POLITECNICO
MILANO 1863

Master of Science in Geoinformatics Engineering
School of Industrial and Information Engineering

# Improving Typha Classification using Spatial Information

**Supervisor: Giacomo Boracchi, Ph.D.**

**Graduation Thesis of:**
**Guidi Matteo**
**Student ID: 878827**

**Academic Year 2018-2019**

# Contents

# Acknowledgments

In first place i'd like to thanks Prof. Giacomo Boracchi for the help he provided and the availability to solve all my doubts.

I am also grateful to Prof. Ludovico Biagi, that accepted me in this MSc. course and motivated me to take this path. I would also like to thanks all the professors that helped me in my journey.

# Abstract

Remote sensing, the science of obtaining information about objects or areas from a distance, typically from aircrafts or satellites, has gained a lot of attention in the latest years. Due to the increasing number of satellites, providing daily and high resolution data, the need of automatic systems, that can analyze images and extract important features, is progressively increasing, and one of the most advanced method is represented by the neural networks. In this thesis, we used neural networks, a set of model mimicking the functioning of the human brain and designed to recognize patterns, in order to analyze, classify and extract important features from satellite images. In particular, we focused on the classification of Typha australis, an invasive aquatic plant that is spread in African regions and which correlates with the presence of the dangerous parasite Schistosoma. In order to classify this plant, we propose a classification method that exploits not only the spectral characteristics of the plant, but also spatial informations, given that Typha plants tend to grow close to or inside rivers. Differently from previous studies, where the neural network learns spatial constraints from data themselves, we integrated Geographic Information System (GIS) data in the classification as an "a priori" input. This integration turned out to substantially help the model to perform a better classification process. The solution of exploiting GIS data, which are updated every few days with high precision and freely available from websites like OpenStreetMap, represents an advantageous strategy for improving environmental assessments, easier to implement with respect to other state-of-the-art solutions. Furthermore, since the groundtruths provided were not precise, and the pixels therein contained did not refer to a single class, we also proposed a pre-processing technique, aimed at identifying and removing all the pixels which did not belong to the Typha's class. This process is based on the k-nearest neighbours algorithm. The integration of these approaches turned out to substantially helped the model to classify with high precision Typha's plants.

# Sommario

Il “remote sensing”, cioe’ la scienza che consente di ottenere informazioni a distanza su oggetti o su aree territoriali per mezzo di velivoli o satelliti, ha riscosso una grande attenzione negli ultimi anni. L’enorme quantita’ di dati ad alta risoluzione ottenuti su base giornaliera offre la possibilita’ di sfruttare le informazioni ottenibili da tali dati, grazie al numero sempre crescente di satelliti. Di conseguenza, sta diventando sempre piu’ pressante la necessita’ di mettere a punto sistemi automatici che siano in grado di analizzare le immagini ed estrarne le piu’ importanti caratteristiche. Le reti neurali, un insieme di algoritmi che analizzano, classificano ed estraggono le caratteristiche delle immagini attraverso procedimenti che simulano il funzionamento del cervello umano, rappresentano uno dei metodi piu’ avanzati in questo ambito. In questa tesi, il modello delle reti neurali e’ stato utilizzato per la classificazione della Typha australis, una pianta acquatica invasiva che si sviluppa nelle regioni africane e la cui distribuzione e’ correlata con la presenza del pericoloso parassita Schistosoma. Per ottimizzare la classificazione della Typha australis, abbiamo deciso di sfruttare non solo le caratteristiche spettrali della pianta, ma anche alcune specifiche informazioni spaziali. In particolare, dato che le piante di Typha tendono a crescere vicino o all’interno dei fiumi, abbiamo integrato nella classificazione, come input “a priori”, i dati che descrivono la localizzazione dei corsi d’acqua ottenuti dal Geographic Information System (GIS). Questa implementazione, che rappresenta una novita’ rispetto alla normale procedura in cui le reti neurali apprendono dai dati forniti i vincoli spaziali durante il processo di training, ha migliorato in modo sostanziale il processo di classificazione. La strategia di sfruttare i dati GIS, come quelli forniti da OpenStreetMap che vengono aggiornati ogni giorno con alta precisione e che sono liberamente disponibili nel Web, puo’ consentire di migliorare in modo sensibile le valutazioni ambientali eseguite tramite le analisi delle immagini satellitari, soprattutto se queste ultime sono, per vari motivi, di qualita’ non elevata. Inoltre, poiche’ i “groundtruth" forniti sono non precisi e i pixel contenuti non si riferiscono a una singola classe, e’ stata anche implementata una tecnica di pre-elaborazione che, usando l’algoritmo “k-nearest neighbours”, si e’ mostrata capace di identificare e rimuovere i pixel non appartenenti alla classe della groundtruth. L’integrazione di questi approcci si e’ rivelata di notevole utilita’ per aiutare il modello a classificare con precisione le piante di Typha.

# Chapter 1

# Introduction

This thesis is framed in the Remote Sensing and Machine Learning areas. It focuses on the segmentation of satellite images by means of machine learning techniques. Remote sensing allows to detect and monitor the physical characteristics of an area by measuring its reflected and emitted radiation from a distance. Earth remote sensing is performed by sensors which can be mounted on satellites or aircrafts. Since each object has its own unique spectral signature, which also depends on its chemical composition, the ability of satellites to sense information in various wavelengths of the spectrum is critical for their use in mapping, where the distinction between different objects is essential. In the recent years, a huge number of new satellites are being launched, in order to monitor the Earth in its entirety. The number of data extracted is growing every day, providing the opportunity to make extensive research in many fields, from oil spills detection for environmental preservation to locating wetland ecosystems to prevent their degradation, from locating groundwater activities to monitoring dangerous vegetation spread. This is the case of Typha, an invasive aquatic plant that has largely spread out in some African lands, disrupting farming and fishing activities and causing ecological imbalance. Besides being highly invasive, the localization of this plant is correlated with the presence of the Schistosoma parasite, which induces schistosomiasis, a very dangerous illness, especially in developing countries. Schistosomiasis belongs to the group of the so called Neglected Tropical Diseases, which globally affect more than 200 million people [1]. The economic and health effects of schistosomiasis are therefore considerably high as stated by the World Health Organization (WHO) [2].

Mapping the risk of schistosomiasis spread is fundamental, in order to minimize the population exposure and increase awareness of local communities. In this frame, the project MASTR-SLS (Mapping Schistosomiasis Risk in the Saint Louis region, Senegal, [3]) aims at collecting high resolution geographical information in order to create eco-epidemiological models that can be translated into operative tools to fight the disease. The Senegal River Valley, a prototypical endemic region of Senegal, was selected as a case study. In order to provide groundtruths of aquatic Typha australis, in March/April

2019 multispectral images of prototypical waterpoints of the lower Senegal River Valley were taken with drones. In this thesis, we exploited the groundtruths provided by the project for satellite image interpretation.

One of the most important and common method to interpret a satellite image, providing a map that is easier to understand, is classification, which means assigning to each pixel of the image a specific class that it represents [4]. Different types of image classification algorithms, ranging from visual interpretation methods to advanced machine learning algorithms, are being used since the early 1970s, when Earth observation's information became available. Early works for classification mainly used handcrafted features, and were based on, for example, color histograms or texture descriptors, and, subsequently, on SIFT [5] and HOG [6]. These methods proved to be ineffective, since they captured only a single characteristic inside the image. More recently, the unsupervised learning was considered a more attractive alternative, allowing the learning of features from images instead of relying on manually designed information [7]. Typical unsupervised feature learning methods included principal component analysis (PCA) [8], k-means clustering, sparse representations and autoencoders. Although unsupervised feature learning methods have achieved good performance for land-use classification, the lack of semantic information provided by the category labels does not allow an optimal discrimination ability between classes. Most of the current state-of-the-art approaches rely on supervised learning to obtain good feature representations, and the methods that are mostly used are decision trees [9], support vector machines [10] and nearest neighbour.

Deep learning represents a further improvement for machine learning. When compared with unsupervised feature learning methods, deep learning models, exploiting multiple processing layers, are able to learn additional data features. To achieve this, deep learning uses a layered structure of algorithms called artificial neural network (ANN) that recognizes patterns through processes simulating the way by which a human being learns [11]. Neural networks automatically extract features from data using a general-purpose training procedure and classify large amounts of data based on a labeled dataset on which they previously trained. Neural networks have been exploited in the remote sensing area to perform pixel-wise classification of satellite images, that is assigning with precision to each single pixel a specific label that represents a semantic class, e.g. vegetation, buildings, vehicles or roads, thus providing a map that can be easily interpreted. Indeed, per-pixel classifiers develop a signature obtained by combining the spectra of all training-set pixels, assuming that the contributions of all materials present is homogeneous and thus neglecting the impact of the mixed pixels [12].

The aim of this thesis was to provide a detailed analysis of the multispectral images of the Senegal River Valley using neural networks, in order to localize Typha with the highest precision. The attempt to classify available satellite images (4 band RGB +NIR, 3m resolution, Section 4.1) using the provided Typha's groundtruths produced very poor

results. We investigated therefore the possibility to improve the process of classification by integrating spatially derived information.

A large number of remote sensing classification techniques utilizes the spectral characteristics of the image only, while ignoring the spatial constraints [13]. Conversely, the spatial correlation among different classes could be succesfuly exploited to significantly enhance the classification, especially when the proximity between different pixels belonging to specific classes is known a-priori. This is the case of Typha, which is known to grow near or inside the rivers. The spatial correlation between these classes was obtained from the Geographic Information System (GIS) data of the river, and integrated into the classifier as an "a priori" input. To the best of my knowledge, the process of integrating GIS data into the classification algorithm has not been used yet for similar projects. Besides being easier to implement with respect to other state-of-the-art solutions, this method offers the advantage that OpenStreetMap, an initiative to create and make geographic data freely available [38], provides GIS data of rivers, cities, roads and other classes, for almost every country in the world. GIS data are updated every few days with high precision, thus providing a relevant source for improving environmental assessments and monitoring. To further improve the classification performance, we propose an additional method. As Typha plants grow near or inside the river, the provided groundtruths of Typha inevitably contain pixels that are wrongly labeled, as they belong to the river. The proposed method consists in the cleaning of the groundtruths in order to obtain a more homogeneous and correct dataset. The integration of these methods significantly improved the process performance and allowed to obtain more refined and clean data compared to the basic method.

The thesis is structured as follows:

- In Chapter 2 we present the background concepts necessary to understand this thesis together with the state-of-the-art researches.
- In Chapter 3 we first introduce the characteristcs of Typha and the MASTR-SLS project. Then the aim of this thesis is explained in details and formally defined. We next present the proposed approach for classification with the integration of spatial constraints in details. The algorithm proposed to clean the groundtruth data is discussed thoroughly.
- In Chapter 4 we present the datasets. Then the preprocessing steps performed in order to use them properly are explained in details.
- In Chapter 5 we discuss all the details necessary to implement the neural networks models. The problem of class imbalance is also discussed.
- In Chapter 6 we assess the performances of the experiments and present the results.

- In Chapter 7 we summarize the contribution of this work and propose future improvements.
- In Appendix A we report an extensive version of the experiments.

# Chapter 2

# Backgrounds and Related Works

The use of remote sensing has gained increasing attention in the latest years. This is because the number of satellites providing daily and high resolution data offers the possibity to exploit the knowledge that can be extracted from satellite images. The need of automatic systems, that can analyze and extract important features, is increasing, and one of the methods that developed the most is the neural networks.

The following chapter is focused on how neural networks can be used to classify and segmentate remote sensed satellite images. A detailed explanation on the basis of neural networks and how they work, is provided.

This chapter starts with an overview of remote sensing in Section 2.1. Then the theory behind neural networks and how they work is discussed in details in Section 2.2. In Section 2.3 convolutional neural networks are introduced, and the most important elements which compose them are introduced. Finally in Section 2.4 other studies which use machine learning algorithms to classify satellite images are briefly presented.

## 2.1 Introduction to Remote Sensing

Altough it has been described in many different ways, remote sensing can be defined as "the science of gaining information from a distance" [14].

Commonly, remote sensing refers to the act of scanning the Earth by sensors in order to obtain information on Earth's land and water surfaces. The used sensors (usually cameras and digital scanners) can be installed on airborne (aircraft and balloons) or spaceborne (satellites and space shuttles) platforms. A sensing system usually produces digital pictures representing the objects/events being observed, that need to be analyzed and interpreted in order to extract useful information. The quality and quantity of these information is called resolution, and can be divided in three main categories:

1. *Spectral resolution* describes the number of spectral bands in which a sensor is able to resolve the wavelenghts of the electromagnetic spectrum. There are two

important characteristics that define the spectral resolution:

- Number of spectral bands, which defines how many bands the sensor is able to distinguish. There are 3 different categories: monospectral, meaning the sensor is able to collect data in only one single band; multispectral, meaning that the sensor is able to collect data in multiple bands, typically from 4 to 12; hyperspectral, meaning that the sensor is able to collect data in hundreds of bands.
- Width of each bands, which defines the portion of the spectrum in which each band can resolve the energy received.

2. *Spatial resolution* defines how much detailed the image is. The area captured from a sensor is defined by:

   - Istantaneous Field of View (IFOV), is the angular cone over which radiation is detected. A narrow angle produces a smaller IFOV, while a larger angle produces a higher IFOV.
   - Height, i.e. the distance from Earth (altitude), which also determines the resolution. A higher altitude produces a lower resolution.

   A pixel, defined as the smallest area that is identifiable on the image, can be expressed by the following equation:

   $$pixel = Height \times IFOV \tag{2.1}$$

   Which describes the aforementioned relations. A smaller pixel size means a higher resolution. The spatial resolution defines the area of the real world which is covered by a pixel. For example a spatial resolution of 10 meters means that each pixel of the image covers an area of $10 \times 10$ meters.

3. *Temporal resolution* is defined as the amount of time needed to revisit and acquire data for the exact same location. When applied to remote sensing, this amount of time depends on the altitude and orbit of the satellite as well as its sensor characteristics. A key feature that satellites systems provide is the ability to systematically observe multiple sequential image taken in the same area at different times. This allows to compare images with very high precision, and extract important information regarding the changes occured in the area sensed.

There are different kinds of sensors, but the most important distinction is based on the different ways with which they capture images:

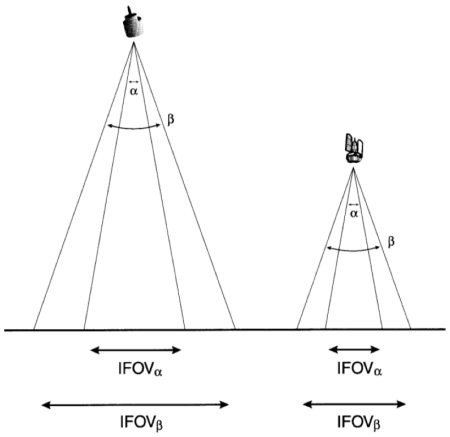



*Figure 2.1: IFOV variation with altitude and angle, as described in Formula 2.1 [15].*

– *Passive Sensors* are instruments that capture the sun's energy that is either emitted or absorbed by an object. The signal received depends on the physical characteristics of the object. This kind of sensing can be exploited only when the sun is illuminating the object. During the night, or in presence of shadows and clouds, the receipt of the signal is impossible.

– *Active Sensors* are instruments that send a signal towards an object and measure the radiation reflected. These kind of sensors do not need the light in order to be able to gather information.

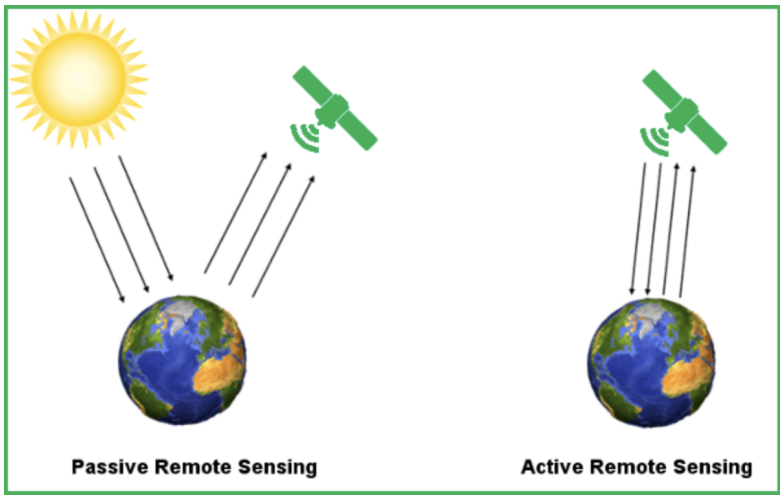



*Figure 2.2: Passive sensors vs active sensors [16].*

## 2.2 Neural Networks

In recent years, there has been an increase interest in the use of neural networks to classify remote-sensed data. These methods have proven to give better results than the other state-of-the-art classifiers, thanks to their capability to model non-linear processes and identify unknown patterns [17]. Indeed, neural networks are a set of algorithms design to simulate the way by which a human being learns. Specifically, neural networks analyze a defined input, called training data, until they have "learned" how to correctly relate the defined input to the desired output. Neural networks are based on key elements which are briefly introduced below.

### 2.2.1 Neurons

The fundamental unit of a neural network is called an artificial neuron. A neuron is a simple mathematical model, that produces an output (makes a decision) based on the inputs received (Figure 2.3). The different inputs are multiplied by specific weights and then the arithmetic sum is converted in output through an activation function, which allows data to be passed to the other neurons. The equation defining the output result is the following:

$$y = f(\sum_{j=0}^{k}(w_j x_j) + b_j) \tag{2.2}$$

Where $y$ is the output, $f(x)$ is a non-linear activation function (described in Section 2.2.2), $k$ is the number of inputs, $w_j$ are the so called weights and $b_j$ is the neuron bias. The weights are real numbers that express the importance of a certain input: a higher weight means that an input will contribute more to the final sum. The bias is an additional constant parameter which is used to help fitting the model in the best possible way for the given data.

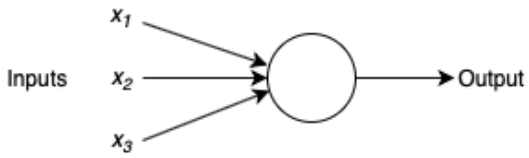



*Figure 2.3: Example of a neuron.*

A neural network is an architecture composed of different layers of neurons. The first layer is usually called the input layer, the last layer is the output layer, and all the ones in between are called hidden layers. The first layer takes the data as inputs, and make some simple decisions. The subsequent layers make more and more complex decisions depending on the outputs of the previous layers. The deeper the network, the more sophisticated and complex the decisions are.

These types of networks, and the models used for this thesis, are called *feedforward* neural networks, that means that the output from one layer is used as input to the next layer and that there are no loops. The information goes directly from the input to the output. The feedforward neural network was the first and simplest type of artificial neural network devised, where the information moves in only one direction, forward, from the input nodes, through the hidden nodes (if any) and then the output nodes.

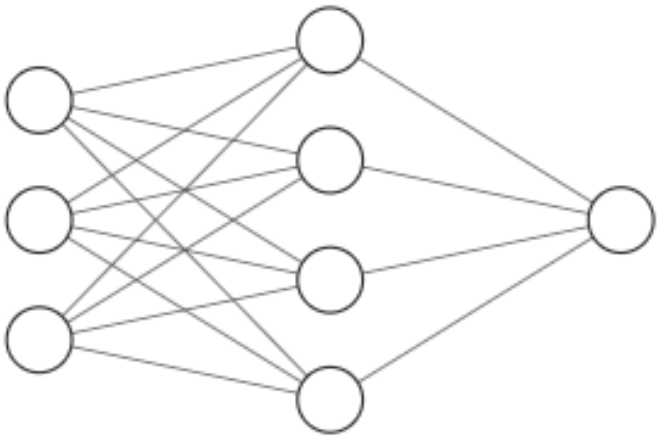


*Figure 2.4: Example on a network composed of an input layer with 3 neurons, an hidden layer with 4 neurons and an output layer with 1 neuron [18].*

### 2.2.2 Activation Functions

The decision of which neuron produces an output for the next layers and the value which is passed is determined on the basis of the activation functions, which define the output of that node given an input or set of inputs. The activation function is used to introduce non-linearity in the modeling capabilities of the network. There are different kinds of activation functions, and here some are presented.

#### ReLU

The Rectified Linear Unit (ReLU) is the most commonly used activation function in deep learning models. This function outputs the value of 0 if it receives any negative input, whereas, for any positive value, it returns that value like a linear function (Figure 2.5). From a mathematical point of view, this can be expressed as:

$$f_{RELU}(z) = max(0, z)$$

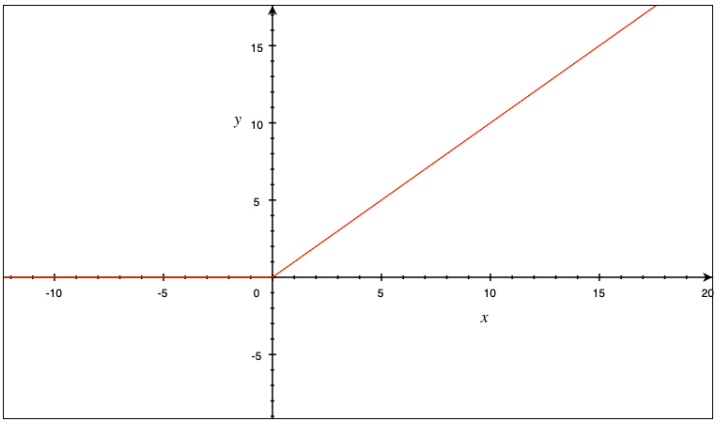



*Figure 2.5: Relu activation function.*

### Softmax

The Softmax activation function is different from the others, since its purpose is to output a probability of each target class over all the possible classes. Typically in neural networks, Softmax is the output function of the last layer, which has the role to turn the score produced by the entire network into values that can be easily interpreted by humans. The range of each probability is between 0 and 1, and the sum of all probabilities is equal to 1. The mathematical formula is:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}$$

Where $z_i$ is the vector of inputs and $k$ is the number of classes. The exponential is applied to each element of the input vector and then is divided by the sum of the exponentials of all these elements.

## 2.2.3 Loss Functions

After the input data passes through all the neurons of the network, which apply their transformation and send data to the neurons of the next layer, the final layer is reached with a result. The divergence between the estimated and expected value is called loss function. The loss function has an important job, since it reduces all the aspects of a complex system down to a single number, a scalar value, which allows candidate solutions to be ranked and compared, in order to select the one which reduces the most the error in prediction. Therefore, as the model is being trained, the weights of the neuron interconnections will gradually be adjusted until good predictions are obtained. In other words, the neural network learns to map a set of inputs, given as training data, to a set of outputs.

Specifically, when a set of training data, whose output is known, is given as inputs to the model, weights and biases are adjusted with an iterative process in order for the algorithm to model at best the given data. If predictions deviate too much from actual results, the loss function will have high values. The goal of training a neural network is to find the particular set of weights that minimizes this value.

There are multiple loss functions that can be used, depending on the network architecture and the data used. Among the many loss functions, the most importants include the mean squared error, the mean absolute error, the cross entropy and the Hinge loss.

The loss function used in this thesis was the categorical cross entropy, which is the most common setting for classification problems. Cross entropy is used to assess the performance of the classification, in particular by comparing how well a set of predicted probabilities matches the true probability distribution. The cross entropy loss becomes smaller as the prediction gets more accurate, and turns to zero if the prediction is perfect. The formula is the following:

$$L(p, q) = -\frac{1}{m} \sum_{i=1}^{m} \sum_{k=1}^{K} y_k^{(i)} \log(\hat{p}_k^{(i)}) \tag{2.3}$$

Where $m$ is the number of observations, $K$ is the number of classes. $y_k^{(i)}$ is equal to 1 if the target class for the $i$-th instance is $k$; otherwise, it is equal to 0. $\hat{p}_k^{(i)}$ is the probability for the $i$-th observation to belong in the $k$-th class.

### 2.2.4 Network Training

The training phase of a neural network is the process of adjusting weights and biases with an iterative process, in order to obtain the smallest loss value. Optimisation algorithms are used to update weights and biases, i.e. the internal parameters of a model, to reduce the error. Gradient Descent is one of the most commonly used techniques to optimize neural networks. It is a first-order iterative optimization algorithm which updates the weights by moving in the direction opposite to the gradient of the objective function with respect to the network parameters.

Indeed, given that the gradient is a vector that mathematically gives the direction of steepest increase and since the objective is to minimize the loss function, parameters are updated in the negative gradient direction [19].

As previously said, training is the adaptation of the weights in such a way that the loss function is minimized. After the Forward Pass, where the data are used as inputs and the weights are randomly initialized, backpropagation allows the updating of weigths based on the output produced in the first phase. Starting from the last layer, and going backwards, the weights are updated using the gradient descent.

Backpropagation is the essence of neural network training. The fine-tuning of weights

based on the error rate (i.e. loss) obtained in the previous epoch (i.e. iteration), ensures lower error rates, making the model reliable by increasing its generalization (Figure 2.6).

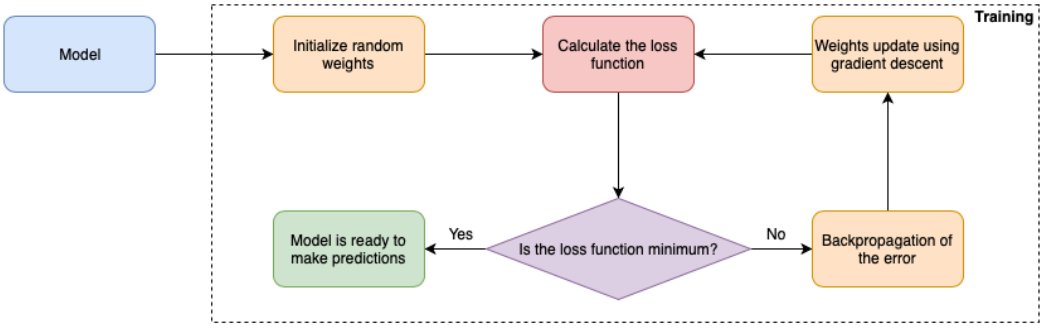



*Figure 2.6: Scheme of the neural network training.*

There are different gradient descent optimisers, which adapt the learning rate component using factors that are functions of the gradients. The most used are Momentum, Nesterov Accelerated Gradient, Adagrad, RMSprop and Adam. In this thesis it was chosen to use the Adaptive Moment Estimation (Adam) optimizer because is an algorithm designed specifically for training neural networks. It shows better performances and results compared to the others methods, combining the advantages of two other extensions of stochastic gradient descent, the AdaGrad and RMSprop [20].

### Optimizer: Adam

Adam is a stochastic gradient descent algorithm based on estimation of 1st and 2nd-order moments. The algorithm estimates 1st-order moment (the gradient mean) and 2nd-order moment (element-wise squared gradient) of the gradient using exponential decay rates for the moment estimates, and corrects its bias [20]. The final weight update is proportional to the learning rate times the 1st-order moment divided by the square root of 2nd-order moment. The formulas used to update the parameters are:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{2.4}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{2.5}$$

$$\theta_t = \theta_{t-1} - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}} + \epsilon} \tag{2.6}$$

Where $t$ is the timestep, $\theta$ is the model weights, $m$ is the 1st moment vector moment vector, $\hat{m}$ is the bias-corrected 1st moment estimate, $v$ is the 2nd moment vector, $\hat{v}$ is the bias-corrected 2nd moment estimate, $\alpha$ is the learning rate and $\epsilon$ is a constant, usually set to $10^{-8}$.

### 2.2.5 Regularization

As described, during training the error gradient is backpropagated from the output layer to the input layer and, through the gradient descent step, the gradient calculated is used to update every weight in the network. The longer the network is trained, the more specialized the weights will become to predict the training data, with the risk of overfitting them. In such a case, the network will likely perform poorly when making predictions on new data. Updating the learning algorithm to encourage the network to minimize the weights can be of help, improving the generalization of the model. This process is called is called "weight regularization". The two most used methods to apply regularization are the L1 and L2.

#### Kernel Regularizer L1

L1 regularization is also called *Lasso Regression*, and adds to the loss function the absolute value of the of all the feature's weights (Equation 2.7, part in the box).

$$L(x, y) = Loss + \boxed{\lambda \sum_{i=1}^{n} |\theta_i|} \tag{2.7}$$

Where $\lambda$ is the parameter of regularization, *Loss* is the loss function (introduced in Section 2.2.3), $n$ is the number of weights and $\theta$ is the value of each weight. It can be noticed that if $\lambda$ is equal to 0, then the regularization term is cancelled and only the loss function remains. Thanks to the addition of this term, a feature selection process is made: features with values that are non significant are set to zero, while useful features are set to non-zero values.

#### Kernel Regularizer L2

L2 regularization is also called *Ridge Regression*, and adds to the loss function the sum of the squares of all the feature's weights (Equation 2.8, part in the box).

$$L(x, y) = Loss + \boxed{\lambda \sum_{i=1}^{n} \theta_i^2} \tag{2.8}$$

Where $\lambda$ is the parameter of regularization, *Loss* is the loss function (introduced in Section 2.2.3), $n$ is the number of weights and $\theta$ is the value of each weight. It can be noticed that if $\lambda$ is equal to 0, then the regularization term is cancelled and only the loss function remains. Thanks to the addition of this term, the weights are kept smaller, thus preventing the overfitting problem.

## 2.3 Convolutional Neural Networks

Convolutional Neural Networks (CNN) are a special type of neural networks, which classify images by taking advantage of the spatial structure. CNN are designed to process data in multiple arrays, for example multi-spectral images composed of multiple two dimensional arrays containing pixel intensities in different band channels. CNN are typically organized as a series of layers. The convolutional layer extracts the features from the image, with the first layers extracting low-level features (edges and corners) and the deeper layers learning more complex features (objects and shapes) by combining the information deriving from the low-level layers. Afterwards, pooling layers compute local non-linear operations of a particular feature over a region of the image, allowing the result consistency even in the presence of small translations or rotations of the image. Subsequently, the normalization layer aims to improve generalization and finally the fully connected layers, which represent the last layers of the network, summarize the information conveyed by lower-level layers in view of the final decision [7]. Additional functions are employed to correct possible over-fitting.

For these reasons, in the last years, convolutional neural networks have been used for a multitude of tasks in remote sensing applications, and have been proven to be especially useful in classification tasks [21] [22] [23].

The main components of a CNN are described in details in the following sections. In this thesis a CNN was used to solve the image segmentation problem.

### 2.3.1 Convolutional Layer

The convolutional layer is one of the most important layers in a convolutional neural network. It consists of a series of filters that take into account different spatial dimensions (width and height) but the whole input depth. Generally, a filter dimension can be described as $(w \times h \times d)$, where $w$ is the width, $h$ is the height and $d$ is the depth. During the forward pass, each filter is convolved (meaning slid across) through the whole image and is multiplied with the input values with an element wise multiplication (Figure 2.7).

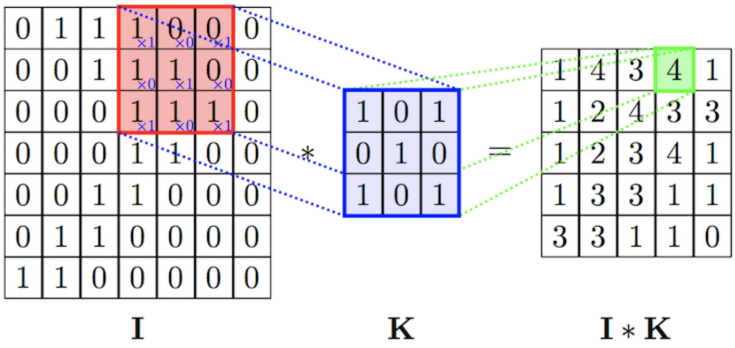



*Figure 2.7: Element wise multiplication between input $I$ and filter $K$, producing a feature map [25]*

This process is performed for all the channels of the image, and the results are summed together to produce a feature map, that is a matrix containing the results of the convolution. Each filter produces different feature maps, that are stacked together to produce the output layer. The network is able to learn which filter responds to a local region of the input, thus exploiting the spatial correlation of the input image.

After the convolution of the input image, a feature map is produced. Conventionally, the first convolutional layer is responsible for capturing edges, colors, gradient orientation, etc, the so called *low level features*. More Convolutional layers can be added, that are able to capture more *high level features*, giving the network a complete understanding of the image (Figure 2.8).

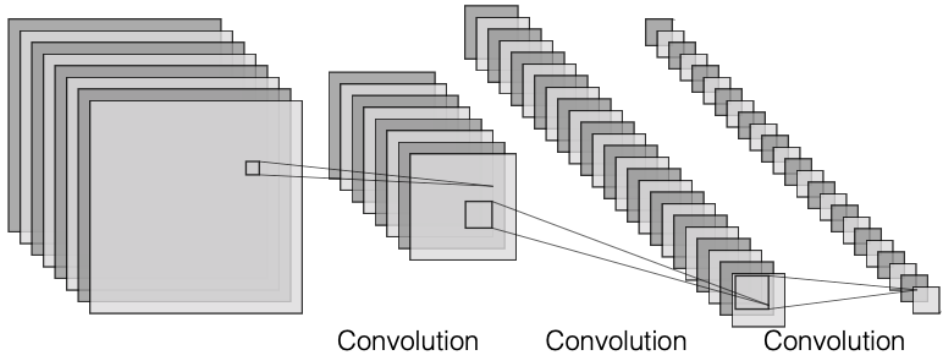



*Figure 2.8: Network composed of many convolutional operations, that decrease the size of the feature map but increase the depth [18].*

### Kernel Initializer

When the training of a model is started, during the first epoch, the weights are unknown. It is common practice to choose the weights randomly before starting to optimize them. This selection however can cause problems such as the vanishing or exploding gradients, meaning that the loss gradient is either too large or too small to perform the backpropagation properly. In these cases, the network will take longer to reach a convergence point, or it might not even reach it.

There are different ways of setting the initial weights using a procedure that prevents gradient problems, and the one chosen in this thesis was the *He initializer* [24]. With this technique, the weights are drawn from the uniform distribution within $[-limit, limit]$ where $limit$ is defined in the following equation:

$$limit = \sqrt{\frac{6}{fan\_in}} \tag{2.9}$$

Where $fan\_in$ is the number of neurons of the previous layer.

### 2.3.2 MaxPooling layer

It is a common practice to insert a Pooling layer between the convolutional layers. A Pooling layer is a sort of filter applied to the feature maps obtained from a convolutional layer. Its function is to reduce the spatial size of each feature map, in order to lower the number of parameters and speed up the computation. The pooling layer chosen in this thesis was a MaxPooling layer of $2 \times 2$, that means that the feature map is divided in many non-overlapping $2 \times 2$ matrices where the maximum value is selected. The new feature map contains these maximum values, in such a way that the feature map size is halved (Figure 2.9). The pooling layer is applied independently on every depth. The purpose of a pooling layer is to create a summarized version of the features detected in the input, where the most prominent features are retrieved even in case of rotation, distortion and other modifications.

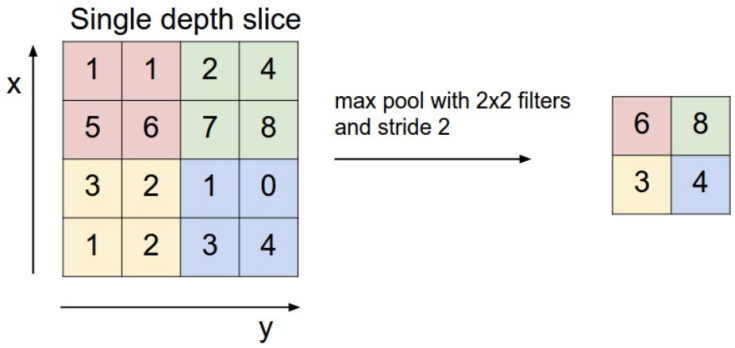



*Figure 2.9: An example of MaxPooling operation applied on a $4 \times 4$ matrix. The result is a $2 \times 2$ matrix where each element is the maximum value over the $2 \times 2$ filter [25].*

### 2.3.3 Activation Functions

In a convolutional neural network, the activation functions have the same purpose as in a neural network (described in Chapter 2.2.2). Typically, convolutional neural networks exploit ReLUs activation function, but since they are more "deep" with respect to feed-forward neural networks, the arising of problems such as overfitting and *dying ReLUs* is more frequent. The *dying ReLUs*, is the situation when a ReLU neuron is stuck in the negative side and always outputs 0, thus becoming dead (essentially an useless neuron). In order to prevent this situation, the ELU activation function can be used.

#### ELU Activation Function

Recently, a new activation function named Exponential Linear Unit (ELU) was introduced [26]. ELU is very similiar to ReLU except for negative inputs. Indeed, with ELU, a negative value is propagated to the next layer as a small value close to zero, while a

positive input is unchanged (Figure 2.10). With respect to the ReLU activation function, ELU can prevent the effect of *dying ReLUs*. The use of ELU instead of ReLU avoids that, over time, a large part of the network ends up not contributing anymore. From a mathematical point of view, it can be expressed as:

$$f_{ELU}(z) = \begin{cases} \alpha(e^x - 1) & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$$

Where $\alpha$ is a constant value that affects the negative values.

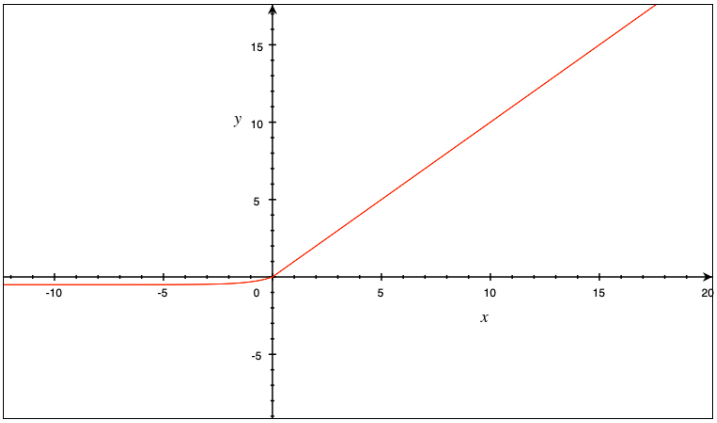



*Figure 2.10: ELU activation function, with $\alpha = 0.5$.*

### 2.3.4 Batch normalization layer

In a neural network, the model is updated backwards from the output to the input. In each hidden layer an estimation of the error is calculated, assuming that the weights in the previous layers are fixed. In reality, the inputs of each layer are updated every time that a parameter of a previous layer is changed, making the updating procedure very slow. This change, happening during training, is often referred to as Internal Covariate Shift.

Batch normalization is a technique proposed by Ioffe and Szegedy (2015) [27] that can help in the update of these parameters. A batch is defined as the number of samples used to train the network before updating the internal model weights. In neural networks models the batch is tipically equal to the whole training dataset, but in CNN models usually the training dataset is split in smaller parts, called mini-batches, that are used to train the model. The fundamental concept is that the normalization of the output of a previous activation layer is obtained by subtracting the batch mean and dividing by the batch standard deviation, rescaling the input data to obtain a fixed mean and variance (Equation 2.10). This can speed up the training process by dramatically reducing the

number of epochs required. The batch normalizing algorithm, applied to a mini-batch, is the following:

$$\begin{aligned}
\mu_{B_j} &= \frac{1}{m}\sum_{i=1}^{m} x_i && \textit{(mini-batch mean)} \\
\sigma^2_{B_j} &= \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2 && \textit{(mini-batch variance)} \\
\hat{x}_i &= \frac{x_i - \mu_B}{\sqrt{\sigma^2_B + \epsilon}} && \textit{(normalization)} \\
y_i &= \gamma\hat{x}_i + \beta && \textit{(scaling and shifting)}
\end{aligned} \tag{2.10}$$

Where $x_i$ are the values over a mini-batch $B_j = \{x_{1...m}\}$ and $\epsilon$ is a constant; $\gamma$ is the scale and $\beta$ is the shift, that are two additional parameters that the model needs to learn.

## 2.4 Classifiers Using Spatial Information

Classification of satellite images is an essential aspect of remote sensing. In recent years a large number of remote sensing classification techniques have been developed. Most methods base the classification on the spectral characteristics of the image, while ignoring the spatial constraints [28]. Spatial dependence is essential when dealing with geographic information, provided by satellite images.

The addition of spatial information in classification techniques has been defined as "spatio-contextual" image classification, indicating that a pixel classified belonging to a specific class is more likely to be surrounded by pixels of the same class [13].

Some of the most important studies dealing with spatial information added to classification are reported below.

**k-NN classifier** Atkinson and Naser (2010) [29] developed a k-nearest neighbor method for pixel classification incorporated with geographical weighting. The basis of the k-NN classifier states that pixels that are close in the feature space are more likely to belong to the same class. This method was improved by incorporating a weight scheme that gives more importance to information derived from close neighbors with respect to distant ones. The geographical weights introduced in this study are based on the inverse distance weighting and the difference distance weighting. With the first one the weights are based on the distance in feature space between two neighboring pixels. With the second one a pixel is assigned to a class using a decision rule that is affected by both the neighboring and distant class training samples. By adding these weights to the k-NN method, the results achieved a better performance accuracy with respect to a simple spectral classification.

**Spectral-Spatial Hypergraph** Ji et al. (2013) [30] created a hypergraph composed of one feature-based hyperedge, that connects pixels based on the spectral features, and one spatial-based hyperedge, that connects pixels based on the spatial layout. This idea was proposed because pixels that are close in the feature space have high probability to belong to the same class, but that holds also for pixels that are spatially nearby, since usually pixels of the same class are spread in an area. Once the hypergraph is created, a semi-supervised learning algorithm is performed, to learn the relation between pixels and the weights related to each hyperdge. The experiments presented affirm that the combination of the spectral-based hyperedge and spatial-based hyperedge was able to improve the classification performance with respect to other spectral based methods.

**Deep learning with spectral-spatial joint information** Chen et al. (2014) [31] proposed a deep learning-based feature extraction method for hyperspectral data classification that deals with joint spectral-spatial information. The method used is based on a stacked autoencoder network that takes spectral and spatial information as separated input. For the spectral information, the spectrum of a single pixel is taken in consideration. For the spatial information a neighbour region of that pixel is extracted. These data are then fed to the neural network to produce the class probabilities for each pixel. The results show that this kind of classifier yields an higher accuracy than traditional spectral classification methods.

**U-Net** Pixel-wise image segmentation is a very challenging task in computer vision. In recent years the CNNs proved their success in image classification, object detection and image segmentation. One of the most successful state-of-the-art deep learning method is based on the Fully Convolutional Neural Networks, and is now known as U-Net neural network [32].

A U-Net is similar to a convolutional encoder-decoder network, with its main feature being the connection between these two parts, in a way that some information in the decoder section comes from the encoder part, bypassing the compressive bottleneck i.e. the deepest part of the network. In this way the network can still generalize the information retrieved from the encoder part but also includes the spatial information, that is necessary when performing a per-pixel classification. The encoder part consists of a contrating path, while the decoder part consists of an expansive path.

The model architecture, as described in [32]: "... consists in the repeated application of two $3 \times 3$ convolutions (unpadded convolutions), each followed by a rectified linear unit (ReLU) and a $2 \times 2$ max pooling operation with stride 2 for downsampling. At each downsampling step the number of feature channels is doubled. Every step in the expansive path consists of an upsampling of the feature map followed by a $2 \times 2$ convolution ("up-convolution") that halves the number of feature channels, a concatenation with the correspondingly cropped feature map from the contracting path, and two $3 \times 3$ convolutions, each followed by a ReLU" (Figure 2.11).

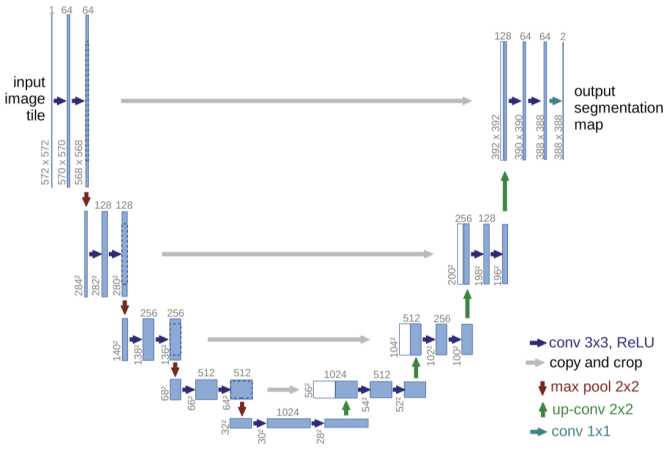



*Figure 2.11: U-Net architecture [32].*

This architecture has proved to be very successful in classification of satellite images, as noted in [33]. In this thesis, some experiments were carried out with a network based on U-Net (Section 5.4.1).

# Chapter 3

# Improving Typha Segmentation Using Spatial Information

Detection of objects over large areas is one of the primary drivers of interest in satellite imagery analytics. Optimization strategies are therefore required to successfully localize small objects in large images. This is the case for the localization of Typha, an invasive acquatic plant that has spread out in some African regions. The localization of this plants correlates with the presence of the *Schistosoma* parasite, which induces the schistosomiasis, a very dangerous illness. The MASTR-SLS is a project that aims to reduce the risk of illness controlling the spread of Typha.

The main contribution of this thesis is the implementation of algorithms designed to improve the localizaton of Typha. Deep learning techniques were applied to the satellite images and groundtruth data provided by the MASTR-SLS project, integrated with public domain GIS information. In particular, the novel approach presented is based on the integration of spatial relations between the plant and the river, based on the knowledge that Typha plants tend to grow near or inside rivers. Furthermore, and additional algorithmic approach was designed in order to remove from the Typha's groundtruths the pixels representing water, thus improving the spectral characterization of groundtruths.

In Section 3.1 the reasons related to the importance of localizing Typha are presented. In Section 3.2 the aim of this thesis is introduced. The Section 3.3 contains the core of this thesis: the algorithms implemented to integrate the spatial constraints and clean the groundtruth are presented. Furthermore the blob split algorithm, used to partition the groundtruth between training and test set, is presented.

## 3.1 Problems related to Typha

It has been widely reported that *Typha australis*, a species of cattail grass, is taking over river banks and farmlands in the wetlands of western african states, disrupting farming

and fishing activities and causing ecological imbalance [34]. Indeed, the presence of Typha reduces plant diversity [35], altering the plant-community system and the physical structure of vegetation [36], by replacing other emergent and submergent aquatic species. This can negatively affect fish communities, and therefore the whole food chain. Furthemore, due to its highly invasive nature, Typha australis destroys forests and harvestable lands. Also, the plant's entangled roots spreads, rapidly blocking waterways and favoring the standing of shallow waters, a condition which further promotes the diffusion of the plant, particularly on the fertile lands close to rivers.

Most importantly, Typha hosts freshwater snails (Figure 3.1), which represent the intermediate hosts of the Schistosoma parasite. These parasitic worms, released from infected snails, contaminate people that enter in contact with infected water during agricultural, domestic and recreational activities, causing the schistosomiasis. This disease might lead to liver damage, kidney failure, infertility and bladder cancer. In children, it may cause poor growth and learning difficulties. Furthermore, urogenital schistosomiasis is considered to be a risk factor for HIV infection. The economic and health effects of schistosomiasis are therefore considerably high as stated by the World Health Organization (WHO) [2].

*Figure 3.1: Typha plants hosting the freshwater snails, carriers of the Schistosoma parasite [3].*

In order to provide ground-truthing for remote detection of aquatic Typha australis, the project MASTR-SLS (Mapping Schistosomiasis Risk in the Saint Louis region, Senegal, [3]) was started, and in March/April 2019 multispectral images of prototypical waterpoints of the lower Senegal River Valley were taken with drones. This project aims to exploit the satellite images to map Typha with high precision.

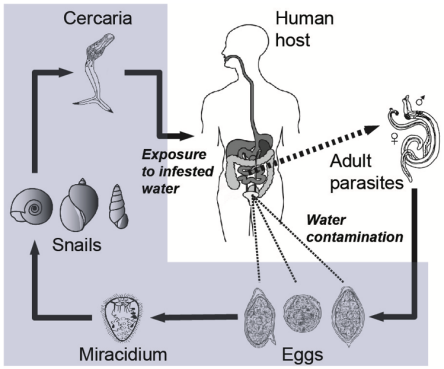



*Figure 3.2: Schistosomiasis transmission cycle [37].*

## 3.2 Aim of the Thesis

In order to precisely locate Typha, satellite images need to be analyzed, and algorithms capable of classifying unlabeled pixels are required. As already stated in Chapter 2 machine learning models, and in particular neural networks, are suitable for these kinds of tasks. The aim of this thesis was to implement neural networks model in order to classify Typha with the highest precision. The attempt of classifying available satellite image using only the spectral information produced very poor results since the spectral signature of the plants is often not sufficient to differentiate plant species. This especially occurs when the satellite images are constituted by few bands, as is in this case, where the bands provided were only 4 (Blue, Green, Red, Near InfraRed). However, the availability of additional information about the specific plant to be located, could be exploited to ease the classification task. As already mentioned, Typha plants tend to grow near or inside the rivers. The novel idea at the basis of this work was to integrate the spatial correlation between these classes, obtained from the Geographic Information System (GIS) data of the river, into the classifier as an "a priori" input. Furthermore, in order to have more precise Typha's groundtruths, a cleaning method was developed, so that the pixels wrongly labeled were removed. These integrations proved to be very useful, and allowed to obtain better results with respect to the basic methods.

Furthermore, a new method of partitioning the inputs, called blob split, was implemented, and the results were compared with the random split method.

## 3.3 Proposed Solutions

In this section the main solutions and the novel ideas are discussed in details. First, the segmentation problem is introduced. Then the creation of a matrix which incorporates spatial constraints is explained in details. Then the algorithm exploited to clean the groundtruth is reported. Finally, the blob split method, used to partition the input data into training and test set, is discussed.

### 3.3.1 Problem Formulation

The goal of the classification process is to find and detect all the pixels $p$ which belongs to a specific class $c_i$ in an image $I$. To this purpose, we will pursue a pixel-based classification approach. To reach our goal we are given:

- An image $I$, composed by a certain number of pixels $p$.
- A set of classes $\mathcal{C} = \{c_1, c_2, \ldots, c_i\}$, where $i$ denotes a different class.
- A set of groundtruths $GT_c = \{GT_1, GT_2, \ldots, GT_i\}$, where each $GT_i$ is the groundtruth of the $c_i$ class, and is composed by a certain number of pixels $p_i$.

Our goal can be defined as:

$$\begin{gathered} \textit{Train a classifier that:} \\ \forall p \in I \text{ assign } p \text{ to the class } c_i \text{ relying on features learned on } GT \end{gathered} \tag{3.1}$$

Thus creating a set of non-overlapping regions $S_1, S_2, \ldots, S_n$ which, when combined, form $I$:

$$\bigcup_{i=1}^{n} S_i = I \text{ where } S_i \cap S_j = \varnothing \tag{3.2}$$

We decided to solve this problem using neural networks and convolutional neural networks, presented in Chapter 5.

### 3.3.2 Integration of Spatial Information

Typha plants tend to grow near or inside the rivers. The novel idea at the basis of this work was to integrate this relationship into neural networks models, in order to detect with more precision the Typha plants present in the area.

Given as inputs:

- An image $I$, of $w \times h \times d$ dimensions.
- A known relationship between Typha and river.

The goal is defined as:

Integrate the known spatial relationship into the neural network model.

There are different solutions that can be employed, but we decided to augment the input data with an image representing the proximity of each pixel with respect to the closest river's pixel 3.3..

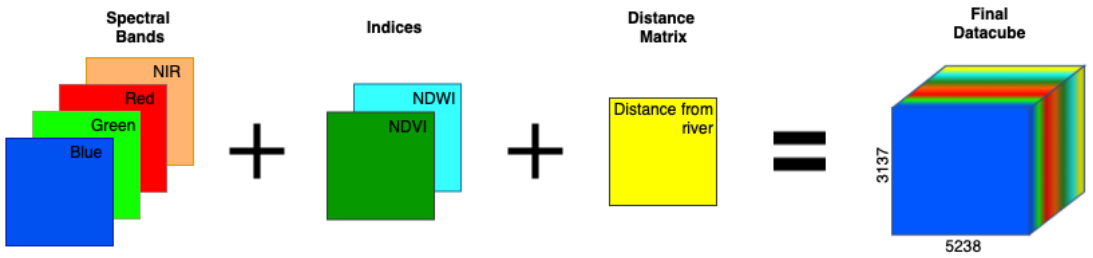



*Figure 3.3: Creation of the datacube used as input for the neural network models. The "distance from river" layer was created an then used as an additional input.*

To achieve that, we created an image $D$ in which every pixel's value is equal to the distance from the closest pixel of the class $r$, that is the river. In order to obtain this image the following input is given:

- A mask $M$ in which the pixels belonging to the class $r$ are labeled as 1, while all other pixels are labeled as 0.

The formal definition of the problem is the following:
*Given a boolean matrix $M$ where the pixels belonging to the river class $r$ are labeled as 1, create a matrix $D$ where, for every pixel, the distance $d$ from the closest pixel of class $r$ is computed.*

The algorithm used in order to create the matrix $D$, is reported below:

**Algorithm 1** Computation of matrix $D$

1: Initialization of $D$ as a zeroes matrix with dimensions equal to $M$
2: **for** $i$ `in` $M$`_rows` **do**
3:  **for** $j$ `in` $M$`_columns` **do**
4:  Initialize $min\_dist = 0$
5:  **if** $M_{i,j} = 0$ **then** ▷ i.e. $p$ does not belong to the class $r$
6:  **for** `Evey pixel` $p$ `= 1 in` $M$ **do** ▷ i.e. $p_r$ belongs to the class $r$
7:  Compute distance $d$ between $p_{i,j}$ and $p$
8:  **if** $d \leq min\_dist$ **then**
9:  $min\_dist = d$
10:  $D_{i,j} = min\_dist$ ▷ i.e. assign to $D_{i,j}$ the minimum distance found

In order to compute the distance between two pixels (line 7 of Algorithm 1) different methods could be employed:

**Chessboard distance** between two pixels $(x_1, y_1)$ and $(x_2, y_2)$ can be computed as:

$$d = \max(|x_1 - x_2|, |y_1 - y_2|) \tag{3.3}$$

**Cityblock distance** between two pixels $(x_1, y_1)$ and $(x_2, y_2)$ can be computed as:

$$|x_1 - x_2| + |y_1 - y_2| \tag{3.4}$$

**Euclidean Distance** between two pixels $(x_1, y_1)$ and $(x_2, y_2)$ can be computed as:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{3.5}$$

**Quasi-Euclidean Distance** between two pixels $(x_1, y_1)$ and $(x_2, y_2)$ can be computed as:

$$d = \begin{cases} |x_1 - x_2| + (\sqrt{2} - 1)\,|y_1 - y_2| & \text{if } |x_1 - x_2| > |y_1 - y_2| \\ (\sqrt{2} - 1)\,|x_1 - x_2| + |y_1 - y_2| & \text{otherwise} \end{cases} \tag{3.6}$$

Given the matrix $M$ and the class $k$ the computation of the distance matrix, using the Euclidean distance described in Equation 3.5, was performed. An illustration is provided in Figure 3.4, and in Figure 3.5 the results of the algorithm applied to an image are reported.

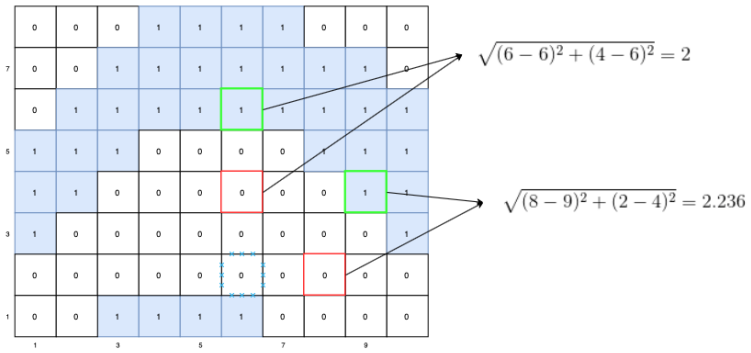



*Figure 3.4: Image showing how the euclidean distance from each pixel labeled as '1' to the closest pixel labeled as '0' is computed. The euclidean distance between a '0' pixel (in red) and the closest '1' pixel (in green) is computed using the formula presented in Equation 3.5.*

*Figure 3.5: Results of the algorithm applied to an image. On the left is presented the river mask, where all the white pixels belong to river, while the black ones do not. On the right is presented the result of the algorithm, where the distance of each pixel from the closest pixel of river is computed. Distance values closer to 0 are in blue, meanwhile higher distance values are in yellow.*

The integration of this additional layer was used by the classifier to "learn" the spatial proximity between every object in the map and the river. This helped to classify the Typha with higher precision. Indeed, the groundtruths of Typha were located close to the river, and this relation was exploited by the model as a further information, added on top of the spectral characteristics. The classifier learned that Typha's pixels have very low values of distance from rivers, thus using this information in the prediction phase.

This novel approach turned out to substantially help the model to perform a better classification process. Differently from previous studies, where the model learned the spatial constraints from the data [29][30][31][32], we provided information about spatial relationship, obtained from Geographic Information System (GIS) data of the river, as an "a priori" input. To the best of my knowledge, the process of integrating GIS data into the classification algorithm has never been used yet for similar projects. Besides being easier to implement with respect to other state-of-the-art solutions, this method offers the advantage that OpenStreetMap, which creates and makes freely available geographic data [38], providing GIS data of rivers, cities, roads and other classes, for almost every country in the world. GIS data are updated every few days with high precision, thus providing a relevant source for improving environmental assessments and monitoring.

### 3.3.3 Preprocessing of Groundtruths

Another contribution of this work was the implementation of the groundtruth cleaning. A groundtruth is defined as the information collected *in-situ*. In remote sensing specifically, the groundtruths are the data that are collected on the ground by a team of experts, which define with almost certainty a specific class. These data can be used to compare a pixel obtained from a satellite image with the real object that is on the ground in the same specific location. When performing classification on satellite images, the set of groundtruths is typically used as training set. The model learns the features of each different class in order to classify with precision other pixels. The groundtruth are typically given as closed contours, where all pixels inside the area are considered to belong to a single specific class (Figure 3.6).
However, this is not always the case as sometime pixels inside a groundtruth belong to a completely different class. This problem happened with the data that we were provided with. According to the experts, the groundtruths of Typha were difficult to collect, since the plants grow in swamps, impossible to be accessed. To recover the groundtruths, a drone was flown over the marshes and the location of the plants was recorded. However, since plant areas and water areas were not well separated, the groundtruths recovered were not precise (Figure 3.7).

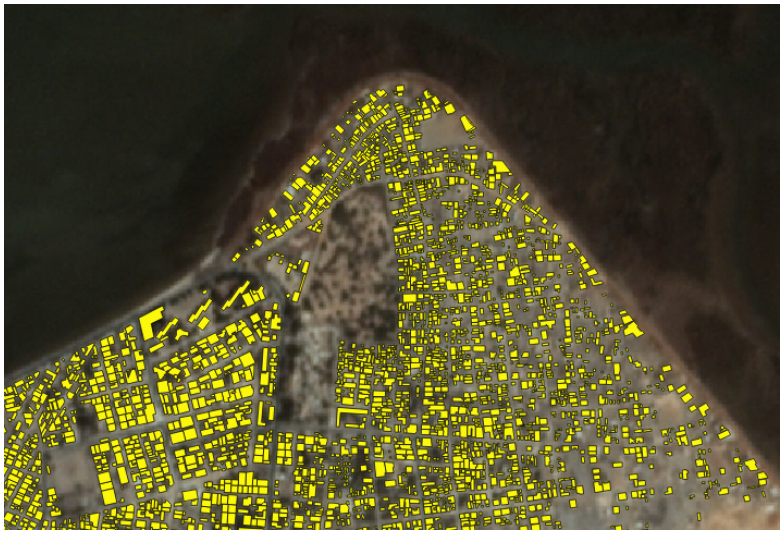


*Figure 3.6: Example of city groundtruth, in yellow. It can be noticed how the groundtruth are provided as multiple different areas.*

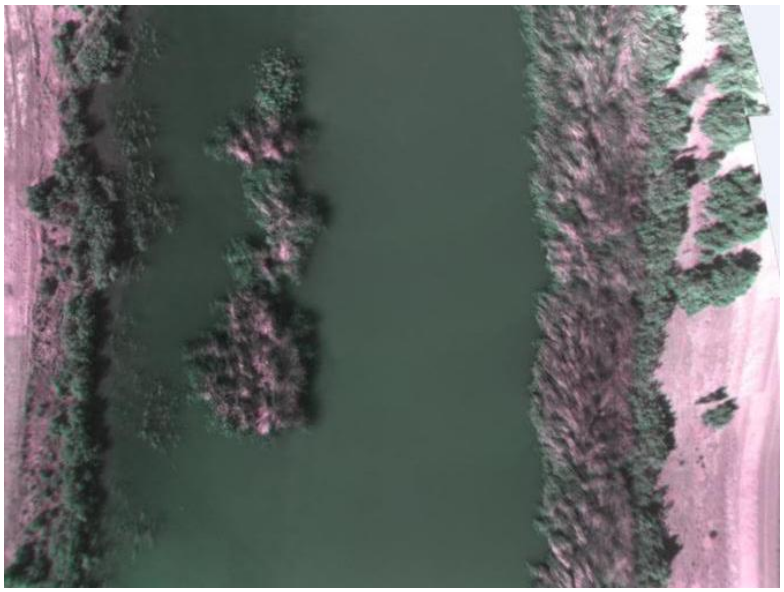


*Figure 3.7: Photo of Typha's plants from the top, taken with the drone, clearly showing that the plants are mixed with the water. Courtesy of [3]*

The goal of the groundtruth cleaning is to identify and remove all the pixels which do not belong to the groundtruth's class, in order to give the model more precise and homogeneous training data, hence guaranteeing that the classifier does not learn from a class features of pixels that do not belong to that class.
The inputs given are:

- An image $I$ of $w \times h \times d$ dimensions, composed by a certain number of pixels $p$
- A label function $L(p)$ which returns the label of the class assigned to the pixel.

- A true label function $\mathcal{L}(p)$ which return the true label of the class to which the pixel belongs to, i.e. the class in the real world to which the pixel belongs to.
- A groundtruth $GT_c = \{\forall p \in GT \mid L(p) = c\}$, which means that for every pixel $p \in GT$ contained in the groundtruth $GT_c$, where $c$ is a specific class, the label $L(p)$ is equal to $c$.
- A set of classes $\mathcal{C} = \{c_1, c_2, \ldots, c_i\}$.
- An unknown number of pixels $r \in GT_c$ for which $\mathcal{L}(r) \neq c$, where $\mathcal{L}(p)$ indicates the true label of the pixel, meaning that the true label of the pixel $r$, that is contained in $GT_c$, is not $c$.

The goal is defined as follow:

$$\forall \text{pixel } \in GT_c \text{ remove pixel from } GT_c \implies \mathcal{L}(pixel) \neq c \tag{3.7}$$

The solution we proposed expoited the k-Nearest Neighbour algorithm, that is a classification algorithm often used in machine learning. It is (i) non-parametric, meaning that it tries to fit the training data in the best possible way- i.e. the model structure is determined from the data - and (ii) lazy, meaning that the training phase is not present, but the data are "memorized" from the input. The k-NN algorithm is based on feature similarity, so it tries to predict a given data searching which training example is the most similar.

As already stated, the groundtruths of Typha contained pixels whose true label was river. Knowing this relation, we decided to exploit the groundtruths of two classes as training inputs: vegetation and water. The k-NN algorithm learned the features of these two classes and, when presented with pixels belonging to the Typha's groundtruth, was able to discern if the pixel was more similar to the vegetation class or to the river class. The pixels which were considered more similar to the river were removed from the Typha's groundtruth, thus creating a dataset more precise and accurate, composed of pixels belonging only to Typha.
The algorithm is explained below:

**Algorithm 2** Groundtruth Cleaning with k-NN

**procedure** FITTING DATA($X, Y$)
    Vegetation and river groundtruths are stored as training data
**procedure** CLASSIFICATION AND REMOVAL($GT_{typha}$)
    **for** $x$ `in` $GT_{typha}$ **do**
        **for** $i$ `in` $X$ **do**
            Compute distance $d(X_i, x)$ ▷ i.e. compute distance between $x$ and all elements of $X$
        Compute set $N$ of the $k$ elements with smallest distance $d$
        **for** $j$ `in` $N$ **do**
            Return $Y(j)$ ▷ i.e. return the class label of the $j$ element
        Count the number of labels of the set $N$
        Assign $x$ to the class which has the majority of labels
        **if** $x$ is classified as river **then**
            Remove $x$ from $GT_{typha}$

Where $X$ is the vector of training data, $Y$ the class labels for each element of the training data; $x$ is the element that needs to be classified, and $k$ is the number of nearest neighbour considered.

Knowing that each pixel has its own vector of values (or features) along the depth, the similarity between two pixels is calculated as a simple distance between their two vectors. In this thesis, the function used to compute the distance was the euclidean formula:

$$E(x, y) = \sqrt{\sum_{i=0}^{d}(x_i - y_i)^2} \tag{3.8}$$

Where $x$ is the first vector, $y$ is the second vector and $d$ is the number of features of the vectors (depth of the image).
This distance is computed between every item that needs to be classified and every item in the training dataset. Then the $k$ closest data points are selected, and the item is assigned to the class which has the "majority of votes".

Using this method, pixels contained in a groundtruth to which they do not belong are removed. This ensures a better classification and better performances, since the neural network models that were developed learned on a training dataset whose groundtruths were composed by pixels belonging to a single class. An example of the result of the algorithm is presented in Figure 3.8.

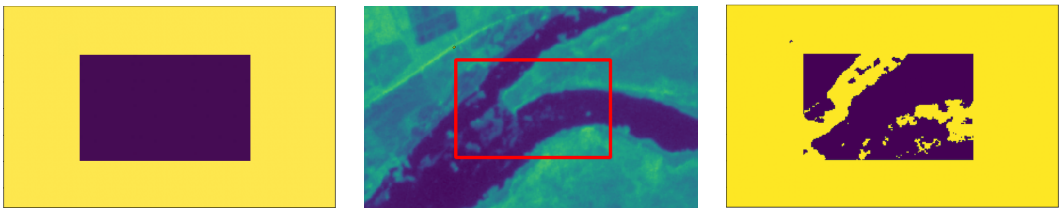


*Figure 3.8: Results of the algorithm applied to a fictional groundtruth of Typha. On the left is presented the groundtruth of Typha, where purple pixels belong to the groundtruth and yellow pixels do not. In the center there is the NIR band of the area, where the red square delimites the groundtruth. Blue pixels belong to the river, while greener pixel to the Typha. On the right is presented the groundtruth after the removal of the pixels found to belong to river. It can be noticed that the majority of pixels that were blue were removed from the groundtruth.*

### 3.3.4 Blob Split Method for Training and Test Set Partioning

A new way of partitioning the training and test dataset was used in this thesis. In machine learning, it is a common practice to split the input data randomly between training and test set. The training set is used as input to train the model. The test set is used to validate the parameters obtained in the training set and to compute the errors. The latter process is performed in order to assess if the model is able to generalize what was learned during the training phase, and predict a set of data that was not used as training.

In classification of remote sensed images, the input data is the set of groundtruths. The pixels therein contained are typically partitioned between training and test set in a random way. In this thesis, instead of using only a random split, we propose a new method, which consists in dividing the groundtruths in different blobs, i.e. agglomerates of contiguous pixels that belong to the same class, as displayed in Figure 3.9.

*Figure 3.9: Groundtruths of vegetation divided in blobs: each color corresponds to a different blob.*

Then all the blobs of a same class were divided, to be later used, either in the training or the test set, using a 2:1 ratio, meaning that 66% of the blobs of a specific class were

selected as training set, while the other 33% were selected as test set. This process was made because we reasoned that different areas of the same class could have slightly different features. The idea was therefore to assess wether the model would be strong enough to predict areas belonging to the same class even when pixels of that specific area were not used for training.

In order to achieve this goal, the inputs given are:

- A map $M$.
- A set $\mathcal{C} = \{c_1, c_2, \ldots, c_i\}$ of classes where $i$ denotes a different class.
- A certain number of pixels $p_i$ located in the map, labeled as one of the $i$ classes. Pixels labeled with a '0' do not belong to any class.

The formal definition of the problem is the following:
*Given a map $M$ composed by a set of pixels $p_i$ which belong to a class $i$, join as a single agglomerate all the pixels which are contiguous (connected) and that are part of the same class $c_i$.*

The algorithm that was implemented in order to split the groundtruths into blobs and then partition them between training and test set is the following:

**Algorithm 3** Blob Split and Training/Test Partitioning

**procedure** BLOB CREATION($M, \mathcal{C}, p_i$)
    Initialize label_counter as a zeroes vector of $i$ dimension
    Initialize the new label map as $L$, with same dimensions as $M$ composed by only zeroes
    Initialize variable *pixel_class*
    **for each** `pixel` $p$ `in` $M$ **do**
        **if** $p \neq 0$ **then** ▷ i.e: pixel belong to a class
            *pixel_class* $= \mathcal{C}(p)$ ▷ i.e: class $k$ of pixel $p$
            Find a set $S$ of pixels connected to $p$ which belong to the same class $\mathcal{C}(p)$
            **for each** *elem* `in` $S$ **do**
                get *elem* coordinates $(x, y)$
                $L(x, y)$ = label_counter[*pixel_class*] ▷ i.e: set in map $L$ the value of label_counter
                remove *elem* from $M$
            label_counter[*pixel_class*] ++ ▷ i.e: increase counter value
    **return** $L$
**procedure** TRAINING/TEST BLOB PARTIONING(L,$\mathcal{C}$)
    **for** $c$ `in` $\mathcal{C}$ **do**
        Find all blobs $B$ that belong to class $c_i$ in matrix $L$
        Select randomly 66% of them, and assign each pixel to the training set
        Assign each pixel of the remaining 33% to the test set
    **return** Training Set, Test Set

Using this algorithm, we first created a new map where the groundtruths were divided in blobs. Subsequently the blobs were partitioned between training and test set in a random way.

# Chapter 4

# Dataset and Preprocessing

In order to obtain a segmented image, several steps have been done to prepare the datasets used as inputs for the models. These steps are described in Figure 4.1. The preprocessing steps were divided in three main branches:

1. Satellite data download and manipulation, as well as creation of NDVI and NDWI indexes
2. Creation of the distance from the river matrix
3. Groundtruth data collection and manipulation

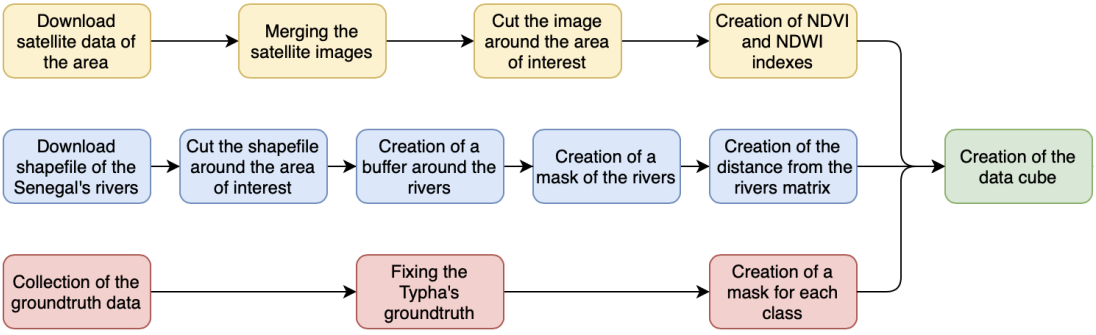



*Figure 4.1: Pre-processing steps realized to create the input data for the models.*

These steps are described in detail in the following sections. In particular, Section 4.1 explains how the satellite images and the groundtruth were retrieved. Section 4.2 reports the preprocessing steps actuated to manipulate the input data in order to make them usable, and the creation of additional data. Then in Section 4.3 the creation of the layer containing the spatial constraints is discussed. Section 4.4 reports the groundtruth cleaning method applied to this dataset. In Section 4.5 the approaches used to partition the data into training and test set are explained. In Section 4.6 the output class of the models are defined. Finally in Section 4.7 the softwares used in this thesis are introduced.

## 4.1 Datasets

### 4.1.1 Study area

The case study area is situated around the Gaston Berger University in Senegal (Figure 4.4). The coordinates of the university are 16°3'48" N, 16°25'33" E. This area was chosen because the MASTR-SLS project (described in Section 3.1) aims to monitor the spread and apply containment measures for Typha, an aquatic flowering plant with invasive potential and environmental impacts, in this area.

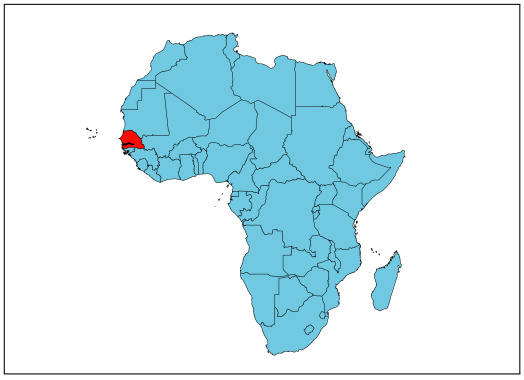


*Figure 4.2: Location of Senegal in Africa.*

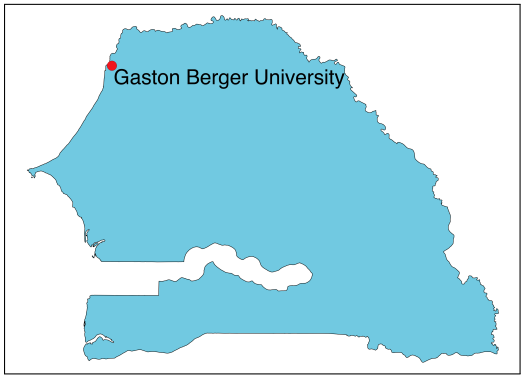



*Figure 4.3: Location of the university.*


*Figure 4.4: Map with the study area (data: Google Maps).*

### 4.1.2 Satellite image

The satellite images of the area were acquired by PlanetScope, a Planet and ESA (European Space Agency) satellite constellation. Each PlanetScope satellite is a CubeSat 3U form factor (10 cm by 10 cm by 30 cm). The complete PlanetScope constellation, of approximately 130 satellites, is able to image the entire land surface of the Earth every day (equating to a daily collection capacity of 200 million km2/day) [39]. The acquisition date was the 24th of March 2019. The Planetscope satellites provide data in 4 bands

(Red, Green, Blue, Near InfraRed), with a resolution of 3 meters (Table 4.1).

| Spectral Band | Wavelenght | Resolution |
|---|---|---|
| Blue | 455-515 nm | 3 m |
| Green | 500 - 590 nm | 3 m |
| Red | 590 - 670 nm | 3 m |
| NIR | 780 - 860 nm | 3 m |

*Table 4.1: Information of PlanetScope satellite [39].*

### 4.1.3 Groundtruth Data

The study area contains different land classes, but in this thesis it was decided to focus on 6 classes: river, Typha, soil, agriculture fields, city and vegetation. The groundtruth data for each of these layers were needed to train and verify the models. The groundtruths were collected by a team of experts, that performed on site surveys and provided GIS data of each class for the area to be examined. The groundtruth data of all Senegal's buildings were downloaded from Open Street Map[1], but only the buildings relative to the area of interest were selected and used.

Each groundtruth was provided as a shapefile layer, that was later transformed in a binary mask of 3137 × 5238 pixels. Each pixel was labeled as either 0 if it was not belonging to the class, or 1 if it was.

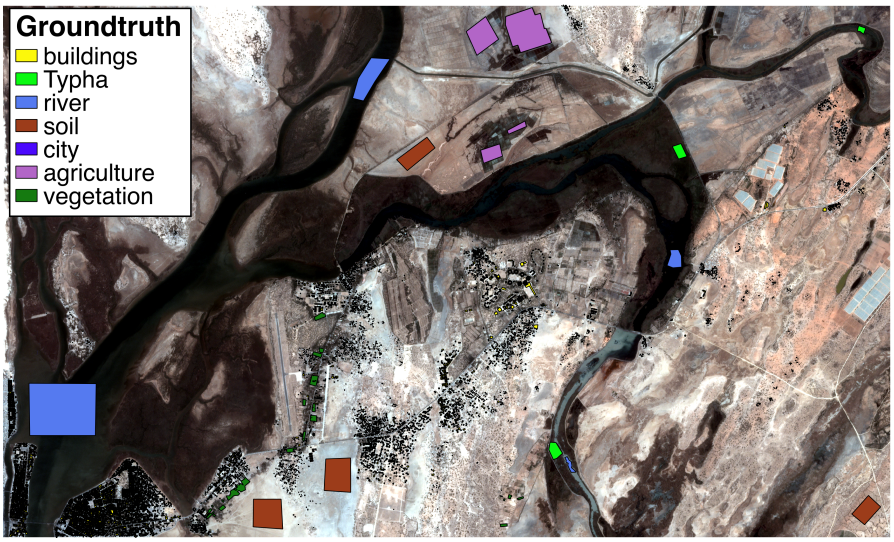



*Figure 4.5: Groundtruth layers overlapped to the satellite image of the area.*

[1]https://download.geofabrik.de/africa/senegal-and-gambia.html

## 4.2 Preprocessing

### 4.2.1 Image Preparation

In order to cover the area of interest, a single satellite image was not enough, so three different images taken in the same day, covering the whole area, were downloaded from Planet website[2]. Then, with the use of QGIS, the images were merged in a single raster and then cut around the study area (Figure 4.6).

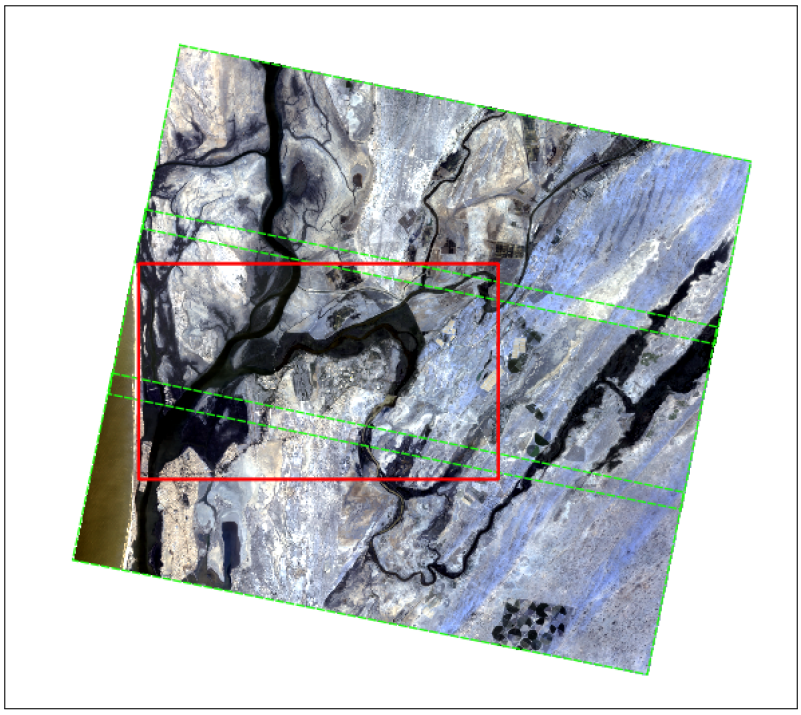


*Figure 4.6: Visualization of the three satellite images, boundaries in green. The area of interest is inside the red box.*

The final image had a size of $3137 \times 5238 \times 4$ pixels, which corresponded to roughly 9×15km. The third dimensions channels were, in order, Blue, Green, Red and Near InfraRed (NIR).

### 4.2.2 Additional Data

The available data on the case study area were integrated with additional information. Two layers were created from a combination of bands: an NDVI layer and an NDWI layer. The integration of these layers was studied to assess how their inclusion changed the results of the predictions of the models used.

**NDVI** The normalized difference vegetation index (NDVI) is a common index that gives the quantitative estimation of vegetation growth and biomass [40], that uses a

[2]https://www.planet.com/products/planet-imagery/

combination of the Red and NIR channels:

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

This index is particularly useful when trying to highlight active and healthy vegetation. Its value ranges from -1 to 1. High values of NDVI indicate presence of healthy green vegetation, because vegetation reflects more in the NIR channel and less in the Red channel, whereas lower values indicate non-healthy vegetation, bare soil and buildings. Very low values are typical for water bodies.

**NDWI** The normalized difference water index (NDWI) is a common index used for water body mapping. It uses a combination of the Green and NIR channels:

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

This index is particularly useful when trying to highlight water bodies, because water reflects almost zero in the NIR channel and has the highest reflection in the Green channel.

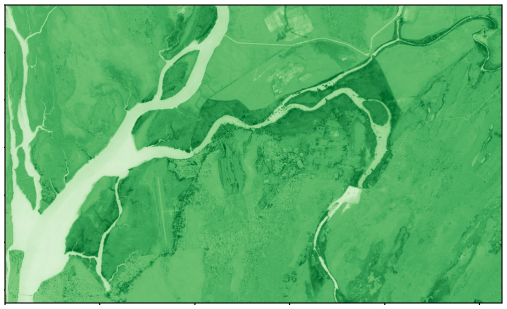


*Figure 4.7: NDVI index clearly shows where the vegetation is located (in dark green).*

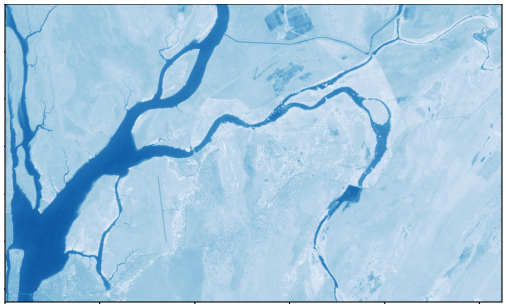


*Figure 4.8: NDWI index clearly shows where the water bodies are located (in dark blue).*

## 4.3 Integration of GIS Data

The GIS data of Senegal's rivers were downloaded from the up-to-date OpenStreetMap website[3]. The data contained the geometries of every waterway in Senegal (Figure 4.9). Only the rivers of the area around Saint Louis were selected (Figure 4.10).

The data of the rivers were provided as a single line feature passing through the center of the river. This representation does not describe the reality, since the width of the river is often many meters long. For this reason, a small buffer was created around each of these lines, providing a more realistic representation of the rivers (Figure 4.11).

[3]https://download.geofabrik.de/africa/senegal-and-gambia.html

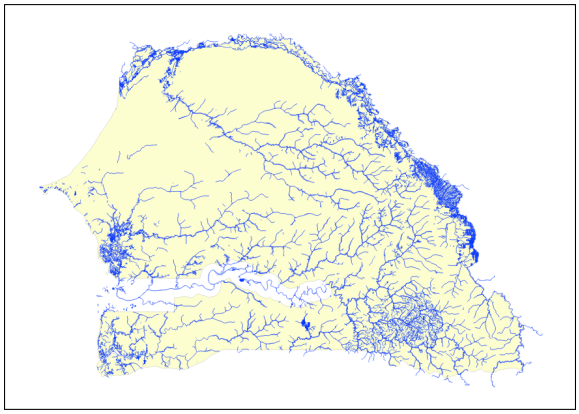


Figure 4.9: Senegal's rivers downloaded from OpenStreetMap.

Figure 4.10: River layer of the St. Louis area. The rivers layout is provided as a line.

Figure 4.11: River layer of the St. Louis area after the buffer operation.

In order to compute the distance from the river, the image was binarized, meaning that each pixel that was part of the river was converted into a '1', all the others pixels were set as '0' (Figure 4.12).

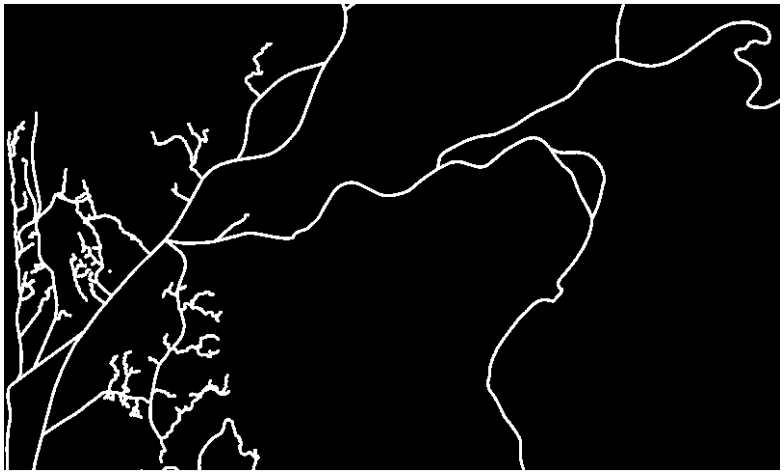


Figure 4.12: Mask of the river layer, where the pixels belonging to the river have value equal to '1', while the pixels not belonging to the river have value equal to '0'.

Then the algorithm described in Section 3.3.2 was applied to the newly created mask. In this way, a matrix where all the pixels have a value equal to the euclidean distance

from the closest pixel of a river was created (Figure 4.13). This matrix was used as an additional layer (Section 5.3.1), given as input to the neural network, ensuring that the proximity relationship that involves the Typha and the river was given as a-priori information.

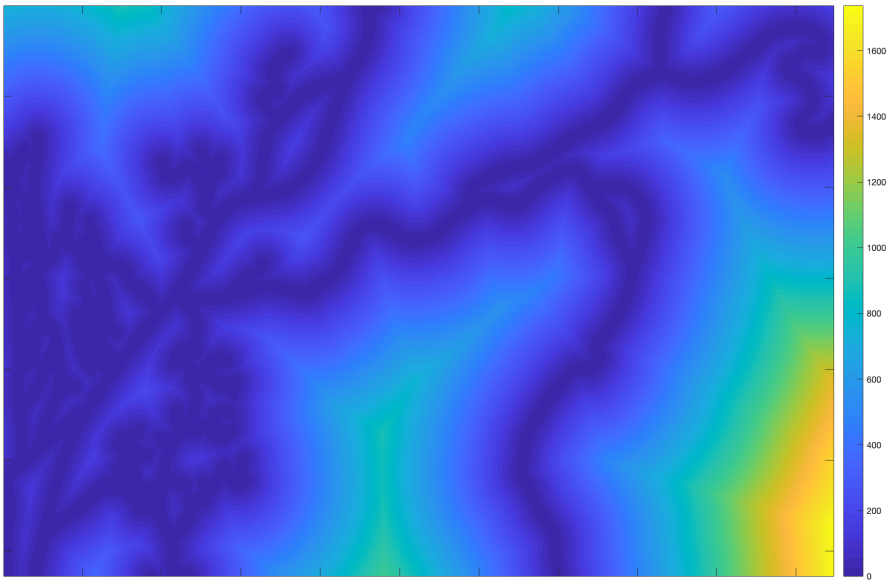



*Figure 4.13: Image representing the distance of each pixel from the closest river. Blue represents values closer to zero, while the highest values appear in yellow.*

The final data consisted of $3137 \times 5238 \times n_c$ pixels. The number of channels $n_c$ changed depending on the case studied. The channels used were, in order: Blue, Green, Red, NIR, NDVI, NDWI, distance from river, as seen in Figure 4.14

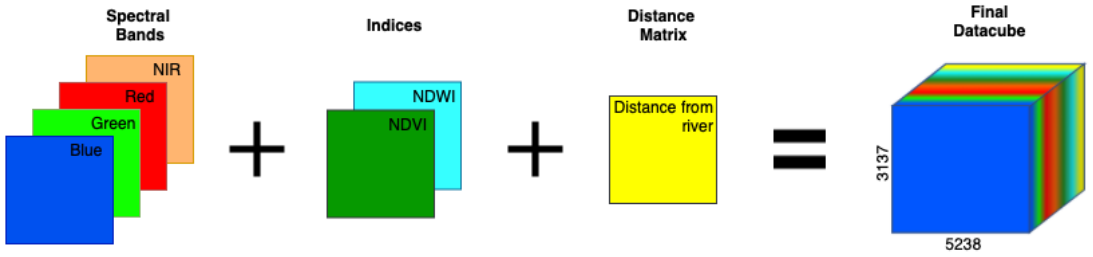



*Figure 4.14: Datacube creation.*

## 4.4 Typha's Groundtruth Cleaning

As already stated in Section 3.3.3 the groundtruths of Typha were difficult to collect, since the plants grow in swamps, difficult to be accessed. To recover the groundtruths a drone was flown over the marshes, and the location of the plants was recorded. However, since plant areas and water areas were not well separated, the groundtruths recovered were not precise. Comparing the data with respect to the actual satellite image, it was

possible to see that some areas labeled as Typha were, in reality, water bodies. This was noticed by checking the NIR band (Figure 4.15), where clearly appeared that the river's pixels had very low values (leaning towards a dark blue color), meanwhile the vegetation's pixels had higher values (leaning towards a green/yellow color).

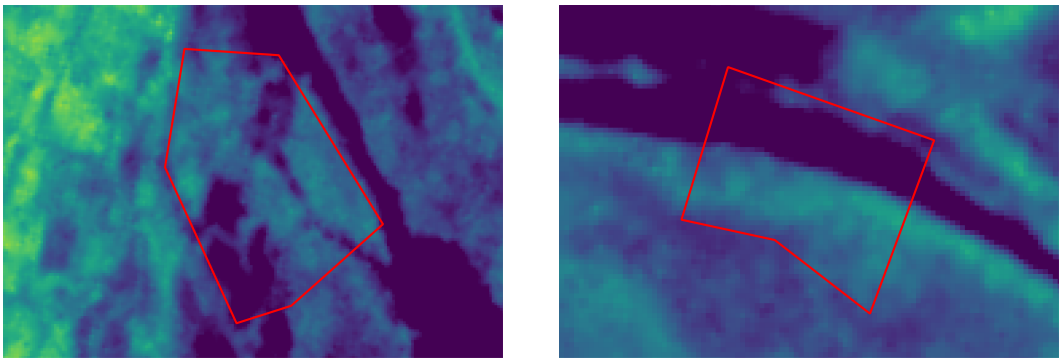


*Figure 4.15: NIR band of two groundtruths of Typha inside the red box: it is evident that some of the pixels do not represent Typha, but belong to the river.*

In order to remove the non-Typha's pixels from the groundtruth, it was decided to use the algorithm described in Section 4.4 (Algorithm 2). To fit the data two classes were chosen: river and vegetation. The groundtruths of these classes were used as training data. Then the pixels of the Typha's groundtruth were classified with respect to either class, in such a way that the pixels of the groundruth that were more similar to the river were removed. The new groundtruths obtained showed that the pixels that looked like water were correctly removed (Fig 4.16).

This process was made in order to have more reliable and more accurate groundtruth on which the neural network model could train on. In fact, having some completely wrong pixels inside the Typha's groundtruth could have worsened the final classification.

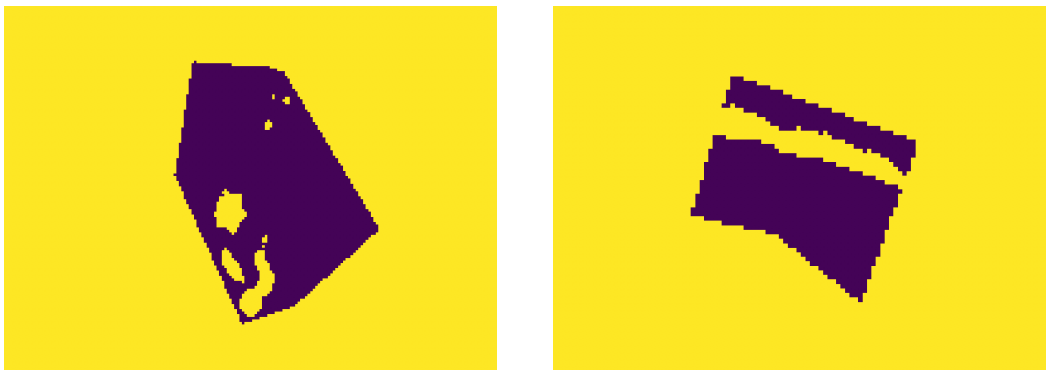


*Figure 4.16: Groundtruths of Typha showing the removal of pixels found to belong to the river.*

## 4.5 Training and Test Partitioning

As already introduced in Section 3.3.4, in machine learning is a common practice to split the input data randomly between training and test set. There are different ratio combinations that can be used, among which one of the most common is the 4:1 ratio, that means 80% of the pixels are chosen as the training set, and 20% as the test set. This ratio is typically used in presence of high amounts of data. Since the groundtruths provided were not enough to justify this choice, it was decided to compare two different methods for training and test set selection.

1. A random split of the groundtruth data between training and test with a ratio of 2:1, meaning 66% of groundtruth's pixels were selected as training set and 33% as test set

2. The blob split method described in Section 3.3.4. All the blobs were divided between training and test set using a 2:1 ratio, meaning that 66% of the blobs of a specific class were selected as training set, while the other 33% were selected as test set.

   Since the groundtruth provided of Typha consisted of three blobs, when the blob split method was performed, two blobs were selected for the training set, and the last one was used as test set. The three different cases that are presented later corresponds to the distinct combinations of the blobs selected as either training or test set.

These choices were applied to the neural networks models. Furthermore, during the training phase of the model, the training set was also split between training set and validation set with a ratio of 4:1. The validation set was composed of 20% of the pixels, and it was used to validate the model performances, meaning that the set of weights which produced the smallest error with respect to the validation set was selected.

For the convolutional neural network model, it was decided to use the same blob split method adopted for the neural network model. The blobs of each class were split with a ratio of 2:1, meaning that 66% of the blobs of a specific class were selected as training set, while the other 33% were selected as test set. This choice was made to test if the model was able to learn on a blob and then predict correctly on a different blob, that might have non-similar characteristcs.

## 4.6 Number of Output Classes

When performing a classification, the number of output classes defines how many classes the model is going to predict. Usually, but not necessarily, is equal to the number of

classes provided as groundtruth. In this thesis, it was decided to study two different cases, a 2-class classification and 6-class classification.

**2-class Classification** In the first case it was decided to perform a 2-class classification, where the two classes were represented by Typha and all the other groundtruths (river, vegetation, soil, agriculture, city) merged in one single class. This analysis was performed to get more insights into the classification of the Typha, and to see if the model was able to perform a precise classification. Furthermore we wanted to determine whether the integration of the spatial constraints improved the classification.

**6-class Classification** In the second case, it was decided to perform a 6-class classification, using all the classes given in the groundtruths. This analysis was performed to get more information on how the pixels of the Typha were classified, and to reveal possible misclassification errors made by the model. Furthermore, the tests were performed to determine if the results obtained in the 2-class classification were convalidated even in this case.

The workflow of the case studied is presented in Figure 4.17.

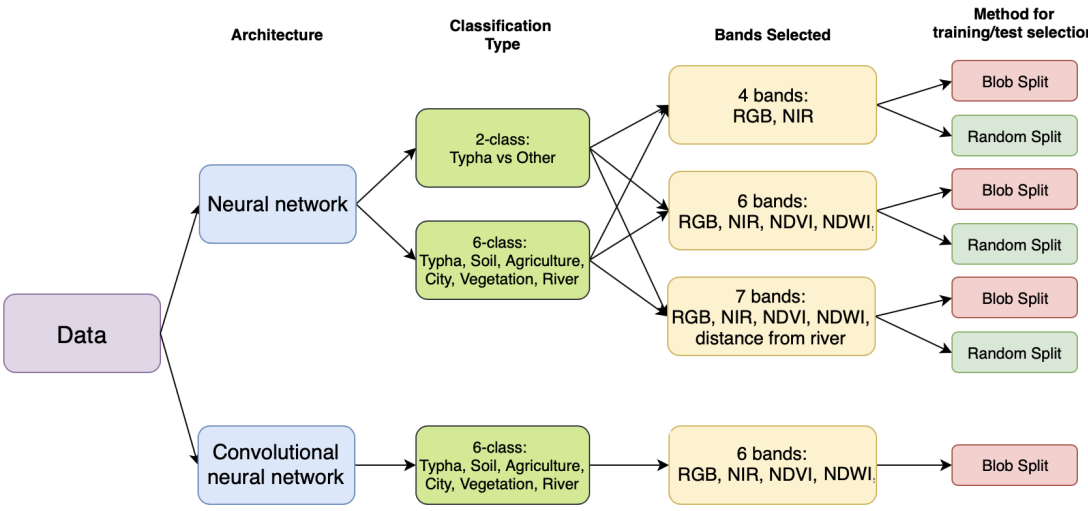



*Figure 4.17: Workflow of the case studied.*

## 4.7 Software

To perform the analyses described in this thesis, the following softwares were used:

**QGIS**[4] QGIS is a professional geographic information system application that is Free and Open Source Software (FOSS). QGIS allows users to analyze and edit spatial information, in addition to composing and exporting graphical maps. QGIS supports both raster and

[4]https://qgis.org/en/site

vector layers; vector data are stored as either points, lines, or polygon features. Multiple formats of raster images are supported, and the software can georeference images. Web services are also supported to allow the use of data from external sources.

**Scikit-learn**[5] Scikit-learn is a machine learning library built for Python that provides simple and efficient tools for data mining and data analysis, as well as several different machine learning techniques such as classification, regression and clustering [41].

**Keras**[6] Keras is a high-level API for neural networks, written in Python. It was used on top of Tensorflow, an open source platform for machine learning. Keras allows to build personalized neural networks models using pre-built blocks, for example different kinds of layers, activation functions and optimizers [42].

**Colaboratory**[7] Google Colaboratory (also known as Colab) is a cloud service based on Jupyter Notebooks. It is especially useful because provides a runtime fully configured for deep learning and free-of-charge access to a robust GPU (Tesla K80 GPU of 12.72 GB), that was needed when training convolutional neural networks [43].

The neural network models were run on a MacBook Pro with 8GB of RAM and an Intel Core i5.

[5] https://scikit-learn.org
[6] https://keras.io
[7] https://colab.research.google.com

# Chapter 5

# Models Setup/Design

Obtaining classification from images requires several passages. The choices that have to be made concern mainly the selection of the training and test data sets and the model architecture design. This chapter starts with a detailed overview of the design steps required before the training of a neural network (Section 5.1). In Section 5.2 the problem of class imbalance is introduced and the methods adopted to overcome it are presented. Then in Section 5.3 the neural networks models used in this thesis are presented, and the main implementation decisions are reported. Finally in Section 5.4 the convolutional neural network model used in this thesis is presented, and the implementation choices are explained in details.

## 5.1 Overview of Design Steps

An overview of the design steps for the neural network model and the convolutional neural network model is presented below. The steps necessary for creating, training and evaluating the neural network can be resumed as:

- Preprocessing of the data
- Partition in training and test set
- Training the model (with intristic split into training and validation of the training set)
- Network prediction and evaluation metrics calculation
- Converting network output into a segmented image

For the convolutional neural network model, the steps can be summarized as:

- Preprocessing of the data
- Partition in training and test set

– Training the model with patches

– Network prediction on patches and evaluation metrics calculation

– Converting network output in a segmented image

This chapter will mainly focus on the steps needed for setting up the networks and on the description of the adopted methods.

## 5.2 Class Imbalance

### 5.2.1 Class Weights

The groundtruths collected were very imbalanced, that means that they consisted of different number of pixels (Table 5.1). Under these circumstances, the model encountered problems to classify the least represented classes. One way to solve this issue was to add class weights to the loss functions (Section 2.2.3). Indeed, as Keras documentation states: "This can be useful to tell the model to "pay more attention" to samples from an under-represented class" [42].

| **Class** | **Number of pixels** |
|---|---|
| River | 154'652 |
| Typha | 11'032 |
| Soil | 90'879 |
| Agriculture | 82'319 |
| City | 141'397 |
| Vegetation | 18'632 |

*Table 5.1: Number of pixels for each groundtruth class.*

The weights for each class were computed using this formula:

$$w_k = \frac{n}{k * n_k} \tag{5.1}$$

Where $n$ is the total number of pixels, $k$ is the number of classes and $n_k$ is the number of pixels belonging to the $k$ class. Note that the $n$ and $n_k$ are computed from the training set. Since the training set was defined randomly each time the model was run (as described in Section 4.5), the weights were different for each iteration, according to the size of the training set and the number of pixels for each class therein contained.

It was decided to add the class weights to the loss function since the Typha's groundtruths contained less pixels than any other class, while being the most important class to classify. Furthermore it was decided to compute them using the training set data, since it contained the number of pixels on which the model was learning on.

### 5.2.2 Loss Function: Weighted Categorical Cross Entropy

The class weights were used to change the loss function, in order to weight more the missclassified elements of a smaller class. The loss function used was the categorical cross entropy (Equation 2.3), and its weighted version is:

$$L_w(p, q) = L(p, q) * w \tag{5.2}$$

Where $L(p, q)$ is the loss function and $w$ is the vector of weights, defined in Equation 5.1.

The weighted categorical cross entropy was used as loss function for both the neural networks and the CNN.

## 5.3 Neural Network Models

### 5.3.1 Model architecture

The neural network was designed using Keras [42]. Figure 5.1 shows the architecture of the model that was used for training. The network had 5 layers, composed by:

- One input layer of variable number of neurons (depending on the number of channels used). The activation function used was the ReLU.
- Three hidden layer composed by 5 neurons each. The activation function used was the ReLU.
- One output layer composed by 2 or 6 neurons, that corresponded to the number of output classes. The activation function used was Softmax.

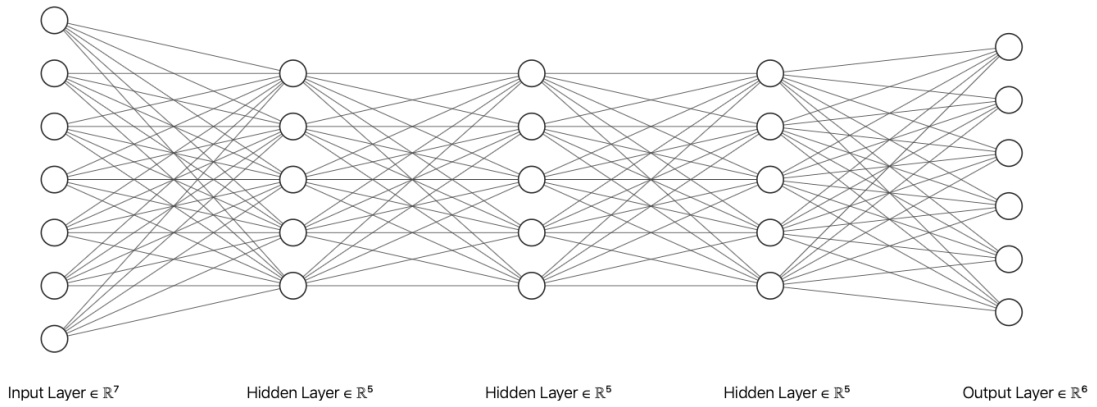



*Figure 5.1: Neural network architecture used in this thesis [18].*

### 5.3.2 Regularization

As stated in Section 2.2.5 a problem that a neural network may suffer is the overfitting of the data. In order to prevent and resolve this issue a Kernel regularizer L2, for each neuron, was used, with the $\lambda$ value equal to 0.01.

### 5.3.3 Gradient Clipping

Another issue that may occur in deep neural networks, is the exploding gradient, that happens when the gradient is *unstable* [44]. The gradient that is computed can get extremely high, if the weights are big enough, and this may cause the model to not learn anymore.

An easy way to fix the exploding gradient is the gradient clipping, that consists in setting a value for the gradients when it goes beyond a certain threshold. In particular, in this thesis the *clipnorm* from Keras, that "clip the gradient if the L2 norm exceed a threshold" [42] was used, with the following formula:

$$gradient = \begin{cases} \frac{gradient * threshold}{\|L2\|} & \text{if } \|L2\| \geq threshold \\ gradient & \text{otherwise} \end{cases} \tag{5.3}$$

In this way the gradient was not able to increase to the point where it could not be handled anymore. The threshold value chosen in this thesis was 1.0

### 5.3.4 Prediction

Once the model was trained, the prediction was carried out. Each pixel of the map was given to the neural network model that produced a probability map - i.e. a vector which contained the probability for the pixel to belong in each class. Each pixel was assigned to the class with the highest probability, thus producing a semantic map where each pixel was assigned to the class which is most similar to.

## 5.4 Convolutional Neural Network model

### 5.4.1 Architecture

The convolutional neural network was designed using Keras. The architecture used for this model was based on the U-Net network model created by Ronneberger, Fischer, and Brox [32] and is displayed in Figure 5.2.

The network has 5 contracting layers, supplemented by other 5 upsampling operators, giving the "U" shape where the model takes the name from. The contracting path resembles a typical architecture from a convolutional network, where each block consists of 2 mini-blocks, composed of a Convolutional Layer, a Batch Normalization Layer,

and a ELU activation function. The last layer of the block is a MaxPooling layer. The contracting blocks are concatenated with simmetric upsampled ones, except for the MaxPooling layer that is converted into a UpSampling layer. The output is given by a final Softmax activation layer.

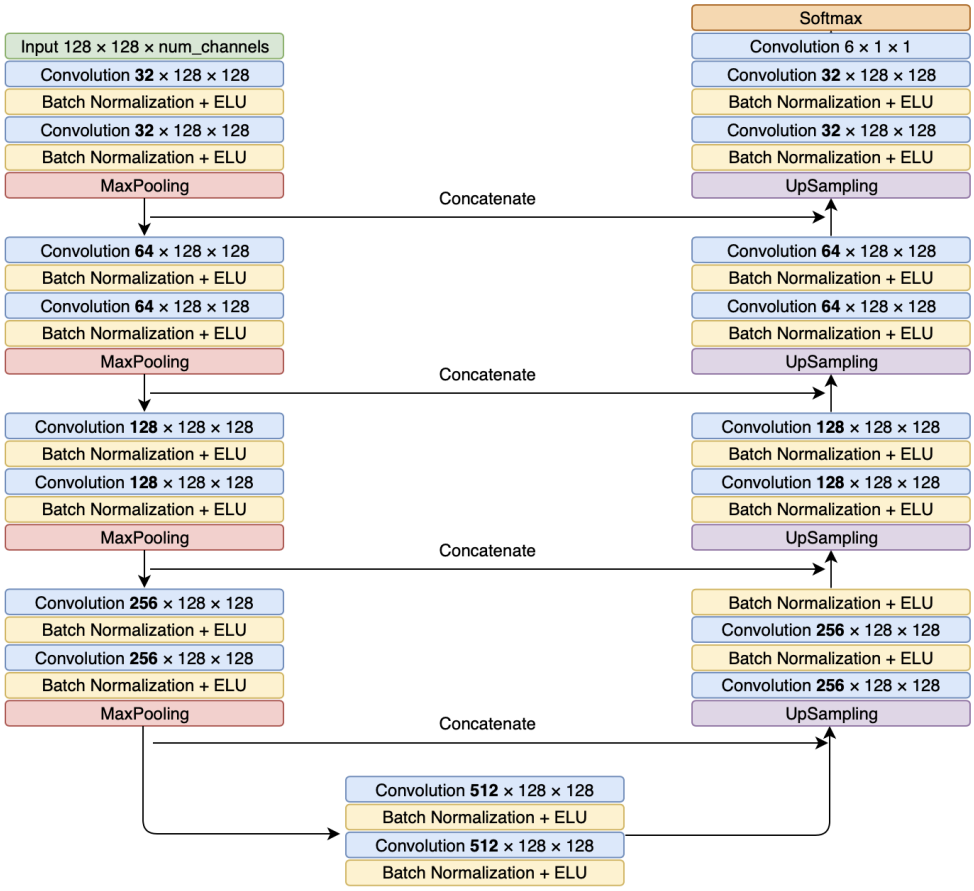



*Figure 5.2: U-Net architecture.*

### 5.4.2 Feature Map Sizes

Every layer of the network produced a feature map size of different dimensions. The patch size used for training had size $128 \times 128 \times 6$. As seen in Table 5.2, the dimensions of the feature map became smaller as it went deeper in the network, while the depth increases. In the 'deepest' point, it was $8 \times 8 \times 512$.

### 5.4.3 Patch Selection Method

As previously stated, each patch had a size of $128 \times 128$ pixels. Usually, in segmentation tasks, patches are selected in random positions throughout the image. Since the groundtruth data were sparse and did not cover the whole image, training patches were

| Layer | Output Size | Layer | Output Size |
|---|---|---|---|
| Input | (128×128×$n_c$) | Concatenate | (16×16×768) |
| Convolution | (128×128×32) | Convolution | (16×16×256) |
| Batch Normalization + ELU | (128×128×32) | Batch Normalization + ELU | (16×16×256) |
| Convolution | (128×128×32) | Convolution | (16×16×256) |
| Batch Normalization + ELU | (128×128×32) | Batch Normalization + ELU | (16×16×256) |
| Max Pooling | (64×64×32) | Up Sampling | (32×32×256) |
| Convolution | (64×64×64) | Concatenate | (32×32×384) |
| Batch Normalization + ELU | (64×64×64) | Convolution | (32×32×128) |
| Convolution | (64×64×64) | Batch Normalization + ELU | (32×32×128) |
| Batch Normalization + ELU | (64×64×64) | Convolution | (32×32×128) |
| Max Pooling | (32×32×64) | Batch Normalization + ELU | (32×32×128) |
| Convolution | (32×32×128) | Up Sampling | (64×64×128) |
| Batch Normalization + ELU | (32×32×128) | Concatenate | (64×64×192) |
| Convolution | (32×32×128) | Convolution | (64×64×64) |
| Batch Normalization + ELU | (32×32×128) | Batch Normalization | (64×64×64) |
| Max Pooling | (16×16×128) | ELU | (64×64×64) |
| Convolution | (16×16×256) | Convolution | (64×64×64) |
| Batch Normalization + ELU | (16×16×256) | Batch Normalization + ELU | (64×64×64) |
| Convolution | (16×16×256) | Up Sampling | (128×128×64) |
| Batch Normalization + ELU | (16×16×256) | Concatenate | (128×128×96) |
| Max Pooling | (8×8×256) | Convolution | (128×128×32) |
| Convolution | (8×8×512) | Batch Normalization | (128×128×32) |
| Batch Normalization + ELU | (8×8×512) | ELU | (128×128×32) |
| Convolution | (8×8×512) | Convolution | (128×128×32) |
| Batch Normalization + ELU | (8×8×512) | Batch Normalization + ELU | (128×128×32) |
| Up Sampling | (16×16×512) | Convolution | (128×128×$n_k$) |

*Table 5.2: Feature map sizes through the network. Notice that the input has size 128×128×$n_c$, where $n_c$ is the number of channels of the datacube (6 in this case), while the output has size 128×128×$n_k$ where $n_k$ is the number of classes (6 in this case).*

selected only in locations that contained at least 1 pixel of a groundtruth (Figure 5.3). In this way a training process that used part of the images containing a groundtruth is assured.

In order to ensure a homogeneous selection of training samples, the number of patches for each class was the same, meaning that with a batch size of 102 and 6 classes, the training patches were 17 for each class. Furthermore, in order to increase the training dataset, augmentation on the patches was performed, by flipping horizontally or vertically or by doing 90° rotations.

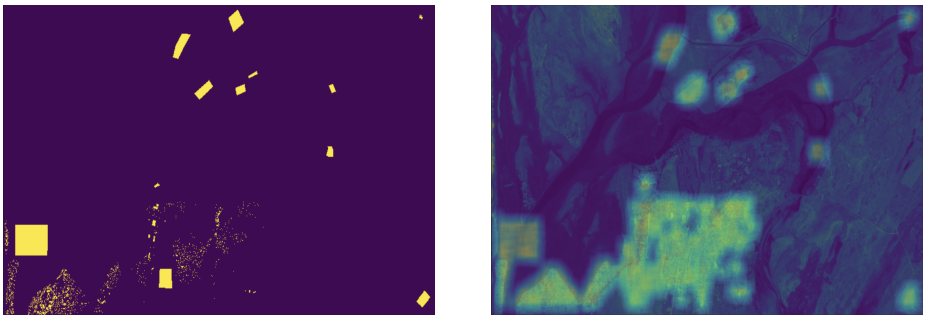


*Figure 5.3: In the left image we can see the groundtruth blobs selected for the training set, in yellow. In the right image we can see where the patches selected are located.*

The batch size selected was 102, meaning that in one epoch 102 patches were created and used for training the model.

### 5.4.4 Prediction on Patches

The prediction was also performed on patches. The whole image was divided into patches of $128 \times 128$ pixels, and the prediction was carried on these patches.

As in the neural network model, the output of the prediction was a patch that contained, for each pixel, the probability values to belong to each class. The highest probability value was selected and the pixel was assigned to the corresponding class.

# Chapter 6

# Experiments and Results

The aim of these experiments was to compare different classification methods with different inputs, and evaluate the advantages and disadvantages of each method. In first place, in Section 6.1 the evaluation metrics used for this study are presented. Then, the experiments performed with the neural network architectures are presented in Section 6.2, where the first test presented is comparison between the cleaning and no-cleaning method. Then, the 2-class and the 6-class methods with blob and random split are presented and the results are compared. In the following Section 6.3 the results obtained with a convolutional neural network architecture are displayed. The comparison of the different methods is then presented in Section 6.4. Final considerations are reported in Section 6.5.

## 6.1 Evaluation Metrics

Evaluation metrics are used to assess how the model is performing. They can be calculated from a confusion matrix, a table used to describe the performance of a classification model on a set of test data for which the true values are known. Each entry of the matrix contains a value which represents the number of pixels of a certain class, classified with a specific label. The diagonal contains all the values that are predicted correctly. An example of a confusion matrix is given in Table 6.1.

| | | Predicted label | | | | | |
|---|---|---|---|---|---|---|---|
| | | River | Typha | Soil | Agriculture | City | Vegetation |
| True Label | River | 50000 | 0 | 0 | 62 | 27 | 0 |
| | Typha | 4 | 3000 | 109 | 0 | 0 | 74 |
| | Soil | 74 | 412 | 28000 | 72 | 18 | 96 |
| | Agriculture | 0 | 24 | 0 | 26000 | 34 | 0 |
| | City | 765 | 0 | 3 | 8 | 33000 | 55 |
| | Vegetation | 9 | 6 | 0 | 0 | 12 | 4000 |

*Table 6.1: Example of a confusion matrix.*

From the confusion matrix it is possible to derive:

**True Positive.** True positives of a certain class are the cases that are predicted in the class they really belong to. From the confusion matrix, the true positive elements are the values that lay on the diagonal.

**False Positive.** False positives of a certain class are the cases that are predicted to be in that class, while they actually belong to another one. From the confusion matrix, the false positive elements of a certain class are the values that lay on the column of that class, excluded the one laying on the diagonal.

**False Negative.** False negatives of a certain class are the cases that actually belong to that class, but they are predicted as belongin to another one. From the confusion matrix, the false negative elements of a certain class are the values that lay on the row of that class, excluded the one laying on the diagonal.

Traditionally, the most important evaluation metrics are Accuracy, Precision, Recall and F1-Score, that are described below.

**Accuracy** The accuracy of a prediction is defined as the fraction of all the correctly predicted values over the total amount of predicted values (Equation 6.1). This metric gives an idea of how close the predicted values are to the actual values.

$$Accuracy = \frac{Correcly\ Predicted\ Pixels}{Total\ Number\ of\ Pixels} \tag{6.1}$$

This metric alone is not sufficient to describe how well the model is predicting the dataset, especially in the case on unbalanced datasets. Additional metrics should be taken into account to evaluate the model performance.

**Precision** The precision of a prediction is defined as the fraction of true positives values divided by the number of true positives plus the number of false positives values (Equation 6.2). This metric gives an idea of how many predicted values for a specific class are actually correct.

$$Precision = \frac{True\ Positive}{True\ Positive\ +\ False\ Positive} \tag{6.2}$$

Taken alone, precision has limitations since it does not consider the class values that are not predicted correctly.

**Recall** The recall of a prediction is defined as the fraction of the true positive values divided by the number of true positive values plus the number of false negative values (Equation 6.3). This metric gives an idea of how many true labels of a class are correctly predicted.

$$Recall = \frac{True\ Positive}{True\ Positive\ +\ False\ Negative} \tag{6.3}$$

Taken alone, recall has limitations since it does not consider how many values of other classes are predicted in the specific class.

**F1-Score** The F1 score of a prediction is the weighted average of Precision and Recall (Equation 6.4). It is a measure that considers both the false positives and false negatives, overcoming the limitation of the Precision and Recall measures when taken singularly.

$$F1\text{-}Score = 2 * \frac{Precision\ *\ Recall}{Precision\ +\ Recall} \tag{6.4}$$

## 6.2 Analysis of the Results obtained with Neural Networks

As previously introduced in Chapter 4, the dataset is composed by a multispectral image, of dimensions $3137 \times 5238 \times n_c$ where $n_c$ is the number of channels selected. For the neural network architecture it was decided to study the differences between 3 different cases:

1. 4-bands: Using only the channels provided by the satellite (meaning: Blue, Green, Red, NIR)
2. 6-bands: Using the channels provided by the satellite joined with the NDVI and NDWI layers
3. 7-bands: Using the channels provided by the satellite, joined with the NDVI and NDWI layers, and further integrated with the distance from the river

Depending on the case, the inputs of the network were changed, and the behaviour of the model was investigated.

### 6.2.1 Results

In this set of experiments the architecture used was the neural network model described in Section 5.3.1. As first instance, the benefits of the cleaning of the groundtruth were

examined. Then two different cases were taken into consideration, the 2-class classification (Typha vs other) and the multi-class classification (using all classes, Typha, Soil, Agriculture, City, River, Vegetation). Each case was exploited using 4, 6 or 7-bands.

A random selection of training/test set pixels was compared to a blob selection, for each case. Furthermore, the metrics were computed on both the Overall dataset - i.e. both training and test set, meaning the whole groundtruth data - and the test set only.

The results were evaluated and the metrics were extracted. In the next sections the results are presented. The accuracy is computed taking into account all the classes, while Precision, Recall and F1-Score were estimated only on the Typha. Each different case was run three times, and the tables present the arithmetic mean of these tests. The singular tests are presented in Appendix A.

The blob split method was run three times with different combinations of Typha blobs, as stated in Section 4.5. For the random split method, each test was performed three times in order to generate a comparable case study and obtain more robust and solid metrics.

### 6.2.2 Groundtruth Cleaning

As previosly introduced in Section 4.4, it was noticed that the groundtruths were not precise. Thus, a way to remove the pixels that belong to the river from the Typha's groundtruth was exploited.

Below are reported the images obtained from the experiments performed with 2-class 6-bands blob split classification with and without cleaning. Only two out of the three cases are described, because they report the most significative difference. The single experiments executed on groundtruth which were not cleaned are reported in Appendix A.1 and A.8.

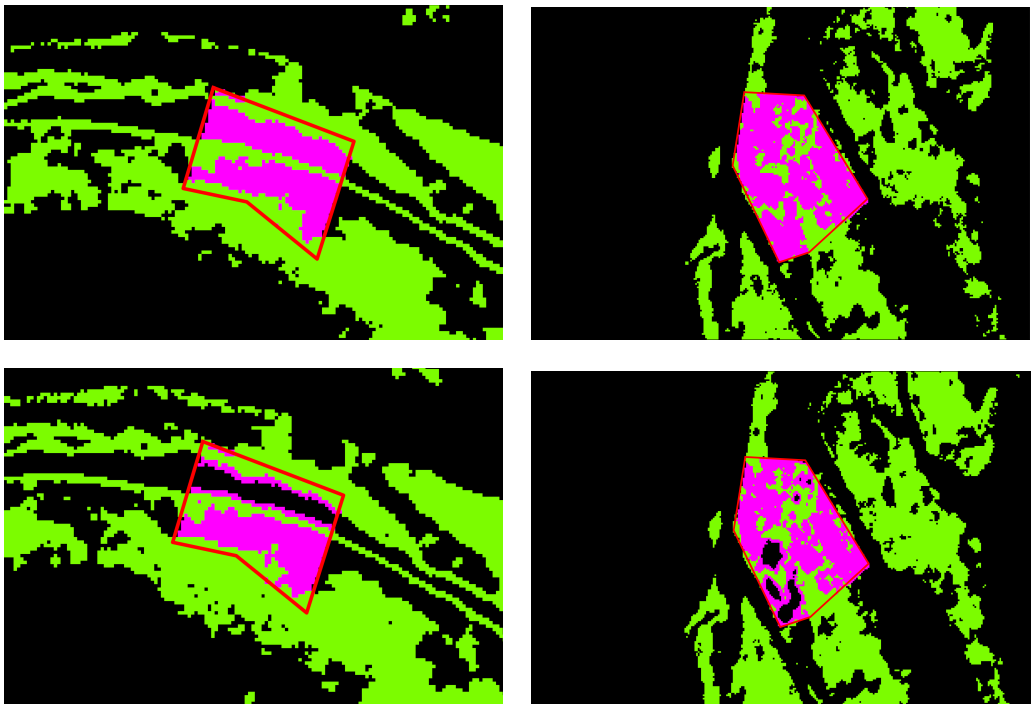


*Figure 6.1: Figure showing how the Typha blobs used as Test Set are classified. In the upper part are the no-cleaning results, below are the cleaning result. The left case is referred to the first blob case, while the right one to the third blob case. The red square delimites the groundruth, pixels classified as Typha are green, pixels that belong to the Typha groundtruth but are not classified as Typha are in purple. The black pixels inside the red square are the pixels removed from the Typha groundtruth with the cleaning method. It can be deduced that these pixels were wrongly labeled pixels inside the groundtruth.*

### 6.2.3 2-class Classification

In this set of experiments, a 2-class (Typha and Other) classification was performed. All the classes that were not Typha, meaning river, soil, agriculture fields, city and vegetation, were merged into a single class. The general goal of these experiments was to classify at best the pixels of Typha, and to assess wether the integration of the spatial constraints affected the results.

The results are presented in the sections below. The first section shows the results when the blob split method was used. The second section shows the results when the random split method was used. The third section presents a comparison between the random and blob split methods. The details of each experiment, and the confusion matrices, are reported in Appendix A.2, A.3, A.4, A.5, A.6, A.7.

#### Blob Split

This set of tests was performed with the blob split method, with either 4, 6 or 7 bands. The metrics are computed for each of these cases on both the Overall dataset and the Test Set only. The results are reported in table 6.2.

| | 4-bands | | 6-bands | | 7-bands | |
|---|---|---|---|---|---|---|
| | Overall | Test Set | Overall | Test Set | Overall | Test Set |
| Accuracy | 99.202 | 99.167 | 99.165 | 99.141 | 99.199 | 98.577 |
| Precision Typha | 71.684 | 58.020 | 71.418 | 57.963 | 76.195 | 53.852 |
| Recall Typha | 84.924 | 80.895 | 83.361 | 78.165 | 84.127 | 39.778 |
| F1-Score Typha | 77.374 | 64.186 | 76.522 | 63.295 | 78.138 | 44.394 |

*Table 6.2: Results with the blob split method.*

The analysis of this table reveals that:

– There is a significant difference between the Overall and Test Set results, when examining each band case singularly, especially for the Typha's metrics. This is expected, since when the Test Set only is considered, the model classifies an area of Typha that was not used for training. These differences indicate that the model is not able to generalize well, especially when the training and test blobs have slightly different features. Figure 6.2 shows the differences, for each band, among every blob of Typha. Given that i) the distance from the river is the value that changes most between the three cases and ii) differences are more noticeable when 7-bands are used, it can be concluded that the distance from the river plays an important role in the blob split classification scenario.

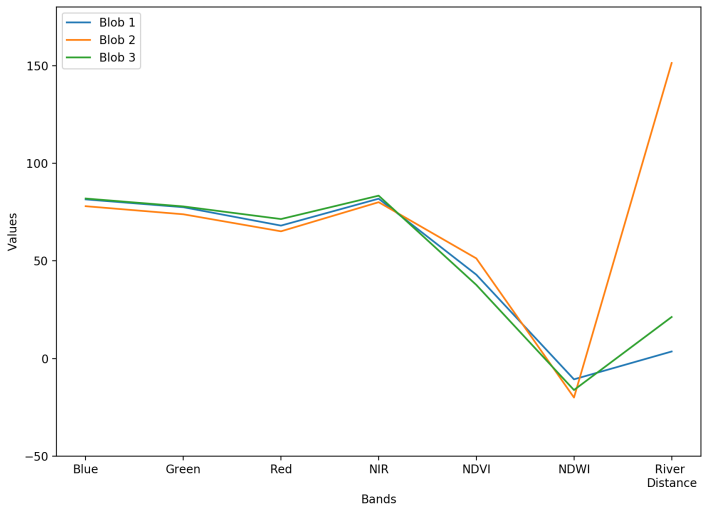



*Figure 6.2: Mean values of each Typha blob.*

– Considering the three different cases and comparing the results on the Overall set, it can be noticed that the addition of more bands does not improve the classification. In fact, the Accuracy remains the almost same, and the metrics that refer to the Typha are comparable, with a slight increase of the F1-Score of ∼2% of the 7-bands case with respect to the 4-bands case.

– Considering the results on the Test Set, the addition of more bands seems to worsen the classification. While for the 6-bands case is hard to provide an explanation for these results, for the 7-bands the reason could be that in one of the tests performed (reported in Appendix A.6, case NN2c7bB1), the Typha blob used in the test set was not detected at all. This could be possible due to a combination of Typha blobs selected for training and the integration of the distance: since the other two blobs of Typha have very low distance from the river, the model learned this feature. The testing blob is farther from the river, and thus is not classified as Typha, as shown in Figure 6.3. This substantially contributes to reduce the overall metrics.

*Figure 6.3: Detail of image showing the problem with this particular case: the blob selected as test set is not recognized at all. The pixels classifed as Typha are in green, while the pixels of Typha missclassified are in purple.*

### Random Split

This set of tests was performed with the random split method, with either 4, 6 or 7 bands. The metrics are computed for each of these cases on both the Overall dataset and the Test Set only. The results are reported in table 6.3.

| | 4-bands | | 6-bands | | 7-bands | |
|---|---|---|---|---|---|---|
| | Overall | Test Set | Overall | Test Set | Overall | Test Set |
| Accuracy | 99.194 | 99.207 | 99.248 | 99.240 | 99.614 | 99.614 |
| Precision Typha | 75.952 | 76.230 | 78.673 | 78.177 | 91.421 | 91.350 |
| Recall Typha | 81.157 | 81.285 | 81.826 | 81.681 | 88.935 | 88.940 |
| F1-Score Typha | 78.466 | 78.676 | 80.198 | 79.873 | 90.161 | 90.129 |

*Table 6.3: Results with the random split methods.*

The analysis of this table reveals that:

– There is no difference between the Overall and Test Set results when examining each band case singularly. This is an expected result, since the random split

method uses as training set pixels from all the three different blobs, creating a more homogeneous and well distributed training set. The differences between the three blobs are not taken into consideration when a random split is used.

– Considering both the Overall set and the Test Set, the 6-bands case performs equally as the the 4-bands case. This is an unexpected result, since the integration of the NDVI and NDWI layers was supposed to increase the classification performance. Altough a specific reason at the basis of this behaviour can be hardly provided, some hypothesis can be made. Since one of the two classes is composed by all the groundtruths with the exception of Typha, the differences provided by the two additional layers between the different classes are included in the mixed class. Since Typha is very similar to the vegetation class, the differences between the Typha class and the Other class are actually diminished, thus resulting in a less precise classification.

– Considering both the Overall and Test Set, the 7-bands case performs significantly better than any other case, with an increment of the F1-score on the Test Set of ∼14%, with respect to the 4-bands case.

### Blob and Random Split Comparison

In this section a comparison between the Random and Blob split methods is presented.

| | Blob Split | | | | Random Split | | | |
|---|---|---|---|---|---|---|---|---|
| | 4 bands | | 7 bands | | 4 bands | | 7 bands | |
| | Overall | Test Set | Overall | Test Set | Overall | Test Set | Overall | Test Set |
| Accuracy | 99.202 | 99.167 | 99.199 | 98.577 | 99.194 | 99.207 | 99.614 | 99.614 |
| Precision Typha | 71.684 | 58.020 | 76.195 | 53.852 | 75.952 | 76.230 | 91.421 | 91.350 |
| Recall Typha | 84.924 | 80.895 | 84.127 | 39.778 | 81.157 | 81.285 | 88.935 | 88.940 |
| F1-Score Typha | 77.374 | 64.186 | 78.138 | 44.394 | 78.466 | 78.676 | 90.161 | 90.129 |

*Table 6.4: Comparison between the results obtained with a 2-class Blob split and a Random split.*

The analysis of this table reveals that:

– Accuracies obtained are similar for all cases.

– Comparing the metrics referred to Typha between the Blob and Random split, it can be noticed that:

1. The results on the Overall set are comparable between blob and random split in case of the 4-bands. Conversely in the 7-bands classification, the random split performs better than the blob split, with an increment of the F1-Score of ∼15%

2. The results on the Test set are significantly improved using a random split. Indeed, for the 4-bands case the increment of F1-Score is ∼22% with respect to the blob split; for the 7-bands the increment further raises to ∼103% with respect to the blob split.

With a random split method, the performances are unquestionably better, and the integration of the spatial constraints consistently improves the classification.

### 6.2.4 6-class Classification

This set of experiments was performed to gain more insights into how the model performs in a multi-case classification scenario. Also, the idea was to assess if the results obtained with the 2-class classification with the integration of the distance were the same. Furthermore, more information on how the pixels of Typha were classified were collected. A 6-class classification (Typha, river, soil, agriculture fields, city and vegetation was performed. The tests were perfomed with three different settings: only 4 bands, i.e. the original satellite data; the 4-bands with the integration of the NDVI and NDWI layers; the 6-bands, previously described, with the integration of the spatial constraints obtained from the river distance.

Results are presented in the section below. The first section shows the results when the blob split method was used. The second section shows the results when the random split method was used. The third section presents a comparison between the random and blob split methods. The details of each experiment, and the confusion matrices, are reported in Appendix A.9, A.10, A.11, A.12, A.13, A.14.

#### Blob Split

This set of tests was performed with the blob split method, with either 4, 6 or 7 bands. The metrics are computed for each of these cases on both the Overall dataset and the Test Set only. The results are reported in table 6.5.

| | 4-bands | | 6-bands | | 7-bands | |
|---|---|---|---|---|---|---|
| | Overall | Test Set | Overall | Test Set | Overall | Test Set |
| Accuracy | 86.217 | 86.001 | 85.769 | 85.707 | 81.091 | 76.909 |
| Precision Typha | 72.575 | 60.637 | 69.654 | 56.752 | 82.960 | 62.464 |
| Recall Typha | 83.130 | 78.226 | 82.438 | 76.678 | 96.586 | 58.159 |
| F1-Score Typha | 76.942 | 64.667 | 74.806 | 60.591 | 87.921 | 59.938 |

*Table 6.5: Results with the blob split method and 6 class.*

The analysis of this table reveals that:

- There is a significant difference between the Overall and Test Set results, when examining each band case singularly, especially for the Typha's metrics. This behaviour is very similar to the 2-class classification case.

- Considering only the Overall set, the 4-bands and 6-bands performs evenly. The 7-bands classification performs better than any other case, with an F1-score increase of ∼14% with respect to the 4-bands case.

- Considering the Test set only, the integration of more bands does not seem to majorly improve the classification. With respect to the 4-bands, the F1-Score of the 6-bands classification decreases of ∼6%, likewise the F1-Score of the 7-bands classification decreases of ∼-7%

  As already shown for the 2-class classification, even in this case, the Typha blob used as test set was not detected at all (reported in Appendix A.13, case NN6c7bB1). Possible reasons were already introduced in Section 6.2.3. The testing blob is farther from the river, and thus is classified as vegetation, as shown in Figure 6.4. This substantially contributes to reduce the overall metrics.

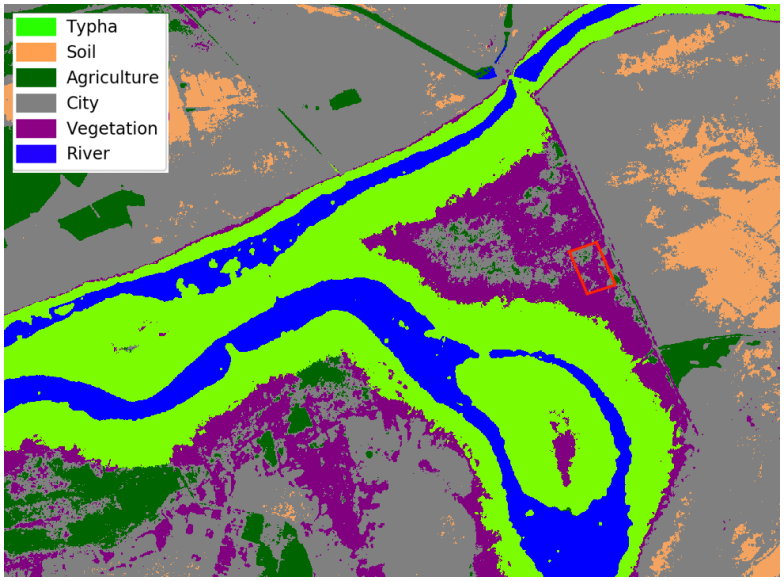



*Figure 6.4: Detail of the image, showing the problem with this particular case. The Typha blob selected as training set (inside the red square) is not recognized at all, as is classified as half city and half vegetation.*

### Random Split

This set of tests was performed with the random split method, with either 4, 6 or 7 bands. The metrics are computed for each of these cases on both the Overall dataset and the Test Set only. The results are reported in table 6.6.

| | 4-bands | | 6-bands | | 7-bands | |
|---|---|---|---|---|---|---|
| | Overall | Test Set | Overall | Test Set | Overall | Test Set |
| Accuracy | 89.874 | 89.845 | 86.534 | 86.488 | 95.559 | 95.589 |
| Precision Typha | 75.634 | 75.918 | 75.208 | 75.074 | 98.795 | 98.717 |
| Recall Typha | 82.188 | 81.978 | 80.809 | 80.663 | 96.149 | 96.129 |
| F1-Score Typha | 78.700 | 78.736 | 77.879 | 77.731 | 97.453 | 97.405 |

*Table 6.6: Results with the random split methods and 6 class.*

The analysis of this table reveals that:

- There is no difference between the Overall and Test Set results when examining each band case singularly. As for the 2-class case, this is an expected result.
- Considering both the Overall set and the Test Set, the 6-bands case performs evenly as the 4-bands case.
- Considering both the Overall and Test Set, the 7-bands case performs significantly better than any other case, with an increment of the F1-score on the Test Set of ∼23%, with respect to the 4-bands case, and an increment of ∼25% with respect to the 6-bands case.

### Blob and Random Split Comparison

In this section a comparison between the Random and Blob split method is presented.

| | Blob Split | | | | Random Split | | | |
|---|---|---|---|---|---|---|---|---|
| | 4 bands | | 7 bands | | 4 bands | | 7 bands | |
| | Overall | Test Set | Overall | Test Set | Overall | Test Set | Overall | Test Set |
| Accuracy | 86.217 | 86.001 | 81.091 | 76.909 | 89.874 | 89.845 | 95.559 | 95.589 |
| Precision Typha | 72.575 | 60.637 | 82.960 | 62.464 | 75.634 | 75.918 | 98.795 | 98.717 |
| Recall Typha | 83.130 | 78.226 | 96.586 | 58.159 | 82.188 | 81.978 | 96.149 | 96.129 |
| F1-Score Typha | 76.942 | 64.667 | 87.921 | 59.938 | 78.700 | 78.736 | 97.453 | 97.405 |

*Table 6.7: Comparison between the results obtained with a 6-class Blob and Random split methods.*

- The table shows that, regarding the Overall set the best accuracy is obtained by the 7-bands random split case. The same applies to the Test Set.
- Comparing the metrics referred to the Typha, the 7-bands random split performs better than the 7-bands blob split, for both for the Overall and the Test Set. The increment of F1-Score on the Overall set is ∼10%, and on the Test Set is ∼62%.

  In conclusion, the integration of the spatial contraints improves significantly the classification, especially with the random split method.

## 6.3 Analysis of the Results Obtained with Convolutional Neural Networks

As previously said in Chapter 4, the dataset is composed by a multispectral image, of dimensions $3137 \times 5238 \times n_c$ where $n_c$ is the number of channel selected. For the convolutional neural network architecture it was decided to focus on only one case: using the channels provided by the satellite joined with the NDVI and NDWI layers (6-bands). The CNN already integrates the spatial data, so there is no need to add the layer of the distance from the river. These tests were made to assess wether a more complex model that intrinsically uses spatial data is able to classify Typha's pixels, and how it compares to our model, that integrate spatial information "a-priori".

As said in Section 4.5, only the blob split method was used for the CNN. The single tests are presented in Appendix A.15. The table 6.8 shows the results.

| | 6-bands | |
|---|---|---|
| | Overall | Test Set |
| Accuracy | 94.667 | 80.886 |
| Precision | 93.757 | 70.613 |
| Recall | 98.866 | 93.949 |
| F1-Score | 96.173 | 78.601 |

*Table 6.8: Results with the CNN method using 6 classes and a blob split method.*

## 6.4 Comparison between NN 7-class and CNN

In this section the final comparison between the results of the NN 7-class classification and the CNN are presented. The 2-class is not considered, since the CNN model was run only with a 6 class setting.

| | NN 6-class 7-bands | | | | CNN | |
|---|---|---|---|---|---|---|
| | Blob Split | | Random split | | Blob Split | |
| | Overall | Test Set | Overall | Test Set | Overall | Test Set |
| Accuracy | 81.091 | 76.909 | 95.559 | 95.589 | 94.667 | 80.886 |
| Precision Typha | 82.960 | 98.795 | 62.464 | 98.717 | 93.757 | 70.613 |
| Recall Typha | 96.586 | 58.159 | 96.149 | 96.129 | 98.866 | 93.949 |
| F1-Score Typha | 87.921 | 59.938 | 97.453 | 97.405 | 96.173 | 78.601 |

*Table 6.9: Comparison between the results obtained with a NN 6-class 7-bands and the CNN.*

The analysis of this table reveals that:

– Comparing the Overall Accuracy, the CNN performs better than the NN blob split, with an increase of ∼16%. Meanwhile, the CNN performs comparably with respect to the NN Random split, with a neglectable decrease.

– Comparing the Test Set accuracies, the CNN performs better than the NN Blob split, with an increase of ∼5%. However, the NN Random Split performs better, with a significant increase of ∼18%, with respect to the CNN.

– Comparing the metrics referred to Typha, it can be noticed that:

1. The results of the Overall set show that the NN 7-bands random split performs better than any other case, with an F1-Score increase of ∼10% with respect to the NN Blob split, and an F1-Score increase of ∼1% with respect to the CNN.
2. The results on the Test set show that the NN Random split provides the best performances. Indeed, besides showing an F1-Score increase of ∼62% with respect to the NN Blob split, it displays an F1-Score increase of ∼23% even with respect to the CNN.

## 6.5 Final Considerations

In this section final considerations are presented. In particular, the differences between the base case i.e. 4-bands with no cleaning, and the case exploiting all the contributions made in this thesis i.e. 6-bands with the addition of spatial information and the groundtruth cleaning, are reported. All the cases below are presented when using a random split, because it achieved the best performances.

#### 2-class classification

**4-bands no cleaning** **7-bands with cleaning**

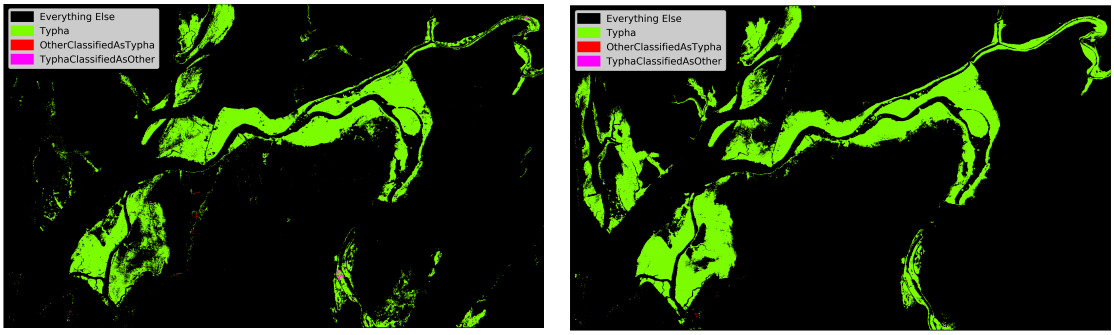



*Figure 6.5: Comparison of the classification on the whole image.*

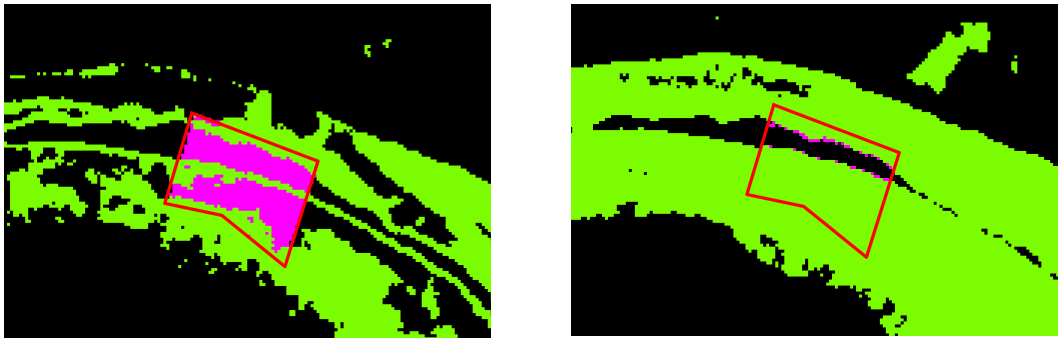


*Figure 6.6: Comparison between the first groundtruth of Typha. In the right image the black pixels inside the red area were removed with the cleaning.*

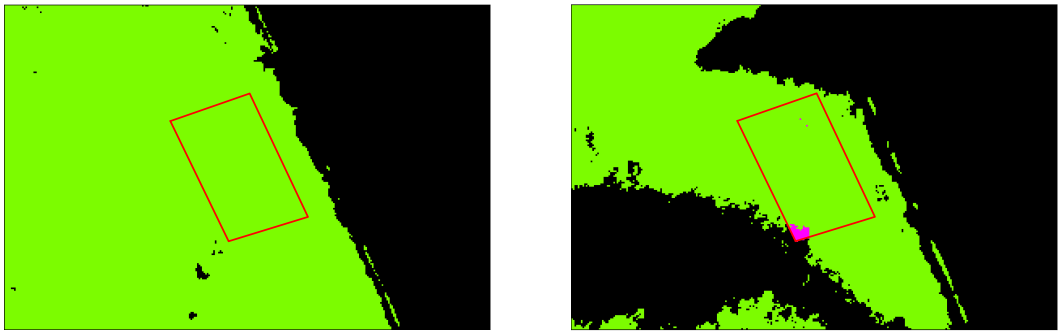


*Figure 6.7: Comparison between the second groundtruth of Typha.*

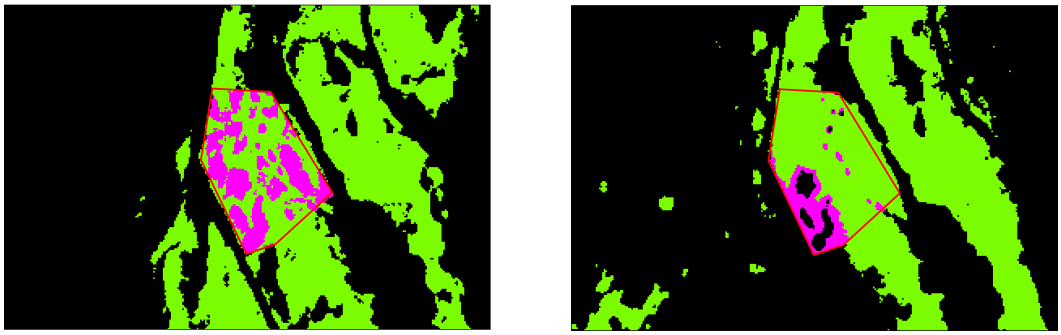


*Figure 6.8: Comparison between the third groundtruth of Typha.*

As Figure 6.5 shows, the full classified map looks similar. Going in details, Figure 6.6 shows much better results when 7-bands method is used: in fact almost every pixel inside the groundtruth is classified as Typha. The same results can be seen in Figure 6.8. In Figure 6.7 the two methods perform equally well.

### 6-class classification

**4-bands no cleaning** **7-bands with cleaning**

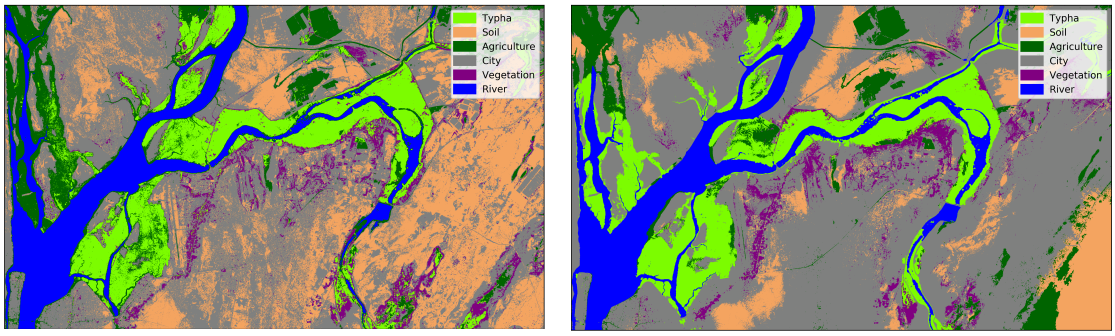



*Figure 6.9: Comparison of the classification on the whole image.*

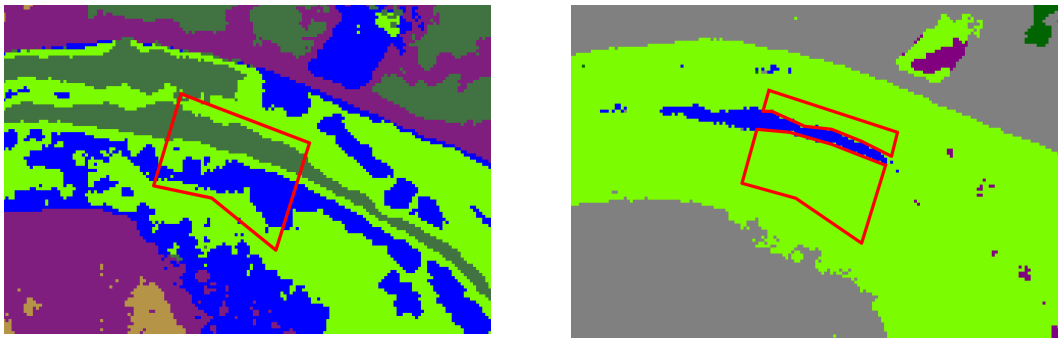


*Figure 6.10: Comparison between the first groundtruth of Typha. In the right image, the groundtruth is split because the center part was removed with the cleaning.*

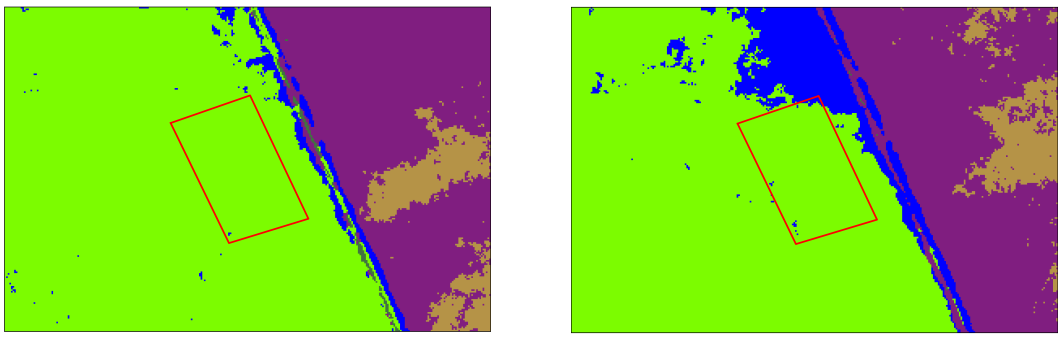


*Figure 6.11: Comparison between the second groundtruth of Typha.*

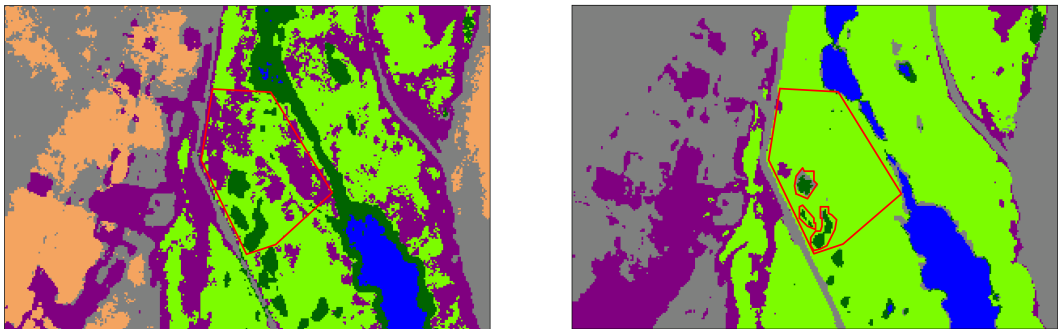


*Figure 6.12: Comparison between the third groundtruth of Typha. In the right image, the red circles inside the groundtruth are the pixels removed with the cleaning.*

As Figure 6.5 shows, the map looks similar. With the 7-bands method, most of the map was classified, not correctly, as city. This problem may arise due to the number of the city's groundtruths, that was much higher when compared to the other. Furthermore the fact that the city is spread homogeneously with respect to the distance of the river, may have worsened the classification.

Going in details in Figures 6.6, the 7-bands methods performs much better. In fact, almost every pixel inside the groundtruth is classified as Typha, and the pixels classified as river were removed from the groundtruth with the cleaning method. In Figure 6.8 the behaviour is very similar: the 7-bands classify almost perfectly every pixel inside the Typha's groundtruth, meanwhile the 4-bands contains a high quantity of wrongly classified pixels. In Figure 6.7 the difference is not noticeable, since both the methods performs well.

# Chapter 7

# Final Conclusions and Future Works

In this work we studied the problem of satellite image segmentation using neural networks, one of the most up-to-dated models. In particular, we developed a novel approach consisting in the integration of spatial information, that produced improvements with respect to spectral-based only classification.
The main contribution are:

- Integration of spatial constraints coming from GIS data in order to improve the classification.
- Cleaning of the Typha's groundtruth in order to improve the classification.
- Split between training and test data with a novel blob split method, and report of the effects on the classification.

We tested our work in different scenarios, and, using a random split, we concluded that:

1. Performing a 2-class classification (Typha vs Other), our method showed superior performances with respect to the basic method, achieving an F1-Score on Typha of 90%.
2. To further confirm the result, a 6-class classification (Typha, Soil, River, City, Agriculture fields, Vegetation) was performed. Even in this case, our method showed superior performances with respect to the basic method, achieving an F1-Score of 97%
3. To confirm even further, a CNN model, which intrinsically exploit spatial constraints contained in the image, was tested against our model. The results showed that the CNN performs better than our model, when using a blob split method, but performs worse when the random split method is employed.

Given these results, the integration of spatial constraints in the classification using GIS data, combined with the groundturth cleaning, can be a valid solution when performing a classification that needs high precision.

The results presented in Chapter 6 can be certainly improved, with more possible scenarios to be explored. Foremost is the fact that the groundtruth dataset is fairly small, does not contain enough pixels and is not precise. A better dataset would yield much better results. Furthermore, the possibility to exploit more groundtruths of Typha could enhance the blob split method. Indeed, given only three blobs of Typha, the cross-validation tests could not be robust.

Another improvement that could be made is the exploiting of different satellite images, derived for example from Sentinel-2, whose data have less spatial resolution but more bands. The increased number of bands could help the model to further discriminate the vegetation, especially considering that the spectral signature of Typha and other plants might be very different. Using only 4 bands, this difference was not detectable, and it was difficult to distinguish these two classes.

Another approach that could be taken is by integrating multi-temporal images, in such a way that the seasonality of the Typha could be taken into account.

The neural networks models could also be improved. The hyperparameter were fine-tuned manually, but an algorithmic approach could be used, in order to selected the best set of hyperparameters.

# Appendix A

# Details of Experiments

In these appendices the details of each experiment are reported. Each of the experiments was performed in both the cases of a random split and the blob split method. Each different instance was defined with a different name, in order to differentiate it from the others (Figure A.1). For example, the code NN6c7bB2 has the meaning: Neural Network model, 6-class classification, 7 bands used, Blob split, case number 2. The results of these analyses are presented in the following sections.

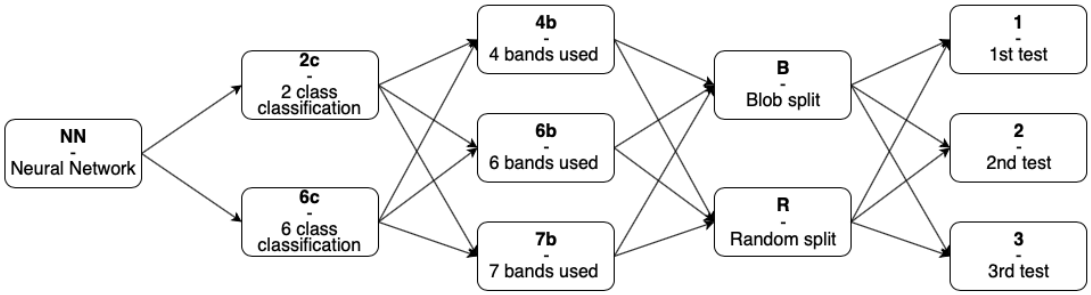



*Figure A.1: Taxonomy of the cases. The codes used for the different cases are explained in this image.*

## A.1 NN 2-class 4-bands Blob Split No cleaning

In this section the results of the Neural Network model, with a 2-class 4-bands blob split classification and no cleaning is presented. The three different experiments represent the three different combination of Typha blobs selected for training and test set.

### NN2c4bB0 - no cleaning

| | | Predicted label | |
|---|---|---|---|
| | | Typha | Other |
| True | Typha | 575 | 866 |
| | Other | 329 | 156375 |

| | Overall | Test Set |
|---|---|---|
| Accuracy | 99.128 | 99.244 |
| Precision Typha | 73.250 | 39.902 |
| Recall Typha | 82.282 | 63.606 |
| F1-Score Typha | 77.504 | 49.040 |

*Table A.1: Results of the test with the neural network 2-class 4-bands blob split and without cleaning, first case. On the left is presented the confusion matrix, on the right the metrics computed.*

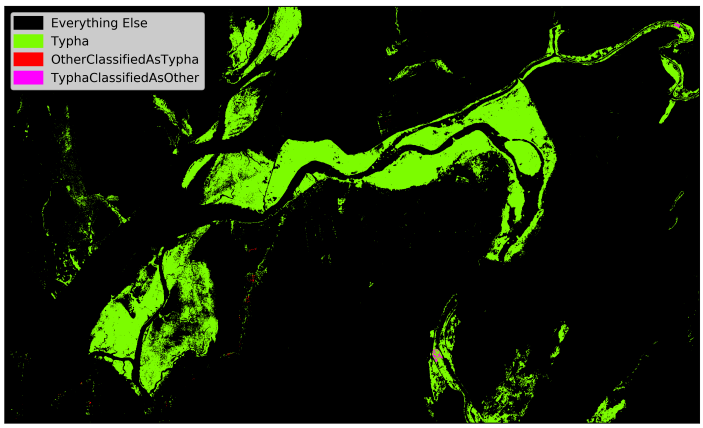



*Figure A.2: Result of the classification.*

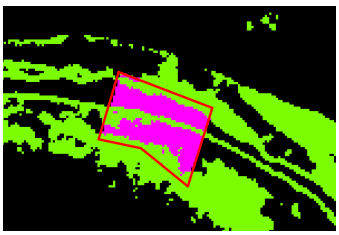


*Figure A.3: Detail of the Typha blob that belongs to the test set.*

### NN2c4bB1 - no cleaning

| | | Predicted label | |
|---|---|---|---|
| | | Typha | Other |
| True | Typha | 4390 | 1 |
| | Other | 526 | 201658 |

| | Overall | Test Set |
|---|---|---|
| Accuracy | 99.146 | 99.744 |
| Precision Typha | 75.933 | 99.977 |
| Recall Typha | 81.203 | 89.303 |
| F1-Score Typha | 78.480 | 94.337 |

*Table A.2: Results of the test with the neural network 2-class 4-bands blob split and without cleaning, second case. On the left is presented the confusion matrix, on the right the metrics computed.*

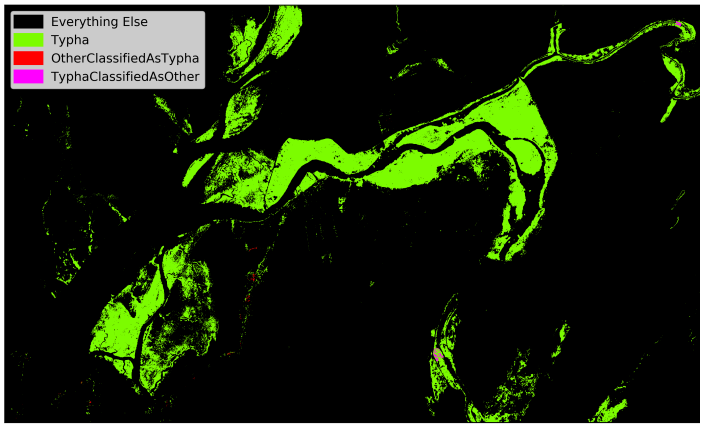



*Figure A.4: Result of the classification*

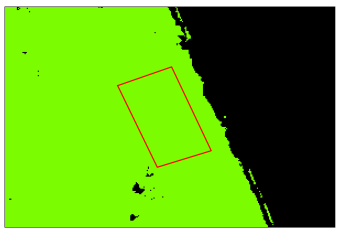


*Figure A.5: Detail of the Typha blob that belongs to the test set.*

**NN2c4bB2 - no cleaning**

| | | Predicted label | |
|---|---|---|---|
| | | Typha | Other |
| True | Typha | 1217 | 3983 |
| | Other | 218 | 211071 |

| | Overall | Test Set |
|---|---|---|
| Accuracy | 98.908 | 98.059 |
| Precision Typha | 54.151 | 23.403 |
| Recall Typha | 88.021 | 84.808 |
| F1-Score Typha | 67.052 | 36.684 |

*Table A.3: Results of the test with the neural network 2-class 4-bands blob split and without cleaning, third case. On the left is presented the confusion matrix, on the right the metrics computed.*

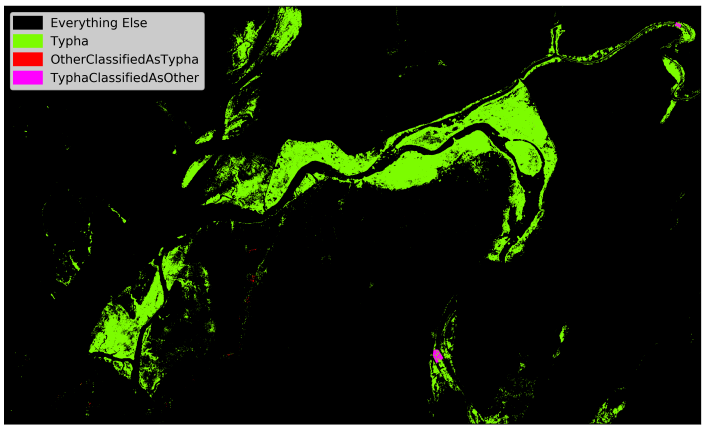



*Figure A.6: Result of the classification*

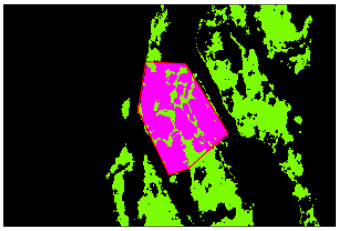


*Figure A.7: Detail of the Typha blob that belongs to the test set.*

## A.2 NN 2-class 4-bands Blob Split

In this section the results of the Neural Network model, with a 2-class 4-bands blob split and cleaning is presented. The three different experiments represent the three different combination of Typha blobs selected for training and test set.

**NN2c4bB0**

| | | Predicted label | |
|---|---|---|---|
| | | Typha | Other |
| True | Typha | 542 | 664 |
| | Other | 252 | 156452 |

| | Overall | Test Set |
|---|---|---|
| Accuracy | 99.254 | 99.420 |
| Precision Typha | 76.234 | 44.942 |
| Recall Typha | 83.740 | 68.262 |
| F1-Score Typha | 79.811 | 54.200 |

*Table A.4: Results of the test with the neural network 2-class 4-bands blob split with cleaning, first case. On the left is presented the confusion matrix, on the right the metrics computed.*

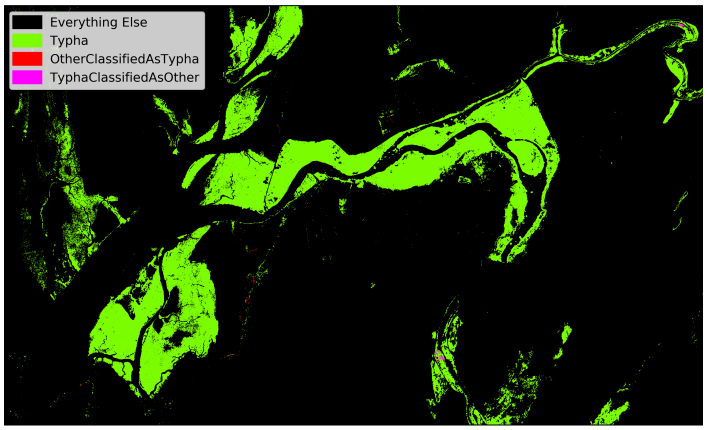



Figure A.8: Result of the classification.

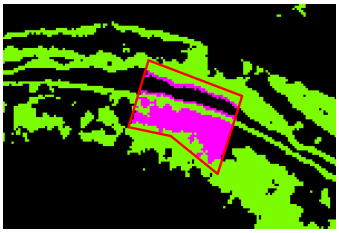


Figure A.9: Detail of the Typha blob that belongs to the test set.

### NN2c4bB1

| | | Predicted label | |
|---|---|---|---|
| | | Typha | Other |
| True | Typha | 4387 | 4 |
| | Other | 454 | 201730 |

| | Overall | Test Set |
|---|---|---|
| Accuracy | 99.310 | 99.778 |
| Precision Typha | 79.244 | 99.909 |
| Recall Typha | 84.183 | 90.622 |
| F1-Score Typha | 81.639 | 95.039 |

Table A.5: Results of the test with the neural network 2-class 4-bands blob split with cleaning, second case. On the left is presented the confusion matrix, on the right the metrics computed.

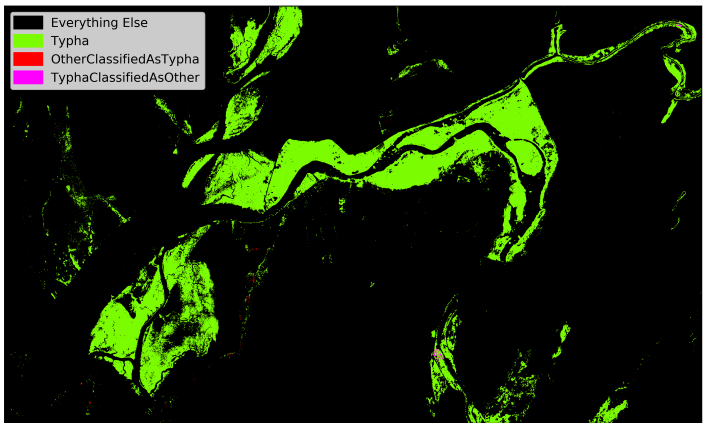



Figure A.10: Result of the classification

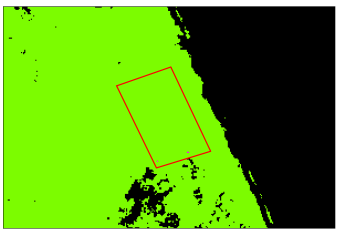


Figure A.11: Detail of the Typha blob that belongs to the test set.

**NN2c4bB2**

| | | Predicted label | |
|---|---|---|---|
| | | Typha | Other |
| True | Typha | 1402 | 3398 |
| | Other | 271 | 211018 |

| | Overall | Test Set |
|---|---|---|
| Accuracy | 99.043 | 98.302 |
| Precision Typha | 59.575 | 29.208 |
| Recall Typha | 86.848 | 83.802 |
| F1-Score Typha | 70.671 | 43.318 |

*Table A.6: Results of the test with the neural network 2-class 4-bands blob split with cleaning, third case. On the left is presented the confusion matrix, on the right the metrics computed.*

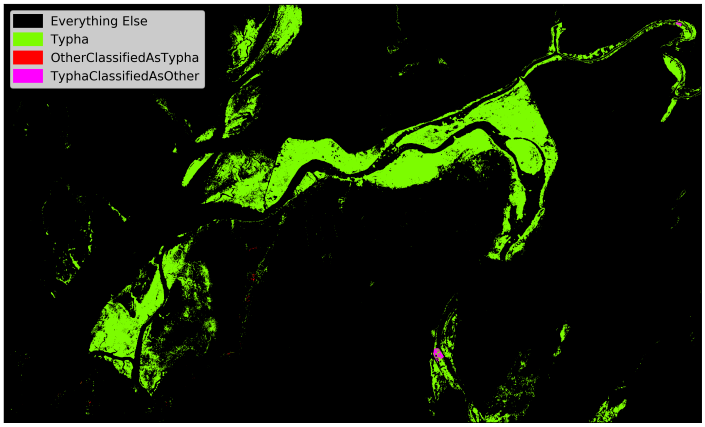



*Figure A.12: Result of the classification*

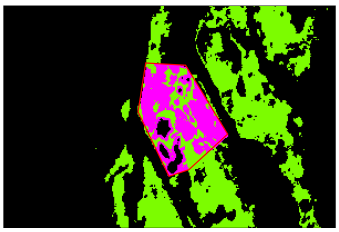


*Figure A.13: Detail of the Typha blob that belongs to the test set.*

## A.3 NN 2-class 4-bands Random Split

In this section the results of the Neural Network model, with a 2-class 4-bands random split classification with cleaning is presented. The three different experiments represent the three different combination of Typha blobs selected for training and test set.

**NN2c4bR0**

| | | Predicted label | |
|---|---|---|---|
| | | Typha | Other |
| True | Typha | 2566 | 846 |
| | Test set | 611 | 173328 |

| | Overall | Test Set |
|---|---|---|
| Accuracy | 99.153 | 99.178 |
| Precision Typha | 74.647 | 75.205 |
| Recall Typha | 80.200 | 80.768 |
| F1-Score Typha | 77.324 | 77.887 |

*Table A.7: Results of the test with the neural network 2-class 4-bands random split with cleaning, first case. On the left is presented the confusion matrix, on the right the metrics computed.*

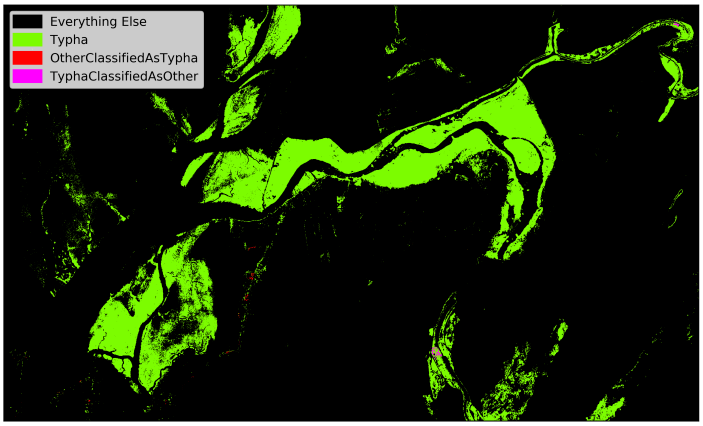



Figure A.14: Result of the classification.

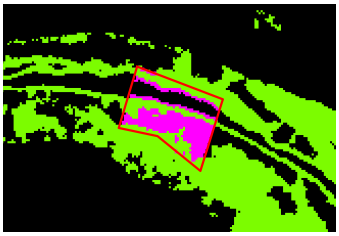


Figure A.15: Detail one of the Typha blobs.

### NN2c4bR1

| | | Predicted label | |
|---|---|---|---|
| | | Typha | Other |
| True | Typha | 2655 | 785 |
| | Other | 589 | 173322 |

| | Overall | Test Set |
|---|---|---|
| Accuracy | 99.219 | 99.225 |
| Precision Typha | 77.359 | 77.180 |
| Recall Typha | 81.357 | 81.843 |
| F1-Score Typha | 79.308 | 79.443 |

Table A.8: Results of the test with the neural network 2-class 4-bands random split with cleaning, second case. On the left is presented the confusion matrix, on the right the metrics computed.

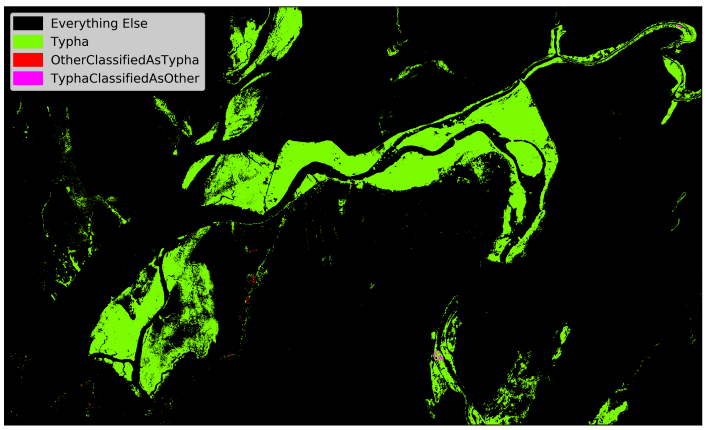



Figure A.16: Result of the classification

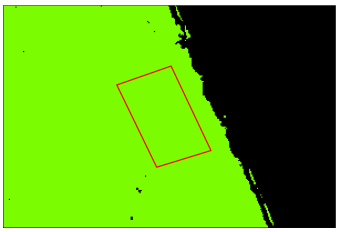


Figure A.17: Detail one of the Typha blobs.

**NN2c4bR2**

| | | Predicted label | |
|---|---|---|---|
| | | Typha | Other |
| True | Typha | 2560 | 795 |
| | Other | 591 | 173405 |

| | Overall | Test Set |
|---|---|---|
| Accuracy | 99.209 | 99.218 |
| Precision Typha | 75.849 | 76.304 |
| Recall Typha | 81.915 | 81.244 |
| F1-Score Typha | 78.765 | 78.697 |

*Table A.9: Results of the test with the neural network 2-class 4-bands random split with cleaning, third case. On the left is presented the confusion matrix, on the right the metrics computed.*

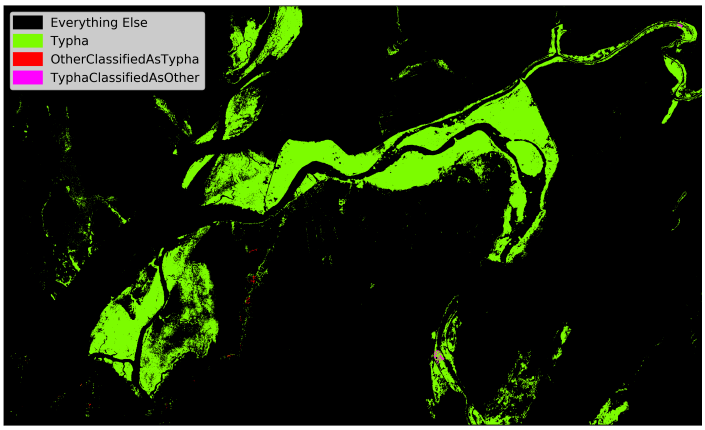



*Figure A.18: Result of the classification*

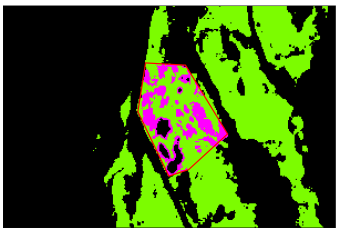


*Figure A.19: Detail one of the Typha blobs.*

## A.4 NN 2-class 6-bands Blob Split

In this section the results of the Neural Network model, with a 2-class 6-bands blob split classification with cleaning is presented. The three different experiments represent the three different combination of Typha blobs selected for training and test set.

**NN2c6bB0**

| | | Predicted label | |
|---|---|---|---|
| | | Typha | Other |
| True | Typha | 519 | 687 |
| | Other | 319 | 156385 |

| | Overall | Test Set |
|---|---|---|
| Accuracy | 99.188 | 99.363 |
| Precision Typha | 74.723 | 43.035 |
| Recall Typha | 81.736 | 61.933 |
| F1-Score Typha | 78.073 | 50.783 |

*Table A.10: Results of the test with the neural network 2-class 6-bands blob split with cleaning, first case. On the left is presented the confusion matrix, on the right the metrics computed.*

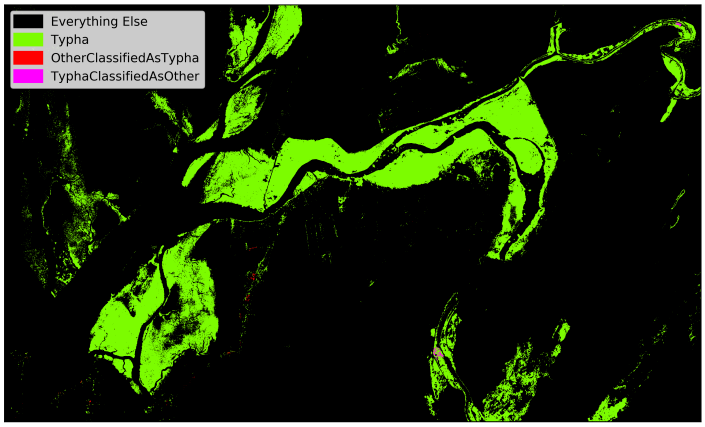



Figure A.20: Result of the classification.

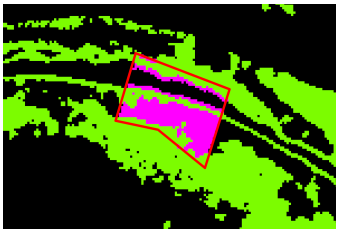


Figure A.21: Detail of the Typha blob that belongs to the test set.

### NN2c6bB1

| | | Predicted label | |
|---|---|---|---|
| | | Typha | Other |
| True | Typha | 4391 | 0 |
| | Other | 562 | 201622 |

| | Overall | Test Set |
|---|---|---|
| Accuracy | 99.261 | 99.728 |
| Precision Typha | 79.610 | 100.000 |
| Recall Typha | 81.724 | 88.653 |
| F1-Score Typha | 80.653 | 93.985 |

Table A.11: Results of the test with the neural network 2-class 6-bands blob split with cleaning, second case. On the left is presented the confusion matrix, on the right the metrics computed.

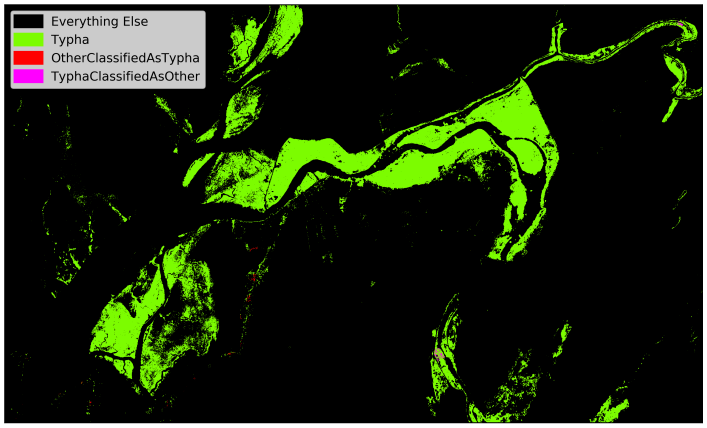



Figure A.22: Result of the classification

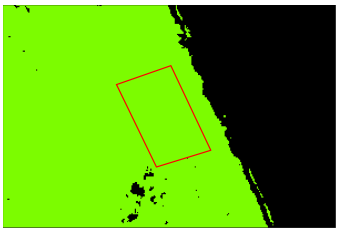


Figure A.23: Detail of the Typha blob that belongs to the test set.

**NN2c6bB2**

| | | Predicted label | |
|---|---|---|---|
| | | Typha | Other |
| True | Typha | 1481 | 3319 |
| | Other | 284 | 211005 |

| | Overall | Test Set |
|---|---|---|
| Accuracy | 99.046 | 98.333 |
| Precision Typha | 59.921 | 30.854 |
| Recall Typha | 86.624 | 83.909 |
| F1-Score Typha | 70.840 | 45.118 |

*Table A.12: Results of the test with the neural network 2-class 6-bands blob split with cleaning, third case. On the left is presented the confusion matrix, on the right the metrics computed.*

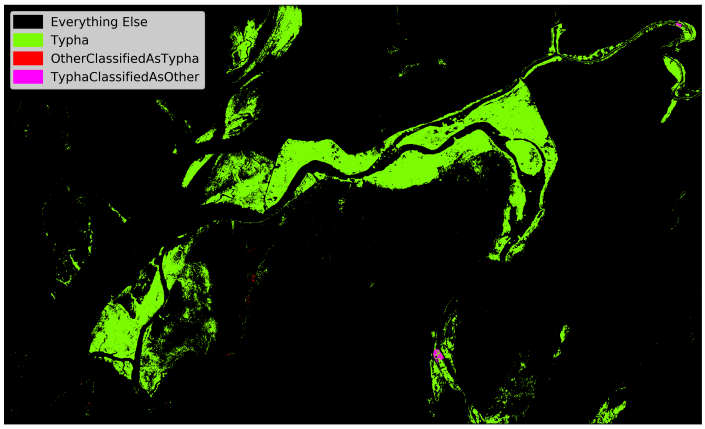



*Figure A.24: Result of the classification*

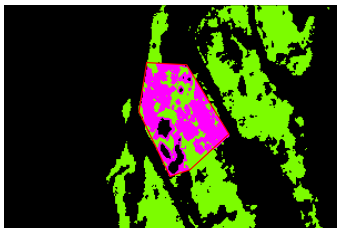


*Figure A.25: Detail of the Typha blob that belongs to the test set.*

## A.5 NN 2-class 6-bands Random Split

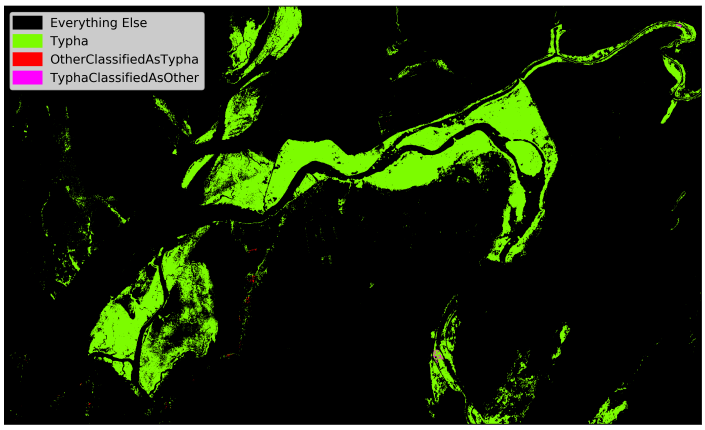
In this section the results of the Neural Network model, with a 2-class 6-bands random split classification with cleaning is presented. The three different experiments represent the three different combination of Typha blobs selected for training and test set.

**NN2c6bR0**

| | | Predicted label | |
|---|---|---|---|
| | | Typha | Other |
| True | Typha | 2699 | 696 |
| | Other | 547 | 173409 |

| | Overall | Test Set |
|---|---|---|
| Accuracy | 99.287 | 99.299 |
| Precision Typha | 79.600 | 79.499 |
| Recall Typha | 82.859 | 83.148 |
| F1-Score Typha | 81.197 | 81.283 |

*Table A.13: Results of the test with the neural network 2-class 6-bands random split with cleaning, first case. On the left is presented the confusion matrix, on the right the metrics computed.*



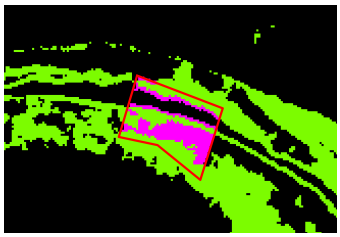



*Figure A.27: Detail one of the Typha blobs*

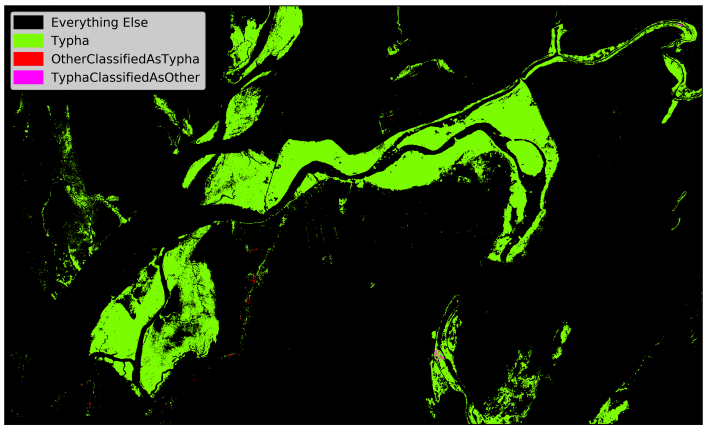
*Figure A.26: Result of the classification.*

### NN2c6bR1

| | | Predicted label | |
|---|---|---|---|
| | | Typha | Other |
| True | Typha | 2673 | 723 |
| | Other | 701 | 173254 |

| | Overall | Test Set |
|---|---|---|
| Accuracy | 99.215 | 99.197 |
| Precision Typha | 79.744 | 78.710 |
| Recall Typha | 79.691 | 79.223 |
| F1-Score Typha | 79.717 | 78.966 |

*Table A.14: Results of the test with the neural network 2-class 6-bands random split with cleaning, second case. On the left is presented the confusion matrix, on the right the metrics computed.*



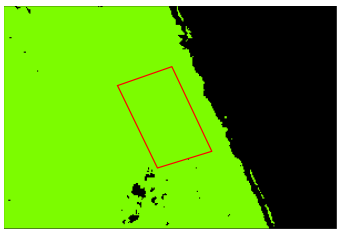



*Figure A.29: Detail one of the Typha blobs.*

*Figure A.28: Result of the classification*

**NN2c6bR2**

| | | Predicted label | |
|---|---|---|---|
| | | Typha | Other |
| True | Typha | 2643 | 820 |
| | Other | 554 | 173334 |

| | Overall | Test Set |
|---|---|---|
| Accuracy | 99.243 | 99.225 |
| Precision Typha | 76.676 | 76.321 |
| Recall Typha | 82.929 | 82.671 |
| F1-Score Typha | 79.680 | 79.369 |

*Table A.15: Results of the test with the neural network 2-class 6-bands random split with cleaning, third case. On the left is presented the confusion matrix, on the right the metrics computed.*

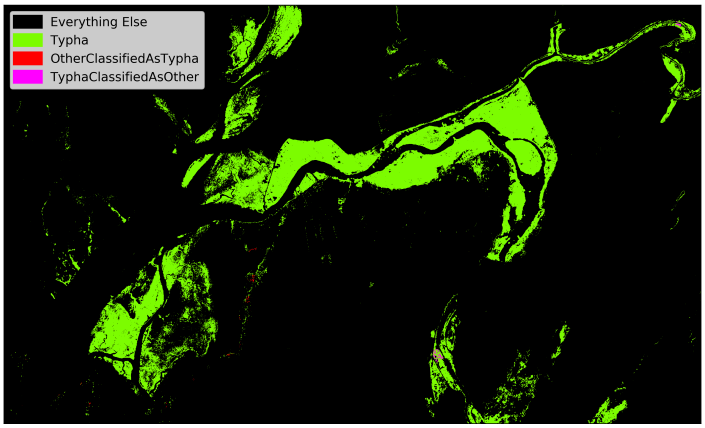



*Figure A.30: Result of the classification*

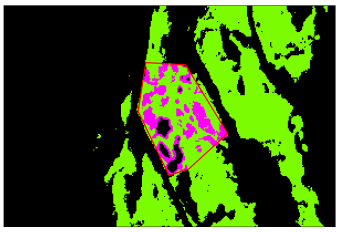


*Figure A.31: Detail one of the Typha blobs.*

## A.6 NN 2-class 7-bands Blob Split

In this section the results of the Neural Network model, with a 2-class 7-bands blob split classification with cleaning is presented. The three different experiments represent the three different combination of Typha blobs selected for training and test set.

**NN2c7bB0**

| | | Predicted label | |
|---|---|---|---|
| | | Typha | Other |
| True | Typha | 1121 | 85 |
| | Other | 1223 | 155481 |

| | Overall | Test Set |
|---|---|---|
| Accuracy | 99.462 | 99.172 |
| Precision Typha | 90.055 | 92.952 |
| Recall Typha | 83.434 | 47.824 |
| F1-Score Typha | 86.618 | 63.155 |

*Table A.16: Results of the test with the neural network 2-class 7-bands blob split with cleaning, first case. On the left is presented the confusion matrix, on the right the metrics computed.*

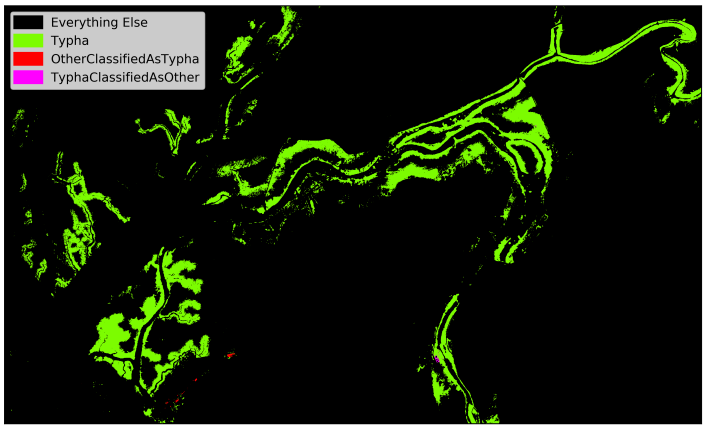



Figure A.32: Result of the classification.

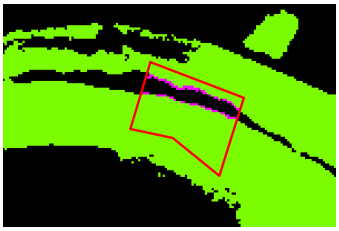


Figure A.33: Detail of the Typha blob that belongs to the test set.

### NN2c7bB1

| | | Predicted label | |
|---|---|---|---|
| | | Typha | Other |
| True | Typha | 0 | 4391 |
| | Other | 21 | 202163 |

| | Overall | Test Set |
|---|---|---|
| Accuracy | 99.145 | 97.864 |
| Precision Typha | 56.795 | 0.000 |
| Recall Typha | 98.253 | 0.000 |
| F1-Score Typha | 71.981 | nan |

Table A.17: Results of the test with the neural network 2-class 7-bands blob split with cleaning, second case. On the left is presented the confusion matrix, on the right the metrics computed.

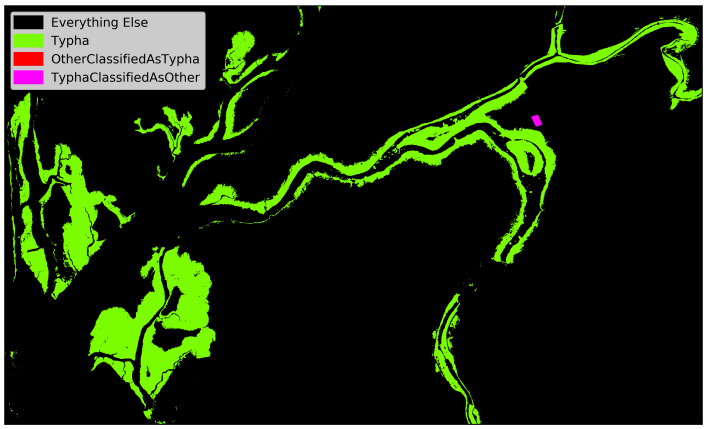



Figure A.34: Result of the classification

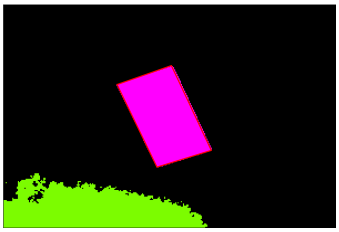


Figure A.35: Detail of the Typha blob that belongs to the test set.

**NN2c7bB2**

| | | Predicted label | |
|---|---|---|---|
| | | Typha | Other |
| True | Typha | 3293 | 1507 |
| | Other | 1312 | 209977 |

| | Overall | Test Set |
|---|---|---|
| Accuracy | 98.991 | 98.695 |
| Precision Typha | 81.735 | 68.604 |
| Recall Typha | 70.693 | 71.509 |
| F1-Score Typha | 75.814 | 70.027 |

*Table A.18: Results of the test with the neural network 2-class 7-bands blob split with cleaning, third case. On the left is presented the confusion matrix, on the right the metrics computed.*

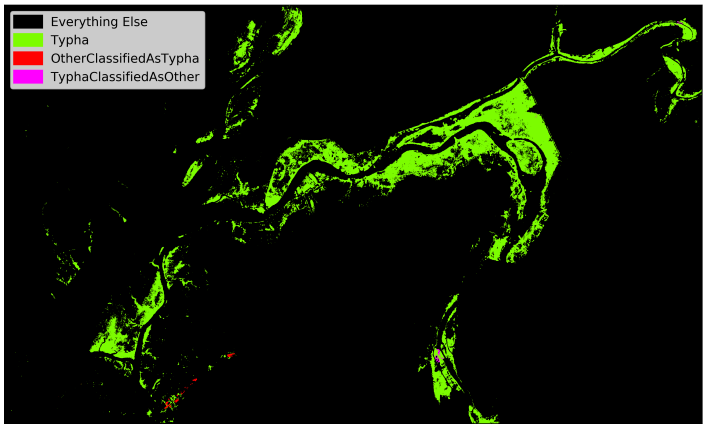



*Figure A.36: Result of the classification*

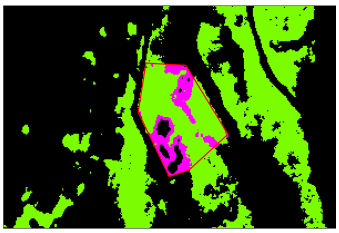


*Figure A.37: Detail of the Typha blob that belongs to the test set.*

## A.7 NN 2-class 7-bands Random Split

In this section the results of the Neural Network model, with a 2-class 7-bands blob split classification with cleaning is presented. The three different experiments represent the three different combination of Typha blobs selected for training and test set.

**NN2c7bR0**

| | | Predicted label | |
|---|---|---|---|
| | | Typha | Other |
| True | Typha | 3316 | 84 |
| | Other | 175 | 173776 |

| | Overall | Test Set |
|---|---|---|
| Accuracy | 99.851 | 99.854 |
| Precision Typha | 97.422 | 97.529 |
| Recall Typha | 94.983 | 94.987 |
| F1-Score Typha | 96.187 | 96.241 |

*Table A.19: Results of the test with the neural network 2-class 7-bands random split with cleaning, first case. On the left is presented the confusion matrix, on the right the metrics computed.*

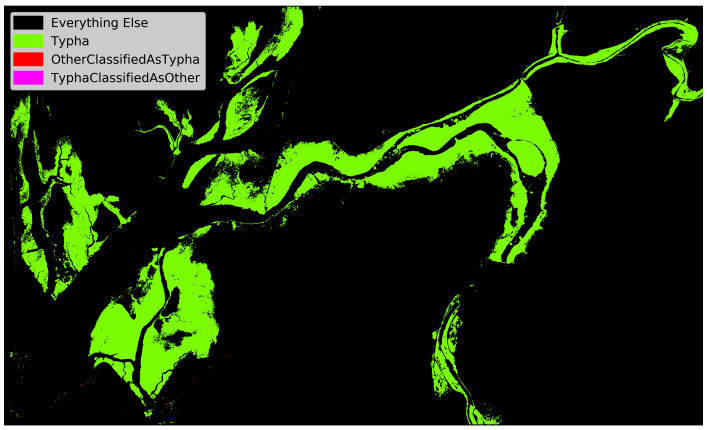



Figure A.38: Result of the classification.

Figure A.39: Detail one of the Typha blobs.

### NN2c7bR1

| | | Predicted label | |
|---|---|---|---|
| | | Typha | Other |
| True | Typha | 3081 | 369 |
| | Other | 464 | 173437 |

| | Overall | Test Set |
|---|---|---|
| Accuracy | 99.523 | 99.530 |
| Precision Typha | 89.180 | 89.304 |
| Recall Typha | 86.589 | 86.911 |
| F1-Score Typha | 87.865 | 88.091 |

Table A.20: Results of the test with the neural network 2-class 7-bands random split with cleaning, second case. On the left is presented the confusion matrix, on the right the metrics computed.

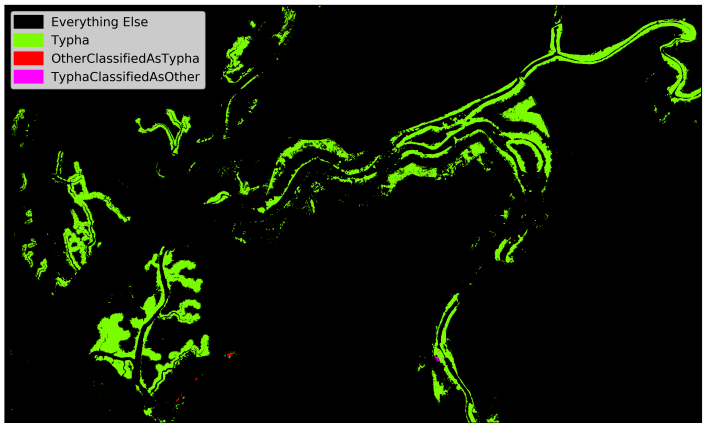



Figure A.40: Result of the classification

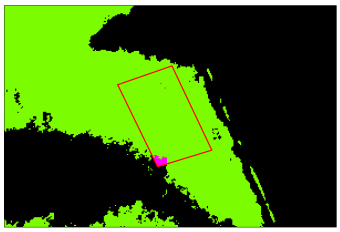


Figure A.41: Detail one of the Typha blobs.

### NN2c7bR2

| | | Predicted label | |
|---|---|---|---|
| | | Typha | Other |
| True | Typha | 2968 | 435 |
| | Other | 527 | 173421 |

| | Overall | Test Set |
|---|---|---|
| Accuracy | 99.467 | 99.458 |
| Precision Typha | 87.660 | 87.217 |
| Recall Typha | 85.233 | 84.921 |
| F1-Score Typha | 86.430 | 86.054 |

*Table A.21: Results of the test with the neural network 2-class 7-bands random split with cleaning, third case. On the left is presented the confusion matrix, on the right the metrics computed.*

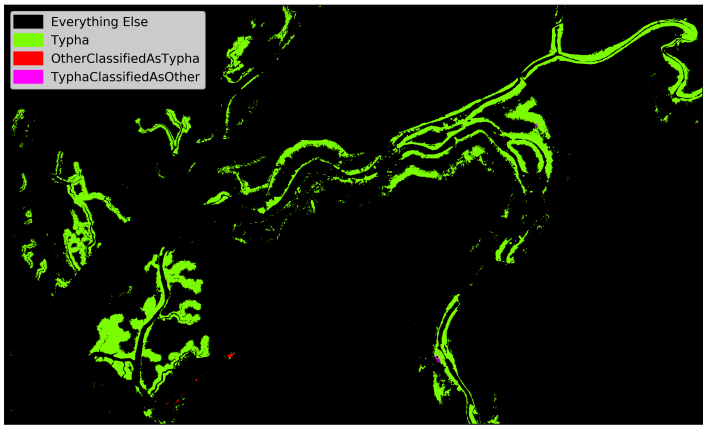



*Figure A.42: Result of the classification*

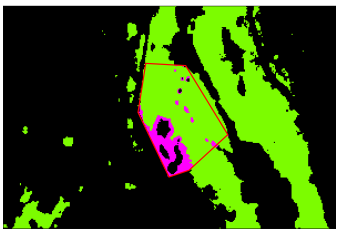


*Figure A.43: Detail one of the Typha blobs.*

## A.8 NN 6-class 4-bands Blob Split No cleaning

In this section the results of the Neural Network model, with a 6-class 4-bands blob split classification without cleaning is presented. The three different experiments represent the three different combination of Typha blobs selected for training and test set.

### NN6c4bB0 - no cleaning

| | | Predicted label | | | | | |
|---|---|---|---|---|---|---|---|
| | | Typha | Soil | Agriculture | City | Vegetation | River |
| True Label | Typha | 2585 | 0 | 356 | 0 | 593 | 0 |
| | Soil | 0 | 22556 | 0 | 7299 | 5 | 0 |
| | Agriculture | 78 | 0 | 26222 | 743 | 0 | 4 |
| | City | 130 | 5069 | 613 | 53157 | 529 | 39 |
| | Vegetation | 464 | 11 | 1 | 1180 | 4584 | 0 |
| | River | 0 | 0 | 11 | 0 | 0 | 51332 |

| | Overall | Test Set |
|---|---|---|
| Accuracy | 90.399 | 90.355 |
| Precision Typha | 72.362 | 73.147 |
| Recall Typha | 79.496 | 79.368 |
| F1-Score Typha | 75.762 | 76.130 |

*Table A.22: Results of the test with the neural network 6-class 4-bands blob split and without cleaning, first case. On the left is presented the confusion matrix, on the right the metrics computed.*

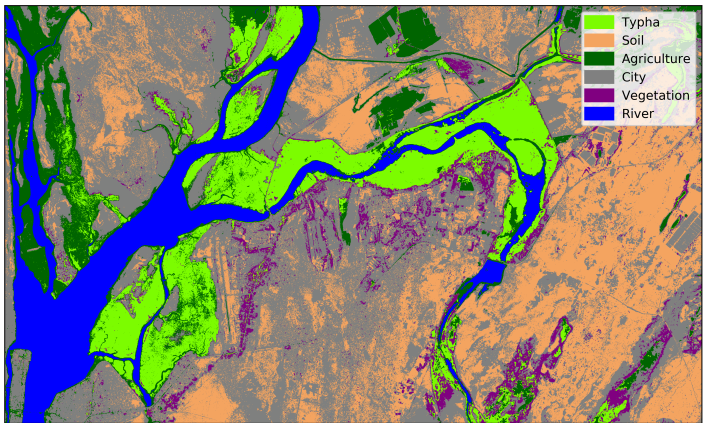



Figure A.44: Result of the classification.

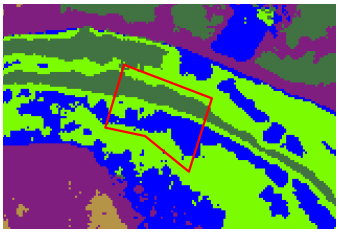


Figure A.45: Detail of the Typha blob that belongs to the test set.

### NN6c4bB1 - no cleaning

| | | Predicted label | | | | | |
|---|---|---|---|---|---|---|---|
| | | Typha | Soil | Agriculture | City | Vegetation | River |
| True Label | Typha | 2405 | 0 | 408 | 0 | 834 | 0 |
| | Soil | 0 | 17838 | 0 | 12131 | 7 | 0 |
| | Agriculture | 4 | 0 | 26290 | 788 | 0 | 31 |
| | City | 136 | 5829 | 576 | 52544 | 614 | 46 |
| | Vegetation | 412 | 9 | 0 | 973 | 4718 | 0 |
| | River | 0 | 0 | 46 | 0 | 0 | 50922 |

| | Overall | Test Set |
|---|---|---|
| Accuracy | 87.200 | 87.135 |
| Precision Typha | 65.836 | 65.945 |
| Recall Typha | 81.387 | 81.332 |
| F1-Score Typha | 72.790 | 72.835 |

Table A.23: Results of the test with the neural network 6-class 4-bands blob split without cleaning, second case. On the left is presented the confusion matrix, on the right the metrics computed.

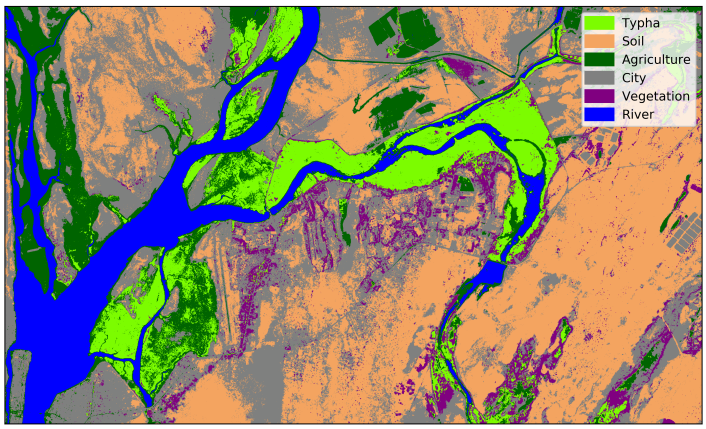



Figure A.46: Result of the classification.

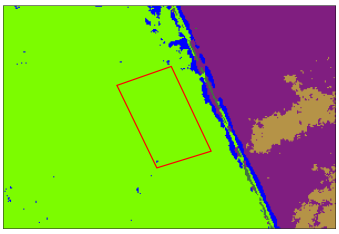


Figure A.47: Detail of the Typha blob that belongs to the test set.

### NN6c4bB2 - no cleaning

| | | Predicted label | | | | | |
|---|---|---|---|---|---|---|---|
| | | Typha | Soil | Agriculture | City | Vegetation | River |
| True Label | Typha | 2482 | 0 | 295 | 1 | 813 | 0 |
| | Soil | 0 | 24938 | 0 | 4982 | 0 | 0 |
| | Agriculture | 117 | 0 | 26278 | 853 | 15 | 30 |
| | City | 169 | 5893 | 546 | 52349 | 563 | 39 |
| | Vegetation | 408 | 46 | 0 | 1093 | 4520 | 0 |
| | River | 0 | 0 | 29 | 0 | 0 | 51102 |

| | Overall | Test Set |
|---|---|---|
| Accuracy | 91.091 | 91.050 |
| Precision Typha | 69.462 | 69.117 |
| Recall Typha | 77.373 | 78.149 |
| F1-Score Typha | 73.204 | 73.356 |

*Table A.24: Results of the test with the neural network 6-class 4-bands blob split without cleaning, third case. On the left is presented the confusion matrix, on the right the metrics computed.*

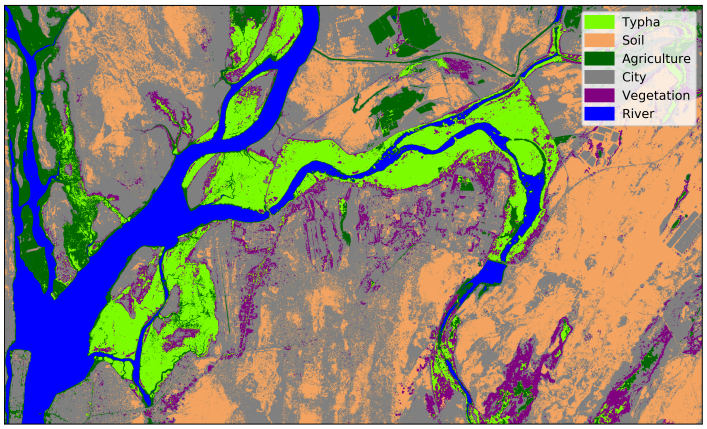



*Figure A.48: Result of the classification.*

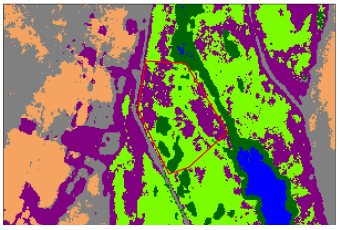


*Figure A.49: Detail one of the Typha blobs.*

## A.9 NN 6-class 4-bands Blob Split

In this section the results of the Neural Network model, with a 6-class 4-bands blob split classification with cleaning is presented. The three different experiments represent the three different combination of Typha blobs selected for training and test set.

### NN6c4bB0

| | | Predicted label | | | | | |
|---|---|---|---|---|---|---|---|
| | | Typha | Soil | Agriculture | City | Vegetation | River |
| True Label | Typha | 666 | 0 | 136 | 0 | 404 | 0 |
| | Soil | 0 | 13622 | 0 | 4663 | 0 | 0 |
| | Agriculture | 5 | 0 | 43791 | 1640 | 0 | 1471 |
| | City | 190 | 9338 | 378 | 51748 | 927 | 0 |
| | Vegetation | 179 | 0 | 0 | 197 | 3574 | 0 |
| | River | 0 | 0 | 0 | 0 | 0 | 24981 |

| | Overall | Test Set |
|---|---|---|
| Accuracy | 86.513 | 87.633 |
| Precision Typha | 81.196 | 55.224 |
| Recall Typha | 82.169 | 64.038 |
| F1-Score Typha | 81.680 | 59.305 |

*Table A.25: Results of the test with the neural network 6-class 4-bands blob split, with cleaning, first case. On the left is presented the confusion matrix, on the right the metrics computed.*

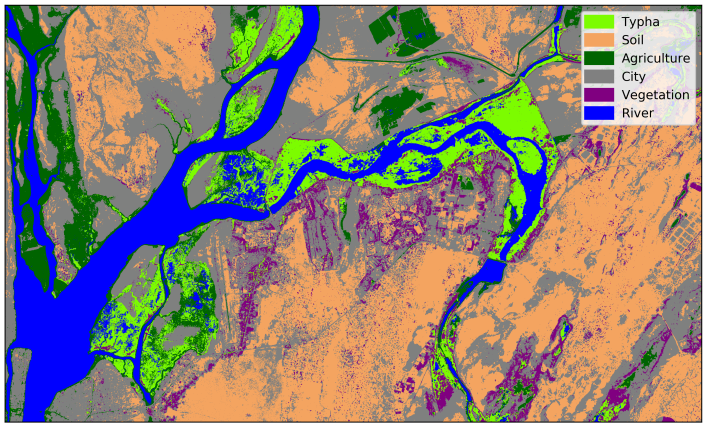



Figure A.50: Result of the classification.

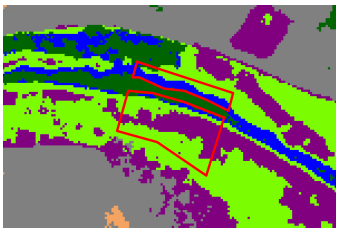


Figure A.51: Detail of the Typha blob that belongs to the test set.

### NN6c4bB1

| | | Predicted label | | | | | |
|---|---|---|---|---|---|---|---|
| | | Typha | Soil | Agriculture | City | Vegetation | River |
| True Label | Typha | 4391 | 0 | 0 | 0 | 0 | 0 |
| | Soil | 0 | 100 | 0 | 14155 | 0 | 0 |
| | Agriculture | 0 | 0 | 3291 | 0 | 0 | 0 |
| | City | 71 | 10340 | 528 | 48799 | 860 | 88 |
| | Vegetation | 435 | 1 | 0 | 51 | 2980 | 0 |
| | River | 0 | 0 | 0 | 0 | 0 | 120485 |

| | Overall | Test Set |
|---|---|---|
| Accuracy | 87.067 | 87.158 |
| Precision Typha | 78.734 | 100.000 |
| Recall Typha | 81.574 | 89.667 |
| F1-Score Typha | 80.129 | 94.552 |

Table A.26: Results of the test with the neural network 6-class 4-bands blob split with cleaning, second case. On the left is presented the confusion matrix, on the right the metrics computed.

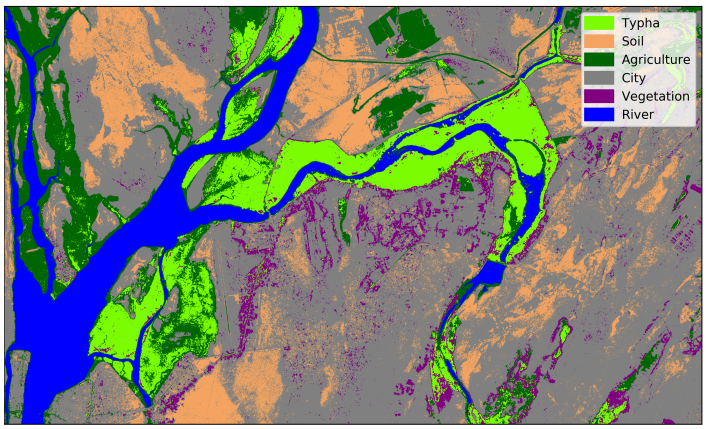



Figure A.52: Result of the classification.

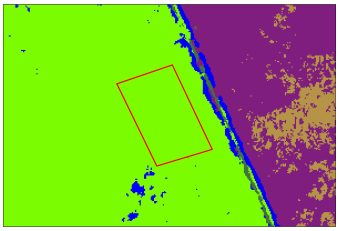


Figure A.53: Detail of the Typha blob that belongs to the test set.

### NN6c4bB2

| | | Predicted label | | | | | |
|---|---|---|---|---|---|---|---|
| | | Typha | Soil | Agriculture | City | Vegetation | River |
| True Label | Typha | 1281 | 0 | 747 | 21 | 2751 | 0 |
| | Soil | 0 | 9 | 0 | 14246 | 0 | 0 |
| | Agriculture | 0 | 0 | 3291 | 0 | 0 | 0 |
| | City | 164 | 13062 | 836 | 49568 | 273 | 172 |
| | Vegetation | 137 | 4 | 0 | 3862 | 5180 | 0 |
| | River | 0 | 0 | 0 | 0 | 0 | 120485 |

| | Overall | Test Set |
|---|---|---|
| Accuracy | 85.070 | 83.213 |
| Precision Typha | 57.796 | 26.687 |
| Recall Typha | 85.647 | 80.973 |
| F1-Score Typha | 69.017 | 40.144 |

*Table A.27: Results of the test with the neural network 6-class 4-bands blob split with cleaning, third case. On the left is presented the confusion matrix, on the right the metrics computed.*

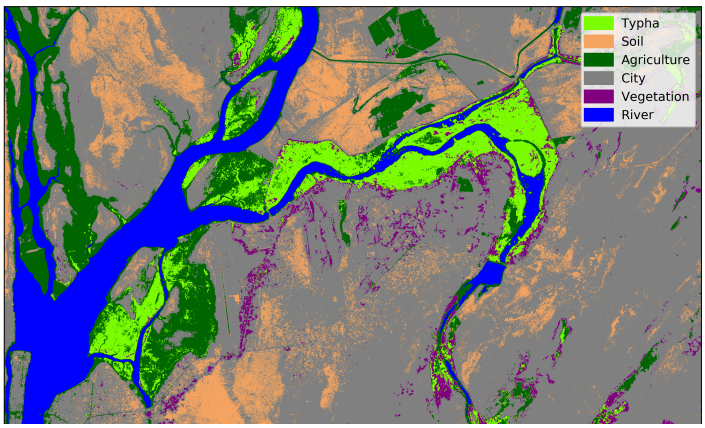



*Figure A.54: Result of the classification.*

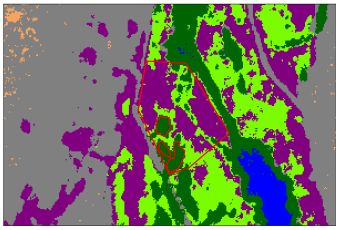


*Figure A.55: Detail of the Typha blob that belongs to the test set.*

## A.10 NN 6-class 4-bands Random Split

In this section the results of the Neural Network model, with a 6-class 4-bands random split classification with cleaning is presented. The three different experiments represent the three different combination of Typha blobs selected for training and test set.

### NN6c4bR0

| | | Predicted label | | | | | |
|---|---|---|---|---|---|---|---|
| | | Typha | Soil | Agriculture | City | Vegetation | River |
| True Label | Typha | 2745 | 0 | 184 | 0 | 445 | 0 |
| | Soil | 0 | 25277 | 0 | 4676 | 1 | 0 |
| | Agriculture | 19 | 0 | 26382 | 735 | 0 | 4 |
| | City | 127 | 5689 | 575 | 52685 | 599 | 33 |
| | Vegetation | 539 | 4 | 0 | 1018 | 4680 | 0 |
| | River | 0 | 0 | 24 | 0 | 0 | 50910 |

| | Overall | Test Set |
|---|---|---|
| Accuracy | 91.761 | 91.727 |
| Precision Typha | 80.821 | 81.357 |
| Recall Typha | 81.235 | 80.029 |
| F1-Score Typha | 81.028 | 80.688 |

*Table A.28: Results of the test with the neural network 6-class 4-bands random split with cleaning, first case. On the left is presented the confusion matrix, on the right the metrics computed.*

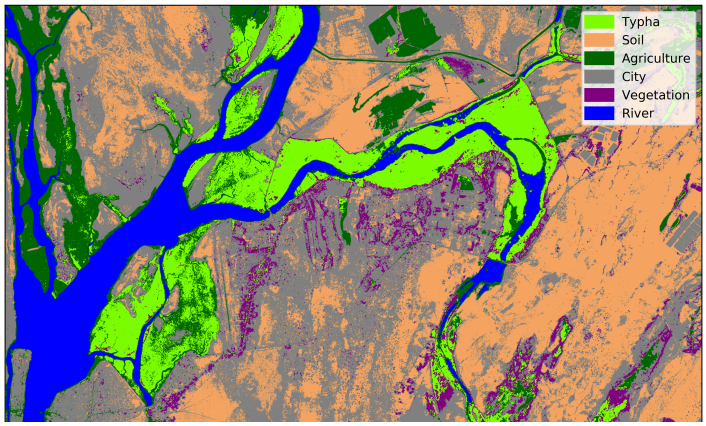



Figure A.56: Result of the classification.

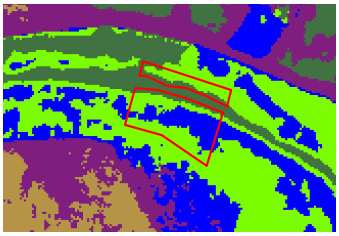


Figure A.57: Detail one of the Typha blobs.

### NN6c4bR1

| | | Predicted label | | | | | |
|---|---|---|---|---|---|---|---|
| | | Typha | Soil | Agriculture | City | Vegetation | River |
| True Label | Typha | 2661 | 0 | 232 | 0 | 631 | 0 |
| | Soil | 0 | 22805 | 0 | 7254 | 0 | 0 |
| | Agriculture | 5 | 4 | 26530 | 736 | 0 | 11 |
| | City | 107 | 7211 | 573 | 50601 | 715 | 46 |
| | Vegetation | 410 | 4 | 0 | 884 | 4742 | 0 |
| | River | 0 | 0 | 26 | 5 | 0 | 51158 |

| | Overall | Test Set |
|---|---|---|
| Accuracy | 89.375 | 89.369 |
| Precision Typha | 75.050 | 75.511 |
| Recall Typha | 82.371 | 83.600 |
| F1-Score Typha | 78.541 | 79.350 |

Table A.29: Results of the test with the neural network 6-class 4-bands random split with cleaning, second case. On the left is presented the confusion matrix, on the right the metrics computed.

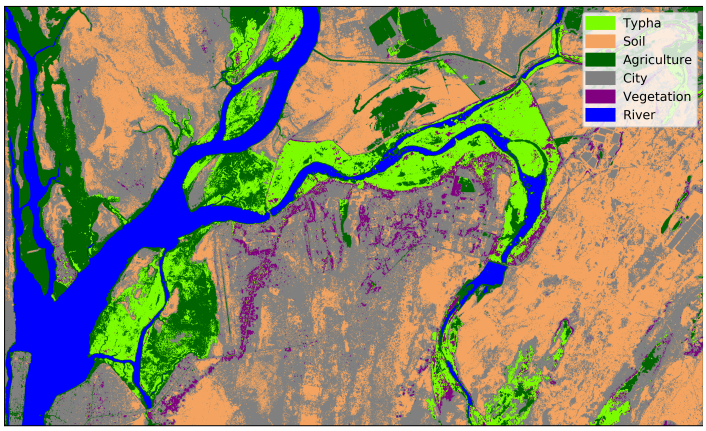



Figure A.58: Result of the classification.

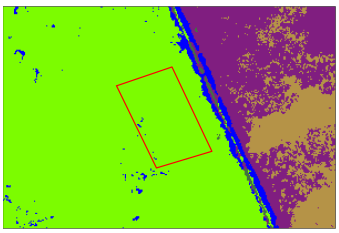


Figure A.59: Detail one of the Typha blobs..

### NN6c4bR2

| | | Predicted label | | | | | |
|---|---|---|---|---|---|---|---|
| | | Typha | Soil | Agriculture | City | Vegetation | River |
| True Label | Typha | 2442 | 0 | 329 | 0 | 674 | 0 |
| | Soil | 0 | 22240 | 0 | 7763 | 17 | 0 |
| | Agriculture | 1 | 2 | 26344 | 937 | 0 | 8 |
| | City | 87 | 7953 | 584 | 50167 | 678 | 48 |
| | Vegetation | 437 | 33 | 1 | 887 | 4828 | 0 |
| | River | 0 | 0 | 65 | 0 | 0 | 50826 |

| | Overall | Test Set |
|---|---|---|
| Accuracy | 88.487 | 88.439 |
| Precision Typha | 71.030 | 70.885 |
| Recall Typha | 82.959 | 82.305 |
| F1-Score Typha | 76.532 | 76.170 |

*Table A.30: Results of the test with the neural network 6-class 4-bands random split with cleaning, third case. On the left is presented the confusion matrix, on the right the metrics computed.*

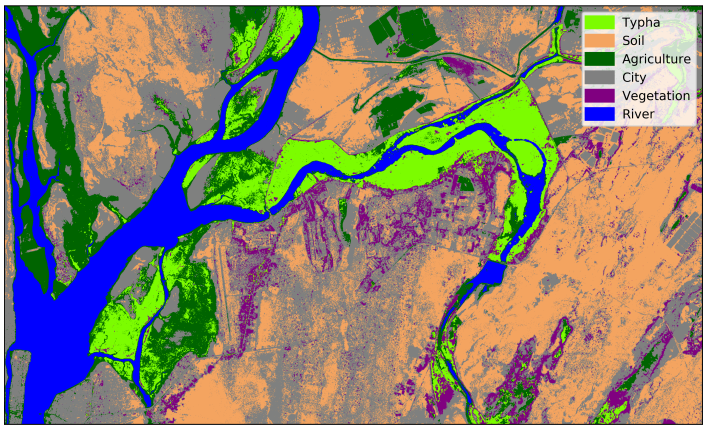



*Figure A.60: Result of the classification.*

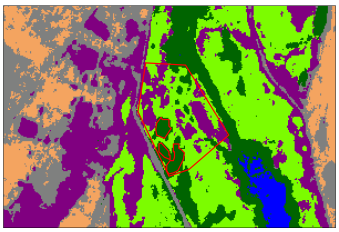


*Figure A.61: Detail one of the Typha blobs..*

## A.11 NN 6-class 6-bands Blob Split

In this section the results of the Neural Network model, with a 6-class 4-bands blob split classification with cleaning is presented. The three different experiments represent the three different combination of Typha blobs selected for training and test set.

### NN6c6bB0

| | | Predicted label | | | | | |
|---|---|---|---|---|---|---|---|
| | | Typha | Soil | Agriculture | City | Vegetation | River |
| True Label | Typha | 585 | 0 | 188 | 0 | 433 | 0 |
| | Soil | 0 | 1710 | 0 | 16575 | 0 | 0 |
| | Agriculture | 10 | 0 | 42180 | 1847 | 5 | 2865 |
| | City | 246 | 6864 | 315 | 54256 | 900 | 0 |
| | Vegetation | 148 | 12 | 0 | 194 | 3596 | 0 |
| | River | 0 | 0 | 0 | 0 | 0 | 24981 |

| | Overall | Test Set |
|---|---|---|
| Accuracy | 88.198 | 80.621 |
| Precision Typha | 73.011 | 48.507 |
| Recall Typha | 80.252 | 59.151 |
| F1-Score Typha | 76.461 | 53.303 |

*Table A.31: Results of the test with the neural network 6-class 6-bands blob split with cleaning, first case. On the left is presented the confusion matrix, on the right the metrics computed.*

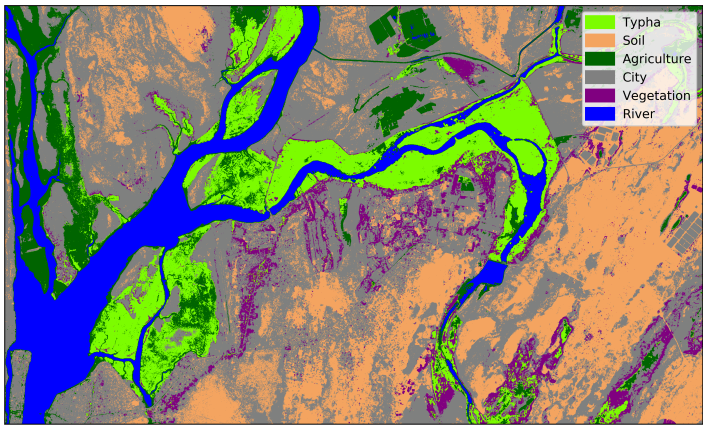



Figure A.62: Result of the classification.

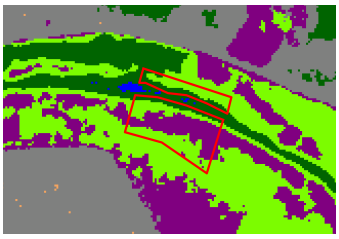


Figure A.63: Detail of the Typha blob that belongs to the test set.

### NN6c6bB1

| | | Predicted label | | | | | |
|---|---|---|---|---|---|---|---|
| | | Typha | Soil | Agriculture | City | Vegetation | River |
| True Label | Typha | 4390 | 0 | 0 | 0 | 1 | 0 |
| | Soil | 0 | 25 | 0 | 14229 | 1 | 0 |
| | Agriculture | 0 | 0 | 3291 | 0 | 0 | 0 |
| | City | 96 | 11816 | 733 | 47228 | 720 | 93 |
| | Vegetation | 458 | 1 | 0 | 62 | 2946 | 0 |
| | River | 0 | 0 | 0 | 0 | 0 | 120485 |

| | Overall | Test Set |
|---|---|---|
| Accuracy | 85.083 | 86.344 |
| Precision Typha | 81.196 | 99.977 |
| Recall Typha | 80.646 | 88.794 |
| F1-Score Typha | 80.920 | 94.055 |

Table A.32: Results of the test with the neural network 6-class 6-bands blob split with cleaning, second case. On the left is presented the confusion matrix, on the right the metrics computed.

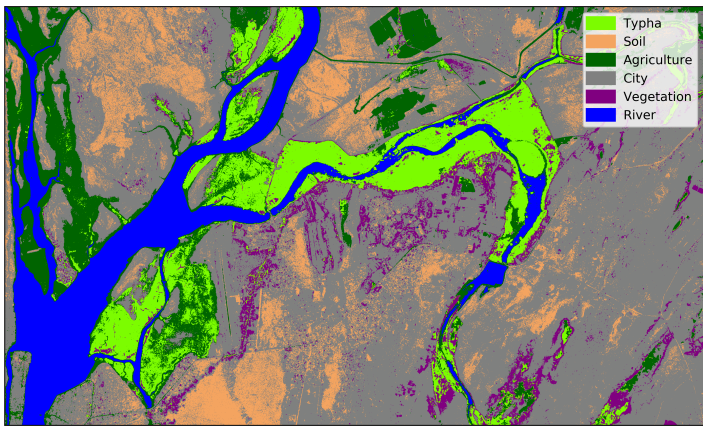



Figure A.64: Result of the classification.

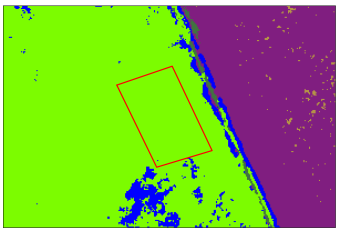


Figure A.65: Detail of the Typha blob that belongs to the test set.

### NN6c6bB2

| | | Predicted label | | | | | |
|---|---|---|---|---|---|---|---|
| | | Typha | Soil | Agriculture | City | Vegetation | River |
| True Label | Typha | 1045 | 0 | 1034 | 0 | 2721 | 0 |
| | Soil | 0 | 14131 | 0 | 0 | 124 | 0 |
| | Agriculture | 0 | 0 | 3291 | 0 | 0 | 0 |
| | City | 52 | 12234 | 691 | 50370 | 467 | 261 |
| | Vegetation | 176 | 867 | 59 | 2589 | 5492 | 0 |
| | River | 0 | 0 | 0 | 0 | 0 | 120485 |

| | Overall | Test Set |
|---|---|---|
| Accuracy | 84.026 | 90.155 |
| Precision Typha | 54.756 | 21.771 |
| Recall Typha | 86.415 | 82.090 |
| F1-Score Typha | 67.036 | 34.415 |

*Table A.33: Results of the test with the neural network 6-class 6-bands blob split with cleaning, third case. On the left is presented the confusion matrix, on the right the metrics computed.*

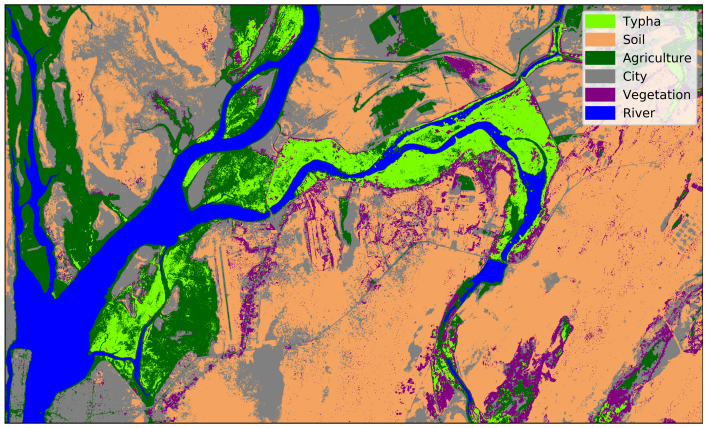



*Figure A.66: Result of the classification.*

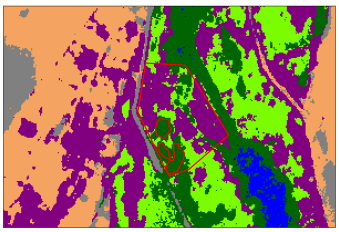


*Figure A.67: Detail of the Typha blob that belongs to the test set.*

## A.12 NN 6-class 6-bands Random Split

In this section the results of the Neural Network model, with a 6-class 4-bands blob split classification with cleaning is presented. The three different experiments represent the three different combination of Typha blobs selected for training and test set.

### NN6c6bR0

| | | Predicted label | | | | | |
|---|---|---|---|---|---|---|---|
| | | Typha | Soil | Agriculture | City | Vegetation | River |
| True Label | Typha | 2607 | 0 | 148 | 0 | 681 | 0 |
| | Soil | 0 | 17297 | 0 | 13040 | 0 | 0 |
| | Agriculture | 49 | 0 | 26305 | 775 | 0 | 3 |
| | City | 130 | 8565 | 631 | 49470 | 615 | 84 |
| | Vegetation | 488 | 3 | 2 | 975 | 4826 | 0 |
| | River | 0 | 0 | 19 | 0 | 0 | 50638 |

| | Overall | Test Set |
|---|---|---|
| Accuracy | 85.276 | 85.223 |
| Precision Typha | 75.743 | 75.873 |
| Recall Typha | 79.106 | 79.627 |
| F1-Score Typha | 77.388 | 77.705 |

*Table A.34: Results of the test with the neural network 6-class 6-bands random split with cleaning, first case. On the left is presented the confusion matrix, on the right the metrics computed.*

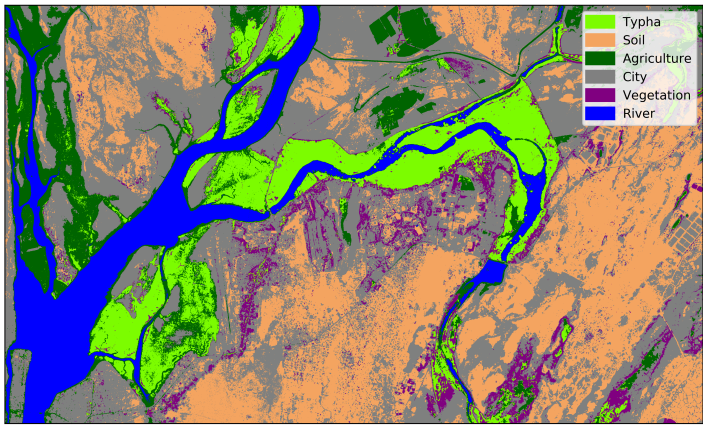



Figure A.68: Result of the classification.

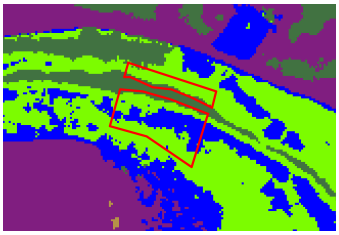


Figure A.69: Detail one of the Typha blobs.

### NN6c6bR1

| | | Predicted label | | | | | |
|---|---|---|---|---|---|---|---|
| | | Typha | Soil | Agriculture | City | Vegetation | River |
| True Label | Typha | 3179 | 0 | 192 | 9 | 0 | 0 |
| | Soil | 0 | 0 | 0 | 30087 | 0 | 0 |
| | Agriculture | 17 | 0 | 26520 | 741 | 0 | 1 |
| | City | 160 | 0 | 552 | 58463 | 0 | 52 |
| | Vegetation | 2754 | 0 | 0 | 3434 | 0 | 0 |
| | River | 0 | 0 | 16 | 0 | 0 | 51174 |

| | Overall | Test Set |
|---|---|---|
| Accuracy | 78.631 | 78.565 |
| Precision Typha | 94.123 | 94.053 |
| Recall Typha | 52.346 | 52.029 |
| F1-Score Typha | 67.276 | 66.997 |

Table A.35: Results of the test with the neural network 6-class 6-bands random split with cleaning, second case. On the left is presented the confusion matrix, on the right the metrics computed.

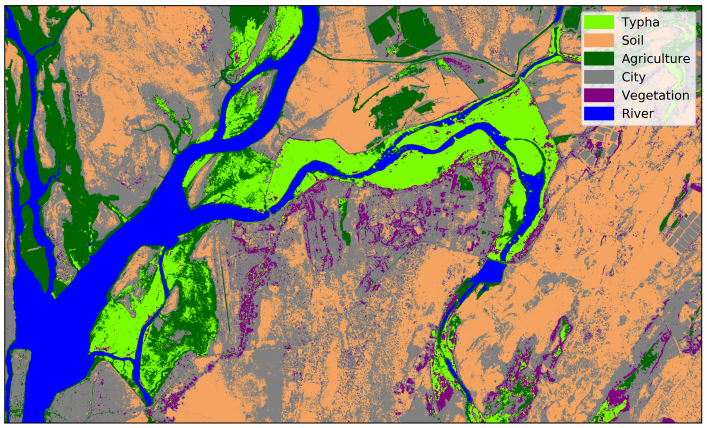



Figure A.70: Result of the classification.

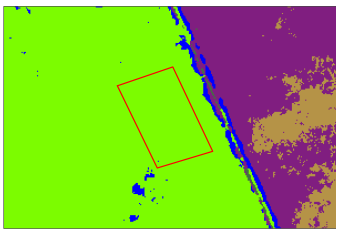


Figure A.71: Detail one of the Typha blobs.

### NN6c6bB2

| | | Predicted label | | | | | |
|---|---|---|---|---|---|---|---|
| | | Typha | Soil | Agriculture | City | Vegetation | River |
| True Label | Typha | 2440 | 0 | 289 | 0 | 746 | 0 |
| | Soil | 0 | 17645 | 0 | 12331 | 6 | 0 |
| | Agriculture | 0 | 3 | 26376 | 836 | 37 | 14 |
| | City | 62 | 6108 | 574 | 51807 | 787 | 112 |
| | Vegetation | 534 | 172 | 7 | 791 | 4608 | 0 |
| | River | 0 | 0 | 80 | 0 | 0 | 50986 |

| | Overall | Test Set |
|---|---|---|
| Accuracy | 86.813 | 86.756 |
| Precision Typha | 70.963 | 70.216 |
| Recall Typha | 80.652 | 80.369 |
| F1-Score Typha | 75.498 | 74.950 |

*Table A.36: Results of the test with the neural network 6-class 6-bands random split with cleaning, third case. On the left is presented the confusion matrix, on the right the metrics computed.*

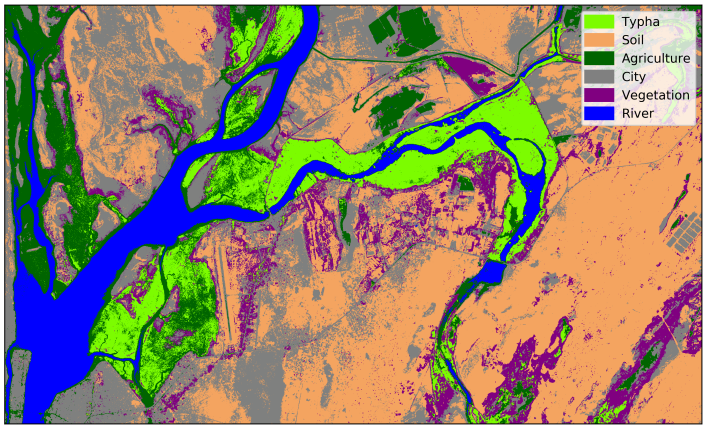



*Figure A.72: Result of the classification.*

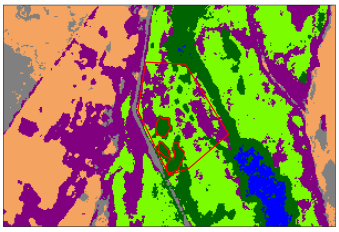


*Figure A.73: Detail one of the Typha blobs.*

## A.13 NN 6-class 7-bands Blob Split

In this section the results of the Neural Network model, with a 6-class 4-bands blob split classification with cleaning is presented. The three different experiments represent the three different combination of Typha blobs selected for training and test set.

### NN6c7bB0

| | | Predicted label | | | | | |
|---|---|---|---|---|---|---|---|
| | | Typha | Soil | Agriculture | City | Vegetation | River |
| True Label | Typha | 1166 | 0 | 0 | 0 | 0 | 40 |
| | Soil | 0 | 5135 | 0 | 13150 | 0 | 0 |
| | Agriculture | 254 | 26 | 191 | 1734 | 44702 | 0 |
| | City | 30 | 3035 | 265 | 58007 | 1242 | 2 |
| | Vegetation | 49 | 82 | 112 | 640 | 3067 | 0 |
| | River | 0 | 0 | 0 | 0 | 0 | 24981 |

| | Overall | Test Set |
|---|---|---|
| Accuracy | 76.260 | 58.607 |
| Precision Typha | 97.528 | 96.683 |
| Recall Typha | 95.498 | 77.785 |
| F1-Score Typha | 96.502 | 86.211 |

*Table A.37: Results of the test with the neural network 6-class 7-bands blob split with cleaning, first case. On the left is presented the confusion matrix, on the right the metrics computed.*

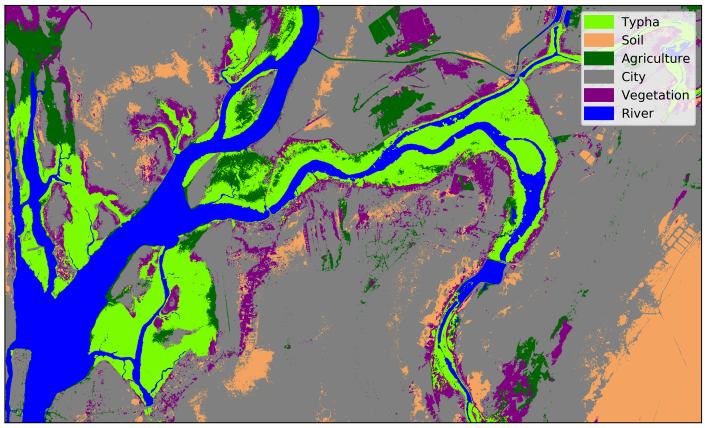



Figure A.74: Result of the classification.

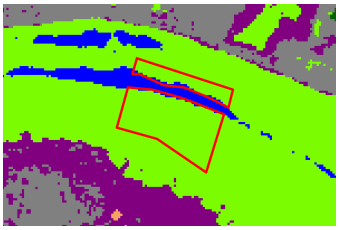


Figure A.75: Detail of the Typha blob that belongs to the test set.

### NN6c7bB1

| | | Predicted label | | | | | |
|---|---|---|---|---|---|---|---|
| | | Typha | Soil | Agriculture | City | Vegetation | River |
| True Label | Typha | 0 | 0 | 104 | 1607 | 2680 | 0 |
| | Soil | 0 | 0 | 13477 | 778 | 0 | 0 |
| | Agriculture | 0 | 0 | 3291 | 0 | 0 | 0 |
| | City | 24 | 3975 | 282 | 55775 | 630 | 0 |
| | Vegetation | 0 | 2 | 0 | 290 | 3175 | 0 |
| | River | 0 | 0 | 0 | 0 | 0 | 120485 |

| | Overall | Test Set |
|---|---|---|
| Accuracy | 84.494 | 88.455 |
| Precision Typha | 56.622 | 0.000 |
| Recall Typha | 98.002 | 0.000 |
| F1-Score Typha | 71.775 | nan |

Table A.38: Results of the test with the neural network 6-class 4-bands blob split with cleaning, second case. On the left is presented the confusion matrix, on the right the metrics computed.

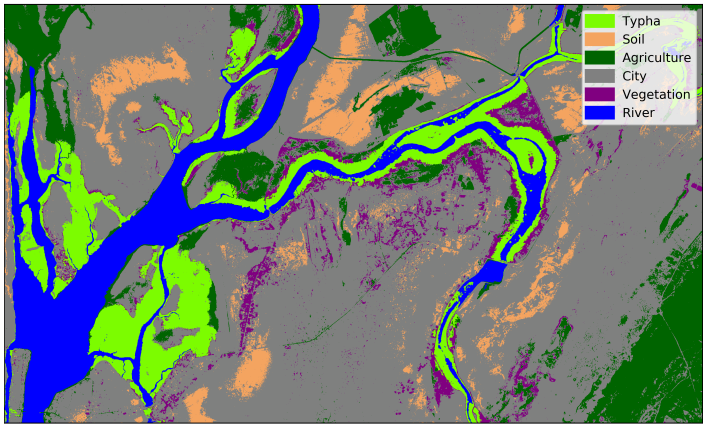



Figure A.76: Result of the classification.

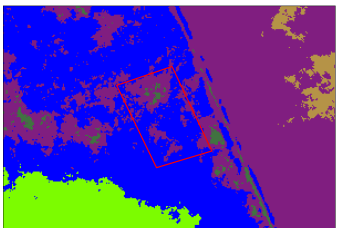


Figure A.77: Detail of the Typha blob that belongs to the test set.

### NN6c7bB2

| | | Predicted label | | | | | |
|---|---|---|---|---|---|---|---|
| | | Typha | Soil | Agriculture | City | Vegetation | River |
| True Label | Typha | 4354 | 0 | 0 | 40 | 406 | 0 |
| | Soil | 0 | 0 | 14067 | 188 | 0 | 0 |
| | Agriculture | 0 | 0 | 3291 | 0 | 0 | 0 |
| | City | 24 | 3607 | 532 | 59192 | 593 | 127 |
| | Vegetation | 125 | 39 | 9 | 5101 | 3909 | 0 |
| | River | 0 | 0 | 10441 | 0 | 0 | 110044 |

| | Overall | Test Set |
|---|---|---|
| Accuracy | 82.520 | 83.665 |
| Precision Typha | 94.729 | 90.708 |
| Recall Typha | 96.257 | 96.691 |
| F1-Score Typha | 95.487 | 93.604 |

*Table A.39: Results of the test with the neural network 6-class 7-bands blob split with cleaning, second case. On the left is presented the confusion matrix, on the right the metrics computed.*

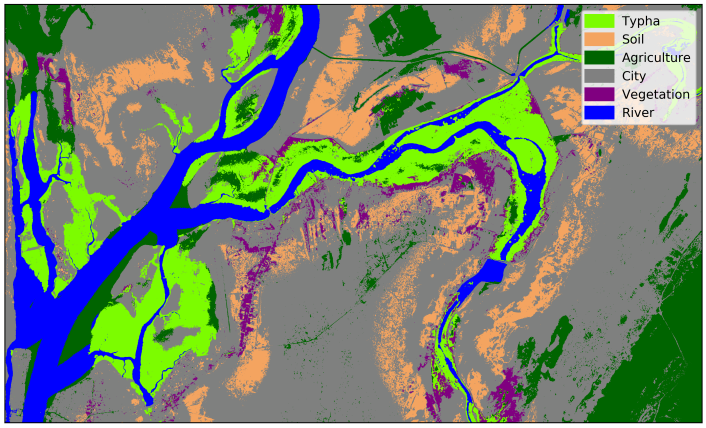



*Figure A.78: Result of the classification.*

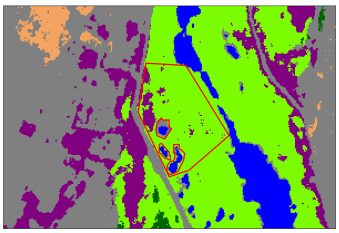


*Figure A.79: Detail of the Typha blob that belongs to the test set.*

## A.14 NN 6-class 7-bands Random Split

In this section the results of the Neural Network model, with a 6-class 4-bands blob split classification with cleaning is presented. The three different experiments represent the three different combination of Typha blobs selected for training and test set.

### NN6c7bR0

| | | Predicted label | | | | | |
|---|---|---|---|---|---|---|---|
| | | Typha | Soil | Agriculture | City | Vegetation | River |
| True Label | Typha | 3398 | 0 | 6 | 19 | 42 | 0 |
| | Soil | 0 | 28679 | 1 | 1351 | 0 | 0 |
| | Agriculture | 0 | 60 | 26885 | 346 | 0 | 0 |
| | City | 99 | 3926 | 412 | 54424 | 529 | 35 |
| | Vegetation | 44 | 1 | 8 | 847 | 5322 | 0 |
| | River | 0 | 0 | 0 | 5 | 0 | 50912 |

| | Overall | Test Set |
|---|---|---|
| Accuracy | 95.618 | 95.641 |
| Precision Typha | 98.326 | 98.066 |
| Recall Typha | 96.171 | 95.962 |
| F1-Score Typha | 97.237 | 97.003 |

*Table A.40: Results of the test with the neural network 6-class 7-bands random split with cleaning, first case. On the left is presented the confusion matrix, on the right the metrics computed.*

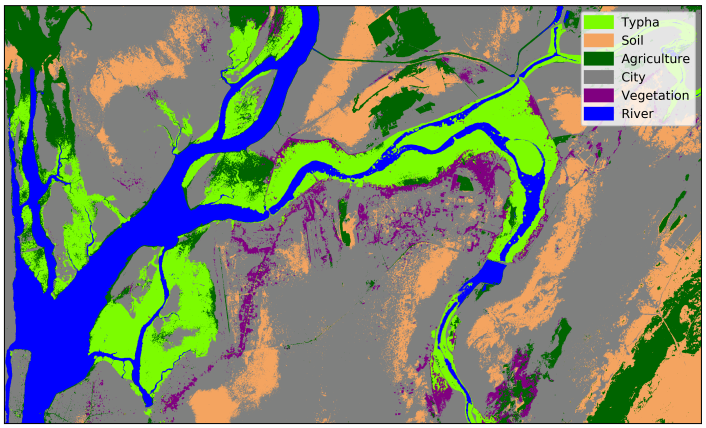



Figure A.80: Result of the classification.

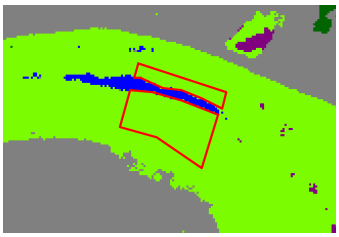


Figure A.81: Detail one of the Typha blobs.

### NN6c7bR1

| | | Predicted label | | | | | |
|---|---|---|---|---|---|---|---|
| | | Typha | Soil | Agriculture | City | Vegetation | River |
| True Label | Typha | 3444 | 0 | 0 | 8 | 17 | 6 |
| | Soil | 0 | 28211 | 0 | 1729 | 0 | 0 |
| | Agriculture | 17 | 2 | 26943 | 307 | 0 | 0 |
| | City | 63 | 3770 | 249 | 54822 | 456 | 27 |
| | Vegetation | 35 | 4 | 0 | 586 | 5435 | 0 |
| | River | 0 | 0 | 0 | 0 | 0 | 51220 |

| | Overall | Test Set |
|---|---|---|
| Accuracy | 95.899 | 95.897 |
| Precision Typha | 99.154 | 99.108 |
| Recall Typha | 96.753 | 96.769 |
| F1-Score Typha | 97.938 | 97.924 |

Table A.41: Results of the test with the neural network 6-class 7-bands random split with cleaning, second case. On the left is presented the confusion matrix, on the right the metrics computed.

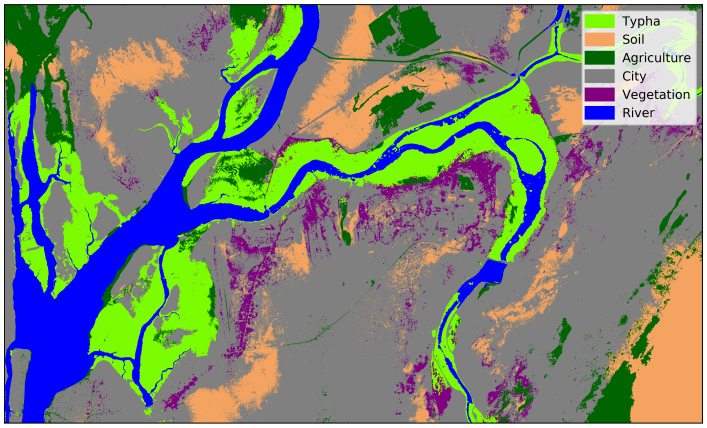



Figure A.82: Result of the classification.

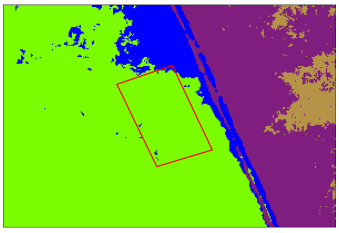


Figure A.83: Detail one of the Typha blobs.

### NN6c7bR2

| | | Predicted label | | | | | |
|---|---|---|---|---|---|---|---|
| | | Typha | Soil | Agriculture | City | Vegetation | River |
| True Label | Typha | 3480 | 0 | 0 | 5 | 31 | 0 |
| | Soil | 0 | 28575 | 1 | 1443 | 0 | 0 |
| | Agriculture | 6 | 0 | 26723 | 504 | 2 | 0 |
| | City | 97 | 4745 | 238 | 53780 | 581 | 34 |
| | Vegetation | 55 | 13 | 2 | 703 | 5365 | 0 |
| | River | 0 | 0 | 0 | 0 | 0 | 50968 |

| | Overall | Test Set |
|---|---|---|
| Accuracy | 95.161 | 95.230 |
| Precision Typha | 98.904 | 98.976 |
| Recall Typha | 95.523 | 95.657 |
| F1-Score Typha | 97.184 | 97.288 |

*Table A.42: Results of the test with the neural network 6-class 7-bands random split with cleaning, third case. On the left is presented the confusion matrix, on the right the metrics computed.*

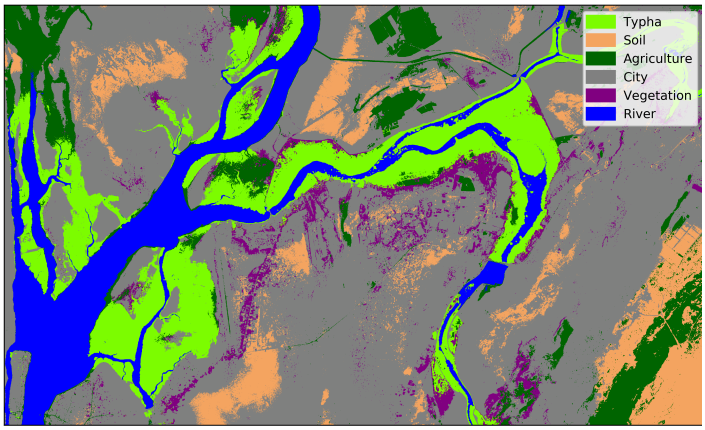



*Figure A.84: Result of the classification.*

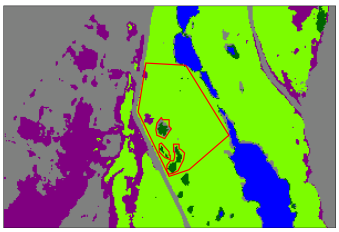


*Figure A.85: Detail one of the Typha blobs.*

## A.15 CNN

### CNN6c6bB0

| | | Predicted label | | | | | |
|---|---|---|---|---|---|---|---|
| | | Typha | Soil | Agriculture | City | Vegetation | River |
| True Label | Typha | 438 | 44 | 90 | 0 | 481 | 153 |
| | Soil | 0 | 2747 | 0 | 26811 | 0 | 0 |
| | Agriculture | 0 | 0 | 23041 | 0 | 0 | 0 |
| | City | 84 | 48 | 52 | 61782 | 165 | 126 |
| | Vegetation | 0 | 59 | 0 | 1737 | 7376 | 0 |
| | River | 0 | 0 | 0 | 0 | 0 | 24981 |

| | Overall | Test Set |
|---|---|---|
| Accuracy | 94.321 | 80.128 |
| Precision Typha | 92.613 | 36.318 |
| Recall Typha | 98.537 | 83.908 |
| F1-Score Typha | 95.483 | 50.694 |

*Table A.43: Result of the test with the convolutional neural network 6-class 6-bands blob split, first case. On the left is presented the confusion matrix, on the right the metrics computed.*

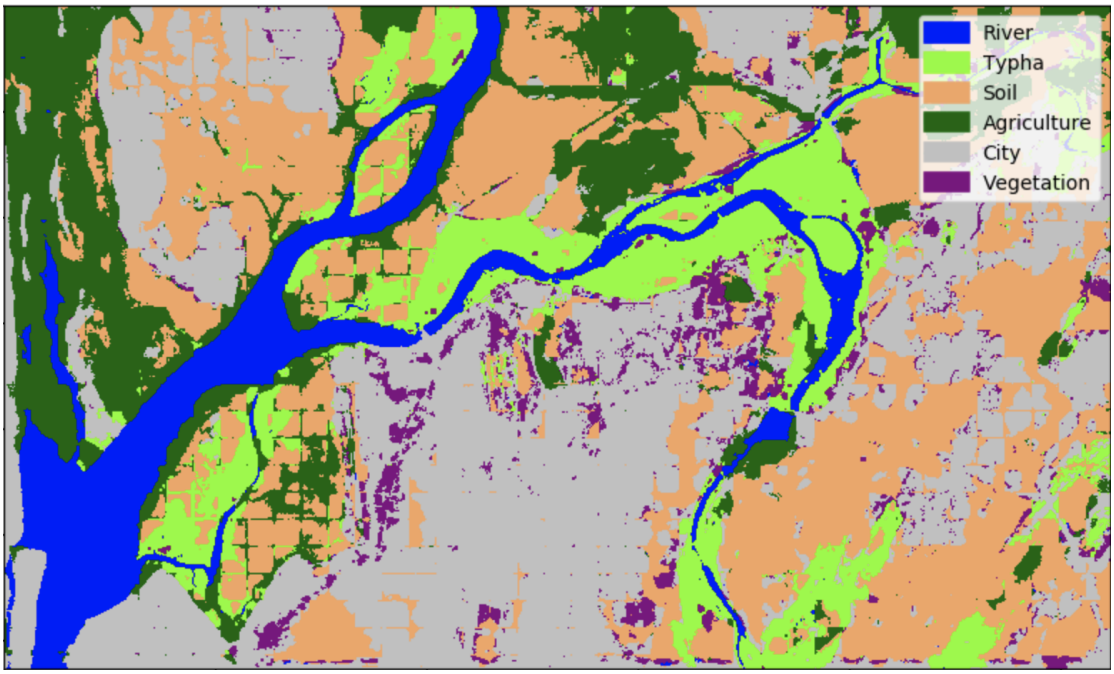



*Figure A.86: Result of the classification.*

### CNN6c6bB1

| | | Predicted label | | | | | |
|---|---|---|---|---|---|---|---|
| | | Typha | Soil | Agriculture | City | Vegetation | River |
| True Label | Typha | 4391 | 0 | 0 | 0 | 0 | 0 |
| | Soil | 0 | 5901 | 0 | 22880 | 0 | 0 |
| | Agriculture | 0 | 42 | 46861 | 4 | 0 | 0 |
| | City | 60 | 111 | 12 | 59974 | 337 | 0 |
| | Vegetation | 0 | 1 | 0 | 129 | 4303 | 0 |
| | River | 0 | 0 | 0 | 0 | 0 | 7211 |

| | Overall | Test Set |
|---|---|---|
| Accuracy | 95.392 | 84.512 |
| Precision Typha | 99.981 | 100.000 |
| Recall Typha | 98.419 | 98.652 |
| F1-Score Typha | 99.194 | 99.321 |

*Table A.44: Result of the test with the convolutional neural network 6-class 6-bands blob split, second case. On the left is presented the confusion matrix, on the right the metrics computed.*

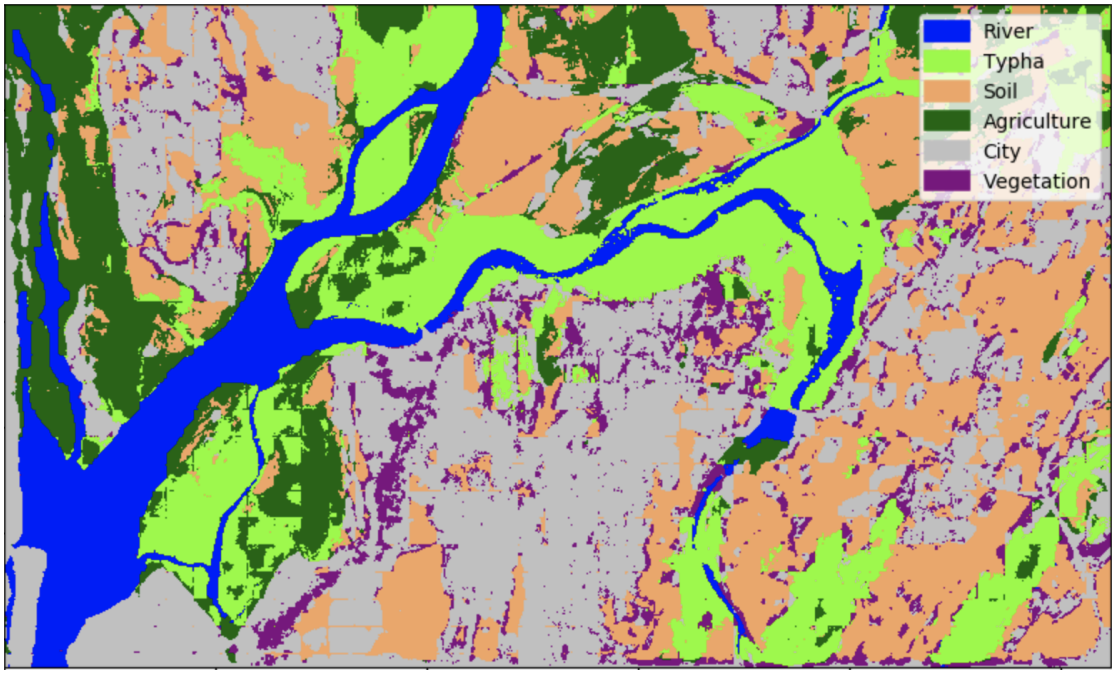



*Figure A.87: Result of the classification*

### CNN6c6bB2

| | | Predicted label | | | | | |
|---|---|---|---|---|---|---|---|
| | | Typha | Soil | Agriculture | City | Vegetation | River |
| True Label | Typha | 3625 | 0 | 0 | 105 | 1070 | 0 |
| | Soil | 0 | 5744 | 0 | 23814 | 0 | 0 |
| | Agriculture | 0 | 0 | 3291 | 0 | 0 | 0 |
| | City | 26 | 68 | 36 | 61608 | 80 | 1 |
| | Vegetation | 0 | 1 | 1 | 4544 | 6375 | 0 |
| | River | 0 | 0 | 0 | 0 | 0 | 24981 |

| Overall | Test Set | |
|---|---|---|
| Accuracy | 94.291 | 78.026 |
| Precision Typha | 88.679 | 75.521 |
| Recall Typha | 99.643 | 99.288 |
| F1-Score Typha | 93.842 | 85.789 |

*Table A.45: Result of the test with the convolutional neural network 6-class 6-bands blob split, third case. On the left is presented the confusion matrix, on the right the mestrics metrics.*

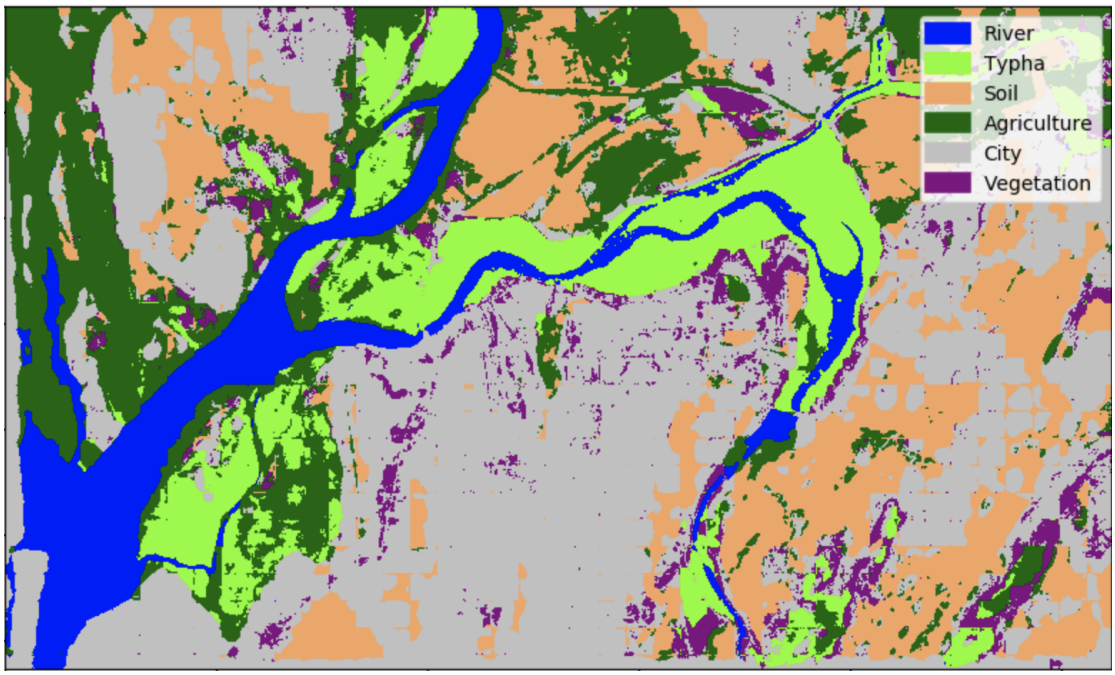



*Figure A.88: Result of the classification*

# Bibliography


[1] Bruno Gryseels et al. "Human schistosomiasis". In: *The Lancet* 368.9541 (2006), pp. 1106–1118.

[2] World Health Organization et al. "Prevention and control of schistosomiasis and soil-transmitted helminthiasis: report of a WHO expert committee". In: (2002).

[3] *Mapping Schistosomiasis Risk in the Saint Louis region, Senegal.* URL: `http://www.mastr-sls.polimi.it`.

[4] Jwan Al-Doski, Shattri B Mansorl, and Helmi Zulhaidi Mohd Shafri. "Image classification in remote sensing". In: *Department of Civil Engineering, Faculty of Engineering, University Putra, Malaysia* (2013).

[5] David G Lowe. "Distinctive image features from scale-invariant keypoints". In: *International journal of computer vision* 60.2 (2004), pp. 91–110.

[6] Navneet Dalal and Bill Triggs. "Histograms of oriented gradients for human detection". In: *international Conference on computer vision & Pattern Recognition (CVPR'05)*. Vol. 1. IEEE Computer Society. 2005, pp. 886–893.

[7] Gong Cheng, Junwei Han, and Xiaoqiang Lu. "Remote sensing image scene classification: Benchmark and state of the art". In: *Proceedings of the IEEE* 105.10 (2017), pp. 1865–1883.

[8] Ian T Jolliffe and Jorge Cadima. "Principal component analysis: a review and recent developments". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 374.2065 (2016), p. 20150202.

[9] J. Ross Quinlan. "Induction of decision trees". In: *Machine learning* 1.1 (1986), pp. 81–106.

[10] Corinna Cortes and Vladimir Vapnik. "Support-vector networks". In: *Machine learning* 20.3 (1995), pp. 273–297.

[11] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[12] Dengsheng Lu and Qihao Weng. "A survey of image classification methods and techniques for improving classification performance". In: *International journal of Remote sensing* 28.5 (2007), pp. 823–870.

[13] Brandt CK Tso and Paul M Mather. "Classification of multisource remote sensing imagery using a genetic algorithm and Markov random fields". In: *IEEE Transactions on Geoscience and Remote Sensing* 37.3 (1999), pp. 1255–1260.

[14] Floyd F Sabins. *Remote sensing: principles and applications*. Waveland Press, 2007.

[15] Bangalore D Nagesh Kumar IISc. "SPATIAL AND SPECTRAL RESOLUTIONS". In: ().

[16] *Know Basics of Remote Sensing Quickly and Become Expert*. 2015. URL: `https://grindgis.com/what-is-remote-sensing/know-basics-of-remote-sensing`.

[17] Priyanka Sharma and Urvashi Mutreja. "Analysis of satellite images using artificial neural network". In: *Int. J. Soft Comput. Eng* 2 (2013), pp. 276–278.

[18] *Tools for designing neural network architecture*. URL: `http://alexlenail.me/NN-SVG/index.html`.

[19] David Kriesel. "A brief introduction on neural networks". In: (2007).

[20] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[21] Kaiming He et al. "Mask r-cnn". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.

[22] Ali Sharif Razavian et al. "CNN features off-the-shelf: an astounding baseline for recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2014, pp. 806–813.

[23] Dan Claudiu Ciresan et al. "Flexible, high performance convolutional neural networks for image classification". In: *Twenty-Second International Joint Conference on Artificial Intelligence*. 2011.

[24] Kaiming He et al. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1026–1034.

[25] *CS231n Convolutional Neural Networks for Visual Recognition*. URL: `https://github.com/cs231n/cs231n.github.io`.

[26] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. "Fast and accurate deep network learning by exponential linear units (elus)". In: *arXiv preprint arXiv:1511.07289* (2015).

[27] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *arXiv:1502.03167* (2015).

[28] Miao Li et al. "A review of remote sensing image classification techniques: The role of spatio-contextual information". In: *European Journal of Remote Sensing* 47.1 (2014), pp. 389–411.

[29] Peter M Atkinson and David K Naser. "A Geostatistically Weighted k-NN Classifier for Remotely Sensed Imagery." In: *Geographical analysis* 42.2 (2010), pp. 204–225.

[30] Rongrong Ji et al. "Spectral-spatial constraint hyperspectral image classification". In: *IEEE Transactions on Geoscience and Remote Sensing* 52.3 (2013), pp. 1811–1824.

[31] Yushi Chen et al. "Deep learning-based classification of hyperspectral data". In: *IEEE Journal of Selected topics in applied earth observations and remote sensing* 7.6 (2014), pp. 2094–2107.

[32] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[33] Vladimir Iglovikov and Alexey Shvets. "Ternausnet: U-net with vgg11 encoder pre-trained on imagenet for image segmentation". In: *arXiv preprint arXiv:1801.05746* (2018).

[34] *Global Invasive Species Database*. URL: `http://www.iucngisd.org/gisd/species.php?sc=895r`.

[35] Christin B Frieswyk and Joy B Zedler. "Do seed banks confer resilience to coastal wetlands invaded by Typha× glauca?" In: *Botany* 84.12 (2006), pp. 1882–1893.

[36] Shane C Lishawa et al. "Biomass harvest of invasive Typha promotes plant diversity in a Great Lakes coastal wetland". In: *Restoration ecology* 23.3 (2015), pp. 228–237.

[37] Lorenzo Mari et al. "Big-data-driven modeling unveils country-wide drivers of endemic schistosomiasis". In: *Scientific reports* 7.1 (2017), p. 489.

[38] OpenStreetMap contributors. *Planet dump retrieved from https://planet.osm.org*. `https://www.openstreetmap.org`. 2017.

[39] *PLANET IMAGERY PRODUCT SPECIFICATIONS*. 2019.

[40] Yanli Wu et al. "Landslide susceptibility assessment using frequency ratio, statistical index and certainty factor models for the Gangu County, China". In: *Arabian Journal of Geosciences* 9.2 (2016), p. 84.

[41] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[42] François Chollet et al. *Keras*. `https://keras.io`. 2015.

[43] Tiago Carneiro et al. "Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications". In: *IEEE Access* 6 (2018), pp. 61677–61685.

[44] Michael A Nielsen. *Neural networks and deep learning*. Vol. 25. Determination press San Francisco, CA, USA: 2015.