POLITECNICO DI TORINO

Master's Degree in Computer Engineering

Master's Degree Thesis

Deep Learning Techniques for Water Level Forecasting in Satellite Images

Supervisors

Prof. Paolo GARZA

Dott. Daniele REGE CAMBRIN

Candidate

Alessandro CHIABODO

October 2024

Abstract

As the climate continues to change, the ability to automatically map bodies of water and predict their future variations is certainly very important, both to ensure that people are able to prevent and detect certain extreme weather events, such as floods and droughts, and to aid urban, agricultural and industrial planning around large lakes or waterways. Since the launch of the first satellites, images from orbit have been used to detect and monitor our planet's water masses, such as oceans and lakes. Over the years, increasingly accurate methods have replaced human work, and in recent years, deep learning algorithms such as Convolutional Neural Networks (CNNs) and Visual Transformers (ViTs) have achieved near-perfect performance in this task. However, although the detection of even the smallest lake from Earth orbit is now near perfect, we are still unable to predict changes in the size and shape of bodies of water with high accuracy. This thesis aims to address this issue by comparing the performance of different state-of-the-art deep learning models in both water detection and water level prediction using freely available satellite data. We not only compared different models and architectures, such as U-Nets and DeepLab for segmentation or ResNet and LeViT for forecasting, but also how varying the properties of the input data, such as the number and type of electromagnetic bands or the number of temporal observations, can change the behaviour of the models and the results obtained. The experimental results show that CNNs perform better in these tasks than previous state-of-the-art machine learning approaches, such as SVM and Random Forest, and classical index-based methods.

Acknowledgements

I would thank professor Paolo Garza and Daniele Rege Cambrin for all the feedbacks and the technical help provided during the work that gave life to this thesis. I thank my family and my friends for the help and the support they gave me during the years of my studies. I have to thank HPC@POLITO for providing computational resources to work on this project.

Table of Contents

List of Tables

List of Figures

Acronyms

CV
Computer Vision

CNN
Convolutional Neural Network

DL
Deep Learning

IoU
Intersection over Union

MAE
Mean Absolute Error

ML
Machine Learning

MLP
Multilayer Perceptron

MSI
Multispectral Imagery

NDWI
Normalized Difference Water Index

NIR
Near-Infrared

NN

Neural Network

R^2

R-squared (Coefficient of Determination)

RGB

Red Green Blue

SAR

Synthetic Aperture Radar

SVM

Support Vector Machine

SWIR

Short-Wave Infrared

Chapter 1

Introduction

In this chapter, we provide an overview of the research problem, the motivation behind this research, and the structure of the thesis.

1.1 Water Mapping in Satellite Images

The detection of water bodies in satellite images is a fundamental task in remote sensing [1] [2], with numerous applications in environmental monitoring [3], agriculture, and urban planning. Since the early days of remote sensing [4], researchers have been searching for reliable methods to detect even the smallest water bodies from satellite images.

The first methods for automatic water mapping were based on the calculation of indices, such as the Normalized Difference Water Index (NDWI) [4], which is based on the difference in reflectance between the near-infrared and green bands. Before the explosion of machine learning-based techniques, much of the extraction of water features from satellite imagery was done using indices calculated by mathematical ratios based on the particular reflectivity of certain materials at certain wavelengths of the electromagnetic spectrum, but in many cases important details were missing.

The first machine learning algorithms revolutionised the field of remote sensing [5], enabling the development of more accurate and efficient methods for detecting water in satellite imagery. Some of the most widely used ML models for surface water mapping have been SVM, Decision Trees, Random Forests and K-Means Clustering [5] with good results across the board.

Yet, with modern deep learning techniques [6, 7], the field could see a further leap in performance, with the development of models capable of exploiting the full potential of multispectral satellite imagery for water detection.

1.2 Water Level Forecasting

While lake water levels are naturally subject to interannual variability, in the last two decades climate change and human activities have played an increasingly important role in water level fluctuations.[8] Monitoring these changes has become an increasingly important aspect of urban planning [9], commercial shipping, hydropower [10], and agricultural planning [11], as well as climate change monitoring [9] in recent decades.

Real-time monitoring and forecasting of these environmental problems can be used to take immediate action to reduce the damage caused by, for example, droughts or floods. Since the launch of the first satellites equipped with optical sensors, research has been conducted on how to use this vast source of data to analyse the evolution of lakes and rivers over time. Some of the first machine learning-based methods used to predict water levels were based on feed-forward networks [12], MLP, autoregressive networks and SVMs[10].

Following the outstanding advances in deep learning in recent years, many research initiatives have turned their attention to the study of multispectral satellite imagery for the purpose of monitoring and assessing changes in water bodies. Deep Learning can be employed to process data from satellites and sensors to monitor some of the major environmental changes [9] of our time, such as deforestation, melting of perennial ice, rising sea levels and water level fluctuations in lakes and rivers. However, there is still considerable room for further research [13], especially with regard to the interannual prediction of such changes.

Given the recent advancements in deep learning and the critical importance of monitoring and predicting water levels for human life, this thesis will explore how various deep learning models, particularly Convolutional Neural Networks (CNNs) and Vision Transformers (ViTs), can predict annual changes in some of the world's major lakes.

Unlike similar studies, our focus will be on short-term forecasting of water level fluctuations in inland water bodies.

1.3 Thesis Structure

The remainder of this thesis is structured into chapters, where each chapter is dedicated to a certain topic:

- **Chapter 2** provides an overview of related work in remote sensing, satellite imagery, machine learning, and deep learning, with a particular focus on the methods and techniques used for water detection and water level prediction.

- **Chapter 3** delineates the methodologies employed in this research, including

data collection, data preprocessing, and the models utilized for water detection and water level prediction.

- **Chapter 4** presents the findings of the experiments conducted in this research, including an evaluation of the models and a comparison of the results.
- **Chapter 5** is a summary of the principal findings and contributions of the thesis, together with suggestions for future works.

Chapter 2

Related Works

Here we provide an overview of the research conducted in the field of remote sensing, satellite imagery, machine learning, and deep learning, with a particular focus on the methods and techniques used for water detection and water level prediction.

2.1 Computer Vision

Computer Vision (CV) is a rapidly developing area of Artificial Intelligence (AI) that enables computers to process visual data from the real world[14]. With applications in diverse fields such as agriculture[15, 16], healthcare [17, 18], climate monitoring [9], transportation [19, 20], and manufacturing [21], CV is playing an increasingly critical role in both scientific research and industry [14]. Modern computer vision research have led to the development of novel algorithms and architectures for complex tasks such as facial recognition [22, 23], autonomous driving [20], medical image analysis [17], and even climate-related monitoring [24, 14]. The idea at the base of CV is to enable computers to "see" and understand the visual world [14] around them by extracting meaningful information from images captured by a variety of sensors, such as cameras, satellites, or medical imaging devices. Some of the most common tasks in computer vision are:

- Image Classification: assigning a label to an image based on its content [25, 18]. This is one of the main tasks in computer vision and is widely used in fields like medical diagnostics [18] and large-scale image retrieval [25].
- Image Regression: focuses on predicting continuous values from an image[26]. For example estimating the age of a person from a photo, or the price of a house from a picture.
- Object Detection: based on the detection and localization of objects in an image [27].

- Semantic Segmentation: this task aims to classify each pixel in an image, providing a dense prediction where every pixel is labeled with a class. Can be seen as a pixel-wise classification task [19, 28, 17]. Semantic segmentation is essential for tasks that require a detailed understanding of the scene, such as autonomous driving and land use monitoring.

- Instance Segmentation: similar to semantic segmentation, instance segmentation labels pixels but also distinguishes between different instances of the same class. For example, in a picture with two cars, it will assign a different label to each car [29].

- Panoptic Segmentation: combines the tasks of semantic and instance segmentation, providing a comprehensive understanding of both the objects presents and the whole scene [30].

Other important tasks in computer vision, but less relevant for the purpose of this thesis, are image captioning [31],which generates descriptive text for an image, pose estimation [32], image (and video) generation [33, 34],which involves creating new visual data based on learned patterns, and De-Noising [35, 36], which removes noise or distortions from images.

In this thesis, we will primarily focus on image classification, image regression and semantic segmentation, as they are directly relevant to the goal of water detection and water level forecasting. Image classification will help identify the presence of water, while image regression will be used to estimate water levels based on visual data. Finally, semantic segmentation will allow us to perform pixel-wise classification of water bodies, providing detailed and accurate water delineation.

2.1.1 Deep Learning

Deep learning (DL) is a subset of machine learning (ML) that uses multi-layer ("deep") neural networks to model and extract patterns from data, in some ways mimicking the decision-making of the human brain. [37]

A so-called deep neural network is nothing more than a stack of several simple computational units [37, 24] capable of learning patterns from data, many of which are also non-linear functions. With numerous non-linear layers, a network is then able to learn extremely complex functions [38] while remaining sensitive to small changes [17] in the data and immune to irrelevant variations. The resulting model can then be trained on a large dataset using a process called backpropagation [39], which applies simple Stochastic Gradient Descent (SGD) to the output [40]. The weights of the network, which are the units responsible for calculating the output, are updated at each step to minimise the error between the expected output and the actual predictions.

(a) Image classification [25].

(b) Image Generation [33].

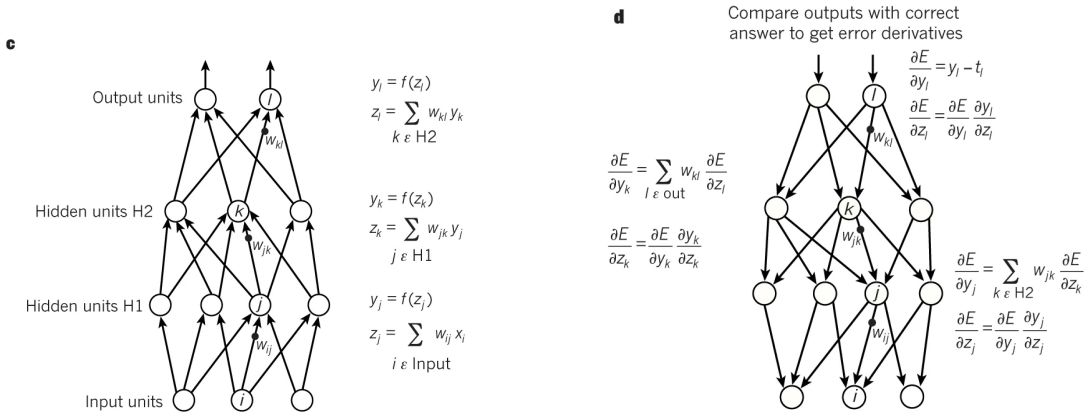
(c) Instance Segmentation [14].

(d) Semantic Segmentation [19].

Figure 2.1: Tasks in Computer Vision

One of the key differences between traditional ML and DL is the ability to extract important features directly from the data, often referred as representation learning [37] without the need for manual feature extraction as in traditional ML algorithms. This is particularly useful in computer vision, where the features to be extracted are often complex and difficult to define manually[14].

The most common types of deep neural networks used in computer vision are convolutional neural networks (CNNs)[24, 37, 14]. These are a particular type of feed-forward neural network that are much easier to train than traditional neural networks, as they are able to learn the spatial hierarchies of features in the data[37].



(a) Multilayer Neural Network **(b)** Backpropagation algorithm

Figure 2.2: Multilayer neural networks and backpropagation. From [37]

2.1.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a type of feed-forward neural network [41], and the most widely used DL network [24], designed to process and analyse data, such as images, that come in the form of multiple stacked arrays [37]. CNNs are best known for their effectiveness in image-related tasks such as image classification[38], object detection[27], and semantic segmentation [24]. This is why they are widely used in fields such as facial recognition, anomaly detection [42], medical diagnostic [18, 17] and autonomous vehicles [20]. They can also be applied to diverse tasks such as time series forecasting[43] and natural language processing, although in recent years Transformers models have surpassed [44] and replaced all alternatives in this area. At last but not least, CNNs are also used in remote sensing for land cover classification, object and change detection [9]. Although precursors of CNNs date back to the 1980s, the modern CNN architecture was first introduced by LeCun et al. in 1998 [45]. A further leap forward was made in 2012, when AlexNet [38] won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC)[25] by a large margin, demonstrating the effectiveness of CNNs in image classification tasks. From that moment on, also thanks to the increasing computational power of modern computers, CNNs became the de-facto standard architecture for image classification.

The architecture of a CNN consists of a series of stages as in Fig.2.3, each of which is designed to perform different operations on the input data [41, 37]. The first stages are mainly composed of two types of layes, convolutional and pooling layers.

The convolutional layer applies a set of trainable filters (or kernels) to the input

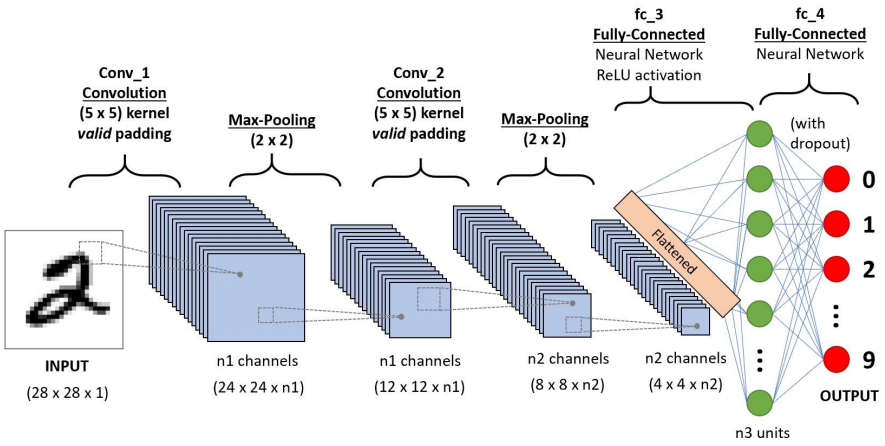


Figure 2.3: How a CNN works [41]

image[41]. Each filter is a matrix that 'slides' over the entire image at a predefined step (called a stride). At each position, a scalar product is calculated between the filter and the corresponding portion of the input image. This process generates a feature map for each filter, representing the filter's response to different areas of the image. The resulting feature maps can then be stacked and become the input for the next layer of the network. In this way, the filters can 'learn' different patterns in the input image[37].

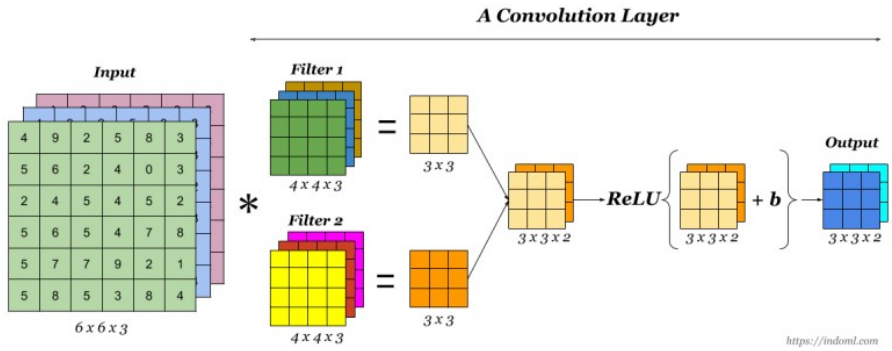


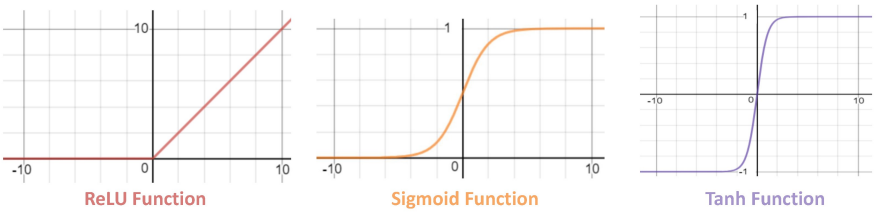
Figure 2.4: A diagram of a convolutional layer. The filter (or kernel) slides over the input image, generating a feature map. Image taken from [46]

The activation function applies a non-linear transformation to the output of the convolutional layer, allowing the network to learn complex patterns in the data. Usually the ReLU function is used, but other functions such as Sigmoid or Tanh can be used. We can see some of the most common activation functions in Fig.2.5a.

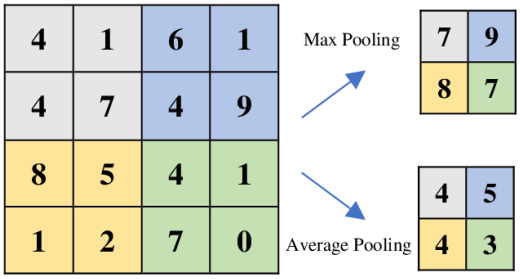
A pooling layer, as in Fig. 2.5b is often applied after a convolutional layer to

reduce the spatial dimensions of the data by downsampling. This reduces the number of parameters and computational load in the network while helping to control overfitting.

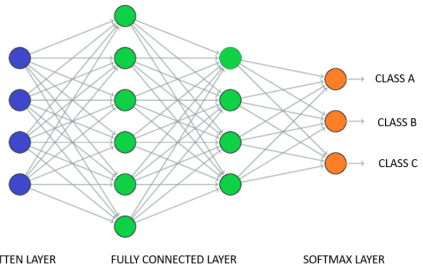
Lastly a fully connected Layer forms the output layer,as shown in Fig. 2.5c, connecting each neuron in the previous layer to every neuron in the output layer, thus allowing, for example, to assign the correct class to the data based on the output of the network. In the following, we will introduce some of the CNN



(a) Activation functions used in CNNs [24]



(b) Pooling functions in CNNs [47]



(c) Fully connected layer [24]

Figure 2.5: All the layers used in a CNN

architectures used in remote sensing and image processing, in particular those that we will use and compare in the course of our research.

ResNet Ultra-deep CNN architecture that introduced the concept of residual connection [48]. ResNet works by adding the input of a layer to the output of the layer itself, allowing the network to learn the residual function, which is the difference between the input and the output of the layer. This architecture allows the training of very deep networks, up to 152 layers, by solving the problem of vanishing gradients. This issue occurs when the gradient of the loss function approaches zero, making the network unable to learn. Residual connections mitigate this by allowing the gradient to skip certain layers of the network [49], thus maintaining the flow of gradients and facilitating the training of deeper networks. ResNet has been widely adopted in many computer vision tasks, including image classification [50], object detection [48] and semantic segmentation [24] and has been the basis for many other architectures [51].

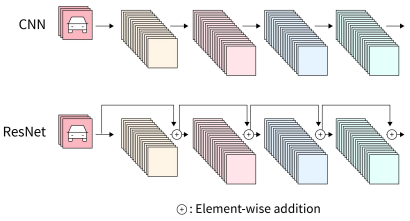
MobileNet Designed to be extremely efficient models for mobile and embedded vision use-cases, MobileNets are a series of lightweight CNNs [28] architectures that are designed to be fast and efficient on devices with limited computational resources. Each different MobileNet "generation" introduces new features and improvements. There are currently four versions of MobileNets, each with different architectures and performance characteristics. MobileNets V1 [52] are based on a depth-wise separable convolution structure to ensure a good trade-off between speed and performance. MobileNets V2 [53] introduced the inverted residual structure, while MobileNets V3 [54] introduced the use of an attention mechanism to further improve the performance of the network. Lastly, MobileNets V4 [55] introduces the so-called Universal Inverted Bottleneck (UIB) search block, an advance structure that allows to merge many of the modern CNNs improvements, like Inverted Bottleneck, ConvNext and Extra DepthWise into a single block. This last network is able to achieve near state-of-the-art performance, while being extremely efficient and able to run on mobile platform. For the scope of this research, we will focus on MobileNet V4 as a lightweight model for water level forecasting and MobileNet V3 for segmentation.

VGG Based on a simple and efficient architecture, the VGG network [56] consists of 16 convolutional layers and 3 fully connected layers, with a total of 138 million parameters. The use of small convolutional filters (3x3), rather than larger (5x5) and (7x7) kernels as in AlexNet, and the stacking of multiple layers [24] allows the network to learn complex patterns in the data, making it particularly effective in image classification tasks. Its main drawback is the high computational cost due to the large number of parameters, almost 140 million, which makes it difficult to train compared to smaller models.

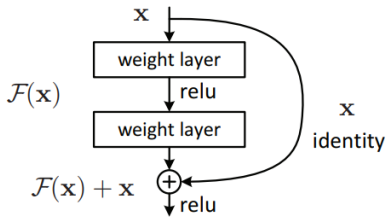
ConvNext A modern Convolutional Neural Network (CNN) architecture designed to compete with Vision Transformers (ViT) while maintaining the simplicity of traditional CNNs [51]. Developed by gradually modernizing the classic ResNet [48] architecture by incorporating some key design choices from ViT, such as larger kernel sizes and inverted bottleneck layers, ConvNeXt outperforms many ViT models in tasks such as image classification, object detection, and semantic segmentation, while retaining the efficiency and scalability of standard ConvNets. One of the key innovations in ConvNeXt is the use of large convolutional kernels (up to 7x7), allowing for a larger receptive field[57].

2.1.3 Semantic Segmentation

Semantic segmentation is the task [28] of computer vision that consists of assigning a class label to each pixel in an image, in other words, it is a pixel-wise classification



(a) Comparison between a traditional CNN and a ResNet architecture



(b) Diagram of a residual block [48]

Figure 2.6: Convolutional Neural Networks

task. This means that each pixel in the input image is classified into a specific category, which could be a person, a car, a tree, a lake, etc., based only on the input data.

The main goal of semantic segmentation is to achieve a comprehensive and accurate understanding of the entire content within an image. This task is crucial for various applications, including autonomous driving [20], medical image analysis [17], and scene understanding in remote sensing [58].

Deep learning models, in particular convolutional neural networks (CNNs), are commonly used to perform semantic segmentation[28]. Advanced architectures such as Fully Convolutional Networks (FCNs), U-Net and DeepLab have been developed to improve the accuracy and efficiency of semantic segmentation. In the following, we will introduce some of the most widely used fully CNN architectures for semantic segmentation, focusing on those that we will use and compare in the course of the thesis.

2.1.4 CNNs for Semantic Segmentation

Due to the specific nature of the semantic segmentation task, traditional CNN architectures are not well suited for this task, as it's essential to maintain detailed spatial information throughout the network in order to accurately differentiate between object boundaries. However, simply maintaining the same input dimensional across the entire network would be computationally infeasible, especially with high-resolution images. To address this challenge, an encoder-decoder architecture is often used, as illustrated in Fig 2.7. In this particular architecture, the encoder role is to extract important features from the input image through a series of convolutional and pooling layers, reducing the spatial resolution but increasing the granularity of informations. The decoder gradually reconstructs the spatial resolution of the image by using upsampling techniques such as transposed convolutions (also called deconvolutions).

In this way, the encoder can extract meaningful feature representations from the input image, while the decoder uses this information to reconstruct the spatial details of the entire image. Several CNN architectures have been designed specifically to address the unique requirements of semantic segmentation tasks, with some of the most widely used being U-Net[17], DeepLab[59] and PSPNet[60]. These specialized architectures demonstrate the evolution of CNNs for semantic segmentation, enabling more accurate and computationally efficient pixel-wise classification by addressing the spatial preservation challenge inherent in traditional CNNs.

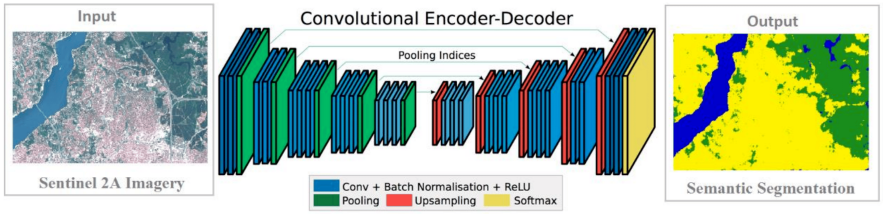


Figure 2.7: Encoder-decoder architecture in semantic segmentation. From [61]

U-Net

Originally developed for biomedical image segmentation, U-Net[17] is a fully convolutional network based on the idea of skip connections between the encoder and decoder layers. The architecture of U-Net, like almost all fully CNNs, is divided into two parts, the contraction path, which is responsible for extracting features from the input image, and the expansion path, which is used to reconstruct the spatial information of the image. The main feature introduced by U-Net are the skip connections, which allow the network to use the features extracted by the encoder to reconstruct the spatial information of the image directly in the decoder, creating a bridge between each layer of the encoder and the decoder, as in 2.8. This particular architecture allows the network to learn both the global and local features of the image, making it particularly effective in semantic segmentation tasks [28]. Even if developed for biomedical image segmentation, U-Net has been widely adopted in almost all semantic segmentation tasks, including remote sensing, where it has been used for land cover classification, interative image denoising [62], and change detection [63].

DeepLab

Semantic segmentation architecture based on a special type of upsampling filter, called atreous convolution[59], which allows control of the resolution at which

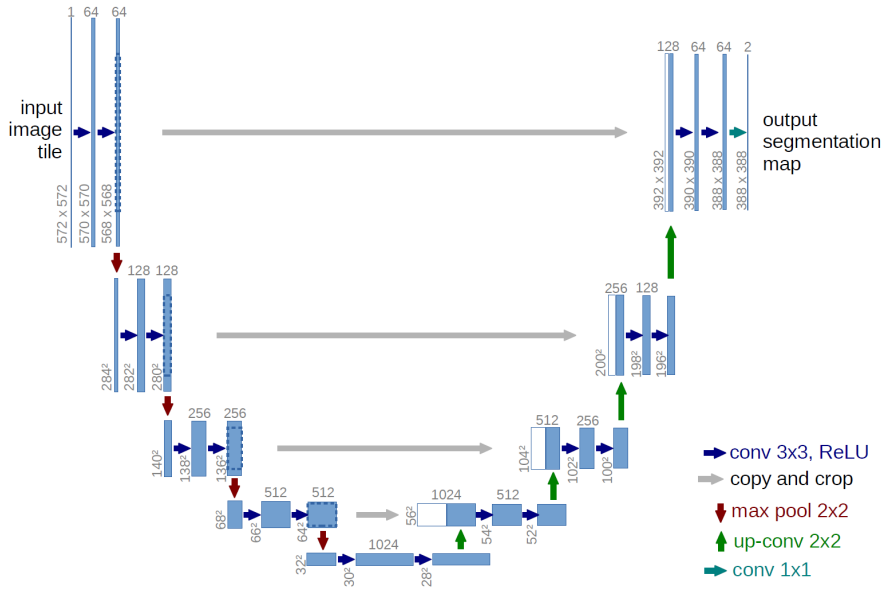


Figure 2.8: Schematic diagram of the Unet architecture. From [17].

features are computed. As a result, more input context can be handled without increasing the number of network parameters. In addition, DeepLab uses a feature called Atreus Spatial Pyramid Pooling (ASPP), which applies multiple parallel atreus convolutions at different rates. This technique captures multi-scale features by applying filters with different receptive fields.

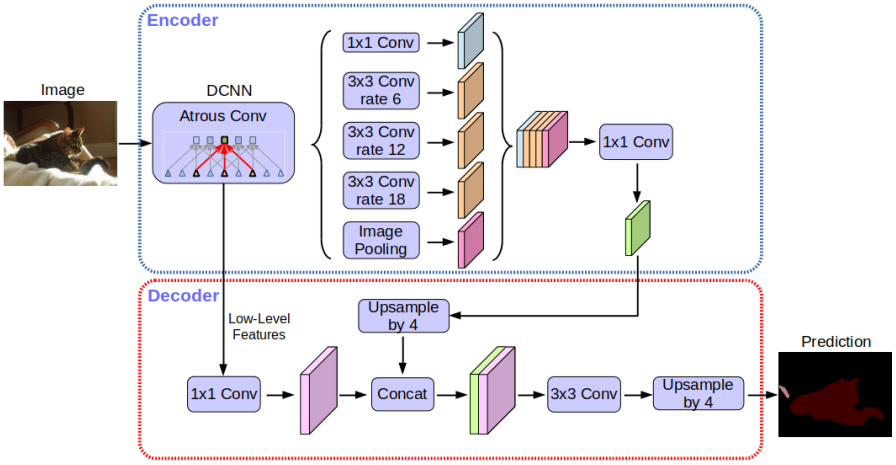


Figure 2.9: DeepLab architecture. From [59]

PSPNet

Based on a piramide structure, as the name might imply, the Pyramid Scene Parsing Network (PSPNet) [60] is a semantic segmentation architecture that uses a pyramid pooling module to capture global context information from the input image. The pyramid pooling module divides the input image into different regions and applies average pooling to each region, capturing the global context information at different scales. In addition to the pyramid pooling module, PSPNet also employs a deep convolutional neural network (CNN) as its backbone, which is responsible for extracting high-level features from the input image.

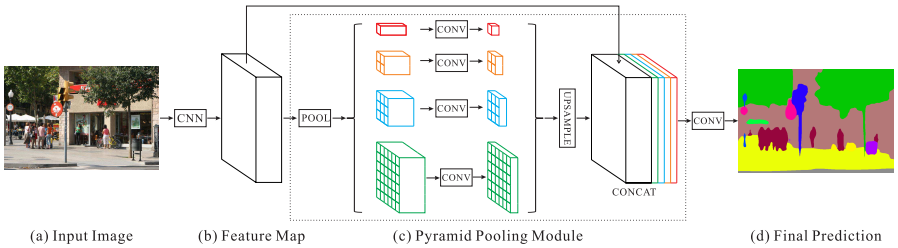


Figure 2.10: PSPNet architecture. From [60]

2.1.5 Vision Transformers

Transformers, an innovative attention-based architecture introduced by Vaswani et al. in 2017[64], has become the de facto standard architecture in Natural Language Processing (NLP) [44], training on large datasets and achieving state-of-the-art results in many tasks. The efficiency and scalability of transformers has allowed the training of models of unprecedented size, with a total of over 100B parameters, such as the new llama 3 model developed and released as open source by Meta AI, which has a total of 405B parameters [65, 66]. Some of the most widely used transformer-based models are BERT, T5 and GPT-3, with the latter being known for generating coherent and fluent text[44].

In recent years, the same architecture has been adopted in computer vision, giving birth to the Vision Transformers (ViT) [67], a pure transformer-based architecture that can be used for almost all computer vision tasks. Unlike Convolutional Neural Networks (CNNs), which dominated the field of computer vision for years, ViTs rely purely on attention mechanisms to process images[64]. Despite the fact that training these transformer-based models requires more computational power than traditional Convolutional Neural Networks (CNNs), they have shown that convolutions are not the only effective way to process images. Visual transformers have excelled in various computer vision tasks, achieving state-of-the-art results in areas such as image classification, object detection, and semantic segmentation[68].

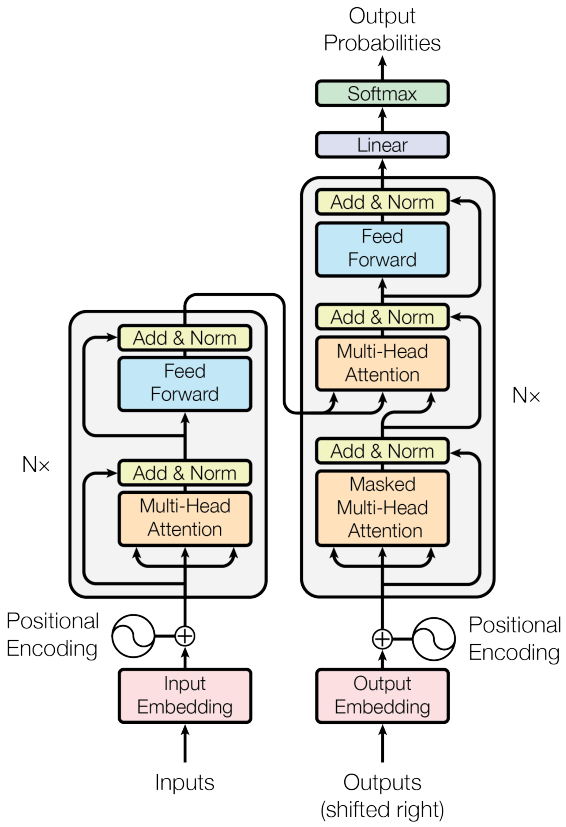


Figure 2.11: Self-attention mechanism used in Transformers. From [64]

One of the main drawbacks of this architecture, as reported in [67], is the absence of some inductive biases that CNNs have, such as translation invariance, which can make the training process more difficult and the model less robust to small changes in the input data. For this reason, the ViT architecture needs to be trained or pre-trained on large datasets to learn the features of the input data, and is not always the best choice for small datasets or when the data is not well balanced[67].

Following the first ViT model, many other architectures have been proposed, such as LeViT[69], Swin Transformer[70] and MaxViT[71], each with its own improvements. The Swin Transformer, for example, adopts a hierarchical input structure, which allows it to process images at different scales with greater flexibility, and its design ensures linear computational complexity, making it more efficient for large-scale vision tasks. On the other hand, many variants propose a hybrid structure [68] that aims to fuse convolutions and self-attention to achieve an optimal cost/performance ratio. One example is LeViT.

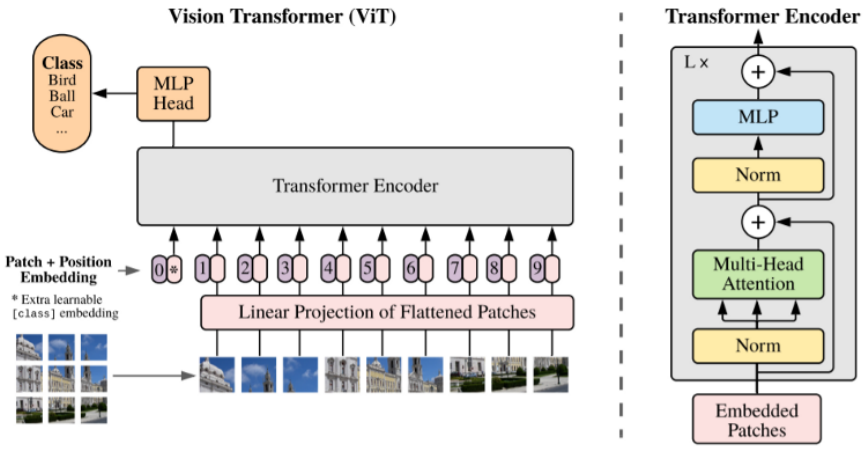


Figure 2.12: Vision Transformers architecture. From [67]

2.2 Remote Sensing

Remote sensing is the science of obtaining information [72] about the Earth's surface from images taken from a distance by optical or electromagnetic sensors mounted on satellites, aircraft, or drones. Using satellite technologies to gather critical information can help address a wide range of environmental[3], agricultural[73], and geopolitical challenges [74], offering insights that would otherwise be impossible to obtain at the same scale and frequency. In recent decades, the availability of global remote sensing has led to major changes in the monitoring and understanding of the Earth's climate, environment, and the impact of human activities on the planet [3]. For example, remote sensing has provided climate scientists with continuous, long-term datasets for the monitoring of deforestation, desertification, melting glaciers, and rising sea levels, all of which are critical indicators of climate change[9]. Among the most common applications of remote sensing are vegetation classification [75], land use [76], and water body monitoring [13], such as lakes, rivers, and wetlands, which are essential to sustaining human life and the environment. Remote sensing provides an efficient way to map the extent of water bodies [72], track the area occupied by water features, and monitor changes in water bodies over time. Even more in recent years, where there is an increased reliance on satellite imagery to assess the impact of humans and climate change [77] on the health of major water bodies. Many studies [13, 6, 78, 7] have been focused on the development of methods and algorithms to detect and monitor water bodies over time using remote sensing data. These approaches often leverage machine learning and deep learning techniques [78, 7], which have shown promise in improving the accuracy and scalability of water body detection. However, the problem remains far from

being fully solved, as challenges such as varying water reflectance, cloud cover, and the dynamic nature of water bodies continue to complicate the detection process.

2.2.1 Satellite Imagery and light spectrum

Satellite imagery is a critical data source for remote sensing applications [5, 79], providing a wealth of information about the Earth's surface across a wide range of spatial and spectral dimensions. The ability to capture vast and varied terrain from space allows for in-depth environmental monitoring and analysis at regional and global scales[3]. These images are taken by multiple sensors on different satellites orbiting the Earth, each with different characteristics and capabilities[72, 80]. Different satellite sensors are tailored to collect data at varying resolutions, spectral ranges, and temporal frequencies, providing flexibility in observing diverse Earth phenomena. Each image consists of several spectral bands, with each band representing the intensity of reflected light at a particular wavelength[81]. These spectral bands represent the intensity of reflected radiation at particular wavelengths and provide crucial informations [4] about different surface features, such as vegetation, water, soil and urban areas, allowing valuable information about the environment to be extracted[72]. Some of the most common bands used in remote sensing includes the Red, Green, Blue (RGB) band, which correspond to the visible part of the spectrum and provide true-color images, as in Fig.(2.13).

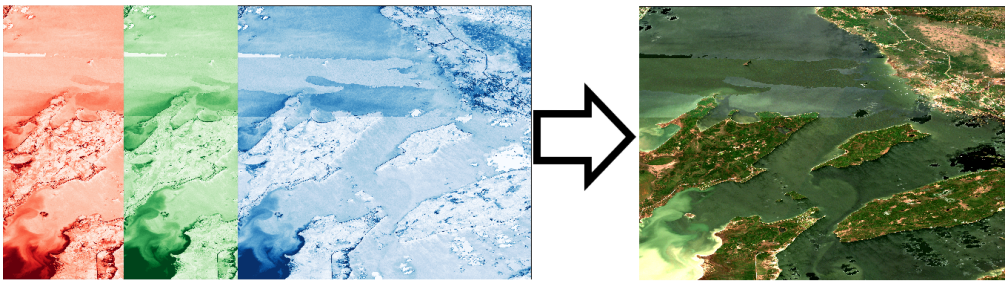
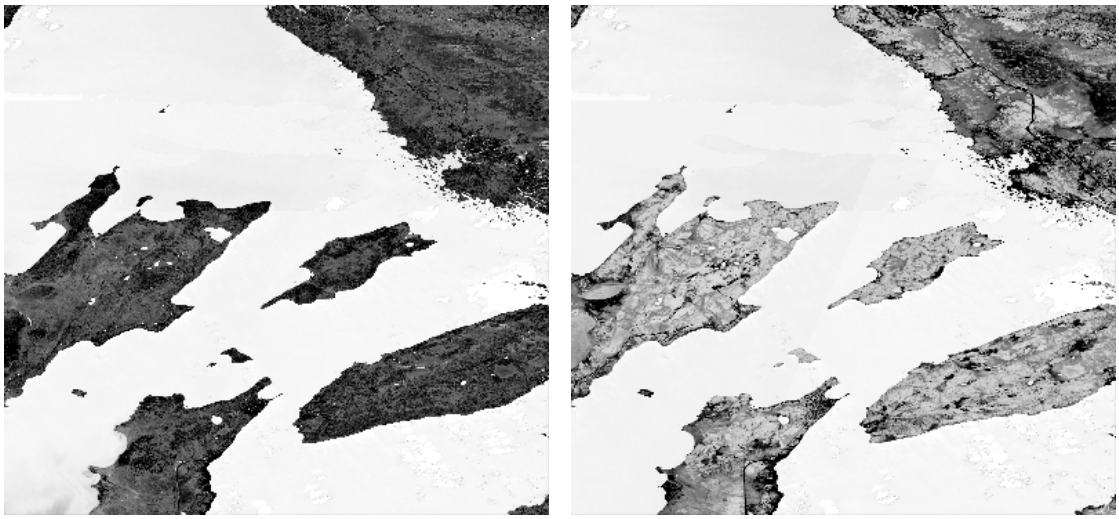


Figure 2.13: Construction of an RGB image of lake Winnipeg from the Red, Green and Blue bands

Bands in the Near Infrared (NIR) region are often used to assess vegetation health and water content in plants[4], while the Short Wave Infrared (SWIR) bands are particularly useful for identifying soil moisture and detecting wildfires[82]. This multi-spectral data allows detailed analysis and interpretation of various surface features and conditions, improving our understanding of environmental change, urban development and natural resource management[9]. In this thesis, we will focus on the Sentinel-2 mission, an Earth observation program operated by the European Space Agency (ESA). Sentinel-2 is equipped with a MultiSpectral

Instrument (MSI) that captures images in 13 spectral bands[83, 84]. These bands span from the visible to the shortwave infrared region, offering a wide spectral range that enables the analysis of diverse land and water features. The resolution of these bands ranges from 10 to 60 meters, providing a rich source of data for studying land cover, water bodies, and environmental changes.



(a) NIR band **(b)** SWIR band

Figure 2.14: Sentinel-2 spectral bands images of lake Winnipeg. Note the difference in features and details between the two bands

2.2.2 Water Indexes

Water indices obtained by combining different bands have been the most widely used method [5] for the detection of water bodies for many years. Their main advantage over single-band approaches is their ability to normalize data, reducing the influence of unwanted factors such as clouds, shadows, or variations in surface reflectance. This normalization makes the detection of water more robust, especially in complex scenes where environmental noise could otherwise interfere with the classification. Over the years, several indices have been introduced for water detection, but among them the most commonly used are the Normalised Difference Vegetation Index (NDVI), originally introduced for vegetation classification [85], and the Normalised Difference Water Index (NDWI) [4].

These indices are based on the normalized difference between two spectral bands, generally one in the visible spectrum and the other in the infrared region. For example, NDVI is computed using the red and NIR bands, taking advantage of the fact that vegetation strongly reflects NIR radiation while absorbing red light.

Similarly, NDWI uses the green and NIR bands, using the fact that water bodies reflect more green light and absorb NIR radiation, allowing for clear separation of water and non-water pixels. However, although widely used, the main limitation of these indices is that their performance can vary depending on the specific scene, the type of sensor used, and environmental conditions such as atmospheric interference, sun angle, or the presence of other reflective surfaces like snow, urban structures, or barren land. Some of the most widely used indices[5], and their equations, are:

- Normalized Difference Vegetation Index (NDVI):

$$NDVI = \frac{NIR - RED}{NIR + RED} \tag{2.1}$$

  NDVI is widely used to analyze vegetation health, as it highlights areas with live green vegetation by measuring the difference in reflectance between near-infrared (NIR) and red light. However, its use for water detection is limited, as it can confuse open water with certain types of vegetation.

- Normalized Difference Water Index (NDWI):

$$NDWI = \frac{GREEN - NIR}{GREEN + NIR} \tag{2.2}$$

  NDWI, introduced by McFeeters in 1996 [4], is designed specifically for water detection. It is based on the contrast between the green and near-infrared (NIR) bands, where water absorbs NIR light but reflects green light, making it easier to distinguish water bodies from surrounding land.

- Modified Normalized Difference Water Index (MNDWI):

$$MNDWI = \frac{GREEN - SWIR}{GREEN + SWIR} \tag{2.3}$$

  MNDWI is a variation of NDWI, which replaces the NIR band with the Shortwave Infrared (SWIR) band. This adjustment helps improve water detection in more challenging or noisy environments, particularly in areas with high reflectance, such as urban regions [86].

- Enhanced water index (EWI):

$$EWI = \frac{GREEN - SWIR + m}{(GREEN + SWIR)x(NDVI + n)} \tag{2.4}$$

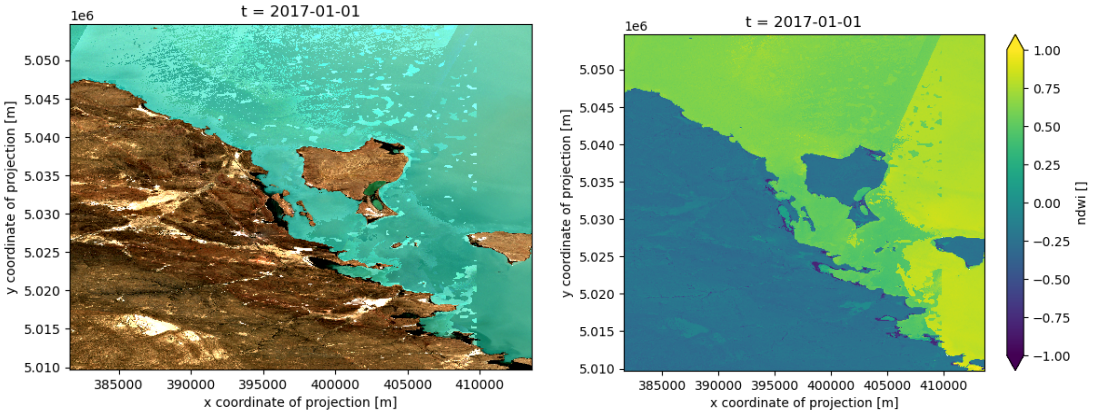
  The Enhanced Water Index (EWI) builds upon both NDWI and MNDWI by introducing additional terms to improve the accuracy of water detection, especially in more complex landscapes. It further refines water body identification by adjusting for vegetation presence using NDVI as part of the calculation.

- Automated Water Extraction Index (AWEI):

$$AWEI = 4 \times (GREEN - SWIR) - (2.5 \times NIR + 2.75 \times BLUE) \quad (2.5)$$

  AWEI is designed to automatically detect water bodies and reduce false positives caused by non-water surfaces like shadows or urban areas. By combining green, SWIR, NIR, and blue bands, it enhances the separation between water and other surfaces, making it particularly effective in built-up areas.

Each of these indices is designed to improve water detection under specific conditions or to mitigate the effects of particular environmental variables, such as urban areas or mixed land-cover types. For example, while NDWI is highly effective in clear, open areas, MNDWI and AWEI are better suited for urban regions or landscapes with mixed land cover [72]. These indices have become essential tools in remote sensing, as they help to mitigate the effects of environmental factors like vegetation, urban structures, or atmospheric noise, which can complicate water detection. While they represent significant advancements in remote sensing methodologies, their application is often context-dependent, and ongoing research continues to refine their robustness and adaptability to different geographical and environmental settings.



(a) Lake Balkhash RGB **(b)** Lake Balkhash NDWI

Figure 2.15: Comparison between RGB image and water index map

In Figure 2.15, we compare a standard RGB image of Lake Balkhash (Figure 2.15a) with its corresponding NDWI map (Figure 2.15b). The NDWI map highlights water bodies more clearly by taking advantage of the reflectance properties of water in the green and NIR bands, demonstrating how this index simplifies water detection from satellite imagery.

With the years many variations of the NDWI have been proposed to improve accuracy, such as for example the Modified Normalized Difference Water Index (MNDWI) [86], which use the Green and the Shortwave Infrared (SWIR) bands respectively as in Eq 2.3, and gives slightly better results in particularly difficult (and noisy) conditions. However, for the purposes of this thesis, we preferred to limit ourselves to using the NDWI as an example of the water index in the comparison experiments, as it remains the de facto standard for water extraction.

2.2.3 Copernicus Programme

The Copernicus programme is the remote sensing component of the European Union Space Programme and aims to develop a European Earth observation capability in partnership with the European Space Agency (ESA). There are currently six Sentinel missions under development, of which three are fully operational (Sentinel 1, 2 and 3) [83], two are partially operational (Sentinel 5 Precursor and Sentinel 6A) and one is in the final stages of development (Sentinel 4). Each Sentinel mission is designed to monitor a specific aspect of the Earth's environment, such as land, ocean, atmosphere and climate, and is equipped with a variety of sensors to collect data in different spectral bands. It is important to note that all data collected by the Sentinel missions are freely available to the public through the Copernicus Dataspace [80] and the openeo platform[87]. This makes the Sentinel missions an ideal source of data for remote sensing research. In our particular case we will focus on the Sentinel-2 mission for our research, which is designed to monitor land and vegetation, and is equipped with a MultiSpectral Instrument (MSI) that captures images in 13 spectral bands.

Sentinel-2

The Sentinel-2 mission [83] consists of two identical satellites, called Sentinel-2A and Sentinel-2B (with a third Sentinel-2C satellite scheduled to be launched in September 2024), which orbit the Earth in a sun-synchronous orbit. This configuration allows for a high revisit [83] [84] frequency of 5 days at the Equator, ensuring consistent and timely data collection. Each satellite is equipped with a MultiSpectral Instrument (MSI) that captures images in 13 spectral bands, each one with a resolution ranging from 10 to 60 meters per pixel. The wide range of spectral bands allows for the monitoring of various environmental parameters [88], such as vegetation, soil, and water quality, making it an ideal tool for environmental monitoring.

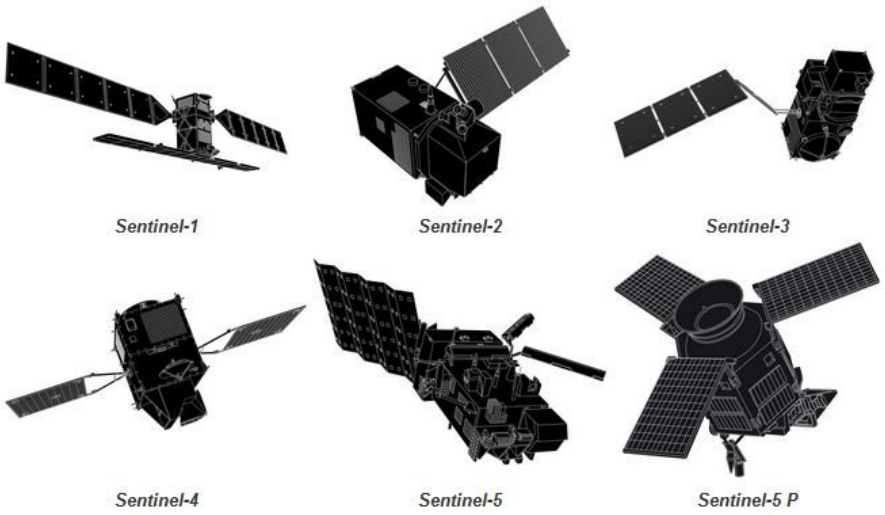


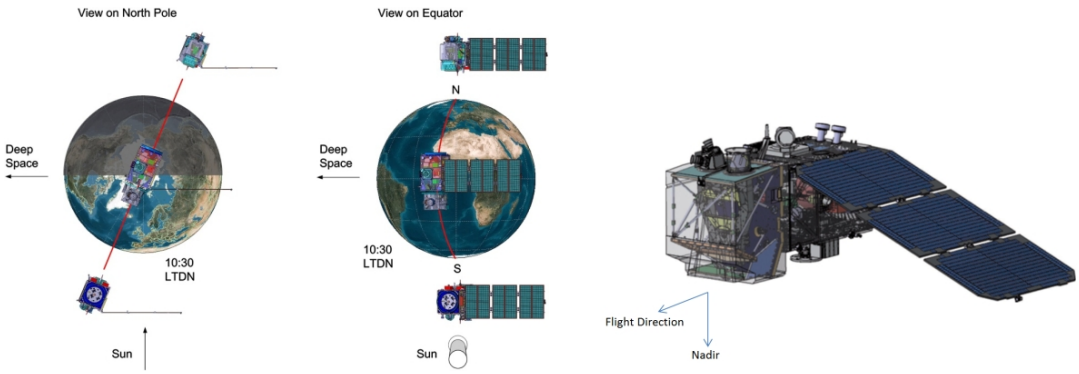
Figure 2.16: All sentinel missions currently deployed or in development. Source: [80]

2.2.4 Deep Learning for Remote Sensing

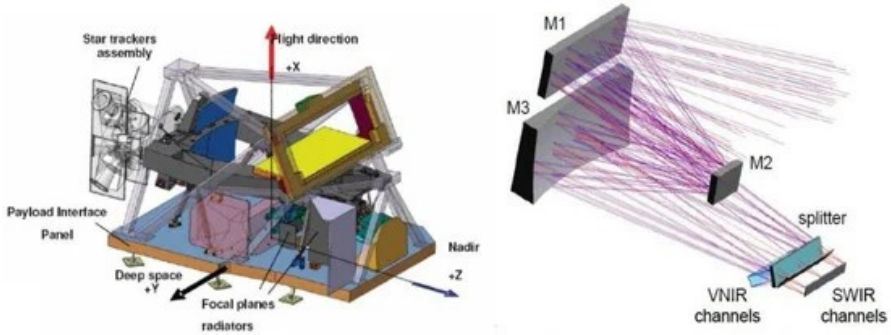
Machine Learning (ML) and Deep Learning (DL) have revolutionized the field of remote sensing by offering advanced tools for high-precision classification and recognition of geological features[58, 89]. Over the past few decades, significant milestones have been achieved in the development of both ML and DL techniques, allowing them to match or even surpass human performance in tasks such as image classification [25], object detection [24], and text processing [44]. As a result, these methods have gradually become the mainstream techniques for interpreting remote sensing data[90, 91].

The key advantage of ML and DL in remote sensing lies in their ability to extract high-level, high-dimensional semantic information [37] from a wide range of remote sensing images, including multispectral, hyperspectral, and radar imagery [92]. These methods can identify patterns and features that are difficult or impossible to discern using traditional methods, making them indispensable tools for monitoring and interpreting complex geological environments[90]. Their capacity to automatically learn from vast datasets has significantly improved the precision, efficiency, and scalability of remote sensing tasks, enabling large-scale, fine-grained geological surveys to be conducted at lower costs and with reduced human intervention[89].

In addition, the integration of ML and DL with multi-source remote-sensing data, such as data from different satellite platforms or sensors [58], has resulted in important advancements in the detection and analysis of geological features. These



(a) Sentinel-2 Orbital Configuration **(b)** Schematic view of Sentinel-2 Spacecraft



(c) Sentinel-2 MultiSpectral Instrument

Figure 2.17: Sentinel-2 Mission. Images from [83]

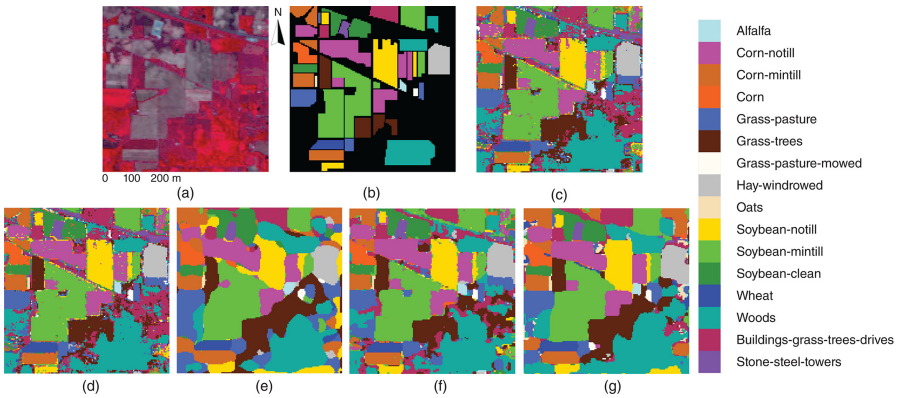


Figure 2.18: Example of land cover classification using various deep learning techniques[50]. (a) Original image, (b) expected labels, (c) stacked auto-encoders (SAE), (d) Deep Belief Network (DBN), (e) (f) (g) various CNNs architectures

advances are particularly relevant for applications such as mineral exploration, land-use monitoring [58], hazard detection, and climate change studies[9]. The ability

to fuse data and extract meaningful informations has become crucial in addressing problems related to sustainable development and environmental protection. For example, ML and DL algorithms are now being used to monitor deforestation, map water resources [78], track desertification, and assess natural disaster risks [82], all of which contribute to land management and conservation planning.

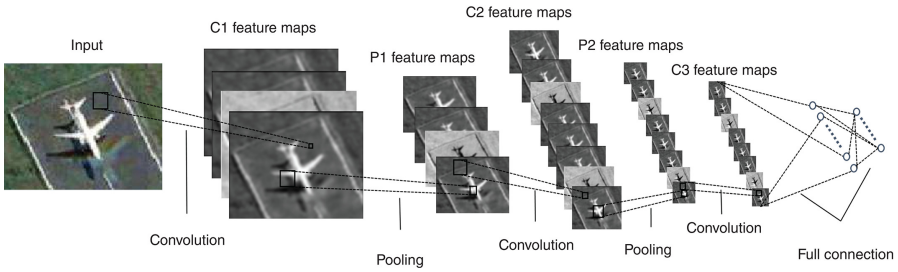


Figure 2.19: Feature mapping in CNNs for remote sensing[50]

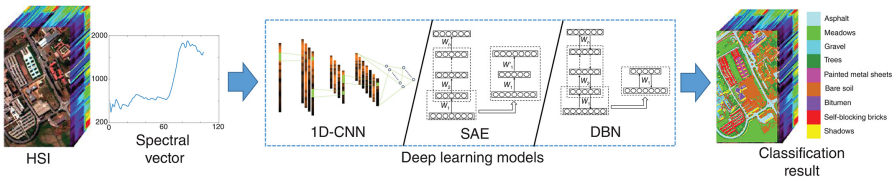


Figure 2.20: Classification process in CNNs for remote sensing[50]

Several freely available datasets[93, 94] are starting to be created, mainly based on either the Copernicus Sentinel missions or the Nasa Landsat missions. These datasets are designed to be real benchmarks for deep learning models[95, 93] for multispectral image analysis, in the same way that Imagenet[25] and Cityscapes[19, 28] are the main benchmarks in their respective disciplines, but also as sources for pretraining and fine-tuning models to be used later in various remote sensing tasks [94], as shown in Fig.2.21.

Despite their transformative potential, ML and DL techniques in remote sensing still face several challenges[58, 79]. One of the primary obstacles is the complexity of geological elements, which can vary significantly in appearance, scale, and composition across different regions[72]. Vegetation cover, topographic variations, and atmospheric conditions can interfere with accurate interpretation[88]. The quality of remote-sensing images may also be unstable due to cloud cover, sensor noise and varying spatial and temporal resolutions, further complicating the analysis[96]. In addition to that, the lack of comprehensive, high-quality ground truth data, which is essential for training and validating ML and DL models[97, 98], remains a significant limitation: field-validated data is often sparse, expensive to collect and

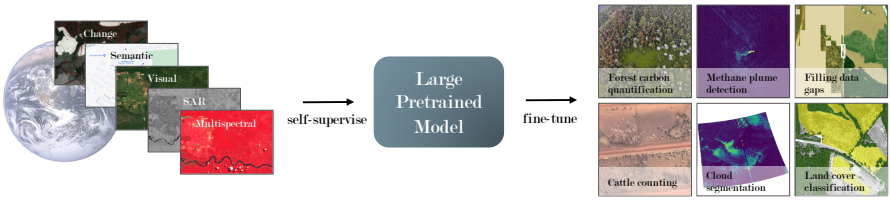


Figure 2.21: Pretraining on large datasets for remote sensing[94]

limited to specific regions, which can interfere with the generalization capabilities of the models.

Chapter 3

Methodology

In this chapter, we are going to present the dataset used for the experiments, how it was obtained, processed and organized, and the details of the models used for the experiments.

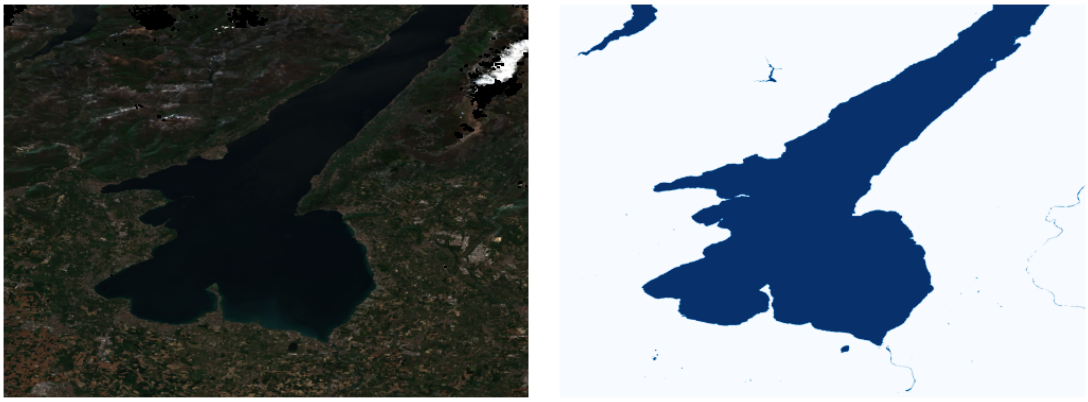
3.1 Problems Statement

In this thesis, we are going to address two main problems related to the detection and forecasting of water levels in satellite images, which are water segmentation and water level forecasting.

3.1.1 Water Segmentation

Water Segmentation is the task of mapping the presence of water in satellite images, which is a fundamental task in remote sensing and environmental monitoring. Accurate water detection aids in flood management, irrigation planning, hydrological modeling, and ecosystem protection. The main goal of water segmentation is to produce precise maps that differentiate water from non-water regions in images, enabling further analysis of water distribution and dynamics over time. Starting with a single satellite image of size WxH, we want to train a model capable of detecting the presence of water in the image. In particular, the model should produce a pixel-wise binary mask, in which each pixel is classified as either water (with a label of 1) or non-water (with a label of 0). The binary nature of the task simplifies the segmentation problem, making it easier to evaluate the performance of the models. It is important to note that for the scope of this experiment, we will ignore the seasonal changes in water levels and focus on the detection of permanent water in the image. Thus, we expect a minimum of inaccuracy related to small differences between the annual mask and the actual water level in the

image, but no more than a few percentage points. Water segmentation has been extensively studied in the past by various researchers using many different methods and algorithms. The aim in this thesis is simply to compare the capabilities of CNNs models with one of the most widely used methods, namely the NDWI-based method.



(a) Lake Garda RGB Image **(b)** Expected water mask for Lake Garda

Figure 3.1: Examples of water detection task

3.1.2 Water Level Forecasting

Water level forecasting is the task of predicting changes in water levels over time using satellite imagery, which is crucial for environmental monitoring and climate change studies. The goal here is to train a model that can forecast future water level changes based on a time series of labeled satellite images of size WxH. Each input image has two corresponding labels: one from the first year and another from one year after the last image in the time series. The model's expected output is the predicted change in water level, defined as the pixel-wise difference between the first and last labels.

Classification In the classification sub-task, we simplify the pixel-wise water level masks to a single value: the percentage of water pixels that have changed in the image. Using a defined threshold, the model will classify each image as either showing a "positive change," "no change," or "negative change" in water levels as shown in Fig.3.2.

Regression For the regression task, we again reduce the pixel-wise water masks to a single value, which is the percentage change in water pixels. Instead of

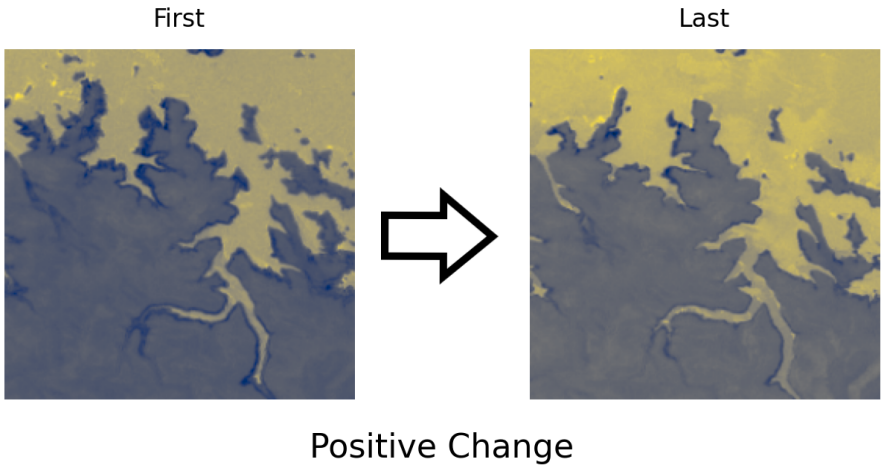


Figure 3.2: Example of water level classification task. Lake Nasser(Egypt).

classifying the change, the model will try to predict the exact percentage change in water pixels as shown in Fig.3.3. This task provides a more accurate numerical estimate of how much the water level has changed between the two labels than simply categorising the change.

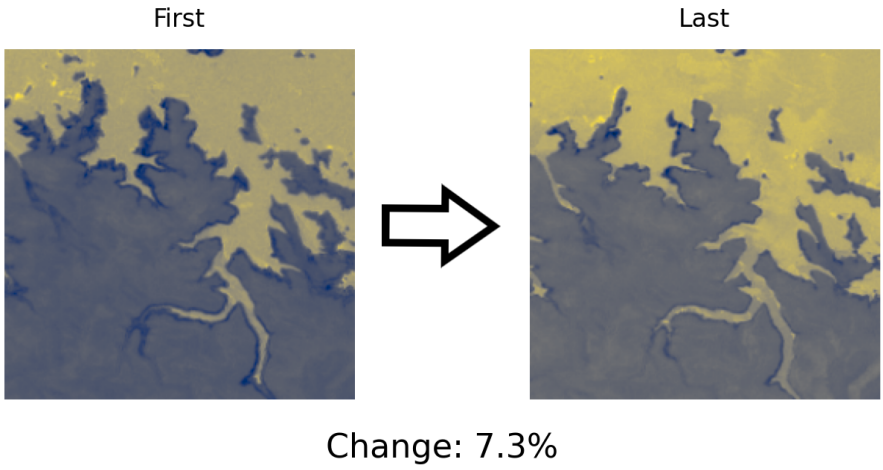


Figure 3.3: Example of water level regression task. Lake Nasser(Egypt).

Segmentation In the segmentation task, the model will directly predict the pixel-wise difference between the first and last water level labels. This involves generating a detailed mask that shows where water levels have changed in the image, offering a more granular view of water distribution over time. An example can be

seen in Fig.3.4 where the (future) change mask of a particular tile is compared with the first and the last images of the timeSeries.

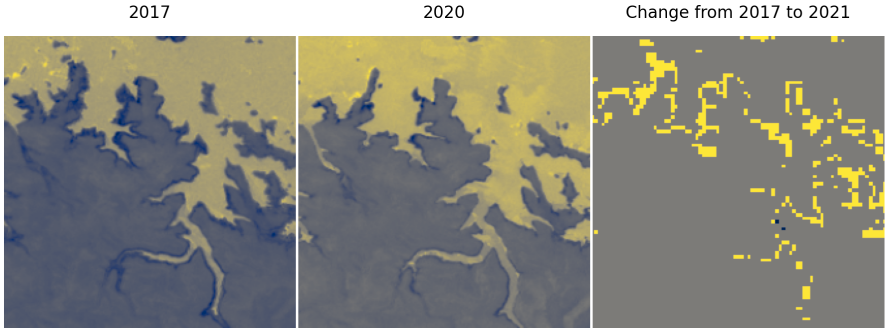


Figure 3.4: Example of water level segmentation task. Lake Nasser(Egypt).
The mask highlighs the areas where the water level has changed, either increasing (yellow , positive change) or decreasing (blue, negative change).

To generate water level predictions, the time series data consists of four satellite images taken over consecutive years (2017 to 2020), while the labels represent two water level masks: one from the first year (2017) and the other from one year after the last image (2021). The model is tasked with predicting the change in water level between the first image (2017) and the final label (2021) using the four images in the time series.

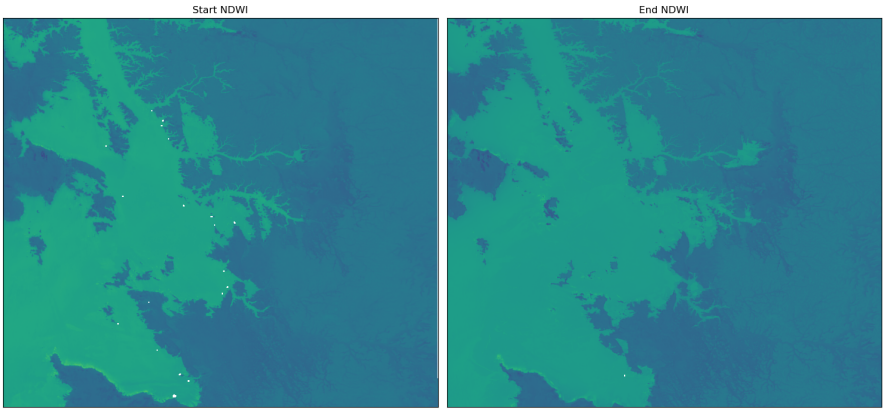


Figure 3.5: First (2017) and last (2020) NDWI images of the time series. Lake Nasser(Egypt).
Slight differences in water levels can be seen, particularly along the more indented coastlines and islands

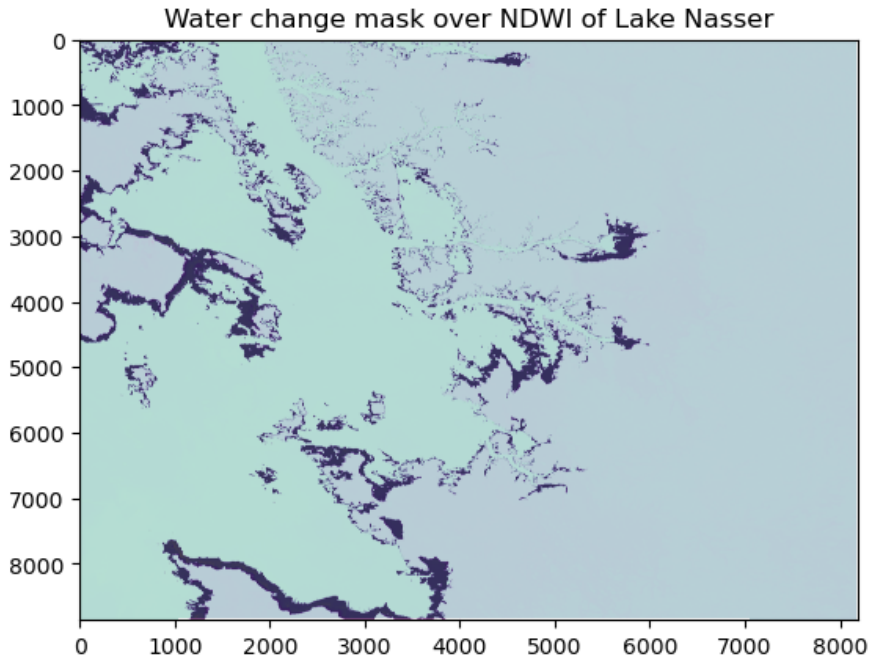


Figure 3.6: Example of water level forecasting task. Lake Nasser(Egypt). The image is obtained by overlapping the change mask that the model is tasked to predict over the NDWI image of the last year (2020).

3.2 Dataset

The data set used for the experiments consists of images taken by the Sentinel-2 satellite, which is part of the European Space Agency's Copernicus programme. The images are taken at a resolution of 10 metres per pixel and are composed of 9 bands out of the original 12 included in Level-2A of the Sentinel-2 products. The bands used have been selected for their importance in detecting water and forecasting water levels, and are the most commonly used bands in remote sensing. In particular, the bands used are the following:

Each image has a size of about 2000 x 2000 pixels and is obtained by merging about 60 days of observations to exclude as many clouds as possible. The cloud exclusion is done using the scene classification map [100] provided by the Copernicus program, which contains for each pixel its possible classification as cloudy or not. The dataset used consists of almost 50 of these patches, taken from 20 different lakes around the world. Each patch consists of 4 images taken in the following years (2017 to 2020), for a total of almost 200 images taken by Sentinel-2, and 2 masks, one taken in the first year (2017) and the other one year after the last image (2021).

Band	Name	Resolution	Wavelength	Description
B02	Blue	10 m	490 nm	Visible blue
B03	Green	10 m	560 nm	Visible green
B04	Red	10 m	665nm	Visible red
B05	Red Edge 1	20 m	705nm	Vegetation classification
B06	Red Edge 2	20 m	740nm	Vegetation classification
B07	Red Edge 3	20 m	783nm	Vegetation classification
B08	NIR	10 m	842nm	Mapping shorelines
B11	SWIR 1	20 m	1610nm	Measuring moisture content
B12	SWIR 2	20 m	2190nm	Measuring moisture content

Table 3.1: Sentinel-2 [99] bands used in dataset

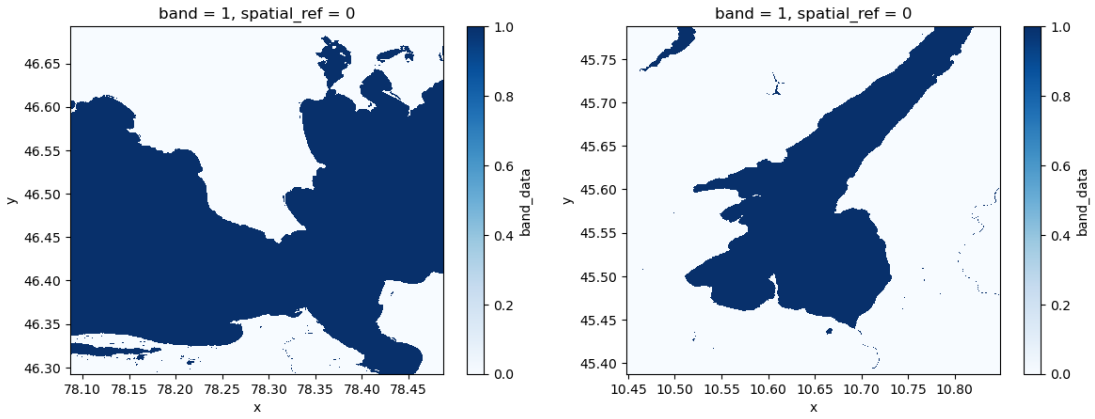
3.2.1 Global Surface Water

All the ground truth data used in this research is obtained from the Global Surface Water dataset [2], which is a freely available collection of water masks obtained from the Landsat 5, 7, and 8 satellites. Following similar works in the field of remote sensing [101, 1], the Global Surface Water dataset was chosen for its global coverage, high resolution, and long temporal range. Each pixel in the dataset was labeled as open water, land or non-valid through an expert system (i.e. manual work) that analyzed each Landsat images to produce a monthly observation mask [2]. The dataset is available at a resolution of 30 meters per pixel and covers the entire globe from 1984 to 2021 with various type of products. Hence, our research will focus on the years were both the Sentinel-2 and the Global Surface Water datasets overlap, from 2017 to 2021. It is important to note that the Global Surface Water dataset is not perfect and may contain errors, estimated to be around 1% for false water detections and 5% of missed water classification of the total area [2]. Another limitation of the dataset is the lack of available observations if the area is covered by clouds, given the inability of Landsat's sensors to pass through heavy clouds, which can lead to missing data in the water mask.

3.2.2 Data Preprocessing and Tiling

The preprocessing of the data is an important step in order to prepare the input raw multispectral images for the training of the models. In this case the main preprocessing steps includes Data Cleaning, Geospatial Normalization, and Tiling.

Data Cleaning is the process of removing any corrupted or missing data from the images, while Geospatial Normalization is the process of aligning the images to a common coordinate system. Each image is checked for missing data, bands or corrupted pixels, with any image with more than 5% of missing data along all



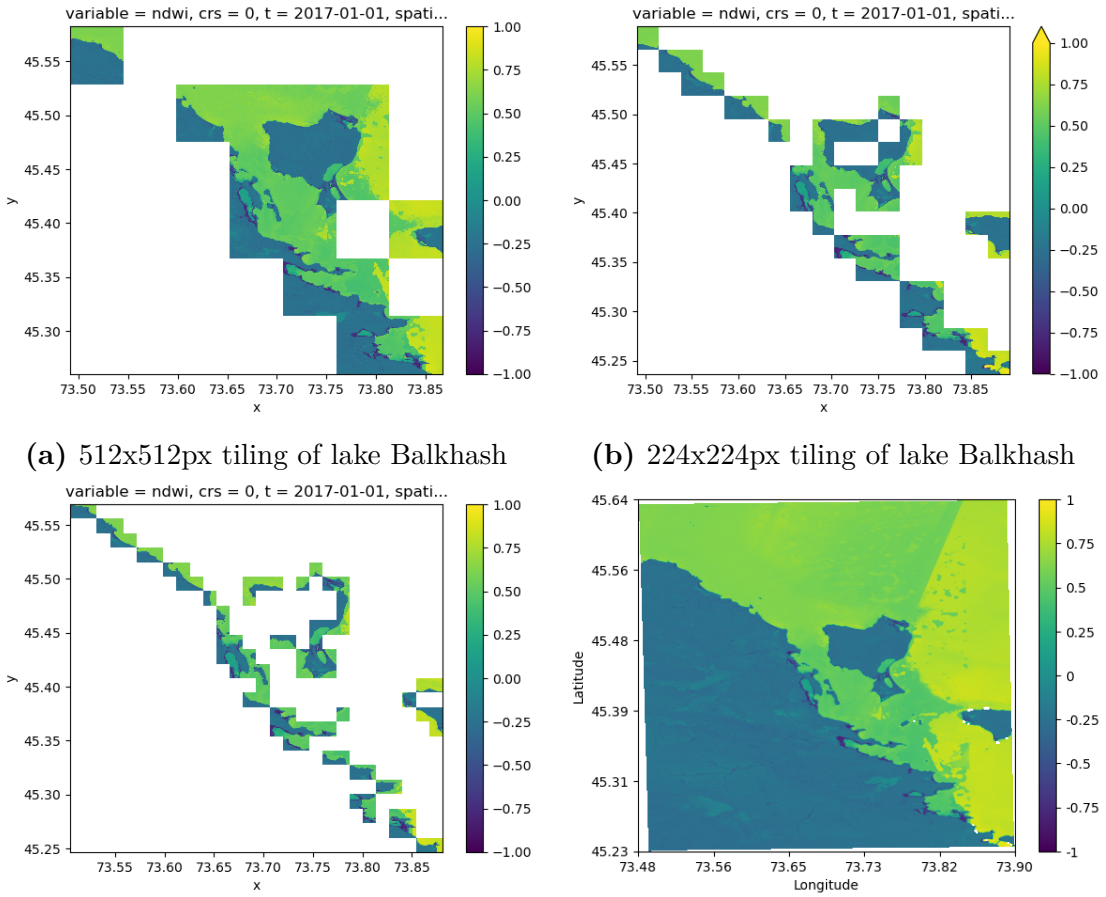
(a) Lake Balkhash (2017) **(b)** Lake Garda (2017)

Figure 3.7: Examples of water mapping from the Global Surface Water dataset

dimensions being discarded.

The Geospatial Normalization is done by aligning the images to a common coordinate system, in this case the WGS84[102, World Geodetic System 1984] coordinate system, which is the most commonly used coordinate system for satellite images. This step is important in order to ensure that each single pixel is perfectly aligned across all the images and the labels, which is critical for the correct training of the models.

Finally, the images are tiled into smaller squared patches, which are used as input for the models. During the tiling process, only some parts of the images are selected, as become clear in Fig 3.8 ,in order to exclude less informative areas such as borders, land areas and open water areas. The latter is done in order to balance the amount of water and non-water pixels in the dataset, mantaining only tiles with at least 5% of either land or water. The size of the tiles must be consistent across all the images and labels, and the correct size must be chosen in order to balance the trade-off between the amount of data contained in each tile and the capacity of each model to process those data. As shown in Fig. 3.8a, Fig. 3.8b and Fig. 3.8c the tiling process is done by dividing the image into smaller squared patches, which are then used as input for the models. As a result, the final dataset used for the training of the models will consist of roughly 1000 or more tiles, depending on the chosen tile size, each containing 4 images and 2 masks, for a total of almost 6000 images and 2000 masks.



(a) 512x512px tiling of lake Balkhash

(b) 224x224px tiling of lake Balkhash

(c) 128x128px tiling of lake Balkhash

(d) Full image of lake Balkhash

Figure 3.8: Examples of tiling using different tile sizes. Lake Balkhash (Kazakhstan)

3.3 Losses and Metrics

In this section, we are going to present the loss functions and metrics used for the experiments, and the reasons behind the choice of each loss function.

3.3.1 Loss Functions

When training a neural network, loss functions help quantify the 'cost' of the model's predictions. They essentially measure how far the predicted output is from the true answer. The smaller the loss, the closer the model's predictions are to being correct, and the better it is performing. Loss functions also play a key role in guiding the optimization process, as the model continuously adjusts its internal

parameters to minimize the loss during training.

The loss function L can be generalised by the following formula

$$L = E(G, S) \tag{3.1}$$

where G are the ground truth (the correct answers) and S are the model's predictions.

Choosing the right loss function can have a major impact on the training process and the final performance of the model. Different tasks require different loss functions to achieve the best results possible. For example, some loss functions work well only on balanced datasets, while others are better suited for unbalanced data, where one class might be more frequent than another. This is particularly important in water segmentation and water level forecasting, where class imbalances or outliers can affect performance. One of the most commonly used loss functions is Cross Entropy Loss (3.2), which is used in both classification and segmentation tasks. However, there are many other loss functions that can be used for different tasks, such as Mean Squared Error (MSE) for regression tasks and Dice Loss for semantic segmentation.

$$-\sum_{c=1}^{M} y_{o,c} \log(p_{o,c}) \tag{3.2}$$

In this thesis, we have used Dice Loss for the water detection task and Focal Loss, Huber Loss, and Generalized Dice Loss for the water level forecasting task. The choice of these loss functions is based on the nature of the tasks where they are used.

Focal Loss (3.3) in the water level forecasting task to handle the classification instead of Cross Entropy Loss. Helps deal with class imbalance by down-weighting the loss contribution of well-classified examples, focusing more on the harder, misclassified examples. This is useful in cases where certain outcomes are more frequent than others, as is the case with the water level forecasting dataset, where most of the samples show no change in water level.

$$\mathrm{FL}(p_\mathrm{t}) = -(1 - p_\mathrm{t})^\gamma \log(p_\mathrm{t}) \tag{3.3}$$

Huber Loss (3.4) was used in the water level forecasting task to handle regression. It is particularly useful when dealing with outliers in the data, as it combines the best of Mean Squared Error (MSE) and Mean Absolute Error (MAE). For smaller errors, it behaves like MSE, while for larger errors (outliers), it behaves like MAE, preventing large deviations from dominating the loss.

$$L_\delta = \begin{cases} \frac{1}{2}(y - \hat{y})^2 & \text{if } |y - \hat{y}| < \delta \\ \delta\left(|y - \hat{y}| - \frac{1}{2}\delta\right) & \text{otherwise} \end{cases} \tag{3.4}$$

For water level forecasting segmentation was used Generalized Dice Loss (3.5). This loss function extends Dice Loss to handle cases with class imbalances more effectively by applying class-specific weights. It ensures that smaller classes, like negative change in water levels, are given enough attention during training, helping the model to perform better on imbalanced datasets.

$$\text{GDL} = 1 - 2\frac{\sum_{l=1}^{L} w_l \sum_n r_{ln} p_{ln}}{\sum_{l=1}^{L} w_l \sum_n (r_{ln} + p_{ln})} \tag{3.5}$$

3.3.2 Metrics

To evaluate the performance of the models, we used several metrics, mainly due to the different tasks addressed. For the classification task, we used the accuracy (3.6) and F1 score (3.7) metrics, which are commonly used in classification tasks. Accuracy is a measure of the overall performance of the model, while F1 Score is a measure of the precision and recall of the model. In this case, the F1 score was the main metric used to evaluate the real performance of the models in the classification tasks, given the high class imbalance in the dataset.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.6}$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN} \tag{3.7}$$

For the regression task, we used the Mean Absolute Error (MAE) (3.8) and the R2 Score (3.9) metrics, which are commonly used in regression tasks. MAE is a measure of the average error of the model, while R2 Score is a measure of the goodness of fit of the model. In this case, the R2 Score was the main metric used to evaluate the real performance of the models in the regression tasks, given the high variability of the data and the presence of outliers.

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{3.8}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2} \tag{3.9}$$

For the segmentation task, we used the Intersection over Union (IoU) (3.10) metric. This is the most commonly used metric for segmentation tasks. IoU is a measure of the overlap between the ground truth and the model prediction, and is used to evaluate the performance of the model in the segmentation task.

$$IoU = \frac{TP}{TP + FP + FN} \tag{3.10}$$

Chapter 4

Experiments

Here we present the results of the experiments conducted in this research, including an evaluation of the models and a comparison of the results.

4.1 Models

In this section we present and explain the models used for the experiments, which are divided into three main categories based on the task at hand: Classification, Regression, and Segmentation.

4.1.1 Classification and Regression

For the classification and regression tasks in water level forecasting, we experimented with several state-of-the-art Convolutional Neural Network (CNN) architectures alongside a Vision Transformer (ViT) model. This allowed us to compare traditional CNN-based approaches with newer transformer-based techniques, providing a more comprehensive understanding of how each method performs in water detection and water level forecasting tasks

As shown in Table 4.1, the models used for the classification and regression tasks are ResNet18, ResNet34, ResNet50, MobileNet, ViT, ConvNext, and VGG. The choice of these models is based on their popularity and performance in similar tasks, as well as their ability to handle the data at hand.

ResNet models We used three different ResNet models for the classification and regression tasks: ResNet18, ResNet34, and ResNet50. The main differences between the three models are the number of layers and the number of parameters, with ResNet18 being the smallest and ResNet50 being the largest.

Model	Params	Year	Architecture Description
ResNet18	11.3M	2015	18-layer network with skip connections
ResNet34	21.8M	2015	34-layer deeper residual network
ResNet50	25.6M	2015	50-layer network with bottlenecks
MobileNet	2.97M	2017	Lightweight model using depthwise convolutions
LeViT	31M	2021	Hybrid model with convolution and self-attention
ConvNext	28.6M	2022	Combines ConvNet and Transformer features
VGG	14M	2014	Simple CNN with stacked convolution layers

Table 4.1: Overview of CNN and ViT models used for classification and regression tasks in water level forecasting

MobileNet We used the MobileNetV4 small convolutional version of the MobileNet model for the classification and regression tasks. This model in particular, among all the different instances of MNv4 models, is designed to be lightweight and efficient, making it ideal for mobile and edge devices, the only downside is the absence of flash attention, used in the hybrid version of the other models.

Vision Transformer The Vision Transformer (ViT) [67] is a transformer-based architecture that has been shown to be highly effective in image classification tasks. In our case, we used the LeViT-256[69] model for the classification and regression tasks, which is a hybrid ViT model that combines the best of both CNNs and transformers. The input images are preprocessed by 4 convolutional layers with 3x3 kernels and a stride of 2, which are responsible for extracting high-level features from the images, before being passed to the transformer stages for further processing. In this way, the model can learn both the local and global features of the images, making it highly effective in image classification tasks.

ConvNext As one of the most recent models, ConvNext [51] is a "pure" convolutional neural network that has been shown to be highly efficient in various computer vision tasks. There are various versions of the ConvNext model, from the convnext femto to the convnext large, with the main difference being the number of layers and the complexity of the network. In our case, we used the ConvNext tiny model for the classification and regression tasks, which is a medium version of the model that is designed to be efficient and fast. An important note is that the ConvNext model is highly sensitive to the learning rate, and it is important to find the right learning rate for the model to perform well.

4.1.2 Segmentation

For the segmentation task in both water detection and water level forecasting, we employed several advanced models: U-Net, DeepLabV3, and PSPNet. These architectures have been extensively used for pixel-level classification tasks, such as semantic segmentation, making them highly suitable for extracting detailed information from satellite images. All models are taken from Pytorch's Segmentation Models library [103], which provides a wide range of pre-trained segmentation architectures for various tasks. For each different architecture we employed various encoders, which are the part of the network responsible for extracting features from the input images. The encoders employed for the segmentation task are listed in Table 4.3. They are all taken from either Segmentation Models Pytorch or Timm libraries, and are pre-trained on the ImageNet dataset.

Model	#Params	Architecture Description
U-Net	31M	U-shaped CNN with skip connections
PSPNet	46.5M	Multi-scale context aggregation
DeepLabV3	59M	Atrous convolutions for dense pixel prediction

Table 4.2: Overview of segmentation models used for water detection and water level forecasting tasks

Model	#Params	Architecture Description
ResNet18	11.3M	18-layer residual network
ResNet34	21.8M	34-layer residual network
ResNet50	25.6M	50-layer network with bottlenecks
MobileNet	2.97M	Lightweight model with depthwise convolutions
MaxViT	31M	Hybrid model combining CNNs and Transformers
VGG	14M	Simple deep CNN for image classification

Table 4.3: Overview of CNN and ViT models used as encoders for classification and regression tasks

4.2 General Settings

All the experiments were runned on a single NVIDIA RTX 2080TI using the Pytorch library [104] and the Pytorch Lightning framework [105]. Timm library[106] was used for the implementation and the pretrained wweights of most of the models, the convolutional Networks for semantic Segmentation were taken from Pytorch

Segmentation Models [103]. For all the experiments we used the AdamW optimizer, with a weight decaying factor of 0.01, and exponential learning rate scheduler with a gamma of 0.9.

4.3 Water Segmentation

The first series of experiment was aimed at comparing the performance of different models in the task of water detection in satellite images. In this first experiment, we used the Sentinel-2 dataset, which consists of 9 bands, with each input image having a size of 224x224 pixels.

4.3.1 Settings

U-Net We used the U-Net architecture for the water detection task, with various encoders to evaluate how the choice of encoder affects the performance of the model. The U-Net model was taken from the library Segmentation Models Pytorch [103] and trained with a learning rate of 0.0001, a batch size of 12, and a total of 50 epochs. We set the encoder depth to 5 and used a decoder with 16, 32, 64, 128 and 256 filters, respectively. In other words, our Unet model has 5 layers in the encoder and 5 layers in the decoder, with the number of filters doubling at each layer, each one connected with a skip-connection.

PSPNet We used the PSPNet architecture for the water detection task, with various encoders to evaluate how the choice of encoder affects the performance of the model. The PSPNet model was taken from the library Segmentation Models Pytorch [103] and trained with a learning rate of 0.0001, a batch size of 12, and a total of 50 epochs. We set the encoder depth to 3 and used a decoder, in this case called Spatial Pyramid, with 512 filters and a spatial dropout of 0.2.

DeepLabV3 We used the DeepLabV3 architecture for the water detection task, with various encoders to evaluate how the choice of encoder affects the performance of the model. The DeepLabV3 model was taken from the library Segmentation Models Pytorch [103] and trained with a learning rate of 0.0001, a batch size of 12, and a total of 50 epochs. We set the encoder depth to 5 and used a Atreus Spatial Pyramid Pooling with 256 filters and a final upsampling of 8x to maintain the input-output spatial consistency.

4.3.2 Results

The first experiment conducted involves the use of a U-Net architecture for water segmentation, paired with various encoders to evaluate how the choice of encoder

affects the performance of the model. From the results shown in Table 4.4, the U-Net model achieves its best performance with a ResNet50 encoder, reaching an Intersection over Union (IoU) of 92.7%. It is worth noting that the ResNet18 and

Model	Encoder	Tiles	IoU	T. Time
U-Net	ResNet18	224px	91.9	**6.28 min**
	ResNet34	224px	91.6	7.37 min
	ResNet50	224px	**92.7**	10.15 min
	VGG-16	224px	92.4	11.33 min
	ViT	224px	91.9	16.27 min
	MobileNetV3	224px	91.4	6.46 min

Table 4.4: Water detection results for U-Net

MobileNetV3 encoders, despite having fewer parameters and requiring significantly shorter training times, yield only slightly lower results, with IoUs of 91.9% and 91.4%, respectively. The faster training time, especially for MobileNetV3, which completed the training in just 6.46 minutes, demonstrates the trade-off between model complexity and computational efficiency. These results indicate that for certain practical applications where computational resources are limited, using a lighter encoder such as MobileNetV3 may be a reasonable compromise, offering competitive performance with much faster training.

Following the results of the U-Net model, we compared the performance of the PSPNet and DeepLabV3 models, using the same encoders and the same data set. From the results in Table 4.5, we observe that the best IoU for the PSPNet model is 92.1%, achieved using a Vision Transformer (ViT) encoder. Interestingly, despite PSPNet's slightly lower performance compared to U-Net, the training time was faster, with ViT requiring only 8.33 minutes. This shows PSPNet's ability to achieve competitive performance while maintaining computational efficiency, particularly with modern transformer-based encoders like ViT.

Model	Encoder	Tiles	IoU	Time
PSPNet	ResNet18	224px	90.0	**5.70 min**
	ResNet34	224px	90.4	5.88 min
	ResNet50	224px	90.0	5.92 min
	VGG-16	224px	91.5	8.38 min
	ViT	224px	**92.1**	8.33 min
	MobileNetV3	224px	86.5	5.75 min

Table 4.5: Water detection results for PSPNet

Comparing the results of the DeepLabV3 model in Table 4.6, we can see that

the best results are obtained with the ResNet50 encoder, with an IoU of 92.0%. However, DeepLabV3's training time with ResNet50 is significantly longer—over 30 minutes—indicating that its architecture, particularly the use of atrous convolutions, may not be well-suited to the multispectral nature of the input images. Additionally, DeepLabV3 was tested with fewer encoder options since VGG-16 and ViT were incompatible with its structure. Despite these limitations, DeepLabV3 achieves comparable performance to U-Net and PSPNet but requires much more computational resources, particularly when using a deep encoder like ResNet50.

Model	Encoder	Tiles	IoU	Time
DeepLab	ResNet18	224px	91.2	11.58 min
	ResNet34	224px	91.8	17.88 min
	ResNet50	224px	**92.0**	30.52 min
	MobileNetV3	224px	88.5	**11.55 min**

Table 4.6: Water detection results for DeepLabV3

Lastly we compared the performance of the best deep learning models with a classical water detection approach using the Normalized Difference Water Index (NDWI). As shown in Table 4.7, the NDWI method performs significantly worse than any of the deep learning approaches, achieving an IoU of only 83.4This result underscores the effectiveness of deep learning techniques in handling the complex task of water segmentation, where the combination of spectral, spatial, and temporal information in satellite imagery allows for much higher accuracy. The best model in this comparison is the U-Net architecture with a ResNet50 encoder, which outperforms all other models with an IoU of 92.7%. In conclusion,

Method	Tiles	IoU	T. Time
UNet	224px	**92.7**	10.15 min
PSPNet	224px	92.1	8.33 min
DeepLab	224px	92.0	30.52 min
NDWI	224px	83.4	

Table 4.7: Comparison of water detection methods

while classical methods like NDWI are still used due to their simplicity, modern deep learning-based approaches, particularly U-Net with ResNet50, offer superior performance in water body segmentation. The choice of encoder and architecture also plays a crucial role in balancing the trade-off between computational efficiency and accuracy, with deeper encoders like ResNet50 providing the best results at the cost of longer training times.

4.4 Water Level Forecasting

In the second series of experiments, we compared three main tasks in the prediction of water levels in lakes and rivers: regression, classification, and semantic segmentation. For this experiment, we always used the same Sentinel-2 based dataset, but with a tile size of 128x128,224x224 and 512x512 pixels, and the NDWI index.

4.4.1 Settings

ResNet models All three models are taken from the Timm library and are pre-trained on the ImageNet dataset. The learning rate used for the training of these models is 0.0001, with a batch size of 12 and a total of 50 epochs.

MobileNet Like the previous,this model is taken from the Timm library and is pre-trained on the ImageNet dataset. The learning rate used for the training of this model is 0.0005, with a batch size of 12 and a total of 50 epochs.

Vision Transformer Like the previous models, this model is taken from the Timm library and is pre-trained on the ImageNet dataset. The learning rate used for the training of this model is 0.0005, with a batch size of 12 and a total of 50 epochs.

ConvNext Like the previous models, this model is taken from the Timm library and is pre-trained on the ImageNet dataset. The learning rate used for the training of this model is 1e-06, with a batch size of 12 and a total of 50 epochs.

4.4.2 Threshold selection

To find the right threshold for the water level forecasting classification task, we compared the results of training a ResNet18 model with 5 different thresholds (0.0001, 0.001, 0.01, 0.02, 0.04), measuring the accuracy, the F1 score, and the balance between the three classes in the train set. The model is trained with suboptimal hyperparameters for 25 epochs using all bands present in the data, excluding the NDWI index, and the results are shown in Table 4.8. We found that the best absolute F1 score was obtained with a threshold of 0.04, corresponding to a change of 4.0% of the total water of the tile, while the best accuracy of 92.0% was obtained with a threshold of 0.02, corresponding to a change of 2.0%. However, due to the high imbalance generated between the different classes, we preferred to use the second best threshold of 0.01, corresponding to a change of 1.0%, which gives us a much more balanced and realistic distinction between "changed" and "constant"

water levels. We will use this threshold in all of the experiments contained in this thesis.

Threshold	Accuracy (%)	F1 Score	Positive	Negative	Zero
0.0001	86.7	77.1	1200	576	28
0.001	86.9	74.5	1132	496	176
0.01	90.9	87.8	928	240	636
0.02	**92.0**	85.6	852	160	792
0.04	91.3	**89.6**	708	88	1008

Table 4.8: Results of the threshold selection

4.4.3 Classification

In this series of experiments, we compared the performance of different models and approaches for the classification of future water level in our dataset. The classification task was performed by assigning at each tile one of three classes:

- Positive change : the level of water is increased between the first sample (2017) and the control label(2021)
- Negative change : the level of water is decreased between the first sample (2021) and the control label(2021)
- No change: the level of water is almost unchanged between the firs sample and the control label.

Note that a certain threshold is taken into account: any change, positive or negative, below this threshold is ignored and the tile is classified as "no change". For all the classification experiments, we used a Focal Loss as loss function, which is a modified version of the Cross Entropy Loss that gives more importance to the misclassified samples. We used F1 score as the main metric to evaluate the performance of the model.

Bands Comparison

One of the main properties of satellite images is the high number of bands that can be used to extract information, and the choice of how many bands and which bands to be used is crucial for the performance of the model. On the other hand, the high number of bands can lead to a high computational cost and a high risk of overfitting. Another alternative is the presence of the NDWI index, which is a widely used index for water features extraction and seems to contain a lot of informations compared to

its size. Here we compare the results of training a ResNet18 and a ResNet50 models with 4 different configurations of bands (NDWI,RGB,RGB+NIR+SWIR and all bands), measuring the accuracy and the F1 score. As we can see in Table 4.9, the best results are obtained by using all the bands, with a F1 score of 89.9% for the ResNet18 model and 90.6% for the ResNet50 model. It is important to note that the 6-band configuration, which includes only RGB, NIR and SWIR bands, gives a slightly worse result than the 9-band configuration, even though it is significantly smaller and easier to handle. So if our focus would be on performance we should consider the correct selection of bands for our training data, given that the NIR and SWIR bands, in literature, have been the most effective to delineate water features. Another interesting result is the performance of the NDWI index, which gives a F1 score of 84.8% for the ResNet18 model and 82.8% for the ResNet50 model. Although not significantly worse than the other configurations, this configuration was found to easily overfit the models during training, as we can see in Fig. 4.1.

Model	Bands	F1 Score	Accuracy
ResNet18[48]	NDWI	84.8	90.7
	RGB	86.4	91.6
	6B	87.8	91.3
	MSI	89.9	92.0
ResNet50[48]	NDWI	82.8	88.4
	RGB	86.3	89.8
	6B	89.8	92.2
	MSI	**90.6**	**93.1**

Table 4.9: Results for bands comparison for Classification

Time Series Analysis

In this experiment, we compared the results of training a ResNet18 model with the best configuration of bands and patch size, using various time series of images as input. The ResNet18 model is trained with 1,2,3 and finally 4 images of the same tile, taken at different times, and the results are shown in Table 4.10.

From the results in table 4.10 we can see that the best results are obtained using 4 images of the same tile taken at different times, with an F1 score of 89.9% and an accuracy of 92.0%. This result is expected as the model can use the temporal information to better understand the changes in the water level of the tile. However, we can see that there is a fairly consistent increase in performance as the number of time samples increases, reaching what appears to be a plateau between 3 and 4 time steps. This result is probably due to the fact that the model is able to use the temporal information to better understand the changes in the water level of

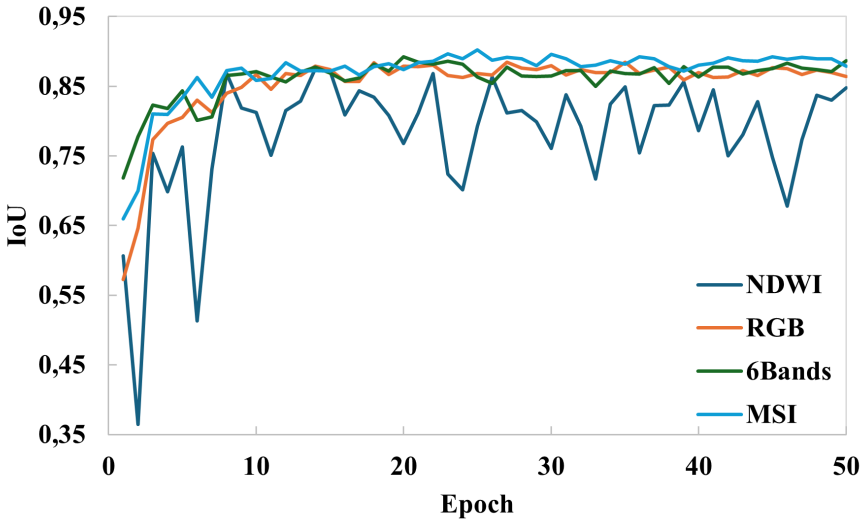


Figure 4.1: IoU during training for different bands configurations

Bands	Tile Size	Time Steps	F1 Score	Accuracy
MSI	224px	1 (only last)	85.8	91.5
MSI	224px	2 (first & last)	87.5	91.6
MSI	224px	3	89.1	91.5
MSI	224px	4	**89.9**	**92.0**

Table 4.10: Results for Time Series Analysis

the tile, but increasing the number of time samples also increases the complexity of the model and the risk of overfitting.

Models Comparison

In this experiment we compare the results of training a ResNet18, a ResNet34, a ResNet50, a MobileNetV4 and a ViT model with the best configuration of bands and patch size, comparing the Accuracy and the F1 score on the test set. From Table 4.11 we can see that the best results are obtained with the ResNet50 model, with an F1 score of 90.6% and an accuracy of 93.1%. Is also whorty of note the comparison with the classic machine learning models, which have a significantly worse performance than the deep learning models, with the Random Forest model being the best with an F1 score of 65.2% and an accuracy of 69.4%. That being said, all the classic models have been trained and used only on the NDWI idex and not on the whole multi spectral images due to the limitations in complexity and efficiency of the algorithms used. Therefore, we can state that the deep learning models

Model	Bands	Tile Size	F1 Score	Accuracy	Time
ResNet18 [48]	MSI	224px	89.8	92.0	**12.69 min**
ResNet34 [48]	MSI	224px	88.5	91.1	23.76 min
ResNet50 [48]	MSI	224px	**90.6**	**93.1**	28.21 min
ViT [69]	MSI	224px	87.9	91.3	16.87 min
MobileNet [55]	MSI	224px	81.6	87.3	13.20 min
ConvNext [51]	MSI	224px	89.4	91.6	15.13 min
SVM-poly	NDWI	224px	40.6	54.3	1.10 min
SVM-rbf	NDWI	224px	60.8	65.6	1.01 min
Random Forest	NDWI	224px	66.7	70.5	1.04 min

Table 4.11: Comparison of models and methods for classification

are more effective than the previous state-of-the-art machine learning methods for this task. Among the deep learning models, we can see that the ViT model, despite being one of the most modern and complex models, has a slightly worse performance than the ResNet18 model, with an F1 score of 87.9% and an accuracy of 91.3%. This result is probably due to the fact that the ViT model is not able to exploit the full potential of the multispectral images, as it does not have the same convolutional structure as the ResNet models and lacks some adaptability to the task.

4.4.4 Regression

In this series of experiments, we compared the performance of different models and approaches for regressing future water levels in our dataset. The regression task was performed by predicting the percentage change in water level between the first sample (2017) and the control label (2021). The output of the model is a single value between -1 and 1, representing the percentage change in water level. To evaluate the performance of the model, we used the Mean Absolute Error (MAE) and the R2 value as metrics, with the latter being the main focus of the comparison. As shown in table 4.12, the best results are obtained with the ConvNext model, with an R2 score of 0.834 and an MAE of 0.032. It should also be noted that the ResNet50 model performs slightly worse than the simpler ResNet18 model, with an R2 of 0.789 and an MAE of 0.035. This can be explained by the low level of complexity of the current task, with the larger and more complex model tending to overfit and not being able to generalise to the data presented.

Model	Bands	Tile Size	R2 Score	MAE
ResNet18[48]	MSI	224px	0.834	0.036
ResNet34[48]	MSI	224px	0.794	0.038
ResNet50[48]	MSI	224px	0.789	0.035
ViT[69]	MSI	224px	0.71	0.036
MobileNet [55]	MSI	224px	0.412	0.062
VGG	MSI	224px	0.385	0.065
ConvNext	MSI	224px	**0.839**	**0.032**
SVM-rbf	NDWI	224px	0.437	0.077
SVM-Poly	NDWI	224px	0.055	0.108
Random Forest	NDWI	224px	0.217	0.097

Table 4.12: Results for models comparison for Regression

Bands Comparison for Regression

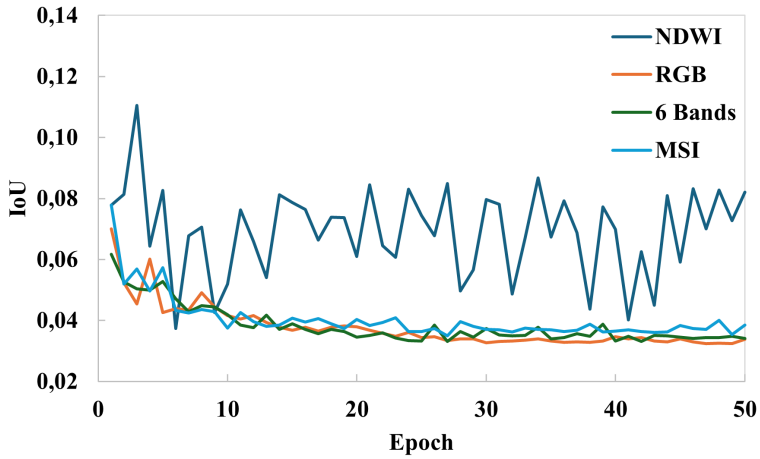
In this experiment, we evaluated the performance of ResNet18 and ConvNext models for water level regression using different combinations of spectral bands. The primary goal was to determine how the choice of bands affects the model's accuracy in predicting water level changes over time. As shown in Table 4.13, both ResNet18 and ConvNext achieved their best R^2 scores when using the 6-band configuration. Specifically, ResNet18 achieved an R^2 score of 0.86 and ConvNext obtained 0.84. In terms of MAE, ConvNext slightly outperformed ResNet18 with the 6-band configuration, achieving an MAE of 0.030, while ResNet18 had 0.033. This indicates that including more spectral information beyond RGB or NDWI bands improves the models' predictive accuracy.

Model	Bands	R2 Score	MAE
ResNet18[48]	NDWI	0.84	0.037
	RGB	0.82	**0.032**
	6B	**0.86**	0.033
	MSI	0.84	0.035
ConvNext[57]	NDWI	0.83	0.038
	RGB	0.73	0.040
	6B	0.82	**0.030**
	MSI	**0.84**	0.032

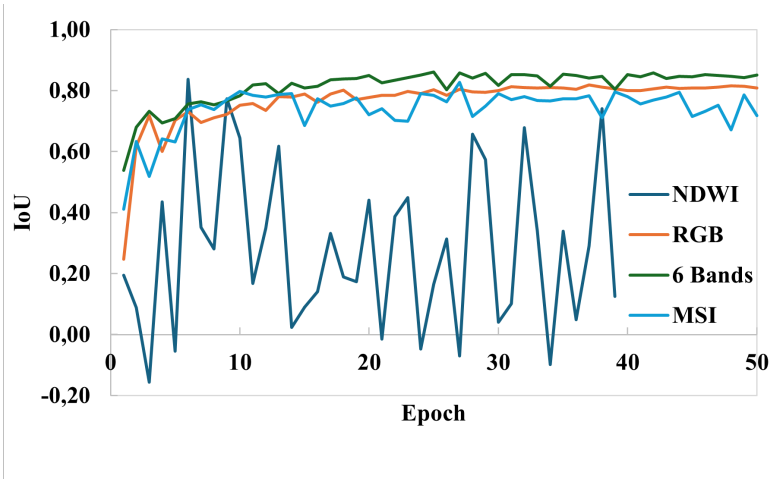
Table 4.13: Results for bands comparison for Water Level Regression

The NDWI-based configuration showed more volatile results, especially visible in the graph comparing Intersection over Union (IoU) across epochs (Figure 4.2b).

The NDWI curve fluctuated significantly, indicating instability during training compared to more stable performances observed for the RGB, 6 Bands, and MSI configurations. This instability suggests that NDWI alone may not be as reliable for regression tasks, likely due to the limited information it captures compared to broader spectral bands.



(a) Mean Absolute Error



(b) R2 Score

Figure 4.2: R2 and MAE during training for bands comparison for Water Level Classification

Overall, the results show that models trained with more comprehensive spectral data (such as 6 Bands and MSI) tend to deliver better and more consistent performance in predicting water level changes, reinforcing the importance of using diverse spectral information for accurate environmental forecasting.

Time Series Analysis for Regression

In this experiment, we investigated how the number of time steps in time series of satellite imagery impacts the performance of two models, ResNet18 and ConvNext, for water level regression. The goal was to determine whether incorporating multiple time steps would lead to improved predictive accuracy, as compared to using only the most recent satellite image. We tested four different configurations for the time series: using only the last image (1 step), using the first and the last images (2 steps), and adding intermediary images for 3 and 4 time steps. As summarized in Table 4.14, the performance of both models improved significantly when more than one image was used. For ResNet18, the best R2 score of 0.86 was achieved with 2 time steps (using the first and last images). However, using 3 and 4 time steps led to slightly lower but still high R2 scores of 0.80 and 0.84, respectively. The lowest MAE, indicating the most accurate predictions, was achieved with 3 time steps, reaching an MAE of 0.031. ConvNext, on the other hand, showed a significant improvement as more time steps were added. The worst performance came when only the last image was used, resulting in an R^2 score of 0.58 and an MAE of 0.057. The performance peaked with 3 time steps, where the model achieved its best R^2 score of 0.85 and its lowest MAE of 0.028.

Model	Time Steps	R2 Score	MAE
ResNet18[48]	1 (only last)	0.72	0.044
	2 (first & last)	**0.86**	0.031
	3	0.80	**0.031**
	4	0.84	0.032
ConvNext[57]	1 (only last)	0.58	0.057
	2 (first & last)	0.84	**0.028**
	3	**0.85**	**0.028**
	4	0.84	0.031

Table 4.14: Comparison of different time series for regression

The graph(figure 4.3) illustrates how the Mean Absolute Error (MAE) behaves as training progresses over 50 epochs. The 1-step configuration consistently performed worse compared to configurations with 2, 3, and 4 steps. Notably, the 2-step model consistently outperformed the others early on, but the difference narrowed after around 30 epochs. By the end of training, all configurations with multiple time steps (2, 3, and 4) performed similarly, with the 1-step configuration trailing behind.

The inclusion of multiple time steps, especially the first and last images, leads to more accurate predictions of water level changes. Both models benefit from temporal information, but ConvNext's improvement was more pronounced with

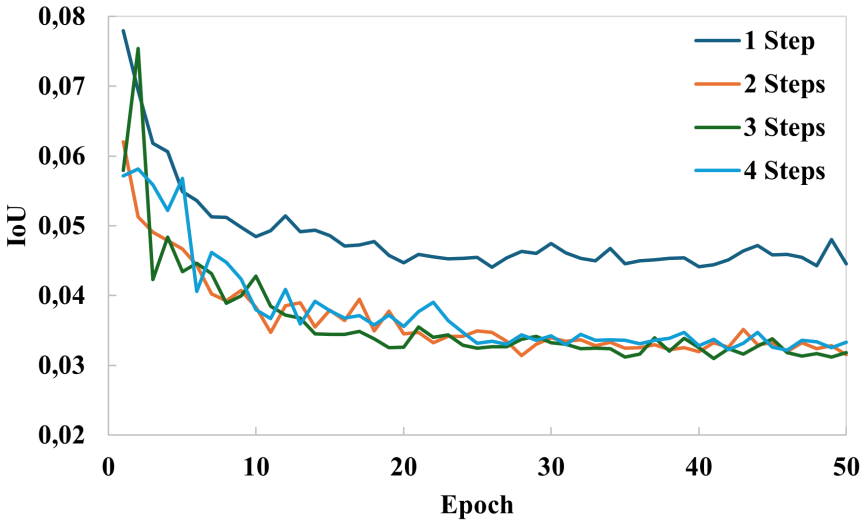


Figure 4.3: Mean Absolute Error (MAE) during training for Time Series Analysis for Regression

more time steps, suggesting that it better leverages temporal dependencies in the data.

4.4.5 Semantic Segmentation

In this last series of experiments, we compared the performance of different models and approaches for the semantic segmentation of water changes in our dataset. This particular task merges together the forecasting of water levels and the recognition of where the change will occur within the image. The output of the model is a mask, where each pixels can be set to

- 0: no change
- 1: positive change
- 2: negative change

To evaluate the performance of the model, we used the Intersection over Union (IoU) as the main metric.

Tile Size Comparison for Semantic Segmentation

In this experiment, we compared the results of training a Unet model with ResNet18 encoder with the best configuration of bands, using different tile sizes, measuring

the IoU. The results, as shown in Table 4.15, show that the best results are obtained using a tile size of 128x128 pixels, with an IoU of 0.624. While the difference in

Model	Bands	Tile Size	IoU
ResNet18	MSI	128px	**0.624**
	MSI	224px	0.613
	MSI	386px	0.592
	MSI	512px	0.605

Table 4.15: Results of the tile size comparison for semantic segmentation

performance between the different tile sizes is not very large, it is worth noting that the larger tile sizes tend to give worse results. This is probably due to the fact that larger tiles contain more information, are more complex and do not allow the model to focus on the important details contained in the image. With 128px and 224px tiles, the model can focus on a smaller area of interest, and the prediction is more accurate also for smaller changes.

Bands Comparison for Semantic Segmentation

In this experiment, we compared the results of training a Unet model with a ResNet18 encoder with different configurations of bands, measuring the IoU. The results, as shown in Table 4.16, show that the best results are obtained using the MSI configuration, with an IoU of 0.613. While this results are not unexpected,

Model	Bands	Tile Size	IoU
ResNet18	NDWI	224px	0.531
	RGB	224px	0.573
	6B	224px	0.605
	MSI	224px	**0.613**

Table 4.16: Results of the bands comparison for semantic segmentation

as more bands contain more information, it is interesting to note that using only the NDWI index as input gives a significantly worse result, with an IoU of 0.531. Another problem with the NDWI is the training instability, as we can see in Fig.4.4 the model tends to alternatively overfit and underfit the data. This is probably due to the fact that the NDWI index, for its nature, captures only information about the presence of water, and tends to hide other image features, such as physical properties of soil, vegetation or atmospheric properties, which are instead well present in the full bands used for the dataset.

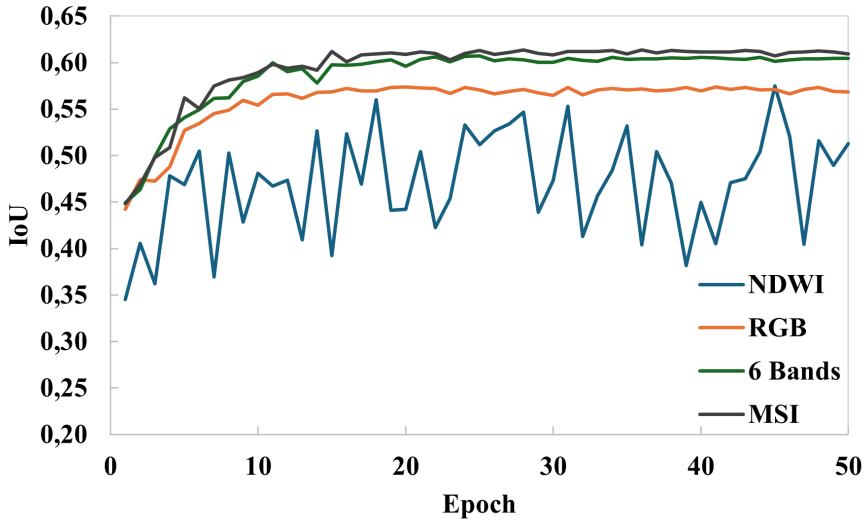


Figure 4.4: Training for the bands comparison for Semantic Segmentation

We can also notice how small the difference in performance is between the 6-band and the MSI configuration, with the latter giving a slightly better result but being way more expensive computationally. This could be due to the fact that some bands in the MSI configuration are not useful for the task and, while introducing some more information and helping the model to generalize better by introducing noise, they do not significantly improve the performance of the model. For a future work, it could be interesting to investigate which bands are the most effective for the task and analyse how the model uses them to make the prediction.

Time Series Analysis for Semantic Segmentation

In this experiment, we compared the performance of a U-Net model for semantic segmentation using various time series of satellite images as input. The primary goal was to assess how effectively the model could exploit temporal information when trained with different encoders, specifically ResNet18 and ResNet50. Both experiments were conducted using the best configuration of spectral bands and patch sizes, and the results are summarized in Tables 4.17 and 4.18. The choice of both ResNet18 and ResNet50 as encoders was made in order to compare the impact of encoder complexity on the model's ability to exploit the temporal information. ResNet18 is a lightweight model with fewer layers, while ResNet50 is significantly deeper, with more layers and parameters, allowing for a more complex feature extraction process. By comparing these two architectures, we aim to understand how increasing encoder complexity influences segmentation accuracy, especially when multiple time steps are introduced.

Model	Encoder	Bands	Time Steps	IoU
UNet	ResNet18	MSI	1 (only last)	0.536
		MSI	2 (first & last)	0.599
		MSI	3	0.601
		MSI	4	**0.614**

Table 4.17: Results of the time series analysis for semantic segmentation with ResNet18

From the results obtained with ResNet18 (Table 4.17) we observe that, as expected, the best performance is achieved when the model is provided with four images of the same tile taken at different times, with an Intersection over Union (IoU) score of 0.614. The results reveal a noticeable improvement in performance when increasing from one to two time steps, with a more marginal improvement as the number of time steps increases further. Specifically, the IoU score improves from 0.536 with a single time step to 0.599 with two time steps, with smaller gains observed when moving to three (0.601) and four (0.614) time steps.

Model	Encoder	Bands	Time Steps	IoU
UNet	ResNet50	MSI	1 (only last)	0.538
		MSI	2 (first & last)	0.627
		MSI	3	0.638
		MSI	4	**0.651**

Table 4.18: Results of the time series analysis for semantic segmentation with ResNet50

Looking at the results obtained with ResNet50 (Table 4.18), a similar trend can be observed, with the best results also obtained using four time steps, where the IoU reaches 0.651. Interestingly, ResNet50 shows a more pronounced improvement with each additional time step, especially between one and two time steps, where the IoU increases from 0.538 to 0.627. The performance continues to improve with three (0.638) and four (0.651) time steps, suggesting that ResNet50 can make more effective use of the temporal information provided by the time series.

Comparing the results of the two encoders, it is clear that the deeper and more complex ResNet50 consistently outperforms ResNet18 in exploiting temporal information. Not only does ResNet50 achieve better overall performance, but it also shows a more consistent increase in IoU as additional time steps are introduced. The increased depth and capacity of ResNet50 allows it to capture more detailed temporal dynamics and spatial features across the time series, resulting in improved segmentation accuracy.

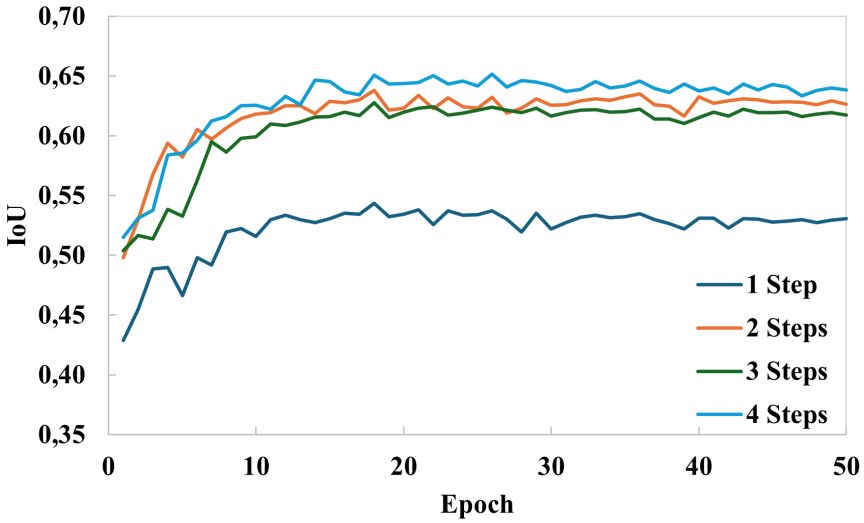


Figure 4.5: Time Series Analysis for Semantic Segmentation

Encoders Comparison

In this experiment, we compared the performance of U-Net, PSPNet, and DeepLabV3 architectures using 6 different encoders: ResNet18, ResNet34, ResNet50, MobileNetV3, VGG and ViT.

The goal was to assess the effectiveness of each model in semantic segmentation tasks, specifically for water detection. To ensure a fair comparison, we used the same dataset, hyperparameters, and configuration across all experiments.

The models were evaluated based on their Intersection over Union (IoU) scores and the total time required for training. For consistency, all models were trained with a tile size of 224x224 pixels. This choice was influenced by the fact that ViT (Vision Transformer) was pretrained on ImageNet using this image size and does not support other input sizes without reconfiguration

Model	Encoder	Tiles	IoU	T. Time
U-Net	ResNet18	224px	61.0	**16.91 min**
	ResNet34	224px	62.3	18.19 min
	ResNet50	224px	**64.8**	34.32 min
	VGG-16	224px	60.8	24.09 min
	ViT	224px	62.9	22.66 min
	MobileNetV3	224px	58.8	18.01 min

Table 4.19: Results of the encoders comparison for U-Net

In Table 4.19, we present the results for the U-Net architecture with various encoders. The best result was achieved with the ResNet50 encoder, yielding an IoU of 64.8%. While ResNet50 provided the highest accuracy, it also required the longest training time (34.32 minutes). On the other hand, MobileNetV3, while being the fastest to train, had the lowest IoU at 58.8%, which is understandable given that it has far fewer parameters (only 2.97M) compared to the other models.

Model	Encoder	Tiles	IoU	Time
PSPNet	ResNet18	224px	59.7	**14.80 min**
	ResNet34	224px	59.3	15.00 min
	ResNet50	224px	59.1	15.89 min
	VGG-16	224px	**62.5**	22.24 min
	ViT	224px	61.7	22.11 min
	MobileNetV3	224px	55.6	14.90 min

Table 4.20: Results of the encoders comparison for PSPNet

Table 4.20 shows the results for the PSPNet architecture. Interestingly, the best performing encoder here was VGG-16, which achieved an IoU of 62.5% with a training time of 22.24 minutes. ViT also performed well, with an IoU of 61.7%, though it required a similar training time. The ResNet50 encoder, which performed best for U-Net, did not fare as well here, with an IoU of 59.1%. In general, all three ResNet-based encoders delivered similar unremarlable performances, with IoU scores ranging from 59.1% to 59.7%. MobileNetV3 again had the fastest training time but delivered the lowest IoU of 55.6%, further emphasizing its trade-off between speed and performance. Finally, the results for DeepLabV3 are shown in

Model	Encoder	Tiles	IoU	Time
DeepLab	ResNet18	224px	58.6	**34.32 min**
	ResNet34	224px	**62.1**	42.48 min
	ResNet50	224px	60.2	1.13 hr
	MobileNetV3	224px	54.4	30.04 min

Table 4.21: Results of the encoders comparison for DeepLabV3

Table 4.21. The best performing encoder was ResNet34, with an IoU of 62.1% and a training time of 42.48 minutes. Although DeepLabV3 generally performed well, it was the slowest architecture to train, particularly with the ResNet50 encoder, which took over an hour. MobileNetV3, while fast to train, again lagged behind in performance, with an IoU of 54.4%.

To summarize, as shown in Table 4.22, the ResNet50 encoder produced the highest IoU of 64.8% when used with U-Net, but this came at the cost of longer

training times. The VGG-16 encoder also performed well with PSPNet, achieving a comparable IoU of 62.5% in a shorter training time. While MobileNetV3 was the fastest to train across all architectures, it consistently had the lowest IoU scores, highlighting the trade-off between model complexity, performance, and training speed.

Model	Encoder	Bands	Tile Size	IoU	Time
UNet	ResNet50	MSI	224px	**64.8**	32.6 min
DeepLabV3	ResNet34	MSI	224px	62.1	42.48 min
PSPNet	VGG-16	MSI	224px	62.5	**22.24 min**

Table 4.22: Models comparison for Semantic Segmentation

In conclusion, while U-Net with a ResNet50 encoder achieved the best results in terms of IoU, other architectures like PSPNet with VGG-16 offer a good balance between performance and training time. The results indicate that the choice of encoder can significantly influence the model's accuracy and efficiency, making it essential to consider both factors when selecting the best model for semantic segmentation tasks.

4.5 Discussion

In summary, deep learning methods significantly outperform classical and shallow learning approaches for segmentation tasks. Increasing the amount of data, especially temporal and multi-band data, generally improves model performance but also raises complexity. Simpler tasks like regression see minimal benefits from additional data. While complex models such as U-Net and ResNet50 excel in accuracy, simpler models like PSP-Net and MobileNet can deliver comparable results with far less computational effort, though they may underperform in more demanding cases. Predicting changes in lake and river levels requires robust models and large, diverse datasets.

Water Segmentation In the experiments comparing various models for water segmentation, the U-Net with a ResNet50 encoder achieved the best performance with an IoU of 92.7%, outperforming other architectures like PSPNet and DeepLabV3. While ResNet50 provided the highest accuracy, lighter encoders such as MobileNetV3 showed competitive performance with much faster training times. Classical methods like NDWI stayed significantly behind, with an IoU of only 83.4%, emphasising even more the superior accuracy of deep learning approaches for water detection.

Water level forecasting classification In this experiment, in addition to seeking the best performance for deep learning models, a comparison was also made with classical machine learning algorithms such as SVM and Random Forest, confirming their superiority. In fact, CNN's best model for this task, ResNet50, achieves an F1 score of 90.6%, while SVM and RF stand at 60.8% and 66.7%. Additionally, using extensive temporal information and multiple bands improved performance, reaching an F1 score of 89.9% instead of a 85.8% obtained with only the last available data and 84.8% obtained with a single input channel. While ViT showed slightly lower results than ResNet models, it still outperformed MobileNet and traditional methods.

Water level forecasting regression In this experiment we focus on predicting water level changes between 2017 and 2021 using satellite images. The ConvNext model achieved the best results, with an R2 score of 0.839 and a Mean Absolute Error (MAE) of 0.032, showing strong performance in water level prediction. Classical methods like SVM and Random Forest performed much worse compared to deep learning models, with R^2 scores below 0.5 and high MAE values, showing that deep learning is far more effective for this task. Also in this case more data and more temporal information improved the performance of the model, but differently from the classification and segmentation tasks, the best results were obtained with 6 bands instead of the full MSI configuration. This is probably due to the easier nature of the task that risks overfitting with too much data.

Water level forecasting segmentation The experiments compared different models and configurations for semantic segmentation of water changes, using the IoU metric for evaluation. The U-Net model with a ResNet50 encoder achieved the best performance with an IoU of 64.8%, though it required longer training times. On the other hand, PSPNet with a VGG-16 encoder offered a strong balance between accuracy (62.5% IoU) and speed. Smaller tile sizes (128x128 pixels) and multiple temporal images improved performance, with four time steps yielding the best results. Being a rather complex task, semantic segmentation of water changes benefits from deep, complex models and temporal data, which provide valuable insights into the dynamics of water bodies over time.

Chapter 5

Conclusion

In this thesis, we explored the potential of deep learning models for detecting water bodies in satellite images and forecasting water levels in lakes. Our findings confirm that deep learning models represent the current state of the art in remote sensing image analysis, significantly outperforming machine learning algorithms like Random Forest and SVM, as well as traditional methods such as water indices.

By comparing some of the most widely used deep learning models, we gained insights into their performance across different tasks, as well as the impact of various data types on their accuracy. While the use of deep learning for water segmentation has been extensively studied, and our work primarily serves to confirm and compare existing methods, the same cannot be said for the short-term prediction of water level changes in inland water bodies. In this area, reliable predictive methods are still in their early stages.

We hope that the results presented in this thesis contribute to the advancement of this critical field, which has significant implications for human life, settlement planning, and agricultural management. In future work, we aim to enhance model accuracy by expanding the dataset and incorporating additional data sources, such as meteorological data, which could provide deeper insights into the causes of water level fluctuations.

Bibliography

[1] Charles Verpoorter, Tiit Kutser, David A. Seekell, and Lars J. Tranvik. «A global inventory of lakes based on high-resolution satellite imagery». In: *Geophysical Research Letters* 41.18 (2014), pp. 6396–6402. DOI: https://doi.org/10.1002/2014GL060641. eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1002/2014GL060641. URL: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2014GL060641 (cit. on pp. 1, 31).

[2] Jean-François Pekel, Andrew Cottam, Noel Gorelick, and Alan S. Belward. «High-resolution mapping of global surface water and its long-term changes». In: *Nature* 540.7633 (Dec. 2016), pp. 418–422. ISSN: 1476-4687. DOI: 10.1038/nature20584. URL: https://doi.org/10.1038/nature20584 (cit. on pp. 1, 31).

[3] Jun Yang, Peng Gong, Rong Fu, Minghua Zhang, Jingming Chen, Shunlin Liang, Bing Xu, Jiancheng Shi, and Robert Dickinson. «The role of satellite remote sensing in climate change studies». In: *Nature climate change* 3.10 (2013), pp. 875–883 (cit. on pp. 1, 16, 17).

[4] S. K. McFEETERS. «The use of the Normalized Difference Water Index (NDWI) in the delineation of open water features». In: *International Journal of Remote Sensing* 17.7 (1996), pp. 1425–1432. DOI: 10.1080/01431169608948714. eprint: https://doi.org/10.1080/01431169608948714. URL: https://doi.org/10.1080/01431169608948714 (cit. on pp. 1, 17–19).

[5] TV Bijeesh and KN Narasimhamurthy. «Surface water detection and delineation using remote sensing images: A review of methods and algorithms». In: *Sustainable Water Resources Management* 6.4 (2020), p. 68 (cit. on pp. 1, 17–19).

[6] Furkan Isikdogan, Alan C Bovik, and Paola Passalacqua. «Surface water mapping by deep learning». In: *IEEE journal of selected topics in applied earth observations and remote sensing* 10.11 (2017), pp. 4909–4918 (cit. on pp. 1, 16).

[7] TS Akiyama, J Marcato Junior, WN Gonçalves, PO Bressan, A Eltner, F Binder, and T Singer. «Deep learning applied to water segmentation». In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 43 (2020), pp. 1189–1193 (cit. on pp. 1, 16).

[8] Bingxin Bai, Lixia Mu, Chunyong Ma, Ge Chen, and Yumin Tan. «Extreme water level changes in global lakes revealed by altimetry satellites since the 2000s». In: *International Journal of Applied Earth Observation and Geoinformation* 127 (2024), p. 103694. ISSN: 1569-8432. DOI: https://doi.org/10.1016/j.jag.2024.103694. URL: https://www.sciencedirect.com/science/article/pii/S1569843224000487 (cit. on p. 2).

[9] David Rolnick et al. «Tackling climate change with machine learning». In: *ACM Computing Surveys (CSUR)* 55.2 (2022), pp. 1–96 (cit. on pp. 2, 4, 7, 16, 17, 23).

[10] Mohammad Sajjad Khan and Paulin Coulibaly. «Application of support vector machine in lake water level prediction». In: *Journal of Hydrologic Engineering* 11.3 (2006), pp. 199–205 (cit. on p. 2).

[11] Tamar Zohary and Ilia Ostrovsky. «Ecological impacts of excessive water level fluctuations in stratified freshwater lakes». In: *Inland waters* 1.1 (2011), pp. 47–59 (cit. on p. 2).

[12] Holger R Maier and Graeme C Dandy. «Neural networks for the prediction and forecasting of water resources variables: a review of modelling issues and applications». In: *Environmental modelling & software* 15.1 (2000), pp. 101–124 (cit. on p. 2).

[13] Kumar Puran Tripathy and Ashok Kumar Mishra. «Deep learning in hydrology and water resources disciplines: Concepts, methods, applications, and research directions». In: *Journal of Hydrology* (2023), p. 130458 (cit. on pp. 2, 16).

[14] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Nature, 2022 (cit. on pp. 4, 6).

[15] Lili Li, Shujuan Zhang, and Bin Wang. «Plant disease detection and classification by deep learning—a review». In: *IEEE Access* 9 (2021), pp. 56683–56698 (cit. on p. 4).

[16] Andreas Kamilaris and Francesc X Prenafeta-Boldú. «Deep learning in agriculture: A survey». In: *Computers and electronics in agriculture* 147 (2018), pp. 70–90 (cit. on p. 4).

[17] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. «U-net: Convolutional networks for biomedical image segmentation». In: *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*. Springer. 2015, pp. 234–241 (cit. on pp. 4, 5, 7, 11–13).

[18] Qing Li, Weidong Cai, Xiaogang Wang, Yun Zhou, David Dagan Feng, and Mei Chen. «Medical image classification with convolutional neural network». In: *2014 13th international conference on control automation robotics & vision (ICARCV)*. IEEE. 2014, pp. 844–848 (cit. on pp. 4, 7).

[19] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. «The cityscapes dataset for semantic urban scene understanding». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 3213–3223 (cit. on pp. 4–6, 24).

[20] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. «A survey of autonomous driving: Common practices and emerging technologies». In: *IEEE access* 8 (2020), pp. 58443–58469 (cit. on pp. 4, 7, 11).

[21] Zhonghe Ren, Fengzhou Fang, Ning Yan, and You Wu. «State of the art in defect detection based on machine vision». In: *International Journal of Precision Engineering and Manufacturing-Green Technology* 9.2 (2022), pp. 661–691 (cit. on p. 4).

[22] Yuval Nirkin, Iacopo Masi, Anh Tran Tuan, Tal Hassner, and Gerard Medioni. «On face segmentation, face swapping, and face perception». In: *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*. IEEE. 2018, pp. 98–105 (cit. on p. 4).

[23] Edeh Michael Onyema, Piyush Kumar Shukla, Surjeet Dalal, Mayuri Neeraj Mathur, Mohammed Zakariah, and Basant Tiwari. «Enhancement of patient facial recognition through deep learning algorithm: ConvNet». In: *Journal of Healthcare Engineering* 2021.1 (2021), p. 5196000 (cit. on p. 4).

[24] Laith Alzubaidi et al. «Review of deep learning: concepts, CNN architectures, challenges, applications, future directions». In: *Journal of big Data* 8 (2021), pp. 1–74 (cit. on pp. 4–7, 9, 10, 22).

[25] Olga Russakovsky et al. «Imagenet large scale visual recognition challenge». In: *International journal of computer vision* 115 (2015), pp. 211–252 (cit. on pp. 4, 6, 7, 22, 24).

[26] Peter Meer, Doron Mintz, Azriel Rosenfeld, and Dong Yoon Kim. «Robust regression methods for computer vision: A review». In: *International journal of computer vision* 6 (1991), pp. 59–70 (cit. on p. 4).

[27] Zhengxia Zou, Keyan Chen, Zhenwei Shi, Yuhong Guo, and Jieping Ye. «Object detection in 20 years: A survey». In: *Proceedings of the IEEE* 111.3 (2023), pp. 257–276 (cit. on pp. 4, 7).

[28] Shijie Hao, Yuan Zhou, and Yanrong Guo. «A brief survey on semantic segmentation with deep learning». In: *Neurocomputing* 406 (2020), pp. 302–321 (cit. on pp. 5, 10–12, 24).

[29] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. «Yolact: Real-time instance segmentation». In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 9157–9166 (cit. on p. 5).

[30] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár. «Panoptic segmentation». In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 9404–9413 (cit. on p. 5).

[31] Xiaoqiang Lu, Binqiang Wang, Xiangtao Zheng, and Xuelong Li. «Exploring models and data for remote sensing image caption generation». In: *IEEE Transactions on Geoscience and Remote Sensing* 56.4 (2017), pp. 2183–2195 (cit. on p. 5).

[32] Erik Murphy-Chutorian and Mohan Manubhai Trivedi. «Head pose estimation in computer vision: A survey». In: *IEEE transactions on pattern analysis and machine intelligence* 31.4 (2008), pp. 607–626 (cit. on p. 5).

[33] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. «High-resolution image synthesis with latent diffusion models». In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 10684–10695 (cit. on pp. 5, 6).

[34] Tim Brooks et al. «Video generation models as world simulators». In: (2024). URL: `https://openai.com/research/video-generation-models-as-world-simulators` (cit. on p. 5).

[35] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. «A review of image denoising algorithms, with a new one». In: *Multiscale modeling & simulation* 4.2 (2005), pp. 490–530 (cit. on p. 5).

[36] Chunwei Tian, Lunke Fei, Wenxian Zheng, Yong Xu, Wangmeng Zuo, and Chia-Wen Lin. «Deep learning on image denoising: An overview». In: *Neural Networks* 131 (2020), pp. 251–275 (cit. on p. 5).

[37] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. «Deep learning». In: *nature* 521.7553 (2015), pp. 436–444 (cit. on pp. 5–8, 22).

[38] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. «Imagenet classification with deep convolutional neural networks». In: *Advances in neural information processing systems* 25 (2012) (cit. on pp. 5, 7).

[39] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. «Learning representations by back-propagating errors». In: *nature* 323.6088 (1986), pp. 533–536 (cit. on p. 5).

[40] Sebastian Ruder. «An overview of gradient descent optimization algorithms». In: *arXiv preprint arXiv:1609.04747* (2016) (cit. on p. 5).

[41] Keiron O'Shea and Ryan Nash. *An Introduction to Convolutional Neural Networks*. 2015. arXiv: 1511.08458 [cs.NE]. URL: https://arxiv.org/abs/1511.08458 (cit. on pp. 7, 8).

[42] Hansheng Ren et al. «Time-series anomaly detection service at microsoft». In: *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2019, pp. 3009–3017 (cit. on p. 7).

[43] Avraam Tsantekidis, Nikolaos Passalis, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. «Forecasting stock prices from the limit order book using convolutional neural networks». In: *2017 IEEE 19th conference on business informatics (CBI)*. Vol. 1. IEEE. 2017, pp. 7–12 (cit. on p. 7).

[44] Diksha Khurana, Aditya Koli, Kiran Khatter, and Sukhdev Singh. «Natural language processing: state of the art, current trends and challenges». In: *Multimedia tools and applications* 82.3 (2023), pp. 3713–3744 (cit. on pp. 7, 14, 22).

[45] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. «Gradient-based learning applied to document recognition». In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791 (cit. on p. 7).

[46] Varsha Kishore and Justin Lovelace. *CS 4782: Machine Learning for Data Science*. https://www.cs.cornell.edu/courses/cs4782/2024sp/lectures/pdfs/week_2_0.pdf. 2024 (cit. on p. 8).

[47] Zhiwu Shang, Jie Zhang, Wanxiang Li, Shiqi Qian, and Maosheng Gao. «A domain adversarial transfer model with inception and attention network for rolling bearing fault diagnosis under variable operating conditions». In: *Journal of Vibration Engineering & Technologies* 12.1 (2024), pp. 1–17 (cit. on p. 9).

[48] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV]. URL: https://arxiv.org/abs/1512.03385 (cit. on pp. 9–11, 44, 46, 47, 49).

[49] Fengxiang He, Tongliang Liu, and Dacheng Tao. «Why resnet works? residuals generalize». In: *IEEE transactions on neural networks and learning systems* 31.12 (2020), pp. 5349–5362 (cit. on p. 9).

[50] Ying Li, Haokui Zhang, Xizhe Xue, Yenan Jiang, and Qiang Shen. «Deep learning for remote sensing image classification: A survey». In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 8.6 (2018), e1264 (cit. on pp. 9, 23, 24).

[51] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. «A convnet for the 2020s». In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 11976–11986 (cit. on pp. 9, 10, 37, 46).

[52] Andrew G Howard. «MobileNets: Efficient convolutional neural networks for mobile vision applications». In: *arXiv preprint arXiv:1704.04861* (2017) (cit. on p. 10).

[53] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. «Mobilenetv2: Inverted residuals and linear bottlenecks». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4510–4520 (cit. on p. 10).

[54] Andrew Howard et al. «Searching for mobilenetv3». In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 1314–1324 (cit. on p. 10).

[55] Danfeng Qin et al. *MobileNetV4 – Universal Models for the Mobile Ecosystem*. 2024. arXiv: `2404.10518 [cs.CV]`. URL: `https://arxiv.org/abs/2404.10518` (cit. on pp. 10, 46, 47).

[56] Karen Simonyan and Andrew Zisserman. «Very deep convolutional networks for large-scale image recognition». In: *arXiv preprint arXiv:1409.1556* (2014) (cit. on p. 10).

[57] Sanghyun Woo, Shoubhik Debnath, Ronghang Hu, Xinlei Chen, Zhuang Liu, In So Kweon, and Saining Xie. «Convnext v2: Co-designing and scaling convnets with masked autoencoders». In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 16133–16142 (cit. on pp. 10, 47, 49).

[58] Sidike Paheding, Ashraf Saleem, Mohammad Faridul Haque Siddiqui, Nathir Rawashdeh, Almabrok Essa, and Abel A Reyes. «Advancing horizons in remote sensing: a comprehensive survey of deep learning models and applications in image classification and beyond». In: *Neural Computing and Applications* (2024), pp. 1–41 (cit. on pp. 11, 22–24).

[59] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. «Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs». In: *IEEE transactions on pattern analysis and machine intelligence* 40.4 (2017), pp. 834–848 (cit. on pp. 12, 13).

[60] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. «Pyramid scene parsing network». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2881–2890 (cit. on pp. 12, 14).

[61] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. «Segnet: A deep convolutional encoder-decoder architecture for image segmentation». In: *IEEE transactions on pattern analysis and machine intelligence* 39.12 (2017), pp. 2481–2495 (cit. on p. 12).

[62] Jonathan Ho, Ajay Jain, and Pieter Abbeel. «Denoising diffusion probabilistic models». In: *Advances in neural information processing systems* 33 (2020), pp. 6840–6851 (cit. on p. 12).

[63] Isaac Corley, Caleb Robinson, and Anthony Ortiz. «A Change Detection Reality Check». In: *arXiv preprint arXiv:2402.06994* (2024) (cit. on p. 12).

[64] A Vaswani. «Attention is all you need». In: *Advances in Neural Information Processing Systems* (2017) (cit. on pp. 14, 15).

[65] Hugo Touvron et al. «Llama: Open and efficient foundation language models». In: *arXiv preprint arXiv:2302.13971* (2023) (cit. on p. 14).

[66] Abhimanyu Dubey et al. «The llama 3 herd of models». In: *arXiv preprint arXiv:2407.21783* (2024) (cit. on p. 14).

[67] Alexey Dosovitskiy. «An image is worth 16x16 words: Transformers for image recognition at scale». In: *arXiv preprint arXiv:2010.11929* (2020) (cit. on pp. 14–16, 37).

[68] Asifullah Khan, Zunaira Rauf, Anabia Sohail, Abdul Rehman Khan, Hifsa Asif, Aqsa Asif, and Umair Farooq. «A survey of the vision transformers and their CNN-transformer based variants». In: *Artificial Intelligence Review* 56.Suppl 3 (2023), pp. 2917–2970 (cit. on pp. 14, 15).

[69] Benjamin Graham, Alaaeldin El-Nouby, Hugo Touvron, Pierre Stock, Armand Joulin, Hervé Jégou, and Matthijs Douze. «Levit: a vision transformer in convnet's clothing for faster inference». In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 12259–12269 (cit. on pp. 15, 37, 46, 47).

[70] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. «Swin transformer: Hierarchical vision transformer using shifted windows». In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 10012–10022 (cit. on p. 15).

[71] Zhengzhong Tu, Hossein Talebi, Han Zhang, Feng Yang, Peyman Milanfar, Alan Bovik, and Yinxiao Li. «Maxvit: Multi-axis vision transformer». In: *European conference on computer vision*. Springer. 2022, pp. 459–479 (cit. on p. 15).

[72] James B Campbell and Randolph H Wynne. *Introduction to remote sensing*. Guilford press, 2011 (cit. on pp. 16, 17, 20, 24).

[73] Wenli Huang et al. Xiao-Peng Song. «An evaluation of Landsat, Sentinel-2, Sentinel-1 and MODIS data for crop type mapping». In: *Science of Remote Sensing* 3 (2021), p. 100018. ISSN: 2666-0172. DOI: https://doi.org/10.1016/j.srs.2021.100018. URL: https://www.sciencedirect.com/science/article/pii/S2666017221000055 (cit. on p. 16).

[74] Quy-Toan Do, Jacob N Shapiro, Christopher D Elvidge, Mohamed Abdel-Jelil, Daniel P Ahn, Kimberly Baugh, Jamie Hansen-Lewis, Mikhail Zhizhin, and Morgan D Bazilian. «Terrorism, geopolitics, and oil security: Using remote sensing to estimate oil production of the Islamic State». In: *Energy research & social science* 44 (2018), pp. 411–418 (cit. on p. 16).

[75] Yichun Xie, Zongyao Sha, and Mei Yu. «Remote sensing imagery in vegetation mapping: a review». In: *Journal of plant ecology* 1.1 (2008), pp. 9–23 (cit. on p. 16).

[76] John Rogan and DongMei Chen. «Remote sensing technology for mapping and monitoring land-cover and land-use change». In: *Progress in planning* 61.4 (2004), pp. 301–325 (cit. on p. 16).

[77] Xuehui Pi et al. «Mapping global lake dynamics reveals the emerging roles of small lakes». In: *Nature Communications* 13.1 (Oct. 2022), p. 5777. ISSN: 2041-1723. DOI: 10.1038/s41467-022-33239-3. URL: https://doi.org/10.1038/s41467-022-33239-3 (cit. on p. 16).

[78] Junjie Li, Yizhuo Meng, Yuanxi Li, Qian Cui, Xining Yang, Chongxin Tao, Zhe Wang, Linyi Li, and Wen Zhang. «Accurate water extraction using remote sensing imagery based on normalized difference water index and unsupervised deep learning». In: *Journal of Hydrology* 612 (2022), p. 128202 (cit. on pp. 16, 24).

[79] John E Ball, Derek T Anderson, and Chee Seng Chan. «Comprehensive survey of deep learning in remote sensing: theories, tools, and challenges for the community». In: *Journal of applied remote sensing* 11.4 (2017), pp. 042609–042609 (cit. on pp. 17, 24).

[80] Wikipedia contributors. *Copernicus Programme — Wikipedia, The Free Encyclopedia*. [Online; accessed 28-September-2024]. 2024. URL: https://en.wikipedia.org/w/index.php?title=Copernicus_Programme&oldid=1244787427 (cit. on pp. 17, 21, 22).

[81] Helder I Chaminé, Alcides JSC Pereira, Ana C Teodoro, and José Teixeira. *Remote sensing and GIS applications in earth and environmental systems sciences*. 2021 (cit. on p. 17).

[82] Daniele Rege Cambrin, Luca Colomba, and Paolo Garza. «CaBuAr: California Burned Areas dataset for delineation». In: *arXiv preprint arXiv:2401.11519* (2024) (cit. on pp. 17, 24).

[83] European Space Agency. *Overview of Sentinel-2 Mission*. https://sentiwiki.copernicus.eu/web/s2-mission. Accessed: (26-08-2024). 2024 (cit. on pp. 18, 21, 23).

[84] European Space Agency. *Sentinel-2 Data Sheet*. https://esamultimedia.esa.int/docs/S2-Data_Sheet.pdf. Accessed: (26-08-2024). 2024 (cit. on pp. 18, 21).

[85] Frank J KRIEGLER. «Preprocessing transformations and their effects on multspectral recognition». In: *Proceedings of the Sixth International Symposium on Remote Sesning of Environment*. 1969, pp. 97–131 (cit. on p. 18).

[86] Hanqiu Xu. «Modification of normalised difference water index (NDWI) to enhance open water features in remotely sensed imagery». In: *International Journal of Remote Sensing* 27.14 (2006), pp. 3025–3033. DOI: 10.1080/01431160600589179. eprint: https://doi.org/10.1080/01431160600589179. URL: https://doi.org/10.1080/01431160600589179 (cit. on pp. 19, 21).

[87] OpenEO. *OpenEO*. https://openeo.cloud/. 2024 (cit. on p. 21).

[88] Volker C. Radeloff et al. «Need and vision for global medium-resolution Landsat and Sentinel-2 data products». In: *Remote Sensing of Environment* 300 (2024), p. 113918. ISSN: 0034-4257. DOI: https://doi.org/10.1016/j.rse.2023.113918. URL: https://www.sciencedirect.com/science/article/pii/S0034425723004704 (cit. on pp. 21, 24).

[89] Xiao Xiang Zhu, Devis Tuia, Lichao Mou, Gui-Song Xia, Liangpei Zhang, Feng Xu, and Friedrich Fraundorfer. «Deep learning in remote sensing: A comprehensive review and list of resources». In: *IEEE geoscience and remote sensing magazine* 5.4 (2017), pp. 8–36 (cit. on p. 22).

[90] Jia Song, Shaohua Gao, Yunqiang Zhu, and Chenyan Ma. «A survey of remote sensing image classification based on CNNs». In: *Big earth data* 3.3 (2019), pp. 232–254 (cit. on p. 22).

[91] Lei Ma, Yu Liu, Xueliang Zhang, Yuanxin Ye, Gaofei Yin, and Brian Alan Johnson. «Deep learning in remote sensing applications: A meta-analysis and review». In: *ISPRS journal of photogrammetry and remote sensing* 152 (2019), pp. 166–177 (cit. on p. 22).

[92] Shutao Li, Weiwei Song, Leyuan Fang, Yushi Chen, Pedram Ghamisi, and Jon Atli Benediktsson. «Deep learning for hyperspectral image classification: An overview». In: *IEEE Transactions on Geoscience and Remote Sensing* 57.9 (2019), pp. 6690–6709 (cit. on p. 22).

[93] Alexandre Lacoste et al. «Toward foundation models for earth monitoring: Proposal for a climate change benchmark». In: *arXiv preprint arXiv:2112.00570* (2021) (cit. on p. 24).

[94] Yi Wang, Nassim Ait Ali Braham, Zhitong Xiong, Chenying Liu, Conrad M Albrecht, and Xiao Xiang Zhu. «SSL4EO-S12: A large-scale multimodal, multitemporal dataset for self-supervised learning in Earth observation [Software and Data Sets]». In: *IEEE Geoscience and Remote Sensing Magazine* 11.3 (2023), pp. 98–106 (cit. on pp. 24, 25).

[95] Nikolaos Dionelis, Casper Fibaek, Luke Camilleri, Andreas Luyts, Jente Bosmans, and Bertrand Le Saux. «Evaluating and Benchmarking Foundation Models for Earth Observation and Geospatial AI». In: *arXiv preprint arXiv:2406.18295* (2024) (cit. on p. 24).

[96] Aozhe Dou, Yang Hao, Weifeng Liu, Liangliang Li, Zhenzhong Wang, and Baodi Liu. «Remote sensing image cloud removal based on multi-scale spatial information perception». In: *Multimedia Systems* 30.5 (2024), p. 249 (cit. on p. 24).

[97] Steven Euijong Whang, Yuji Roh, Hwanjun Song, and Jae-Gil Lee. «Data collection and quality challenges in deep learning: A data-centric ai perspective». In: *The VLDB Journal* 32.4 (2023), pp. 791–813 (cit. on p. 24).

[98] Yuji Roh, Geon Heo, and Steven Euijong Whang. «A survey on data collection for machine learning: a big data-ai integration perspective». In: *IEEE Transactions on Knowledge and Data Engineering* 33.4 (2019), pp. 1328–1347 (cit. on p. 24).

[99] Copernicus Open Access Hub. *Sentinel-2 Bands*. `https://custom-scripts.sentinel-hub.com/custom-scripts/sentinel-2/bands/`. 2021 (cit. on p. 31).

[100] European Space Agency. *Sentinel-2 Clouds Classification*. = https://custom-scripts.sentinel-hub.com/custom-scripts/sentinel-2/scene-classification/, Accessed: (26-08-2024). 2024 (cit. on p. 30).

[101] Saurabh Channan Min Feng Joseph O. Sexton and John R. Townshend. «A global, high-resolution (30-m) inland water body dataset for 2000: first results of a topographic-spectral classification algorithm». In: *International Journal of Digital Earth* 9.2 (2016), pp. 113–133. DOI: `10.1080/17538947.2015.1026420`. eprint: `https://doi.org/10.1080/17538947.2015.1026420`. URL: `https://doi.org/10.1080/17538947.2015.1026420` (cit. on p. 31).

[102] Defense Mapping Agency. *Department of Defense World Geodetic System 1984*. https://apps.dtic.mil/sti/pdfs/ADA280358.pdf. 1991 (cit. on p. 32).

[103] Pavel Iakubovskii. *Segmentation Models Pytorch*. https://github.com/qubvel/segmentation_models.pytorch. 2019 (cit. on pp. 38, 39).

[104] Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, and Animesh et al. Jain. «PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation». In: *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*. ACM, Apr. 2024. DOI: 10.1145/3620665.3640366. URL: https://pytorch.org/assets/pytorch2-2.pdf (cit. on p. 38).

[105] William Falcon and The PyTorch Lightning team. *PyTorch Lightning*. Version 1.4. Mar. 2019. DOI: 10.5281/zenodo.3828935. URL: https://github.com/Lightning-AI/lightning (cit. on p. 38).

[106] Ross Wightman. *PyTorch Image Models*. https://github.com/huggingface/pytorch-image-models. 2019. DOI: 10.5281/zenodo.4414861 (cit. on p. 38).