific data allowing the resilience analysis with the explicit *properties* of interest, in the defined *entity* area. Finally, with the concepts of *observers*, the system uses the specified *start* and *end* date and time to lookup for data respecting the time interval. If a time tolerance is given with the concept of *TimeMargin*, the *start* and *end* events are computed in compliance with the specified time tolerance.

Furthermore, the *Temporal* concept is used in the data collection functionality to retrieve data sources respecting the specified acquisition time interval, particularly useful for the collection of daily updated data for example.

Moreover, due to the fact that biggest source of Earth observation data comes from remote sensing sources, if the *Spectral* concept is specified, the system searches for remote sensing data satisfying not only the previous requirements, but also data collected with satellite instruments having the specified spectral bands.

Finally, if the system does not retrieve data satisfying all the model requirements, the usage of artificial intelligence for the generation of synthetic datasets can be applied using generative adversarial network (GAN) architectures, as presented in the section 2.4. An example where the usage of GANs can be helpful is in cases where

the data is limited in the *start* and *end DateTime* specified or when there is a "gap" of data covering the specified *Ecosystem*.

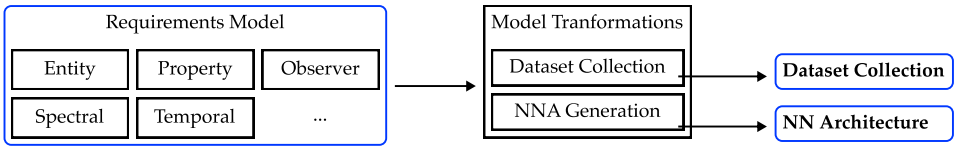


FIGURE 3.6: Illustration of how the specified requirements are used through model-transformations to extract the required information in order to collect a dataset and generate a neural network architecture.

3.2.2 Optimisation

Once data is collected with the system, an additional functionality is proposed to improve the quality of the retrieved data and consequently, increase the reliability of the data for the ecosystem resilience analysis using artificial intelligence.

Based on our research, Earth observation data can be rather heterogeneous with multiple sources (optical, radar, laser, et cetera), file formats, data types, resolutions and coordinate systems [73]. Furthermore, with the high dimensionality of Earth observation data, the interpretation of the data with artificial intelligence can be complex due to the large amount of features and observations.

As dimensionality increases, the volume of space grows so fast that the amount of data required to densely sample it to use with artificial intelligence methods increases exponentially, resulting in the *"curse of dimensionality"* [74], first coined by Richard E. Bellman in 1966 to refer to the intractability of certain algorithms in high dimensionality.

Therefore, to remedy the curse of dimensionality, the system applies a widely used unsupervised machine learning method called Principal Component Analysis (PCA). With this method, the objective of the system functionality to optimise the collected data is two-fold:

1. The PCA method is used to reduce the noise in the collected data.
2. The PCA method attempts to reduce the data dimensionality to decrease the data complexity.

PCA is based on projection methods, used to represent the partition of the data variation and to observe the underlying trends in the features of the data, as well as the clusters and outliers. As such, the important information contained in the dataset is extracted and summarised in a set of indices named *"principal components"*, allowing

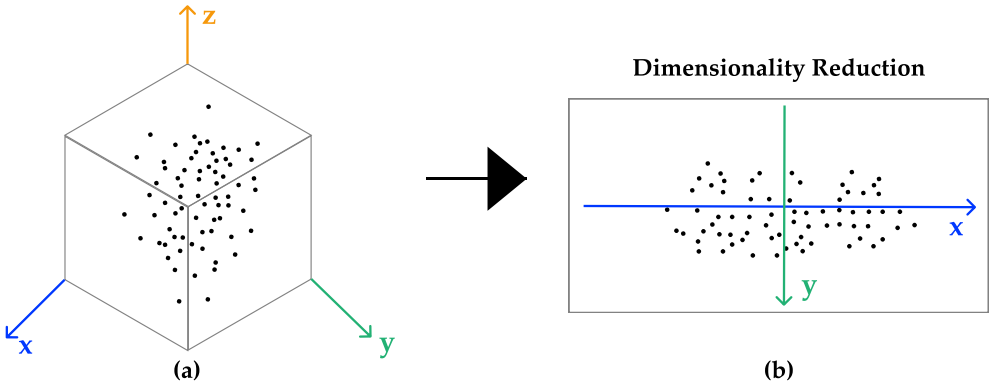


FIGURE 3.7: Simple illustration of the Principal Component Analysis (PCA) method on the reduction of complexity. (a) represents data points in a 3D space before PCA dimension reduction, (b) represents data points after the application of PCA dimension reduction

the reduction of noise and the reduction of dimensionality. Other than that, it allows the analysis, among others, of multicollinearity and missing values in the dataset.

As a result of the application of the PCA method, the complexity of the collected data is reduced. Additionally, if remote sensing data is specified in the requirements model with the *Spectral* concept, the application of PCA is used to select only the specified spectral bands among the set of bands present in the collected remote sensing dataset. This phase is important for remote sensing datasets. Compared to traditional vision datasets, where images are captured in three channels (RGB), satellite sensors as presented in Section 2.2.2 capture from several tens to a few hundreds of spectral bands in the electromagnetic spectrum, increasing not only the data diversity and complexity but the required storage capacity for analysis.

For instance, the Hyperion satellite collects Earth observation data using 242 hyperspectral bands within the electromagnetic spectrum with its imaging spectrometer [75], creating thousands of complex files to analyse, where each file can take up to several gigabytes for a single scene.

Consequently, the application of PCA can lead to a considerable reduction of computational complexity [76] on the usage of high dimensional remote sensing data such as hyperspectral imagery with artificial intelligence architectures.

3.2.3 Information Extraction

After collection and possible optimisation of a given dataset, the system uses the dataset to extract information regarding the retrieved satisfiability values on the specified set of *Properties*.

However, depending on the dataset, the extraction of information can involve additional work. Therefore, if the dataset must be processed to be exploited, we propose the generation of artificial intelligence architectures.

As already presented in Section 2.4.4, the application of artificial intelligence algorithms have shown potential to address problems related to Earth sciences [62, 65, 64].

Thus, we propose the generation of *CNNs*, *GANs* and *Autoencoders* architectures, with the intent of exploiting the dataset and thereby, extract the values on the *Properties* of interest.

As a result, the proposed generation functionality of the different artificial intelligence architectures allows the computation of the *Property* satisfiability values for the ecosystem resilience analysis, even with datasets requiring additional processing such as remote sensing datasets or unprocessed in-situ datasets. For use cases where articial intelligence is not required, the system can generate mathematical linear regressions on a given dataset in order to predict future observations for example.

With the system generation, we aim to reduce the time and complexity that can occur during the creation from scratch of the different proposed architectures for Earth observation problems.

3.3 *Ecosystem Resilience Analysis*

In the previous sections, we presented not only a dedicated metamodel for the modeling of requirements, but also the collection, optimisation and information extraction functionalities for multi-source data, namely remote sensing and in-situ.

With the proposed system functionalities, the necessary artifacts for the last phase of our proposed approach, namely the ecosystem resilience analysis, are consequently automatically generated.

Thus, to analyse the ecosystem resilience, we propose a final system functionality to create an evolution graph with the computed values of the *properties* of interest, compliant and based on the *DREF resilience framework* [7].

3.3.1 Evolution Graph

To generate the *DREF* evolution graph, the stakeholder requirements model, compliant with the proposed metamodel, is once again used. Therefore, model transformations are applied to extract the specified requirements regarding the *properties* of interest, the *indexed nominal satisfiability* values and the associated *tolerance thresholds*, for the analysis of resilience. Moreover, the *property* values obtained with the artifacts generated with the system functionalities presented in the Section 3.2 are drawn in the evolution graph.

Additionally, to allow the assessment of the *ecosystem* resilience evolution over time in relation to the computed satisfiability values of a *property*, the indexed *nominal satisfiability* and the associated *tolerance thresholds* values corresponding to a given *property* are also drawn in the evolution graph.

Finally, with the intent of allowing additional use cases with the generated ecosystem resilience analysis data, the proposed system exports not only the evolution graph but the generated raw data as well.

3.3.2 Analysis

To analyse the ecosystem resilience with respect to the specified *properties* of interest, the stakeholder reviews the evolution graph generated by the proposed system.

Thus, the stakeholder can determine if the computed satisfiability values of a *property* are converging towards the specified nominal satisfiability values or within the *tolerance threshold*, to analyse if the *properties* of interest values are indeed impacting the *ecosystem* resilience or not.

Eventually, the generated raw data can be used with geographic information systems to allow different usage applications.

Chapter 4

Case Study

4.1 Introduction

In this chapter, we present a complete case study in order to experiment and illustrate our proposed *Ecosystem Resilience Analysis* approach.

Hence, to illustrate the requirements specification and the multiple *Ecosystem Resilience Analysis* activities of our approach, we focus on the field of *AI for social good* with a case study targeting a system capable of analysing the resilience of the Luxembourg territory, using software engineering, model-driven engineering and artificial intelligence methods. The case study is relevant to the compliance with the Paris Agreement on the reduction of greenhouse gases emissions, setting a global framework to avoid dangerous climate change by limiting global warming. Since the model transformations have not been implemented in the proposed approach, due to time constraints of this thesis, the *"Intelligent Observers Generation"* is not applied automatically through the presented model transformations for the dataset collection and optimisation. Instead we propose a "manual" approach in the context of this thesis.

As a result of the Paris Agreement, nearly 200 countries ratified the targets to hold by 2050 the average global temperature increase to below $2^{\circ}C$ and to pursue efforts to limit the temperature increase to below an ambitious $1.5^{\circ}C$, above pre-industrial levels.

Among the different climate change mitigation plans, the most imperative mitigation focus on the increase of the carbon sequestration and on the reduction of CO_2 emissions generated in the process of energy production and consumption. For instance, since the Third Industrial Revolution which began in the '50s in the 20th century, CO_2 emissions have increased drastically, as shown in Figure 4.1.

With the increase of CO_2 emissions, the parties involved in the Paris Agreement are responsible for the implementation at national level of climate change mitigation plans, but do not face sanctions in case of breaches.

The aim of our case study is to focus on the Luxembourg territory resilience analysis to CO_2 with the intent of predicting at the target dates (2030, 2050) the Luxembourgish compliance with the Paris Agreement.

Therefore, we concentrate on the capacity of the Luxembourgish territory to capture and store atmospheric CO_2 (CO_2 absorption/sequestration) with the computation of the land cover in the Luxembourgish territory in 2021. With land cover, we aim to compute an estimation of the potential of the territory to absorb CO_2 and additionally, we concentrate on the evolution of the national CO_2 emissions, from 1990 to 2020, to predict and analyse the future ecosystem resilience and the compliance with the Paris Agreement.

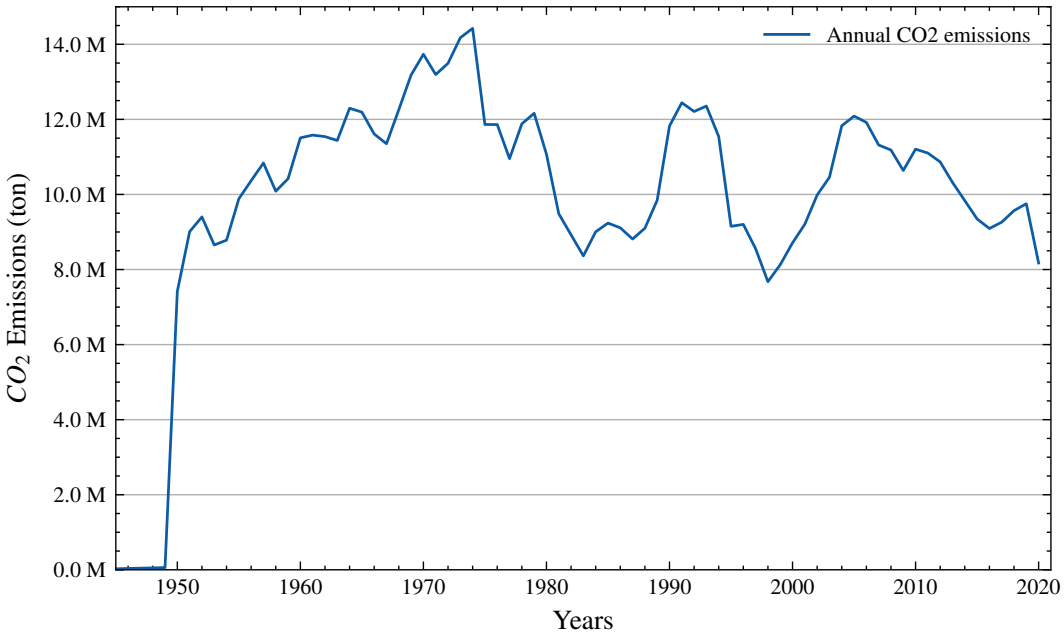


FIGURE 4.1: Annual carbon dioxide (CO_2) emissions since 1945 until 2020, from fossil fuels and industry in Luxembourg. Source: The Global Carbon Project fossil CO_2 emissions dataset [77]

4.2 Requirements Specification

With the requirements elicitation of this case study, we want to illustrate the usage proposed system approach with multi-source data, in particular with a remote sensing dataset and an in-situ dataset. Therefore, we specify the following requirements for the ecosystem resilience analysis:

(i) The ecosystem resilience shall be analysed from the 1st January 1990 until the 13th December 2050 (UTC +1 timezone) in the Luxembourgish territory.

(ii) The property of interest shall be the total CO_2 net emissions in Luxembourg.

(iii) To analyse the resilience of the Luxembourgish *ecosystem* with the property of interest, the CO_2 emissions and CO_2 sequestration properties in Luxembourg shall be evaluated.

(iv) CO_2 emissions shall be evaluated from 1st January 1990 until the 13th December 2020 using an in-situ dataset containing at least updated weekly data. Starting from the 14th December 2020, the satisfiability values of CO_2 emissions shall be predicted until the 13th December 2050.

(v) The dataset for the CO_2 emissions can contain data with a time margin of 1 week.

(vi) For the target year of 2030, the values of CO_2 emissions in Luxembourg shall be evaluated with a nominal satisfiability of 7094002.8, corresponding to the CO_2 target value of -40% for 2030, with respect to CO_2 levels from 1990. Additionally, a tolerance margin of 10% shall be allocated to the specified nominal satisfiability.

(vii) For the target year of 2050, the values of CO_2 emissions in Luxembourg shall be evaluated with a nominal satisfiability of 2364667.6, corresponding to the CO_2 target value of -80% for 2050, with respect to CO_2 levels from 1990. Additionally, a tolerance margin of 20% shall be allocated to the specified nominal satisfiability.

(viii) To analyse the property of interest evolution from 2020 until 2029, the values of CO_2 emissions in the date interval shall be evaluated with a nominal satisfiability of 9458670.4. Moreover, a tolerance margin of 20% shall be allocated to the specified nominal satisfiability.

(ix) The CO_2 sequestration property shall be evaluated with data between the 1st January 2021 until the 13th December 2021, using remote sensing data containing at least the *RGB* band information.

(x) The dataset for the CO_2 sequestration property shall have a maximum revisit time of 5 days.

(xi) The CO_2 sequestration shall be computed for the year 2021 using a land cover classification technique, where each *hectare* of forest in the territory is equal to 3.5 tonnes of CO_2 sequestration, the average typical absorption rates in temperate regions, as defined in [72].

(xii) From 2021 until 2050, a prediction shall allow the evaluation of the future CO_2 satisfiability values, with respect to the land cover classification for the CO_2 sequestration and the historical CO_2 emissions.

To comply with our dedicated requirements elicitation metamodel (Figure 3.2), the requirements were modeled with a metamodel instance using the Unified Modeling Language (presented in Section 2.3.3), as shown in Figure 4.2.

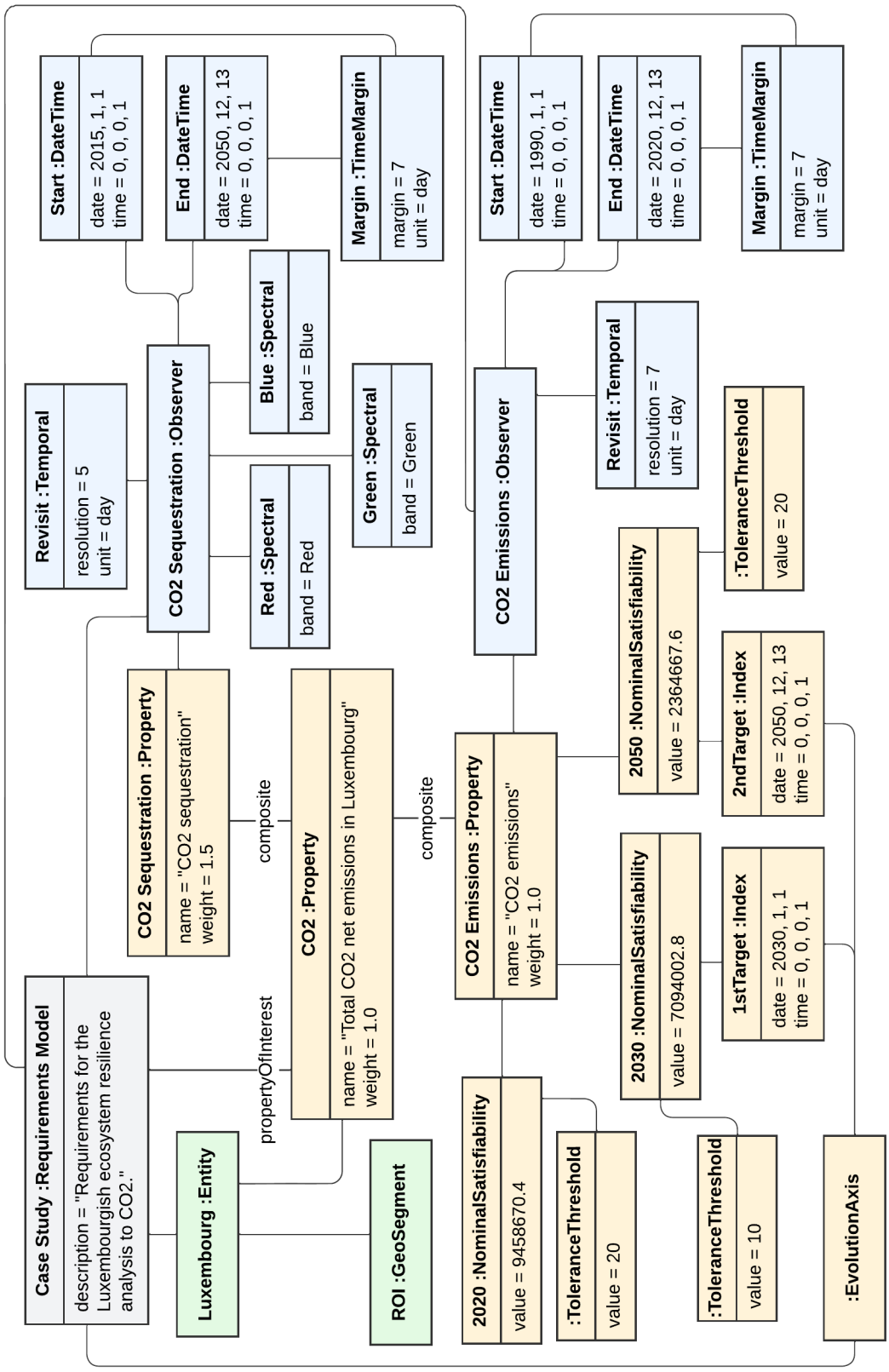


FIGURE 4.2: Requirements Model for the Luxembourgish ecosystem resilience analysis to CO_2

4.3 Intelligent Observers Generation

In this activity of our approach, the requirements model shown in Figure 4.2 would be used as input through model transformations, as presented in Section 3.2.

However, since model transformations have not been implemented in the proposed approach, automation will not be applied in this activity of the approach. Instead, we will present how the system would behave with model transformations but the actual process will be manual in the context of this thesis.

4.3.1 Dataset Collection for the CO_2 Absorption

With the requirements associated with the CO_2 sequestration/absorption *Property* in the Luxembourgish *Ecosystem*, the system functionality would search for datasets respecting our specified requirements as mentioned in Section 4.2.

Since we have defined a *Temporal* requirement of 5 days and the red, green and blue *Spectral* requirements for the *Observer* of the CO_2 sequestration *Property*, the system collection functionality would assign the Sentinel-2 [78] mission as the most adequate remote sensing data source since it respects all the specified requirements.

The Sentinel-2 is an earth observation mission from the European Copernicus Programme that provides systematic coverage with 13 spectral bands and a temporal (revisit) time as low as 5 days with 2 satellites under cloud-free condition, over the following areas:

- Continental land surfaces, with latitudes 56° South and 82.8° North.
- European Union islands.
- Additional islands greater than 100 km^2
- Coastal waters up to 20 km from the shore.
- The Mediterranean Sea.
- Others, based on requests from the member states of the European Union or from the Copernicus Services.

More specifically, the sensor in the Sentinel-2 covers the following spectral bands:

Hence, a Sentinel-2 dataset compliant with the requirements associated with the CO_2 sequestration *Property* was collected (manually), based on the equivalence between the requirements and the characteristics of the data captured with the Sentinel-2 sensor.

Band	**Spatial Resolution** (*meter*)	**Central Wavelength** *(nanometer)*
B01 - Aerosols	60	443
B02 - Blue	10	490
B03 - Green	10	560
B04 - Red	10	665
B05 - Red edge 1	20	705
B06 - Red edge 2	20	740
B07 - Red edge 3	20	783
B08 - NIR	10	842
B08A - Red edge 4	20	865
B09 - Water vapor	60	945
B10 - Cirrus	60	1375
B11 - SWIR 1	20	1610
B12 - SWIR 2	20	2190

TABLE 4.1: Bands covered with the Sentinel-2 multispectral imager sensor, with their corresponding spatial resolution and central wavelength.

4.3.2 Dataset Optimisation for the CO_2 Absorption

Since the primary objective of the Sentinel-2 mission is to provide high resolution satellite data for climate change and land cover and land use monitoring, the dataset associated with the mission contains more spectral bands than the specified red, green and blue (*RGB*) bands in the requirements.

Hence, we have optimised the dataset by reducing its dimensionality and complexity with the application of the *Principal Component Analysis* method (see Section 3.2.2), calculating a projection of the original data into the spectral bands defined in the metamodel instance. Thus, the number of spectral dimensions in the dataset were reduced and as a result, the potential to generalize with a neural network architecture was increased [79]. The result can be seen in Figure 4.3.

4.3.3 Information Extraction for the CO_2 Absorption

Due to the fact that the optimised Sentinel-2 dataset required a neural network architecture for the land cover classification to extract information regarding the potential of the Luxembourgish territory for CO_2 absorption, we have created a CNN architecture.

ResNet Architecture

For this task, we have decided to use a special case of a CNN architecture, namely a Deep Residual Network (ResNet), first introduced in 2015 by Sun et al. [80] and used

FIGURE 4.3: The map of Luxembourg, using the collected and optimised Sentinel-2 data, using only the red, green and blue bands, combined as natural colors. For visualisation purposes, the bounding regions not part of the territory of Luxembourg have been removed.

for deep learning computer vision applications such as object detection or image segmentation, among others.

The main difference between a "traditional" CNN and a ResNet architecture is that the latter has been created to correct the “vanishing gradient” [81] problem that may occur in CNNs with thousands of convolutional layers. As presented in Section 2.4.2, during the training of a neural network the weights are being updated to make the loss function find a value as small as possible, with the intent of creating the best set of weights to improve the prediction or classification. However, if the neural network contains many layers in the architecture, the "vanish gradient" occurs during the backpropagation process, when the gradient becomes extremely small and disappears, constraining the optimisation process during the training of the neural network. ResNet CNN architectures solve this problem with shortcut connections (presented shortly) and therefore are widely used.

There exists various ResNet variants, which follow the same architecture concept

but instead, have a different numbers of layers (depth). Variants of ResNet include *ResNet-18*, *ResNet-34*, *ResNet-50*, *ResNet-101*, *ResNet-110*, *ResNet-152*, *ResNet-164*, *ResNet-1202*. To extract information for the CO_2 Absorption *Property* in our case study, we have decided to use the *ResNet-152* variant, containing 152 layers.

At the core of a ResNet architecture exists the concept of *residual block*, with shortcut connections. Residual blocks are the essential building blocks of ResNet architectures and the difference between a ResNet and a traditional CNN.

Residual blocks, compared to other network layers, feed not only the next layer but also the next two or three layers, bypassing a few layers in between with shortcut connections as depicted in Figure 4.4.

Such concept allows the training of much deeper neural networks without vanishing gradient problems, and hence, are preferred for image tasks.

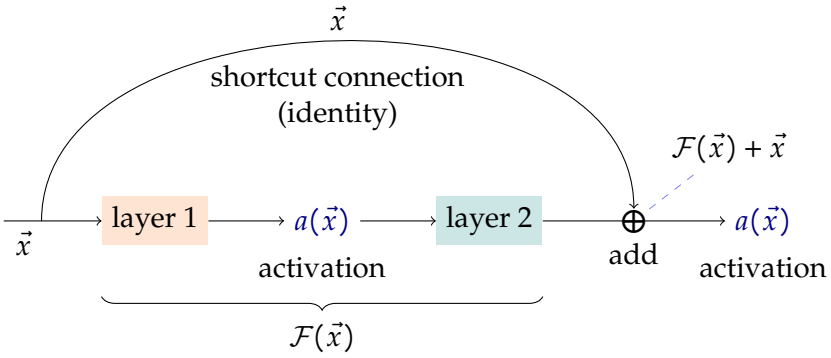


FIGURE 4.4: Illustration of a skip connection in a residual block. Inspired by *"Deep Residual Learning for Image Recognition"* [80].

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

FIGURE 4.5: Some of the building blocks (in brackets) of the ResNet architecture design, with the number of the blocks in the different variants of the network. Source: *"Deep Residual Learning for Image Recognition"* [80]

ResNet-152 Architecture for Land Cover

To undertake the land cover classification, we have applied a ResNet-152 architecture as it achieves the best accuracy among ResNet variants [80].

The model of the architecture, represented in Figure 4.12, has been first designed with PyTorch [67] (illustrated in Listings 4.1, 4.2 and was later implemented using the *Fastai* [82] deep learning library, not only due to its high-level components but also because of its ease of use, flexibility and performance.

To train the architecture, we have used the EuroSAT [83] dataset and deep learning benchmark for Land Use and Land Cover (LULC) classification tasks.

FIGURE 4.6: Sample image of the Eurosat dataset [83].

The dataset contains images from European countries captured with the Sentinel-2 satellite in 13 spectral bands (presented in Table 4.1). More specifically, the dataset is generated with 27000 labeled and geo-referenced image patches with a measure of 64 x 64 pixels, with 2000 to 3000 images per class, distributed among 10 different classes, given in Table 4.2.

With the EuroSAT dataset selected, we started the training of the model architecture, which had 60255296 total trainable parameters and a cross entropy loss function for the classification task. Since the dataset contains 27000 images across 10 different land use and land cover classes, we began by splitting the dataset into two different subsets with a 90/10 split rule, splitting the original dataset into a training dataset with 24300 images and into a validation dataset with 2700 images, only used for later model validation. A batch sample of the images in the training dataset is shown in Figure 4.7.

To speed up the learning process of the 60255296 parameters in the architecture, we have applied *transfer learning* to our model. *Transfer learning* is a machine learning technique where we reuse a model - more specifically its final weights - previously trained on other domains and apply it on our own model, in another domain. The

Classes	Description
Industrial Building	Areas for machine sheltering and working.
Residential Buildings	Areas for human sheltering.
Annual Crop	Areas occupying the soil and yielding harvests for limited months.
Permanent Crop	Areas occupying the soil and yielding harvests for several consecutive years.
River	Areas with flowing watercourse towards the ocean, sea, or lake.
Sea & Lake	Areas surrounded by water.
Herbaceous Vegetation	Areas with herbaceous plants, reaching their full height and producing flowers within one year.
Highway	Areas for human traveling.
Pasture	Areas of land for domesticated livestock.
Forest	Areas of land covered with trees.

TABLE 4.2: Land Use and Land Cover classes and their description in the EuroSAT dataset.

benefit of applying this technique is two-fold. First, it allows to reduce the training time on a new model and second, it can result in lower generalisation errors [51].

For this case-study, we have applied *transfer learning* with a neural network model previously trained for a large image classification challenge called *ImageNet* [84], consisting of more than a 1000000 images among 1000 different classes. Given the important number of image classes in the neural network model for *ImageNet*, it can effectively serve as a generic model to begin the training of our own model, composed of only 10 classes of images.

Thus, with the pre-trained *ImageNet* model acting as a starting point, we started the training of our architecture with *transfer learning*. We first "froze" our model and only trained the last 2 layers with the weights of the *ImageNet* model, during 5 epochs (one complete pass of the training dataset with the ResNet-152 architecture). After only 5 epochs, we reached a top accuracy of 98%, a training loss of 0.6% and an error rate of 2%, as referenced in Table 4.3.

epoch	train_loss	valid_loss	accuracy	error_rate	time
0	0.162469	0.129777	0.958148	0.041852	02:39
1	0.132108	0.103005	0.968148	0.031852	02:29
2	0.069783	0.097046	0.971111	0.028889	02:29
3	0.028569	0.083158	0.981111	0.018889	02:29
4	0.006590	0.081798	0.980000	0.020000	02:33

TABLE 4.3: Training results after applying transfer learning and training the last few layers of our model for 5 epochs.

FIGURE 4.7: Batch sample of the images contained in the training EuroSAT subset.

With such results, we had confidence to improve even further the model. Hence, we applied a method to find the optimal learning rate for our model with the objective of improving the learning by reducing the loss during training. This optimisation step is important because if the learning rate is too slow, the model will take a long time to learn, and if it is too high, the model may never locate the function minima through the gradient descent optimisation, as presented in Section 2.4.2.

After applying the method to find the optimal learning rate, the model was trained with different learning rate values. At the end, we found that the best learning rate would be between 10^{-5} and 10^{-4}, as shown in Figure 4.8.

Using 10^{-5} as the learning rate for the first layer in our model and 10^{-4} for the following layers, we trained our model architecture during 20 epochs and we obtained the results referenced in Table 4.4.

With 98.5556% of accuracy, a training loss of 0.5% and an error rate of 1.4% after

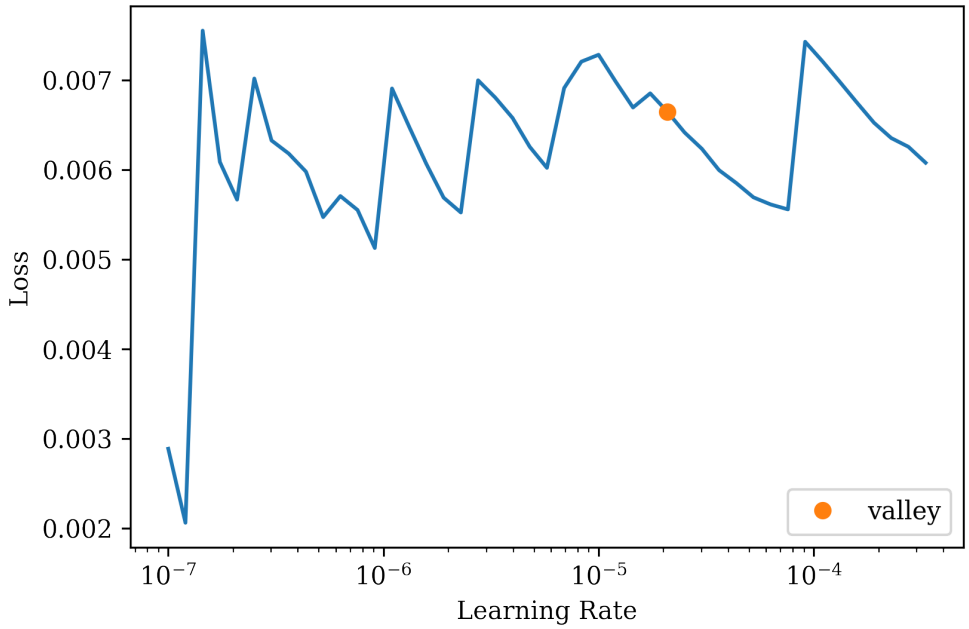


FIGURE 4.8: Learning rate finder result.

training our model, we reached a State-of-the-Art model accuracy using the RGB images in the EuroSAT dataset.

Hence, compared to the accuracy scores on the original EuroSAT paper [83], we obtained a top-1 classification accuracy in the 90/10 training split rule in the dataset benchmarking. Additionally, compared to other State-of-the-Art models present in the online and up-to-date benchmarking leaderboard[1] such as [85], we also achieved a top-1 accuracy with respect to other AI models on the EuroSAT benchmark, using only RGB images.

epoch	train_loss	valid_loss	accuracy	error_rate	time
0	0.008711	0.080200	0.981852	0.018148	02:27
1	0.010479	0.083087	0.981111	0.018889	02:28
2	0.006459	0.080294	0.983333	0.016667	02:30
3	**0.005026**	**0.076134**	**0.985556**	**0.014444**	**02:34**
.....	...	...	...	...	
18	0.000658	0.104153	0.982222	0.017778	02:30
19	0.000172	0.104982	0.982593	0.017407	02:25

TABLE 4.4: Training results for our model with transfer learning and an optimised learning rate.

To better illustrate the robustness and performance of our model in the classification of land use and land cover, we plotted a confusion matrix, show in Figure 4.9.

[1]State-of-the-Art leaderboard for the EuroSAT benchmarking is available in the following URL: https://paperswithcode.com/sota/image-classification-on-eurosat

Moreover, a sample of the classification results on the validation subset is shown in Figure 4.10 and the top losses are shown in Figure 4.11.

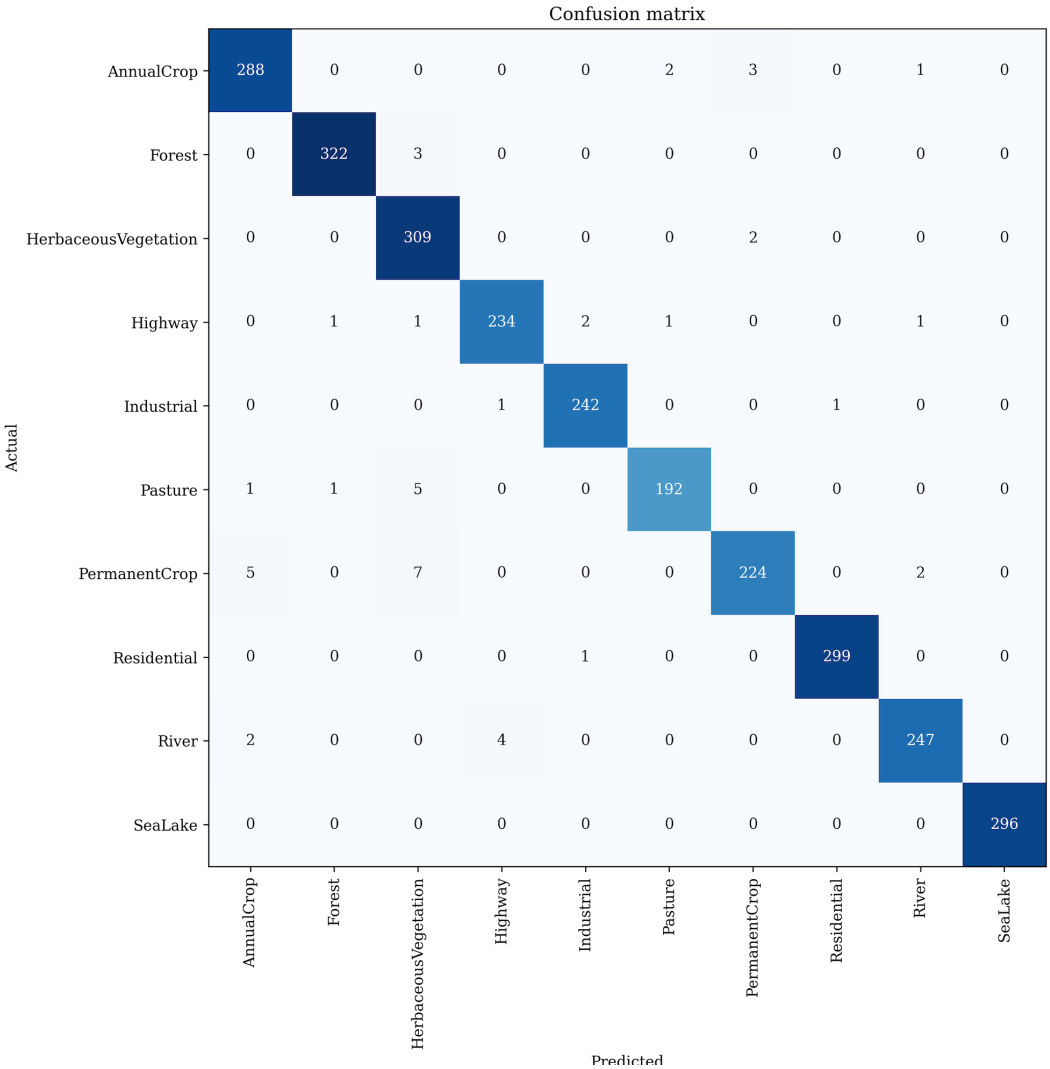


FIGURE 4.9: Confusion matrix of our model for land use and land cover classification.

FIGURE 4.10: Sample of the classification results on the validation subset with our trained model.

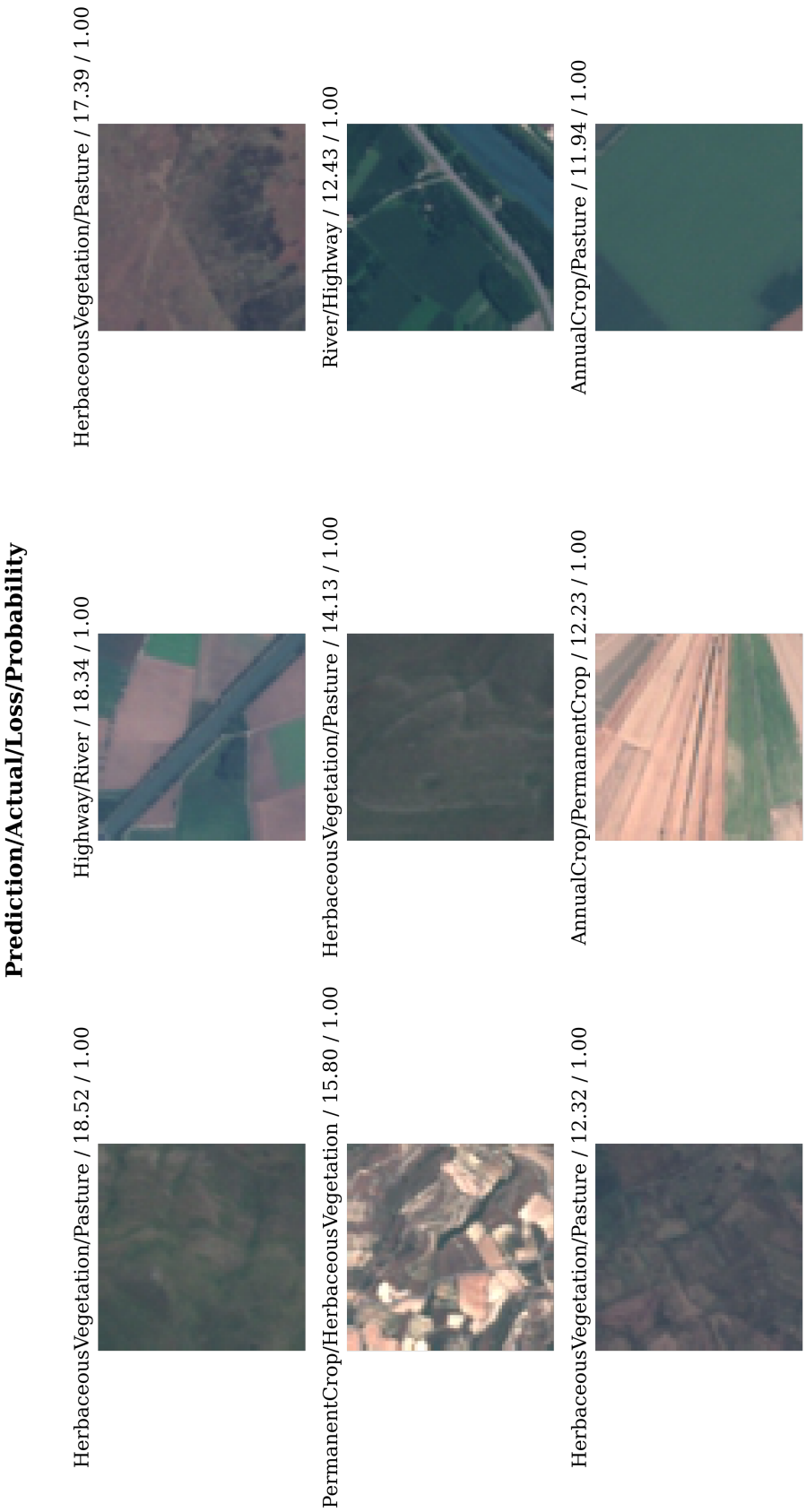


FIGURE 4.11: Top losses obtained during classification on the validation subset with our trained model.

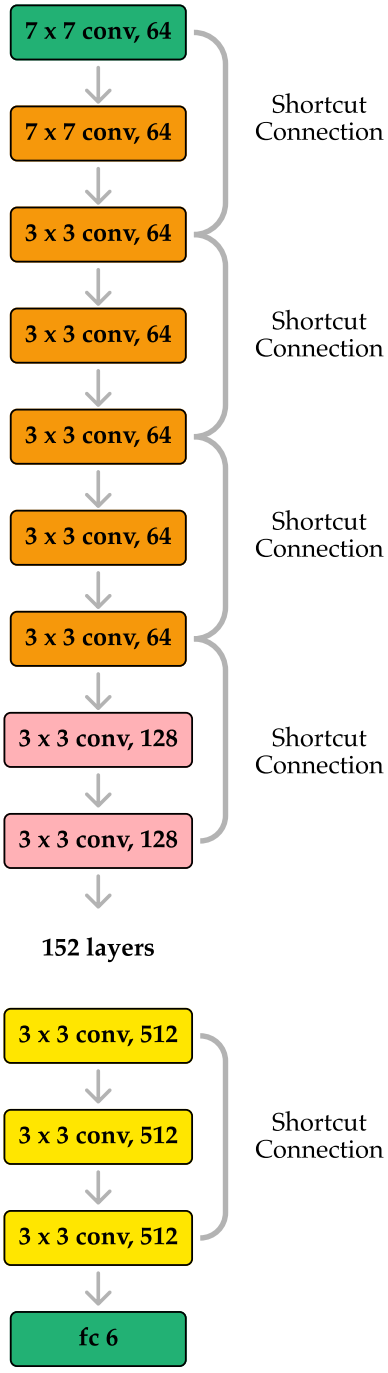


FIGURE 4.12: Representation of the ResNet-152 architecture used for land cover classification.

```python
import torch
from torch import nn

class ResidualBottleneckBlock(nn.Module):
    def __init__(self, in_channels, out_channels, downsample):
        super().__init__()
        self.downsample = downsample
        self.conv1 = nn.Conv2d(in_channels, out_channels//4, kernel_size=1, stride=1)
        self.conv2 = nn.Conv2d(out_channels//4, out_channels//4, kernel_size=3, stride=2 if downsample else 1, padding=1)
        self.conv3 = nn.Conv2d(out_channels//4, out_channels, kernel_size=1, stride=1)
        self.shortcut = nn.Sequential()
        if self.downsample or in_channels != out_channels:
            self.shortcut = nn.Sequential(
                nn.Conv2d(in_channels, out_channels, kernel_size=1, stride=2 if self.downsample else 1),
                nn.BatchNorm2d(out_channels)
            )
        self.bn1 = nn.BatchNorm2d(out_channels//4)
        self.bn2 = nn.BatchNorm2d(out_channels//4)
        self.bn3 = nn.BatchNorm2d(out_channels)

    def forward(self, input):
        shortcut = self.shortcut(input)
        input = nn.ReLU()(self.bn1(self.conv1(input)))
        input = nn.ReLU()(self.bn2(self.conv2(input)))
        input = nn.ReLU()(self.bn3(self.conv3(input)))
        input = input + shortcut
        return nn.ReLU()(input)
```

LISTING 4.1: Listing with the package imports and the implementation of the ResNet Residual Block, inspired by [80].

```
class ResNet(nn.Module):
    def __init__(self, in_channels, residualBlock, repeat):
        super().__init__()
        self.layer0 = nn.Sequential(
            nn.Conv2d(in_channels, 64, kernel_size=7, stride=2, padding=3),
            nn.MaxPool2d(kernel_size=3, stride=2, padding=1),
            nn.BatchNorm2d(64),
            nn.ReLU())
        filters = [64, 256, 512, 1024, 2048]
        self.layer1 = nn.Sequential()
        self.layer1.add_module('conv2_1', residualBlock(filters[0], filters[1],
            downsample=False))
        for i in range(1, repeat[0]):
                self.layer1.add_module('conv2_%d'%(i+1,), residualBlock(filters[1],
                    filters[1], downsample=False))
        self.layer2 = nn.Sequential()
        self.layer2.add_module('conv3_1', residualBlock(filters[1], filters[2],
            downsample=True))
        for i in range(1, repeat[1]):
                self.layer2.add_module('conv3_%d' % (i+1,), residualBlock(filters[2],
                    filters[2], downsample=False))
        self.layer3 = nn.Sequential()
        self.layer3.add_module('conv4_1', residualBlock(filters[2], filters[3],
            downsample=True))
        for i in range(1, repeat[2]):
            self.layer3.add_module('conv2_%d' % (i+1,), residualBlock(filters[3],
                filters[3], downsample=False))
        self.layer4 = nn.Sequential()
        self.layer4.add_module('conv5_1', residualBlock(filters[3], filters[4],
            downsample=True))
        for i in range(1, repeat[3]):
            self.layer4.add_module('conv3_%d'%(i+1,), residualBlock(filters[4], filters
                [4], downsample=False))
        self.gap = torch.nn.AdaptiveAvgPool2d(1)
        self.fc = torch.nn.Linear(filters[4], outputs)

    def forward(self, input):
        input = self.layer0(input)
        input = self.layer1(input)
        input = self.layer2(input)
        input = self.layer3(input)
        input = self.layer4(input)
        input = self.gap(input)
        input = torch.flatten(input, start_dim=1)
        input = self.fc(input)
        return input

resnet152 = ResNet(3, ResidualBottleneckBlock, [3, 8, 36, 3])
```

LISTING 4.2: Listing with the implementation of the ResNet-152 architecture, inspired by [80].

Applying the Model on the Collected & Optimised Dataset

After the training and validation of our ResNet-152 model for land use and land cover classification, we could start the classification on the collected and optimised Sentinel-2 dataset for the Luxembourgish territory. However, since the optimised dataset had a size of $7366 \times 6850 \times 3$ and our model accepted only inputs with the size of $64 \times 64 \times 3$, we were required to create small image "patches" with the size of the model input.

Thus, with the Python code[2] in Listing 4.3, we created 106114 image patches to classify the land cover of Luxembourg with our ResNet-152 model. A sample of the generated image patches is shown in Figure 4.13.

```
1  import numpy as np
2  from patchify import patchify
3  from PIL import Image
4  import cv2
5
6  image = Image.open('./lu.tif')
7  print(image.format, image.size, image.mode)
8  # 7366, 6850, 3
9  lu_np = np.array(image)
10 patches = patchify(lu_np, (64,64, 3), step=64)
11 for i in range(patches.shape[0]):
12     for j in range(patches.shape[1]):
13         single_patch_img = patches[i, j, 0, :, :, :]
14         if not cv2.imwrite('./patches/' + 'image_' + '_'+ str(i)+str(j)+'.jpg',
               single_patch_img):
15             raise Exception("Image cannot be written.")
```

LISTING 4.3: Listing with the image patch implementation for the collected dataset.

With the generated image patches, we used the trained model to predict the land use and land cover classes (presented in Table 4.2) on the collected and optimised Sentinel-2 dataset, with images covering the Luxembourgish territory.

After classifying the 106114 image patches, the results (referenced in the Table 4.5) shown that $\approx$ 25.65% of the image patches had the land use and land cover class classification with the `Forest` class. Since the Luxembourg territory has an area of $2586km^2$, our model classified that $\approx 663km^2$ of the area were forests.

Among the biggest classification "surprises" were the number of classified images with the `SeaLake` land use and land cover class. After investigation, we found that

[2]The `patchify` library imported on line number 2 is an open-source library to split images into small and overlappable patches. The library can be found in the following URL: `https://pypi.org/project/patchify/`

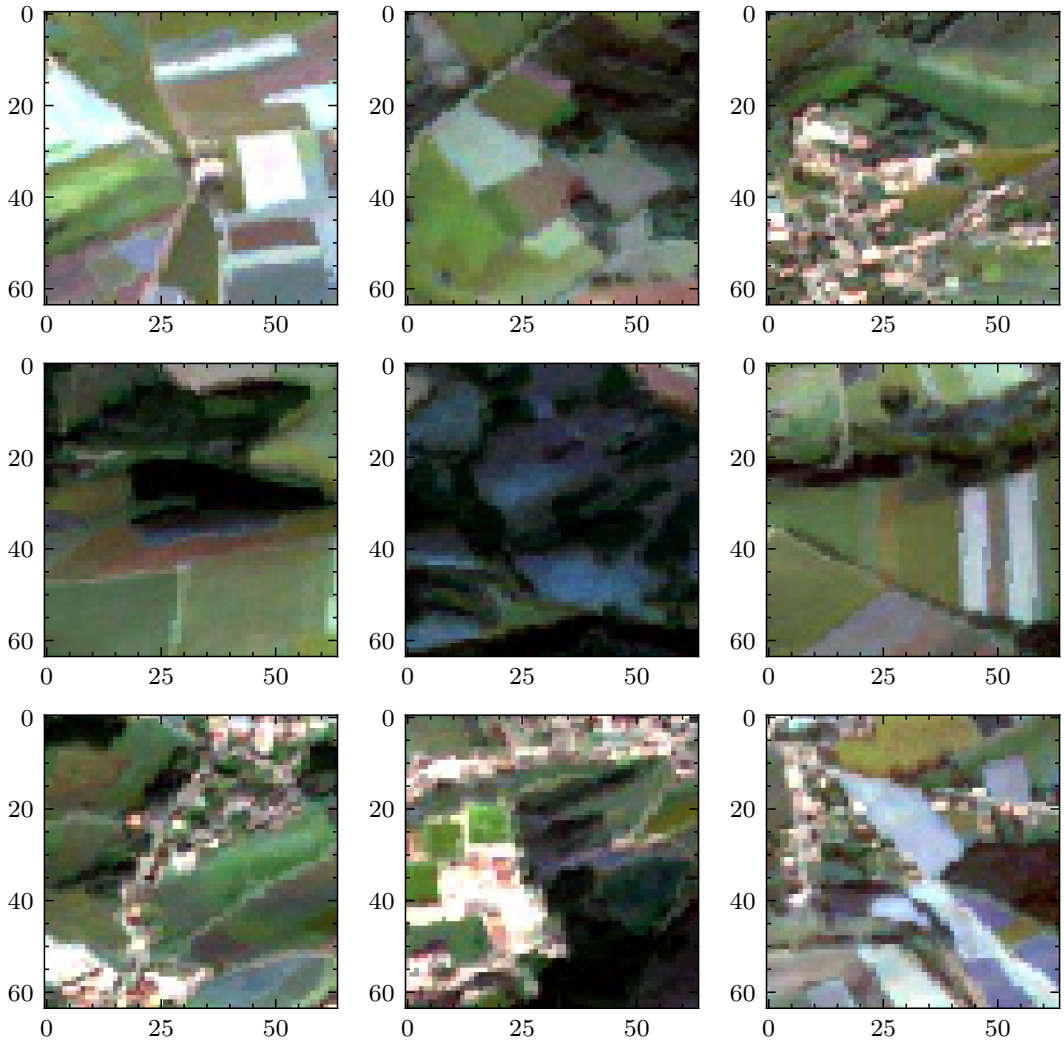


FIGURE 4.13: Generated image patches with a size of $64 \times 64 \times 3$ from the collected Sentinel-2 dataset covering the territory of Luxembourg. Patches used to predict land cover with our ResNet-152 model.

thousands of image patches were created with a black color. During the patch generation, portions of the original dataset were filled with a black color since originally, portions of the image did not contain any pixel information. Thus, by adding a black color in the 3 channels of the image a perfect square image with full pixel information was available.

Hence, patch images with a size of $64 \times 64 \times 3$ were perfectly created. An illustration of the problem is shown in Figure 4.14. The downside of the method was the number of wrong classifications with the `SeaLake` land use and land cover class.

Therefore, if we do not include the number of misclassified `SeaLake` classes to estimate the total area of forests in Luxembourg, we achieve a total of $\approx 35.62\%$ of forest areas or a total of $\approx 921 km^2$ in the territory.

Class	**Amount**
AnnualCrop	4520
Forest	27223
HerbaceousVegetation	12128
Highway	503
Industrial	289
Pasture	6321
PermanentCrop	5125
Residential	17111
River	3187
SeaLake	29707

TABLE 4.5: Land Use and Land Cover (LUL) results after using the trained model to classify LULC on the collected dataset.

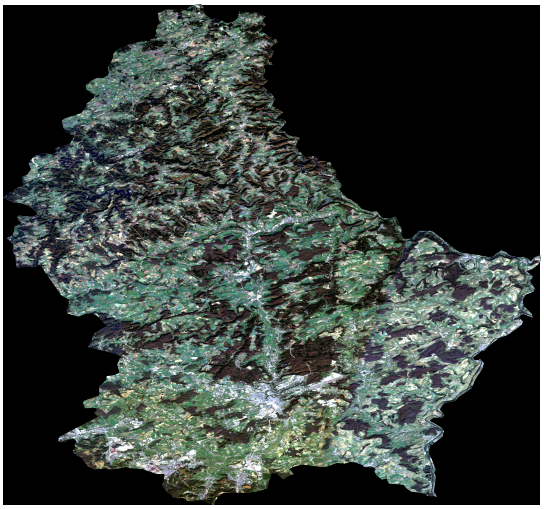


FIGURE 4.14: Illustration of the patch generation issue where a black color was added to allow the export of square image patches.

Based on [72], the typical CO_2 absorption rates in temperate regions are 3.5 tonnes per hectare of forest. With respect to the results of our model, we can estimate a total of $\approx 921 km^2$ or $\approx$ 92100 hectare of forests in the Luxembourgish territory.

Thus, we assume a maximum of 322350 tonnes of CO_2 absorbed by the Luxembourgish forests.

4.3.4 Dataset Collection for the CO_2 Emissions

With the requirements associated with the CO_2 Emissions *Property* in the Luxembourgish *Ecosystem*, the system functionality of the proposed system would search for datasets respecting our specified requirements as mentioned in Section 4.2.

We have defined a *start DateTime* requirement to search for data from midnight of the 1st January 1990 until the midnight of the 13th December 2020, our *end DateTime* requirement for the *Observer*. Additionally, we specified a *Temporal* requirement of 7 days, meaning that data shall have been collected at least every week during the period of time between the *start* and *end DateTime* requirements.

Due to those *Observer* requirements for the the CO_2 Emissions, the system collection functionality would retrieve an open-access dataset within the *European Environment Agency data platform*, which is compliant with the specified requirements. The retrieved dataset, shown in Figure 4.15, contains weekly updated values corresponding to the CO_2 emissions in the Luxembourgish territory from the 1st January 1990 until the end of the year 2020

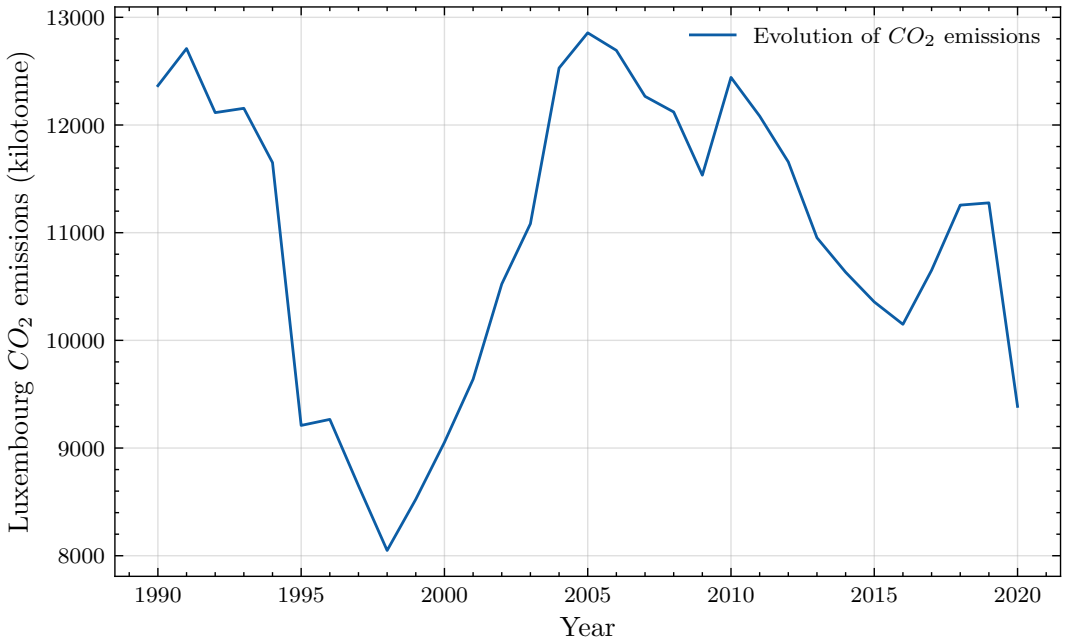


FIGURE 4.15: Evolution of CO_2 emissions with data from the collected dataset from the European Environment Agency.

4.3.5 Dataset Optimisation for the CO_2 Emissions

Since the dataset from the *European Environment Agency data platform* contained only weekly observation numerical values in relation to the CO_2 emissions in Luxembourg, further optimisations were not necessary to be made.

4.3.6 Information Extraction for the CO_2 Emissions

To predict the satisfiability values of CO_2 emissions from the 14th December 2020 until the 13th December 2050, we have applied a linear regression technique, namely a least squares method allowing the generation of new observations for the *ecosystem resilience analysis*. Hence, the regression has been applied with respect to the relationship between the year and the CO_2 emissions variables in the collected dataset.

Since we are predicting the Luxembourgish compliance to the Paris Agreement, only adopted on the 12th December 2015, we have only take into account the evolution of the emissions since the adoption date (2015-2020) for the prediction of the satisfiability values in the future.

The prediction of the CO_2 emissions until 2050 is shown in Figure 4.16.

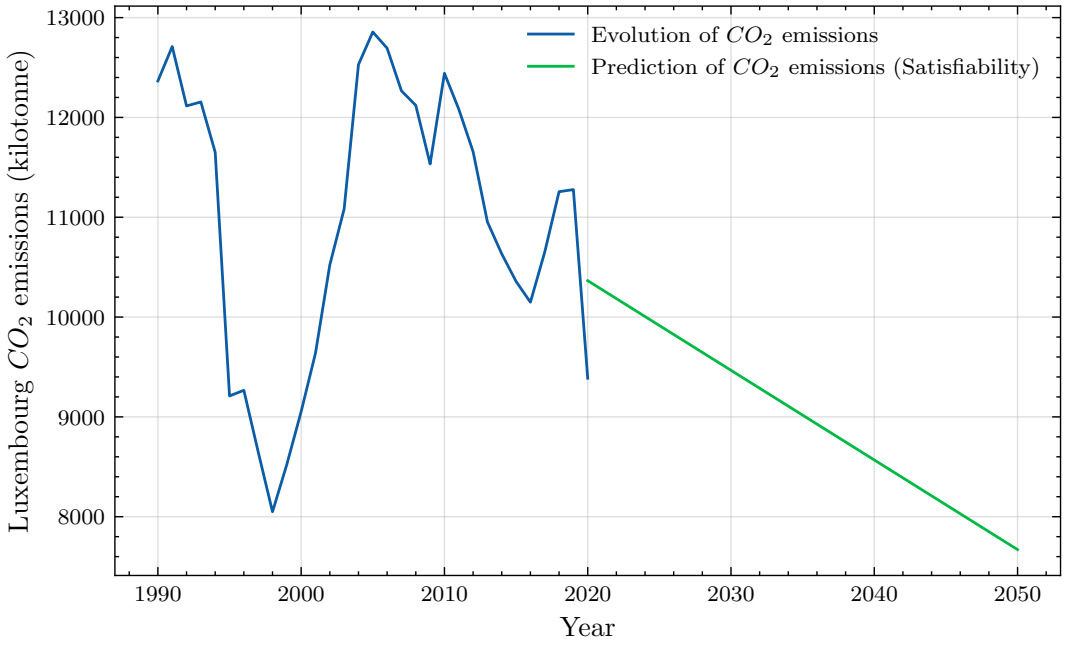


FIGURE 4.16: Predicition of the CO_2 emissions from 2020 until 2050.

The information extraction for the CO_2 absorption *Property* has allowed to estimate a maximum of 322350 tonnes (322.35 kilotonne) of CO_2 absorbed by the Luxembourgish forests, based on the collected remote sensing information from 2020.

If we take into account that maximum value for the CO_2 absorption, we can integrate it to compute an additional prediction regarding the net CO_2 emissions from 2020 until 2050. The prediction of the net CO_2 emissions until 2050 is shown in Figure 4.17.

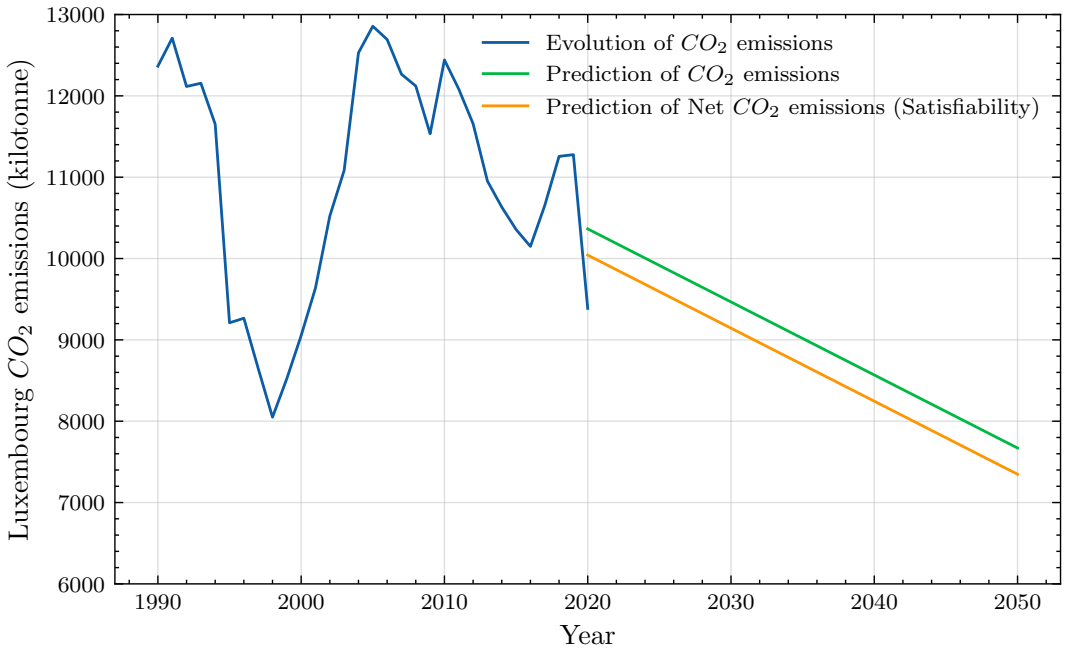


FIGURE 4.17: Predicition of the Net CO_2 emissions from 2020 until 2050.

Hence, we can clearly see how important are forests for CO_2 absorption and how they help in the moderation of the Earth's carbon balance.

4.4 *Ecosystem Resilience Analysis* Results

In this section, we present the result of the ecosystem resilience analysis in relation with the property of interest of our case study, namely the total CO_2 net emissions in Luxembourg.

With the generation of the *DREF* [7] evolution graph shown in Figure 4.18, we have compiled the information extraction from both the *Properties* of the case study, namely the CO_2 absorption and CO_2 emissions. Hence, with the results from the previous activity of our proposed *Ecosystem Resilience Analysis* approach, we can determine a possible compliance with the Paris Agreement from 2020 until 2029, with net CO_2 emissions values respecting the target of -20% compared to 1990 levels.

As shown in the Figure 4.18, the computed satisfiability respected the tolerance margin assigned from 2020 until 2030. After 2025, the system predicted a more important decrease of CO_2 net emissions compared to the target of -20% for the interval of 2020 until 2030, where the satisfiability values were predicted to be below the specified nominal satisfiability.

However, with the target of -40% of CO_2 net emissions from 2030 until 2050, we can verify with the generated *DREF* evolution graph a nonconformity with respect to the 10% of tolerance margin specified in the requirements model.

As thus, the target for 2030 appears to be only attainable in 2045, when the predicted satisfiability values respects the tolerance margin specified for interval of time between 2030 and 2050. However, the nominal satisfiability value assigned from 2030 until 2050 is not accomplished. Likewise, we predict a noncompliance with the nominal satisfiability corresponding to the Paris Agreement target for 2050.

As a consequence of the predicted satisfiability values for the total CO_2 net emissions *property*, we can estimate at most a reduction of 40% in relation to 1990 levels for 2050.

Thus, we can conclude that the ecosystem may not be resilience to the CO_2 net emissions at the Paris Agreement targets of 2030 and 2050.

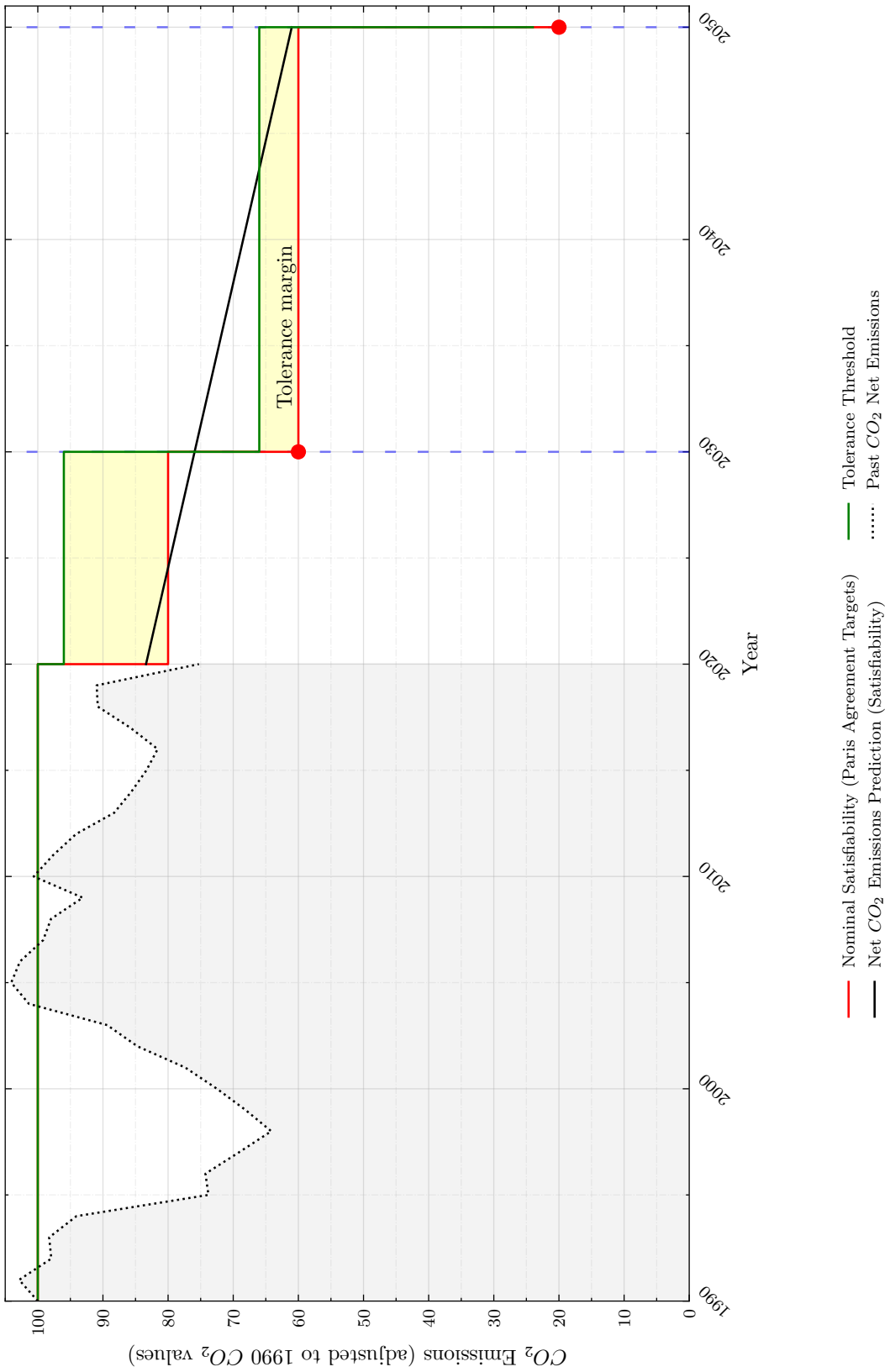


FIGURE 4.18: Ecosystem Resilience Analysis to CO_2 net emissions, with respect to the Paris Agreement targets

Chapter 5

Conclusion

In this thesis, we have presented our approach for the development of systems analysing the resilience of ecosystems using Earth observation multi-source data such as remote sensing or in-situ, following a model-driven engineering approach coupled with artificial intelligence (MDE4AI) methods.

We have presented how the requirements engineering phase in a model-driven engineering context can be associated for the specification of Earth observation multi-source data. Therefore, we have created a dedicated metamodel for the specification of requirements, allowing stakeholders to model the *Ecosystem Resilience Analysis* requirements in the system under development using the presented concepts such as *entities*, *properties* and *observers*. Such requirements are in turn used in subsequent activities of the approach for the development of systems analysing the resilience of ecosystems. Additionally, with the specification of requirements, we have shown how the requirements can be used through model transformations to automate the collection of Earth observation datasets as well as their optimisation.

Based on the recent success of the application of artificial intelligence methods on Earth observation data, we have presented how the generation of different neural network architectures such as convolutional neural networks or generative adversarial networks can be automatised to extract informations from the multi-source collected data. Hence, with the proposed *Ecosystem Resilience Analysis* approach, the software system generates intelligent skeletons of neural network architectures to make use of the collected data with artificial intelligence architectures, to evaluate the resilience of the *properties* of interest on the specified *entities*. Lastly, we have presented a final system functionality to create an evolution graph with the computed values of the *properties* of interest, compliant and based on the *DREF resilience framework*.

With the intent of illustrating and experimenting our proposed *Ecosystem Resilience Analysis* approach, we have presented a case study targeting a system capable of analysing the resilience of the Luxembourg territory, relevant to the compliance with the Paris Agreement on the reduction of greenhouse gases emissions.

Chapter 6

Future Work

In this thesis, our main contribution was the development of an approach for the *Ecosystem Resilience Analysis*. Currently, we have created a dedicated metamodel for the requirements specification associated with Earth observation multi-source data for the development of systems analysing the resilience of ecosystems. Hence, stakeholders can create models for the *Ecosystem Resilience Analysis* requirements, compliant with the metamodel. Moreover, in the proposed approach, we describe informally how model transformations can be used to automate the data collection, optimisation and information extraction on Earth observation multi-source data. However, due to time constraints, some limitations have been setup and hence, the automation via model transformations in a model-driven engineering context has not been implemented yet during this thesis. Thus, as a first future work, the model transformations will be formalised and implemented, to allow the complete automation of our approach and consequently, a step towards a complete model-driven engineering method.

Additionally, during the thesis, we found opportunities for improvements regarding the generation of artificial neural networks for information extraction. First, we found that genetic algorithms can be coupled with artificial neural networks to accelerate and automating the learning process to solve problems such as remote sensing image classification and land use and land cover. Hence, as future work, we would like to allow the generation of genetic algorithms for the classification of remote sensing imagery.

Second, despite the powerful capabilities of current machine learning architectures, the usage of Earth observation data such as remote sensing datasets can lead to important increases in computation times. Hence, we have identified another future work related to the machine learning aspect of our approach. Therefore, we plan to add the functionality to apply transfer learning to the proposed architectures with the usage of public pre-trained models, compliant with the stakeholder requirements.

Finally, once we have implemented the functionality to apply transfer learning in the proposed architectures, we intend to propose new remote-sensing scene-classification

methods based on vision transformers. First recognised as the state-of-the-art models in natural language processing, vision transformers have demonstrated recently great capabilities in image classification with their multi-head self-attention mechanism as the main building block of the neural network, allowing the capture of contextual representations between images.

Bibliography

[1] F. S. Chapin *et al.*, *Principles of Terrestrial Ecosystem Ecology*. New York: Springer, 2011.

[2] C. S. Holling, "Resilience and Stability of Ecological Systems," *Annual Review of Ecology and Systematics*, vol. 4, no. 1, pp. 1–23, 1973.

[3] L. H. Gunderson, "Ecological Resilience—In Theory and Application," *Annual Review of Ecology and Systematics*, vol. 31, no. 1, pp. 425–439, 2000.

[4] B. Walker *et al.*, "Resilience, Adaptability and Transformability in Social–Ecological Systems," *Ecology and Society*, vol. 9, no. 2, 2004.

[5] E. A. Rosa *et al.*, "Tracking the Anthropogenic Drivers of Ecological Impacts," *AMBIO: A Journal of the Human Environment*, vol. 33, no. 8, pp. 509–512, Dec. 2004.

[6] IPCC, "Summary for Policymakers," in *Global Warming of 1.5°C: IPCC Special Report on Impacts of Global Warming of 1.5°C above Pre-Industrial Levels in Context of Strengthening Response to Climate Change, Sustainable Development, and Efforts to Eradicate Poverty*. Cambridge: Cambridge University Press, 2022, pp. 1–24.

[7] N. Guelfi, "A Formal Framework for Dependability and Resilience from a Software Engineering Perspective," *Central European Journal of Computer Science*, vol. 1, no. 3, pp. 294–328, 2011.

[8] J. R. Schott, *Remote Sensing: The Image Chain Approach; 2nd Ed.* Oxford: Oxford University Press, 2007.

[9] M. A. Wulder *et al.*, "Current Status of Landsat Program, Science, and Applications," *Remote Sensing of Environment*, vol. 225, pp. 127–147, May 2019.

[10] T. M. Lillesand, *Remote Sensing and Image Interpretation*. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2006.

[11] R. F. Kokaly *et al.*, "USGS Spectral Library Version 7," U.S. Geological Survey, Reston, VA, USGS Numbered Series 1035, 2017.

[12] H. Nagendra *et al.*, "Remote Sensing for Conservation Monitoring: Assessing Protected Areas, Habitat Extent, Habitat Condition, Species Diversity, and Threats," *Ecological Indicators*, vol. 33, pp. 45–59, Oct. 2013.

[13] M. Réjou-Méchain *et al.*, "Local Spatial Structure of Forest Biomass and Its Consequences for Remote Sensing of Carbon Stocks," *Biogeosciences*, vol. 11, no. 23, pp. 6827–6840, Dec. 2014.

[14] G. Patenaude *et al.*, "Synthesis of Remote Sensing Approaches for Forest Carbon Estimation: Reporting to the Kyoto Protocol," *Environmental Science & Policy*, vol. 8, no. 2, pp. 161–178, Apr. 2005.

[15] X. Cui *et al.*, "Application of Remote Sensing to Water Environmental Processes under a Changing Climate," *Journal of Hydrology*, vol. 574, pp. 892–902, Jul. 2019.

[16] E. Chuvieco, Ed., *Earth Observation of Global Change: The Role of Satellite Remote Sensing in Monitoring Global Environment*. New York: Springer, 2008.

[17] D. Rocchini, "Earth Observation for Ecosystems Monitoring in Space and Time: A Special Issue in Remote Sensing," *Remote Sensing*, vol. 7, no. 6, pp. 8102–8106, Jun. 2015.

[18] W. B. Cohen *et al.*, "Landsat's Role in Ecological Applications of Remote Sensing," *BioScience*, vol. 54, no. 6, pp. 535–545, Jun. 2004.

[19] S. Goetz *et al.*, "Advances in Remote Sensing Technology and Implications for Measuring and Monitoring Forest Carbon Stocks and Change," *Carbon Management*, vol. 2, no. 3, pp. 231–244, Jun. 2011.

[20] A. Lausch *et al.*, "Understanding Forest Health with Remote Sensing -Part I—A Review of Spectral Traits, Processes and Remote-Sensing Characteristics," *Remote Sensing*, vol. 8, no. 12, p. 1029, Dec. 2016.

[21] ——, "Understanding Forest Health with Remote Sensing, Part III: Requirements for a Scalable Multi-Source Forest Health Monitoring Network Based on Data Science Approaches," *Remote Sensing*, vol. 10, no. 7, p. 1120, Jul. 2018.

[22] G. J. Toomer, *Ptolemy's Almagest*, Jan. 1998.

[23] Object Management Group (OMG), "Unified Modeling Language (UML) Specification, Version 2.5.1," 2017.

[24] E. W. Dijkstra, "The Humble Programmer," in *Edsger Wybe Dijkstra: His Life, Work, and Legacy*, 1st ed. New York, NY, USA: Association for Computing Machinery, Jul. 2022, vol. 45, pp. 5–18.

[25] P. Naur *et al.*, *Software Engineering: Report of a Conference Sponsored by the NATO Science Committee, Garmisch, Germany, 7-11 Oct. 1968, Brussels, Scientific Affairs Division, NATO*, 1969.

[26] I. Sommerville, *Software Engineering*, tenth edition ed. Pearson, 2016.

[27] W. W. Royce, "Managing the Development of Large Software Systems: Concepts and Techniques," in *Proceedings of the 9th International Conference on Software Engineering*, ser. ICSE '87. Washington, DC, USA: IEEE Computer Society Press, 1987, pp. 328–338.

[28] D. C. Schmidt, "Model-Driven Engineering," *IEEE Computer*, vol. 39, no. 2, p. 9, 2006.

[29] T. Kühne, "Matters of (Meta-) Modeling," *Software & Systems Modeling*, vol. 5, no. 4, pp. 369–385, Nov. 2006.

[30] L. Gammaitoni *et al.*, "F-Alloy: An Alloy Based Model Transformation Language," in *Theory and Practice of Model Transformations*, ser. Lecture Notes in Computer Science, D. Kolovos *et al.*, Eds. Cham: Springer International Publishing, 2015, pp. 166–180.

[31] "OMG: Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification. Version 1.1 (January 2011), http://www.omg.org/spec/QVT/1.1/."

[32] G. Booch *et al.*, "Object-Oriented Analysis and Design with Applications, Third Edition," *ACM SIGSOFT Software Engineering Notes*, vol. 33, no. 5, p. 11:29, Aug. 2008.

[33] J. Rumbaugh *et al.*, "Object-Oriented Modeling and Design," p. 8.

[34] I. Jacobson, *Object-Oriented Software Engineering*. New York, NY, USA: Association for Computing Machinery, 1991.

[35] L. J. Jackson *et al.*, "An Introduction to the Practice of Ecological Modeling," *BioScience*, vol. 50, no. 8, pp. 694–706, Aug. 2000.

[36] P. Palladino, *Entomology, Ecology and Agriculture: The Making of Scientific Careers in North America, 1885-1985*. Psychology Press, 1996.

[37] V. H. Dale, "Opportunities for Using Ecological Models for Resource Management," in *Ecological Modeling for Resource Management*, V. H. Dale, Ed. New York, NY: Springer, 2003, pp. 3–19.

[38] P. Thorbek *et al.*, *Ecological Models for Regulatory Risk Assessments of Pesticides: Developing a Strategy for the Future*. CRC Press, Nov. 2009.

[39] S. Dufour-Kowalski *et al.*, "Capsis: An Open Software Framework and Community for Forest Growth Modelling," *Annals of Forest Science*, vol. 69, no. 2, pp. 221–233, Mar. 2012.

[40] M. G. Turner *et al.*, "Ecosystem Modeling for the 21st Century," *Ecosystems*, vol. 20, no. 2, pp. 211–214, Mar. 2017.

[41] R. Seidl, "To Model or Not to Model, That Is No Longer the Question for Ecologists," *Ecosystems*, vol. 20, no. 2, pp. 222–228, Mar. 2017.

[42] R. M. Scheller *et al.*, "Increasing the Reliability of Ecological Models Using Modern Software Engineering Techniques," *Frontiers in Ecology and the Environment*, vol. 8, no. 5, pp. 253–260, 2010.

[43] D. Mladenoff *et al.*, "Design, Behavior and Applications of LANDIS, an Object-Oriented Model of Forest Landscape Disturbance and Succession," Jan. 1999.

[44] J. McCarthy *et al.*, "A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence, August 31, 1955," *AI Magazine*, vol. 27, no. 4, pp. 12–12, Dec. 2006.

[45] S. J. Russell *et al.*, *Artificial Intelligence: A Modern Approach*, 2nd ed. Pearson Education, 2003.

[46] T. M. Mitchell, *Machine Learning*, 1st ed. USA: McGraw-Hill, Inc., 1997.

[47] A. L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers. II—Recent Progress," in *Computer Games I*, D. N. L. Levy, Ed. New York, NY: Springer New York, 1988, pp. 366–400.

[48] W. S. McCulloch *et al.*, "A Logical Calculus of the Ideas Immanent in Nervous Activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, Dec. 1943.

[49] D. E. Rumelhart *et al.*, "Learning Representations by Back-Propagating Errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986.

[50] H. Robbins *et al.*, "A Stochastic Approximation Method," *The Annals of Mathematical Statistics*, vol. 22, no. 3, pp. 400–407, Sep. 1951.

[51] I. Goodfellow *et al.*, *Deep Learning*, ser. Adaptive Computation and Machine Learning. Cambridge, Massachusetts: The MIT Press, 2016.

[52] J. Mira *et al.*, Eds., *From Natural to Artificial Neural Computation: International Workshop on Artificial Neural Networks, Malaga-Torremolinos, Spain, June 7-9, 1995: Proceedings*, ser. Lecture Notes in Computer Science. Berlin ; New York: Springer-Verlag, 1995, no. 930.

[53] S. Qiu *et al.*, "FReLU: Flexible Rectified Linear Units for Improving Convolutional Neural Networks," *arXiv:1706.08098 [cs]*, Jan. 2018.

[54] D. E. Rumelhart *et al.*, "Learning Internal Representations by Error Propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*. Cambridge, MA, USA: MIT Press, Jan. 1986, pp. 318–362.

[55] I. Goodfellow *et al.*, "Generative Adversarial Nets," in *Advances in Neural Information Processing Systems*, Z. Ghahramani *et al.*, Eds., vol. 27. Curran Associates, Inc., 2014.

[56] B. Jahic *et al.*, "Software Engineering for Dataset Augmentation Using Generative Adversarial Networks," Oct. 2019.

[57] Y. LeCun *et al.*, "Backpropagation Applied to Handwritten Zip Code Recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, Dec. 1989.

[58] W. Zhang *et al.*, "Remote Sensing Image Scene Classification Using CNN-CapsNet," *Remote Sensing*, vol. 11, no. 5, 2019.

[59] C. Zhang *et al.*, "Detecting Large-Scale Urban Land Cover Changes from Very High Resolution Remote Sensing Images Using CNN-Based Classification," *ISPRS International Journal of Geo-Information*, vol. 8, no. 4, p. 189, Apr. 2019.

[60] Z. Zhang *et al.*, "Change Detection between Multimodal Remote Sensing Data Using Siamese CNN," Jul. 2018.

[61] R. Dechter, "Learning While Searching in Constraint-Satisfaction-Problems," in *Proceedings of the Fifth AAAI National Conference on Artificial Intelligence*, ser. AAAI'86. Philadelphia, Pennsylvania: AAAI Press, Aug. 1986, pp. 178–183.

[62] G. Camps-Valls *et al.*, *Deep Learning for Earth Sciences*. Wiley Online Library, 2021.

[63] L. Ma *et al.*, "Deep Learning in Remote Sensing Applications: A Meta-Analysis and Review," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 152, pp. 166–177, Jun. 2019.

[64] D. J. Lary *et al.*, "Machine Learning in Geosciences and Remote Sensing," *Geoscience Frontiers*, vol. 7, no. 1, pp. 3–10, Jan. 2016.

[65] RolnickDavid *et al.*, "Tackling Climate Change with Machine Learning," *ACM Computing Surveys (CSUR)*, Feb. 2022.

[66] A. J. Stewart *et al.*, "TorchGeo: Deep Learning With Geospatial Data," Jun. 2022.

[67] A. Paszke *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems 32*, H. Wallach *et al.*, Eds. Curran Associates, Inc., 2019, pp. 8024–8035.

[68] L. Zhang *et al.*, "Deep Learning for Remote Sensing Data: A Technical Tutorial on the State of the Art," *IEEE Geoscience and Remote Sensing Magazine*, vol. 4, no. 2, pp. 22–40, Jun. 2016.

[69] "ISO/IEC 19510:2013," https://www.omg.org/spec/BPMN/ISO/19510/PDF.

[70] C. Toth *et al.*, "Remote Sensing Platforms and Sensors: A Survey," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 115, pp. 22–36, May 2016.

[71] R. B. Jackson *et al.*, "Increasing Anthropogenic Methane Emissions Arise Equally from Agricultural and Fossil Fuel Sources," *Environmental Research Letters*, vol. 15, no. 7, p. 071002, Jul. 2020.

[72] S. Brown, "Measuring Carbon in Forests: Current Status and Future Challenges," *Environmental Pollution*, vol. 116, no. 3, pp. 363–372, Mar. 2002.

[73] M. Chi *et al.*, "Big Data for Remote Sensing: Challenges and Opportunities," *Proceedings of the IEEE*, vol. 104, no. 11, pp. 2207–2219, Nov. 2016.

[74] R. Bellman, "Dynamic Programming," *Science*, vol. 153, no. 3731, pp. 34–37, Jul. 1966.

[75] J. Pearlman *et al.*, "Hyperion, a Space-Based Imaging Spectrometer," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 41, no. 6, pp. 1160–1173, Jun. 2003.

[76] K. Koonsanit *et al.*, "Band Selection for Dimension Reduction in Hyper Spectral Image Using Integrated InformationGain and Principal Components Analysis Technique," *International Journal of Machine Learning and Computing*, pp. 248–251, 2012.

[77] R. M. Andrew *et al.*, "The Global Carbon Project's Fossil CO2 Emissions Dataset," Oct. 2021.

[78] M. Drusch *et al.*, "Sentinel-2: ESA's Optical High-Resolution Mission for GMES Operational Services," *Remote Sensing of Environment*, vol. 120, pp. 25–36, May 2012.

[79] B. P. P. Garcia-Salgado *et al.*, "Efficient Dimension Reduction of Hyperspectral Images for Big Data Remote Sensing Applications," *Journal of Applied Remote Sensing*, vol. 14, no. 3, p. 032611, Mar. 2020.

[80] K. He *et al.*, "Deep Residual Learning for Image Recognition," Dec. 2015.

[81] R. Pascanu *et al.*, "On the Difficulty of Training Recurrent Neural Networks," Feb. 2013.

[82] J. Howard *et al.*, "Fastai: A Layered API for Deep Learning," *Information*, vol. 11, no. 2, p. 108, Feb. 2020.

[83] P. Helber *et al.*, "EuroSAT: A Novel Dataset and Deep Learning Benchmark for Land Use and Land Cover Classification," Feb. 2019.

[84] J. Deng *et al.*, "Imagenet: A Large-Scale Hierarchical Image Database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*. Ieee, 2009, pp. 248–255.

[85] P. Gómez *et al.*, "MSMatch: Semi-Supervised Multispectral Scene Classification with Few Labels," Aug. 2021.