Unsupervised Illegal Landfills Detection using Land Cover specific Autoencoders

Tesi di Laurea Magistrale in
Mathemathical Engineering - Ingegneria Matematica

Author: **Elena Musiari**

Student ID: 970068
Advisor: Prof. Giacomo Boracchi
Co-advisor: Luca Frittoli, Ph.D.
Academic Year: 2022-23

Abstract

In contemporary society, protecting the environment has become an urgent call to action. Illegal landfill sites not only pose immediate environmental risks, but also have long-term impacts on ecosystems and human health. The detection of illegal waste disposal sites could play a key role in the fight against organised environmental crime. Our objective is to automatically identify and segment illegal landfills in satellite images using deep learning methods. This work is part of the European PERIVALLON project, which aims to prevent illegal activities such as illegal waste disposal.

We address the segmentation problem using an unsupervised anomaly detection method, where normality is defined as the absence of illegal landfills. We employ Autoencoders to learn the normality of satellite images in order to detect any irregular instances of illegal dumping sites. In particular, we generate error maps by computing the pixel-wise error between the original images and the images reconstructed by the Autoencoder. We create binary segmentation masks by applying a threshold to these error maps. Moreover, we apply morphological image postprocessing to the segmentation masks to obtain connected and compact segmented objects, and improve the performance evaluation. Since normality is highly varied, a model trained on the entire dataset is not effective. Therefore, we cluster the land cover types to divide images into subsets, based on the land cover they contain. This method enables us to build land cover specific Autoencoders, achieving superior segmentation performances when compared to models trained on the entire dataset.

Keywords: deep learning, image segmentation, anomaly detection, illegal landfills detection, autoencoder, land cover clusterization

Abstract in lingua italiana

Nella società contemporanea, la tutela dell'ambiente è diventata un appello urgente ad agire. Le discariche abusive non solo comportano rischi ambientali immediati, ma hanno anche un impatto a lungo termine sugli ecosistemi e sulla salute umana. L'individuazione delle discariche illegali potrebbe svolgere un ruolo chiave nella lotta contro la criminalità ambientale organizzata. Il nostro obiettivo è identificare e segmentare in maniera automatica le discariche illegali nelle immagini satellitari utilizzando metodi di deep learning. Questo lavoro fa parte del progetto europeo PERIVALLON, che mira a prevenire attività illegali come lo smaltimento illegale dei rifiuti.

Affrontiamo il problema della segmentazione utilizzando un metodo di rilevamento delle anomalie non supervisionato, in cui la normalità viene definita come assenza di terreni illegali. Utilizziamo degli Autoencoder per apprendere la normalità delle immagini satellitari, al fine di rilevare eventuali casi irregolari di discariche illegali. Calcolando la differenza tra l'immagine originale e quella ricostruita, generiamo le mappe di errore di ricostruzione dell'Autoencoder. Da queste, creiamo maschere di segmentazione binarie applicando una soglia all'errore. Sulle maschere di segmentazione applichiamo una post-elaborazione morfologica dell'immagine per ottenere oggetti segmentati connessi e compatti, e migliorare le prestazioni. Poiché la normalità è molto varia, un modello addestrato sull'intero set di dati non è efficace. Pertanto, raggruppando i tipi di copertura del suolo, dividiamo le immagini in sottoinsiemi, in base alla copertura del suolo che contengono. Questo ci permette di costruire Autoencoder specifici per la copertura del suolo, ottenendo prestazioni di segmentazione superiori rispetto ai modelli addestrati sull'intero set di dati.

Parole chiave: deep learning, segmentazione di immagini, rilevamento di anomalie, rilevamento di discariche abusive, autoencoder, clusterizzazione della copertura del suolo

Contents

1 | Introduction

In the last decades, we have witnessed the exponential growth in global industrial production that has led to a rapid increase in waste generation. Waste management has major implications for human health, environment preservation, sustainability and circular economy [32]. However, waste disposal is sometimes managed outside of the legal framework, posing a persistent problem across all regions, particularly in developing countries [29]. Illegal waste dumping can occur as isolated illicit events or at larger-scale sites, namely illegal landfills. This uncontrolled waste disposal leads to contamination of the surrounding areas, with harmful effects on humans, animals and plants. Indeed, soluble pollutants are likely to be transported into groundwater by rain and spread out into the environment. Gaseous pollutants, generated from degradation processes in illegal landfills or from illegal waste combustion, are dispersed by the wind to the surrounding areas. Limoli et al. [17] explain that landfill leachates have major effects of acidification, eutrophication and oxygen depletion. Moreover, the organoleptic qualities of groundwater can be compromised. Gaseous emissions spread out in the atmosphere or penetrate the ground, leading to global warming, acidification, photochemical smog and the formation of ground-level ozone. All these hazards represent a strong call to action, not to mention the significant impact on the health of humans and other living creatures. To ensure our safety and protect the environment, it is crucial to fight these environmental crimes.

Our goal is to create an automated model that can detect and segment illegal landfill sites through the analysis of satellite imagery. In this work, we address the problem of illegal landfill detection through satellite images as an anomaly detection and segmentation problem, where landfills represent anomalies and images without waste constitute normality. Our work will help identify dangerous sites and prevent ecological damage.

Our research is part of the European PERIVALLON project [11], which aims to detect organised environmental crimes such as illegal waste disposal. Using the latest technologies in the fields of artificial intelligence, computer vision, geospatial intelligence, remote sensing, and online monitoring, PERIVALLON improves investigation procedures creating an Environmental Crime Observatory that will detect and prevent the environmental crimes throughout Europe.

Detecting illegal landfills in satellite images is mainly performed by human experts, which is inefficient and limited compared to a possible automated version. Hence, our automated tool for landfill detection based on deep neural networks can provide a more efficient and effective solution. Indeed, it is capable of identifying anomalies that human observation might miss, by continuously scanning the ground with satellite images.

Traditional monitoring techniques for such tasks are supervised classification and segmentation, which rely on prior knowledge of the waste types for model training. Instead, we employ an unsupervised approach to detect illegal waste, without requiring advance information about the categories of garbage to perform the training. Supervised methods tackle the problem as an image-level classification problem, while we address it as a pixel-level segmentation problem.

The main challenge tackled by this thesis work is the great heterogeneity of land covers without illegal waste. Indeed, in our dataset there is an extensive variety of normal images, meaning that a huge number of land cover types are considered normal, from meadows to rivers, from beaches to streets, from residential houses to industrial buildings. This variability in the concept of normality can lead to the development of a model able to well reconstruct each type of ground, even those containing anomalies. Thus, instead of creating a general neural network for all the satellite images, our solution suggests developing a specific model for each group of land covers and using it to detect illegal landfills only on images including those land cover types. This means that our neural network will be specialized on the land covers on which it has been trained, managing to identify anomalies between the normal ground.

In order to develop land cover specific models, we combine methodologies from the fields of anomaly detection and graph theory. We build a graph based on land covers, and we use spectral clustering on the land cover types to define strongly related groups of land covers. From these groups, we split images into subsets according to the land covers they contain. On these subsets of images, we apply Fully Convolutional Autoencoders, followed by morphological image operations, to produce more efficient and accurate identification of illegal landfill sites.

We organise the thesis work in the following chapters:

- Chapter 2 formally states the formulation of our problem, how we address it, and the characteristics of our research.
- Chapter 3 covers the theoretical background of our work. We illustrate the theory

behind anomaly detection, Convolutional Neural Networks, which are the most common network for image analysis, Autoencoders, which are the family of models we chose to use, Structural Similarity, two different algorithms for graph clusterization, and eventually the mathematical operations of morphological image processing.

- Chapter 4 illustrates the solution we propose to solve this anomaly detection problem, dwelling on the presentation of the dataset.
- Chapter 5 explains the implementation of our model, from the preprocessing and postprocessing to the architecture and loss used during the training phase.
- Chapter 6 describes the figures of merit used in the testing phase and reports the results of the experiments performed on our model.
- In Chapter 7, we draw the conclusions of our work and suggest potential ideas for future work.

2 | Problem formulation

In this chapter we formally state the anomaly detection problem that we address in this thesis. We refer to images containing an illegal landfill as *anomalous*, while we consider *normal* all the images that do not contain any illegal waste.

In input, we can consider an RGB image as a three dimensional matrix $X \in \mathbb{R}^{w \times h \times 3}$, where values are normalized between $[0, 1]$. Here w is the width and h is the height of the image. Our goal is to locate anomalous regions in the image X defining as output an *anomaly mask*:

$$\Omega_X(i, j) = \begin{cases} 0 & \text{if } X(i, j) \text{ is normal} \\ 1 & \text{if } X(i, j) \text{ belongs to an anomaly} \end{cases}, \tag{2.1}$$

with $X(i, j)$ the pixel at row i and column j in X. We face this anomaly detection problem using an *unsupervised* approach, which means that we employ unlabeled data. During the training phase, we take into account only *normal* images $X_1, ..., X_n \in \mathcal{X}$, namely images not containing any illegal landfill. An unsupervised approach that considers only a known type of data, in our case only normal images, is also known as a *semisupervised* approach.

Our work belongs to the PERIVALLON project [11], funded by the European Union, which addresses the critical issue of organized environmental crime. This illicit activity includes intentional pollution and illegal disposal of different kind of waste, posing complex challenges for detection and investigation. The project's primary objective is to fight organized environmental crime by advancing tools and solutions as long as promoting international cooperation. Using the latest technologies in the fields of artificial intelligence, computer vision, geospatial intelligence, remote sensing, and online monitoring, PERIVALLON improves investigation procedures creating an Environmental Crime Observatory that will prevent the environmental crimes throughout Europe. With a diverse consortium of law enforcement agencies, academic institutions and industry partners, PERIVALLON is well equipped to effectively address the multifaceted challenges posed by organised environmental crime.

Our role in this project is to develop a solution able to analyze the satellite images and detect any illegal dumping site. We decide to address this problem using deep learning techniques from a different perspective with respect to the approaches taken by other researchers involved in this project. Indeed, the other researchers use binary or multi-class classification, which requires knowledge of each type of waste in order to detect it. However, we use an unsupervised anomaly detection approach that can detect any type of waste without prior knowledge. This gives the model greater flexibility and avoids the need for retraining the model each time the satellite image dataset is updated.

The dataset used in this study is AerialWaste [30], which collects satellite images of the Lombardia region and labels them according to the presence of illegal landfills. As stated earlier, illegal dumpsites are referred to as anomalies, and images containing them are considered anomalous. Normality, on the other hand, implies the absence of illegal waste, and therefore normal images contain no illegal landfill.

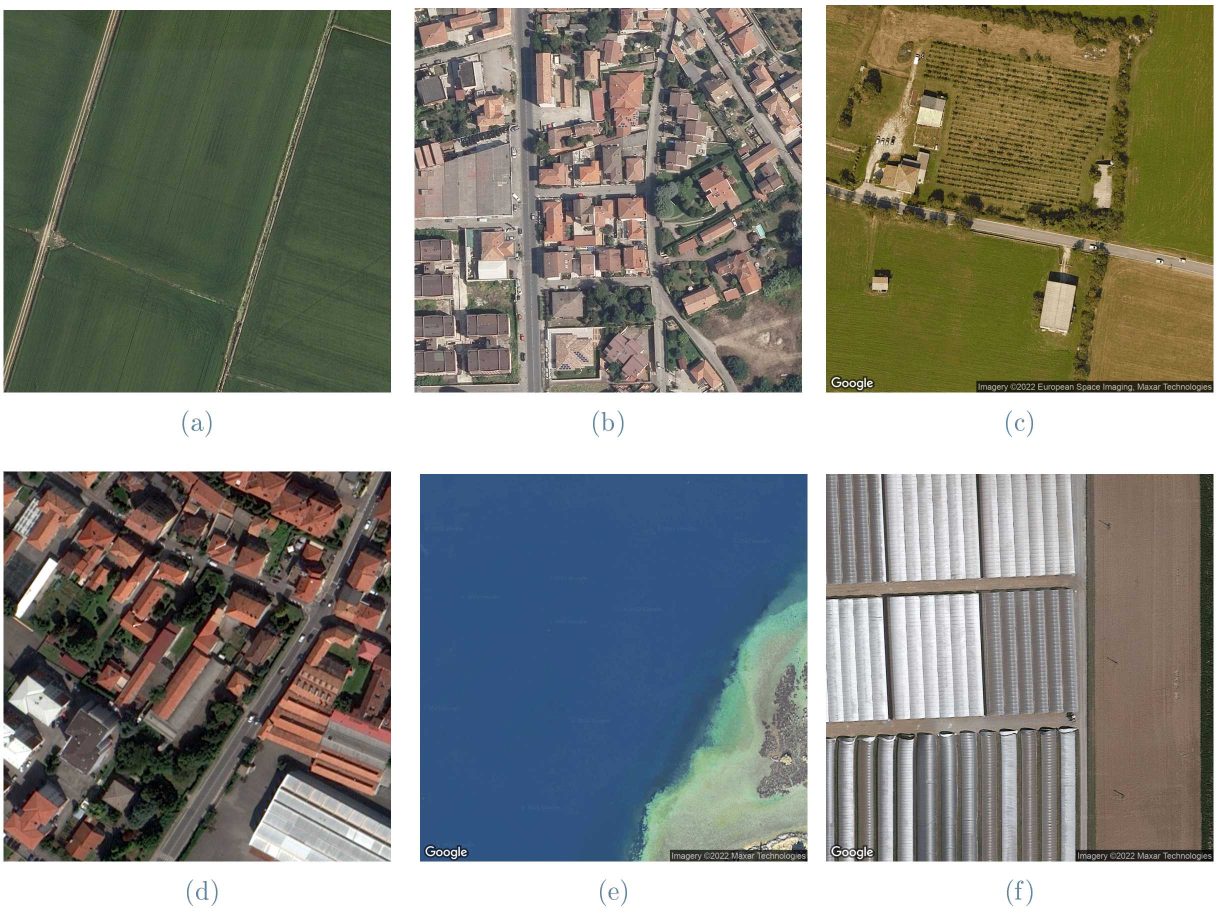
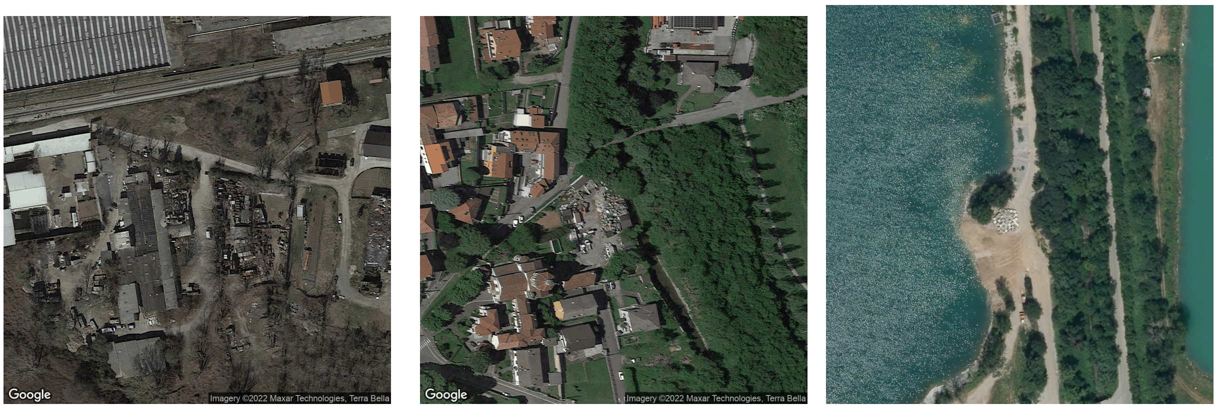

(a)
(b)
(c)
(d)
(e)
(f)
Google
Imagery ©2022 European Space Imaging, Maxar Technologies
Imagery ©2022 Maxar Technologies

Figure 2.1: Example of normal images in AerialWaste dataset.

Some examples of normal images included in this dataset are presented in Figure 2.1.

Figure 2.2 shows examples of anomalous images from the dataset, where the dumping sites are typically captured in the central area of the image.


Google
Imagery ©2022 Maxar Technologies, Terra Bella
Google
Imagery ©2022 Maxar Technologies, Terra Bella

(a) (b) (c)

Figure 2.2: Example of anomalous images in AerialWaste dataset

3 | Background

In this chapter we present the state-of-the-art of the methods applied in this thesis. In addition, the background topics are described to introduce the context and challenges addressed in this work.

First of all, in Section 3.1 we introduce the field in which our work is developed, and the techniques that can be applied. Afterwards, we describe in Section 3.2 the basis of deep convolutional networks, especially of Convolutional Neural Networks, while in Section 3.3 we illustrate the Autoencoder models. In Section 3.4 the Structural Similarity Index is introduced. In Section 3.5 we explain the theory behind graphs, clustering and two algorithms to find the best graph partition. Finally, in Section 3.6 the morphological image processing techniques are described.

3.1. Anomaly detection

Anomaly detection surrounds our lives without us noticing it, as Lukas Ruff et al. said [25]. Indeed, anomaly detection has a key role in identifying irregularities, defects, or unusual occurrences that could signify potential issues, fraud, or simply a novel phenomena. The goal of anomaly detection is to uncover hidden insights within data that might not be apparent through standard data analysis techniques, or that could be missed if the detection is performed by humans [18]. Formally, we can describe an *anomaly* as "*an observation deviating considerably from the notion of normality* ".

Let $\mathcal{X} \subseteq \mathbb{R}^T$ be the data space from a certain given task. The concept of *normality* can be defined as the distribution $\mathbb{P}^+$ on $\mathcal{X}$, namely as the ground-truth law of normal behavior in the given task [25]. Thus, an *anomaly* is an observation deviating from this law, belonging to a low probability region under $\mathbb{P}^+$, and the corresponding set of anomalies can be defined as

$$\mathcal{A} = \{x \in \mathcal{X} \mid p^+(x) \leq \lambda\}, \quad \lambda \geq 0 \tag{3.1}$$

with $p^+(x)$ the probability density function of $\mathbb{P}^+$, λ a certain threshold under which $\mathbb{P}^+$ is sufficiently small.

There exist three main categories of anomalies[9], at which it can be added a fourth category [25] [2]:

1. *Point anomalies*, where the observation deviating from the normal behavior is an individual data point $x \in \mathcal{A}$; it is the easiest type to be detected and the most commonly studied.

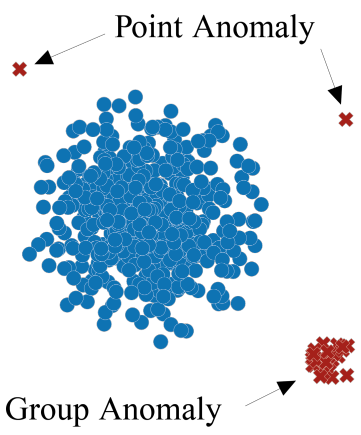
2. *Conditional* or *Contextual anomalies*, where the instance is anomalous under specific condition and in a specific context; for this type the normal law is a conditional distribution $\mathbb{P}^+ \equiv \mathbb{P}^+_{X|C}$, with $p^+(x \mid c)$ conditional probability density function depending on the context variable C.

3. *Collective* or *Group anomalies*, where a group of data is considered as anomalous but not necessarily the single observations themselves; indeed only the collection $\{x_i \in \mathcal{X} \mid i \in I\}$ of instances together differs from normality, with $I \subseteq \mathbb{N}$ a set of indexes implying a relation.

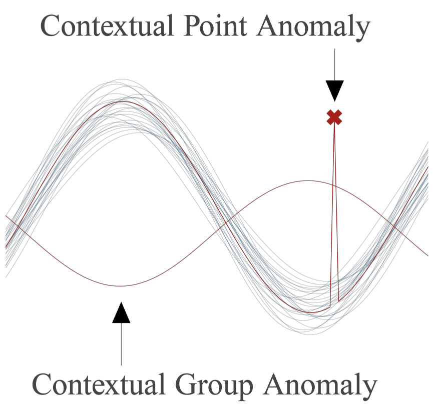
4. *Low-level sensory anomalies* and *high-level semantic anomalies*, where the terms high and low refer to the level of feature hierarchy; low-level anomalies differs, for instance, at pixel-level in texture and shape from the normal data, while high-level anomalies differ in sense, as the type of object in an image or the topic in a text.

Figure 3.1 illustrates some examples of the different types of anomalies just discussed.

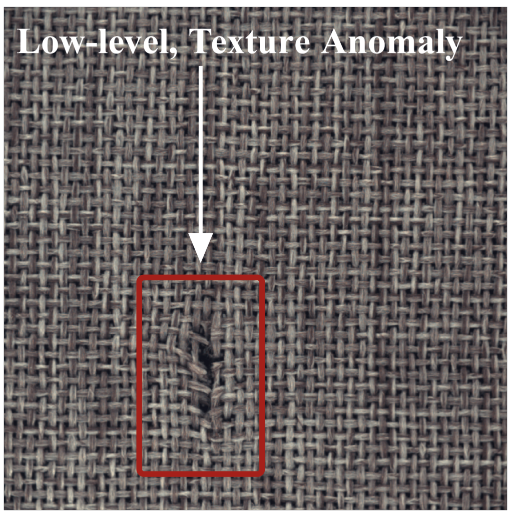
Moreover, it could be defined a distinction between *anomaly, outlier* and *novelty*, even if all of them refer to instances belonging to a low probability region under $\mathbb{P}^+$: an anomaly follows a distribution different from $\mathbb{P}^+$, an outlier is a rare or low-probability observation from $\mathbb{P}^+$, while a novelty belongs to a new region of a non-stationary $\mathbb{P}^+$.
Furthermore, the aim of finding anomalies, outlier and novelties could be really different. Usually, an anomaly is a data of interest to be found, while outliers are considered as measurement errors to be deleted. Instead, a novelty is a new instance that should be considered, and implies that the model has to be updated to a new version.
In Figure 3.2 an example of the difference between anomaly, outlier and novelty in images.



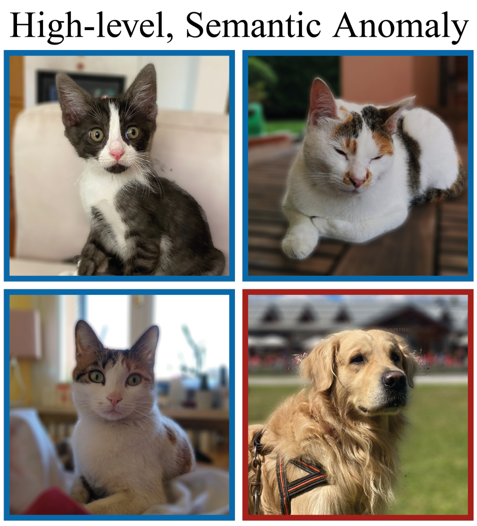
(a)



(b)

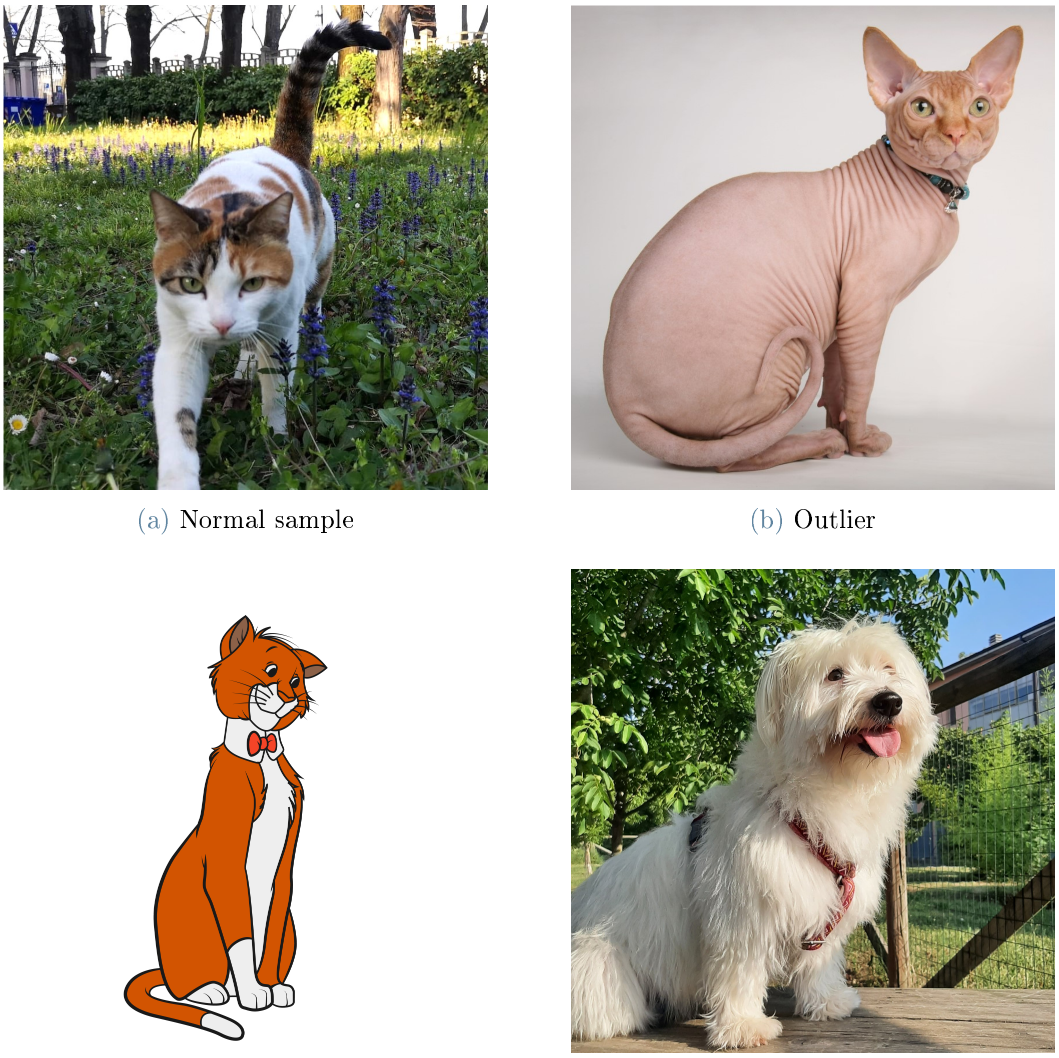


(c)



(d)

Figure 3.1: The different types of anomalies: 3.1a shows examples of two point anomalies and a group anomaly, 3.1b shows two examples of contextual point anomalies, one point and one group, 3.1c shows a low-level anomaly in a texture image, namely a hole in the fabric from the industrial MVTec dataset [6], 3.1d shows an example of high-level semantic anomaly, namely an image of a dog between cat images.



(a) Normal sample (b) Outlier (c) Novelty (d) Anomaly

Figure 3.2: An example of difference between a normal sample, an outlier, an anomalous sample, or a novelty. Here the normal image is a cat 3.2a, an outlier is a specific rare race as the Canadian Sphynx 3.2b, an anomaly is a high-level semantic error like a dog 3.2d, and a novelty is a cat but the animated film version 3.2c.

3.1.1. Main challenges

Conceptually, an anomaly is defined as a pattern not following the expected normal behavior, therefore in a direct approach to anomaly detection is to define the region of this normal behavior and mark any observations falling outside it as anomalous. However, there are several challenges that anomaly detection methods have to face, as stressed from Chandola et al. [9] and Pang et al. [24]:

- The concept of normality could evolve in time, such that what is now normal may no longer be normal in the future.
- The notion of anomaly could totally change for different task domains, as the description of anomalies is irregular.
- It could be not explicit the difference between anomalies and normal observations, such that it results difficult to define the boundaries of the normal region; indeed the distribution, behavior and data structure of anomalies is unknown.
- Often the anomalies are disguised as normal observations, especially when results of illegal actions.
- Sometimes noise is present in the data making normal observations similar to actual anomalies, thus it becomes difficult to recognize and remove them.
- As anomalies are rare occurrences, there is usually a lack of labeled anomalous data available, hence the problem can be considered as unbalanced.

Therefore, the general anomaly detection problem is though to solve, leading to the definition of task-specific problems, driven by the nature of the data and the type of application.

3.1.2. Machine learning approaches

Data instances are provided with labels indicating if that instance is *normal* or *anomalous*. However, it is usually expensive obtaining accurate and representative labeled data, since this collection is often manually performed by human experts. Moreover, getting labels for anomalous data is typically more difficult than getting the normal ones, considering that anomalous behavior is often dynamic in nature. Besides, in certain tasks anomalies are very rare, such as for space crashes or natural disasters. Anomaly detection methods can operate in three manners [9], depending on the available labels.

Supervised setting

Supervised anomaly detection methods assume the availability of labels in the training data both for anomalies and normal items. It follows two major problems: typically anomalies are fewer than normal items, thus the classes distributions are imbalanced, and the collection of accurate and representative labels for anomalous items is challenging. Formally, the data available in supervised anomaly detection setting are defined as follows

$$(x_1, y_1), \dots, (x_n, y_n) \in \mathcal{X} \times \mathcal{Y} \tag{3.2}$$

where $\mathcal{X} \subseteq \mathbb{R}^T$, and $\mathcal{Y} = \{0, 1\}$ with $y = 0$ denotes a normal instance, $y = 1$ denotes an anomalous instance.

Unsupervised setting

Unsupervised methods do not require labels to train the data, and for this reason they are the techniques most commonly applied. These methods implicitly make the assumption that normal instances are more frequent with respect to anomalous instances in test data, but if this assumption does not occur then a high rate of false alarms are risen. The unsupervised setting is formally defined as

$$x_1, \dots x_n \in \mathcal{X} \tag{3.3}$$

namely, only unlabeled data are available in $\mathcal{X} \subseteq \mathbb{R}^T$.

Semisupervised setting

The techniques operating in a semisupervised setting assume that the training data has labeled instances only for the normal class. These methods are more applicable than supervised techniques, since we avoid the issue of labeling anomalies. Usually, the semisupervised techniques build a model for the normal class behavior and use it to detect anomalies in the test data.
Many semisupervised methods operate in an unsupervised mode during the training using unlabeled data, knowing that they are only normal data. Formally, the semisupervised setting is defined as follows

$$x_1, \dots x_k \in \mathcal{X} \quad \text{and} \quad (x_{k+1}, y_1), \dots, (x_{k+m}, y_m) \in \mathcal{X} \times \mathcal{Y} \tag{3.4}$$

where, as before, $\mathcal{X} \subseteq \mathbb{R}^T$ and $\mathcal{Y} = \{0, 1\}$, with $y = 0$, $y = 1$ denote normal and anomalous instances respectively.

3.1.3. Addressing the anomaly detection problem

In order to address this problem, we can present three different classes of methods used in the literature [25][24]. The first are density estimation and probabilistic models, predicting anomalies by estimating the probability distribution of normal data. These methods include, for instance, classical statistical models.
The second category is composed of one-class classification methods, trying to directly determine a decision boundary for the desired level of the normal distribution.

Lastly, there are the *reconstruction models*, the approach most commonly used in anomaly detection neural networks. The idea is to build a model able to reconstruct well normal items and that fails in the reconstruction of anomalies. This thesis has developed a reconstruction model, hence this section will focus on this field.

First and foremost, let us define the reconstruction objective. Consider a feature map from $\mathcal{X}$ to itself, an encoding function $\mathcal{E} : \mathcal{X} \to \mathcal{Z}$ and a decoding function $\mathcal{D} : \mathcal{Z} \to \mathcal{X}$, respectively the *encoder* and the *decoder*. $\mathcal{Z}$ is called the *latent space*, and $\mathcal{E}(X) = Z$ the *latent representation* of X. Then, le the composition $\phi_\theta \equiv (\mathcal{D} \circ \mathcal{E})_\theta$ be the feature function with θ the parameters of the model. Hence the reconstruction objective is to learn ϕ_θ such that

$$\phi_\theta(X) = \mathcal{D}(\mathcal{E}(X)) = \bar{X} \approx X \tag{3.5}$$

meaning to find an encoding and a decoding function such that the input is reconstructed with minimal error, namely:

$$\min_{\mathcal{D},\mathcal{E},\theta} \left\| X - (\mathcal{D} \circ \mathcal{E})_\theta(X) \right\| \tag{3.6}$$

where $\|\cdot\|$ is the *reconstruction loss.*

In order to avoid the identity function as trivial solution to 3.6, it must be considered the so called *manifold assumption* entailing the existence of a lower dimensional latent space such that 3.5 has solution. In an anomaly detection ideal perspective, the resulting reconstruction loss will be low on normal data, and high on anomalous data.

3.2. Convolutional Neural Network

Convolutional Neural Network (CNN) is a type of deep Feed Forward Neural Networks designed to extract interesting features in the images domain. Traditional *Feed Forward Neural Networks* are composed of an input layer, one or more hidden layers, an output layer and a set of weights, that are considered the network parameters. Each layer is composed by neurons, namely a set of nodes connected to the previous and the following layers. Each layer has an input and an output: the inputs is multiplied by the weights of the layer, summed and passed through an activation function, which is the output of the neuron. This passes to the next layer, and the process is iterated for each hidden layer.

Convolutional neural networks were proposed for the first time by LeCun et al. [16] in 1989 and have immediately emerged as one of the most powerful tools in the field of artificial neural networks, especially excelling in pattern recognition. Thanks to the reduction in terms of number of parameters, CNNs can be applied to solve complex tasks, which was impossible with classic artificial neural networks.

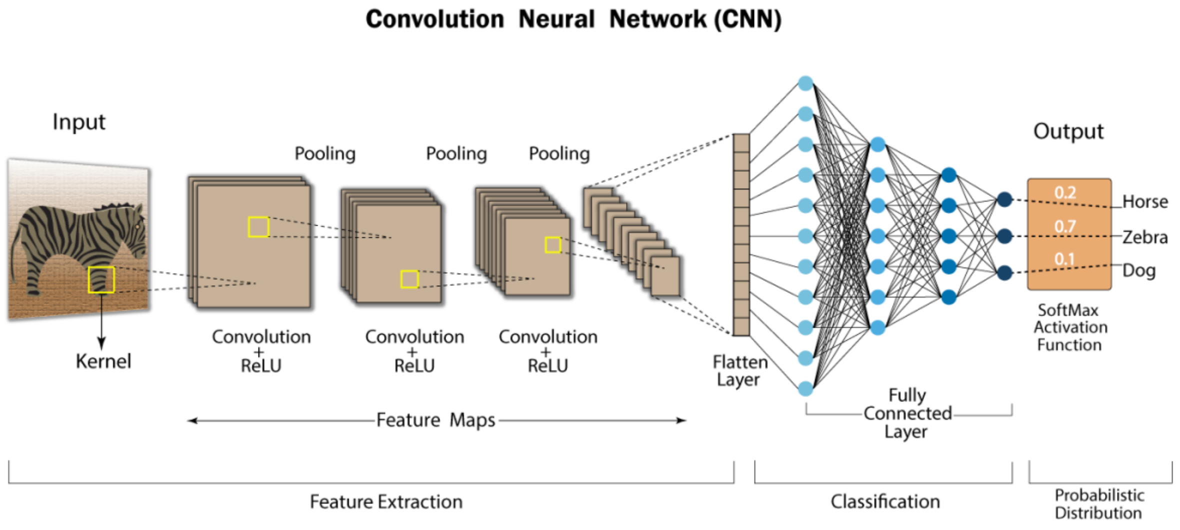


Figure 3.3: Example of CNN to classify images.

The term *convolutional* refers to the mathematical operation of convolution, which is performed between matrices, and will be explained in 3.2.2. Unlike classical neural networks, CNNs succeed in capturing spatial features in data, making them particularly suitable for image and video analysis. Indeed, their first application [16] was the recognition of handwritten characters.
Figure 3.3 illustrates an example of a CNN where in output it classifies the image received in input.

3.2.1. Convolution operation

Convolution is an operation on two real valued functions f and g that produces another function h which represents the effect of one function over the other. It is represented with the $\circledast$ symbol and defined as[13]:

$$h(t) = (f \circledast g)(t) = \int f(x)g(t-x)dx. \tag{3.7}$$

In general, convolution is defined for any functions for which the above integral is defined. In the context of deep learning, more specifically in the CNN vocabulary, the function f is often referred to as the *input*, function g is the *kernel* and function h is the *feature map*.

Equation (3.7) can be used when dealing with continuous functions, but standard convolution must be replaced with discrete convolution when dealing with discrete functions. In image processing field, discrete convolution is used to convolve a kernel (also known as *filter*) on an image to detect, for instance, edges. In convolutional neural networks context, discrete convolution is applied over the input image in order to obtain a linear combination of values in the corresponding region of the image. Discrete convolution applied over the input volumes in CNNs is defined as follows:

$$\begin{aligned} H(i,j) =& (I \circledast K)(i,j) = \sum_x \sum_y I(x,y)K(i-x, j-y) = \\ =& (K \circledast I)(i,j) = \sum_x \sum_y I(i-x, j-y)K(x,y) \end{aligned} \tag{3.8}$$

where I is an input image, K the convolutional kernel, H the output feature map indexed at position (i,j). The equality in (3.8) follows from the fact that convolution is a commutative equation.
In Figure 3.4 we can observe a practical example of discrete convolutional operation over image pixels, using kernel:

$$\begin{bmatrix} 0 & 1 & 2 \\ 2 & 2 & 0 \\ 0 & 1 & 2 \end{bmatrix} \tag{3.9}$$

3.2.2. Layer types

Convolutional Neural Networks consist of several layers, each serving a specific purpose in the network's functioning. Typically, the main layers in a CNN architecture include convolutional layers, pooling layers, activation layers (non-linear), and fully connected layers.

Let us assume to have in input a three-dimensional object such as an image or a video. The *convolutional layer* is the main component of a CNN. It applies the convolution operation, in its discrete form (3.8), to the input data, using filters or kernels to extract spatial features. By sliding these filters across width and height of the input volume, the convolutional layer effectively captures local patterns and creates feature maps. In a convolutional layer the discrete convolution is applied to the whole input volume by shifting the kernel of a certain number of pixels called *stride*, namely the parameter to control the overlapping of the filters. As the filter slides over the width and height of the input volume, it produces a 2-dimensional activation map. The network will learn filters that activate when some types of visual features are given in input, such as edges,

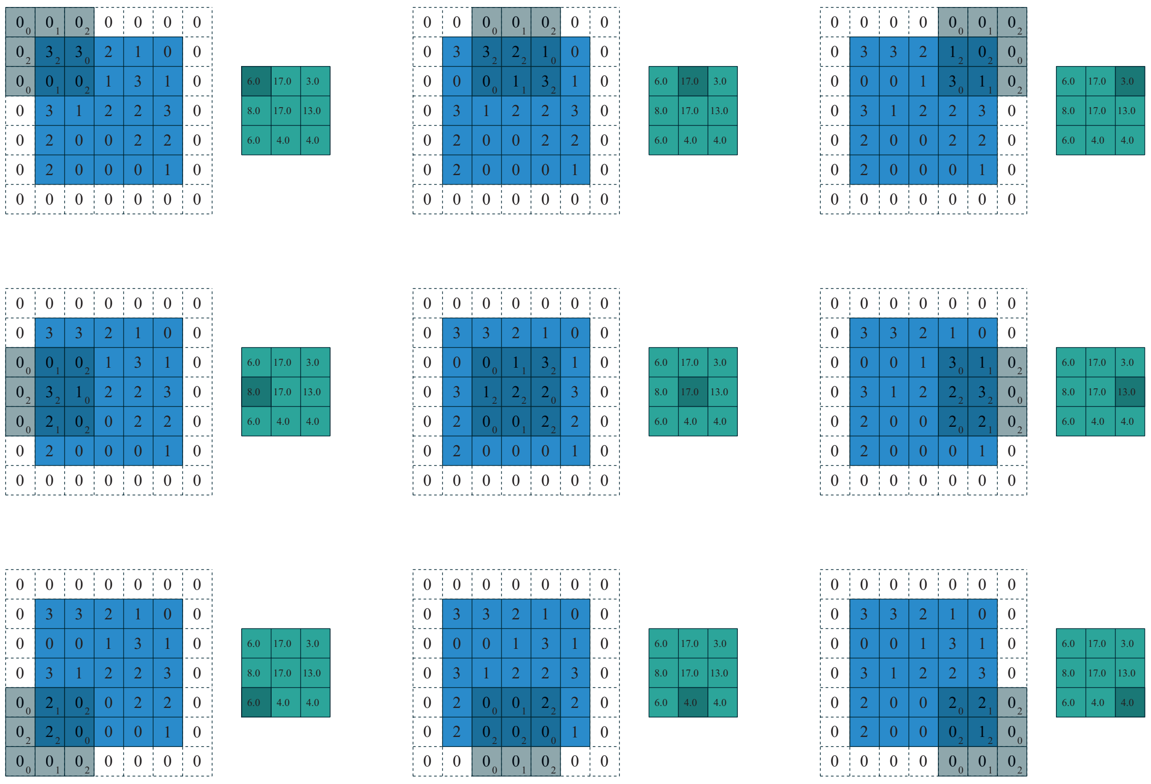


Figure 3.4: Practical example of the mathematical convolution operation.

particular shapes or patterns. Several filters can be applied in each convolutional layer, producing a separate 2-dimensional activation map.

One of the drawbacks [3] of the convolution step is the information loss that might happen on the image border. Indeed, since they are only captured when the filter slides, they never have the chance to be observed. A very simple and efficient method to solve this issue is to use *zero-padding*, namely adding pixels with value$=0$ around the image. Using this simple strategy, the filter is able to slide on the borders of the image without losing any information. Another benefit of zero-padding is the ability to manage the output size, obtaining the output of the same exact dimension of the original input.
Figure 3.5 shows an example of convolution with kernel defined in Equation (3.9), where the image is padded with a 1×1 border of zeros [10].

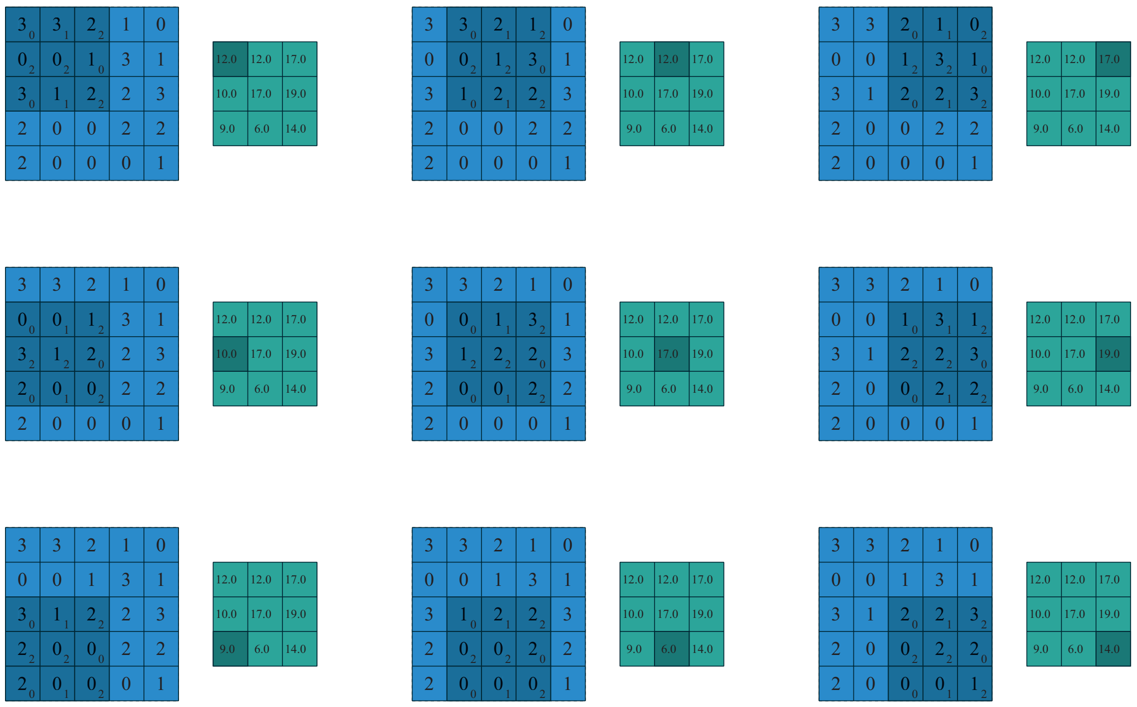


Figure 3.5: Example of convolution where the image is padded with a 1×1 border of zeros.

Following the convolutional layer, an *activation layer* is applied to introduce non-linearities into the model. This layer helps in capturing complex relationships and introducing non-linear transformations to the feature maps. For many years, sigmoid and tanh were the most popular activation functions. Sigmoid activation function squeezes the output of a neuron between 0 and 1: $\sigma(x) = \frac{1}{1+e^{-x}}$. It is commonly used in binary classification tasks where the output needs to be interpreted as a probability. Tanh function squeezes the output between -1 and 1: $\tanh(x) = \frac{2}{1+e^{-2x}} - 1$. It is often used in scenarios where negative values are more prevalent, such as sentiment analysis. In Figure 3.6 are shown different activation functions

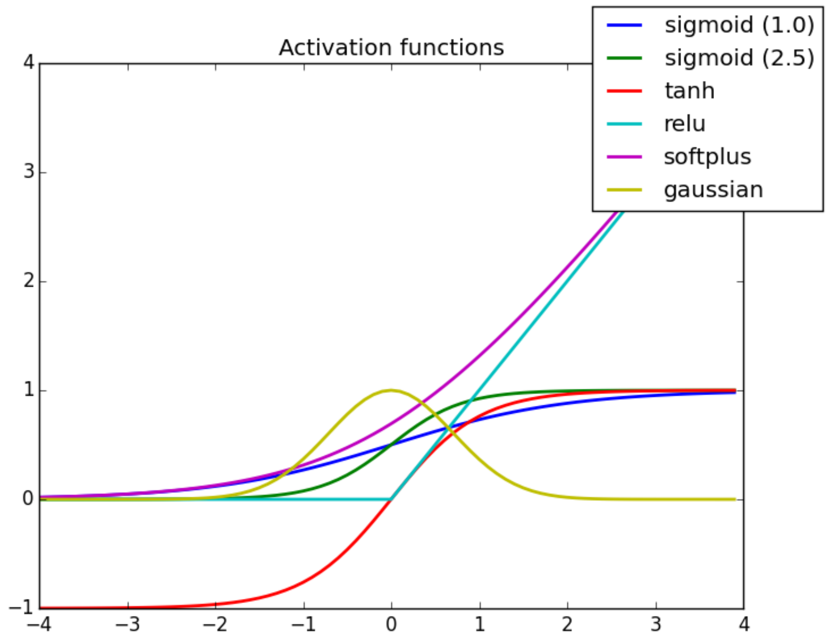


Figure 3.6: Different activation functions.

Another more recent activation function is the *Rectified Linear Unit* (ReLU), that has become popular for the following reasons [3]:

1. ReLU has a simple definition, for both the function and the gradient:

$$ReLU(x) = \max(0, x) \qquad \frac{d}{dx}ReLU(x) = \{1 \text{ if } x > 0\ ,\ 0 \text{ otherwise}\}. \tag{3.10}$$

2. Sigmoid and tanh functions cause problems due to the gradient signals that is close to zero but in the center, called vanishing gradient problem, that affects deep network. ReLU function helps with this issue, having a constant gradient fo positive inputs.

3. ReLU function creates a sparser representation, since the gradient is exactly 0 for negative inputs not all the neurons are activated.

4. ReLU function is sale invariant, namely $\max(0, bx) = b\max(0, x)$ for $b \geq 0$

ReLU also have some disadvantages: it is differentiable everywhere except in 0, but this can be handled in the implementation, and it causes a problem called Dying ReLU, that causes the inactivity of some neurons for almost all inputs. The latter problem can be alleviated by using some variations of the standard ReLU, as the Leaky ReLU $LeakyReLU(x) = max(ax, x)$ with $a \in (0, 1)$, which have a gradient value different from

0 for negative inputs.

The *pooling layer* reduces the spatial dimensions of the feature maps produced by the convolutional layer, without affecting the number of filters. It is usually inserted a pooling layer in-between successive convolutional layers. By aggregating information within local regions, pooling layers help to extract the most salient features while reducing the computational complexity of subsequent layers.

In the image processing domain, it is similar to reduce the resolution. Max pooling is one of the most common types of pooling layers: the image is divided into sub-region rectangles, and it returns the maximum value of the pixels inside that sub-region. It is often used the size 2×2 in max pooling.
Average pooling is another typical choice of pooling layer: for each sub-region it returns their mean value.

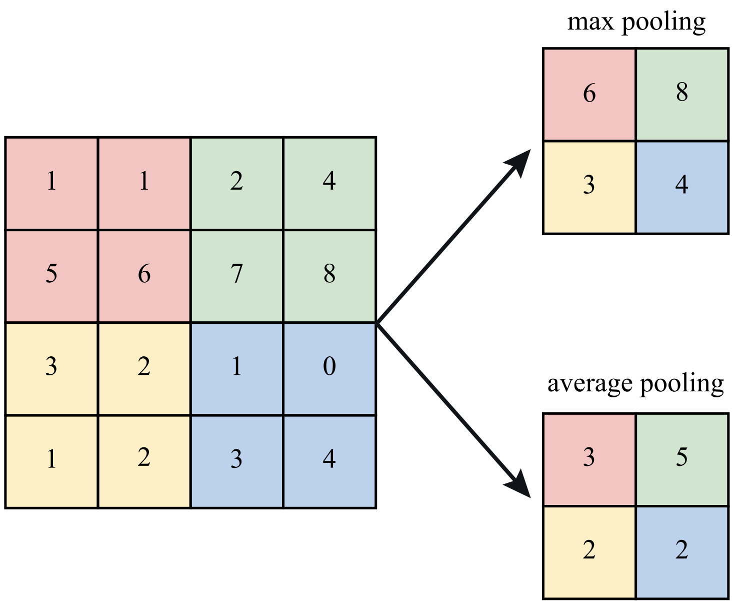


Figure 3.7: Different activation functions.

As we observe in Figure 3.7, when pooling is performed with filter 2×2 and stride 2, the image is split into the colored sub-regions and for each sub-region it is performed the maximum or the mean. Pooling can be used with different filters and strides in order to improve the efficiency.

The *fully-connected layer* connects every neuron from the previous layer to the next, mimicking the structure of traditional neural networks. Therefore, each node in a fully-connected layer is directly connected to every node in both the previous and in the following layer as shown in Figure 3.8.

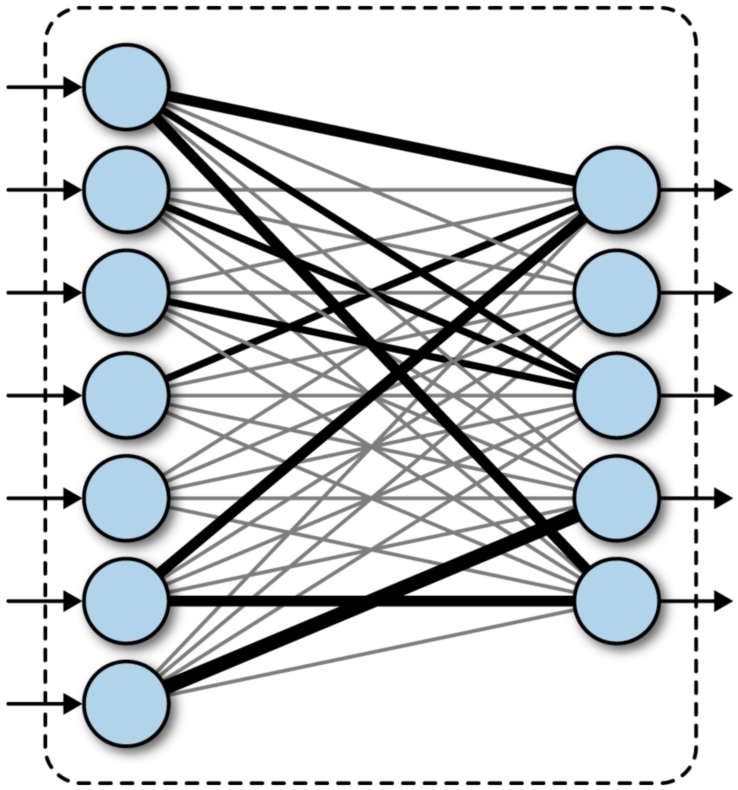


Figure 3.8: Example of fully connected layer.

This layer is responsible for the final classification or regression task and incorporates the extracted features into the final prediction. The major drawback of a fully-connected layer, is that it includes a lot of parameters that need complex computational in training examples. Therefore, we try to remove the number of nodes and connections using the dropout technique.

3.3. Autoencoders

One of the most popular approaches for deep anomaly detection in images are *autoencoders* [25], namely feed-forward multi-layer neural networks in which the desired output is the input itself [35]. Autoencoders belong to the reconstruction model category, defined in Section 3.1.

Autoencoders were first introduced during the 80s [26][15] for dimensionality reduction or feature learning, but they were then adopted for deep anomaly detection.
The main goal [4] is to learn in an unsupervised manner an informative representation of the data that can be used for various applications.

An autoencoder (AE) is composed of two main components: an encoder and a decoder.

The encoder $\mathcal{E}(\cdot)$ takes the data X in input and maps them into a hidden layer, describing an informative feature representation called *latent representation*. Then the decoder $\mathcal{D}(\cdot)$ maps the encoded data $\mathcal{E}(X)$ back to the original space, producing a reconstruction of the input.
Formally, the encoding function $\mathcal{E} : \mathcal{X} \to \mathcal{Z}$ and the decoding function $\mathcal{D} : \mathcal{Z} \to \mathcal{X}$ are respectively the *encoder* and the *decoder*. Hence the reconstruction map is defined $\phi_\theta(X) = \mathcal{D}(\mathcal{E}(X)) = \bar{X} \approx X$, as described in equation 3.5.

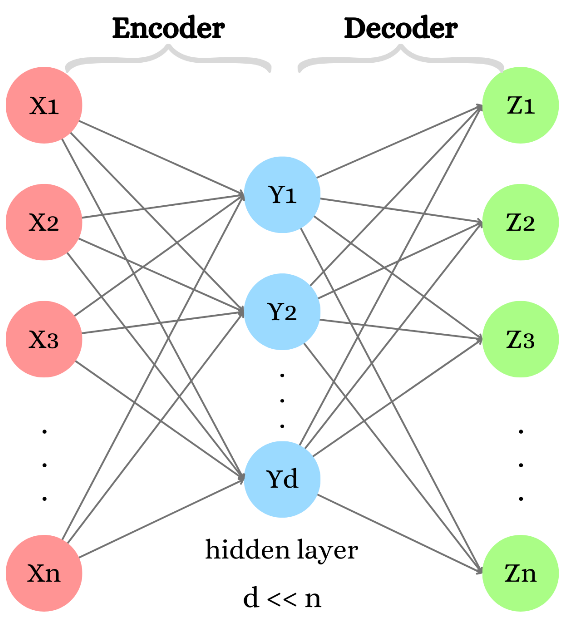


Figure 3.9: Example of autoencoder with 1 layer with n neurons in input, 1 with n neurons in output, and 1 hidden layer with d neurons.

The basic autoencoder has encoder composed of a single layer, a hidden layer and the decoder composed of one layer, as shown in Figure 3.9.
An autoencoder with multiple hidden layers is considered a *deep* autoencoder, able to represent complicated distributions over the input. From now on with the word autoencoders we will refer to the deep version.

The objective is to train the encoder and the decoder such that the difference between the original image and the reconstructed image is minimized, namely:

$$\min_w \frac{1}{d} \sum_{i=1}^{d} \|x_i - (\mathcal{D} \circ \mathcal{E})_w(x_i)\|^2 + \mathcal{R}, \tag{3.11}$$

that is a realization of the general reconstruction objective in 3.6. The optimization is performed over the weights w of the neural network encoder and decoder, $\mathcal{R}$ indicates the regularization.

To avoid a trivial reconstruction, namely having a reconstruction function equal to the identity map, the number of neurons of the hidden layer must be strictly smaller than the number of neurons in input and output layers, thus $n \ll d$. More in general, the latent space where the input is mapped should have a smaller dimension with respect to the space where our data live, creating a 'bottleneck' that enforces the data compression and limits the dimensionality. This is considered a form of regularization.
An autoencoder whose latent dimension is smaller than the input dimension is called undercomplete. Learning an undercomplete representation forces the autoencoder to capture the most salient features of training data.

Hence the learning process could be described as the minimization of the loss function

$$\mathcal{L}(X, \mathcal{D}(\mathcal{E}(X))), \tag{3.12}$$

where the loss function $\mathcal{L}$ is defined penalizing the dissimilarity between $\mathcal{D}(\mathcal{E}(X)) = \bar{X}$ and X.
Examples of viable losses are the *Mean Squared Error* and the *Mean Absolute Error*:

$$MSE(x, y) = \frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2 \tag{3.13}$$

$$MAE(x, y) = \frac{1}{N}\sum_{i=1}^{N}|x_i - y_i| \,. \tag{3.14}$$

Convolutional Autoencoders belong to the family of convolutional neural networks and they are used for image reconstruction. Both the encoder and the decoder are CNN, the encoder taking in input an image and extracting its main characteristics, while the decoder reconstruct the image. Thus, all the layers in the autoencoder are convolutional layers or pooling layers. In the encoder the pooling layers reduce the spatial dimension of the input, while in the decoder they increase it.

Considering convolutional autoencoders, in equations (3.13) (3.14) x and y are two images and N is the number of pixel involved in the comparison. These point-by-point metrics compute the l^2 and l^1 distances between the corresponding pixels in the two images and average the distance.

Figure 3.10 illustrates an example of convolutional autoencoder.

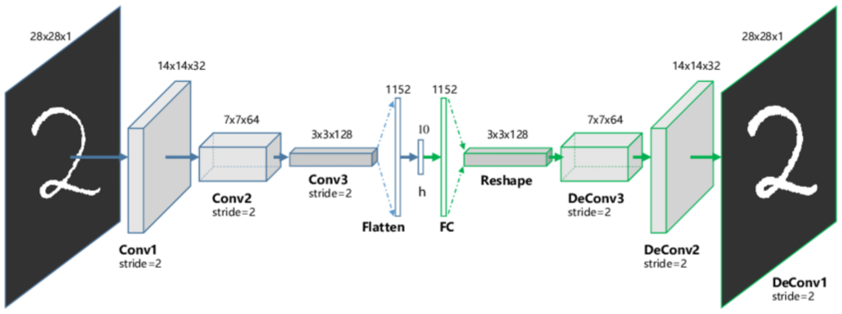


Figure 3.10: Example of a convolutional autoencoder that reconstructs handwritten digits.

3.4. Structural Similarity

Convolutional autoencoders, widely used in unsupervised defect segmentation, commonly use per-pixel reconstruction errors based on an ℓ^p-distance. However, Bergman et al. [5] suggest that these functions often lead to large residuals when the reconstruction includes slight localization inaccuracies around edges, and they propose the application of a perceptual loss function based on structural similarity (SSIM) [34] to autoencoders. Indeed, SSIM can grab inter-dependencies between regions in the image, considering luminance, contrast and structural information rather than comparing single pixel values.

The idea is to extract structural information of the entire image separating the objective into three comparisons: *Luminance, Contrast* and *Structure*. Indeed these three components are relatively independent, since a change in luminance or contrast will not affect the structure.
Hence the similarity measure can be written as

$$S(x,y) = f(l(x,y), c(x,y), s(x,y)) \tag{3.15}$$

Assuming the mean intensity and the standard deviation as follows

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{3.16}$$

$$\sigma_x = \left(\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)^2\right)^{\frac{1}{2}}, \tag{3.17}$$

then the luminance comparison is a function of the mean intensities defined as

$$l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \tag{3.18}$$

where C_1 is a constant included to avoid instability when $\mu_x^2 + \mu_y^2$ tends to zero.
The contrast comparison function, instead, is a comparison between the standard deviations of x abd y, defined as

$$c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \tag{3.19}$$

where C_2 is a constant defined similarly to C_1.
Lastly, we define the structure comparison function

$$s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3} \tag{3.20}$$

with C_3 a constant, and σ_{xy} can be estimated in discrete form as

$$\sigma_{xy} = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)(y_i - \mu_y). \tag{3.21}$$

Finally, we can combine (3.18), (3.19) and (3.20) defining:

$$SSIM(x,y) = \big(l(x,y)\big)^\alpha \cdot \big(s(x,y)\big)^\beta \cdot \big(c(x,y)\big)^\gamma \tag{3.22}$$

where $\alpha > 0,\ \beta > 0,\ \gamma > 0$ parameters to tune the importance of the components.
Wang et al.[34] set $\alpha = \beta = \gamma = 1$ and $C_3 = C_2/2$, obtaining the final version of SSIM index:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}\ . \tag{3.23}$$

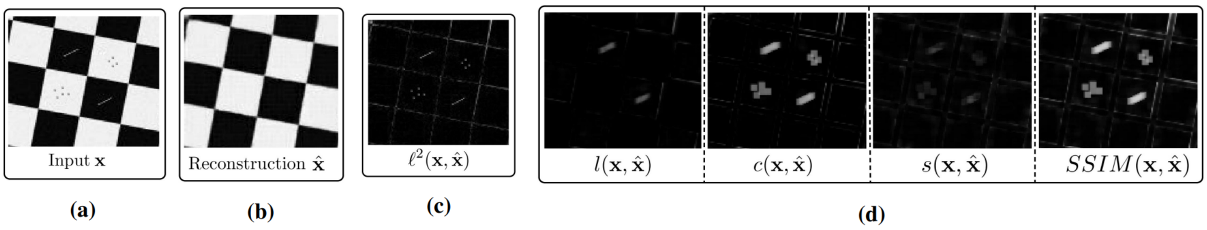

Input $\mathbf{x}$
Reconstruction $\hat{\mathbf{x}}$
$\ell^2(\mathbf{x},\hat{\mathbf{x}})$
$l(\mathbf{x},\hat{\mathbf{x}})$
$c(\mathbf{x},\hat{\mathbf{x}})$
$s(\mathbf{x},\hat{\mathbf{x}})$
$SSIM(\mathbf{x},\hat{\mathbf{x}})$
(a)
(b)
(c)
(d)

Figure 3.11: Example from [5] of SSIM with respect to ℓ^2 loss for defect segmentation. (a) checkerboard with simulated defeats, (b) the output reconstruction of the input image by an autoencoder. Residuals (d) for luminance l, contrast c, structure s, and their pointwise product that yields the final SSIM residual map.

Figure 3.11 reports an example of comparison between SSIM and ℓ^2, where we notice that SSIM gives more importance to the visually more salient disturbances than to the slight inaccuracies around reconstructed edges [5].

3.5. Graph clustering

A directed graph $G = (S, A)$ is determined by:

- a set S of elements called *vertices* or *nodes*,
- a set A of elements which are pairs (i, j) called *arcs*. The initial vertex i or an arc is called *origin* and j *destination*.

When the definition of a graph does not require to distinguish between origins and destinations of arcs, the graph is called *undirected*. In this case the elements of A will be called *edges*.
An arc or edge is said to be *incident* to the two nodes it connects.
Two vertices connected by an arc or edge are said to be *adjacent* or called *neighbors*. Figure 3.12 shows an example of an undirected weighted graph.

Let consider $G = (V, E)$ an undirected weighted graph, with weights $w_{ij} \geq 0$ between two vertices i and j. The weight on each edge w_{ij} is a function of the similarity between the two vertices i and j.

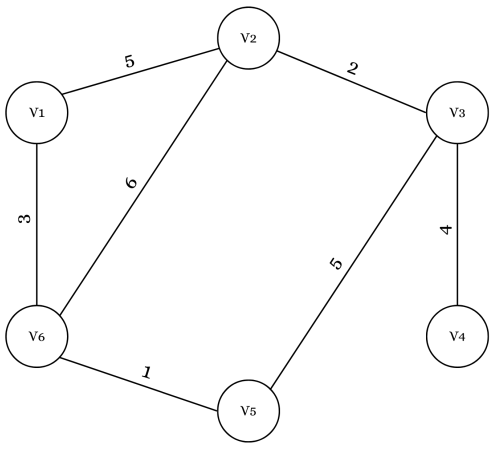


Figure 3.12: Example of an undirected weighted graph.

3.5.1. Normalized cut and spectral clustering

Given a set of data points $x_1, ..., x_n$ and a definition of *similarity* $s_{ij} \geq 0$, the aim of clustering is to split the data points into groups such that the points within the same group are *similar*, whereas those in different groups are *dissimilar* [33].
Considering the graph G previously defined, we desire to partition the set of nodes into disjoint sets $V_1, V_2, \ldots, V_k$, where, by measure s_{ij}, the similarity among vertices in a set V_i is high and, across different sets V_i, V_j is low.

We start partitioning the graph into two disjoint sets of vertices A and B, ie such that $A \cup B = V$ and $A \cap B = \emptyset$, by removing edges connecting the two sets [28]. A *cut* is the sum of the weights of the removed edges, representing the degree of dissimilarity:

$$cut(A, B) = \sum_{i \in A,\ j \in B} w_{ij}. \tag{3.24}$$

The bipartition of the graph is optimal if the *cut* is minimized. In Figure 3.13 an example of minimal cut in an undirected weighted graph. Afterwards, the current partition can be further recursively subdivided in order to create more than two groups.

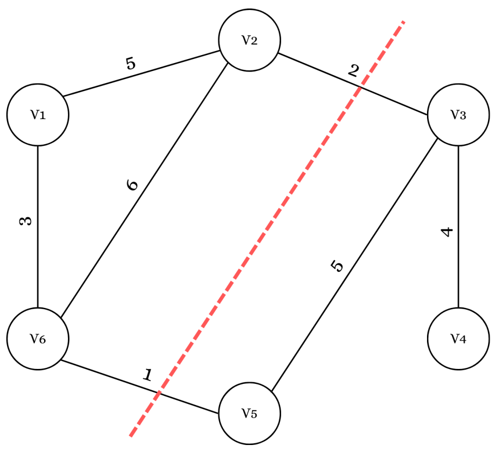


Figure 3.13: Example of a minimal cut equal to 3 of an undirected weighted graph.

In order to avoid cutting small sets of isolated nodes in the graph, Shi and Malik [28] introduced the *normalized cut (Ncut)*:

$$Ncut(A, B) = \frac{cut(A, B)}{assoc(A, V)} + \frac{cut(A, B)}{assoc(B, V)} \tag{3.25}$$

where $assoc(A, V) = \sum_{i \in A,\, k \in V} w_{ik}$ is the total connection from the vertices in A to all the vertices in the graph, and similarly defined for B.
The definition of normalized cut is also known as *disassociation* between the groups, and using this definition there will no longer be small Ncut value for the cut that partitions out small isolated points, since the cut value will be a large percentage of the total connection from that small group of nodes to all other vertices.

Similarly, it can be defined the measure for total normalized association between groups:

$$Nassoc(A, B) = \frac{assoc(A, A)}{assoc(A, V)} + \frac{assoc(B, B)}{assoc(B, V)} \tag{3.26}$$

where $assoc(A, A),\ assoc(B, B)$ the total weights of edges connecting nodes within A and B respectively.

It follows the relation:

$$\begin{aligned} Ncut(A,B) &= \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)} = \\ &= \frac{assoc(A,V) - assoc(A,A)}{assoc(A,V)} + \\ &\quad \frac{assoc(B,V) - assoc(B,B)}{assoc(B,V)} = \\ &= 2 - \Big(\frac{assoc(A,A)}{assoc(A,V)} + \frac{assoc(B,B)}{assoc(B,V)}\Big) = \\ &= 2 - Nassoc(A,B). \end{aligned} \tag{3.27}$$

Hence, minimizing the disassociation between groups and maximizing the association between groups is equivalent and simultaneously satisfied.

Unfortunately, minimizing the normalized cut is an NP-complete problem, but it can be efficiently found an approximate discrete solution.
Indeed, relaxing the hypothesis to real values only, the problem can be rewritten as the solution of the generalized eigenvalue system

$$(\mathbf{D} - \mathbf{W})y = \lambda \mathbf{D} y \tag{3.28}$$

where $\mathbf{D}$ is the $N \times N$ diagonal matrix with the total connection from each node $d(i) = \sum_j w_{ij}$ in the diagonal, $\mathbf{W}$ is the $N \times N$ symmetrical matrix with the graph weights as elements.
It is found in (3.28) that the second smallest eigenvector is the solution (in real values) of the normalized cut problem, thus it could be used to bipartition the graph. Hence, the *spectral clustering* can be considered as relaxation of the Normalized minimal cut problem, and can be used as graph partitioning in real values.

Finally, the grouping algorithm proposed by Shi and Malik [28] can be summarized in the following steps:

Algorithm 3.1 Normalized cut algorithm

```
1: Given a weighted graph $G = (V, E)$, set the weight on the edge connecting two nodes
   to be a measure of the similarity between the two nodes
2: Solve $(\mathbf{D} - \mathbf{W})y = \lambda \mathbf{D} y$ for eigenvectors with the smallest eigenvalues
3: Use the eigenvector with the second smallest eigenvalue to bipartition the graph
4: while the current partition should be subdivided do
5:     Repartition the segmented parts
6: end while
```

3.5.2. Louvain algorithm

Another way to approach clusterization is the Louvain algorithm, first introduced by Blondel et al. [8]. It is a heuristic method that consists in decomposing a networks into sub-units or *communities*, which are sets of highly interconnected vertices.
The quality of the resulting partitions is measured by the *modularity* of the partition, namely a scalar value in $[-1, 1]$ measuring the density of connections inside communities compared to connections between different communities [23].
In case of weighted networks, it is defined as [22]:

$$Q = \frac{1}{2m} \sum_{i,j} \left[w_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j) \tag{3.29}$$

where w_{ij} represents the weight of the edge between i and j, $k_i = \sum_j w_{ij}$ is the sum of the weights of the edges incident to vertex i, c_i is the community to which node i is assigned, $\delta(u, v) = 1$ if $u = v$ and $\delta(u, v) = 0$ otherwise, and $m = \frac{1}{2} \sum_{ij} w_{ij}$.
Since exact modularity optimization is a problem that is computationally hard, it is necessary to use an approximation algorithms when dealing with large networks.

Louvain algorithm is divided into two phases that are repeated iteratively. Assume to start with a weighted network of N nodes and consider the definition of *gain* in modularity as:

$$\Delta Q = \left[\frac{\Sigma_{IN} + 2k_{i,IN}}{2m} - \left(\frac{\Sigma_{OUT} + k_i}{2m} \right)^2 \right] - \left[\frac{\Sigma_{IN}}{2m} - \left(\frac{\Sigma_{OUT}}{2m} \right)^2 - \left(\frac{k_i}{2m} \right)^2 \right] \tag{3.30}$$

where Σ_{IN} sum of the weights of the links inside C, Σ_{OUT} is the sum of weights of the links incident to vertices in C, $m = \frac{1}{2} \sum_{ij} w_{ij}$, $k_i = \sum_j w_{ij}$, $k_{i,IN}$ the sum of weights of the edges from i to vertices in C.

Notice that since a node may be, and often is, considered several times, the first phase of Algorithm 3.2 stops when a local maxima of the modularity is reached, namely when no single vertex move can improve the modularity. Blondel et al. [8] indicate that the ordering of the nodes considered does not have a significant influence on the modularity obtained, but it can affect the computation time.

Algorithm 3.2 Louvain Algorithm

```
1: A different community is assigned to each vertex of the graph, such that $\#communities = N$.
2: while there is improvement in gain do
3:     for each node $i$ do
4:         consider all its neighbours $j$ and measure the gain of modularity that we would
           have removing $i$ from its community and placing it in the community of $j$
5:         if $gain \geq 0$ then
6:             node $i$ is then placed in the community for which this gain is maximum
7:         end if
8:         if $gain < 0$ then
9:             node $i$ stays in its original community
10:        end if
11:    end for
12: end while
13: Build a new network whose nodes are the communities found in the previous steps.
```

The final graph built in the last step of the Louvain algorithm 3.2 has vertices composed of communities of nodes, and edges with weights given by the sum of the weights of all the edges between nodes in the corresponding two communities.
It is possible to reapply the algorithm multiple times restarting from step 2 of 3.2.

3.6. Morphological image processing

Mathematical morphology provides an approach based on shape to process digital images. Properly used, these operations can be helpful in the extraction of essential shape characteristics and properties, deleting what is irrelevant.

As Haralick et al. explained [14], mathematical morphology has the same language as set theory.
Indeed, in binary images foreground regions can be denoted as Euclidean 2-space sets, while images with more information like color can be expressed by sets in higher dimensional spaces. Mathematical morphological transformations apply to sets of any dimen-

sions, even if in this thesis will be applied only to binary images. We will discuss about binary morphological operations of dilation and erosion, expanding then the concepts to opening and closing.

3.6.1. Dilation and erosion

Dilation is the morphological transformation using vector addition of set elements to combine two sets. It was first proposed in 1903 by Minkowski [21] as a set theoretic operation to characterize integral measures, but it was then studied for image shape extraction by Serra [27] and Matheron [20]. We will follow the definition provided by Haralick et al. [14].

Let A and B be sets in N-space E^N, with elements $a = (a_1, \ldots, a_N)$ *and* $b = (b_1, \ldots, b_N)$. Then the *dilation* of A by B is the set of all possible vector sums of pair of elements, namely

$$A \oplus B = \{c \in E^N \mid c = a + b, \text{ for some } a \in A, \ b \in B\}. \tag{3.31}$$

The roles of sets A and B is symmetric since the addition is commutative, hence the dilation is commutative, namely

$$A \oplus B = B \oplus A.$$

However, in practice operands A and B are treated differently: the first is considered the image to be treated, the latter is the *structuring element*, namely a shape parameter. Structuring elements can be, for instance, a disk, a square or an ellipse, and from this depends the transformation result.

Since addition is associative, it can be stated the property known as *chain rule* for dilation:

$$A \oplus (B \oplus C) = (A \oplus B) \oplus C$$

that is computationally advantageous from the complexity point of view, saving operations if $B \oplus C$ is considered the structuring element.

The dilation can also be defined in terms of image translation. Let $A \subseteq E^N$, $x \in E^N$, then the *translation* of A by x is defined by

$$(A)_x = \{c \in E^N \mid c = a + x, \text{ for some } a \in A\}$$

allowing the definition of dilation

$$A \oplus B = \bigcup_{b \in B} (A)_b. \tag{3.32}$$

From a practical perspective, this means that the dilation is the translation of a structuring element all over the image, from top left to bottom right [19]. In Figure 3.14 it is shown an example of dilation operation applied to a binary image [7].

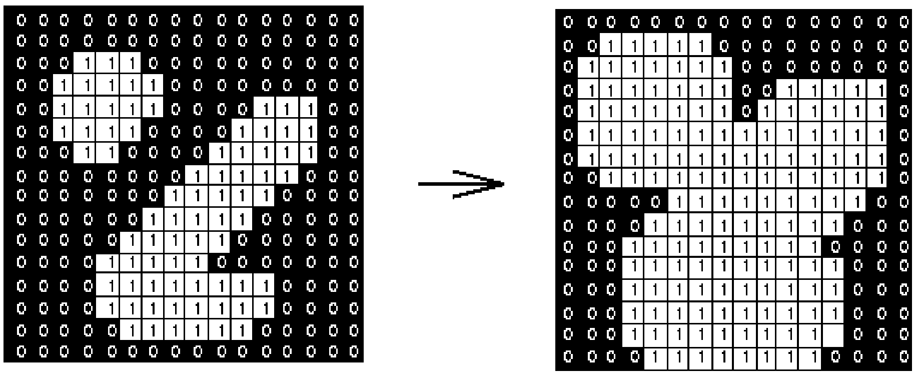


Figure 3.14: Example of dilation on a binary mask.

Erosion is the morphological dual to dilation, that combines two sets using vector subtraction of set elements. For this reason, erosion is also known as shrink or reduce operation. Let A and B sets in E^N, with elements a and b respectively. Then the *erosion* of A by B is defined as

$$A \ominus B = \{c \in E^N \mid c + b \in A, \ \forall b \in B\}$$

which can also be expressed as the set of differences between elements a and b, namely

$$A \ominus B = \{c \in E^N \mid \forall b \in B \ \exists a \in A \text{ such that } c = a - b\}. \tag{3.33}$$

Hence, the erosion of an image A by a structuring element B is the set of all elements $c \in E^N$ for which B translated to c is contained in A, as defined in [20].

Moreover, we can state that:

$$A \ominus B = \{c \in E^N \mid (B)_c \subseteq A\},$$

meaning that the structuring element B can be visualized as a probe sliding across the image A. When B translated to c ca be contained in A, the c belongs to the erosion of A

by B. This erosion can be rewritten as intersection of translation

$$A \ominus B = \bigcap_{b \in B} (A)_{-b}. \tag{3.34}$$

Erosion is widely considered as a shrinkage of the objects in the original image, hence the eroded set is thought as contained in the original set. This gives the anti-extensive property to the erosion transformation, but actually erosion is necessaruly anti-extensive only if the origin belongs to the structuring element, namely

$$\text{if } 0 \in B \rightarrow A \ominus B \subseteq A.$$

Let $A^c = \{x \in E^n \mid x \notin A\}$ be the complement of A, let $\hat{B} = \{x \mid \text{ for some } b \in B,\ x = -b\}$ be the reflection of B.

As mentioned before, dilation and erosion are dual operations and the duality is stated as

$$(A \ominus B)^c = A^c \oplus \hat{B}.$$

Figure 3.15 illustrates an example of erosion on the same binary mask used in Figure 3.14.

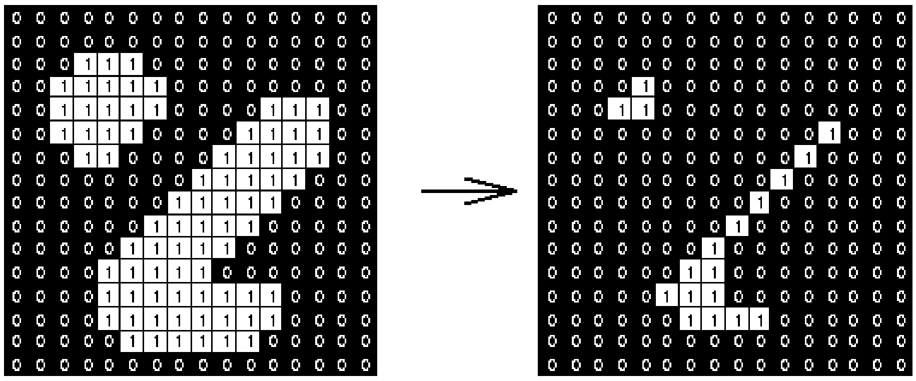


Figure 3.15: Example of erosion on a binary mask.

3.6.2. Opening and closing

In practice, dilations and erosions are applied together: an image dilation followed by an erosion of the dilated result, or vice versa. Alternatively, these two transformations can be applied iteratively to delete specific details smaller than the structuring element.

Opening combines erosion and dilation in order to smooth the contour in an image, eliminating peaks and protrusions.

Instead, closing combines dilation and erosion to remove small holes, fill gaps, but also smooth contour areas.

Opening of image A by structuring element K is denoted by $A \circ K$ and defined by

$$A \circ K = (A \ominus K) \oplus K.$$

Closing of image A by structuring element K is denoted by $A \bullet K$ and defined as

$$A \bullet K = (A \oplus K) \ominus K.$$

This means that opening is an erosion followed by a dilation, closing is a dilation followed by an erosion.

Image transformations applying iteratively dilations and erosions are idempotent, meaning that their reapplication effects no further changes to the previous transformed results [14]. From $A \oplus K = (A \oplus K) \circ K = (A \bullet K) \oplus K$, follows the idempotency of closing operation:

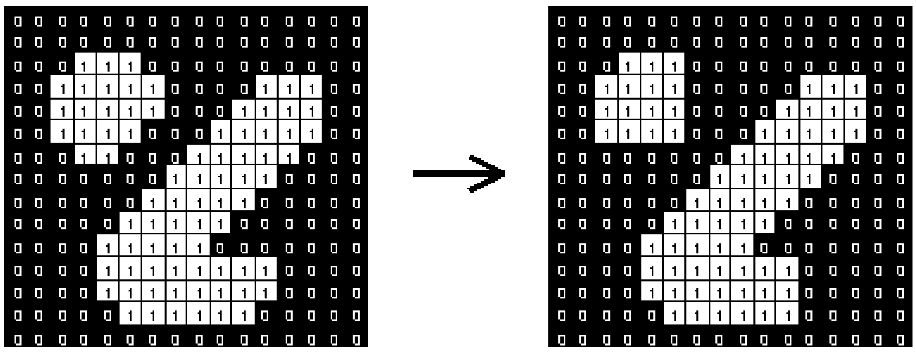
$$(A \bullet K) \bullet K = A \bullet K.$$

Similarly, from $A \ominus K = (A \circ K) \ominus K = (A \ominus K) \bullet K$ follows the idempotency of opening transformation:

$$A \circ K = (A \circ K) \circ K.$$

In Figure 3.16 we observe the difference between the effects of opening and closing: opening operation smooths the edges, while closing operation fills the holes.

Differently from erosion and dilation, opening and closing are invariant to translation of the structuring elements, namely $A \circ (B)_x = A \circ B$ and $A \bullet (B)_x = A \bullet B$.


(a) Opening

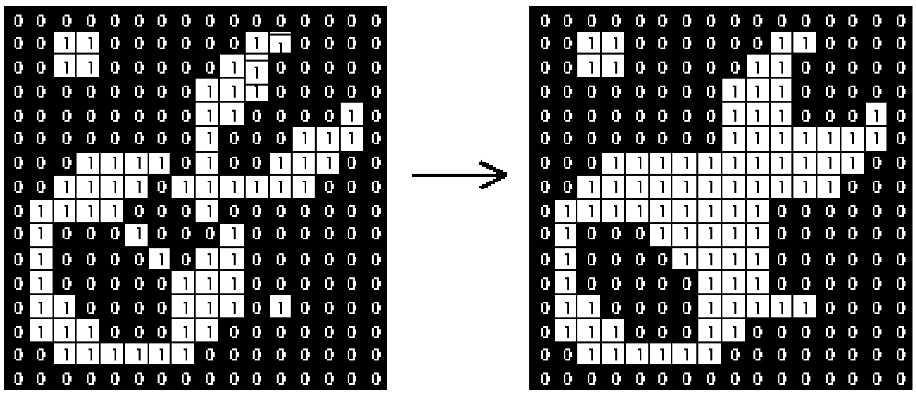

(b) Closing

Figure 3.16: Example of opening (3.16a) and closing (3.16b) morphological operations on a binary mask.

4 | Proposed solution

In this chapter we illustrate our solution to the anomaly detection problem discussed in the previous chapters. More specifically, we introduce our first approach to the matter in Section 4.1, as a general anomaly detection problem. Then, in Section 4.2 we present the dataset on which we developed our work, with the any modifications we made. In Section 4.3 we explain how to use the methods illustrated in the Background Chapter in order to improve the segmentation and anomaly detection effectiveness in our problem, by selecting the images based on the land cover types included in the images. Eventually, in Sections 5.1 and 5.2 we describe, respectively, the preprocessing and postprocessing phases that we apply in our method.

4.1. Autoencoders for Anomaly Detection

We address the anomaly detection problem using an *Autoencoder*, namely a Convolutional Neural Network that is trained with the objective of reconstructing the input as desired output [35]. As explained in Section 3.3, the encoder $\mathcal{E}$ maps the input into a low-dimensional latent space, from which the decoder $\mathcal{D}$ reconstruct the input image $\bar{X} = \mathcal{D}(\mathcal{E}(X))$.

Our idea is to train the encoder and the decoder using only normal images, namely without illegal landfills, such that the difference between the original image and the reconstructed image is minimized. In this way, when we give in input to the model an image that is not normal, the pixel-wise error between the original image and the reconstructed image is high in the anomalous areas. When the reconstruction error is below a certain threshold τ, the image pixel $X(i,j)$ is considered normal, while above the threshold the pixel is considered anomalous, namely

$$\begin{cases} \ell(X(i,j), \mathcal{D}(\mathcal{E}(X(i,j)))) \leq \tau & \text{if normal} \\ \ell(X(i,j), \mathcal{D}(\mathcal{E}(X(i,j)))) > \tau & \text{if anomalous} \end{cases}, \tag{4.1}$$

with $\ell(\cdot,\cdot)$ the reconstruction loss of the Autoencoder.

However, since in our dataset the concept of normality is too broad, namely the normal images are highly heterogeneous, training the model on the entire set of normal images results in poor segmentation performances. Therefore, we want to train a model on a limited amount of normal images, in order to have a more coherent training set. We decide to split the images into smaller groups according to their land cover, namely what type of activities are performed in the region captured in the images (for instance sparse and nucleiform residential fabric, or agricultural production facilities), or what type of ground is present (for instance medium and high density coniferous forests, or natural water basins). Formally, we create N groups of images

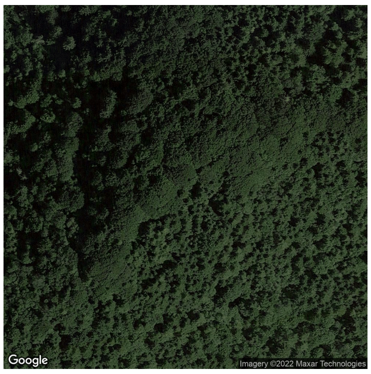
$$\mathcal{X}_1, \dots \mathcal{X}_N \subseteq \mathcal{X} \quad \text{s.t.} \quad \bigcup_{i=1}^{N} \mathcal{X}_i \approx \mathcal{X},$$

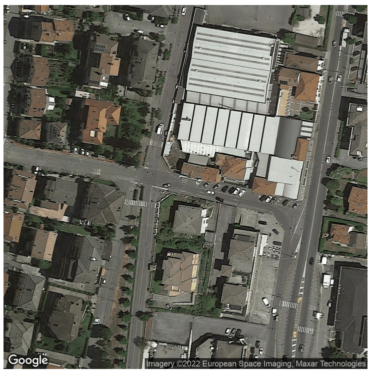
with $\mathcal{X}_i$ set of images consistent from the land cover perspective.

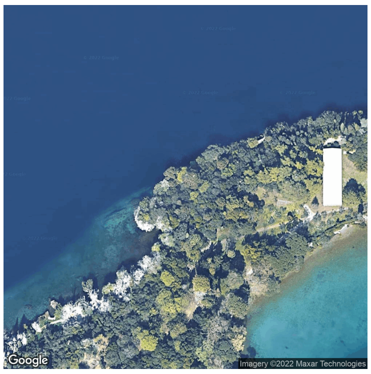
4.2. Dataset

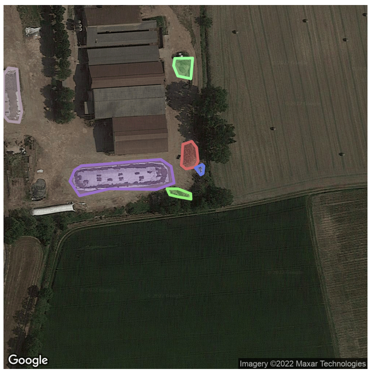
In order to develop this thesis, we worked with the AerialWaste dataset [30] that collects satellite images of Lombardia region under the control of ARPA agency [1], the environment monitoring agency of Lombardia region. The dataset was originally created to classify the satellite images containing illegal landfills as legal or illegal.
The latest version of this dataset is composed by 10977 satellite images taken by three different sources: AGEA Ortophotos, WorldView-3 and GoogleEarth. Most images containing illegal landfills are characterized by metadata about the evidence, severity, and area type of the site. Among the test set, a subset of anomalous images is annotated with the class of the waste objects visible in the image, chosen among 22 different categories. Between these images, 169 are provided with segmentation masks in the standard COCO format that can be used as Ground Truth (GT) labels.
Dataset images are already split into training set (75% of the total number of images) and test set (25% of the total). We decided to follow the division provided by the dataset authors, even though in the training we use only normal images, namely 5579 images of the training set.

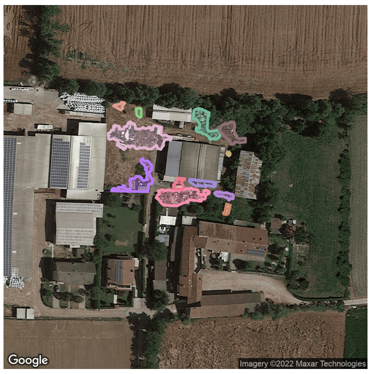
The images from different sources have different GSD (Ground Sampling Distance), namely a different pixel resolution based on the type of source from which the images come from. Images from AGEA Ortophotos have GSD equal to 20 cm, images from WorldView-3 have GSD of 30 cm, and images from GoogleEarth have GSD of 50 cm.


(a) 3121, 31311


(b) 1121, 1122, 1221, 12111, 1112


(c) 3113, 5121


(d) 12112, 2111


(e) 12112, 2111, 11231

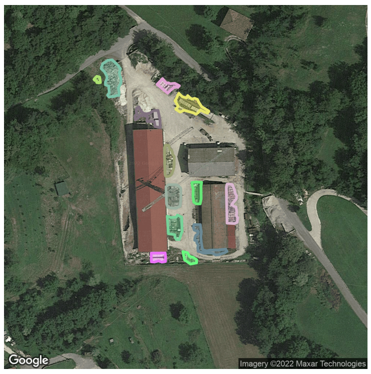

(f) 1123, 31111, 2311, 3241, 12112

Figure 4.1: The first three images are labeled as *normal*: (a) has categories "medium and high density coniferous forests (3121)", "medium and high density mixed forests governed by coppice (31311)"; (b) has "discontinuous residential fabric (1121)", "sparse and nucleiform residential fabric (1122)", "road networks and ancillary spaces (1221)", "industrial, craft, commercial settlements (12111)", "medium dense continuous residential fabric (1112)"; (c) has "riparian formations (3113)", "natural water basins (5121)". Instead (d), (e), (f) are *anomalous* images with their segmentation masks, the first one belonging to the original dataset, the last two manually segmented by us: (d) has land cover categories "agricultural production facilities (12112)", "simple arable land (2111)"; (e) has the same categories of (d) and also "farmhouses (11231)"; (f) has categories "sparse residential fabric (1123)", "medium and high density deciduous forests governed by coppice (31111)", "permanent meadows in the absence of tree and shrub species (2311)", "bushes with significant presence of tall shrub and tree species (3241)", "agricultural production sites (12112)".

DUSAF, which stands for "Destinazione d'Uso dei Suoli Agricoli e Forestali"(namely for "Agricultural and Forest Land Cover"), is a detailed geographical database created in the early 2000s containing all the information about land cover in the Lombardia region. The latest version is DUSAF 7.0 which includes orthophotos of the territories updated in 2021. Thanks to DUSAF 7.0 dataset published in the Geoportal of Lombardia region [12], we were able to associate each image to one or (usually) more types of land covers in the form of numerical codes. Indeed, AerialWaste dataset have no annotations about the land cover of each image, forcing one to consider as *normal* a huge variety of images. Figure 4.1 illustrates images from different land cover types, (a), (b) and (c) are normal images, while (d), (e) and (f) anomalous images with their ground truth segmentation masks.

4.3. Methods

The idea of our solution is to create N small groups of strongly related land cover types, where the concept of *normality* has limited variability within the group. This means that the land covers in the same group are highly connected, from a semantic point of view but also from a practical perspective, namely they are frequently close on the Lombardia territory. Thanks to these groups identifying different types of land areas, we can divide images into smaller subsets. The images in these subsets will be similar (from the land cover perspective) to the images within the same subset, and they will be dissimilar from the images of different subsets. On the found subsets of images, we train an Autoencoder that will be specific for the land covers of each group. Then, at test time, we evaluate the performances of that specific Autoencoders on normal and anomalous test images belonging to the same group.

In order to select only a subset of normal images to focus the work on, we group the different type of land cover using a *clustering* algorithm. To achieve this, we build an undirected weighted graph $G = (V, E)$ where each node $i \in V$ is a different type of land cover, and each edge $e_{ij} \in E$ connecting i and j has weight w_{ij} equal to the number of images containing simultaneously i and j as land covers. We notice that the created graph has all nodes connected, namely all land cover types occur in conjunction with other land covers in the dataset images and never alone. The weight on each edge is a function of *similarity* between the two vertices that the edge connects, namely represents the similarity between two land cover types. Our aim is to highlight the connections between types of land cover that are strongly related, since a high edge weight corresponds to a high number of images where those types are co-present, meaning that those land covers are close on Lombardia territory. In Figure 4.2 is shown the undirected weighted graph we just described, where the land covers are the nodes and the edge weights are emphasized by the line thickness.

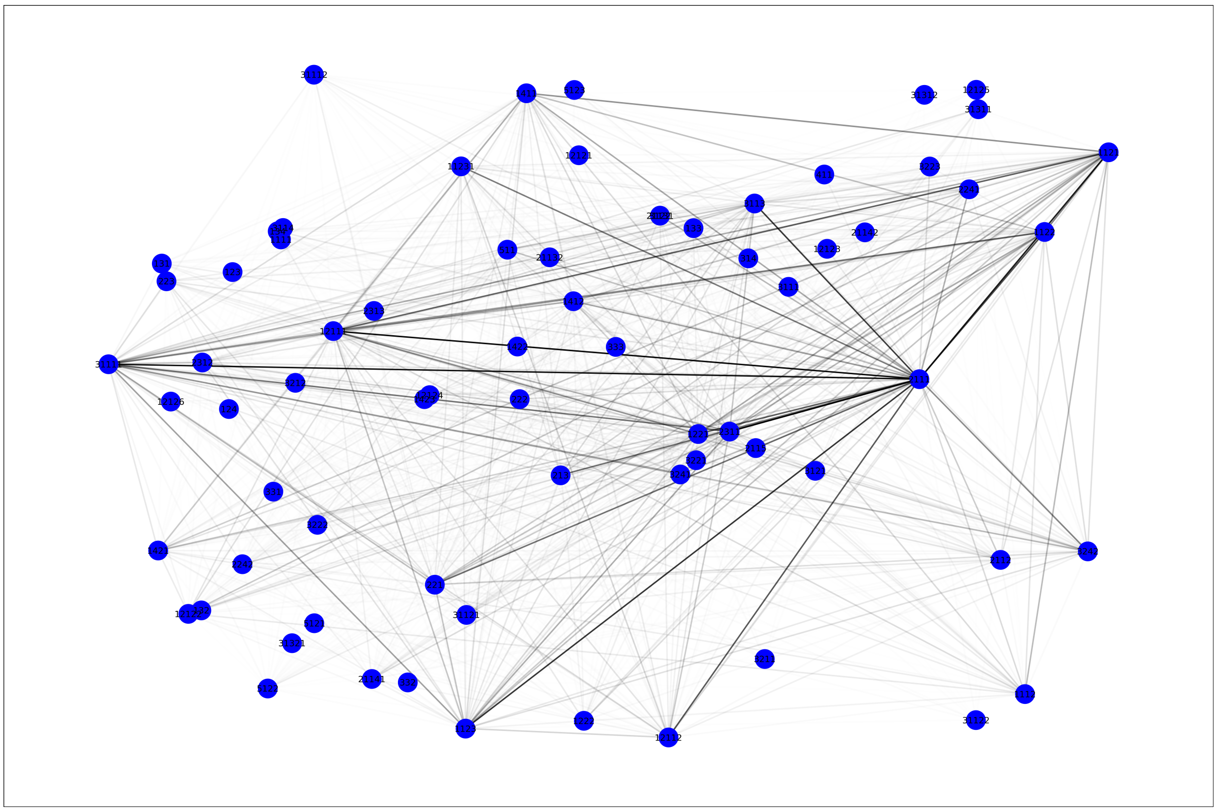


Figure 4.2: The undirected weighted graph built using land cover types as nodes. The thickness of the edges reflects the edge weight.

Hence the land cover types are the vertices of our graph and to group them we apply an algorithm of minimal cut, as the normalized minimal cut algorithm provided by Shi and Malik [28] that we discussed in Section 3.5.1. We partition the graph G into disjoint sets of vertices $A_1, \dots, A_K$, ie such that $\bigcup_{i=1}^{K} A_i = V$ and $\forall i, j \ i \neq j, \ A_i \cap A_j = \emptyset$, by removing edges connecting the two sets. As explained in Section 3.5, the problem can be rewritten relaxing the hypothesis to real values only, and found as the solution of the generalized eigenvalue system

$$(\mathbf{D} - \mathbf{W})y = \lambda \mathbf{D} y \tag{4.2}$$

where $\mathbf{D}$ is the $N \times N$ diagonal matrix with the total connection $d(i) = \sum_j w_{ij}$ from node i to all the other vertices on its diagonal, $\mathbf{W}$ is the $N \times N$ symmetrical matrix with the graph weights as elements $\mathbf{W}(i, j) = w_{ij}$. In order to catch possible relations between land cover types in a context of normality, we perform this analysis on the training set, where there are only *normal* images, namely without illegal landfills.

In order to choose the optimal number of clusters k, we look for the value k maximizing the eigengap, namely the difference between consecutive eigenvalues, that turns out to be

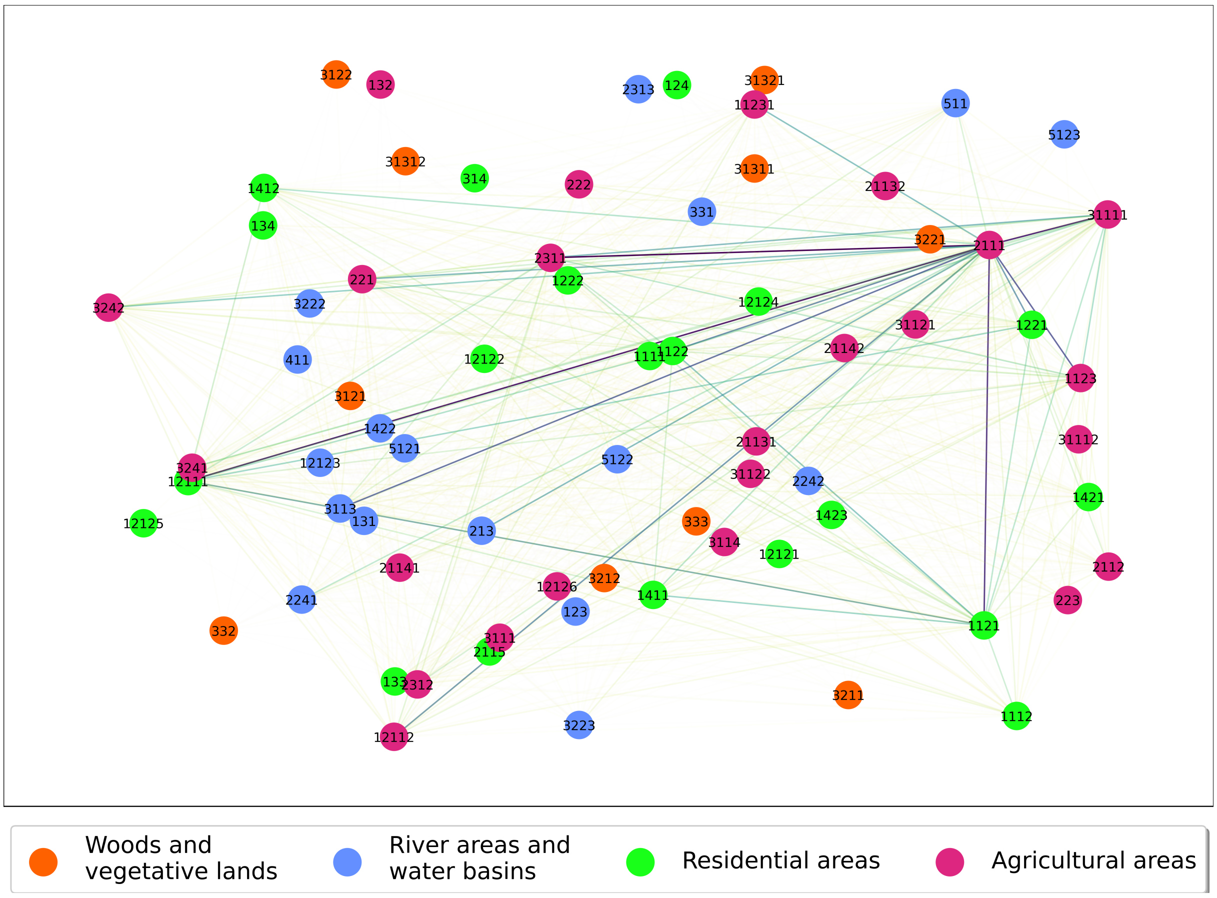


Figure 4.3: The clustered graph found using spectral clustering: each node i is a different land cover, each edge (i, j) has weight w_{ij} equal to the number of images containing simultaneously land covers i and j. The colour of the edges stresses the value of its weight (lighter colour smaller weight, darker colour bigger weight). The clusterization represents the main groups of land covers we can find in Lombardia territory.

$k = 4$. Performing the spectral clustering based on land covers, we discover 4 different groups, as shown in Figure 4.3, that can be referred to as:

0. Woods and vegetative lands
1. River areas and water basins
2. Residential areas
3. Agricultural areas.

In Tables 4.1, 4.2, 4.3 and 4.4 are illustrated the numerical codes with their description of, respectively, Woods and vegetative lands cluster, River areas and water basins cluster, Residential areas cluster and Agricultural areas cluster.

Code	Description
31311	Coppice-governed mixed forests of medium and high density
3121	Coniferous forests of medium and high density
31312	Medium- and high-density mixed forests managed as copses
3221	Bushes
332	Debris mounds and lithoid outcrops devoid of vegetation
333	Sparse vegetation
3211	High-altitude natural grasslands without tree and shrub species
3212	Natural high-altitude grasslands with scattered tree and shrub species
3122	Low-density coniferous forests
31321	Mixed low-density coppice-governed forests

Table 4.1: Codes and descriptions of the land covers in *Woods and vegetative lands* cluster, found using *Normalized cut* algorithm.

Figures 4.5 and 4.6 show the distribution of different land cover types in the clusters previously found, considering only training images. Instead, in Figures 4.7 and 4.8 are illustrated the distribution of the clustered land cover types considering the test set, where the red bars represent the amount of anomalous images with a specific land cover type. These clusters are intended as found with the normalized cut algorithm, as previously described.

We take into consideration also another type of clusterization, splitting the vertices into communities using the Louvain algorithm [8] explained in 3.5.2. Following the algorithm, we group the nodes, namely the land cover types, into 3 clusters which can be summarized as:

0. Anthropized areas
1. Wooded areas
2. Agricultural areas.

In Figure 4.4 the result of graph clusterization is shown. As before, each node is represented by a numerical code specifying the land cover.

However, we decide to use the clusters shown in Figure 4.3 found with the normalized cut algorithm, since they are more specific. Indeed, using normalized cut algorithm river areas are separated from agricultural and residential areas, resulting in a more precise

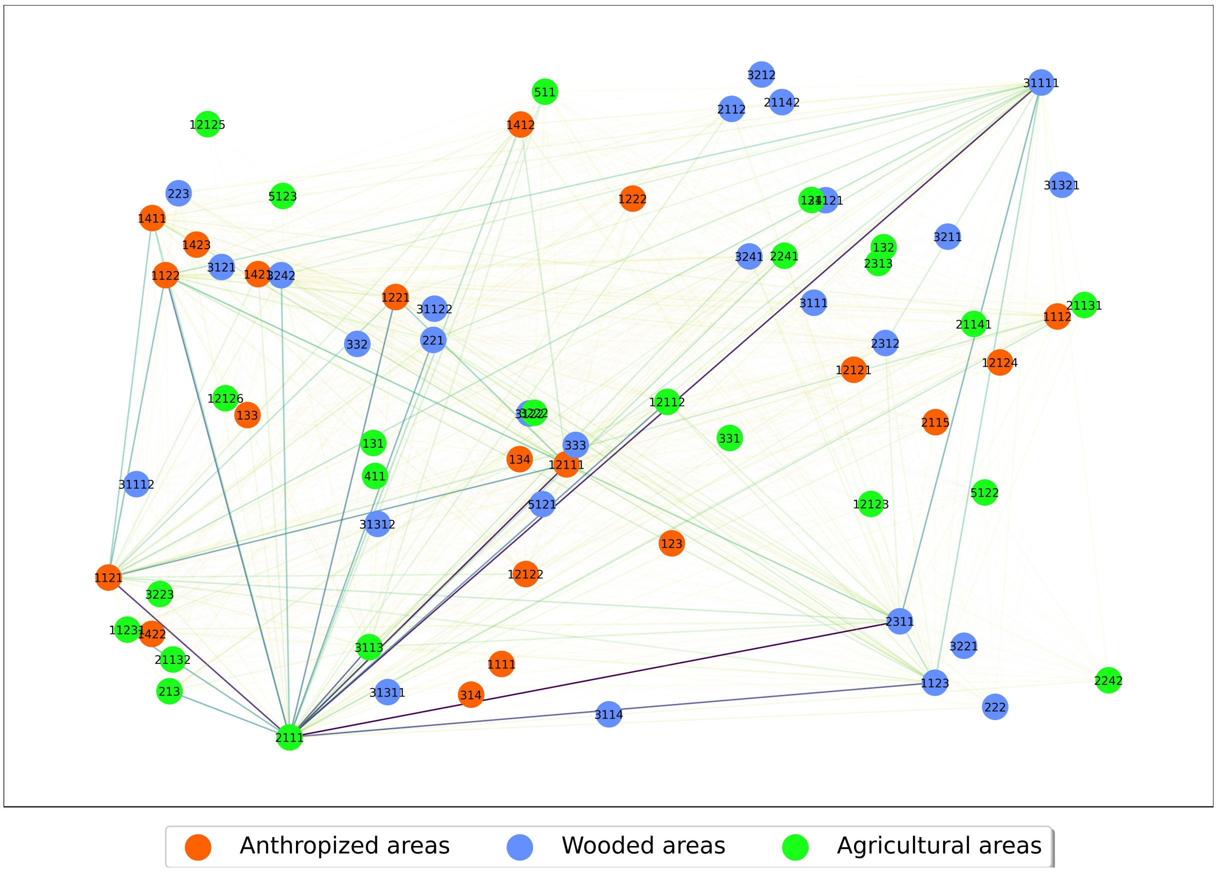


Figure 4.4: The clustered graph found with *Louvain* algorithm: each node i is a different land cover, each edge (i, j) has weight w_{ij} equal to the number of images containing simultaneously land covers i and j. The colour of the edges stresses the value of its weight (lighter colour smaller weight, darker colour bigger weight). The clusterization represents the main groups of land covers we can find in Lombardia territory.

clusterization. Our study is developed on the Agricultural areas cluster found with the normalized cut algorithm, which contains the land cover types listed in Table 4.4.

Code	Description
3223	Vegetation of raised banks
2241	Poplar groves
12123	Technological installations
3113	Riparian formations
511	Riverbeds and artificial watercourses
213	Rice paddies
331	Beaches, dunes and gravel banks
3222	Vegetation of riverbanks
2242	Other agricultural woodlands
5121	Natural watersheds
1422	Campsites and tourist and accommodation facilities
5122	Artificial water basins
123	Port areas
131	Quarries
411	Vegetation of inland wetlands and peat bogs
5123	Reservoirs from mining activities affecting the water table
2313	Water meadows

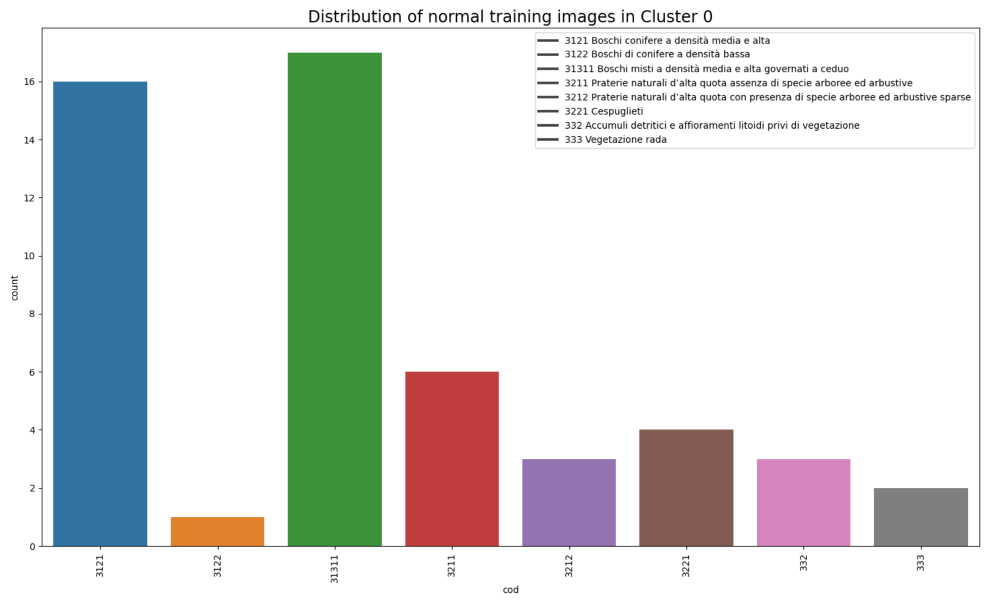
Table 4.2: Codes and descriptions of the land covers in *River areas and water basins* cluster, found using *Normalized cut* algorithm.

Code	Description
1111	Continuous dense residential fabric (>80% - large residential buildings)
1112	Medium dense continuous residential fabric (>80% - small residential buildings)
1121	Discontinuous residential fabric (50 - 80%)
1122	Sparse, nucleiform residential fabric (30 - 50%)
1411	Parks and gardens
12111	Industrial, craft, commercial settlements
12122	Public and private facilities
1221	Road networks and ancillary spaces
1421	Sports facilities
1412	Unplanted green areas
12121	Hospital settlements
1222	Railway networks and ancillary spaces
2115	Kitchen gardens
12124	Cemeteries
134	Unused and unvegetated degraded areas
133	Construction sites
314	Recent reforestation
1423	Amusement parks
124	Airports and heliports
12125	Obliterated military areas

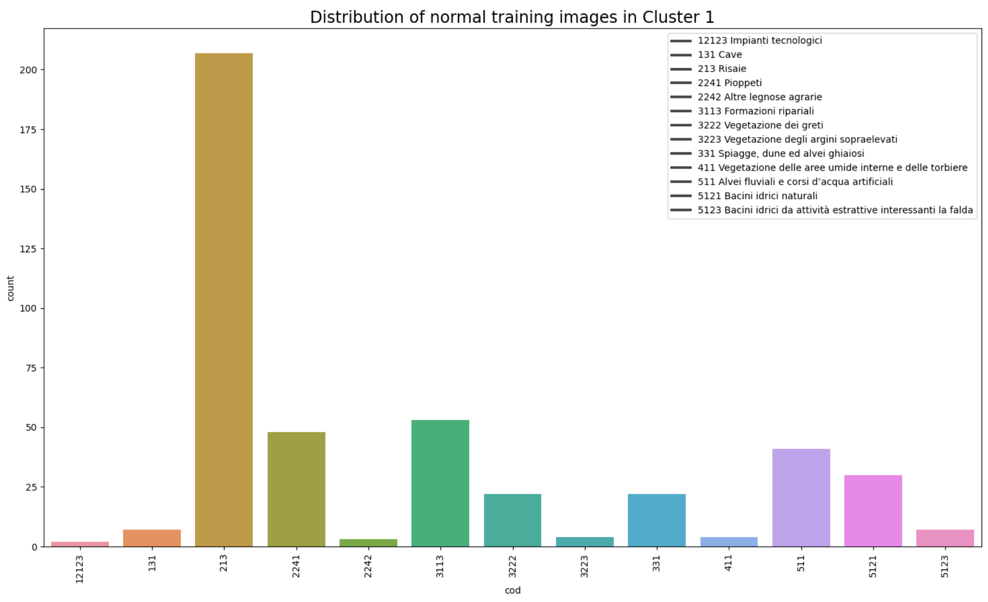
Table 4.3: Codes and descriptions of the land covers in *Residential areas* cluster, found using *Normalized cut* algorithm.

Code	Description
223	Olive groves
31111	Medium and high density deciduous forests governed by coppice
2111	Simple arable crops
2311	Permanent meadows in the absence of tree and shrub species
2112	Arable land trees
12112	Agricultural production sites
3241	Bushes with significant presence of tall shrub and tree species
3242	Bushes in abandoned agricultural areas
1123	Sparse residential fabric
11231	Farmhouses
221	Vineyards
2312	Permanent meadows with scattered tree and shrub species
31122	Low density broad-leaved forests governed by high trunk
222	Orchards and minor fruit
21141	Open field floro-nursery crops
21142	Protected floro-nursery crops
21131	Vegetable crops in open field
21132	Protected horticultural crops
31121	Low density deciduous forests governed by coppice
31112	Medium and high density broad-leaved forests governed by high stems
12126	Photovoltaic systems on the ground
3114	Chestnut groves
3111	Medium and high density hardwood forests

Table 4.4: Codes and descriptions of the land covers in *Agricultural areas* cluster, found using *Normalized cut* algorithm.

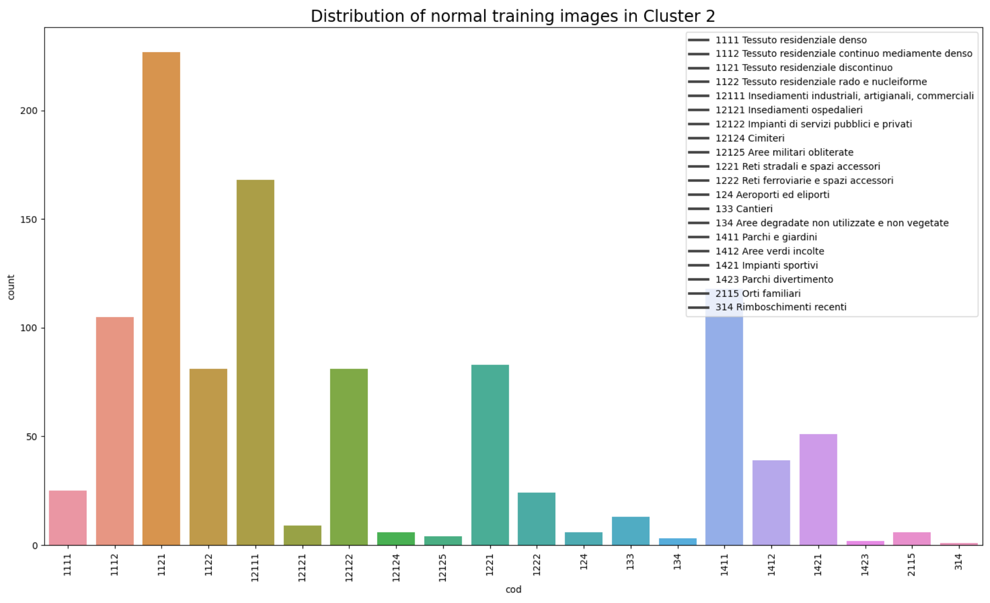


(a) Woods and vegetative lands cluster

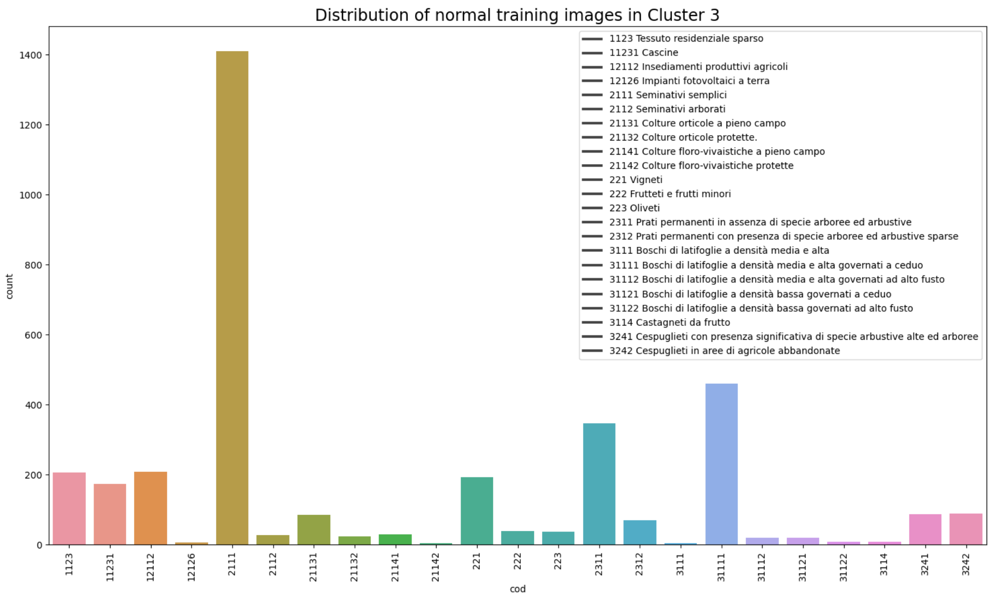


(b) River areas and water basins cluster

Figure 4.5: Distribution of the first two spectral clusters in the training set images, found with normalized cut algorithm.

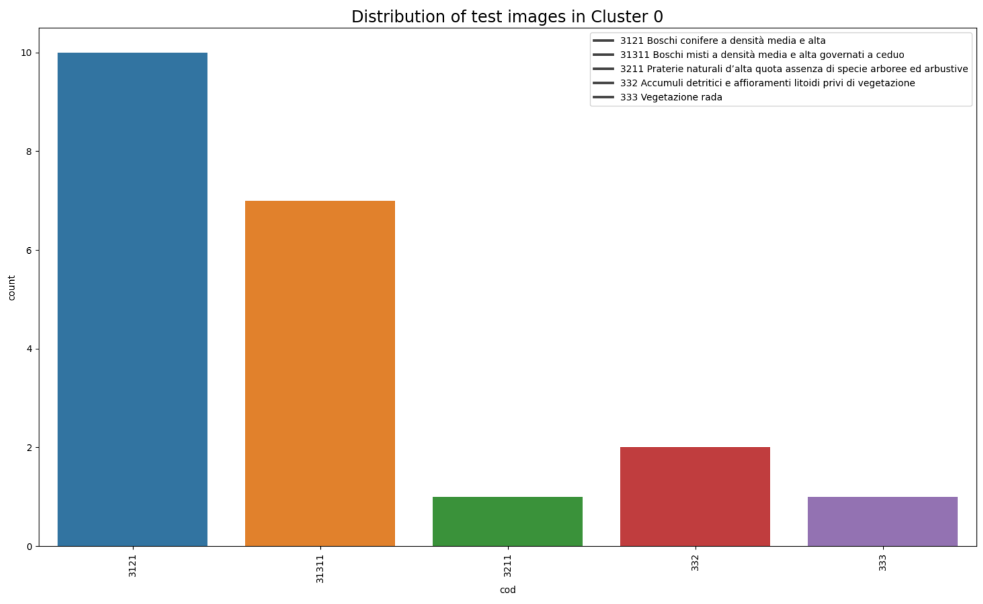


(a) Resident areas cluster

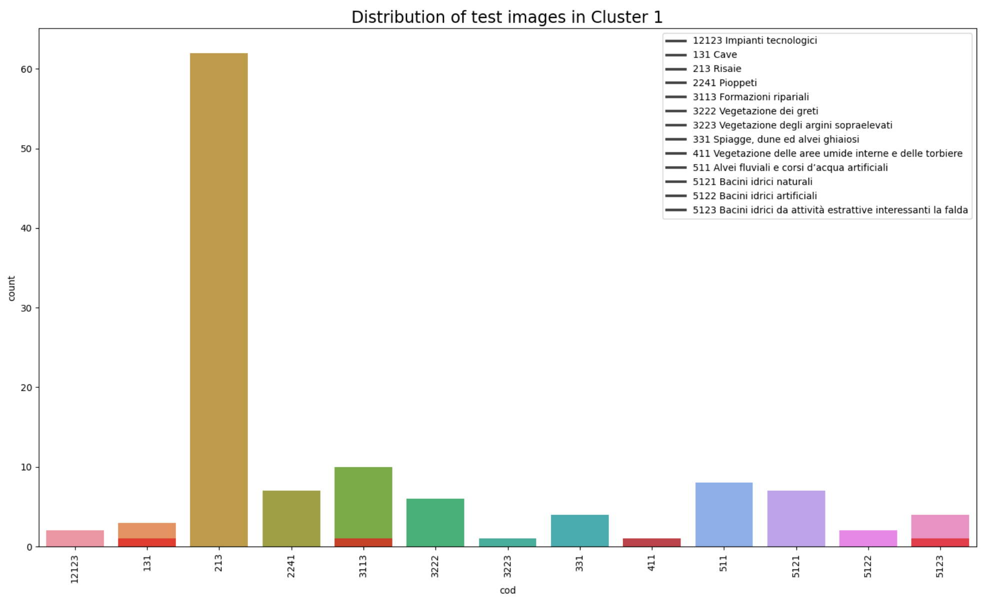


(b) Agricultural areas cluster

Figure 4.6: Distribution of the last two spectral clusters in the training set images, found with normalized cut algorithm.

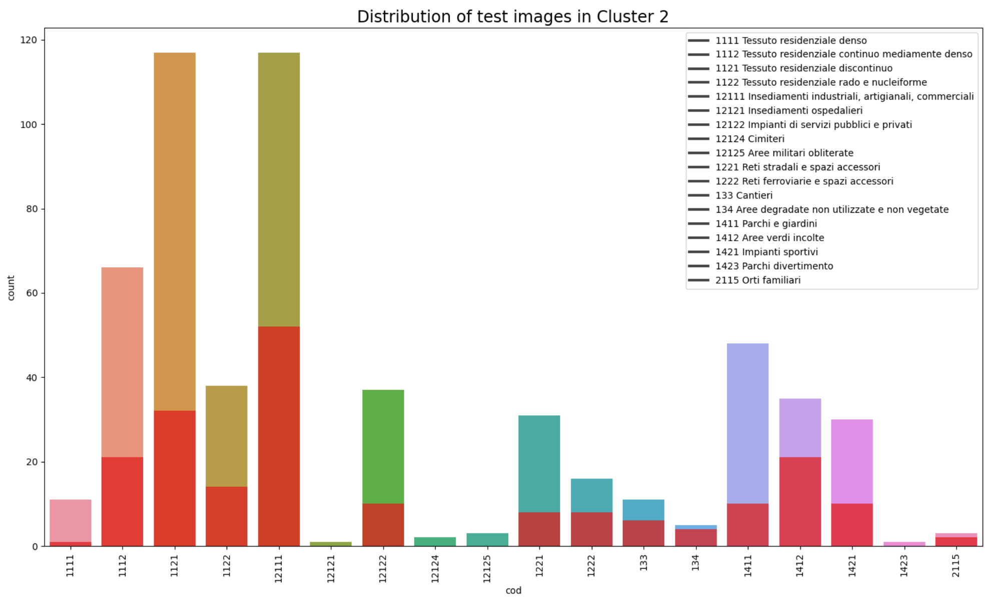


(a) Woods and vegetative lands cluster

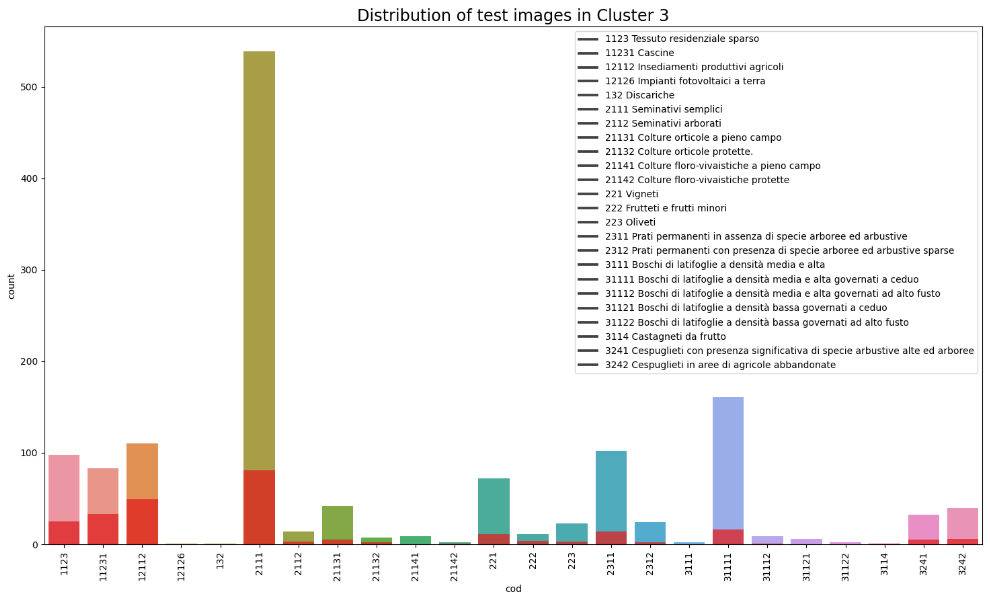


(b) River areas and water basins cluster

Figure 4.7: Distribution of the first two spectral clusters in the test set images, found with normalized cut algorithm. The red bars stand for the amount of anomalous images for each land cover type.



(a) Resident areas cluster



(b) Agricultural areas cluster

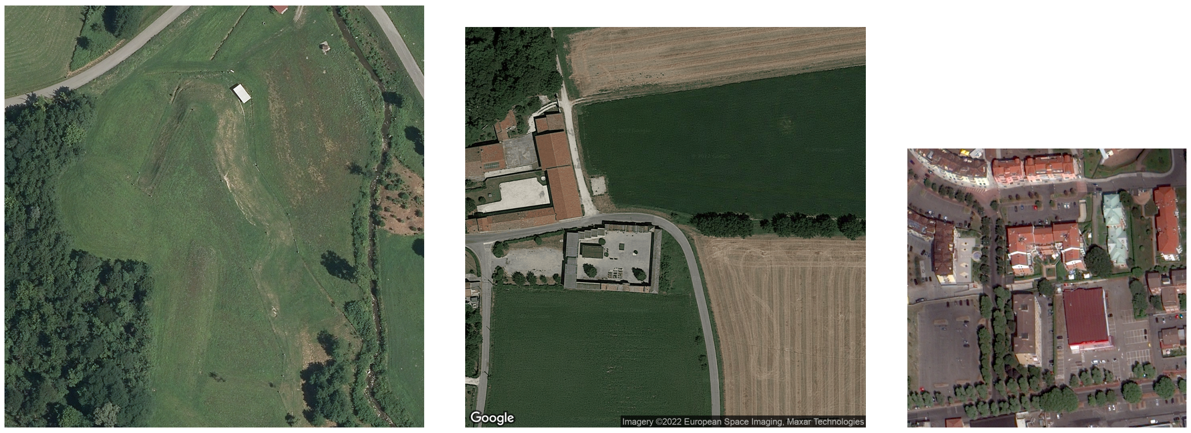
Figure 4.8: Distribution of the last two spectral clusters in the test set images, found with normalized cut algorithm. The red bars stand for the amount of anomalous images for each land cover type.

5 | Implementation details

In this chapter we describe the implementation of our models, the training loss used during the training and the model architecture. We also illustrate the preprocessing and postprocessing we apply to our data.

5.1. Preprocessing

The images in AerialWaste dataset have different GSD (Ground Sampling Distance), namely a different pixel resolution, based on the type of source from which the images were collected. Indeed, images from AGEA Ortophotos have GSD of 20 cm per pixel, from WorldView-3 have GSD of 30 cm per pixel, and images from GoogleEarth have GDS of 50 cm per pixel.



(a) AGEA source (b) GE source (c) WV3 source

Figure 5.1: Example of three normal images from the training set with different sources and size.

In Figure 5.1 three examples of different GSD based on the source of the image. Figure 5.1a has AGEA Ortophotos as source, with size 1048×1054, 5.1b has source Google Earth and size 1000×1000, while 5.1c is an image from WorldView-3 with size 697×697. In order to have all the images coherently with the same pixel resolution, we scale them according to their source, choosing to set 30 cm/pixel for all of them.

The designed network is fully convolutional, meaning that it can take in input images of all sizes. However, in order to help the learning, we create batches of patches of the same size randomly cropped from the training images, and we give them in input to the econder.

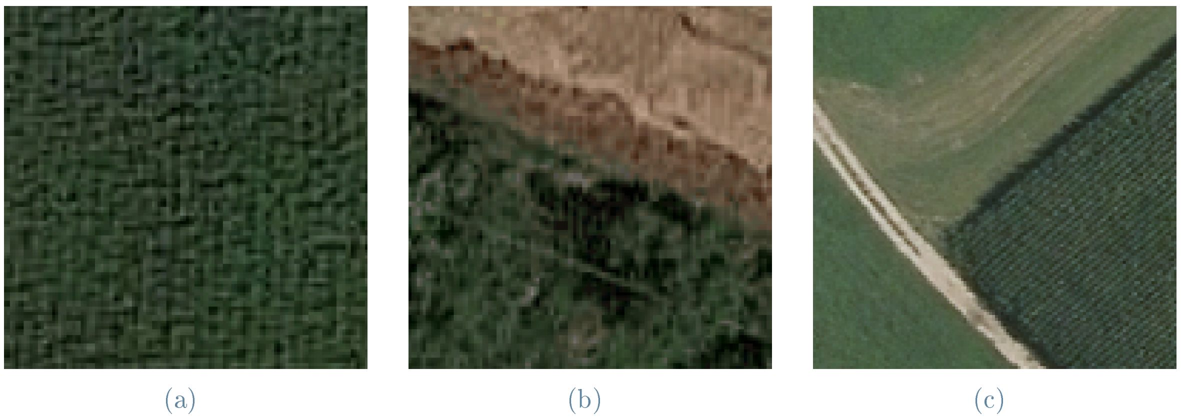


Figure 5.2: Example of three patches of size 128×128 cropped from normal images from the training set.

5.2. Postprocessing

During the testing phase, training losses are used to create 2-dimensional error maps, which are the reconstruction errors of the images. Considering an input image $X \in \mathbb{R}^{w \times h \times 3}$, the reconstruction error map of X can be described as a matrix $Z = \ell(X, \bar{X}) \in \mathbb{R}^{w \times h}$, with w the width of the image in input, and h the height of the image. We note that the values of Z are high where the image reconstruction fails, low (ideally equal to 0) where the image is well reconstructed by our model.

In order to create a *score map* as defined in Equation 2.1 of our Problem formulation in Chapter 2, we have to establish a criterion to discern areas predicted as anomalous from those considered normal. We choose a threshold τ equal to the empirical quantile at 98% and 99% of the error distribution on a subset of normal images, in order to have only the 2%, 1% of false positive in our images. Thereby, following Equation (2.1), the score map could be created in each pixel $X(i, j)$ as:

$$\Omega_X(i,j) = \begin{cases} 0 & \text{if } \ell(X(i,j), \bar{X}(i,j)) < \tau \\ 1 & \text{if } \ell(X_{ij}, \bar{X}_{ij}) \geq \tau \end{cases} \tag{5.1}$$

with i, j indicating the row and column locating the pixel in the image, $i \in \{0, \dots, w-1\}$, $j \in \{0, \dots, h-1\}$, ℓ the reconstruction loss, $\bar{X}$ the reconstructed image.

Because the raw score maps computed using eq. (5.1) are characterized by scattered positive pixels, while the ground truth masks usually display connected and compact regions, we perform postprocessing by applying the morphological methods discussed in Section 3.6. Our idea is to use morphological image postprocessing to smooth all the possible false positives in the score maps.
In particular, we first want to close the holes that can be present in the objects of our score maps. Then, we want to remove from the score maps all the noisy false detections that are too small to be a real piece of garbage and the object edges that sometimes are detected as anomalous but are too thin to be a landfill.

The available morphological operations are erosion, which shrinks the shape of an object in the image by removing pixels from its edge, dilation, which increases the shape by adding pixels, or their combination. Our decision is to combine closing, which applies first dilation and then erosion, followed by opening, which uses erosion before and dilation after.

In Figure 5.3 are illustrated some examples of score maps before using postprocessing and after postprocessing with quatile at 98% and 99%. As we can notice, the maps after the chosen sequence of postprocessing are the more definite, namely the one less noisy and with less holes in the objects predicted as anomalous, hence the more similar to the format of our ground truth masks we have. Moreover, the edges of what could be building that we observe having high score in the maps 5.3a and 5.3c, after the postprocessing are correctly removed, indeed they are too thin to be a landfill.

5.3. Neural network training

In this thesis we consider three different losses to train our networks. The first loss is *MSE* that computes the *mean squared error* between each pixel of the two images taken in comparison, namely the original and the reconstructed image, and it is defined as

$$MSE(X, \bar{X}) = \frac{1}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} (X(i,j) - \bar{X}(i,j))^2 \tag{5.2}$$

where the difference between images is considered pixel-wise, (i, j) the pixel ranging in n rows and m columns.

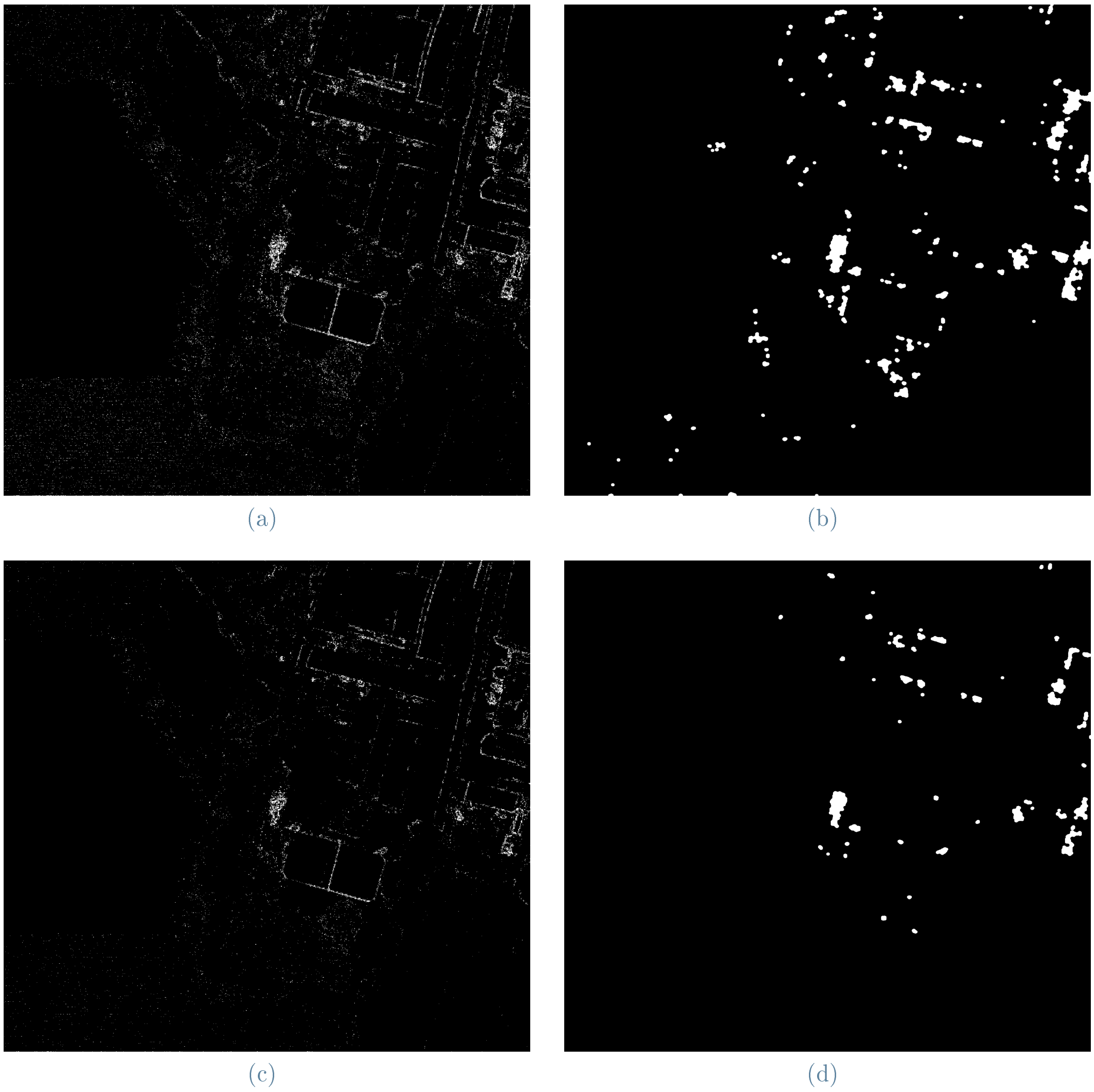

(a) (b) (c) (d)

Figure 5.3: Examples of score maps using quantile at 98% in (a) and (b), 99% in (c) and (d). (a) and (c) are the raw versions of the score maps, while (b) and (d) are the score maps after the application of postprocessing.

Besides, the *Structural Similarity Index* (*SSIM*) looks for the similarity between pixels of the two images, producing a score $\in \{-1, 1\}$, with 1 indicating the maximal similarity, as explained in Section 3.4. It is defined as

$$SSIM(X, \bar{X}) = \frac{(2\mu_X\mu_{\bar{X}} + C_1)(2\sigma_{X\bar{X}} + C_2)}{(\mu_X^2 + \mu_{\bar{X}}^2 + C_1)(\sigma_X^2 + \sigma_{\bar{X}}^2 + C_2)} \tag{5.3}$$

where μ_X, $\mu_{\bar{X}}$ are the mean of X and $\bar{X}$, σ_X, $\sigma_{\bar{X}}$ the variance of X and $\bar{X}$, $\sigma_{X\bar{X}}$ their covariance, C_1, C_2 constants to avoid instability when the sum of squared means is close to 0. In order to use this metric during the training as a loss, we considered $1-SSIM(X,\bar{X})$.

Finally, we defined a mixed loss as

$$\begin{aligned} \ell_{mixed} = \quad & w_{MSE} \cdot MSE(X,\bar{X}) \quad + \\ & w_{SSIM} \cdot (1 - SSIM(X,\bar{X})) \end{aligned} \tag{5.4}$$

with w_{MSE}, w_{SSIM} weights arbitrarily chosen as hyperparameters. Indeed, in this way we tried to combine the MSE reconstruction precision with the SSIM ability to grasp the luminance, contrast and structure of an image.

Before performing the training, we further split the cluster of agricultural areas into *frequent* and *rare* subsets. The frequent set contains only the most frequent land cover types, namely those with at least 50 images containing them, and the rare set contains the other more rare types, namely with less than 50 occurrences.

Code	Description
31111	Medium and high density deciduous forests governed by coppice
2111	simple arable crops
2311	Permanent meadows in the absence of tree and shrub species
12112	Agricultural production sites
3241	Bushes with significant presence of tall shrub and tree species
3242	Bushes in abandoned agricultural areas
1123	Sparse residential fabric
11231	Farmhouses
221	Vineyards
2312	Permanent meadows with scattered tree and shrub species
21131	Vegetable crops in open field

Table 5.1: Codes and descriptions of the *frequent* land covers in Agricultural areas cluster ($\geq$ 50 occurrences).

Code	Description
223	Olive groves
2112	Arable land trees
31122	Low density broad-leaved forests governed by high trunk
222	Orchards and minor fruit
21141	Open field floro-nursery crops
21142	Protected floro-nursery crops
21132	Protected horticultural crops
31121	Low density deciduous forests governed by coppice
31112	Medium and high density broad-leaved forests governed by high stems
12126	Photovoltaic systems on the ground
3114	Chestnut groves
3111	Medium and high density hardwood forests

Table 5.2: Codes and descriptions of the *rare* land covers in Agricultural areas cluster (< 50 occurrences).

Considering the Agricultural cluster, we extract from the training set those images considered as frequent, namely with land cover contained in Table 5.1, creating the frequent training set composed by 1560 normal images. From the test set we pick normal and anomalous images having land cover types in Table 5.1 in order to create the frequent test set. We also add anomalous images from the original training set (created by the authors of AerialWaste [30]) having land cover types belonging to the frequent subset. In this way, we build a frequent test set of 787 images, of which 285 are anomalous.
Finally, we create a rare test set with normal and anomalous images from both the training and test sets, containing all the images with land cover types belonging to Table 5.2 It consists of 307 images, of which 54 are anomalous. Our aim is to use the frequent training set during the training phase, the frequent test set during the testing phase, and the rare test set at the end of the testing phase in order to test the generalization ability of the network.

5.4. Autoencoder Architecture

Since the images have varying dimensions, we have designed a fully convolutional Autoencoder which is capable of accepting input images of any size. Thus, even if during the training phase we provide in input to the model batches of patches of the same size, in the testing phase we can give in input to our model entire images, without needing to crop them. In this manner our model will be able to detect anomalies in the entire given image.

Keeping in mind the size of the receptive field of our network, we crop the patches of size 128×128 pixels, namely 3840×3840 cm, and we create batches of 128 patches each. This way, our network is able to learn the normal ground accurately, using patches of size 128×128 pixels during training, and then apply what it has learned to full images of any size. This feature grants the network flexibility, as it can accept novel images of any size as input without the need for retraining.

Two different fully convolutional architectures are used. As shown in Figure 5.4, the first network has 3 convolutional layers in the encoder, with ReLu as activation function, padding = same, stride = 2 and 4, and in the decoder 3 convolutional layers with ReLu as activation function, padding = same, alternated with upsampling layers of sizes 2×2 and 4×4. We train this network with MSE loss.

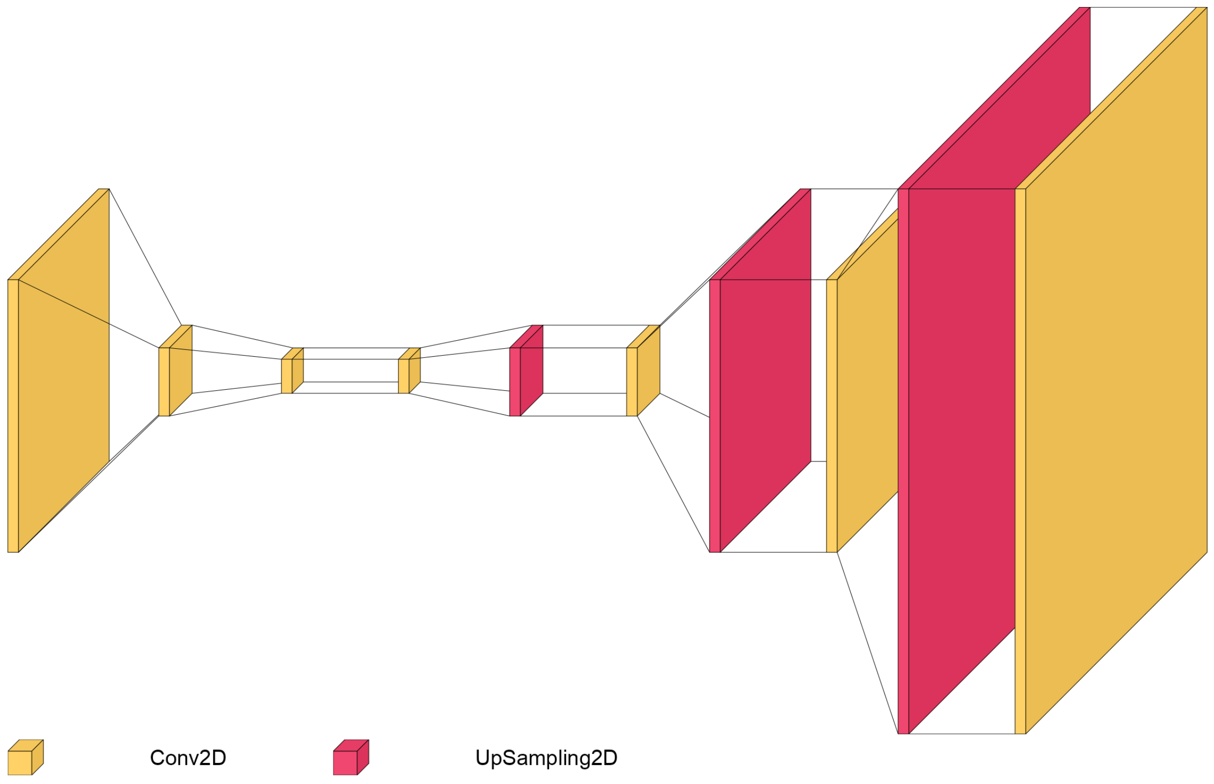


Figure 5.4: Architecture of the network trained with MSE loss.

A similar network is built with 4 layers instead of 3, as it can be observed in Figure 5.5. We train this model with the mixed loss between MSE and SSIM as defined in Equation

(5.4). After different numerical trials over the weights to be considered when applying the mixed loss, we decide to use weights $w_{MSE} = 0.7$, $w_{SSIM} = 0.3$.

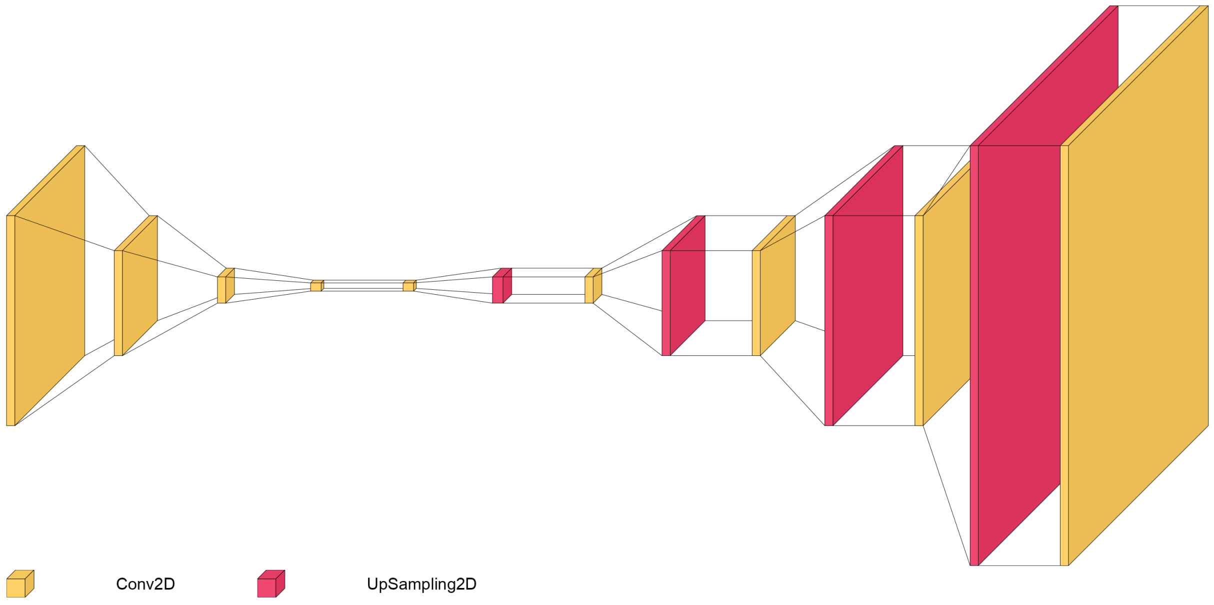


Figure 5.5: Architecture of the network trained with mixed MSE-SSIM loss.

6 | Experiments and performance evaluation

In this chapter we discuss the tests that were carried out throughout the experiments phase of the project. More precisely, we first introduce the figure of merits that we use during our experiments evaluation phase. Afterwards, we illustrate the numerical results obtained during our experiments and we compare them.

We perform three different type of model evaluation: we test the model performance using AUROC score, then we test the model ability to segment anomalous images into binary masks, eventually we test the model ability to find a sufficient amount of waste in anomalous images and not find any waste in normal images. The evaluation experiments are conducted on the test set, containing both normal and anomalous images. The first test set considered is the frequent test set, then the rare test set, both found in Section 5.3. Since the number of segmented images in the dataset was scarce, especially if we consider only the subset of images extracted by clustering, we manually segmented approximately 100 other images using Odin annotator [31] provided by the authors of AerialWaste dataset [30].

6.1. Figures of merit

We consider different evaluation metrics to assess our models. Let TP, TN, FP, FN be the True Positive, True Negative, False Positive, False Negative, namely the correct indication of the presence of an anomaly (TP), correct indication of the absence of an anomaly (TF), erroneous indication of the presence of an anomaly (FP), erroneous indication of the absence of an anomaly (FN), hence we can define:

$$Precision = \frac{TP}{FP + TP} \tag{6.1}$$

$$Recall = \frac{TP}{FN + TP} \tag{6.2}$$

$$IoU = \frac{Pred \cap GT}{Pred \cup GT} = \frac{TP}{FP + FN + TP} \tag{6.3}$$

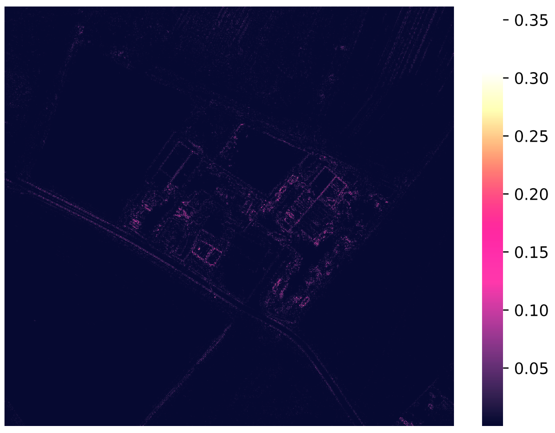
$$Dice = \frac{2(Pred \cap GT)}{Pred + GT} = \frac{2Precision \cdot Recall}{Precision + Recall} = \frac{2TP}{FP + FN + 2TP} \tag{6.4}$$

$$AUROC: \text{ area under the ROC curve} \tag{6.5}$$

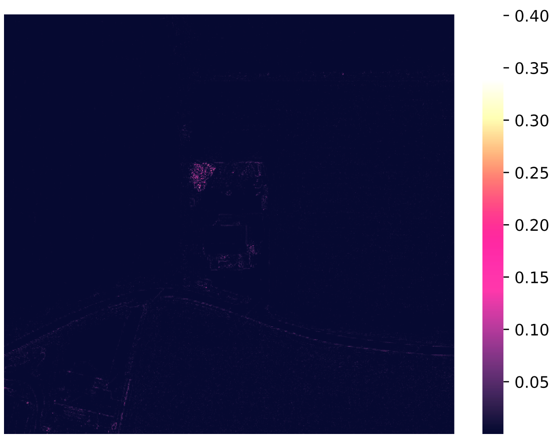
where $Pred$ stands for Predicted images, GT for Ground Truth, $Dice$ coefficient is also known as $F1$ score and the ROC curve shows the performance of models at all the thresholds, computed using the False Positive rate and the True Positive rate. These metrics will be used to evaluate the results obtained in our experiments.

6.2. Results and comparison

The reconstruction error maps are 2-dimensional matrices where each pixel $E(i, j)$ is the loss computed between original pixel $X(i, j)$ and the pixel of the reconstructed image $\bar{X}(i, j)$. Examples of reconstruction error maps for model trained with MSE and mixed loss are displayed in Figure 6.1 using heatmaps. We observe that there are higher values where the images are poorly reconstructed, namely where anomalies are predicted to occur.



(a)



(b)

Figure 6.1: Heatmaps showing examples of reconstruction error maps.

The ground truth masks are 2-dimensional binary matrices with pixel equal to 1 if the pixel belongs to an anomalous region, equal to 0 if the pixel is normal. We first estimate the AUROC score on the reconstruction error maps compared to the ground truth masks, finding the results shown in Table 6.1. We refer to the model trained with MSE loss with ℓ_{MSE}, to the model trained with the mixed loss as ℓ_{mixed}, both trained on the frequent training set defined in Section 5.3, and we consider as *Baseline* a model trained on the complete dataset.

	AUROC
Baseline	69.91 %
$\ell_{\mathbf{MSE}}$	70.12 %
$\ell_{\mathbf{mixed}}$	70.99 %

Table 6.1: Evaluation of the models trained on the frequent training set (Agricultural areas cluster) with different losses using AUROC score; as baseline we consider a model trained on the complete dataset.

Then, as explained in Section 5.2, we determine the score maps for the anomalous test images of the Agricultural cluster and we compute the *IoU* and *Dice* scores of those predicted maps with respect to the ground truth masks. Initially we do not apply the morphological postprocessing, but we notice that applying it on the score maps the results improve significantly. In Figure 6.2 an example of the difference between the predicted masks with (c) and without (b) postprocessing is shown. The raw mask is created considering at 99% the quantile threshold cutoff defined in Section 5.2. We observe that the computed raw score maps (b) and (e) contain clustered dots and scattered dots: by employing postprocessing in (c) and (f) the clustered dots are aggregated in larger connected components, while the scattered dots, which are sparsely distributed, can be considered as noise and are removed.

In Table 6.2 are listed the results considering quantile at 98% and 99% the threshold cutoff defined in Section 5.2, with and without applying postprocessing. We can observe that after the postprocessing the performances improve in all our models (except for the mixed loss model with the last threshold); the mixed-loss model has best AUROC score in Table 6.1 but it is the worst considering IoU and Dice scores in Table 6.2, where excels the MSE-based model. The main issue with these results is the false detection of anomalies where object edges are, for instance roof boarders, street edges, small items. We can notice it also in Figure 6.2, especially in the raw mask (e). This matter is directly related

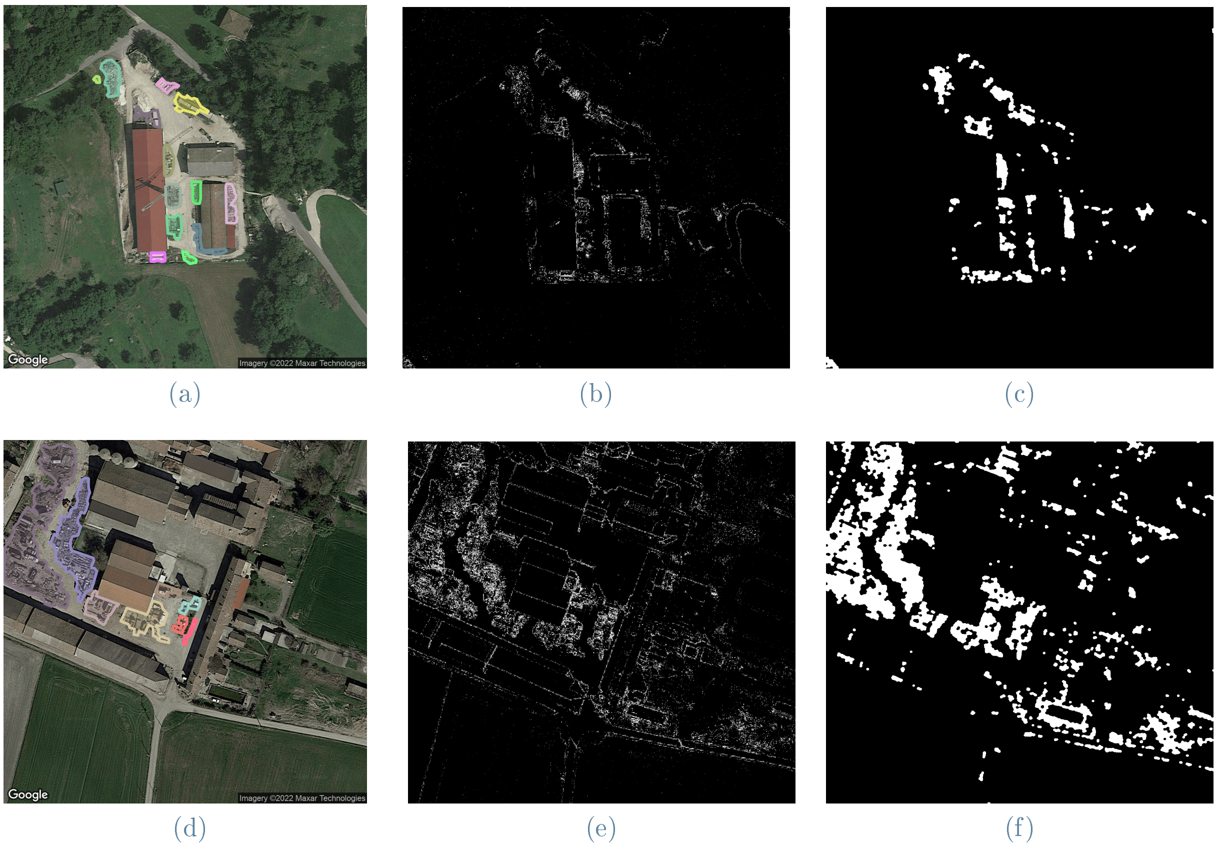

Google
Imagery ©2022 Maxar Technologies
(a)
(b)
(c)
Google
Imagery ©2022 Maxar Technologies
(d)
(e)
(f)

Figure 6.2: Images (a) and (d) are the GT masks, (b) is the raw mask computed through the model trained with MSE before postprocessing and has scores $IoU = 0.1509$, $Dice = 0.3017$, (c) after postprocessing has scores $IoU^{postpr} = 0.4704$, $Dice^{postpr} = 0.9408$. (e) is the raw mask computed using model trained with MSE before postprocessing and has scores $IoU = 0.1781$, $Dice = 0.3561$, (f) is after postprocessing with scores $IoU^{postpr} = 0.4180$, $Dice^{postpr} = 0.8360$.

to the problem type that we are addressing, that is at pixel-level, and with the type of loss we are using, namely MSE. Figure 6.3 shows examples of the difference between the segmented mask using the cutoff of quantile at 98% and at 99%.

As previously stated, we also perform the evaluation on the rare subset of the cluster in order to know if the models were able to generalize what learnt on the frequent subset. Performances are shown in Table 6.3 and are consistent with the results of the frequent subset. We decide to consider the postprocessed score maps since we proved before they improve the performances of our model. Instead, evaluating the models on the images contained in another different cluster, more specifically on the Residential areas cluster, we have results shown in Table 6.4.

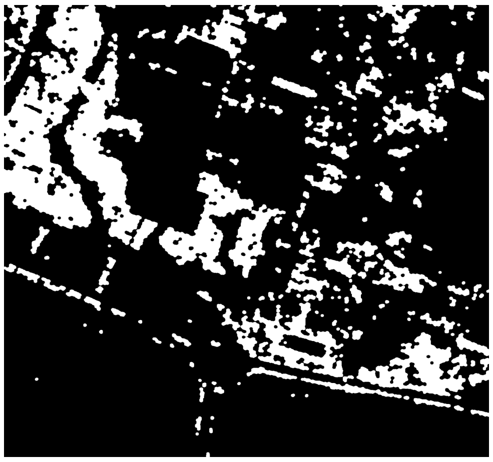
	Threshold	Postprocessing	IoU	Dice
Baseline	98%	–	0.0616	0.1232
		closing+opening	0.1158	0.2317
	99%	–	0.0628	0.1257
		closing+opening	0.1446	0.2893
$\ell_{\mathbf{MSE}}$	98%	–	0.0616	0.1233
		closing+opening	0.1181	0.2362
	99%	–	0.0629	0.1257
		closing+opening	0.1482	0.2964
$\ell_{\mathbf{mixed}}$	98%	–	0.0464	0.0972
		closing+opening	0.0986	0.1973
	99%	–	0.0452	0.0903
		closing+opening	0.0881	0.1762

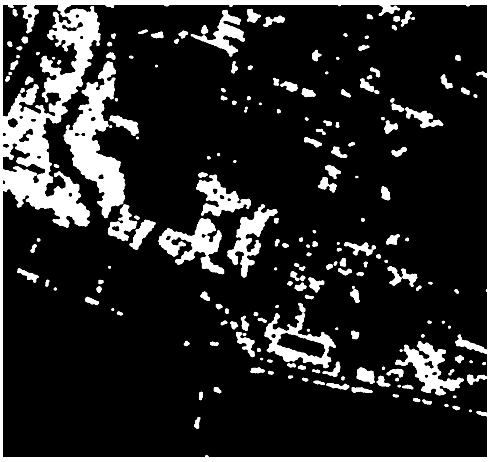
Table 6.2: Evaluation of the models trained with different losses, before and after the application of morphological postprocessing; as baseline we considered a model trained on the complete dataset. The threshold cutoff is computed on the score maps using heuristic quantile at 98% and 99%.

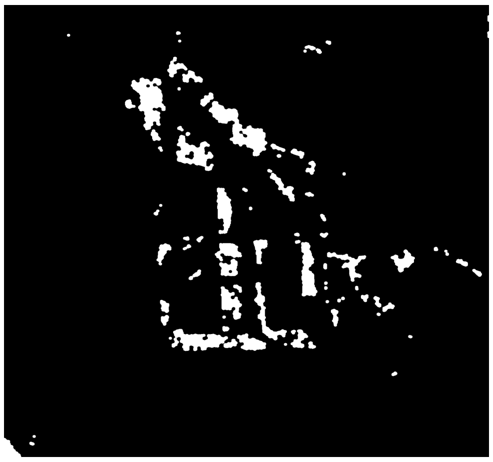
	Threshold	IoU	Dice	AUROC
${\ell_{\mathbf{MSE}}}^{\mathbf{postpr}}$	98%	0.1261	0.2523	78.01 %
	99%	0.1581	0.3161	
${\ell_{\mathbf{mixed}}}^{\mathbf{postpr}}$	98%	0.0776	0.1552	77.51 %
	99%	0.0676	0.1352	

Table 6.3: Evaluation on the rare set of cluster Agricultural areas of the models trained on frequent set with MSE loss and mixed loss.

The results in Table 6.3 denote that our models can broaden what learnt to land cover types connected and affine to those on which the models were trained, more specifically if trained on the images belonging to the frequent land cover types of the considered cluster, they can also generalize the segmentation ability to the rare land cover types. However, we also notice from Table 6.4 that the mixed loss model is not specific for Agricultural cluster, since it has similar performances also on another different cluster, more precisely on Residential areas cluster. Therefore, we perform only on the model built with MSE


(a) Quantile at 98%


(b) Quantile at 99%


(c) Quantile at 98%

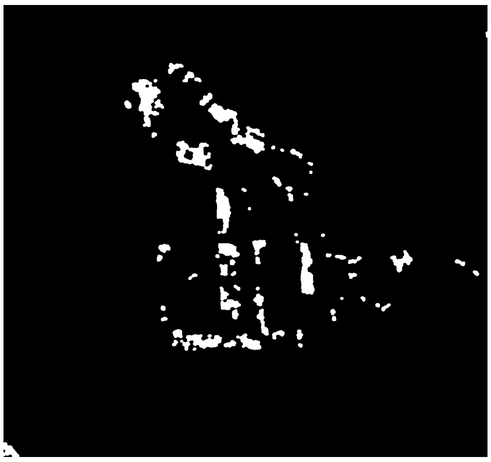

(d) Quantile at 99%

Figure 6.3: Images (a) and (d) are the GT masks, (b) is the raw mask computed through the model trained with MSE before postprocessing and has scores $IoU = 0.1509$, $Dice = 0.3017$, (c) after postprocessing has scores $IoU^{postpr} = 0.4704$, $Dice^{postpr} = 0.9408$. (e) is the raw mask computed using model trained with MSE before postprocessing and has scores $IoU = 0.1781$, $Dice = 0.3561$, (f) is after postprocessing with scores $IoU^{postpr} = 0.4180$, $Dice^{postpr} = 0.8360$.

loss the last evaluation in predicting a sufficient amount of waste in anomalous images of the frequent subset.
Computing the value minimizing the distance between Precision and Recall scores, we find the optimal threshold for the IoU score. Figure 6.4 shows the visual computation of the optimal threshold for the MSE model.

We use this threshold to discriminate between images predicted as *anomalous*, namely with a sufficient amount of waste detected, or *normal*, finding: 59 images properly pre-

	Threshold	IoU	Dice	AUROC
$\ell_{\mathbf{MSE}}{}^{\mathbf{postpr}}$	98%	0.1089	0.2179	68.94 %
	99%	0.1273	0.2547	
$\ell_{\mathbf{mixed}}{}^{\mathbf{postpr}}$	98%	0.0797	0.1595	68.33 %
	99%	0.0757	0.1515	

Table 6.4: Models trained on cluster of Agricultural areas evaluated on cluster of Residential areas.

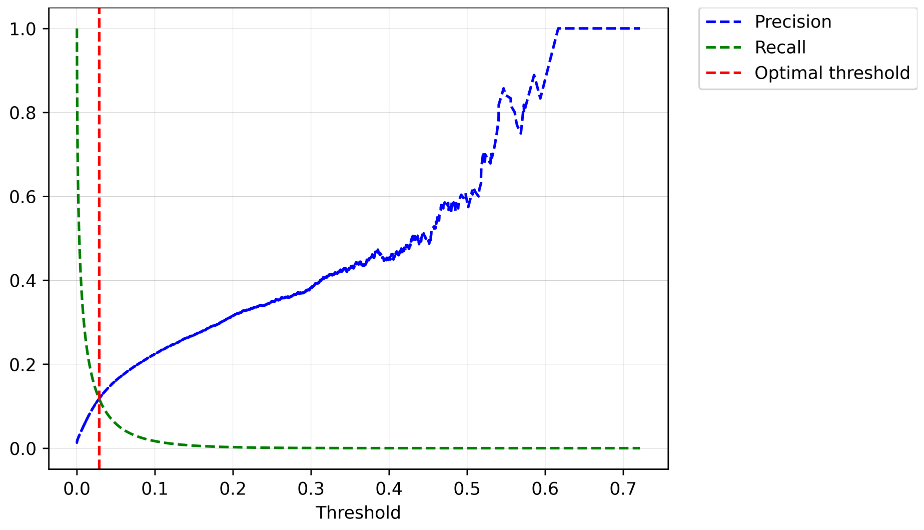


Figure 6.4: Precision and recall curves used to find the optimal threshold for the IoU score, for MSE model. The threshold found is $\tau_{MSE} = 0.028445369$

dicted as anomalous (True Positive), 13 incorrectly predicted (simultaneously False Negative and False Positive, since the masks were misplaced). Performing the same evaluation considering only normal images we find 62 True Negative and 440 False Positive.
Hence, we can complete the testing computing

$$Recall = 0.952 \quad \text{and} \quad Precision = 0.134.$$

Since in the real-world scenario the most important thing is not to miss any illegal landfill, the major result is that almost all the anomalies are correctly detected, as enhanced by the high Recall value.

7 | Conclusion and future developments

In this work, we approached the illegal landfill detection problem from a different perspective with respect to the previous works. Indeed, we tackled the task as an unsupervised anomaly detection problem, while other researchers use a supervised classification approach. Addressing the problem in an unsupervised pixel-level manner is less effective with respect to supervised image-level approaches, but it gives us the benefit of using the same network with new different images, without retraining it. Indeed, we can receive in input new images to be analysed without needing prior information about the anomalies to be detected, and we do not have to retrain the model so that it is effective on new data. In a real-world scenario such as illegal landfills detection, it is important to be flexible, since the satellite image dataset could be in continuous update. Hence, it is essential to have the ability to receive in input any new images, without distinction on the type of waste contained in them.

Clustering the land cover types slightly improved the performances of our models, even if we notice that the edges of objects in the image are likely to raise false alarms. This issue is clearly due to the complexity of the addressed task, which is at pixel-level, and to the use of MSE loss to train our model. Indeed, as explained in Section 3.4, MSE loss has problems detecting the contours of objects in images. For this reason, we created a new loss that is able to catch the structure of the image as the SSIM, and, at the same time, it is also precise at pixel-level as the MSE. This loss is obtained as a weighted sum of SSIM and MSE. However, the mixed loss turned out to under-perform with respect to the classical MSE loss. As a future approach, it could be developed a mixed loss using a different type of structural similarity or a different metric.

Overall, we notice that the results also depend on the quality of the ground truth masks. Indeed, model predictions are pixel-sensitive, and sometimes their assessment is affected by the quality of the masks: the more precise the mask is around the edges of the waste, the more accurate the model prediction results.

As explained, other researchers use different methods with respect to us, but we have shown that our unsupervised approach is feasible and viable, bringing reasonable results. We are among the first to tackle the illegal landfills identification problem as an unsupervised anomaly detection problem, offering an alternative perspective to the existing studies. Indeed, clustering the different land cover types allows us to have distinct, specific models for each land cover, meaning that we can receive a new dataset of images with whichever land cover in it, and we will have the correct model to analyse them.

We underline that in our context it is more important to have a high recall with respect to a high precision, meaning that false alarms of illegal landfills could be risen, but almost all of the illegal areas are detected: sending someone to investigate a suspicious area is better than not spotting it.

Bibliography

[1] ARPA Lombardia. `https://www.arpalombardia.it`, updated 2023.

[2] F. Ahmed and A. Courville. Detecting semantic anomalies. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:3154–3162, 04 2020. doi: 10.1609/aaai.v34i04.5712.

[3] S. Albawi, T. A. Mohammed, and S. Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6, 2017. doi: 10.1109/ICEngTechnol.2017.8308186.

[4] D. Bank, N. Koenigstein, and R. Giryes. Autoencoders, 2021.

[5] P. Bergmann, S. Löwe, M. Fauser, D. Sattlegger, and C. Steger. Improving unsupervised defect segmentation by applying structural similarity to autoencoders. In *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. SCITEPRESS - Science and Technology Publications, 2019.

[6] P. Bergmann, K. Batzner, M. Fauser, D. Sattlegger, and C. Steger. The mvtec anomaly detection dataset: A comprehensive real-world dataset for unsupervised anomaly detection. *International Journal of Computer Vision*, 2021.

[7] S. Bhutada, N. Yashwanth, P. Dheeraj, and K. Shekar. Opening and closing in morphological image processing. *World Journal of Advanced Research and Reviews*, 06 2022.

[8] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10), oct 2008. doi: 10.1088/1742-5468/2008/10/p10008.

[9] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3), 2009. DOI = 10.1145/1541880.1541882.

[10] V. Dumoulin and F. Visin. A guide to convolution arithmetic for deep learning, 2018.

[11] European Commission - CORDIS. PERIVALLON Protecting the EuRopean terrItory from organised enVironmentAl crime through inteLLigent threat detectiON tools. `https://cordis.europa.eu/project/id/101073952`, 2022-2025.

[12] Geoportale della regione Lombardia. Uso e copertura del suolo 2021 (dusaf 7.0). `https://www.geoportale.regione.lombardia.it`, updated 2023.

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

[14] R. M. Haralick, S. R. Sternberg, and X. Zhuang. Image analysis using mathematical morphology. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(4):532–550, 1987. doi: 10.1109/TPAMI.1987.4767941.

[15] Y. Lecun and F. Soulie Fogelman. Modeles connexionnistes de l'apprentissage. *Intellectica, special issue apprentissage et machine*, 2, 01 1987. doi: 10.3406/intel.1987.1804.

[16] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989. doi: 10.1162/neco.1989.1.4.541.

[17] A. Limoli, E. Garzia, A. D. Pretto, and C. D. Muri. Illegal landfill in Italy (EU)—a multidisciplinary approach. *Environmental Forensics*, 20(1):26–38, 2019. doi: 10.1080/15275922.2019.1566291.

[18] J. Liu, G. Xie, J. Wang, S. Li, C. Wang, F. Zheng, and Y. Jin. Deep Industrial Image Anomaly Detection: A Survey. *arXiv e-prints*, Jan. 2023. doi: 10.48550/arXiv.2301.11514.

[19] K. A. Mat Said, A. Jambek, and N. Sulaiman. A study of image processing using morphological opening and closing processes. *International Journal of Control Theory and Applications*, 9:15–21, 01 2016.

[20] G. Matheron. *Random Sets and Integral Geometry [By] G. Matheron*. Wiley Series in Probability and Mathematical Statistics. Wiley, [1974, C1975], 1974.

[21] H. Minkowski. Volumen und oberfläche. *Mathematische Annalen*, 57:447–495, 1903.

[22] M. E. J. Newman. Analysis of weighted networks. *Phys. Rev. E*, 70:056131, Nov 2004. doi: 10.1103/PhysRevE.70.056131.

[23] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69:026113, Feb 2004. doi: 10.1103/PhysRevE.69.026113.

[24] G. Pang, C. Shen, L. Cao, and A. Van Den Hengel. Deep learning for anomaly detection: A review. *ACM Comput. Surv.*, 54(2), mar 2021. ISSN 0360-0300. doi: 10.1145/3439950. URL `https://doi.org/10.1145/3439950`.

[25] L. Ruff, J. R. Kauffmann, R. A. Vandermeulen, G. Montavon, W. Samek, M. Kloft, T. G. Dietterich, and K.-R. Muller. A unifying review of deep and shallow anomaly detection. *Proceedings of the IEEE*, 2021.

[26] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Learning Internal Representations by Error Propagation*, page 318–362. MIT Press, Cambridge, MA, USA, 1986. ISBN 026268053X.

[27] J. Serra. *Image Analysis and Mathematical Morphology.* Academic Press, Inc., USA, 1983. ISBN 0126372403.

[28] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000. doi: 10.1109/34.868688.

[29] A. Siddiqua, J. N. Hahladakis, and W. A. K. A. Al-Attiya. An overview of the environmental pollution and health effects associated with waste landfilling and open dumping . *Environmental Science and Pollution Research*, 2022.

[30] R. N. Torres and P. Fraternali. Aerialwaste dataset for landfill discovery in aerial and satellite images. *Scientific Data*, 10(1):63, 2023.

[31] R. N. Torres, F. Milani, and P. Fraternali. ODIN: Pluggable Meta-annotations and Metrics for the Diagnosis of Classification and Localization. In *International Conference on Machine Learning, Optimization, and Data Science*, pages 383–398. Springer, 2021.

[32] M. D. Vaverková. Landfill Impacts on the Environment—Review. *Geosciences*, 9 (10), 2019.

[33] U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 2007.

[34] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13 (4):600–612, 2004. doi: 10.1109/TIP.2003.819861.

[35] C. Zhou and R. Paffenroth. Anomaly detection with robust deep autoencoders. pages 665–674, 08 2017. doi: 10.1145/3097983.3098052.

List of Figures

List of Tables

List of Algorithms

Ringraziamenti

Vorrei innanzitutto ringraziare il Professor Giacomo Boracchi, che mi ha offerto la possibilità di prendere parte all'avventura che è stata questa tesi.
Ringrazio il Professor Piero Fraternali, grazie al quale ho avuto l'occasione di partecipare allo stimolante progetto di PERIVALLON.
Un ringraziamento a Luca Frittoli, per la sua disponibilità e per essere riuscito a dedicarmi del tempo quando avevo bisogno.

Un grazie di cuore a tutte le persone che, anche se solo di passaggio, mi hanno accompagnato in questi anni della mia vita.

Elena